Coordenação em Sistemas Multi-Agente:

Aplicações na Gestão Universitária e Futebol Robótico

Luís Paulo Reis

Universidade do Porto
Faculdade de Engenharia
FEUP

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Junho de 2003

Coordenação em Sistemas Multi-Agente:

Aplicações na Gestão Universitária e Futebol Robótico

Luís Paulo Reis

Mestre em Engenharia Electrotécnica e de Computadores pela
Faculdade de Engenharia da Universidade do Porto

Dissertação submetida para a obtenção do grau de Doutor em
Engenharia Electrotécnica e de Computadores da Faculdade de
Engenharia da Universidade do Porto

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Rua Roberto Frias, s/n, 4200-465 Porto, Portugal

Junho de 2003

Dissertação realizada sob a orientação científica do

Professor Doutor Eugénio da Costa Oliveira

Professor Catedrático do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Resumo

Neste trabalho são estudadas as metodologias de coordenação em sistemas computacionais multi-agente e propostas metodologias adequadas à coordenação de agentes em dois domínios: i) resolução de problemas de escalonamento na gestão universitária; ii) execução de tarefas cooperativas por equipas de agentes com mobilidade espacial (construção de uma equipa para a liga de simulação do RoboCup).

São estudados os conceitos de Agente computacional e Sistema Multi-Agente e as metodologias de coordenação, negociação e cooperação entre agentes. Deste estudo resulta a proposta de algumas novas metodologias de alto-nível de coordenação de agentes tais como a coordenação estratégica, coordenação por percepção inteligente e coordenação por controlo parcialmente hierárquico. Apresenta-se também a extensão de outras metodologias de coordenação propostas previamente por outros investigadores tais como a definição e troca de papéis e coordenação por modelização mútua.

As estratégias de coordenação propostas são aplicadas a domínios multi-agente relevantes e generalizadas (com as devidas limitações) de forma a serem aplicadas a outros domínios socialmente úteis. Através de resultados experimentais no domínio do futebol robótico, obtidos em competições oficias e experiências controladas, é comprovado que a coordenação estratégica proposta no âmbito desta tese, permite realizar uma coordenação de alto-nível muito eficaz na realização de tarefas complexas através de equipas de agentes robóticos.

Palavras Chave: Coordenação, Agentes, Sistemas Multi-Agente, Cooperação, Negociação, Futebol Robótico, RoboCup, Escalonamento, Geração de Horários

Email: lpreis@fe.up.pt

Homepage: http://www.fe.up.pt/~lpreis

Abstract

This thesis presents a study about Multi-Agent systems coordination methodologies and proposes adequate coordination methodologies for two distinct domains: i) scheduling problems within university management, namely staff allocation and timetabling; ii) execution of cooperative tasks using teams of agents with spatial mobility (development of a team for RoboCup simulation league)

This work discusses the concepts of Computational Agent and Multi-Agent System, coordination, negotiation and cooperation methodologies for multi-agent societies. New high-level methodologies are proposed such as strategic coordination, intelligent perception coordination and partially hierarchical control. Also, extensions are proposed for known coordination methodologies such as role definition and exchange and mutual modeling coordination.

The coordination methodologies are applied to significant multi-agent scenarios and generalized (with appropriate limitations) to relevant socially useful multi-agent domains. Through experimental results, achieved in official multi-agent competitions and other controlled experiments, it is proved that strategic coordination enables a very efficient high-level coordination of multi-agent robotic teams performing complex tasks.

Keywords: Coordination, Agents, Multi-Agent Systems, Cooperation, Negotiation, Robotic Soccer, RoboCup, Scheduling, Timetabling

Email: lpreis@fe.up.pt

Homepage: http://www.fe.up.pt/~lpreis

Résumé

Ce travail étudie les méthodologies de coordination dans un Système informatique Multi-Agent et propose des méthodologies adéquates de coordination des agents chez deux environnements: i) résolution des problèmes de l'échelonnement dans la gestion universitaire; ii) exécution des activités de coopération en employant une équipe d'agents ayant la propriété de la mobilité (construction de une équipe pour la ligue de simulation du RoboCup – football robotique)

Ce travail étudie les concepts d'Agent et Système Multi-Agent, aussi bien que les méthodologies de coordination, négociation et coopération entre agents. A partir de cette étude on obtien une définition de nouvelles méthodologies de haut-niveau pour la coordination d'agents comme: la coordination stratégique, la coordination pour perception intelligente et coordination pour contrôle partialement hiérarchique. Nous proposons aussi d'autres méthodologies de coordination comme la définition et échange des activités et positions, et aussi la coordination par modélisation mutuelle.

Les stratégies de coordination proposées sont appliquées aux ambiants multi-agent, et généralisées (avec des limitations) de façon à pouvoir être appliquées à d'autres ambiants socialement importants. Les résultats expérimentaux dans le football robotique, obtenus avec des compétitions officielles et dans les expériences contrôlées, certifient qui la coordination stratégique (proposée dans cette thèse), permet de réaliser une coordination de haut-niveau très efficace pour la réalisation des activités complexes.

Mots Clé: Coordination; Agents; Systèmes Multi-Agent; Coopération; Négociation; Football Robotique; Échelonnement ; Génération des Horaires.

Email: lpreis@fe.up.pt

Homepage: http://www.fe.up.pt/~lpreis

Prefácio

O trabalho que se dá a conhecer nesta tese é a síntese de um trabalho de investigação mais abrangente que decorreu durante seis anos e envolveu diversas pessoas, entidades e instituições. É, para mim, um grande prazer escrever esta tese, não só pelo significado que ela tem por si mesma, mas também pelo facto de sintetizar esse trabalho de investigação mais amplo que obteve extensa divulgação nacional e internacional nos meios científicos apropriados.

Nos diversos projectos que conduziram à elaboração desta tese, pude trabalhar com investigadores de grande qualidade e que contribuíram de diversas formas para o resultado final desta tese. Neste prefácio vou tentar contextualizar no tempo e no espaço este trabalho, explicando resumidamente as contribuições dessas pessoas (a quem já tive o prazer de agradecer, pessoalmente ou na secção dos agradecimentos, pela sua contribuição) assim como a envolvente global em que se desenrolou este projecto.

Esta tese resulta essencialmente da definição e aplicação de metodologias de coordenação de agentes inteligentes em diversos domínios analisados em diferentes projectos de investigação. Dois projectos constituíram o núcleo central da aplicação prática dessas metodologias: o projecto *UNIPS – University Planning and Scheduling System* e o projecto *FC Portugal – Uma Equipa de Futebol Robótico para a Liga de Simulação.* Outros projectos estiveram ligados perifericamente a este tese, incluindo-se nestes o projecto *5dpo* – Equipas de futebol robótico para as ligas de robôs pequenos e médios (e o seu sucedâneo *Portus*), o projecto *D. Dinis* – um agente para o Micro-Rato (e o seu sucedâneo *Micro-Pessoa*), o projecto *TALK – Talking Agent Construction Kit.*

O projecto UNIPS resultou directamente do projecto inicial associado a esta tese de doutoramento que consistia na aplicação de metodologias da Inteligência Artificial ao problema da geração de horários em Universidades. Após a análise inicial do problema da geração de horários e da sua inserção no contexto global da gestão universitária, foi concluído que os Sistemas Multi-Agente constituíam uma metodologia promissora para efectuar a resolução global dos problemas de escalonamento associados à gestão universitária. Desta forma, e tendo em conta que não foi possível realizar o projecto nas condições previstas inicialmente (que previa a utilização de Programação Lógica com Restrições na geração de horários em cooperação com a University of Leeds), o projecto evoluiu para a utilização de metodologias de coordenação aplicadas a um conjunto de problemas mais abrangente encontrados no contexto da gestão universitária. Embora o projecto UNIPS não tenha contado com qualquer financiamento externo ou com uma equipa de investigação fixa, diversos investigadores colaboraram nele através da execução de trabalhos relacionados que eu orientei (no âmbito de monografias) ou co-orientei (no

âmbito de teses de Mestrado). Entre eles assumiram particular relevância os trabalhos realizados pelo Pedro Faria, Pedro Silva, Ricardo Dias, Paulo Teixeira e Manuel Chavarria no âmbito das suas teses e/ou monografias.

O projecto FC Portugal foi desenvolvido em conjunto com a Universidade de Aveiro. Inicialmente dois investigadores desta Universidade, Nuno Lau e Luís Seabra Lopes, participaram neste projecto. Após a fase inicial do projecto, unicamente o Nuno Lau permaneceu no projecto e contribuiu para o seu sucesso, sendo responsável por uma parcela importante da implementação realizada. Foi também responsável pela definição da estratégia de compressão e encriptação da informação comunicada, modelo de sincronização com o servidor, comportamentos de baixo nível de intercepção, drible e remate da bola, movimentação de jogadores no campo, comportamento individual dos jogadores e diversos outros tópicos relacionados com Sistemas Operativos, Redes de Computadores. A parte conceptual referente à abordagem de coordenação do FC Portugal, incluindo a definição formal do conceito de estratégia para uma competição, tácticas, formações e comportamentos, posicionamento estratégico baseado em situações, troca dinâmica de posições e papéis, comunicação avançada, percepção inteligente, gramática da linguagem para o treinador, etc., resultou essencialmente da investigação realizada no âmbito desta tese. No entanto, deve ser ressalvado que todos os tópicos descritos nesta tese relacionados com o FC Portugal foram objecto de discussão profunda com o Nuno Lau e que uma parcela muito importante da implementação dos mesmos foi realizada pelo mesmo Nuno Lau. Devido à conquista de dois campeonatos europeus e um campeonato mundial de futebol robótico simulado, o projecto FC Portugal teve grande visibilidade nacional e internacional. Felizmente, a esta visibilidade, esteve também associado o reconhecimento pela grande maioria dos pares da investigação realizada no âmbito do projecto. Deste reconhecimento resultou a minha eleição e posterior reeleição para o comité técnico da Liga de Simulação (responsável pela decisão do futuro científico do simulador e da competição associada) onde desde Setembro de 2001, desempenho funções.

O projecto *5dpo* – Equipa de Futebol Robótico para as Ligas de Robôs Pequenos e Robôs Médios é um projecto da Faculdade de Engenharia da Universidade do Porto. Diversos investigadores desta Faculdade a que se juntam alguns colaboradores externos estão envolvidos no projecto. A equipa inicial era composta por Paulo Costa, António Paulo Moreira, Armando Sousa, Paulo Marques, Pedro Costa e Susana Gaio. A minha participação no projecto iniciou-se unicamente em Dezembro de 2000, após a vitória do FC Portugal no mundial de futebol robótico disputado em Melbourne. O objectivo principal desta participação consistia em preparar a arquitectura de *software* do 5dpo de forma a possibilitar o futuro transporte, a médio prazo, das metodologias de IA utilizadas na liga de simulação, para as ligas robóticas e para as duas equipas do 5dpo (robôs pequenos e robôs médios). Embora, nestes últimos dois anos, o projecto tenha recuperado grande parte do dinamismo e sucesso que teve no seu início, conquistando diversas taças a

nível internacional, uma nota de ressalva importante é que a minha participação nesse incremento de qualidade foi meramente residual. A qualidade da investigação realizada pelo Paulo Costa, António Paulo Moreira e restantes elementos do projecto foi a grande responsável pelo dinamismo do projecto.

Mais do que concluir esta fase do trabalho, dá-me especial prazer ter a certeza que esta tese terá uma continuação efectiva e que este projecto está longe de acabar. O projecto originou já diversas teses de Mestrado, o início de três projectos de investigação científica financiados pela FCT, uma nova linha de investigação no LIACC NIAD&R – coordenação em Sistemas Multi-Agente e revitalizou uma outra linha de investigação – Robótica Inteligente e despertou ainda um enorme entusiasmo nos alunos da FEUP a todos os níveis. Ficam também objectivos de curto e longo prazo que deixam antever uma continuação efectiva da tese nos projectos e linhas de investigação referidos. A curto prazo desencadeou projectos com o objectivo de participar em todas as ligas do RoboCup 2004 que se irá realizar em Lisboa em Junho de 2004. A longo prazo, contribuir para o objectivo de construir realmente uma equipa capaz de vencer os campeões humanos de futebol segundo as regras da FIFA [FIFA, 2001] em 2050 e os desafios de investigação que tal acarreta.

Resta-me desejar ao leitor que o texto constante desta tese possa ser objecto de uma leitura agradável e possa contribuir, de uma forma útil, para a sua investigação pessoal com ideias, metodologias, resultados experimentais ou conclusões.

Agradecimentos

É sempre complicado agradecer a todos os que contribuíram para um trabalho desta índole. Com a certeza que muitas outras pessoas ou entidades contribuíram para a realização deste trabalho (e com o meu muito obrigado a esses também), vou tentar agradecer àqueles que de forma mais intensa contribuíram para a realização deste projecto.

Gostaria de agradecer em primeiro lugar e de forma muito especial ao meu orientador, Professor Doutor Eugénio Oliveira, por me ter orientado e sempre me ter proporcionado as melhores condições para a execução deste trabalho. Gostaria ainda de lhe agradecer os seus muito valiosos conselhos, a sempre total disponibilidade e o rigor científico com que reviu o texto final desta tese. Finalmente, gostaria de lhe agradecer a paciência e compreensão que teve, sempre acreditado nas minhas capacidades mesmo quando passei por alguns períodos muito atribulados na minha vida pessoal. Os seus ensinamentos sobre investigação e ciência foram essenciais para a execução deste trabalho. A sua postura crítica, por vezes até demasiado dura, mas sempre franca e muito sensata foram não só um guia para a escrita desta tese como também a essência para a minha formação como cientista. Com a certeza de que fica uma grande amizade e uma mútua vontade de colaboração em futuros projectos científicos aqui lhe transmito o meu profundo agradecimento.

À Fundação de Ciência e Tecnologia que através da bolsa BD5663/95 do programa Praxis XXI, me permitiu iniciar este trabalho de investigação.

À Universidade Fernando Pessoa e em particular ao seu Magnífico Reitor, Professor Doutor Salvato Trigo, por me ter proporcionado as condições necessárias na UFP para poder realizar a minha investigação no LIACC conducente à elaboração deste tese.

À Faculdade de Engenharia da Universidade do Porto e em particular ao Departamento de Engenharia Electrotécnica e de Computadores por me ter proporcionado todas as condições necessárias para poder realizar a minha investigação nas suas instalações do NIAD&R – LIACC.

A todas as instituições que através dos seus subsídios e prémios contribuíram para o sucesso deste projecto, nomeadamente à Associação Portuguesa de Reconhecimento de Padrões, Ordem dos Engenheiros, Fundação Eng. António Almeida, Faculdade de Engenharia da Universidade do Porto, Associação Portuguesa para a Inteligência Artificial, RoboCup Federation e Universidade de Aveiro.

Aos meus colegas do NIAD&R - LIACC, nomeadamente à Ana Paula, Benedita, Conceição, Marcos, Mouta, Paulo Marinho, António Pereira, Andreia, Carlos Miranda,

Célia, Henrique, Li Xin, Max, Pedro Faria e Pedro Silva, o companheirismo e a amizade que criaram um muito bom ambiente de trabalho no laboratório.

Aos meus colegas da Universidade Fernando Pessoa. Um agradecimento especial ao Feliz, Luís Gouveia, Álvaro, David, Nuno Ribeiro, José Vasconcelos, Rui, Torres, Paulo Rurato, Paulo Teixeira, Ivone, Sandra Bernardo e Luísa Vasconcelos.

A todos aqueles que através da leitura de parte ou da totalidade desta tese contribuíram com as suas sugestões ou correcções, nomeadamente à Adelaide João, Ana Paula, Andreia, António Pereira, Armando Sousa, José Torres, Nabais, Nuno Lau, Leão, Pedro Faria, Pedro Moreira e Pedro Silva.

Aos meus alunos e sobretudo aos meus orientados com quem ao longo destes anos fui criando amizades muito fortes, nomeadamente ao Pedro Faria, Pedro Silva, Sérgio Louro, Frederico, Patrícia, Margarida, Ricardo, Paulo Teixeira, Magda, Paulo Cardoso, Paula Salvador, Pedro Santos, Sara Madureira, Evelina, Paula Silva, Cláudia, Sônia Baganha, Armandina, Nuno Reis, Manuel Chavarria, Bruno, Luís Baixinho, Fernando Caldeira, Eduardo Evangelista e Evelina.

Aos meus colegas Nuno Lau e Luis Lopes e à Universidade de Aveiro pela participação projecto FC Portugal em que me envolvi profundamente no ano transacto e cujo sucesso contribuiu para que finalmente me decidisse a escrever esta tese. Aos meus colegas do projecto 5DPO, nomeadamente ao Paulo Costa, António Paulo, Armando, Paulo Marques, Pedro Costa e Susana Gaio pela compreensão do meu pouco tempo disponível nesta fase para colaborar mais activamente neste projecto.

Aos meus amigos e amigas que ao longo destes anos contribuíram para recriar uma vida pessoal muito interessante que permitisse o escape ao stress do trabalho: ao Américo, Cláudia, Lau, Lúcia, Nabais, Torres, Leão, Miguel, Morado, Pedro Faria, Pedro Moreira, Pedro Silva, Sara, Sofia e Zé Carlos. Um agradecimento ainda a todas as pessoas que me aturaram a outro nível ao longo destes anos, nomeadamente à: Alexandra, Ana, Bárbara, Carla, Céu, Paula e Susana.

Este projecto sofreu profundas alterações devido a motivos pessoais que nada tiveram a ver com a sua parte científica. À Nela agradeço por sete anos em conjunto, alguns deles bastante felizes. Desta vez, não te posso agradecer teres ajudado a elaborar as ilustrações, mas posso agradecer-te por me teres encorajado a fazer este trabalho e por um dia teres existido na minha vida. Aquele sorriso e aquele português com sotaque em Woodhouse Lane, o brilho daqueles olhos verdes um ano depois em Pedras Rubras e aquela lágrima na Serra do Pilar, marcaram para sempre a minha vida e este trabalho... Desculpa! A muitos milhares de quilómetros e a cinco anos de distância, um agradecimento especial para a responsável pela mudança profunda que a minha vida sofreu, pela alteração e pelo enorme atraso deste trabalho. Para ti Sônia, que um dia me deste um coração mas que, durante muitos anos, ficaste com ele, do outro lado do oceano, vai o meu agradecimento.

À minha família, especialmente à minha mãe mas também aos meus irmãos Miguel e Rui, à Olga, aos meus Avós e Tios, a todos os meus Priminhos e ao meu sobrinho. O bom ambiente familiar que sempre me proporcionaram foi decisivo em alguns momentos para poder continuar o trabalho conducente à elaboração desta tese.

Ao meu pai vou repetir parcialmente o agradecimento que lhe fiz há sete anos na minha tese de Mestrado. Desta vez as lágrimas são substituídas pela saudade. Custou-me muito ver-te "abandonar" uma vida cheia de futuro... Depois de ter passado nos últimos anos por períodos semelhantes e pela mesma emoção com mais sorte, consigo-te compreender muito melhor. A alguém que tinha um grande futuro pela frente mas que o destino quis que tivesse um fim prematuro. Com a certeza que a quase todos os níveis continuas a ser o meu modelo como Homem e como Cientista. A alguém cuja bondade, criatividade, imaginação e simpatia um dia gostava de alcançar...

Por fim, vou agradecer à minha "princesinha"... À Joãozinha que me conheceu e "aturou" nesta fase complicada de mudança de Universidade, serviço docente a duplicar, projectos de investigação a arrancar e escrita da tese de doutoramento mas que, com o seu amor, me deu a força necessária para a conseguir atravessar. Não foi fácil começar uma vida a dois nestas circunstâncias especiais e eu sei que, nem sempre, te pude dar a atenção e o carinho que mereces. O brilho desse teu olhar meigo que me faz sonhar é a inspiração da minha vida, e foi a inspiração que me fez escrever este trabalho. Essa tua alma linda que todos os dias me faz acreditar é a razão da minha vida, e foi a força que me fez acabar este trabalho. Para alguém que me ensinou o que eu pensava ser já impossível: a amar. Com muito amor e com a promessa que as coisas agora vão ser diferentes...

Índice

1. Introdução **1**

1.1 Enquadramento 1

1.1.1 Coordenação em Sistemas Multi-Agente 2

1.1.2 Escalonamento e Gestão Universitária 2

1.1.3 O Futebol Robótico e a Liga de Simulação 3

1.2 Motivação 5

1.3 Objectivos 6

1.4 Contribuições Científicas 7

1.5 Estrutura da Tese 8

2. Agentes Autónomos **11**

2.1 Áreas Científicas que Inspiraram os Agentes 12

2.2 Agentes 13

2.2.1 Definições de Agente 14

2.2.2 Atributos dos Agentes 19

2.2.2.1 Autonomia 19

2.2.2.2 Mobilidade 20

2.2.2.3 Reactividade 21

2.2.2.4 Pró-Actividade 22

2.2.2.5 Comunicação 22

2.2.2.6 Habilidade Social 23

2.2.2.7 Cooperação 23

2.2.2.8 Aprendizagem 24

2.2.3 Agentes vs. Objectos 24

2.3 Ambientes 26

2.4 Classificação dos Agentes 30

2.5 Arquitecturas de Agentes 32

2.5.1 Arquitecturas Deliberativas 34

2.5.2 Arquitecturas Reactivas 35

2.5.3 Arquitecturas Híbridas 37

2.5.4 Arquitectura por Camadas 38

2.5.5 Arquitectura BDI 40
2.5.6 Arquitectura Genérica de um Agente Social 43
2.5.7 Selecção da Arquitectura Mais Adequada 45
2.6 Aplicações de Agentes Autónomos 45
2.7 Conclusões 47

3. Sistemas Multi-Agente 49

3.1 O Conceito de Sistema Multi-Agente 50
3.2 Inteligência Artificial Distribuída vs. SMA 52
3.3 Motivação dos SMA 55
3.4 Comunicação em SMA 57
3.4.1 Arquitecturas do Sub-Sistema de Comunicação 58
3.4.2 Características da Comunicação 59
3.4.2.1 Caracterização do Significado das Mensagens 61
3.4.2.2 Tipo de Mensagens 62
3.4.2.3 Protocolos e Níveis de Comunicação 62
3.4.2.4 Actos de Discurso 63
3.4.3 Linguagens de Comunicação 64
3.4.3.1 KIF – Knowledge Interchange Format 65
3.4.3.2 KQML - Knowledge and Query Manipulation Language 66
3.4.3.3 FIPA ACL – Agent Communication Language 67
3.4.3.4 Ontologias 68
3.5 Aprendizagem em SMA 70
3.5.1 Tipos de Aprendizagem em SMA 71
3.5.2 Determinação do Crédito ou Culpa 71
3.5.3 Características Importantes na Aprendizagem 72
3.5.4 Algoritmos de Aprendizagem 74
3.5.5 Dificuldades Específicas da Aprendizagem em SMA 75
3.6 Coordenação em SMA 75
3.6.1 Definição de Coordenação 76
3.6.2 A Necessidade de Coordenar Agentes 77
3.6.3 SMA Cooperativos vs. Competitivos 80
3.6.4 Coordenação de Agentes Competitivos 80
3.6.5 Coordenação de Agentes Cooperativos 81
3.7 Conclusões 82

4. Coordenação de Agentes Competitivos 83

4.1 Negociação em SMA ... 84

4.2 Resolução de Conflitos ... 85

4.3 Princípios da Negociação Computacional ... 90

4.3.1 Tipos de Negociação ... 91

4.3.2 Fases do Processo de Negociação ... 93

4.3.3 Espaço de Negociação ... 95

4.3.4 Parâmetros do Espaço de Negociação ... 95

4.3.4.1 Cardinalidade da Negociação ... 96

4.3.4.2 Características dos Agentes ... 97

4.3.4.3 Ambientes e Características dos Produtos ... 98

4.3.4.4 Parâmetros dos Eventos ... 99

4.3.4.5 Parâmetros de Informação ... 100

4.3.4.6 Parâmetros de Alocação ... 100

4.3.5 Protocolos de Negociação ... 100

4.3.5.1 Características do Protocolo de Negociação ... 102

4.3.5.2 Esquema Genérico para Negociação Electrónica ... 103

4.4 Leilões ... 104

4.5 Modelos de Negociação ... 106

4.5.1 Negociação Baseada na Teoria dos Jogos ... 106

4.5.2 Negociação Baseada em Planos ... 107

4.5.3 Negociação Baseada em Técnicas Humanas ... 108

4.6 Negociação Multi-Atributo ... 109

4.7 Estratégias e Tácticas de Negociação ... 110

4.8 Utilização Prática da Negociação ... 116

4.8.1 O Mercado Kasbah ... 116

4.8.2 O Sistema SMACE ... 117

4.8.3 O Sistema MACIV ... 118

4.8.4 FOREV - Formação de Empresas Virtuais ... 119

4.8.5 O *AuctionBot* e a *Trading Agent Competition* ... 121

4.9 Conclusões ... 123

5. Coordenação de Agentes Cooperativos 125

5.1 Objectivos da Coordenação de Agentes Cooperativos ... 126

5.2 Formação de Equipas de Agentes ... 127

5.3 Resolução Distribuída e Cooperativa de Problemas 128
5.3.1 Coordenação por Partilha de Tarefas e de Resultados 129
5.3.2 Coordenação por Partilha de Tarefas – Protocolo da Rede Contratual 131
5.3.3 Coordenação por Comunicação – Partilha de Resultados 132
5.4 Coordenação por Planeamento Multi-Agente 134
5.4.1 Planeamento Global Parcial 135
5.4.2 Planeamento Global Parcial Generalizado 136
5.5 Modelos de Coordenação 138
5.5.1 Intenções Conjuntas ("*Joint Intentions Framework*") 138
5.5.2 Planos Partilhados ("*Shared Plans*") 140
5.5.3 Acordo de Balneário ("*Locker Room Agreement*") 141
5.5.4 TAEMS Framework 143
5.5.5 STEAM – simply a Shell for Teamwork 145
5.6 Coordenação sem Comunicação 146
5.6.1 Coordenação por Modelização Mútua 146
5.6.2 Coordenação por Percepção Inteligente 149
5.7 Coordenação por Organização Estrutural 150
5.7.1 Coordenação por Normas e Leis Sociais 151
5.7.2 Coordenação por Conhecimento à-Priori 152
5.8 Coordenação por Controlo Parcialmente Hierárquico 153
5.9 Coordenação Estratégica 155
5.9.1 Coordenação Situacional (Temporal) 156
5.9.2 Coordenação Espacial por Posicionamento Estratégico 157
5.9.3 Coordenação por Definição e Troca de Papéis 157
5.10 Conclusões 158

6. Problemas de Escalonamento na Gestão Universitária 161

6.1 Organização e Gestão Universitária 162
6.2 Problemas de Optimização Combinatória 164
6.3 Problemas de Escalonamento na Gestão Universitária 166
6.3.1 Distribuição de Serviço Docente 169
6.3.1.1 Descrição do Problema 169
6.3.1.2 Solução do Problema 170
6.3.1.3 Restrições Rígidas 170
6.3.1.4 Restrições Flexíveis 171
6.3.2 Geração de Horários 172

6.3.2.1 Descrição do Problema ... 172
6.3.2.2 Solução do Problema ... 173
6.3.2.3 Restrições Rígidas ... 173
6.3.2.4 Restrições Flexíveis ... 174
6.3.3 Calendarização de Exames ... 175
6.3.3.1 Descrição do Problema ... 175
6.3.3.2 Solução do Problema ... 176
6.3.3.3 Restrições Rígidas ... 177
6.3.3.4 Restrições Flexíveis ... 177
6.3.4 Alocação de Salas ... 178
6.3.5 Visão Integrada ... 178
6.4 Técnicas de Resolução dos Problemas de Escalonamento ... 179
6.4.1 Coloração de Grafos ... 180
6.4.2 Programação Inteira ... 180
6.4.3 Heurísticas ... 183
6.4.4 Meta-Heurísticas ... 184
6.4.4.1 Arrefecimento Simulado ... 184
6.4.4.2 Pesquisa Tabu ... 185
6.4.4.3 Algoritmos Genéticos ... 188
6.4.5 Programação em Lógica ... 190
6.4.6 Programação em Lógica com Restrições ... 191
6.4.7 Sistemas Periciais ... 192
6.4.8 Sistemas Interactivos ... 193
6.4.9 Sistemas Multi-Agente ... 194
6.5 Sistemas Comerciais de Geração de Horários ... 195
6.6 Representação de Problemas de Escalonamento Universitários ... 201
6.6.1 A Linguagem TTL de Cooper e Kingston ... 201
6.6.2 A Linguagem Proposta por Cumming e Paechter ... 201
6.6.3 O GATT de Collingwood, Ross e Corne ... 202
6.6.4 A Linguagem de Burke, Kingston e Pepper ... 202
6.6.5 A Linguagem STTL de Kingston ... 203
6.6.6 A Linguagem Unilang ... 203
6.7 Conclusões ... 204

7. Coordenação em SMA na Gestão Universitária 207

7.1 Tipos de Coordenação Aplicáveis ao Domínio ... 208

7.2 Formalização de Problemas de Escalonamento Universitário 209
7.3 A Linguagem *SCHEDULING UNILANG* 214
7.3.1 Sub-Problemas da Construção de Horários 215
7.3.2 Requisitos da Linguagem *SCHEDULING UNILANG* 217
7.3.3 Componentes da Linguagem 218
7.3.4 Representação do Tempo 220
7.3.5 Representação do Espaço 222
7.3.6 Descrição de Eventos 223
7.3.7 Turmas e Alunos 224
7.3.8 Docentes e Vigilantes 225
7.3.9 Restrições de Carga de Trabalho, Espalhamento e Ordenação 226
7.3.10 Mecanismos de Sobreposição e Omissão 227
7.3.11 Solução Final e Função de Avaliação 227
7.4 O Sistema UNIPS 228
7.4.1 Linguagem e Plataforma de Desenvolvimento Seleccionada 228
7.4.2 Potencialidades Requeridas ao Sistema 229
7.4.3 Arquitectura do Sistema 229
7.4.4 Interface com o Utilizador 230
7.4.5 Resolução Centralizada de Problemas de Escalonamento 233
7.4.6 O Sistema UNIPS – Modo Multi-Agente 234
7.4.6.1 Agentes Escalonadores 235
7.4.6.2 Agentes Pessoais 235
7.4.6.3 Agentes Directores 236
7.4.7 Discussão e Análise de Resultados 236
7.5 Conclusões 237

8. Análise do Domínio de Aplicação: Futebol Robótico 239

8.1 Introdução 240
8.1.1 Objectivos do RoboCup 240
8.1.2 História do RoboCup 244
8.1.3 A Federação do RoboCup 246
8.2 As Ligas do RoboCup 247
8.2.1 A Liga de Simulação 248
8.2.2 A Liga de Robôs Pequenos – *Small-Size* 250
8.2.3 A Liga de Robôs Médios – *Middle-Size* 251
8.2.4 A Liga de Robôs com Pernas – Cães da Sony 252
8.2.5 O *RoboCup Rescue:* Salvamento em Catástrofes 255

8.2.6 O RoboCup Júnior 256
8.2.7 Outros Desafios Associados 258
8.2.7.1 Treinador *On-Line* 258
8.2.7.2 Visualizadores 3D 258
8.2.7.3 Comentador Inteligente 259
8.3 O Futebol Robótico Simulado 260
8.3.1 Aplicações Constituintes do Simulador 260
8.3.1.1 O Simulador – *SoccerServer* 261
8.3.1.2 O Monitor – *Soccer Monitor* 262
8.3.1.3 O Vídeo - *LogPlayer* 264
8.3.2 O Mundo Simulado e as Regras do Jogo 264
8.3.2.1 O Campo e os Objectos 264
8.3.2.2 Os Árbitros Artificial e Humano 265
8.3.3 Protocolos de Comunicação com o Servidor 266
8.3.3.1 Protocolo de Conexão dos Clientes 266
8.3.3.2 Protocolo de Percepção dos Clientes 267
8.3.3.3 Protocolo de Acção dos Clientes 268
8.3.4 Percepção dos Agentes 269
8.3.4.1 Informação Visual 269
8.3.4.2 Informação Auditiva 273
8.3.4.3 Informação Física 274
8.3.5 Acção dos Agentes 275
8.3.5.1 Movimento – Move 276
8.3.5.2 Movimento – Aceleração e Energia 277
8.3.5.3 Movimento – Rotação 279
8.3.5.4 Controlo da Bola – Chuto (*Kick*) 280
8.3.5.5 Controlo da Bola – Agarrar a Bola (*Catch*) 282
8.3.5.6 Controlo da Bola - *Tackle* 283
8.3.5.7 Controlo da Percepção – Rotação do Pescoço Flexível 283
8.3.5.8 Controlo da Percepção – Configuração da Visão 284
8.3.5.9 Controlo da Percepção – Configuração da Audição 284
8.3.5.10 Comunicação - Say 285
8.3.5.11 Comunicação - Pointto 285
8.3.6 Agentes Heterogéneos 286
8.3.7 O Treinador 287
8.3.7.1 Capacidades do Treinador *off-line* 287
8.3.7.2 Capacidades do Treinador on-line 287

8.3.7.3 Linguagem "Standard" de Comunicação 288
8.4 Abordagens mais significativas ao Futebol Robótico Simulado 289
8.5 Trabalho Relacionado e Aplicações do RoboCup 290
8.6 O RoboCup como Competição Científica 292
8.7 Conclusões 295

9. Coordenação em SMA no Futebol Robótico 299

9.1 Tipos de Coordenação Aplicáveis ao Domínio 301
9.2 Influência do Futebol no Futebol Simulado 302
9.3 Arquitectura dos Agentes 303
9.3.1 A Arquitectura *Strategic Agent* 303
9.3.2 Estado do Mundo Multi-Resolução 305
9.3.2.1 Características da Simulação 306
9.3.2.2 Estado do Mundo 306
9.3.2.3 Estrutura do Estado do Mundo Multi-Resolução 308
9.3.2.4 Actualização do Estado do Mundo 309
9.3.2.5 Predição dos Efeitos das Acções 312
9.4 Comportamentos de Baixo-Nível *(Low-Level Skills)* 313
9.4.1 Movimentação Evitando Adversários 314
9.4.2 Intercepção da Bola 314
9.4.3 Drible Inteligente 315
9.4.4 Chuto Optimizado *("Optimization Kick")* 316
9.5 Módulo de Decisão de Alto Nível 323
9.6 Coordenação Estratégica no Futebol Robótico 325
9.6.1 Definição Formal da Estratégia da Equipa 325
9.6.2 Definição de Situações 329
9.6.3 SBSP – Posicionamento Estratégico Baseado em Situações 330
9.6.4 DPRE – Troca Dinâmica de Papéis e Posicionamentos 335
9.7 Outros Mecanismos de Coordenação 340
9.7.1 Coordenação por Comunicação Avançada - ADVCOM 340
9.7.2 Coordenação por Modelização Mútua 344
9.7.3 Percepção Inteligente 346
9.8 Decisão de Alto-Nível 349
9.8.1 Decisão com Bola 351
9.8.1.1 Geração e Avaliação de Passes 352

9.8.1.2 Geração e Avaliação de Passes em Profundidade (Forwards), Remates e Dribles 354
9.8.1.3 Fusão das Avaliações 355
9.8.1.4 Avaliação da Decisão com Bola 355
9.8.2 Decisão sem Bola 356
9.8.3 Decisão do Guarda-Redes 357
9.9 Ferramentas Associadas 358
9.9.1 Visual Debugger 359
9.9.2 Offline Client 361
9.9.3 Analisador do Erro no Estado do Mundo 361
9.9.4 Analisador de Jogo e Cálculo de Estatísticas 362
9.9.5 Visualizador 3D 362
9.10 A Linguagem *Coach Unilang* 364
9.10.1 Introdução 364
9.10.2 Requisitos da Linguagem 366
9.10.3 Especificação Formal 367
9.10.3.1 Definição de Conceitos 368
9.10.3.2 Regiões do Campo 369
9.10.3.3 Períodos de Tempo 370
9.10.3.4 Condições e Situações 371
9.10.3.5 Tácticas 371
9.10.3.6 Formações 375
9.10.3.7 Tipos de Jogadores 376
9.10.4 A Linguagem do Treinador Adjunto 378
9.10.4.1 Informação sobre o Oponente 378
9.10.4.2 Estatísticas de Jogo 380
9.10.5 Instruções do Treinador 382
9.10.6 A Linguagem *COACH UNILANG LIGHT* 382
9.10.7 Conclusões 383
9.11 Coordenação por Controle Parcialmente Hierárquico 384
9.11.1 Definição de um Treinador on-line 384
9.11.2 Utilização das Instruções do Treinador nos Jogadores 386
9.12 Resultados em Competições RoboCup 387
9.12.1 Avaliação dos Resultados Obtidos 387
9.12.2 Campeonato Europeu – Amesterdão 2000 388
9.12.3 RoboCup 2000 - Melbourne 389
9.12.4 German Open 2001 – Paderborn 390

9.12.5 RoboCup 2001 – Seattle 392
9.13 Resultados em Experiências Controladas 393
9.13.1 Definição das Experiências 393
9.13.2 Coordenação Estratégica 395
9.13.2.1 Definição de Tácticas, Formações e Papéis 395
9.13.2.2 Coordenação por Posicionamento Estratégico 397
9.13.2.3 Coordenação por Troca Dinâmica de Papéis e Posicionamentos 399
9.13.3 Coordenação por Controlo Parcialmente Hierárquico 400
9.13.4 Coordenação por Comunicação, Percepção Inteligente e Modelização Mútua 401
9.13.5 Análise de Resultados 403
9.14 Conclusões 404

10. Conclusões e Perspectivas de Desenvolvimento 405

10.1 Síntese do Trabalho Desenvolvido 405
10.2 Principais Resultados e Conclusões 406
10.3 Originalidades 408
10.4 Limitações 409
10.5 Perspectivas de Desenvolvimento 409

Lista de Figuras

FIGURA 1: CAMPOS QUE INSPIRARAM OS AGENTES E SISTEMAS MULTI-AGENTE 12
FIGURA 2: ESQUEMA TÍPICO DE UM AGENTE 14
FIGURA 3: CATEGORIAS DE AGENTES DEFINIDAS POR NWANA 31
FIGURA 4: CATEGORIAS DE AGENTES DEFINIDAS POR FRANKLIN E GRAESSER 31
FIGURA 5: ESQUEMA GENÉRICO DE UMA ARQUITECTURA DELIBERATIVA 35
FIGURA 6: ESQUEMA GENÉRICO DE UMA ARQUITECTURA REACTIVA 36
FIGURA 7: ESQUEMA GENÉRICO DE UMA ARQUITECTURA HÍBRIDA 37
FIGURA 8: REPRESENTAÇÃO DA ARQUITECTURA POR CAMADAS HORIZONTAIS 39
FIGURA 9: REPRESENTAÇÃO DA ARQUITECTURA POR CAMADAS VERTICAIS 39
FIGURA 10: ESQUEMA GENÉRICO DE UMA ARQUITECTURA BDI 41
FIGURA 11: ESQUEMA GENÉRICO DE UMA ARQUITECTURA DE UM AGENTE SOCIAL 44
FIGURA 12: ESTRUTURA DE UM SISTEMA MULTI-AGENTE 51
FIGURA 13: INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA 53
FIGURA 14: AGENTE COM CAPACIDADE DE COMUNICAÇÃO 57
FIGURA 15: ARQUITECTURAS DE COMUNICAÇÃO DE AGENTES A) DIRECTA E B) ASSISTIDA 58
FIGURA 16: EXEMPLO DE UMA MENSAGEM KQML 66
FIGURA 17: CARACTERÍSTICAS DA APRENDIZAGEM EM SMA 72
FIGURA 18: DIAGRAMA DAS CAUSAS E CONSEQUÊNCIAS DO CONFLITO. 85
FIGURA 19: TIPOS DE NEGOCIAÇÃO 92
FIGURA 20: FASES DO PROCESSO DE NEGOCIAÇÃO 93
FIGURA 21: PARÂMETROS DO ESPAÇO DE NEGOCIAÇÃO [REIS, 2002D] 96
FIGURA 22: PROTOCOLO DE NEGOCIAÇÃO PARA DOIS AGENTES [FARATIN ET AL., 1999] 101
FIGURA 23: ESPAÇO DE ACORDOS DE NEGOCIAÇÃO 103
FIGURA 24: DIAGRAMA DO SISTEMA MACIV [FONSECA, 2000] 118
FIGURA 25: FORMAÇÃO DE UMA EMPRESA VIRTUAL NO SISTEMA FOREV [MACEDO, 2001] 120
FIGURA 26: DIAGRAMA DA TAC - TRADING AGENT COMPETITION 122
FIGURA 27: MODELO DE COORDENAÇÃO 127
FIGURA 28: TAREFAS COOPERATIVAS NA RDPC: A) PARTILHA DE TAREFAS E B) PARTILHA DE RESULTADOS 131
FIGURA 29: PROTOCOLO CONTRACT-NET – REDE DE CONTRATUAL 132
FIGURA 30: EXEMPLO DE UMA ESTRUTURA DE TAREFAS TAEMS 144

FIGURA 31: ARQUITECTURA DOS AGENTES ARCHON 148
FIGURA 32: ESTRUTURA DA UNIVERSIDADE INTEGRADA 163
FIGURA 33: ESTRUTURA DA UNIVERSIDADE FEDERATIVA 163
FIGURA 34: PROBLEMAS DE ESCALONAMENTO NUMA INSTITUIÇÃO DE ENSINO 168
FIGURA 35: EXEMPLO DE UM HORÁRIO 172
FIGURA 36: EXEMPLO DE UM CALENDÁRIO DE EXAMES 176
FIGURA 37: JANELA DE VISUALIZAÇÃO DA GERAÇÃO AUTOMÁTICA DO GPUNTIS (ESQUERDA) E DEFINIÇÃO DE PLANOS CURRICULARES DO GESTHOR (DIREITA) 199
FIGURA 38: REPRESENTAÇÃO GENÉRICA DE UM PROBLEMA DE CONSTRUÇÃO DE HORÁRIOS 215
FIGURA 39: ARQUITECTURA DO SISTEMA UNIPS 230
FIGURA 40: INTERFACE GRÁFICA DO SISTEMA UNIPS 231
FIGURA 41: AGENTE ESCALONADOR GERADOR DE HORÁRIOS DO SISTEMA UNIPS 232
FIGURA 42: ARQUITECTURA DE RESOLUÇÃO CENTRALIZADA DE PROBLEMAS DO SISTEMA UNIPS 233
FIGURA 43: FACTOS PROLOG GERADOS APÓS A CONVERSÃO UNILANG-CLP DE UM PROBLEMA DE CALENDARIZAÇÃO DE EXAMES 234
FIGURA 44: INTERFACE DE DEFINIÇÃO DOS PESOS DAS RESTRIÇÕES FLEXÍVEIS DO UNIPS 234
FIGURA 45: ASPECTO GERAL DO FUKUOKA DOME – ROBOCUP2002 245
FIGURA 46: AS TRÊS VERTENTES DO ROBOCUP 247
FIGURA 47: A LIGA DE ROBÔS PEQUENOS (SMALL-SIZE) 250
FIGURA 48: A LIGA DE ROBÔS MÉDIOS (MIDDLE-SIZE) 252
FIGURA 49: A LIGA DE ROBÔS COM PERNAS DA SONY 253
FIGURA 50: ROBOCUP RESCUE (VISUALIZADORES – EM CIMA E COMPETIÇÃO – EM BAIXO) 256
FIGURA 51: ROBOCUP JÚNIOR (MODALIDADES DE FUTEBOL E DANÇA) 257
FIGURA 52: APLICAÇÕES CONSTITUINTES DO SISTEMA DE SIMULAÇÃO 260
FIGURA 53: ARQUITECTURA DO SISTEMA DE SIMULAÇÃO SOCCERSERVER 261
FIGURA 54: MONITOR TRADICIONAL PARA LINUX DO SOCCER SERVER. 263
FIGURA 55: MONITORES PARA WINDOWS DO SOCCER SERVER (FREIBURG E TSINGHUA). 263
FIGURA 56: LOCALIZAÇÃO E NOMES DOS MARCOS DE ORIENTAÇÃO NO SOCCER SERVER. 264
FIGURA 57: PERCEPÇÃO VISUAL DOS AGENTES [STONE, 1998]. 272
FIGURA 58: MODELO DE CATCH DO SIMULADOR 282
FIGURA 59: ARQUITECTURA STRATEGIC AGENT DOS AGENTES FC PORTUGAL 304
FIGURA 60: INFORMAÇÃO NO NÍVEL INFERIOR DO ESTADO DO MUNDO RELATIVA AOS OUTROS JOGADORES 307
FIGURA 61: COORDENADAS ABSOLUTAS E ÂNGULOS DO CAMPO DE JOGO 307
FIGURA 62: ESTADO DO MUNDO MULTI-NÍVEL DOS AGENTES FC PORTUGAL 308

FIGURA 63: ACTUALIZAÇÃO DO ESTADO DO MUNDO 310
FIGURA 64: CHUTO UTILIZANDO VÁRIOS KICKS, EVITANDO ADVERSÁRIOS 316
FIGURA 65: RESULTADOS OBTIDOS NA PRIMEIRA FASE DO OPTIMIZATION KICK EM 2000 EXPERIÊNCIAS (COM KICK_POWER_RATE = 0.016) 322
FIGURA 66: RESULTADOS OBTIDOS NA SEGUNDA FASE DO OPTIMIZATION KICK EM 2000 EXPERIÊNCIAS (COM KICK_POWER_RATE = 0.016) 322
FIGURA 67: FLUXOGRAMA DO MÓDULO DE DECISÃO DE ALTO-NÍVEL 323
FIGURA 68: DIAGRAMA DA ESTRATÉGIA DO FC PORTUGAL 326
FIGURA 69: EXEMPLOS DE APLICAÇÃO DO ALGORITMO SBSP EM DIVERSAS SITUAÇÕES 334
FIGURA 70: COMPARAÇÃO ENTRE O SBSP E O ALGORITMO DE POSICIONAMENTO SPAR 334
FIGURA 71: CRIAÇÃO DE UM ESTADO DO MUNDO COMUNICADO E SUA UTILIZAÇÃO NA DECISÃO DE COMUNICAÇÃO 340
FIGURA 72: GERAÇÃO DE PASSES 352
FIGURA 73: GERAÇÃO DE PASSES EM PROFUNDIDADE (FORWARDS) 355
FIGURA 74: INTERFACE DO VISUAL DEBUGGER – DECISÃO DE CHUTO À BALIZA 360
FIGURA 75: INTERFACE DO VISUAL DEBUGGER – DECISÃO SEM BOLA 360
FIGURA 76: ARQUITECTURA DO VISUALIZADOR 3D 363
FIGURA 77: INTERFACE DO VISUALIZADOR 3D 364
FIGURA 78: ARQUITECTURAS DE TREINADOR POSSÍVEIS UTILIZANDO A LINGUAGEM COACH UNILANG 365
FIGURA 79: ESTRUTURA MULTI-RESOLUÇÃO COMPOSTA PELAS REGIÕES PREDEFINIDAS NA LINGUAGEM 369
FIGURA 80: DEFINIÇÃO DE FORMAÇÕES UTILIZANDO O QUADRO VIRTUAL 7*5 375
FIGURA 81: ARQUITECTURA DO TREINADOR DA EQUIPA FC PORTUGAL 385
FIGURA 82: REGIÕES PERIGOSA E MUITO PERIGOSA CONSIDERADAS COMO CONDUCENTES A OPORTUNIDADES DE GOLO 388

Lista de Tabelas

TABELA 1: EVOLUÇÃO DOS PARADIGMAS DE PROGRAMAÇÃO. 26
TABELA 2: CAPACIDADES DE COMUNICAÇÃO DE DIFERENTES AGENTES [HUHNS E STEPHENS, 1999] 62
TABELA 3: TIPOS DE MENSAGENS INTER-AGENTES DERIVADAS DA TEORIA DO ACTO DE DISCURSO 64
TABELA 4: PARÂMETROS DE UMA MENSAGEM KQML 66
TABELA 5: PERFORMATIVAS PROVIDENCIADAS PELA LINGUAGEM FIPA ACL. 67
TABELA 6: TIPOS DE DEPENDÊNCIA ENTRE ACTIVIDADES [MALONE E CROWSTON, 1994] 78
TABELA 7: COMPARATIVO ENTRE AS APLICAÇÕES DE CONSTRUÇÃO DE HORÁRIOS. 200
TABELA 8: PROBLEMAS DE ESCALONAMENTO UNIVERSITÁRIO CONSIDERADOS. 237
TABELA 9: DIFERENÇAS ENTRE AS CARACTERÍSTICAS DOS DOMÍNIOS DO ROBOCUP E XADREZ. 242
TABELA 10: PROTOCOLO DE CONEXÃO DOS CLIENTES AO SIMULADOR 266
TABELA 11: PROTOCOLO DE PERCEPÇÃO DOS CLIENTES 268
TABELA 12: PROTOCOLO DE ACÇÃO DOS CLIENTES 269
TABELA 13: FREQUÊNCIA VISUAL DOS AGENTES DE ACORDO COM O SEU MODO VISUAL 270
TABELA 14: ALTERAÇÕES NO MODELO DE COMUNICAÇÕES DE 2001 PARA 2002 274
TABELA 15: ACÇÕES POSSÍVEIS PARA OS AGENTES NA LIGA DE SIMULAÇÃO DO ROBOCUP 276
TABELA 16: PARÂMETROS QUE AFECTAM A ACÇÃO KICK (E VALORES PARA COMPETIÇÕES EM 2002). 282
TABELA 17: PARÂMETROS QUE AFECTAM A ACÇÃO CATCH. 283
TABELA 18: PARÂMETROS DOS JOGADORES HETEROGÉNEOS 286
TABELA 19: TRINTA LINHAS DE ORIENTAÇÃO PARA CRIAR UMA EQUIPA DE FUTEBOL SIMULADO 303
TABELA 20: SITUAÇÕES UTILIZADAS E SITUAÇÕES SUBSTITUTAS 329
TABELA 21: DESCOORDENAÇÕES NA TROCA DINÂMICA DE POSIÇÕES (10 JOGOS CONTRA CADA EQUIPA) 339
TABELA 22: ERRO MÉDIO NO ESTADO DO MUNDO COM DIFERENTES ESTRATÉGIAS DE COMUNICAÇÃO 344
TABELA 23: ERRO MÉDIO NO ESTADO DO MUNDO COM DIFERENTES ESTRATÉGIAS DE MODELIZAÇÃO MÚTUA 346
TABELA 24: ERRO MÉDIO NO ESTADO DO MUNDO COM DIFERENTES ESTRATÉGIAS DE VISÃO ESTRATÉGICA 349
TABELA 25: RESULTADOS EXPERIMENTAIS NA AVALIAÇÃO DA DECISÃO COM BOLA 356
TABELA 26: RESULTADOS OBTIDOS PELA FC PORTUGAL NO CAMPEONATO EUROPEU – AMESTERDÃO 2000 389
TABELA 27: RESULTADOS OBTIDOS PELA FC PORTUGAL NO CAMPEONATO MUNDIAL – MELBOURNE 2000. 390
TABELA 28: RESULTADOS OBTIDOS PELA FC PORTUGAL NO GERMAN OPEN 2001 391
TABELA 29: RESULTADOS OBTIDOS PELA FC PORTUGAL NO CAMPEONATO MUNDIAL – SEATTLE 2001 392

TABELA 30: RESULTADOS DAS EXPERIÊNCIAS CONTROLADAS SIMPLES. 394
TABELA 31: RESULTADOS DAS EXPERIÊNCIAS COM VARIAÇÃO DE TÁCTICAS 396
TABELA 32: RESULTADOS DAS EXPERIÊNCIAS COM VARIAÇÃO DE ESQUEMA DE POSICIONAMENTO 398
TABELA 33: RESULTADOS DAS EXPERIÊNCIAS COM VARIAÇÃO DO ESQUEMA DE TROCA DINÂMICA DE POSICIONAMENTOS 399
TABELA 34: RESULTADOS DAS EXPERIÊNCIAS COM COORDENAÇÃO PARCIALMENTE HIERÁRQUICA 401
TABELA 35: RESULTADOS DAS EXPERIÊNCIAS COM COORDENAÇÃO POR COMUNICAÇÃO, MODELIZAÇÃO MÚTUA E PERCEPÇÃO INTELIGENTE 402
TABELA 36: MENSAGENS DO ÁRBITRO ARTIFICIAL E SEU SIGNIFICADO 410

Capítulo 1

1. Introdução

1.1 Enquadramento

O trabalho descrito nesta tese está relacionado com o estudo e proposta de metodologias de coordenação de Agentes computacionais[1] autónomos e sua aplicação em domínios que implicam a execução de tarefas complexas através de grupos de agentes, nomeadamente:

- Resolução de problemas de escalonamento na gestão universitária: distribuição de serviço docente, geração de horários, calendarização de exames, alocação de salas, etc.; e
- Execução de tarefas cooperativas através de equipas de agentes com mobilidade espacial, nomeadamente através da construção de uma equipa para a liga de simulação do futebol robótico – RoboCup.

Os dois domínios de teste seleccionados e as respectivas tarefas cooperativas a realizar, são ambos bastante complexos e debruçam-se sobre a coordenação de agentes, possuindo, no entanto, diversas características que os diferenciam significativamente. A resolução de problemas de escalonamento na gestão universitária é realizada por um número elevado de agentes heterogéneos que competem e cooperam uns com os outros nesta resolução. No futebol robótico, o grau de heterogeneidade dos agentes é reduzido e os agentes são totalmente cooperativos, formando aquilo que vulgarmente se designa por uma equipa. No entanto, o domínio introduz diversas complexidades adicionais como sejam o seu dinamismo, a distribuição espacial dos agentes, erros na percepção e acção dos mesmos e a complexidade da tarefa cooperativa a executar (jogar um desafio de futebol). Devido às diferenças nas suas características, os dois domínios implicam a utilização de metodologias de coordenação bastante distintas das actividades realizadas pelos agentes.

---

[1] Ao longo da tese, os termos agente e sistema multi-agente são utilizados referindo-se a sistemas computacionais.

1.1.1 Coordenação em Sistemas Multi-Agente

Um *Agente* pode ser definido como uma entidade computacional (vulgarmente *software*), situado num dado ambiente, que tem a percepção desse ambiente através de sensores, tem capacidade de raciocínio e age de forma autónoma nesse ambiente através de actuadores, de forma a desempenhar uma dada função para a qual foi projectado. Alguns investigadores afirmam que um agente para ser considerado como tal, deve também possuir capacidades de comunicação de alto-nível com outros agentes e/ou humanos. No entanto, a definição do conceito de agente não é consensual e é tão problemática para a comunidade de Inteligência Artificial Distribuída como a definição de inteligência para a comunidade de Inteligência Artificial.

Os *Sistemas Multi-Agente* são sistemas compostos por múltiplos agentes, que exibem um comportamento autónomo mas ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais: serem capazes de agir de forma autónoma, tomando decisões, levando à satisfação dos seus objectivos; serem capazes de interagir com outros agentes utilizando protocolos de interacção social inspirados nos dos humanos e requerendo coordenação, cooperação e negociação.

A *Coordenação* pode ser definida como o acto de trabalhar em conjunto de forma harmoniosa no sentido de atingir um acordo ou objectivo comum. No entanto, o trabalhar em conjunto não implica necessariamente cooperação entre os agentes. No caso de agentes competitivos ou egoístas (i.e. agentes que estão unicamente interessados no seu bem pessoal), a coordenação estará centrada em processos adequados de resolução de conflitos, argumentação e negociação. Quando os agentes possuem uma noção de bem social e estão preocupados em atingir objectivos partilhados com outros agentes, ou seja, os agentes são cooperativos, então as metodologias de coordenação estão centradas na cooperação entre agentes e incluem essencialmente formas de resolver problemas globais de forma distribuída, criar equipas de agentes e realizar trabalho de grupo.

1.1.2 Escalonamento e Gestão Universitária

As Instituições de Ensino Superior têm periodicamente de resolver um vasto conjunto de problemas de escalonamento de forma a funcionarem correctamente. Estes problemas incluem a determinação de uma distribuição temporal admissível de recursos a tarefas, com o objectivo de garantir a execução dessas tarefas mas minimizar os custos e maximizar a eficiência da sua execução.

Os principais problemas de escalonamento que surgem no contexto do funcionamento usual de uma Universidade ou outra instituição de Ensino Superior incluem a Distribuição de Serviço Docente (DSD), as diversas variantes da Geração de Horários (GH), a Calendarização de Exames (CE) e a Alocação de Salas (AS). Normalmente, estes

problemas são resolvidos de forma separada ou pelo menos não totalmente interligada. É usual efectuar a distribuição de serviço docente completa e só depois começar a resolução do problema de geração de horários. No entanto, dadas as interligações profundas entre estes problemas, a sua resolução de forma integrada seria muito desejável.

No âmbito de uma Universidade são por vezes resolvidas várias instâncias de problemas semelhantes por órgãos de decisão distintos. Este caso verifica-se quando a Universidade se encontra dividida em faculdades, departamentos ou secções e a resolução dos problemas de DSD, GH ou CE é efectuada de forma distribuída por essas unidades. Esta resolução distribuída coloca problemas consideráveis devido ao facto de as unidades não serem totalmente autónomas, partilhando recursos tais como docentes ou salas. Inclusivamente são usuais as situações em que não existe uma definição clara dos limites das responsabilidades ou recursos alocados a cada departamento ou unidade. A complexidade na resolução distribuída do problema é ainda maior se pensarmos que grande parte dos recursos geridos (i.e. docentes, alunos, etc.), são também em si, unidades com um elevado grau de autonomia e com objectivos e vontades próprias. Isto implica que a resolução automática dos problemas de GH ou DSD seja muito complexa pois é virtualmente impossível especificar todos os objectivos e preferências de todos os recursos envolvidos.

Embora sendo desejável, a resolução distribuída, integrada e automática, dos problemas de escalonamento mencionados, a sua execução nesta forma é complexa. Por um lado, existe a necessidade de utilizar metodologias de resolução de problemas de escalonamento tais como métodos de optimização, meta-heurísticas ou programação lógica com restrições para resolver os problemas em cada unidade. Por outro lado é necessário coordenar a resolução dos problemas das diversas unidades e gerir as interacções entre a resolução de problemas que envolvem os mesmos recursos. Para o realizar é necessário efectuar processos de cooperação e negociação com outras unidades que partilhem os mesmos recursos. Torna-se também necessário gerir a alocação de tarefas a entidades com elevado grau de autonomia e, por vezes, encetar processos de negociação, com e entre essas entidades, de forma a obter uma solução aceitável para todas.

1.1.3 O Futebol Robótico e a Liga de Simulação

O futebol robótico – RoboCup – é um projecto internacional de investigação e educação, com o objectivo de promover a investigação em Inteligência Artificial (Distribuída) e Robótica Inteligente. O projecto baseia-se na utilização de um problema *standard* – o futebol – como forma de estimular a investigação científica em diversas áreas relacionadas com a Inteligência Artificial e Robótica e ao mesmo tempo permitir, todos os anos, comparar de diversos modos a investigação realizada pelos diferentes grupos de investigação participantes. Um elevado conjunto de tecnologias é necessário para ser capaz de construir uma equipa de robôs, reais ou virtuais, que seja capaz de participar num

jogo semelhante ao futebol, utilizando um conjunto de regras pré-especificado. Desta forma, a investigação em futebol robótico e a competição internacional associada – campeonato do mundo de RoboCup – atraíram a participação regular de algumas das melhores empresas, laboratórios de investigação e universidades mundiais, incluindo a Sony, Honda, SGI, Philips, GMD e as universidades de Carnegie Mellon, Cornell, Freiburg, Karlsruhe, UNSW, Tsinghua, entre outros.

O futebol robótico inclui diversas ligas que se dividem em dois tipos: ligas robóticas (robôs pequenos, médios, "cães" e humanóides) e a liga de simulação. Cada liga coloca um conjunto próprio de desafios de investigação e tem ênfase em determinados tópicos necessários para efectivamente colocar equipas de robôs a disputar uma partida de futebol. A liga de simulação é baseada no sistema de simulação "*soccerserver*" que permite a duas equipas de agentes autónomos disputar um jogo de futebol simulado. O "*soccerserver*" providencia um domínio muito realista no sentido em que inclui muitas complexidades do futebol real e dos sistemas robóticos, tais como: erros nos sensores, erros nos actuadores e energia limitada. Os desafios que o domínio do futebol robótico simulado coloca são muito superiores aos desafios colocados por alguns problemas *standard* da Inteligência Artificial, como o xadrez. No futebol robótico o mundo é dinâmico, parcialmente inacessível aos robôs virtuais, contínuo, não determinístico, multi-objectivo, parcialmente cooperativo e parcialmente adverso.

As diversas ligas do RoboCup foram projectadas de forma a contemplar em mundos limitados, um conjunto elevado de complexidades do mundo real, mantendo no entanto o custo, complexidade global e dimensão do problema, acessível aos grupos de investigação em Robótica e Inteligência Artificial. Os problemas de investigação colocados pelo RoboCup de uma forma integrada, cobrem uma vasta área dos domínios da IA e Robótica, incluindo coordenação, cooperação e comunicação multi-agente, arquitecturas de agentes inteligentes, aprendizagem, planeamento em tempo-real, decisão estratégica e táctica, comportamento reactivo, visão, processamento e análise de imagem, controlo, sistemas de locomoção, sistemas sensoriais, fusão sensorial em tempo-real, navegação, controlo robótico, entre outros. O desafio inerente ao RoboCup é simultaneamente estimulante do ponto de vista científico, colocando um vasto conjunto de problemas aos investigadores das diversas áreas científicas mas, simultaneamente, atraente para o público em geral e para os meios de comunicação social[2].

O sucesso numa competição como o RoboCup, para além do desenvolvimento de robôs (reais ou virtuais) com boas qualidades individuais, reside fundamentalmente no desenvolvimento de metodologias de coordenação que lhes permitam trabalhar em

[2] Com esta vertente, pretende-se que o RoboCup seja capaz de chamar a atenção do público e comunicação social, não só para o evento como competição científica, mas sobretudo para a investigação realizada em Inteligência Artificial e Robótica Inteligente pelos diversos laboratórios e empresas participantes.

conjunto de forma harmoniosa no sentido de atingir os seus dois objectivos principais: marcar golos na baliza adversária e evitar sofrer golos na sua baliza. A coordenação é desta forma uma das áreas de investigação essenciais no contexto do RoboCup e um factor crucial para o sucesso de uma equipa nas competições internacionais realizadas neste âmbito.

1.2 Motivação

A motivação principal para a realização deste trabalho resulta da importância crescente do paradigma dos *Agentes* e *Sistemas Multi-Agente (SMA)* e das metodologias de *Coordenação em SMA*. De facto, sob a designação geral de coordenação assentam metodologias como a cooperação, a negociação, a resolução de conflitos e a alocação de recursos em SMA que constituem uma parcela muito relevante da investigação global realizada em Inteligência Artificial Distribuída.

A inexistência de uma abordagem global ao problema da geração de horários e problemas associados, no contexto da gestão universitária, foi outra motivação relevante. Este problema é usualmente analisado de forma isolada, quer a nível de trabalhos de investigação científica realizados sobre o assunto, quer a nível de pacotes de *software* comerciais. Neste trabalho o problema foi analisado como parte integrante de um contexto mais alargado de problemas de escalonamento na gestão universitária, incluindo a distribuição de serviço docente, constituição de turmas, alocação de salas, calendarização de exames, etc. Foi também analisado, utilizando diversas metodologias, desde uma abordagem essencialmente centralizada, em que um agente é responsável pela produção completa das soluções, até uma abordagem completamente distribuída em que metodologias de coordenação em Sistemas Multi-Agente são utilizadas de forma a possibilitar que um conjunto de agentes negoceie e coopere com vista à resolução do problema.

A importância crescente do RoboCup como domínio de teste e problema *standard* para a Inteligência Artificial Distribuída e Robótica Inteligente constitui outra motivação evidente deste trabalho. O xadrez foi durante muitos anos um dos domínios por excelência da aplicação das metodologias de Inteligência Artificial. No entanto, após a vitória da máquina (Deep Blue) sobre o campeão do mundo humano (Gary Kasparov), novos domínios mais estimulantes e complexos tornaram-se necessários. É neste contexto que surge o futebol robótico. Este é um domínio bem mais complexo que o xadrez pois o ambiente é multi-objectivo, dinâmico, contínuo, inacessível e não determinístico. Para além disso o domínio é inerentemente distribuído, descentralizado, parcialmente cooperativo e parcialmente adverso. Os problemas de investigação colocados pelo futebol robótico cobrem uma área muito ampla, mas onde se destacam claramente os problemas da coordenação e cooperação entre agentes. Este domínio foi desta forma, devido às

características enunciadas, seleccionado como principal domínio de teste das metodologias de coordenação de agentes cooperativos desenvolvidas no âmbito desta tese.

Para além da natural motivação que resulta da definição de metodologias de resolução de problemas com que nos deparamos todos os dias na Universidade e da enorme motivação que resulta da participação em grandes competições envolvendo equipas de investigação científica, como o RoboCup, este trabalho teve ainda a motivação adicional de: desenvolver metodologias de coordenação em SMA, suficientemente genéricas para que possam ser aplicadas a curto-prazo em problemas socialmente relevantes envolvendo robôs: a limpeza de minas ou lixo radioactivo, resgate e salvamento em grandes catástrofes, exploração espacial, coordenação de transportes públicos, etc.

1.3 Objectivos

O objectivo deste trabalho consiste em estudar as metodologias de coordenação e cooperação em sistemas multi-agente e propor metodologias adequadas de coordenação para domínios relevantes de SMA. Como domínios de teste foram seleccionados os problemas de escalonamento na gestão universitária e a liga de simulação do futebol robótico.

Os objectivos específicos deste trabalho são os seguintes:

- Estudo dos conceitos de Agente, Sistemas Multi-Agente e de metodologias de coordenação, negociação e cooperação entre agentes inteligentes autónomos;
- Estudo de problemas específicos de escalonamento frequentes na gestão universitária, tais como a distribuição de serviço docente, geração de horários, calendarização de exames, calendarização de reuniões e a alocação de salas;
- Estudo do domínio do futebol robótico nas suas variadas vertentes, incluindo todas as ligas robóticas e desafios associados, mas com ênfase no futebol robótico simulado;
- Definição de linguagens que permitam a coordenação de agentes autónomos através de conhecimento *a priori*, nos domínios da gestão universitária e futebol robótico;
- Definição e implementação de estratégias de coordenação de agentes que possibilitem a resolução de problemas de escalonamento na gestão universitária;
- Definição e implementação de estratégias de coordenação de equipas de agentes que permitam a coordenação eficiente de agentes no futebol robótico;
- Implementação de uma equipa de futebol robótico capaz de participar na liga de simulação do RoboCup;

- Generalização, com a devida limitação, das estratégias de coordenação de forma a serem aplicáveis a outros domínios em que a coordenação de agentes seja essencial.

Para além destes objectivos específicos, este trabalho teve como objectivo demonstrar a validade das metodologias conceptualizadas em competições internacionais, através de comparação directa com trabalhos realizados noutras universidades e institutos de investigação de qualidade internacionalmente reconhecida.

1.4 Contribuições Científicas

É sempre complexo sintetizar as contribuições científicas de uma tese em alguns parágrafos. No entanto é minha convicção que algumas das contribuições científicas deste trabalho são facilmente reconhecidas e vou tentar sumariar nesta secção, aquelas que, na minha opinião, são as principais:

- Síntese e análise crítica das perspectivas dos autores mais relevantes da área sobre os conceitos de Agente, SMA e Coordenação em SMA;
- Definição de uma linguagem – *SCHEDULING UNILANG* – que permite através da especificação completa de problemas de geração de horários, coordenar um conjunto de agentes que representam as entidades numa Universidade através de conhecimento comum *a priori*. A linguagem é capaz de representar problemas globais de geração de horários, incluindo a representação de dados, restrições, medidas de avaliação e soluções deste tipo de problemas. Permite ainda coordenar dinamicamente o conjunto de agentes envolvidos na execução da tarefa cooperativa (geração de horários) no decurso da execução dessa tarefa;
- Definição de uma linguagem – *COACH UNILANG* – que permite a um treinador (artificial) de futebol coordenar a sua equipa através de conhecimento comum *a priori* (estratégia para um dado jogo de futebol simulado). A linguagem permite ainda coordenar dinamicamente a equipa no decurso da execução da sua tarefa cooperativa;
- Extensão do conceito de coordenação de SMA por *organização estrutural a priori* através da definição formal do conceito de estratégia para uma competição (baseada em tácticas, formações, situações e tipos de agentes com comportamentos distintos). Aplicação com sucesso deste conceito na criação de equipas flexíveis no domínio do futebol robótico simulado;
- Definição de metodologias de coordenação para SMA aplicáveis a diversos domínios, nomeadamente a coordenação estratégica, por definição e troca de papéis, por posicionamento estratégico baseado em situações, parcialmente hierárquica, por comunicação avançada, por modelização mútua e por percepção

inteligente.

Para além destas contribuições, o trabalho descrito nesta tese contribuiu cientificamente para o incremento do bom nome da ciência portuguesa através dos bons resultados que obteve nos campeonatos da Europa e do Mundo de Futebol Robótico – RoboCup.

1.5 Estrutura da Tese

Este trabalho encontra-se estruturado em dez capítulos dos quais, o primeiro é composto por esta introdução ao trabalho.

No segundo capítulo é apresentado o conceito de agente e são discutidas as definições apresentadas por diferentes autores para este conceito. São também analisadas as características dos ambientes em que operam os agentes, as áreas científicas que os inspiraram e as classificações propostas por diferentes autores para os diversos tipos de agentes. O capítulo concluiu-se com uma análise detalhada das arquitecturas mais comuns de agentes autónomos e uma breve descrição das áreas de aplicações desta nova tecnologia.

O terceiro capítulo analisa a criação de sociedades de agentes, designadas vulgarmente por Sistemas Multi-Agente. Nesta perspectiva, apresenta a distinção entre os SMA e a Inteligência Artificial Distribuída e descreve a motivação para os SMA. A parte central do capítulo analisa a forma como pode ser efectuada a comunicação a alto-nível entre agentes. É ainda apresentada, de forma resumida, a aprendizagem em SMA, essencialmente no que concerne à sua distinção relativamente à aprendizagem individual. O capítulo conclui-se com a justificação da necessidade de coordenar os agentes constituintes do sistema e com uma introdução às duas principais áreas de investigação em coordenação: coordenação de agentes competitivos (egoístas) e coordenação de agentes cooperativos.

O capítulo quatro analisa a coordenação de agentes competitivos. São introduzidos os conceitos de conflito e processo de resolução de conflitos entre agentes. A negociação como metodologia utilizada por excelência neste tipo de coordenação é analisada com particular detalhe. Desta forma são descritos os princípios da negociação computacional e as formas de projectar protocolos de negociação adequados. É dado algum destaque à descrição do conceito de leilão e dos tipos de leilão existentes. Refere-se ainda a negociação multi-atributo e exemplos de utilização prática de negociação.

No capítulo cinco é analisada a investigação realizada na área da coordenação de agentes cooperativos e formação de equipas de agentes. Inicialmente é apresentada a investigação realizada em resolução distribuída e cooperativa de problemas. Diversos tipos de coordenação propostos na literatura da especialidade são descritos e comparados, incluindo: coordenação por troca de informação e resultados, planeamento multi-agente,

modelização mútua e organização estrutural. São também analisados modelos de coordenação mais específicos propostos por diversos autores. Propõem-se ainda tipos de coordenação distintos dos usuais, incluindo a coordenação por percepção inteligente, controlo parcialmente hierárquico e coordenação estratégica.

O sexto capítulo inclui uma análise dos problemas de escalonamento encontrados na gestão universitária e das metodologias vulgarmente utilizadas para os resolver. São descritos os problemas de *Distribuição de Serviço Docente, Geração de Horários* e *Calendarização de Exames* e outros problemas associados. O capítulo inclui também uma análise comparativa dos principais pacotes de *software* existentes no mercado para construção de horários e das metodologias utilizadas por diferentes investigadores da área para realizar a geração automática de horários e resolução dos problemas associados. São também analisadas e comparadas diversas linguagens e sistemas de representação de problemas de horários propostas por investigadores da área.

Da análise dos problemas de escalonamento apresentada no sexto capítulo resulta a definição de uma linguagem de descrição de problemas globais de escalonamento na gestão universitária – SCHEDULING UNILANG – que é objecto de descrição detalhada no sétimo capítulo. O capítulo inclui ainda a descrição de um SMA para geração de horários – *UNIPS – University Planning and Scheduling System* projectado no âmbito deste trabalho.

O capítulo oito analisa um domínio particularmente atraente e indicado para a aplicação de metodologias de coordenação: o futebol robótico. Na descrição, para além de uma análise da envolvente do futebol robótico e das competições internacionais RoboCup, é dada particular relevância à descrição da liga de simulação e do simulador de futebol robótico – *soccerserver*. Este simulador é descrito com detalhe de forma a facilitar a compreensão do capítulo seguinte da tese. Inclui-se ainda uma descrição mais abrangente da investigação realizada internacionalmente, utilizando o *soccerserver* e discute trabalho relacionado e aplicações práticas da investigação realizada no âmbito do futebol robótico simulado. O capítulo conclui-se com uma análise do RoboCup como competição científica.

O nono capítulo descreve o projecto e implementação de metodologias de coordenação aplicáveis ao futebol robótico simulado, bem como, a outros domínios complexos. Este capítulo contém a maioria das contribuições científicas originais da tese. A descrição é centrada na equipa FC Portugal desenvolvida num projecto associado a esta tese. É descrita a arquitectura dos agentes da equipa, assim como o seu estado no mundo, as suas capacidades individuais e os seus mecanismos de decisão. É também analisada a aplicação de diversos mecanismos de coordenação, incluindo a coordenação estratégica, por comunicação, modelização mútua, percepção inteligente e controlo parcialmente hierárquico. É dado particular destaque à linguagem *COACH UNILANG* que permite a um agente treinador coordenar a alto-nível uma equipa de futebol robótico simulado. Inclui-se

uma extensa análise de resultados obtidos quer em competições oficiais, quer em experiências controladas.

O décimo e último capítulo apresenta as conclusões gerais deste trabalho, analisa os seus principais resultados, originalidades e limitações e propõe algumas perspectivas de desenvolvimentos futuros.

Capítulo 2

2. Agentes Autónomos

A conceptualização, especificação e implementação de Agentes Autónomos é matéria constituinte de uma disciplina derivada essencialmente da Inteligência Artificial cuja investigação científica teve grandes desenvolvimentos nos últimos anos. Os agentes são aplicados nas mais diversas áreas que variam desde a interacção homem-máquina até complexos processos de controlo industrial. Devido ao elevado número de aplicações e à relativa abertura do conceito, existem diversas definições sobre o conceito de *Agente* e não existe um consenso alargado entre os autores da área sobre esta matéria. No entanto, baseando-nos no trabalho de Pattie Maes [Maes, 1996], uma primeira e simples definição de *agente* é: *"Um sistema computacional que habita um dado ambiente, sente e age nesse ambiente, e ao fazê-lo realiza um conjunto de objectivos ou tarefas para o qual foi projectado"*.

O crescimento da investigação realizada no campo dos agentes está ligado ao facto de ser uma convicção geral dos investigadores que os agentes constituem um paradigma de *software* apropriado para desenvolver aplicações para ambientes abertos, heterogéneos e distribuídos como, por exemplo, a Internet. Desta forma, o crescimento do número e dimensão de ambientes com estas características constitui uma forte motivação para a investigação em agentes. A adequação dos agentes a processos de resolução de problemas cuja perspectiva centralizada não demonstra ter capacidade de os resolver satisfatoriamente é outra razão para o crescimento da investigação realizada em agentes. Por outro lado, os sistemas compostos por múltiplos agentes (Sistemas Multi-Agente) constituem uma metáfora natural para perceber, simular ou construir um vasto conjunto de sociedades artificiais.

Neste capítulo é discutido o conceito de agente. Nesta perspectiva, inicialmente é analisada a controvérsia em torno da definição de agente e as principais definições de agente propostas por investigadores da área. São também discutidos os principais atributos que um agente deve possuir tais como a autonomia, reactividade, pró-actividade, habilidade social, mobilidade, comunicação, cooperação e aprendizagem. São ainda analisados os possíveis ambientes em que os agentes se podem incluir e os domínios que inspiraram o aparecimento da investigação na área dos agentes inteligentes. As diferentes

arquitecturas de agentes, propostas na literatura da especialidade, são finalmente analisadas e comparadas entre si.

2.1 Áreas Científicas que Inspiraram os Agentes

A área da investigação designada por Agentes Autónomos surgiu inspirada nas áreas científicas da Inteligência Artificial, Engenharia de Software, Sistemas Distribuídos e Redes de Computadores, Sociologia, Teoria dos Jogos e Economia. A influência destas áreas no campo dos agentes autónomos situa-se aos níveis de (figura 1):

- **Inteligência Artificial** – Micro-aspectos, como a resolução de problemas, raciocínio lógico, representação e utilização de conhecimento, planeamento, aprendizagem, etc.;
- **Engenharia de Software** – O agente como uma abstracção, programação orientada por agentes;
- **Sistemas Distribuídos e Redes de Computadores** – Arquitecturas de agentes, Sistemas Multi-Agente, comunicação e coordenação;
- **Sociologia** – Macro-aspectos como a formação de sociedades virtuais e a interacção entre agentes;
- **Teoria dos Jogos e Economia** – Negociação, resolução de conflitos, mecanismos de mercado.

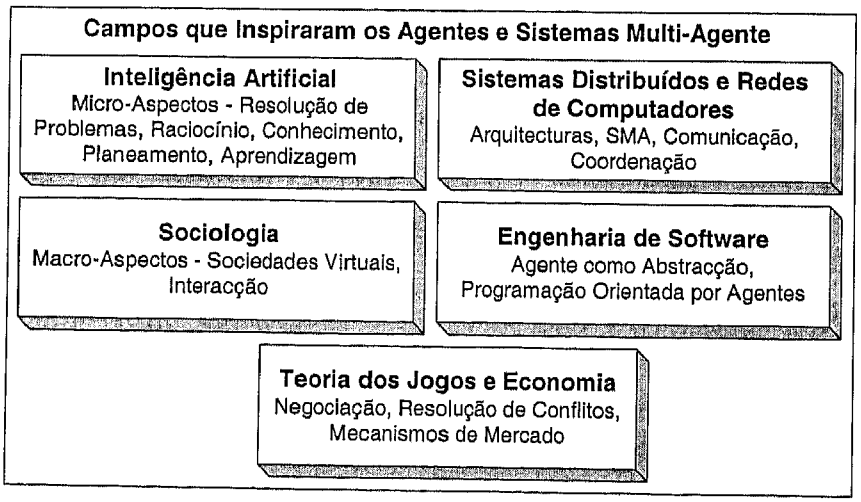

Figura 1: Campos que inspiraram os Agentes e Sistemas Multi-Agente

Embora a Inteligência Artificial tenha exercido inicialmente uma influência muito forte sobre o campo dos Agentes Autónomos e Sistemas Multi-Agente, verifica-se hoje em dia que este campo já evoluiu, muito para além de poder ser considerado uma sub-área da IA.

2.2 Agentes

Ao longo dos últimos anos verificou-se uma grande explosão na investigação na área dos agentes inteligentes ou agentes autónomos. No entanto, a controvérsia que envolve esta área continua a ser elevada. Este facto deve-se essencialmente a:

- Inexistência de um paradigma de programação bem definido para sistemas distribuídos;
- O termo agente ser vulgarmente utilizado para descrever *software* em geral devido às definições vagas e contraditórias de que é objecto;
- O paradigma dos agentes tentar resolver o problema da assunção do "mundo fechado" na orientação a objectos;
- Ao interesse da comunicação social no assunto que resultou na extrapolação da área científica para o público em geral, sem que o seu significado fosse correctamente explicado.

Desta forma o termo agente é por vezes utilizado para descrever coisas com um nível de complexidade muito reduzido tais como um simples termostato (ou outro instrumento de controlo e/ou regulação) ou um *daemon* de *software* (como um servidor de impressão ou um servidor *http*).

Embora não exista uma definição consensual do conceito de agente, existe a noção de que a autonomia é essencial num agente. Baseados nesta noção central de autonomia e nos trabalhos de Wooldridge e Jennings [Wooldridge e Jennings, 1995] [Wooldridge, 2002], Pattie Maes [Maes, 1996] e Russel e Norvig [Russel e Norvig, 1995], iremos adoptar a seguinte definição de agente:

> *"Um Agente é um sistema computacional, situado num dado ambiente, que tem a percepção desse ambiente através de sensores, tem capacidade de decisão, age de forma autónoma nesse ambiente através de actuadores, e possui capacidades de comunicação de alto-nível com outros agentes e/ou humanos, de forma a desempenhar uma dada função para a qual foi projectado." [Reis, 2002b]*

A definição apresentada diz-nos que um agente é um sistema computacional que se encontra situado num dado ambiente. Este ambiente pode ser uma parte do mundo real (universidade, fábrica, hospital, campo de futebol, etc.), um ambiente simulado ou mesmo um computador. Os agentes mais vulgares são os agentes de *software*. No entanto, os agentes podem ter uma existência física (possuindo um corpo), designando-se nesse caso por agentes robóticos.

Independentemente do tipo de agente e ambiente, essencial na definição de agente que adoptamos, é a capacidade do mesmo se aperceber do ambiente e nele agir de forma autónoma. Desta forma, o agente deve possuir sensores e actuadores apropriados ao seu

ambiente e à execução das tarefas para as quais foi projectado. A Figura 2 apresenta um esquema típico de um agente.

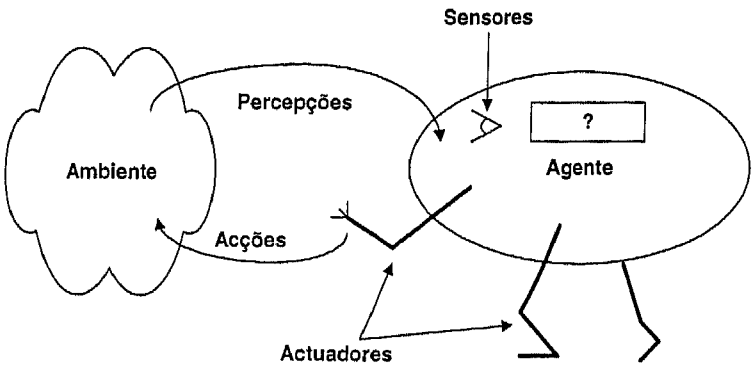

Figura 2: Esquema Típico de um Agente

Um humano apercebe-se do seu ambiente através dos seus olhos, ouvidos, do olfacto, do gosto e do tacto. Age nesse ambiente utilizando os seus actuadores: braços, pernas, cordas vocais, etc. Os sensores de um agente robótico podem incluir câmaras, microfones, sensores de proximidade (incluindo, entre outros, infravermelhos e ultra-sons), sensores de tacto e aceleração, etc. Usualmente os actuadores dos robôs são braços e pernas robóticas, motores e rodas, e altifalantes. Nos agentes de *software* é mais difícil definir o que são os sensores e actuadores do agente. Por exemplo, para um agente que joga uma partida de xadrez, os sensores permitir-lhe-ão determinar a posição das peças no tabuleiro e os actuadores serão capazes de agir no tabuleiro realizando jogadas. A definição exacta da forma de funcionamento dos sensores e actuadores do agente pode, no entanto, apresentar diversas alternativas.

2.2.1 Definições de Agente

Existem diversas definições de agente que, no entanto, são normalmente muito estreitas relativamente ao que definem ou, por oposição, extremamente gerais. Entre as mais conhecidas, encontram-se:

> "... Algo que age ou tem o poder ou autoridade para agir ... ou representar alguém" [AHD, 2000]

> "Um agente é algo que pode ser visto como tendo a percepção de um ambiente através de sensores e agindo nesse ambiente através de actuadores" [Russel e Norvig, 1995]

> "Um agente é uma peça de hardware ou (mais normalmente) um sistema computacional baseado em *software* que goza das seguintes propriedades: autonomia, reactividade, pró-actividade e habilidade social" [Wooldridge e Jennings, 1995]

> "... Sistemas computacionais que vivem em ambientes complexos e dinâmicos,

sentem e agem autonomamente nesse ambiente e ao fazerem-no realizam um conjunto de objectivos e tarefas para o qual foram projectados" [Maes, 1996]

"... Um agente é uma entidade persistente de *software*, dedicada a um objectivo específico. A persistência distingue os agentes de subrotinas pois estes têm as suas próprias ideias sobre como realizar tarefas e têm as suas próprias agendas. O objectivo específico distingue-os de aplicações multifunção; os agentes são tipicamente muito mais pequenos." [Smith et al., 1994]

"Os agentes inteligentes desempenham continuamente 3 funções: percepção das condições dinâmicas do ambiente; acção para afectar as condições desse ambiente; raciocínio para interpretar as percepções, resolver problemas, realizar inferências e determinar acções." [Hayes-Roth, 1995]

"Agentes inteligentes são entidades de *software* que realizam um conjunto de operações em nome de um dado utilizador ou de outro programa com um dado grau de independência ou autonomia, e que fazendo-o, utilizam algum conhecimento ou representações dos objectivos e desejos do utilizador." [IBM, 1997]

"Uma entidade é um agente de *software*, unicamente se for capaz de comunicar correctamente com outros numa linguagem de comunicação de agentes (ACL)" [Genesereth e Ketchpel, 1994]

"Um agente autónomo é um sistema situado e que faz parte de um dado ambiente, que sente esse ambiente e age nele ao longo do tempo, de forma a realizar a sua própria agenda e de forma a afectar o que sentirá no futuro" [Franklin e Graesser, 1996]

"Um Agente é um tuplo $<P,A,I,in,out>$ onde P é um conjunto de objectos de entrada (percepções), A é um conjunto de objectos de saída (acções), I é um conjunto de estados internos, *in* é uma função de $P \times I$ para I (a função de entrada do agente) e *out* é uma função de I para A. O ambiente do agente é um tuplo $<A,P,W,see,do>$ onde A é um conjunto de operações de saída do agente, P é o conjunto de operações de entrada do agente, W é o conjunto de estados possíveis, *see* é uma função de W para P, e *do* é uma função de $A \times W$ para W" [VanLehn, 1991]

Nestas definições destaca-se o facto de os agentes serem capazes de se aperceberem do seu ambiente e agirem nesse mesmo ambiente [Russel e Norvig, 1995] [Maes, 1996] [Hayes-Roth, 1995] [Franklin e Graesser, 1996] [VanLehn, 1991], representarem um dado utilizador ou realizarem um dado conjunto de tarefas [AHD, 2000] [Maes, 1996] [IBM, 1997] [Franklin e Graesser, 1996], serem persistentes [Smith et al., 1994] [Franklin e Graesser, 1996], capazes de comunicar utilizando uma linguagem de alto-nível [Genesereth e Ketchpel, 1994] e autónomos [Maes, 1996] [IBM, 1997] [Franklin e Graesser, 1996].

Uma das definições de agente mais conhecidas e aceites na comunidade é apresentada por Wooldridge e Jennings [Wooldridge e Jennings, 1995] que definem um agente como uma peça de hardware ou (mais usual) um sistema computacional baseado em *software* que goza das seguintes propriedades:

- **Autonomia.** Os agentes operam sem a intervenção directa de humanos ou outros agentes e possuem algum tipo de controlo sobre as suas acções e estado interno;
- **Reactividade.** Os agentes têm a percepção do seu ambiente e respondem rapidamente às alterações que nele ocorrem;
- **Pró-Actividade.** Os agentes não se limitam a agir em resposta ao seu ambiente. Eles são capazes de tomar a iniciativa e exibir comportamento direccionado por objectivos;
- **Habilidade Social.** Os agentes são capazes de interagir com outros agentes (e possivelmente com humanos) através de uma dada linguagem de comunicação de agentes[3].

Estas propriedades são mais complexas de obter do que pode parecer à primeira vista. A autonomia, embora consensual na comunidade dos Agentes e essencial na nossa definição adoptada de agente, nunca pode ser totalmente obtida. É evidente que o agente tem de ser criado e colocado em funcionamento originalmente por um humano (ou por um outro agente). A assunção de que o funcionamento do agente não terá um término também não é totalmente válida. Evidentemente que, à luz da ciência actual, o agente terá um tempo de vida limitado e um final de operação. Por outro lado, embora seja essencial para poder ser considerado um agente, que o seu funcionamento seja efectuado sem operação directa de um humano, usualmente a interacção com o humano é desejável (ou mesmo essencial como no caso da definição apresentada em [IBM, 1997]). Assim, é usual a construção de agentes que embora se possam comportar de forma autónoma, são também capazes de acatar ordens ou instruções de humanos, se tal for desejável.

Construir agentes puramente reactivos é muito simples mas não é desejável[4]. Um agente puramente reactivo, reagiria consecutivamente às mudanças no ambiente sem procurar atingir os seus objectivos de médio ou longo prazo, isto é, de exibir comportamento orientado por objectivos. Interessa pois definir agentes capazes de balancear o comportamento reactivo com o comportamento pró-activo. No entanto, a dificuldade em

---

[3] Estas linguagens são vulgarmente designadas por *ACL – Agent Communication Language*, na literatura anglo-saxónica.

[4] No entanto, alguns autores afirmam que os agentes não necessitam de possuir objectivos explícitos e raciocinar em termos de coordenação pois a racionalidade emerge da interacção entre os agentes reactivos [Brooks, 1991].

balancear estes dois tipos de comportamento é elevada [Hannenbauer et al., 2001] [Reis et al., 2001].

A pró-actividade, isto é, a obtenção de comportamentos orientados por objectivos, é simples de conseguir em sistemas funcionais. No entanto, esta simplicidade só se verifica se for considerado que o ambiente é estático, isto é, não muda durante a execução de um dado procedimento ou função. Para além disso, o agente tem que dispor de toda a informação que necessita para executar esse procedimento ou função, sem incerteza no ambiente. No entanto, como será analisado na secção seguinte, estas assunções não são válidas para a maioria dos ambientes. Para ambientes dinâmicos e não totalmente acessíveis, os agentes têm de ser capazes de reagir às mudanças no ambiente e raciocinar se os objectivos originais ainda são válidos, em face das mudanças no ambiente enquanto executam um dado procedimento. Isto significa que os agentes têm de ser reactivos e consequentemente capazes de reagir rapidamente às mudanças no ambiente.

A capacidade social de um agente está relacionada com a sua capacidade de trocar mensagens de alto-nível (e não só *bytes* de dados sem um significado associado) e efectuar processos de interacção social com outros agentes (e/ou humanos) semelhantes aos utilizados pelos humanos no seu dia a dia. Estes processos incluem a coordenação, cooperação, competição e negociação. De forma a efectuá-los, é necessário raciocinar acerca dos objectivos dos outros agentes (eventualmente humanos) presentes no ambiente ou, pelo menos ter a noção da sua existência. É também necessário compreender que eles são, também, agentes autónomos e que não partilham necessariamente dos nossos objectivos. Desta forma pode ser necessário negociar e cooperar com os outros agentes eventualmente trocando informação e/ou bens. Por exemplo, de forma a convencer um agente a ser cooperativo pode ser necessário efectuar um pagamento ou oferecer um determinado bem ou serviço. Podem, inclusivamente, existir agentes no ambiente com objectivos totalmente opostos[5] e não ser possível, consequentemente, efectuar qualquer processo cooperativo que os inclua.

O balanceamento da capacidade social com as capacidades pró-activas e reactivas é também de enorme importância. Esta importância é ainda maior no caso da realização de trabalho cooperativo por um conjunto de agentes que partilham de um objectivo comum. Neste caso, cada agente necessita de balancear a reacção a eventos no ambiente com a decisão individual necessária para desempenhar tarefas individuais e com o comportamento social necessário para desempenhar tarefas colectivas [Reis et al., 2001].

---

[5] Um exemplo típico consiste no futebol robótico em que os agentes de uma equipa têm como objectivo marcar golos numa baliza e evitar que sejam marcados golos na outra baliza. Os agentes adversários têm exactamente os objectivos opostos. Desta forma é impossível qualquer tipo de actividade cooperativa com esses agentes sendo necessário definir formas de competição entre eles.

Para além das propriedades essenciais mencionadas, Wooldridge e Jennings definem ainda uma *noção forte de agente*, que deriva essencialmente da área da Inteligência Artificial, e envereda por uma visão antropomórfica, onde um agente é visto como uma entidade cognitiva e com consciência, que é capaz de exibir sentimentos, percepções e emoções, à semelhança dos humanos. Assim, nesta noção, as seguintes propriedades podem também ser desejáveis nos agentes [Wooldridge e Jennings, 1995]:

- **Mobilidade.** Capacidade de um agente se movimentar de um local para outro. Usualmente esta capacidade é mencionada no contexto de agentes de *software* e como tal a movimentação verifica-se no interior de uma rede de computadores;
- **Verdade.** Um agente deve sempre ser verdadeiro e não comunicar informação falsa propositadamente;
- **Benevolência.** Os agentes não devem assumir um comportamento contra-produtivo e devem sempre tentar fazer aquilo que lhes é solicitado;
- **Conhecimento e Crença.** Possuir *conhecimento* consiste em, para além de possuir uma colecção de informação dinâmica, possuir também capacidade de raciocínio sobre essa informação. É necessário definir qual a melhor estratégia de raciocínio a aplicar numa determinada situação, e porquê (meta-conhecimento). Uma *crença* representa a noção actual que o agente possui sobre determinado facto. As crenças são geralmente dinâmicas, isto é, podem alterar o seu valor de verdade com o tempo.
- **Intenções e Obrigações.** *Intenções* são objectivos de longo prazo do agente. Resultam em padrões de comportamento que levam à execução de um determinado conjunto de acções individuais. As *obrigações* estão relacionadas com compromissos que o agente assumiu anteriormente. A partir do momento em que o agente expressou a sua disponibilidade para executar determinada tarefa, é responsável por realizar as acções necessárias para essa execução.
- **Racionalidade.** Um agente agirá de forma a atingir os seus objectivos e não agirá de forma a impedir que esses mesmos objectivos sejam atingidos. Em cada instante, em face do seu conhecimento e de acordo com as suas capacidades, tentará executar a melhor acção para cumprir esses objectivos;

Usualmente, para além das propriedades apresentadas, são também referidas como propriedades comuns dos agentes:

- **Inteligência.** O estado de um agente é formalizado por conhecimento (i.e. crenças, objectivos, planos e assunções) e ele interage com outros agentes utilizando uma linguagem simbólica. Possui capacidade de raciocínio abstracto e de resolução de novos problemas e adaptação a novas situações;
- **Continuidade Temporal.** O agente é um processo que é executado continuamente ao longo do tempo. Usualmente são utilizados também os termos

persistente ou "com uma vida longa" para designar a continuidade temporal dos agentes;

- **Carácter.** O agente possui uma personalidade credível e eventualmente possui também um "estado emocional";
- **Capacidade de Aprendizagem.** Capacidades de aprendizagem que fazem com que o agente adquira novo conhecimento e altere o seu comportamento baseado na sua experiência prévia.

É visível que existem diversas definições do conceito de agente nem sempre concordantes. A autonomia é consensual como uma capacidade necessária para um agente. A reactividade, pró-actividade e exibição de comportamento social são também importantes e como tal consideradas na maioria das definições de agente. Outras capacidades como a inteligência e a racionalidade, embora importantes, estão já um pouco contidas nas anteriores. A aprendizagem, embora importante em alguns sistemas, pode não ser desejável noutros. Exemplos disto incluem sistemas em que embora a tarefa a desempenhar seja complexa, é crítico o agente não falhar. Como tal, não é particularmente útil aprender com as suas falhas pois é suposto ele não falhar o que resultaria em consequências muito graves. Por exemplo um agente concebido para efectuar a aterragem de um avião. Neste caso, pretende-se que o agente seja capaz de realizar a sua tarefa de forma autónoma e inteligente mas pode ser argumentado que não será desejável que ele possua aprendizagem pois interessa ser capaz de prever as acções que o agente irá executar na realização desta tarefa. Pelo que se conclui que será bastante mais difícil aos humanos confiarem no agente se este efectuar acções imprevisíveis.

2.2.2 Atributos dos Agentes

Foi discutido anteriormente que um agente pode ser definido a partir de um conjunto de características básicas. O conjunto destas características pode ainda ser utilizado como forma de agrupar os agentes em classes ou tipologias. Deve ser referido que um agente não precisa possuir todas estas características ou atributos, embora as suas capacidades estejam directamente associadas à presença delas. A escolha de quais os atributos que devem estar presentes num dado agente depende da funcionalidade que o projectista pretende dar a esse mesmo agente.

2.2.2.1 Autonomia

A autonomia é provavelmente a característica mais consensual na definição do conceito de agente pelos investigadores da área dos agentes inteligentes. Segundo Nwana [Nwana, 1996], autonomia refere-se ao princípio de que os agentes podem agir baseados nas suas próprias regras de decisão, sem existir a necessidade de serem guiados por humanos. Os agentes possuem estados e metas internas, agindo de forma a atingir estas metas em prol

dos seus utilizadores. Alguns autores defendem que a autonomia aumenta com o aumento da pró-actividade, i.e. da capacidade dos agentes agirem por iniciativa própria, sem a necessidade de agirem em virtude de mudanças no ambiente ou a pedido de um humano ou outro agente [Wooldridge e Jennings, 1994].

Huhns e Singh [Huhns e Singh, 1997a] classificam a autonomia dos agentes em cinco tipos distintos:

- **Autonomia Absoluta.** O agente possui controlo completo sobre as suas percepções, raciocínio e acções, e é muito pouco previsível;
- **Autonomia Social.** Um agente conhece os outros agentes presentes no sistema e é sociável, exercendo no entanto a sua autonomia em certas circunstâncias. A autonomia está em tensão natural com a coordenação ou com outras noções de alto-nível tais como compromissos;
- **Autonomia de Interface.** Em grande parte dos sistemas, onde a autonomia absoluta é impossível, a autonomia máxima possível para o agente é a autonomia respeitante à sua forma de interface com o exterior;
- **Autonomia de Execução.** Liberdade que o agente possui na execução de acções no ambiente;
- **Autonomia de Projecto.** Grau de autonomia dos projectistas do agente na sua construção. Quanto maior for a autonomia de projecto, maior poderá ser a heterogeneidade dos agentes.

De acordo com Foner [Foner, 1993], os agentes detentores de um elevado grau de autonomia podem manter a sua agenda independente das agendas dos seus utilizadores. Para tal, os agentes apresentam três requisitos básicos: capacidade de execução de acções periódicas, execução espontânea e iniciativa própria. Estes requisitos habilitam o agente autónomo a realizar acções preemptivas, independentes, que não são despoletadas por qualquer evento exterior, e que poderão eventualmente beneficiar o seu utilizador.

2.2.2.2 Mobilidade

A mobilidade é a capacidade de um agente se poder movimentar de um local para outro. No caso dos agentes de *software* refere-se à capacidade de estes se poderem movimentar através de uma rede de computadores. Este atributo é particularmente interessante para agentes que auxiliam os seus utilizadores na pesquisa de informações, principalmente na Internet. No entanto, este atributo pode causar sérios problemas de sobrecarga na rede, uma vez que implica a necessidade dos agentes transitarem entre diferentes máquinas através dessa mesma rede. Dependendo do balanceamento entre a dimensão do código do agente e a dimensão dos dados que este troca com o exterior, a mobilidade pode até ser vantajosa em termos de sobrecarga da rede. Suponha-se por exemplo o caso em que o agente é de dimensão reduzida (em termos do seu código) e a sua função consiste em

realizar pesquisas de informação que implicam uma análise de largas quantidades de informação até à selecção da informação mais adequada. Neste caso, seria desejável o agente movimentar-se para junto da fonte dos dados, de forma a que, durante a pesquisa, os dados atravessassem uma reduzida parcela da rede. No regresso, somente necessitaria de trazer o seu código fonte e os dados seleccionados, diminuindo claramente o tráfego total na rede.

Knapik e Johnson [Knapik e Johnson, 1998] apresentam outro problema que está relacionado com a segurança: um agente móvel pode conter defeitos sérios no seu código ou mesmo transportar um vírus. Sendo assim, a implantação de agentes móveis deve ser acompanhada de processos de autorização e validação, além da garantia de que a memória e os recursos da máquina estarão sempre protegidos. Isto implica o desenvolvimento de plataformas especiais para agentes móveis que permitam a navegação controlada destes agentes na rede.

A mobilidade pode também ser considerada numa outra perspectiva, mais abrangente, no caso dos agentes físicos (agentes que possuem um corpo). Neste caso, a mobilidade refere-se à capacidade do agente físico (robô) se deslocar fisicamente no seu ambiente, movimentando-se de uma localização para outra. Esta mobilidade física coloca um vasto conjunto de questões e desafios de investigação, incluindo: integração sensorial, percepção activa, determinação de trajectórias, navegação robusta, metodologias de localização própria e mapeamento do ambiente.

2.2.2.3 Reactividade

A reactividade é a capacidade de um agente reagir rapidamente a mudanças no seu ambiente. Para tal, o agente deve ser capaz de se aperceber do seu ambiente e actuar sobre ele. Este atributo está também presente directa ou implicitamente em quase todas as definições de agente [Wooldridge e Jennings, 1994], [Russel e Norvig, 1995], [Franklin e Graesser, 1996].

O termo reactividade é, no entanto, utilizado na literatura com pelo menos dois significados distintos. O primeiro refere-se à capacidade de um agente reagir às mudanças no seu ambiente rapidamente. O segundo refere-se à capacidade dos agentes decidirem as suas acções sem consultarem um modelo interno do mundo. Wooldridge e Jennings [Wooldridge e Jennings, 1995] usam o primeiro significado de reactividade e referem agentes que se apercebem do seu ambiente e respondem de forma rápida às mudanças que ocorrem nesse ambiente. Russel e Norvig [Russel e Norvig, 1995], por oposição, referem-se a reactividade quando os agentes utilizam planeamento reactivo e regras de condição-acção para definir o seu comportamento.

Embora a reactividade (no sentido de reacção atempada às mudanças do ambiente) seja desejável, um agente puramente reactivo, reage consecutivamente às mudanças no ambiente sem procurar atingir os seus objectivos de médio ou longo prazo. Se estes

objectivos não forem triviais, ou seja a tarefa a desempenhar pelo agente for complexa, será muito difícil a um agente reactivo realizá-la. De forma a que o agente possa exibir um comportamento orientado por objectivos interessa definir metodologias que permitam balancear o comportamento reactivo com as restantes capacidades do agente.

O interesse pela reactividade e pelos agentes reactivos levou ao aparecimento de uma tendência de investigação designada por "escola reactiva" [Ferber, 1999]. Os investigadores desta "escola" afirmam que não é necessário que os agentes individuais sejam deliberativos e possuam inteligência para que um Sistema Multi-Agente demonstre inteligência como um todo [Brooks, 1991] [Ferber e Drogoul, 1992] [Ferber, 1999]. Exemplos de sociedades biológicas tais como colónias de formigas foram utilizados de forma a provar que agentes reactivos muito simples podem, em conjunto, comportar-se de forma inteligente, resolvendo tarefas que podem ser classificadas como complexas.

2.2.2.4 Pró-Actividade

Este atributo pode ser também denominado de capacidade de iniciativa, uma vez que representa um comportamento independente. As acções são seleccionadas de acordo com os objectivos gerais do agente e não simplesmente devido a mudanças que ocorrem no ambiente do agente. A pró-actividade é essencial para que os agentes possam exibir comportamento orientado por objectivos, os quais podem ser gerados internamente.

2.2.2.5 Comunicação

Quando existe mais do que um agente presente no ambiente, há uma necessidade óbvia de dispor de capacidade de comunicação entre estes. Entretanto, o conceito de capacidade de comunicação não se restringe apenas à capacidade de troca de informações entre agentes. Franklin e Graesser [Franklin e Graesser, 1996] afirmam que agentes podem comunicar com outras entidades além dos agentes, incluindo humanos e o seu próprio ambiente.

De forma a dispor de capacidade de comunicação com o exterior, os agentes necessitam de dispor de actuadores apropriados ao envio dessas mensagens (placa de rede, colunas de som, etc.) e conhecimento que lhes permita utilizar protocolos de transporte dessas mensagens. Por outro lado, e a um nível de análise mais elevado, os agentes necessitam de dispor de uma linguagem que possa também ser entendida pelos outros agentes presentes no ambiente. Precisam ainda de partilhar a semântica do que é veiculado.

A disponibilidade de actuadores apropriados, protocolos e linguagem para efectuar a comunicação é essencial para dispor de outras capacidades mais elaboradas, construídas com base na capacidade de comunicação, como sejam a habilidade social e a capacidade de cooperação. Este tema será objecto de uma análise mais aprofundada na secção 9.7, em que se analisa a forma de definição de procolos de comunicação que facilitem a cooperação de agentes.

2.2.2.6 Habilidade Social

A habilidade social está directamente associada com a característica de capacidade de comunicação, uma vez que representa a capacidade de interagir com outros agentes. Genesereth e Ketchpel [Genesereth e Ketchpel, 1994] afirmam que um agente tem de ser capaz de comunicar numa linguagem de alto-nível, e indicam a necessidade da existência de uma linguagem comum para a comunicação de agentes, de forma a que estes possam trocar informação. No entanto, embora a utilidade da existência desta linguagem seja reconhecida por todos os investigadores da área, o problema é definir qual será a linguagem ideal. Actualmente, não existe uma linguagem "standard" aceite mundialmente, para a comunicação entre agentes.

A habilidade social implica ainda mais do que a existência de uma linguagem comum. Os agentes devem ser capazes de compartilhar a semântica das mensagens e utilizar o mesmo vocabulário (ontologia) de acordo com a aplicação em causa [Uschold e Grüninger, 1996]. Por exemplo, um "rato" terá um significado completamente diferente no contexto de uma loja de computadores ou de um jardim zoológico. Desta forma é necessário que ambos os agentes interpretem o conceito da mesma forma, ou seja, utilizem a mesma ontologia no decurso da sua comunicação.

2.2.2.7 Cooperação

Cooperação pode ser entendida como a capacidade que os agentes têm de trabalhar em conjunto de forma a concluírem tarefas de interesse comum. Nwana [Nwana, 1996] acredita que a cooperação entre agentes é fundamental, sendo a razão principal para a existência de um ambiente multi-agente. Tal como os humanos, os agentes têm de combinar os seus esforços de forma a atingir objectivos comuns que não podem ser atingidos individualmente.

Para permitir esta cooperação, o agente deve ser dotado de uma certa habilidade social, capacitando-o a interagir com outros agentes e possivelmente humanos, através de alguma linguagem de comunicação [Wooldridge e Jennings, 1994]. Deve ainda dispor de metodologias apropriadas para realizar essa cooperação, de acordo com a tarefa cooperativa que irá ser executada pelo conjunto de agentes. As estratégias propostas na literatura para efectuar a cooperação entre agentes incluem estratégias fixas, aprendizagem de trabalho de equipa, organizações hierárquicas, normas e leis sociais, modelização mútua, etc. Usualmente a comunicação é vital para a realização da cooperação, embora em alguns casos seja possível a cooperação sem comunicação [Genesereth et al., 1986]. Esta comunicação é usualmente realizada a um nível abstracto e elevado, utilizando linguagens e protocolos apropriados para a comunicação inter-agente.

2.2.2.8 Aprendizagem

Uma característica que é frequentemente confundida com a inteligência é a capacidade de aprender. Um agente só pode possuir uma autonomia completa quando possuir a capacidade de avaliar as variações de seu ambiente externo, e propor a acção mais correcta. No mundo real, o ambiente muda e não se pode esperar que, se o agente possuir continuidade temporal, o ambiente que ele vá encontrar dentro de alguns anos seja semelhante ao ambiente para o qual foi inicialmente projectado. Acresce a este facto que será muito difícil ao projectista prever, na fase de projecto do agente, a variação futura do ambiente onde este irá operar. Assim, mesmo quando um agente não reconhece a existência de uma acção com elevada probabilidade de sucesso para ser executada, é esperado que ele procure, encontre e experimente alternativas. A questão não é acertar sempre, mas aprender continuamente por experiência, seja através de sucessos ou de fracassos. No entanto, a inclusão desta capacidade de aprendizagem não é trivial em agentes. Por vezes é difícil identificar a falha ou sucesso da acção e mais complexo ainda identificar o motivo que levou a essa falha ou sucesso. Acresce a este facto que, mesmo após uma correcta identificação dos motivos de uma dada falha, por vezes é também difícil utilizar esta informação de forma a encontrar planos alternativos com maior probabilidade de sucesso.

A aprendizagem pode ser também um processo interactivo em que um treinador ensina um dado agente (ou conjunto de agentes) a executar uma determinada tarefa individual ou cooperativa. Nestes casos, o treinador pode fornecer ao agente o conhecimento, através de uma sequência de instruções ou informá-lo apenas quando o agente não possuir o conhecimento necessário para executar a sua tarefa [Lemon et al., 1996]. O treinador pode ainda possuir uma visão mais geral sobre a tarefa, e capacidade de percepção e conhecimento específico que lhe permita detectar situações de falha ou sucesso com maior facilidade do que o agente que executa a tarefa. No caso da aprendizagem multi-agente, os agentes aprendem a realizar uma tarefa que envolve mais do que um agente na sua execução. Neste caso, a figura do treinador é novamente bastante importante pois uma visão global da situação permite detectar falhas e construir planos de execução da tarefa com maior facilidade.

2.2.3 Agentes vs. Objectos

Um dos grandes desafios que se coloca à comunidade de investigadores em Agentes Inteligentes consiste em estabelecer claramente a distinção entre os conceitos de Agente e Objecto, mostrando a inovação que existe por detrás do conceito de Agente [Wooldridge, 2002].

O conceito de Objecto para a comunidade de Programação Orientada por Objectos, consiste numa entidade computacional que encapsula um estado e é capaz de executar um conjunto de acções (métodos) sobre este estado e comunicar através de mensagens.

Embora as semelhanças entre os conceitos sejam evidentes, as diferenças são também visíveis:

- Os Agentes possuem capacidade de decisão autónoma;
- Os Agentes são capazes de comportamento reactivo, pró-activo e social;
- Os Agentes possuem um fluxo de controlo próprio e distinto dos restantes agentes que compõe um dado Sistema Multi-Agente.

Embora os objectos possuam autonomia sobre o seu estado (dados) não possuem autonomia sobre o seu comportamento, ou seja, se disponibilizarem um determinado método, sempre que outro objecto o invoque este estará disponível. Pelo contrário, os agentes possuem controlo sobre o seu comportamento e, como tal, um outro agente terá de solicitar ao agente que execute uma dada acção. Este pode decidir, em cada situação concreta, efectuar ou não a acção solicitada. No modelo orientado a objectos a decisão é efectuada pelo objecto que solicita a execução do método. Esta distinção entre objectos e agentes é vulgarmente sumariada pelo seguinte slogan:

"Objects do it for free; agents do it for money"[6]

Relativamente à pró-actividade, reactividade e comportamento social, o modelo orientado a objectos não contém qualquer referência a estes tipos de comportamentos.

A distinção relativa ao fluxo de controlo próprio baseia-se essencialmente no fluxo de controlo único do modelo orientado a objectos "standard". Neste modelo o fluxo único de controlo invalida a noção de entidade autónoma. No entanto, os recentes objectos activos [Booch, 1994], diminuem esta distinção. Os objectos activos são supostos terem o seu fluxo de controlo próprio sem necessariamente terem a capacidade de exibir um comportamento autónomo.

Embora existam distinções importantes entre agentes e objectos, é claro que o modelo orientado a objectos é bastante adequado à implementação de agentes. Desta forma, linguagens tais como Java [Deitel e Deitel, 1999], o C++ [Bronson, 1997] e suas extensões, encontram-se entre as mais utilizadas na implementação de agentes autónomos. Numa extensão a este raciocínio [Shoham, 1993] os agentes podem ser vistos como um novo paradigma de programação: a Programação Orientada por Agentes.

Ao longo da história da *Engenharia de Software* as novas metodologias têm vindo a surgir no sentido de incrementar a localização e encapsulamento das unidades básicas de programação. O surgimento do paradigma da Programação Orientada a Agentes confirma

---

[6] Este slogan pode ser traduzido para Português como: "Os objectos fazem-no de graça; os agentes fazem-no por dinheiro.". Outra versão do slogan [Wooldridge 2002] é: "Objects do it for free; agents do it because they want to", ou seja, "Os objectos fazem-no de graça; os agentes fazem-no porque querem."

esta tendência. A tabela 1 demonstra os diversos paradigmas de programação e suas características principais, ao longo dos anos.

	Programação Monolítica	*Programação Estruturada*	*Programação Orientada a Objectos*	*Programação Orientada a Agentes*
Comportamento das Unidades	Externa	Local	Local	Local
Estado das Unidades	Externa	Externa	Local	Local
Invocação/Comportamento das Unidades	Externa	Externa (chamada)	Externa (mensagem)	Local (regras e objectivos)

Tabela 1: Evolução dos Paradigmas de Programação.

A tendência verificada na evolução da engenharia de *software* (tabela 1) vai no sentido de um cada vez maior encapsulamento. As unidades de programação têm-se tornado cada vez mais autónomas e a programação orientada por agentes incrementa ainda mais esta autonomia dando às unidades de programação, agora designadas por agentes, controlo sobre o seu próprio comportamento.

2.3 Ambientes

Um agente é um sistema computacional que "vive" num determinado ambiente, apercebendo-se e agindo nesse ambiente. As características do ambiente em que o agente "habita" são determinantes na definição da arquitectura do agente e da sua forma de operação. Antes de projectar um agente é necessário uma análise cuidada das percepções e acções possíveis e da complexidade da tarefa a executar pelo agente. Neste contexto a distinção entre ambientes reais e simulados não implica que um ambiente real seja mais complexo do que um ambiente simulado. Por vezes, agentes de *software* construídos para ambientes simulados possuem um conjunto possível de percepções e acções mais rico do que agentes robóticos construídos para operar em determinado ambiente real.

Seguindo uma classificação baseada no trabalho de Russel e Norvig [Russel e Norvig 1995], podemos classificar as propriedades dos ambientes segundo as seguintes classes:

- **Acessível vs. Inacessível.** Um ambiente acessível é aquele onde um agente consegue obter, através dos seus sensores, informação actualizada, precisa e completa sobre o ambiente. Grande parte dos ambientes típicos não são acessíveis neste sentido. Entre os ambientes inacessíveis destacam-se a Internet e todos os ambientes físicos reais.
- **Determinístico vs. Não determinístico.** Num ambiente determinístico cada acção tem um efeito único garantido, não existindo qualquer incerteza quanto ao

resultado da sua execução.

- **Estático vs. Dinâmico.** Um ambiente estático é suposto permanecer inalterado enquanto o agente decide a próxima acção a executar. Em contraste, num ambiente dinâmico outros agentes encontram-se a agir no ambiente ao mesmo tempo. Todos os ambientes físicos do mundo real e a Internet são dinâmicos neste sentido.
- **Discreto vs. Contínuo.** Um ambiente diz-se discreto se existe um número finito de percepções e acções possíveis para o agente e contínuo caso contrário. Um ambiente pode ser contínuo no que diz respeito às percepções do agente e discreto no que diz respeito às acções (e vice versa).

Quanto mais acessível um ambiente for, mais simples será projectar e construir agentes para nele operarem[7]. A qualidade das decisões efectuadas por um agente está obviamente ligada à quantidade e qualidade da informação que este possui, sendo desta forma muito mais simples tomar as decisões correctas em ambientes acessíveis. É importante também referir que a escala de acessibilidade não pode ser considerada uma escala discreta (acessível / não acessível). Em ambientes não acessíveis é importante considerar o grau de acessibilidade do ambiente ao agente. Por exemplo, para um agente robótico colocado num ambiente físico, será muito mais simples tomar decisões correctas se possuir câmaras, sensores de infravermelhos e outros equipamentos sensoriais avançados, de forma a ter uma razoável acessibilidade ao estado do seu mundo. Neste contexto é também importante considerar a precisão da informação recolhida pelos sensores do agente. Os sensores do agente podem ter acesso à grande maioria da informação necessária para a operação desse agente, mas não serem capazes de obter essa informação de forma precisa. A existência de erros na percepção do agente torna as suas tarefas bastante mais complexas de executar.

O não determinismo de um ambiente é uma grande fonte de complexidade para o mesmo, impondo sérias restrições à capacidade de um agente para efectuar previsões quanto ao estado futuro do mundo. Esta característica captura no entanto características importantes do mundo real, como sejam a possibilidade de as acções executadas por um dado agente falharem ou não terem um efeito único, bem determinado e garantido à partida. Desta forma é muito mais complexo construir agentes para ambientes não determinísticos, pois é necessário providenciar capacidades de recuperação e tolerância a falhas. No entanto, mesmo ambientes determinísticos tais como os jogos de tabuleiro[8] podem ser complexos.

---

[7] É no entanto de referir que, mesmo num ambiente acessível, se a tarefa a realizar pelo agente for complexa, não será trivial projectá-lo e construí-lo.

[8] Exemplos típicos são os jogos como o Xadrez e o Go. Para este último jogo, devido à enorme quantidade de acções possíveis em cada instante (cerca de 200) e à elevada complexidade envolvida no raciocínio

O dinamismo de um ambiente é uma das fontes mais significativas de complexidade. Grande parte da investigação realizada no início da inteligência artificial, nomeadamente a realizada no âmbito do desenvolvimento de algoritmos de planeamento, estava preocupada em determinar a sequência de acções que permitia a um dado agente, partindo de um estado inicial especificado, atingir um dado estado objectivo também especificado [Allen et al. 1990]. O desenvolvimento deste tipo de algoritmos assumia implicitamente que o ambiente era estático, ou seja, que não seria alterado por qualquer outro agente no decurso da execução do plano [Wooldridge 2002]. No entanto, claramente que os ambientes físicos do mundo real e grande parte dos ambientes computacionais não são estáticos e este tipo de planeamento rígido não pode neles ser aplicado.

Em ambientes dinâmicos, os agentes não podem assumir que se não executarem alguma acção durante um dado intervalo de tempo, o ambiente vai permanecer inalterado. Acresce a este facto que os agentes têm inclusivamente de entrar em consideração com o tempo de raciocínio na decisão das acções a executar. O efeito de uma dada acção no instante t_i+t_a poderá ser totalmente diferente do efeito da mesma acção no instante t_i+t_b. Desta forma, supondo que t_i é o instante de tempo actual, processos de raciocínio que tenham como resultado a execução da mesma acção mas que demorem respectivamente t_a e t_b poderão conduzir a resultados totalmente distintos.

O dinamismo implica também que os agentes têm de efectuar acções apropriadas para recolherem informação do ambiente [Moore, 1990]. Neste contexto é importante o agente possuir sensores flexíveis (como por exemplo câmaras direccionáveis). Outra característica dos ambientes dinâmicos é a presença de outros agentes que agem no ambiente e que podem interferir (de forma positiva ou negativa) nas acções efectuadas pelo agente. Aqui, o agente deixa de estar isolado e surge a necessidade de se coordenar, comunicar, cooperar, competir ou negociar com estes agentes adicionais. Um ambiente estático é inerentemente muito mais simples, pois o agente pode recolher informação sobre uma dada característica do ambiente unicamente uma vez e assumir que este ambiente não será modificado, a não ser pelas suas próprias acções. Nos ambientes estáticos, os agentes não têm também necessidade de se sincronizarem ou coordenarem as suas acções com as de outros agentes no ambiente [Bond e Gasser, 1988b] nem de interagirem de qualquer outra forma com outros agentes.

A última característica dos ambientes diz respeito a estes serem discretos ou contínuos. Um ambiente discreto é aquele que só pode assumir um conjunto predefinido de estados discretos. Por oposição, num ambiente contínuo é impossível enumerar todos os estados possíveis. Desta forma, o Xadrez e outros jogos de tabuleiro, constituem ambientes discretos pois existe um número finito (embora muito grande) de estados possíveis do

---

necessário para o jogar, não foi ainda possível construir um agente capaz de vencer os melhores jogadores humanos mundiais.

tabuleiro. Ambientes físicos do mundo real são por natureza intrinsecamente contínuos. A continuidade de um ambiente pode-se verificar a três níveis:

- **Estado do Mundo (Ambiente).** O estado do mundo pode ser discreto ou contínuo. Por exemplo, no caso de um ambiente para um jogo de tabuleiro, como o Xadrez ou as Damas, o ambiente será discreto. Em diversos ambientes simulados, o estado do mundo é também discreto, no sentido em que é representado por variáveis discretas e existe um número finito de estados possíveis para esse ambiente;
- **Percepções do Agente.** Embora um dado ambiente possa ser contínuo, as percepções de um agente inserido nesse ambiente podem ser discretas. Este é o caso, por exemplo, de um robô com sensores *booleanos* de proximidade que se encontra num ambiente físico;
- **Acções do Agente.** Tal como no caso das percepções, as capacidades de acção de um agente que se encontra num dado ambiente contínuo podem ser unicamente discretas, i.e. o agente dispor de um conjunto limitado de acções.

Os ambientes discretos são bastante mais simples para os agentes por diversos motivos. O motivo principal está directamente relacionado com o facto dos sistemas computacionais serem por natureza discretos e, como tal, para tratarem ambientes contínuos, terem de efectuar algum tipo de discretização e aproximação. É este o caso, por exemplo, na percepção através de uma câmara, em que é necessário efectuar a digitalização da imagem previamente ao seu processamento através do sistema computacional. Embora a precisão na aproximação digital possa ser elevada (estando limitada unicamente pela capacidade do hardware do sistema computacional), em qualquer tipo de digitalização, existe sempre informação que é perdida ou impossível de ser tratada em tempo útil. Desta forma, num ambiente contínuo os agentes irão sempre raciocinar com informação aproximada. Acresce a este facto a complexidade da dimensão do espaço-de-estados infinito do ambiente[9].

Analisando as propriedades dos ambientes é fácil concluir que os ambientes mais complexos para os agentes são os ambientes inacessíveis, não determinísticos, dinâmicos e contínuos. Ambientes com estas propriedades são por vezes designados por ambientes abertos [Hewitt, 1986]. Embora estes ambientes sejam os mais complexos, verifica-se que os ambientes físicos do mundo real são intrinsecamente abertos. Como tal, e numa perspectiva de futuro desenvolvimento de agentes para operarem no mundo real, interessa ser capaz de definir metodologias que permitam o desenvolvimento de agentes para

[9] Embora um ambiente discreto possa ter um número muito elevado e virtualmente infinito de estados possíveis.

operarem em domínios que têm a maioria ou mesmo todas as características dos ambientes abertos.

2.4 Classificação dos Agentes

Devido ao largo espectro de aplicações de agentes, à dificuldade em definir o que é realmente um "agente inteligente" e à enorme dinâmica que esta área de investigação tem tido ao longo dos últimos anos, existem diversos sinónimos do termo "agente inteligente" criados por diferentes investigadores na tentativa de melhor caracterizar o seu próprio trabalho. Assim, é vulgar encontrar na literatura da especialidade designações de agente tais como: robôs, agentes de *software* (ou *softbots*), *knowbots*, *taskbots* ou *userbots*, assistentes pessoais, personagens virtuais, etc. Embora o aparecimento destes sinónimos seja perfeitamente compreensível, por vezes obscurece o conceito tornando mais difícil analisar o que é a investigação em agentes.

De forma a caracterizar melhor a área científica dos agentes, para além de analisar os domínios que inspiraram a área, é útil dividir os agentes em classes, analisando as diversas tipologias de agentes propostas na literatura. O elevado número de atributos discutido nas secções anteriores, permite perceber como é difícil implementar um agente que incorpore todos aqueles atributos. Este facto está também relacionado com o facto das características de um agente serem idealmente dependentes do tipo de aplicação a que se propõe. A análise dos atributos que estão presentes nos agentes tem sido utilizada pelos investigadores da área para classificar os agentes em tipologias. Uma tipologia é basicamente uma classificação por tipos de agentes que possuem atributos em comum.

Nwana [Nwana, 1996] propõe uma tipologia de agentes que identifica sete dimensões de classificação distintas:

- **Mobilidade.** Agentes estáticos ou móveis. Os agentes móveis, podem estar residentes na sua máquina de origem ou temporariamente numa outra máquina;
- **Modelo de Raciocínio.** Presença ou não de um modelo de raciocínio simbólico, ou seja, um agente pode ser deliberativo ou puramente reactivo;
- **Função do Agente.** A função principal assumida pelo agente, como por exemplo um agente de pesquisa de informação (que procura informação adequada a um dado utilizador na Internet) ou de interface (que facilita a interacção homem-máquina de uma determinada aplicação);
- **Autonomia.** Grau de autonomia do agente;
- **Cooperação.** Realização ou não de acções cooperativas com outros agentes;
- **Aprendizagem.** Inclusão ou não de capacidades de aprendizagem no agente;
- **Características Híbridas.** Estas combinam duas ou mais filosofias diferentes

num mesmo agente.

Combinando as características de autonomia, cooperação e aprendizagem, Nwana [Nwana, 1996] deriva quatro tipos de agentes principais: agentes colaborativos, agentes colaborativos com capacidade de aprendizagem, agentes de interface e agentes verdadeiramente inteligentes (figura 3). É importante notar que os limites desta classificação não devem ser interpretados como linhas rígidas bem definidas. Na verdade, o facto de agentes colaborativos terem mais ênfase na cooperação e autonomia que os agentes com capacidade de aprendizagem, evidentemente não exclui a possibilidade de os primeiros desenvolverem capacidades de aprendizagem.

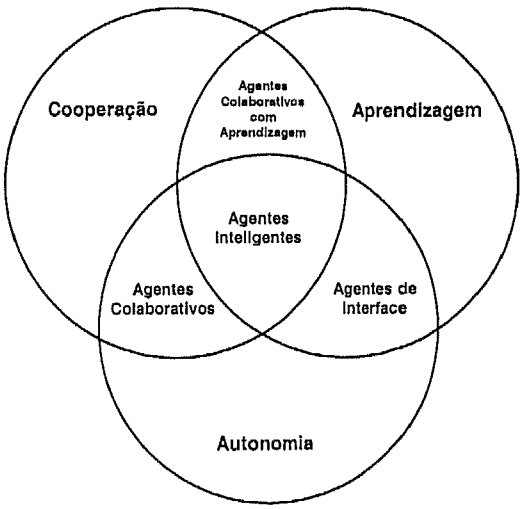

Figura 3: Categorias de Agentes definidas por Nwana.

Após estabelecer a sua tipologia, Nwana estabeleceu 7 categorias de agentes de acordo com a sua arquitectura e função: agentes colaborativos, agentes de interface, agentes móveis, agentes de informação, agentes reactivos, agentes híbridos e agentes inteligentes.

Franklin e Graesser [Franklin e Graesser ,1996] acreditam que um agente, por definição, deve ser um processo em contínua execução e apresentaram a taxonomia representada esquematicamente na figura 4, que divide os agentes autónomos em três grupos principais: agentes biológicos, agentes robóticos e agentes computacionais.

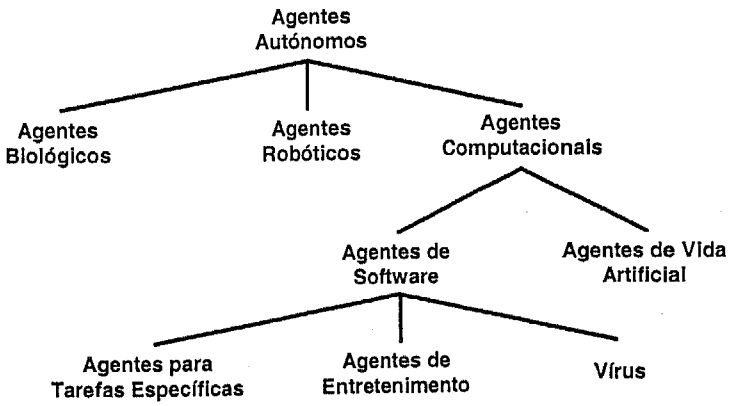

Figura 4: Categorias de Agentes definidas por Franklin e Graesser.

Outras classificações de agentes baseiam-se essencialmente nas aplicações dos agentes inteligentes. Estas incluem, entre outras, a pesquisa de informação, assistentes pessoais, gestão do correio electrónico, controle de energia eléctrica, gestão de redes de telecomunicações, gestão do trânsito, exploração submarina, controle de veículos e naves espaciais, manufactura integrada por computador, gestão de tráfego aéreo, gestão de transportes, negociação, comércio electrónico, transacções financeiras e bolsas de valores, formação de equipas, jogos, entretenimento e personagens virtuais.

2.5 Arquitecturas de Agentes

Uma arquitectura de *software* pode ser descrita como sendo a configuração dos componentes que constituem um sistema e das conexões que coordenam as actividades entre estes componentes [Abowd et al., 1996]. Quando se fala de arquitecturas de agentes podemo-nos referir não só à arquitectura interna de cada agente mas também à arquitectura do próprio sistema multi-agente, ou seja ao modo de organização dos agentes dentro de um sistema e como estão estruturados os seus relacionamentos e interacções.

Assim como existem diversas arquitecturas de *software*, o mesmo ocorre em relação às arquitecturas de agentes (de *software*), as quais possuem certas características que permitem a avaliação da sua qualidade e eficácia. Relativamente aos agentes robóticos, a sua arquitectura compreende não só a arquitectura de *software* como também a sua arquitectura de hardware (componentes físicos e sua interligação).

Wooldridge e Jennings [Wooldridge e Jennings, 1994] afirmam que a arquitectura de um agente pode ser estruturada através de uma metodologia específica para definir agentes, abrangendo técnicas e algoritmos para suportar essa mesma metodologia. Knapik e Johnson [Knapik e Johnson, 1998] destacam que a discussão sobre a qualidade de uma arquitectura de agentes é extremamente subjectiva, pois os detalhes desta discussão estão intimamente relacionados com aspectos específicos da aplicação de agentes que se pretende desenvolver. Uma boa arquitectura de agentes para um dado domínio poderá ser completamente desadequada para outro domínio. No entanto, Mowbray [Mowbray, 1995] estabelece alguns conceitos que podem ser úteis para o desenvolvimento de uma arquitectura promissora:

- **Simplicidade.** Idealizar a arquitectura e seus componentes de forma que sejam fáceis de entender, implementar e manter.
- **Funcionalidade.** Seleccionar uma arquitectura e ferramentas de desenvolvimento que focalizem os aspectos específicos do problema a ser abordado.
- **Expansibilidade.** A arquitectura deve poder ser facilmente ampliada, uma vez que nem todas as necessidades futuras podem ser previstas inicialmente.

- **Isolamento ou Portabilidade.** Uma arquitectura, para poder ser expandida, deve possuir uma implementação portável, evitando-se soluções não padronizadas.

Uma classificação muito simples para as arquitecturas de agentes foi proposta por Russel e Norvig [Russel e Norvig, 1995]. Esta classificação divide os agentes nas seguintes classes:

- **Agentes Reflexos Simples**: Utilizam um conjunto de regras condição-acção preestabelecido. Perante a percepção de uma dada situação, seleccionam e executam uma dada acção pré-especificada. Os agentes deste tipo podem ser considerados os mais simples pois o seu comportamento é puramente reactivo;
- **Agentes com Representação do Estado do Mundo**: Estes agentes possuem uma representação do estado do mundo que é actualizada dinamicamente com a percepção do agente. Desta forma, a reactividade destes agentes é condicionada pelas experiências anteriores que se encontra reflectida nesse estado do mundo. Este modo de funcionamento implica que a mesma percepção ocorrida em momentos distintos e consequentemente estados diferentes do mundo, tenha como resultado acções diferentes;
- **Agentes Baseados em Objectivos**: Os agentes baseados em objectivos, para além da descrição do estado corrente do mundo, utilizam a informação sobre os objectivos que pretendem atingir. Neste caso, a decisão pode implicar a pesquisa e planeamento prévio, antes da selecção da acção a executar em cada instante. Este tipo de agentes é mais flexível pois diferentes comportamentos podem ser obtidos para o mesmo estado do mundo, dependendo dos objectivos do agente;
- **Agentes Baseados em Utilidade**: A utilidade corresponde a uma medida de satisfação do agente relativa ao cumprimento dos seus objectivos. A utilidade será mais elevada se o estado do mundo actual do agente estiver próximo de atingir os seus objectivos. As utilidades podem ser usadas para decidir entre objectivos em conflito ou ainda, quando existe incerteza nas acções, para medir a proximidade no alcance dos objectivos. Os agentes baseados em utilidade conseguem ser mais racionais pois possuem capacidades para avaliar a utilidade da execução de uma determinada acção;

A descrição que apresentam Russel e Norvig [Russel e Norvig, 1995], baseia-se na crescente complexidade da estrutura interna de cada agente. Um agente que se baseia, sucessivamente, em mais factores e capacidades para tomar uma decisão e/ou aprende com as suas interacções com o mundo exterior, pode tornar-se mais eficaz na persecução dos seus objectivos. No entanto, a divisão apresentada baseia-se essencialmente no tipo de raciocínio efectuado pelos agentes e considera unicamente agentes com arquitecturas muito simples. No seu trabalho, Russel e Norvig [Russel e Norvig, 1995] apresentam também outras arquitecturas de agentes mais avançadas em que se incluem: agentes com capacidades de aprendizagem, agentes de decisão e agentes com capacidades de planeamento.

Em oposição, Wooldridge e Jennings [Wooldridge e Jennings, 1994] apresentam uma descrição mais geral das arquitecturas de agentes. Baseiam-se na forma de construção dos agentes envolvidos para dividir as arquitecturas dos agentes computacionais em três grandes categorias:

- **Arquitecturas Deliberativas**: Seguem a abordagem clássica da Inteligência Artificial, onde os agentes actuam com pouca autonomia e possuem modelos simbólicos explícitos dos seus ambientes.
- **Arquitecturas Reactivas**: Procuram não utilizar algum tipo de modelo ou raciocínio simbólico complexo e tomar decisões "em tempo real". Correspondem aos agentes reflexos simples na classificação de Russel e Norvig [Russel e Norvig, 1995].
- **Arquitecturas Híbridas**: Combinam as características das duas abordagens anteriores, nomeadamente as capacidades deliberativa e reactiva.

Para além destas arquitecturas gerais, Wooldridge e Jennings [Wooldridge e Jennings, 1995], apresentam ainda o conceito de arquitectura por camadas (horizontais ou verticais). Neste tipo de arquitecturas, os vários subsistemas dos agentes organizam-se em hierarquias e interagem por níveis.

Uma outra arquitectura que assume particular destaque na literatura da especialidade é a arquitectura BDI *("Belief-Desire-Intention")* [Rao e Georgeff, 1991]. Esta é uma arquitectura essencialmente deliberativa em que o estado interno de processamento de um agente é descrito através de um conjunto de "estados mentais", usualmente: *crenças ("beliefs")*, *desejos ("desires")* e *intenções ("intentions")*.

Alguns autores propuseram arquitecturas mais complexas que procuram incluir todas ou pelo menos a grande maioria das capacidades referidas na secção 2.1. De entre estas arquitecturas destaca-se a arquitectura de um agente social que procura fornecer ao agente capacidades deliberativas e cooperativas de forma a poder interagir com os restantes agentes presentes no sistema [Bond e Gasser, 1988b] [Moulin e Chaib-Draa, 1996].

As secções seguintes apresentam uma análise mais detalhada e comparativa das arquitecturas referidas anteriormente.

2.5.1 Arquitecturas Deliberativas

As *Arquitecturas Deliberativas* seguem a abordagem clássica da Inteligência Artificial, onde os agentes actuam com pouca autonomia e possuem modelos simbólicos explícitos dos seus ambientes. Estas arquitecturas interpretam em grande parte os agentes como parte de um sistema baseado em conhecimento. As decisões dos agentes (selecção de qual a próxima acção a executar) são realizadas através de raciocínio lógico. O agente possui

uma representação interna do mundo, e um estado mental explícito que pode ser modificado por alguma forma de raciocínio simbólico (figura 5).

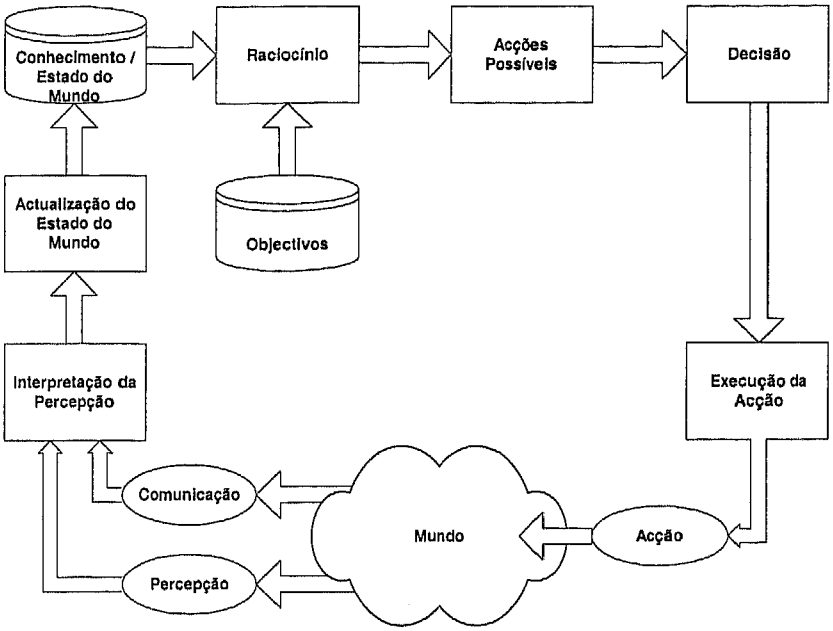

Figura 5: Esquema genérico de uma Arquitectura Deliberativa

Na construção de agentes seguindo a arquitectura deliberativa, colocam-se dois importantes problemas: como traduzir o mundo real para uma descrição simbólica e utilizar a percepção para manter essa estrutura simbólica actualizada? Como raciocinar utilizando essa informação simbólica de forma a decidir as acções a executar em cada instante?

A figura 5 apresenta um esquema típico de uma arquitectura deliberativa. Após a interpretação da percepção proveniente do ambiente, o agente utiliza esta informação para manter actualizada uma representação interna, usualmente simbólica, do estado do mundo. Este estado do mundo em conjunto com os objectivos do agente são utilizados como forma de gerar as possíveis acções a executar pelo agente e seleccionar as mais apropriadas a serem executadas por esse mesmo agente.

2.5.2 Arquitecturas Reactivas

As *Arquitecturas Reactivas* procuram não utilizar um tipo de modelo ou raciocínio simbólico complexo. Este tipo de arquitectura baseia-se na proposta de que um agente pode desenvolver inteligência a partir de interacções com o seu ambiente, não necessitando de um modelo pré-estabelecido [Brooks, 1991]. O agente reactivo habitualmente toma as suas decisões "em tempo real", com base num conjunto de informação muito limitado, e regras simples de situação/acção que permitem seleccionar um dado comportamento. A informação proveniente dos sensores é usualmente utilizada

directamente no processo de decisão não sendo criada qualquer representação simbólica do mundo (figura 6).

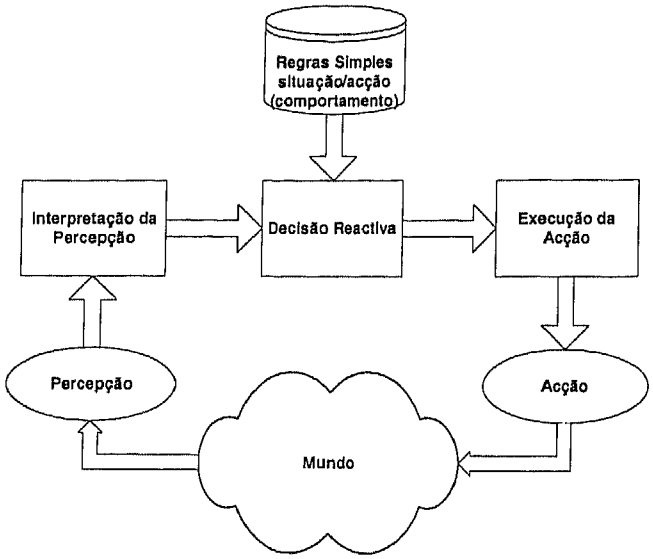

Figura 6: Esquema genérico de uma Arquitectura Reactiva

A arquitectura reactiva melhor conhecida é a *Arquitectura de Subordinação ("Subsumption Architecture")*, de Rodney Brooks [Brooks, 1986]. Existem duas características principais na arquitectura de subordinação, que são:

- A tomada de decisão do agente deve ser efectuada através de um conjunto de comportamentos direccionados para a execução de tarefas, isto é, cada comportamento deve ser pensado como uma função individual de acção, que vai continuamente tomando conhecimento do ambiente e traduz essa percepção para uma acção a executar;
- Vários comportamentos podem ser accionados simultaneamente. Assim, deverá existir um mecanismo para seleccionar de entre as diferentes acções possíveis, aquela que será a acção óptima em cada momento. Desenvolve-se então uma Hierarquia de Subordinação, que organiza os comportamentos por níveis. Os níveis mais baixos da hierarquia são prioritários em relação aos comportamentos de níveis superiores, isto porque, as acções de níveis mais altos representam comportamentos mais abstractos.

A arquitectura reactiva, nomeadamente a arquitectura de subordinação, tem grandes vantagens, como sendo, a simplicidade, economia, bom trato computacional e boa robustez contra falhas [Wooldridge e Jennings, 1995], mas também tem desvantagens evidentes, nomeadamente no que se refere à característica de os agentes decidirem unicamente baseados em informação da percepção actual, possuirem uma hierarquia de comportamentos pré-fixada e serem incapazes de realizar acções que impliquem a execução de planos de longo-prazo.

2.5.3 Arquitecturas Híbridas

Como o próprio nome sugere, as *Arquitecturas Híbridas* combinam as características das duas abordagens anteriores. Os agentes puramente reactivos têm uma limitação significativa que é a sua dificuldade em implementar comportamento orientado por objectivos. Os agentes puramente deliberativos são baseados em mecanismos de raciocínio simbólico muito complexos e tornam-se, por vezes, incapazes de uma reacção imediata a estímulos do exterior. Um agente híbrido combina estas duas componentes, e caracteriza-se por uma arquitectura composta por níveis ou camadas. A possibilidade de realizar uma disposição por níveis é uma ferramenta poderosa de estruturar as funcionalidades e controlo do agente. A ideia principal é categorizar as funcionalidades do agente em camadas dispostas hierarquicamente onde geralmente a camada reactiva tem alguma prioridade sobre a camada deliberativa, de modo a permitir uma resposta rápida aos eventos mais importantes detectados no ambiente (figura 7).

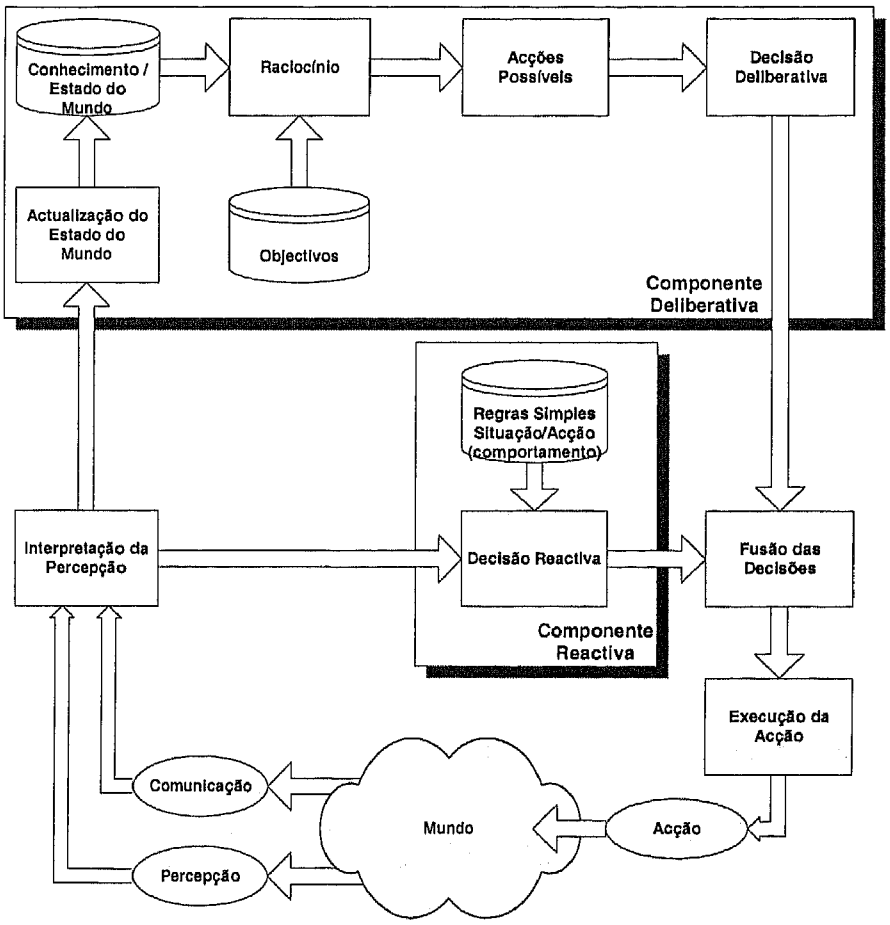

Figura 7: Esquema genérico de uma Arquitectura Híbrida

Na figura 7 é visível o esquema genérico de uma arquitectura híbrida. Após a interpretação da percepção e da comunicação provenientes do ambiente, o agente possui duas componentes: uma componente reactiva e uma componente deliberativa. O funcionamento da componente reactiva é extremamente simples e baseia-se num conjunto

de regras situação/acção que associam directamente a determinados estímulos, determinadas decisões. A componente deliberativa implica a construção de um estado simbólico do mundo e a utilização de raciocínio simbólico de forma a decidir em cada instante as acções a executar de forma a atingir os objectivos. Uma parte complexa e sujeita a um elevado número de trabalhos de investigação, é a forma de efectuar a fusão das decisões deliberativa e reactiva como forma de seleccionar a acção final a executar.

Diversos investigadores da área utilizam arquitecturas híbridas para agentes capazes de executar diferentes tarefas. Neves e Oliveira [Neves e Oliveira, 1997] [Neves, 1999] propõem a utilização desta solução para controlar um robô móvel deslocando-se num ambiente não totalmente previsível. A componente deliberativa tem aqui como função guiar o comportamento da componente reactiva, através de sugestões de planos a realizar. Uma das inovações deste trabalho está contida no facto de existir uma interacção entre as duas componentes, podendo a componente deliberativa alterar as regras de situação/acção que a componente reactiva utiliza. A componente reactiva, em certas situações de emergência tais como a iminência de uma colisão, sobrepõe-se à deliberativa, assumindo o controlo do robô.

2.5.4 Arquitectura por Camadas

A arquitectura por camadas, como o próprio nome indica, compreende a existência de vários subsistemas que se organizam em hierarquias que interagem por níveis [Wooldridge e Jennings, 1995]. Aliás, o esquema típico de uma arquitectura híbrida é a arquitectura por camadas. Podem-se considerar dois tipos de camadas, de acordo com o fluxo de controlo:

- **Camadas Horizontais.** Onde as camadas de *software* estão directamente ligados aos sensores de *input* e às acções de *output*, ou seja, cada camada actua como um agente, uma vez que apresenta sugestões no que diz respeito à acção a efectuar pelo agente (Figura 8); e
- **Camadas Verticais.** Onde os sensores de *input* e as acções de *output* têm, no mínimo uma camada entre eles, isto é, têm de ser trabalhados, pelo menos, por uma camada (ver Figura 9).

A grande vantagem da arquitectura horizontal sobre a arquitectura vertical, é a sua simplicidade conceptual. Isto porque, se é pedido ao agente para ter n tipos diferentes de comportamentos, então apenas será necessário implementar *n* camadas diferentes. Por outro lado, uma vez que as camadas competem umas com as outras, poderá ser necessária

a introdução de um mediador[10] que, embora venha para resolver o problema pode fazer com que haja um estrangulamento no processo de decisão do agente.

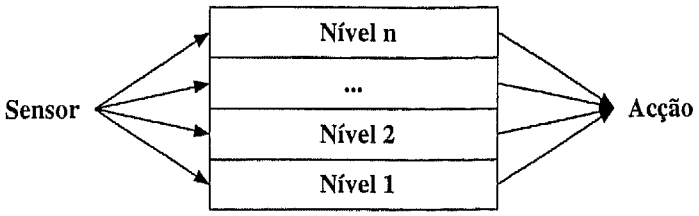

Figura 8: Representação da Arquitectura por Camadas Horizontais

Este problema praticamente não acontece com a arquitectura vertical, uma vez que o fluxo de controlo passa sequencialmente por cada camada, até que a última camada gere a acção a executar. No entanto, nesta arquitectura existe pouca flexibilidade, já que para o agente tomar uma decisão é necessário que o fluxo de controlo atravesse cada uma das diferentes camadas, ou seja, caso haja uma falha numa das camadas, o agente terá problemas para gerar e executar uma acção.

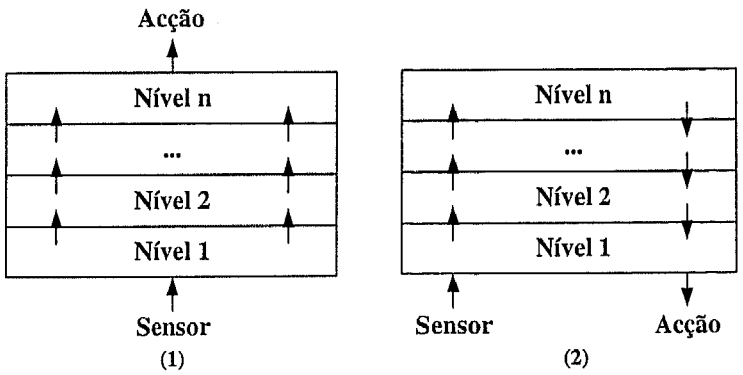

Figura 9: Representação da Arquitectura por Camadas Verticais

Conforme demonstra a Figura 9, a arquitectura por camadas verticais pode incluir unicamente uma passagem de controlo (Figura 9 (1)) ou duas passagens de controlo (Figura 9 (2)). Na arquitectura com uma passagem, o fluxo de controlo passa sequencialmente pelas diferentes camadas da arquitectura, até ao último nível, onde é gerada a acção a executar. Na arquitectura com duas passagens, a informação sobe as várias camadas até atingir o nível superior, sendo esta a primeira passagem. Em sentido inverso, o fluxo de controlo desce e atravessa as várias camadas, dando origem à acção a executar, sendo esta a segunda passagem.

[10] Este mediador irá funcionar de forma semelhante ao módulo de fusão das decisões apresentado para a arquitectura híbrida.

2.5.5 Arquitectura BDI

A arquitectura BDI ("*Belief-Desire-Intention*") [Rao e Georgeff, 1991] é uma arquitectura essencialmente deliberativa, onde o estado interno de processamento de um agente é descrito através de um conjunto de "estados mentais". Estes "estados mentais" adoptados na definição das arquitecturas BDI, diferem ligeiramente de autor para autor. No entanto, as noções de *crença* ("*belief*"), *desejo* ("*desire*") e *intenção* ("*intention*") são aceites unanimemente, até porque estão na origem do nome atribuído a esta classe de arquitecturas de agentes. As intenções numa arquitectura BDI podem ser definidas a diferentes níveis de detalhe. Por exemplo, um agente pode ter como intenção comprar um dado livro mas não ter ainda decidido em que loja o comprar. Desta forma a estrutura de intenções assemelha-se a um plano hierárquico onde os operadores aplicáveis nos vários níveis de abstracção do plano e as intenções na estrutura de intenções possuem propriedades semelhantes.

As *crenças* de um agente referem-se ao que o agente acredita em cada instante, e descrevem o estado do mundo do agente (o seu conhecimento sobre o ambiente). As crenças representam, desta forma, informação.

Os *desejos* de um agente referem-se ao que o agente deseja obter. No entanto, a forma de alcançar esses desejos pode não ser conhecida num dado instante. Os desejos podem ser, num dado momento, inconsistentes. Os objectivos dos agentes resultam de um processo de raciocínio, por parte do agente, que consiste numa escolha de um subconjunto dos desejos que são consistentes e atingíveis.

As *intenções* referem-se a um conjunto de acções ou tarefas que o agente seleccionou, comprometendo-se assim na realização dos seus objectivos. As intenções devem ser consistentes internamente e representam o resultado do processo de deliberação.

O comportamento de um agente "BDI" é ditado pelas relações existentes entre os três "estados mentais" básicos, sendo identificados, na literatura, três tipos de comportamento para estes agentes [Rao e Georgeff, 1995]:

- **Realismo Forte.** O conjunto de intenções é um subconjunto do conjunto de desejos, que é, por sua vez, influenciado por um subconjunto das crenças. Se um agente não acredita em algo, nunca irá desejar ou intentar esse algo.
- **Realismo.** O conjunto de crenças é um subconjunto do conjunto dos desejos que é, por sua vez, influenciado por um subconjunto do conjunto de intenções. Um agente realista, se acredita em algo, então deseja e intenta esse algo.
- **Realismo Fraco.** O agente não deseja algo se a negação desse algo é acreditada; não intenta algo se a negação desse algo é desejada; e não intenta algo se a negação desse algo é acreditada.

A arquitectura genérica de um agente BDI [Wooldridge, 1999] é constituída por sete componentes: conjunto de crenças, função de revisão de crenças, função de geração de opções, conjunto de opções, função de filtragem, conjunto de intenções e função de selecção de acção (figura 10).

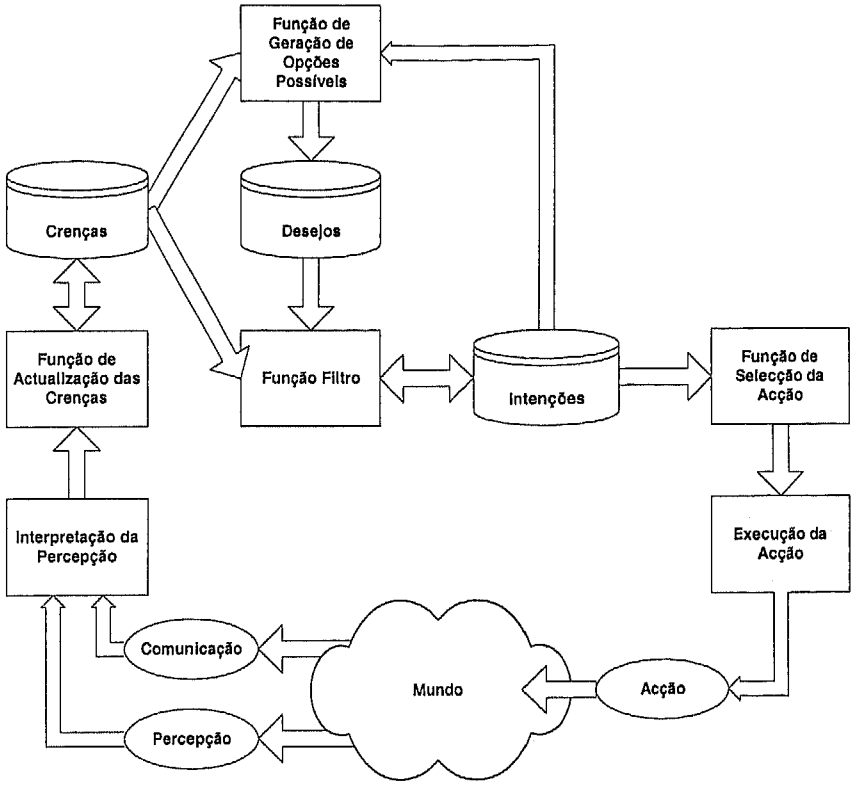

Figura 10: Esquema genérico de uma Arquitectura BDI

Estes componentes podem ser descritos da seguinte forma:

- ***Conjunto de crenças.*** Representa a informação que o agente possui em cada instante sobre o ambiente.
- ***Função de revisão de crenças.*** Função que actualiza o conjunto de crenças, de acordo com novas percepções e comunicações recebidas pelo agente e tendo em conta as crenças actuais do agente.
- ***Função de geração de opções.*** Função que determina as opções (desejos) disponíveis para o agente, com base nas suas crenças e intenções actuais. É o processo que permite ao agente decidir a linha de acção a seleccionar em cada instante, de forma a satisfazer as suas intenções. Algumas das opções resultantes da aplicação desta função tornar-se-ão intenções, e servirão de "*feedback*" para geração de novas opções, resultando na geração de opções mais concretas. O agente irá considerar assim comprometer-se a intenções progressivamente mais específicas, até que o conjunto das suas intenções corresponda a acções imediatamente executáveis. Esta função deve apresentar-se consistente - as opções geradas devem ser consistentes com crenças e intenções actuais - e

oportunista - deve ser capaz de reconhecer quando o ambiente se altera de modo a oferecer ao agente novas formas de satisfação das suas intenções, ou a possibilidade de atingir intenções que eram inatingíveis.

- ***Desejos.*** Representam os possíveis cursos de acção (opções) disponíveis para o agente.
- ***Função de filtro.*** Esta função representa o processo de deliberação do agente, que determina as intenções deste com base nas suas crenças e desejos actuais e nas suas intenções prévias. É o processo onde o agente decide o que fazer e actualiza em consonância as suas intenções. Esta actualização implica, por vezes, a desistência de intenções que deixam de ser atingíveis ou apresentam um custo demasiado elevado; e ainda a adopção de novas intenções que resultaram da adopção de novas opções.
- ***Intenções.*** Representa o conjunto de intenções, i.e., os estados que o agente está actualmente comprometido a tentar obter.
- ***Função de selecção de acção.*** Esta função determina qual a acção a realizar em cada momento com base nas intenções actuais do agente.

Refere-se ainda uma outra categorização dos "estados mentais" de um agente, realizada a um nível superior, de acordo com o conceito de atitude [Wooldridge e Jennings, 1995]. Segundo estes autores, os "estados mentais" de um agente podem ser incluídos em duas classes: atitude de informação e pró-atitude. A atitude de informação está relacionada com a informação que o agente possui sobre o mundo no qual está inserido. Nesta categoria inclui-se a *crença* e o *conhecimento*. A pró-atitude diz respeito às atitudes que guiam as acções do agente. Nesta categoria, incluem-se o *desejo*, a *intenção*, a *obrigação*, o *compromisso* e a *escolha*.

Para melhor compreender a distinção entre os conceitos de *desejos* e *intenções* surge a noção de *compromisso*. Pode afirmar-se que um agente está sempre comprometido com as suas *intenções*, e não, obrigatoriamente, com os seus *desejos* (objectivos). Este compromisso define o grau de persistência que o agente terá com as suas intenções. Determina ainda o modo como as intenções são revistas e actualizadas. As condições de término de um compromisso, classificam estes em três tipos diferentes: compromisso cego (o agente mantém a intenção até que acredite que atingiu essa intenção), compromisso honesto (o agente mantém a intenção enquanto acreditar que esta ainda pode ser atingida) e compromisso aberto (o agente desiste da intenção, não só se já não acredita que esta possa ser atingida, mas também se a intenção já não for um objectivo).

A existência de outros agentes no sistema levanta problemas adicionais ao processo de deliberação de um qualquer agente, pois é necessário raciocinar sobre as intenções dos outros, e possivelmente, sobre as intenções de conjuntos de agentes que se juntam em

actividades de grupo[11]. Norman [Norman *et al.*, 1998] introduz a noção de *acordos* entre agentes, que se baseia em *direitos* e *acções* a serem realizadas. As acções que um agente pode realizar para atingir os seus objectivos sem incorrer em penalidades impostas pelos outros membros da sociedade, são determinadas pelas acções que é capaz de executar e tem direito a realizar. Os agentes podem ainda *delegar* acções, isto é, dar permissão a outros agentes para executar acções que apenas eles têm direito a executar. Os agentes envolvidos num acordo têm de se comprometer a agir em conformidade com esse acordo. Para assegurar a estabilidade da actividade do grupo, aos agentes que estão envolvidos num acordo, é requerida uma noção de *compromisso de grupo* para agir em concordância com o acordo. Tal compromisso de grupo assegura que cada agente inserido no grupo está obrigado a fazer cumprir a sua parte no acordo. Neste contexto, Castelfranchi [Castelfranchi, 1995] faz uma distinção entre *compromisso individual*, *compromisso social* e *compromisso colectivo*. Um *compromisso social* é mais do que um *compromisso individual* partilhado por um grupo de agentes e envolve a relação entre pelo menos três agentes e uma acção. Um *compromisso colectivo* é o compromisso individual de um conjunto de agentes a uma determinada intenção, existindo conhecimento da sociedade sobre esse facto.

2.5.6 Arquitectura Genérica de um Agente Social

Uma arquitectura extremamente completa mas também bastante complexa é a de um agente deliberativo social. Este tipo de agente deverá possuir modelos explícitos dos outros agentes e capacidades para manter estes modelos actualizados em face da sua percepção e das comunicações recebidas. Estes modelos incluem crenças, objectivos e planos dos outros agentes. O agente deverá ser capaz de raciocinar utilizando o conhecimento introduzido neste módulos (intenções, expectativas, compromissos, predições, reacções antecipadas, etc.), realizando decisões e criando planos com respeito aos outros agentes. O nível de complexidade das tarefas executadas pelo agente ditará o nível de complexidade das suas estruturas de conhecimento e raciocínios efectuados. Na figura 11 é apresentada uma arquitectura genérica para um agente social baseada nos trabalhos de Bond e Gasser [Bond e Gasser, 1988b] e Moulin e Chaib-Draa [Moulin e Chaib-Draa, 1996].

[11] No futebol robótico este raciocínio é essencial, uma vez que existe uma equipa de agentes com objectivos opostos à equipa do agente.

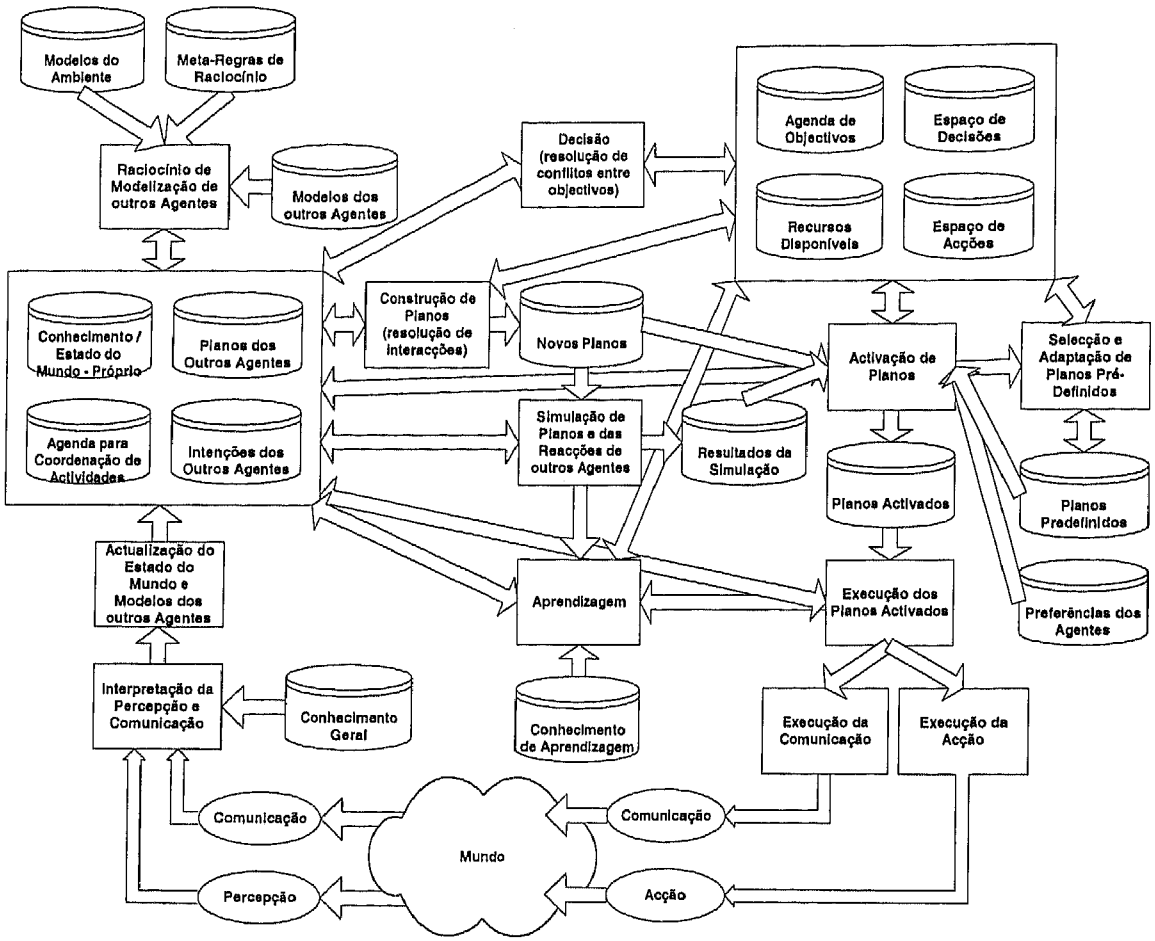

Figura 11: Esquema Genérico de uma Arquitectura de um Agente Social

Como é visível analisando a figura 11, a arquitectura proposta por Bond e Gasser é extremamente complexa, incluindo uma grande variedade de módulos que lidam com conhecimentos distintos.

A percepção e a comunicação provenientes do ambiente são interpretadas pelo agente à luz do seu conhecimento. Utilizando esta informação devidamente tratada, o agente mantém actualizada uma estrutura de informação que contém o estado do mundo actual, a agenda do agente que é responsável por activar as suas actividades activando oportunisticamente os módulos necessários, conhecimento próprio e sobre os outros agentes, nomeadamente no que diz respeito aos seus possíveis planos e intenções. As informações relativas aos outros agentes são derivadas por um módulo de raciocínio e modelização dos outros agentes. Este módulo utiliza modelos pré-definidos dos agentes e do ambiente e meta-regras de raciocínio. O módulo de modelização dos outros agentes pode ser activado de forma a inferir as intenções e planos dos restantes agentes.

O módulo de decisão é responsável por seleccionar do espaço de decisão, quais os objectivos a atingir dadas as crenças actuais do agente e as intenções dos outros agentes. Os objectivos seleccionados são colocados na agenda de objectivos que vai ser utilizada pelas funções de planeamento.

O agente possui um conjunto de planos pré-definidos e um módulo que procura seleccionar e adaptar esses planos à situação corrente. Por vezes, é impossível utilizar planos pré-definidos e como tal novos planos têm de ser construídos tendo em conta as crenças actuais do agente e os recursos que dispõe. O agente pode também ter a capacidade de simular o efeito dos novos planos tendo em conta o estado do mundo actual e as suas crenças sobre as intenções e planos dos outros agentes. Os resultados desta simulação podem ser armazenados e utilizados no processo de decisão.

A aprendizagem pode também ser incluída no agente de forma a que este utilize os resultados da execução das suas acções e conhecimento específico de aprendizagem no sentido de melhorar as suas decisões futuras. Utilizando aprendizagem, novos planos e novos objectivos podem ser incluídos respectivamente no espaço de acções e espaço de decisões do agente.

A decisão de quais os planos que vão ser activados tem em consideração as preferências do agente, os resultados da simulação, os planos disponíveis (novos e pré-definidos) e os objectivos do agente. Os planos seleccionados são então executados através da execução de acções no ambiente e comunicações com os outros agentes. As crenças do agente são actualizadas tendo em conta a informação relativa aos planos em execução.

Esta arquitectura, embora bastante complexa, contém uma base para a discussão de arquitecturas de agentes sociais envolvidos num sistema multi-agente.

2.5.7 Selecção da Arquitectura Mais Adequada

Habitualmente os diferentes investigadores na área dos agentes autónomos definem arquitecturas específicas para os agentes que projectam no âmbito de cada aplicação. Embora baseadas nas arquitecturas básicas apresentadas nesta secção, usualmente, as arquitecturas utilizadas em aplicações práticas incluem módulos específicos como por exemplo, módulos de: aprendizagem, raciocínio estratégico, escalonamento, planeamento, modelização, comunicação, cooperação, negociação, revisão de crenças, simulação, manutenção de conhecimento, etc.

A selecção da arquitectura a utilizar depende em grande medida da complexidade da tarefa que o agente irá executar e do ambiente em que ele irá operar, nomeadamente no que diz respeito às suas características (acessibilidade, dinamismo, continuidade, determinismo, etc.). Depende ainda dos outros agentes que irão coexistir com o agente no ambiente e das suas características: competitivos, cooperativos, desconhecidos à partida, etc.

2.6 Aplicações de Agentes Autónomos

O número de aplicações potenciais da tecnologia dos agentes a nível industrial e comercial é enorme. De entre estas aplicações destacam-se:

- **Aplicações Industriais.** Embora a tecnologia dos agentes seja relativamente recente, diversas aplicações industriais foram projectadas e implementadas. Aliás estas aplicações estiveram entre as primeiras a ser desenvolvidas [Jennings et al. 1998]. De entre as aplicações industriais mais relevantes da tecnologia dos agentes autónomos destacam-se: aplicações na área das telecomunicações, controlo de fabrico e controlo de processos, distribuição de energia eléctrica [Wittig et al., 1994], controlo de tráfego aéreo e sistemas de transportes [Jennings et al., 1998].

- **Agentes de Pesquisa de Informação.** Com o enorme crescimento verificado na Internet ao longo dos últimos anos e devido à natureza dinâmica e heterogénea da informação e da Internet, o paradigma dos agentes revela-se adequado à implementação de pesquisadores de informação. Desta forma as aplicações de agentes de pesquisa, recuperação e filtragem de informação na Internet constituem uma área em franca expansão no âmbito dos agentes autónomos. Neste contexto, têm também ganho relevância aplicações de agentes vulgarmente designadas por assistentes pessoais e que possuem o objectivo de auxiliar um dado utilizador a realizar as suas tarefas mais repetitivas. De entre as aplicações mais conhecidas destacam-se a gestão de correio electrónico, filtragem de notícias e marcação distribuída de reuniões.

- **Comércio Electrónico.** O enorme crescimento do comércio electrónico nos últimos anos e a evidente aplicabilidade dos agentes a este domínio levaram ao aparecimento de inúmeras aplicações de mercados electrónicos e agentes para comércio electrónico. De entre as aplicações mais relevantes destacam-se os mercados electrónicos para B2C ("business-to-consumer") tais como o Kasbah [Chavez e Maes, 1996], SMACE [Cardoso e Oliveira, 2000] [Cardoso, 1999] ou o AuctionBot [Wurman et al., 1998] e para B2B ("business-to-business") como o FOREV [Macedo, 2001] [Rocha e Oliveira, 2001a].

- **Aplicações de Entretenimento.** Os jogos e outros sistemas destinados ao lazer, sempre foram uma das aplicações mais relevantes de grande parte da tecnologia humana. Embora por vezes as aplicações relacionadas com o entretenimento não sejam consideradas investigação a sério, na realidade é frequente encontrar avanços científicos aplicados a áreas socialmente úteis que derivam directamente do desenvolvimento de aplicações na área dos jogos e lazer. De entre as aplicações de agentes neste domínio destacam-se os jogos, o desenvolvimento de personagens virtuais e histórias interactivas.

- **Aplicações Médicas.** A área da saúde sempre foi uma das áreas por excelência de aplicação da Inteligência Artificial. Sistemas Periciais tais como o sistema de diagnóstico médico MYCIN, encontram-se entre as mais mediáticas e bem sucedidas realizações da IA. Este tipo de sistemas periciais foram subsituídos ao

longo dos últimos anos por aplicações baseadas no conceito de agente. Os agentes são também aplicados no domínio da saúde e as suas principais áreas de utilização situam-se ao nível de tarefas tais como o controlo de robôs e outros equipamentos hospitalares e sistemas de tratamento de pacientes distribuídos.

- **Agentes para Simulação.** A utilização de agentes em simuladores encontra-se cada vez mais generalizada. As áreas de aplicação neste contexto incluem desde simuladores de condução, voo, combate aéreo, futebol, produção e manufactura e diversas áreas da robótica.
- **Controlo de Robôs.** A utilização de agentes no controlo de robôs destina-se a obter uma navegação segura e eficiente e um funcionamento global versátil do robô. Na maioria das aplicações, o controlo de robôs efectuado directamente por humanos é, na prática, impossível e como tal o recurso a agentes autónomos é a solução mais aconselhável. Arquitecturas baseadas em agentes para o controlo de robôs móveis, incluem [Neves e Oliveira, 1997] [Reis, 2002a] e [Lopes et al., 1999] [Lopes et al., 2000].
- **Competições de Investigação Científica.** Uma área em grande expansão actualmente é a área das competições científicas. Neste tipo de competições, a investigação realizada por investigadores de uma dada área, é comparada utilizando um problema ou plataforma comum. O paradigma dos agentes autónomos é bastante adequado para permitir a realização deste tipo de competições mesmo no caso de a investigação a comparar não estar directamente relacionada com os agentes. De facto, se cada universidade ou empresa projectar o seu agente para resolver determinado tipo de problemas e operar num dado ambiente, será muito simples comparar esses agentes posteriormente. De entre as competições mais significativas, merecem destaque o RoboCup – Futebol Robótico [Kitano, 1997] [Kitano et al., 1997], o *TAC – Trading Agent Competition* [TAC, 2001] e a competição portuguesa *Micro-Rato* [Reis, 2002a].

2.7 Conclusões

Embora a construção de agentes individuais seja muito importante, na grande maioria das aplicações de agentes autónomos, estes não vão trabalhar isolados mas sim em conjunto com outros agentes e/ou humanos. Desta forma interessa estudar as formas de colocar os agentes a trabalhar em conjunto com outros agentes, ou seja, construir sociedades de agentes designadas vulgarmente por Sistemas Multi-Agente. Um dos pontos essenciais para permitir a construção dessas sociedades de agentes consiste em conseguir gerir as interacções e as dependências das actividades dos diferentes agentes no contexto do Sistema Multi-Agente, de forma a que estes possam trabalhar em conjunto de forma harmoniosa, i.e., coordenar esses agentes. Desta forma os próximos capítulos analisam a

investigação realizada na área dos Sistemas Multi-Agente e na coordenação de agentes competitivos (interessados no seu bem próprio) e cooperativos (interessados no bem da comunidade).

Este capítulo não teve como objectivo fornecer um "*overview*" detalhado da área dos agentes autónomos mas sim introduzir os principais conceitos necessários para a compreensão dos restantes capítulos da tese. Para uma análise mais profunda e detalhada do campo dos agentes autónomos e do campo dos Sistemas Multi-Agente analisado no capítulo seguinte, sugere-se a consulta de [O'Hare e Jennings, 1996], [Huhns e Singh, 1997a], [Weiss, 1999] ou [Wooldridge, 2002].

Capítulo 3

3. Sistemas Multi-Agente

Os Sistemas Multi-Agente (SMA) são sistemas compostos por múltiplos agentes, que exibem um comportamento autónomo mas ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais: serem capazes de agir de forma autónoma tomando decisões levando à satisfação dos seus objectivos; serem capazes de interagir com outros agentes utilizando protocolos de interacção social inspirados nos humanos e incluindo pelo menos algumas das seguintes funcionalidades: coordenação, cooperação, competição e negociação.

Os SMA constituem um campo relativamente novo nas ciências da computação. Embora o início da investigação neste campo se tenha dado nos anos 80, só em meados dos anos 90 esta ganhou uma notoriedade digna de destaque [Wooldridge, 2002]. Ao longo dos últimos anos a investigação no campo dos Sistemas Multi-Agente tem sofrido um acentuado crescimento. Este crescimento levou ao aparecimento de revistas, livros e conferências internacionais sobre o assunto. Conduziu também ao aparecimento de uma rede europeia de excelência para computação baseada em agentes – *AgentLink*[12] - com o objectivo de promover e coordenar a investigação realizada na Europa nesta área. Outra demonstração do crescimento da área está relacionado com o aparecimento e sucesso de diversas competições internacionais de investigação na área tais como o *RoboCup - Futebol Robótico* (e mais especificamente a liga de simulação) em 1997 [Kitano, 1997] e o *TAC – Trading Agent Competition* em 2000 [TAC, 2001]. Na primeira destas competições, equipas de agentes autónomos disputam um torneio de futebol. Na segunda, agentes individuais ou equipas de agentes competem de forma a adquirirem e fornecerem os melhores pacotes de viagens aos seus clientes. Em ambas as competições, através da utilização de um problema standard, é possível testar e comparar a investigação realizada pelos melhores grupos de investigação mundiais em diversas sub-áreas dos agentes autónomos e Sistemas Multi-Agente e áreas relacionadas.

---

[12] http://www.agentlink.org

Neste capítulo são discutidos os Sistema Multi-Agente e analisadas as principais perspectivas relacionadas com a comunicação entre agentes, aprendizagem em SMA e coordenação de agentes. Inicialmente é analisado o conceito de SMA, a área de investigação em inteligência artificial distribuída e a motivação para os Sistemas Multi-Agente. Em seguida são analisadas as formas de comunicação entre agentes, sendo dado particular destaque às linguagens propostas para efectuar esta comunicação e ao conceito de ontologia como forma de partilhar o significado das mensagens enviadas. Depois é analisada a aprendizagem em SMA, distinguindo as suas particularidades da aprendizagem individual. Finalmente é discutida a necessidade de coordenar agentes como forma de os colocar a trabalhar em conjunto. A investigação realizada na área da coordenação de agentes é dividida em duas áreas: coordenação de agentes competitivos, nomeadamente no que diz respeito à resolução de conflitos, negociação e leilões; coordenação de agentes cooperativos, em que é dado particular destaque à área da formação de equipas e trabalho em equipa ("*teamwork*"). O capítulo conclui-se com uma análise crítica da investigação realizada na área e com alguns apontadores para a forma como alguns dos conceitos apresentados são utilizados e estendidos no âmbito do trabalho prático realizado no contexto desta tese.

3.1 O Conceito de Sistema Multi-Agente

Um Sistema Multi-Agente é um sistema computacional em que dois ou mais agentes interagem ou trabalham em conjunto de forma a desempenhar determinadas tarefas ou satisfazer um conjunto de objectivos. A investigação científica e a implementação prática de Sistemas Multi-Agente está focalizada na construção de *standards*, princípios e modelos que permitam a criação de pequenas e grandes sociedades de agentes semi-autónomos, capazes de interagir convenientemente de forma a atingirem os seus objectivos [Lesser, 1999].

Um dos pontos essenciais para permitir a construção de sociedades de agentes, consiste em conseguir gerir as interacções e as dependências das actividades dos diferentes agentes no contexto do Sistema Multi-Agente, i.e., coordenar esses agentes. Desta forma, a coordenação desempenha um papel essencial nos SMA porque estes sistemas são inerentemente distribuídos. Aliás, o tema designado genericamente por coordenação constitui um dos maiores domínios científicos da informática e ciências da computação. Trabalhos científicos abrangidos por este domínio frequentemente incluem aspectos conceptuais e metodológicos, mas também implementacionais, de forma a conseguirem expressar e implementar aplicações informáticas distribuídas.

Diversas metodologias de coordenação foram propostas por diferentes autores dividindo-se em dois grupos principais: metodologias aplicáveis em domínios contendo agentes competitivos ("*self-interested*"), i.e. agentes preocupados com o seu bem próprio, e

metodologias aplicáveis a domínios contendo agentes cooperativos, i.e., agentes que incluem uma noção de preocupação pelo bem do conjunto. No primeiro caso a coordenação por negociação é a metodologia mais estudada na comunidade internacional. Nesta metodologia, assumem particular relevância os mercados electrónicos e os leilões. Relativamente à coordenação de agentes cooperativos interessa estudar metodologias que permitam construir equipas de agentes. Neste contexto assumem particular relevância as metodologias que permitem definir uma organização estrutural da sociedade de agentes, a definição e troca de papéis, a definição e alocação de tarefas aos diversos agentes e o planeamento conjunto multi-agente.

Os Sistemas Multi-Agente incluem diversos agentes que interagem ou trabalham em conjunto, podendo compreender agentes homogéneos ou heterogéneos. Cada agente é basicamente um elemento capaz de resolução autónoma de problemas e opera assincronamente, com respeito aos outros agentes. Para que um agente possa operar como parte do sistema, é necessária a existência de uma infra-estrutura que permita a comunicação e/ou interacção entre os agentes que compõe o SMA (figura 12).

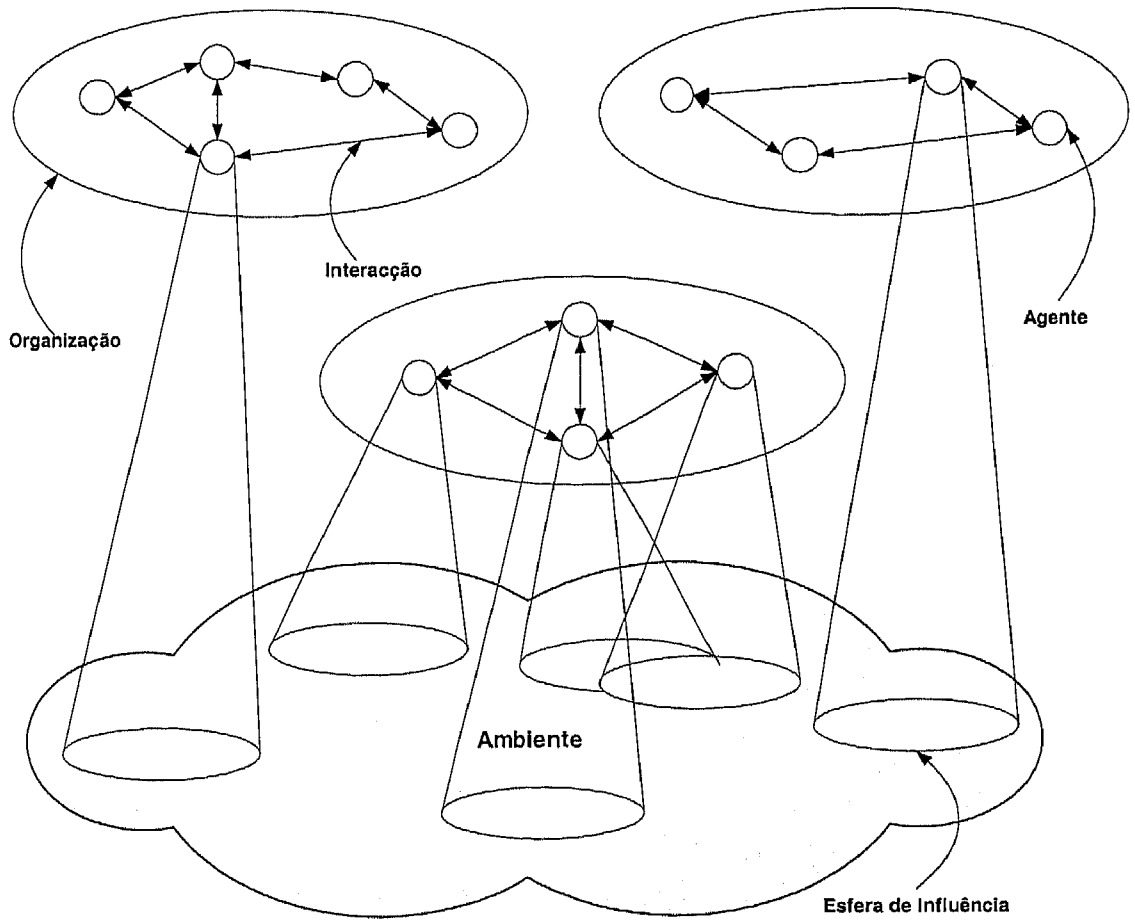

Figura 12: Estrutura de um Sistema Multi-Agente

O Sistema Multi-Agente vai conter múltiplos agentes, cada qual com diferentes capacidades de percepção e acção no mundo. Cada agente terá uma esfera de influência distinta sobre o ambiente, ou seja, será capaz de influenciar diferentes partes do ambiente

[Jennings, 2000]. Estas esferas de influência podem coincidir dependendo das relações existentes entre os agentes. Por exemplo, na geração de horários de dois departamentos, os agentes responsáveis pela geração de cada horário terão como esfera de influência os horários dos docentes que leccionam no seu departamento, das turmas dos cursos desse departamento e das salas alocadas a esse departamento. Esses agentes irão interagir no caso de efectuarem alocações em docentes, turmas ou salas que lhes sejam comuns. Interacções entre agentes que se encontram no mesmo ambiente podem ocorrer de diversas formas e como tal interessa estudar os tipos e características de cada possível interacção.

A investigação em Sistemas Multi-Agente está focada no desenvolvimento de princípios e modelos computacionais para construir, descrever, implementar e analisar as formas de interacção e coordenação de agentes em sociedades de reduzida ou elevada dimensão [Lesser, 1999]. O campo dos Sistemas Multi-Agente derivou do campo originalmente designado por Inteligência Artificial Distribuída, constituindo actualmente o núcleo deste campo. Inclui a investigação realizada em diversas áreas científicas e como tal é extremamente abrangente, sendo muito difícil expor todas as suas particularidades, num trabalho desta dimensão.

3.2 Inteligência Artificial Distribuída vs. SMA

A Inteligência Artificial é tradicionalmente apresentada como a parte da ciência da computação cujo ênfase está no estudo de sistemas inteligentes, ou seja, sistemas que exibam características associadas à inteligência no comportamento humano – compreensão de linguagem, aprendizagem, raciocínio, resolução de problemas [Barr e Feigenbaum, 1981]. Um estudo mais aprofundado e actual nesta área pode ser encontrado em [Russel e Norvig, 1995], [Nilsson, 1998] e [Coelho, 1994].

O paradigma de decompor sistemas complexos em diferentes entidades, na tentativa de obter maior eficiência, não é uma abordagem recente nas ciências da computação. Durante muitos anos, os investigadores da área da Inteligência Artificial (IA) procuraram construir agentes isolados com algum comportamento inteligente. No entanto, o crescente interesse por paralelismo e distribuição na IA marcou a última década. O desenvolvimento de computadores mais poderosos, a proliferação das redes de computadores e a constatação da utilidade do uso de grupos de indivíduos para resolver problemas, fizeram com que a Inteligência Artificial Distribuída (IAD) se tornasse uma das áreas emergentes com um maior crescimento [Bond e Gasser, 1988b].

Nwana [Nwana, 1996] também enfatiza o crescimento da IAD e separa a investigação realizada nesta área em duas linhas distintas: a primeira do período de 1977 até aos dias actuais, e a segunda a partir de 1990. A primeira linha trabalha com agentes

essencialmente deliberativos que possuem modelos simbólicos internos. Um agente deliberativo possui uma representação explícita, um modelo simbólico do mundo, e as suas decisões (acções a executar) são realizadas através de um raciocínio simbólico [Wooldridge e Jennings, 1994]. A segunda linha de investigação, em rápido desenvolvimento, enfatiza a diversificação das tipologias ou classes de agentes, onde os agentes apresentam uma evolução relativamente à sua inteligência e autonomia. Bradshaw [Bradshaw, 1997] afirma que esta linha demonstra a evolução da deliberação para a execução e do raciocínio para a acção remota.

A Inteligência Artificial Distribuída não está preocupada com assuntos relacionados com a coordenação de processos concorrentes ao nível da resolução de problemas ou representação, mas sim a um nível superior. Não está também preocupada com arquitecturas de computadores paralelos, linguagens de programação paralela ou sistemas operativos distribuídos. Outros assuntos estudados no âmbito dos sistemas distribuídos tais como semáforos, monitores e "*threads*" não são também preocupações da IAD. A IAD está preocupada essencialmente com a resolução de problemas onde diversos agentes resolvem sub-tarefas e comunicam numa linguagem de alto-nível.

Durante vários anos, a investigação realizada na área da computação distribuída situou-se num nível estritamente relacionado com a ligação física de diferentes processadores, com a transmissão de dados e seus consequentes problemas. Estas questões dominaram as discussões de tal modo que o termo "processamento distribuído" era utilizado essencialmente para designar apenas este conjunto de problemas [Chandrasekaran, 1981].

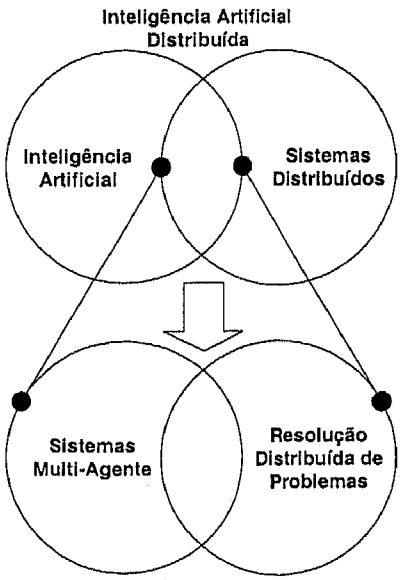

Figura 13: Inteligência Artificial Distribuída

A partir das últimas duas décadas as questões relativas ao processamento distribuído começaram a ser tratadas a um nível mais elevado de abstracção. Nesta abordagem, a atenção volta-se para as estratégias pelas quais a decomposição e coordenação do

processamento num sistema distribuído seriam conjugadas com os requisitos estruturais do domínio das tarefas. Este nível de abstracção passou a ser designado por Resolução Distribuída de Problemas (RDP) [Chandrasekaran, 1981]. Paralelamente a este desenvolvimento, a investigação efectuada em IA, inspirada em conceitos clássicos de organizações e sociedades, começava a romper com as soluções típicas até então adoptadas na construção de sistemas inteligentes.

A Inteligência Artificial Distribuída está preocupada essencialmente com a resolução de problemas onde diversos agentes resolvem sub-tarefas (macro-nível). A resolução de problemas abordados pela IAD, como foi analisado anteriormente, encontra-se dividida em dois grupos principais [Bond e Gasser, 1988a]:

- **RDP – Resolução Distribuída de Problemas.** O trabalho de resolver um problema particular pode ser dividido entre um número de módulos que cooperam e compartilham conhecimento sobre o problema e sobre o desenvolvimento da solução [Bond e Gasser, 1988a]. Sendo assim, o planeamento das acções a desenvolver é o resultado da decomposição do problema em vários subproblemas que são distribuídos aos diversos agentes envolvidos. Como resultado desta partilha, os agentes cooperam apenas na divisão do esforço e na partilha de conhecimentos e resultados;
- **SMA – Sistemas Multi-Agente.** Nesta segunda linha, contrariamente à RDP, não se parte de um problema específico, mas pretende-se coordenar um comportamento inteligente de um conjunto de agentes autónomos. Procura-se definir formas de coordenação dos seus conhecimentos, objectivos, habilidades e planos de forma a que conjuntamente possam realizar acções ou resolver problemas [Bond e Gasser, 1988a]. Assim, os agentes podem estar a trabalhar em direcção a um único objectivo global ou rumo a objectivos individuais separados que podem interagir, sendo então a autonomia dos agentes relacionada com a existência de cada agente independentemente da existência dos demais [Demazeau e Muller, 1990].

Por fim, verifica-se ainda uma terceira linha de trabalhos relacionada com a IAD, denominada *IA Paralela* [Bond e Gasser, 1988a] que trata do desenvolvimento de arquitecturas paralelas, linguagens e algoritmos para IA. Contudo, estes assuntos referem-se mais directamente aos problemas de desempenho de sistemas de IA e não a aspectos conceptuais sobre a compreensão da natureza do raciocínio e comportamento inteligente entre múltiplos agentes. Assim, e apesar dos seus desenvolvimentos certamente contribuírem para esta área, a IA Paralela não faz parte da IAD.

Pode afirmar-se que a Inteligência Artificial Distribuída está preocupada essencialmente com quatro assuntos:

- Granularidade dos agentes;

- Heterogeneidade dos agentes;
- Métodos para a distribuição do controlo (entre os agentes);
- Possibilidades de comunicação entre os agentes.

O campo dos Sistemas Multi-Agente estuda essencialmente os agentes de "*grão-grosso*", normalmente heterogéneos que se agrupam em comunidades sem controlo centralizado e que comunicam utilizando uma linguagem de alto-nível. Este campo, ganhou larga preponderância ao longo dos últimos anos, sendo neste momento claramente responsável pela maioria da investigação científica realizada no âmbito da IAD.

3.3 Motivação dos SMA

A motivação principal dos SMA encontra-se relacionada com o facto de grande parte dos problemas mais frequentemente encontrados, serem inerentemente distribuídos de uma ou várias formas. Outras motivações estão relacionadas com:

- A dimensão do problema ser demasiado elevada para poder ser resolvido por um único agente monolítico;
- Permitir a interconexão e interoperação de múltiplos sistemas legados ("*legacy*"), i.e. sistemas de gerações anteriores cuja manutenção do código já não é possível;
- Providenciar uma solução natural para problemas geográfica e/ou funcionalmente distribuídos;
- Fornecer soluções para problemas em que os peritos, os conhecimentos ou as informações necessárias para a sua resolução, se encontram distribuídos;
- Permitir uma interface cooperativa homem-máquina mais natural em que ambos funcionam como agentes no sistema;
- Oferecer uma maior clareza e simplicidade conceptual de projecto.

Num SMA, os agentes que o compõem, podem estar a trabalhar em conjunto para atingir um só objectivo geral, ou rumo a objectivos individuais separados que podem no entanto estar relacionados, sendo desta forma necessária a interacção entre os diferentes agentes no sentido de atingirem os seus próprios objectivos.

Os SMA oferecem potencialmente uma maior rentabilidade de recursos para problemas onde o conhecimento ou actividade é distribuído. Existem desta forma, várias razões adicionais para a utilização de um SMA [Stone e Veloso, 1996]:

- O próprio domínio do problema o exige, por exemplo devido à distribuição espacial dos intervenientes;
- O paralelismo, atribuindo diferentes tarefas a diferentes agentes de forma a que a

execução seja mais rápida;

- A robustez, pois utilizam-se diferentes agentes não existindo desta forma um ponto único de falha no sistema;
- A escalabilidade, permitindo o aumento dos agentes intervenientes num determinado sistema aberto;
- A simplificação das tarefas individuais de programação, dividindo o problema global em vários subproblemas;
- O estudo da inteligência individual e do comportamento social, pois os SMA permitem a interoperacionalidade entre os agentes;
- A manutenção da privacidade da informação e conhecimentos individuais de cada agente.

Em algumas circunstâncias a própria natureza do problema necessita de um SMA, como o caso da marcação distribuída de reuniões, onde os agentes autónomos se encontram realmente geograficamente distribuídos. Neste exemplo, a utilização de um SMA tem ainda como vantagem o facto de manter a privacidade da informação dos vários agentes envolvidos.

A utilização de Sistemas Multi-Agente na resolução de problemas de Inteligência Artificial apresenta diversos benefícios, nomeadamente uma maior rentabilidade de recursos para problemas onde o conhecimento ou actividade é distribuído:

- Resolução mais rápida de problemas devido ao processamento concorrente;
- Diminuição da comunicação devido ao processamento estar localizado junto à fonte de informação e a comunicação ser realizada a alto-nível;
- Aumento da flexibilidade e escalabilidade resultantes da possibilidade de interconexão de múltiplos sistemas com arquitecturas distintas;
- Aumento da fiabilidade devido à inexistência de um ponto singular de falha;
- Aumento da capacidade de resposta devido aos sensores, sistemas de processamento e actuadores estarem localizados em conjunto, no interior dos agentes;
- Facilidade acrescida de desenvolvimento de sistemas devido à modularidade resultante da decomposição dos problemas e da decomposição dos sistemas em agentes semi-autónomos.

3.4 Comunicação em SMA

A comunicação entre entidades computacionais foi desde sempre considerada um dos problemas mais importantes das ciências da computação. No entanto, na área dos Sistemas Multi-Agente, a comunicação é tratada a um nível muito mais elevado do que nas outras áreas das ciências da computação. Aqui estamos interessados na comunicação de alto-nível, utilizando linguagens de comunicação próximas das linguagens utilizadas por humanos.

Como foi analisado anteriormente, um agente para ser considerado como tal, possui capacidades de percepção, processamento e actuação num dado ambiente. Para além disso, um agente deliberativo (ou híbrido) possui uma representação interna do seu ambiente, conhecimento e capacidade de raciocinar baseado no seu conhecimento, de forma a decidir em cada instante qual a melhor acção a executar. Da mesma forma, na nossa definição de agente é assumido que um agente tem a capacidade de comunicar e habilidade social, ou seja, capacidade para interagir com outros agentes e/ou humanos presentes no seu ambiente.

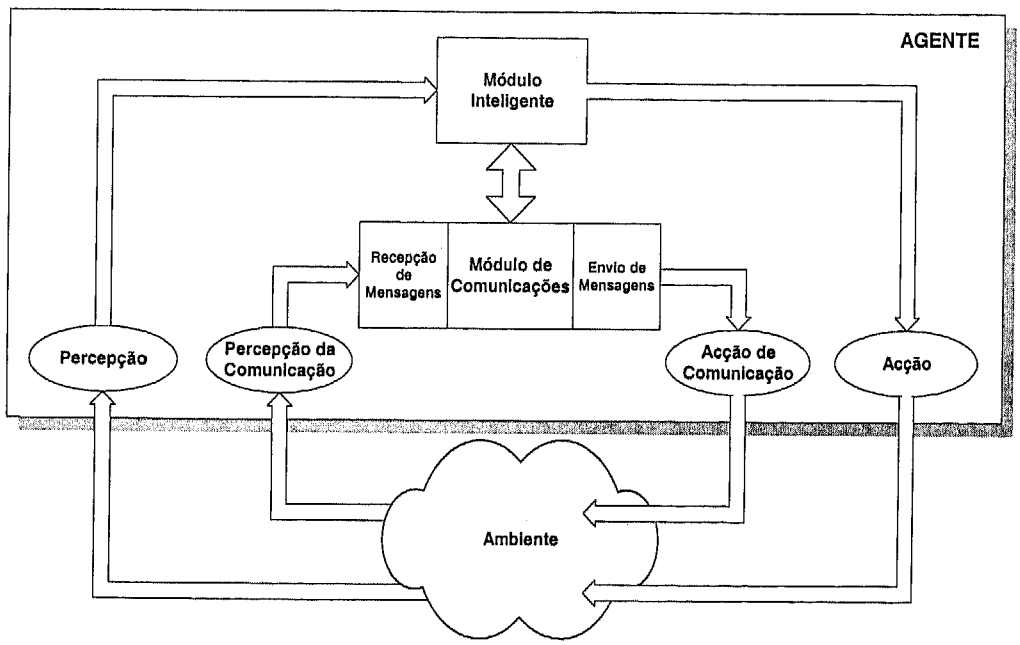

Figura 14: Agente com capacidade de comunicação

De forma a incluir a capacidade de comunicação num agente, é usual incluir um módulo de comunicações na sua arquitectura que se subdivide nas componentes de percepção (recepção de mensagens) e de acção (envio de mensagens). Este módulo de comunicações está directamente ligado ao módulo central do agente (módulo inteligente) permitindo desta forma ao módulo inteligente ter acesso às mensagens recebidas e definir quais as mensagens a enviar. Na figura 14 é representado o esquema genérico de um agente com capacidade de comunicação.

3.4.1 Arquitecturas do Sub-Sistema de Comunicação

Huhns e Stephens denotam que o sub-sistema de comunicação entre os agentes pode assumir uma de duas arquitecturas básicas [Huhns e Stephens, 1999], representadas esquematicamente na figura 15:

- **Comunicação Directa.** Os agentes tratam da sua própria comunicação sem intervenção de qualquer outro agente. Para tal, partilham especificações, enviando aos outros agentes as suas capacidades e/ou necessidades de forma a cada agente poder tomar individualmente as suas decisões relativas à comunicação. Neste tipo de arquitectura cada agente comunica directamente com qualquer outro agente, sem qualquer intermediário. Um dos principais problemas que se coloca nesta arquitectura está relacionado com a inexistência de um elemento coordenador da comunicação, o que pode originar o bloqueio do sistema se, por exemplo, todos os agentes decidirem enviar mensagens ao mesmo tempo.
- **Comunicação Assistida.** Os agentes apoiam-se em agentes especiais designados "*agentes facilitadores*", de forma a efectuarem a comunicação com os outros agentes. Nesta arquitectura a organização de agentes é do tipo sistema federado. Nestes casos, se um dado *agente i* desejar enviar uma mensagem a um outro *agente j*, terá primeiro de a enviar para o "*agente facilitador*", que se encarregará de a reencaminhar ao seu destinatário. Esta arquitectura resolve parcialmente o problema da coordenação da comunicação e diminui consideravelmente a complexidade necessária aos agentes individuais na realização de comunicação. Os agentes não necessitam de armazenar informações detalhadas sobre todos os outros agentes e nem sequer necessitam de saber o seu endereço de forma a comunicarem com eles. Basta comunicar com o "*agente facilitador*". No entanto, a existência do "*agente facilitador*" pode introduzir uma certa centralização no sistema e um estrangulamento (*bottleneck*) no sistema de comunicações. Se este agente deixar de funcionar, o sistema de comunicações deixa também de funcionar.

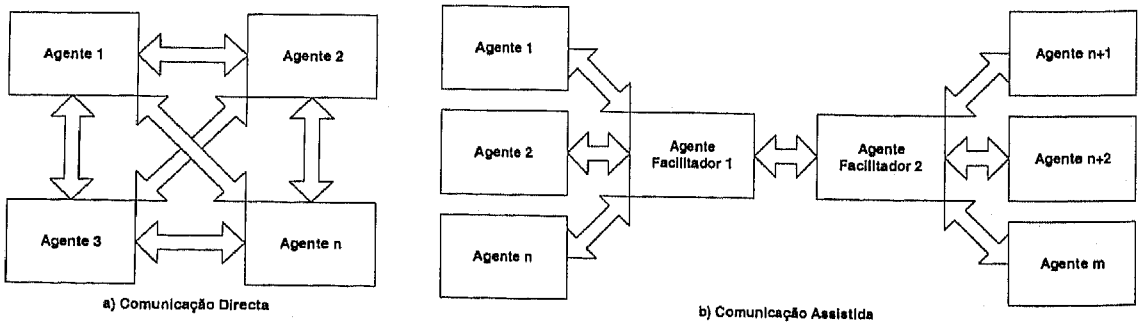

Figura 15: Arquitecturas de comunicação de agentes a) directa e b) assistida

A comunicação entre agentes está directamente relacionada com o nível de implementação das comunicações. Dependendo de vários factores, ter-se-á de considerar o modo de comunicar entre agentes. Esta comunicação pode ser implementada de duas maneiras:

- Memória partilhada (quadro-negro) por todos os agentes da comunidade;
- Passagem de mensagens entre agentes ou módulos destes.

De uma forma geral, a passagem de mensagens entre agentes é o modo de comunicação mais utilizado. A privacidade e rapidez com que a mensagem é transmitida, asseguram ao sistema uma eficácia, que de outra forma poderia ser comprometida. As soluções de memória partilhada são em geral mais difíceis de distribuir e obrigam a metodologias adicionais para a sincronização dos agentes, enquanto que a passagem de mensagens pode facilmente ser usada para sincronização. No entanto, na passagem de mensagens entre agentes devemos considerar a arquitectura do sistema de comunicação a implementar. Se implementarmos uma arquitectura de comunicação directa, deveremos assegurar que os agentes podem receber mensagens constantemente e inversamente prever a possibilidade de uma mensagem enviada não ser recebida pelo agente destinatário. A implementação de uma arquitectura de comunicação assistida pode resolver os problemas levantados pela arquitectura anterior, nomeadamente, se o agente "facilitador" for munido da capacidade de manter as mensagens ainda não recebidas pelos destinatários e reenviá-las periodicamente até que o agente destinatário a receba. No entanto, a implementação desta arquitectura implica geralmente uma menor eficiência ao nível da rapidez de transmissão.

Optar por um modo de comunicação com recurso a memória partilhada, resolve os problemas suscitados nas implementações referidas anteriormente. No entanto, a privacidade da informação e a rapidez, podem ser comprometidas se a implementação do próprio sistema não previr mecanismos muito eficientes de controlo de acesso à informação que se encontra na memória partilhada.

3.4.2 Características da Comunicação

A comunicação tem dois fins principais: partilha do conhecimento, informação, crenças ou planos com outros agentes; e coordenação de actividades entre agentes. No entanto a realização de comunicação que permita atingir estas duas metas, requer a definição de uma linguagem comum ou partilhada, caracterizada por:

- **Sintaxe.** Estrutura da ligação dos símbolos que compõem o discurso, ou seja, a parte da estrutura gramatical da linguagem que contém as regras relativas à combinação das palavras em unidades maiores, e as relações existentes entre as palavras dentro dessas unidades;
- **Semântica.** Significado dos símbolos e das suas combinações. É o estudo da linguagem do ponto de vista do significado das palavras e dos enunciados;

- **Vocabulário.** Conjunto de símbolos usados. Lista dos vocábulos da linguagem, em geral desacompanhados da respectiva definição ou com uma explicação muito sucinta;
- **Pragmática.** Conjunto de regras de acção e fórmulas de interpretação dos símbolos utilizados na comunicação;
- **Modelo do domínio do discurso.** Significado que um conjunto de símbolos assume quando interpretado num determinado contexto de conversação.

Na decisão do acto de comunicação, é importante ter em consideração quatro pontos essenciais [Reis, 2002c]:

- **O que comunicar?** Independentemente do fim a que se destina (por exemplo realizar partilha de conhecimento ou reforçar a coordenação entre agentes), as mensagens enviadas não podem conter toda a informação que o agente emissor possui. Este agente, tendo em consideração as características do canal de comunicação e o fim a que se destina a comunicação, tem de raciocinar sobre qual a informação mais importante a incluir nas suas mensagens.
- **Quando comunicar?** Sobretudo em domínios com reduzida largura de banda disponível para operações de comunicação[13], é muito importante decidir correctamente o momento temporal em que é efectuada a comunicação. Por vezes, se a informação a comunicar não for muito importante, será preferível não realizar qualquer comunicação, deixando desta forma, a largura de banda do canal de comunicação, disponível para os outros agentes presentes no sistema.
- **A quem comunicar?** Como será analisado mais à frente, as mensagens podem ser enviadas para um, vários ou todos os agentes. A decisão do receptor da mensagem assume particular importância nos casos de comunicação directa entre dois agentes.
- **Como comunicar?** No caso de existirem diversos meios de comunicação disponíveis ao agente é necessário seleccionar, para cada comunicação, qual o meio mais adequado para a executar.

Um agente comunica uma dada mensagem num dado momento e de acordo com um dado contexto. Neste sentido, é importante que as mensagens sejam interpretadas de acordo com o contexto em que são enviadas ou recebidas. Por exemplo, em situações que envolvam a troca sucessiva de mensagens entre dois agentes (diálogo) é importante garantir que a sequência das mensagens não se altera e que nenhuma mensagem foi

[13] Nos capítulos oito e nove será analisado um domínio (futebol robótico simulado) em que, devido a possuir uma muito reduzida largura de banda para comunicação, a decisão de quando comunicar é muito importante.

perdida ao longo do diálogo. Caso contrário, uma mensagem pode ser interpretada de forma totalmente errada.

3.4.2.1 Caracterização do Significado das Mensagens

Genericamente podemos caracterizar o significado das mensagens trocadas por agentes inteligentes, segundo as características [Singh, 1997]:

- **Descritiva vs. Prescritiva**. Descrição de fenómenos ou prescrição de comportamentos. As descrições, embora muito importantes no comportamento humano, são complexas de entender para os agentes. Desta forma, os agentes comunicam informação acerca de actividades e comportamentos.
- **Significado Personalizado vs. Convencional.** A interpretação que um agente possui de uma mensagem pode não ser a mesma que os restantes agentes lhe conferem (significado convencional). Nos sistemas Multi-Agente, nomeadamente nos sistemas abertos é aconselhável a utilização de significados convencionais, pois a qualquer momento são adicionados novos agentes.
- **Subjectiva vs. Objectiva.** O significado é objectivo quando este é interpretado externamente ao agente e tem um efeito explícito no ambiente.
- **Perspectiva do Remetente vs. Destinatário vs. Sociedade.** Independentemente do significado convencional ou objectivo da mensagem, esta pode ser expressa segundo o ponto de vista do remetente, destinatário ou dos restantes agentes presentes no ambiente.
- **Semântica vs. Pragmática.** A pragmática de uma comunicação está focada em como os agentes usam a comunicação, isto é, como agem em função da comunicação. Esta tem em consideração o estado dos agentes, o meio envolvente, a sintaxe e a semântica da comunicação.
- **Contextualidade.** As mensagens não devem ser entendidas isoladamente. A sua interpretação deve levar em consideração o estado do agente, o estado do meio e a história do meio. A interpretação deve ser directamente influenciada pelo contexto da comunicação.
- **Cobertura.** As linguagens com uma dimensão reduzida podem ser mais flexíveis e mais facilmente utilizáveis. No entanto, têm de ser suficientemente abrangentes para que os agentes possam expressar todos os significados que necessitem para comunicar com os outros agentes.
- **Identidade.** O sentido de uma mensagem depende da identidade dos interlocutores e do papel que cada um desempenha no sistema.
- **Cardinalidade.** Uma mensagem enviada a um agente em particular, deve ser entendida de forma diferente de uma mensagem pública, enviada a todos os

agentes do sistema.

3.4.2.2 Tipo de Mensagens

Todos os agentes têm de possuir a capacidade de comunicar independentemente do conjunto de capacidades funcionais que os caracterizam. De acordo com as características do agente e papel a desempenhar no sistema, podemos distinguir diversos níveis de capacidades de comunicação. É assumido que um agente pode enviar e receber mensagens, existindo dois tipos de mensagens básicas: as asserções e as perguntas [Huhns e Stephens, 1999].

Podemos considerar quatro tipos de agentes (tabela 2), de acordo com as capacidades de comunicação que apresentam:

- **Agente Básico.** Capaz de aceitar informação do exterior sob a forma de asserções mas incapaz de efectuar outras formas de comunicação;
- **Agente Passivo.** Capaz de um diálogo sendo, além de aceitar asserções, capaz de aceitar perguntas do exterior e responder-lhes na forma de asserções;
- **Agente Activo.** Capaz de realizar um diálogo em que assume um papel activo. Além de aceitar asserções, tem a capacidade de realizar perguntas e asserções;
- **Agente Interlocutor.** Capaz de realizar um diálogo em que assume um papel interlocutor entre os outros agentes. Para além de aceitar asserções, é capaz de fazer e receber perguntas e realizar asserções.

	Agente Básico	Agente Passivo	Agente Activo	Agente Interlocutor
Receber Asserções	Sim	Sim	Sim	Sim
Receber Perguntas	Não	Sim	Não	Sim
Enviar Asserções	Não	Sim	Sim	Sim
Enviar Perguntas	Não	Não	Sim	Sim

Tabela 2: Capacidades de comunicação de diferentes agentes [Huhns e Stephens, 1999]

3.4.2.3 Protocolos e Níveis de Comunicação

Os protocolos de comunicação são usualmente definidos a vários níveis [Huhns e Stephens, 1999]. Os níveis inferiores definem o método de interligação dos agentes. Os níveis intermédios definem o formato (sintaxe) da informação transmitida. Nos níveis superiores encontram-se as especificações do sentido (semântica) da informação.

No que diz respeito à aridade, os protocolos de comunicação subdividem-se em protocolos de aridade binária e aridade *n*. Enquanto um protocolo binário, envolve apenas um emissor e um receptor, um protocolo de aridade *n* implica a existência de um emissor e múltiplos receptores. Genericamente podemos definir que um protocolo contém a seguinte estrutura de dados [Huhns e Stephens, 1999]:

- Emissor;

- Receptor(es);
- Linguagem utilizada;
- Funções de codificação e descodificação da linguagem;
- Acções que o receptor deve executar.

3.4.2.4 Actos de Discurso

A Teoria dos Actos de Discurso iniciou-se com o trabalho do filósofo John Austin [Austin, 1962] e foi mais tarde estendida por John Searle [Searle, 1969]. Esta teoria resulta da análise do discurso humano relativa às acções, pedidos, sugestões, compromissos e respostas. Este é usado como modelo de comunicação aplicado à comunicação dos agentes.

O princípio básico da teoria dos actos de discurso é a assunção de que as acções de discurso são executadas pelos agentes da mesma forma que quaisquer outras acções e de acordo com o cumprimento dos seus objectivos. No seu estudo, John Austin [Austin, 1962] concluiu que os actos de discurso tinham características semelhantes às acções, no sentido em que estes actos poderiam mudar o estado do mundo de forma análoga às acções físicas. Austin distinguiu ainda três aspectos essenciais dos actos de discurso [Austin, 1962]:

- **A Locução.** Acto físico de discurso;
- **A Elocução.** Sentido atribuído à locução;
- **A "Perlocução".** Efeito da acção resultante da locução.

John Searle estendeu o trabalho de Austin [Searle, 1969] e identificou diversas propriedades necessárias para o sucesso de actos de discurso. De entre estas condições destacam-se as condições normais de transmissão (*input/output*), condições preparatórias e condições de sinceridade. Searle identificou ainda um conjunto de classes para os actos de discurso, incluindo [Searle, 1969]: directivas (pedidos), promessas, expressivas, representativas (informações) e declarativas.

A Teoria do Acto de Discurso utiliza o termo "performativa" para identificar a elocução (tabela 3). As "performativas" podem ser verbos como: prometer, convencer, insistir, dizer, pedir, oferecer, requerer, etc. A utilização das performativas destina-se exclusivamente a definir inequivocamente a elocução desejada. Assim, definem o sentido com que deve ser interpretado o conteúdo da mensagem. O resultado reflecte a acção esperada do outro interveniente face ao primeiro acto de discurso.

Acto de Comunicação	Elocução	Resultado
Asserção	Informativo	Aceitação
Pergunta	Questionar	Resposta
Resposta	Informativo	Aceitação
Pedido	Pedido	
Explicação	Informativo	Acordo
Comando	Pedido	
Permissão	Informativo	Aceitação
Recusa	Informativo	Aceitação
Oferta/Factura	Informativo	Aceitação
Aceitação		
Concordância		
Proposta	Informativo	Oferta
Confirmação		
Retracção		
Negação		

Tabela 3: Tipos de mensagens inter-agentes derivadas da Teoria do Acto de Discurso

3.4.3 Linguagens de Comunicação

No início dos anos 90, foi fundado nos Estados Unidos da América, o *Knowledge Sharing Effort (KSE)* financiado pelo *DARPA*[14], com o objectivo de desenvolver protocolos para a troca e representação de informação entre sistemas de informação autónomos. O *KSE* gerou dois produtos finais principais [Finin et al., 1993]:

- **A Linguagem *KQML* (*Knowledge and Query Manipulation Language*).** KQML é uma linguagem externa para comunicações entre agentes. Define um invólucro para formatar mensagens que determina o significado locutório da mensagem. O KQML não está preocupado com o conteúdo da mensagem mas sim com a caracterização da informação necessária à compreensão desse conteúdo.
- **O Formato *KIF* (*Knowledge Interchange Format*).** A *KIF* é uma linguagem que se destina explicitamente a representar o conhecimento sobre um domínio de discurso específico. Foi desenvolvido primariamente como forma de definir o conteúdo de mensagens expressas em *KQML*.

Para além do *KQML* e *KIF*, existem várias linguagens definidas no âmbito da comunicação em Sistemas Multi-Agente. De entre as mais utilizadas destaca-se o *FIPA ACL* (*Agent Communication Language*). Em 1995, a *FIPA – Foundation for Intelligent Physical Agents* iniciou o desenvolvimento de *standards* para Sistemas Multi-Agente. A parte fulcral desta iniciativa situava-se ao nível do desenvolvimento de uma linguagem de comunicação para agentes [FIPA, 1999]. A *ACL* resultante é semelhante ao *KQML*, sendo primariamente uma linguagem de comunicação externa e não obriga à utilização de qualquer linguagem específica para o conteúdo.

[14] DARPA – Defense Advanced Research Projects Agency, E.U.A.

Para além da preocupação com a definição de uma linguagem de comunicação, é óbvio que para os agentes interagirem num sistema terão, todos, não só que falar a mesma linguagem mas atribuir significados idênticos aos conceitos em discussão. Só assim serão capazes de entender e serem entendidos pelos outros agentes. Torna-se assim necessário a existência de uma Ontologia que especifique o significado dos objectos e conceitos em discussão.

Comum a todas as linguagens é o facto de estas não terem ou não especificarem uma infra-estrutura própria de comunicação. As especificações das linguagens e das plataformas de comunicação são totalmente independentes. Nesta secção é apresentada uma breve descrição das linguagens *KQML, KIF* e *FIPA ACL* e analisado o conceito de Ontologia e a sua utilidade no âmbito dos Sistemas Multi-Agente.

3.4.3.1 KIF – Knowledge Interchange Format

O *KIF* [Genesereth e Fikes, 1992] foi desenvolvido originalmente como uma linguagem comum para representar propriedades de um determinado domínio. Foi proposto como uma linguagem normalizada para a descrição de entidades em sistemas periciais, bases de dados, agentes inteligentes, etc. O *KIF* não foi desenvolvido como um formato para expressar mensagens completas mas sim como um formato para expressar o conteúdo das mensagens. De facto, o *KIF* é baseado em lógica de primeira ordem e tem uma aparência muito semelhante a esta lógica, utilizando uma notação prefixa semelhante à linguagem *LISP*. A linguagem não é difícil de ser interpretada quer por pessoas quer por sistemas computacionais. O *KIF* foi também definido para servir de "interlingua", isto é, para servir, nos processos de tradução entre linguagens, de linguagem intermédia entre as linguagens inicial e final.

O *KIF* permite descrever, de forma simples, propriedades de um dado domínio, relações entre os objectos presentes nesse domínio e propriedades gerais desse domínio. De forma a permitir a expressão destes conceitos, o *KIF* providencia operadores comuns em lógica de primeira-ordem: operadores *booleanos* e lógicos (*and*, *or*, *not* ,etc.), quantificadores universal e existencial, etc. Providencia também estruturas de dados típicas tais como números, caracteres, *strings* ou listas e relações sobre esses tipos de dados.

O *KIF* é uma linguagem prefixa da lógica de predicados de primeira ordem com extensões para o suporte de raciocínio não monótono e de definições. A descrição da linguagem inclui a especificação da sintaxe e da semântica. Além da descrição de entidades, factos e conhecimento, o *KIF* permite ainda a descrição de procedimentos, isto é, a descrição de sequências de acções para que os agentes as executem. Excluindo as especificidades próprias da linguagem, realmente o *KIF* baseia-se fundamentalmente na lógica de primeira ordem e destina-se à expressão do conteúdo de mensagens *KQML*.

3.4.3.2 *KQML - Knowledge and Query Manipulation Language*

O *KQML* é uma linguagem para troca de informação e conhecimento baseada em troca de mensagens. É caracterizada por especificar toda a informação necessária à compreensão do conteúdo da mensagem. Cada mensagem *KQML* é composta por uma performativa (que pode ser interpretada como a classe de mensagem) e por um determinado número de parâmetros (cada qual especificando um atributo e seu respectivo valor). Na figura 16 é apresentado um exemplo de uma mensagem *KQML*.

```
(ask-one
        :content (SCHEDULE LPR)
        :sender ECO
        :receiver TIMETABLE_AGENT
        :reply-with SCH_LPR1
        :language SCHEDULE_UNILANG
        :ontology TIMETABLING
)
```

Figura 16: Exemplo de uma mensagem KQML

No exemplo representado no figura 16, o agente *ECO* envia ao agente *TIMETABLE_AGENT*, o pedido de informação relativo ao horário do agente *LPR*. A mensagem especifica ainda a linguagem e ontologia utilizadas. A performativa *ask-one* utilizada destina-se a realizar uma questão a um dado agente em que exactamente uma resposta é esperada. A resposta a esta mensagem deve ser dada contendo o identificador enviado no parâmetro *:reply-with*. A linguagem utilizada é definida no parâmetro *:language* e o parâmetro *:ontology* define a terminologia utilizada nesta comunicação. A tabela 4, apresenta os principais parâmetros utilizados em mensagens *KQML*.

Parâmetro	Significado
:content	Conteúdo da mensagem
:sender	Emissor da mensagem
:receiver	Receptor da mensagem
:language	Linguagem do conteúdo da mensagem
:ontology	Ontologia utilizada no conteúdo da mensagem
:force	Especifica se o conteúdo da mensagem é definitivo ou o emissor o poderá alterar no futuro.
:reply-with	Definição se o emissor da mensagem aguarda por uma resposta e se tal for verdadeiro, qual o identificador para essa resposta
:in-reply-to	Referência ao identificador de resposta fornecido por um reply-with prévio

Tabela 4: Parâmetros de uma mensagem KQML

Diversas versões do *KQML* foram propostas ao longo dos últimos anos com um conjunto diferente de performativas. Em [Finin et al., 1993], o número de performativas incluídas na linguagem, ascendia a 41, contendo, entre outras: *advertise, ask-all, ask-one, broadcast, discard, evaluate, forward, recommend-all, recommend-one, reply, tell, untell,* etc. Para um estudo mais alargado sobre a linguagem *KQML* recomenda-se a leitura de [Finin et al., 1993] ou [Labrou e Finin 1994].

As principais vantagens da utilização da linguagem *KQML*, residem no facto de esta ser uma linguagem *standard* e de efectuar a separação entre o domínio da "performativa" *KQML* e a semântica da mensagem. No entanto, diversos investigadores apontaram

também diversas críticas a esta linguagem: o conjunto de performativas alterou-se significativamente ao longo dos tempos (implicando dificuldades de comunicação entre sistemas *KQML* desenvolvidos em instantes temporais distintos); a semântica do *KQML* nunca foi definida com precisão (as definições são dadas em inglês corrente); a linguagem não continha performativas para efectuar compromissos entre agentes; o conjunto de performativas era demasiado extenso e desenvolvido de forma muito "*ad hoc*".

No entanto, *KQML* é uma linguagem em desenvolvimento contínuo, onde a sua sintaxe e semânticas vão sendo progressivamente melhoradas [Labrou e Finin, 1994]. *Labrou* e *Finin* propuseram uma nova especificação que refina a original [Labrou e Finin, 1997] e que foi uma das bases da *ACL* suportada pela *FIPA* (*Foundation for Intelligent Physical Agents*) [FIPA, 1999].

3.4.3.3 *FIPA ACL – Agent Communication Language*

A *FIPA ACL* é semelhante no formato à linguagem *KQML*. Define essencialmente a estrutura exterior da mensagem. No entanto, o número de performativas *ACL* é bastante inferior ao número de performativas *KQML*. A linguagem *ACL* define unicamente 20 performativas (tabela 5) para definir a interpretação desejada para cada tipo de mensagem.

Performativa	Significado
accept-proposal	Aceitação de proposta numa negociação
agree	Aceitação de desempenhar uma dada acção
cancel	Cancelamento da execução de uma dada acção
cfp	"*Call for proposals*". Utilizada para iniciar uma dada negociação
confirm	Confirmação da veracidade de uma dada mensagem
disconfirm	Inverso da mensagem anterior
failure	Uma tentativa de executar uma dada acção (usualmente requisitada por outro agente) que falhou
inform	Uma das performativas mais importantes da *FIPA ACL*. Permite comunicar informação aos outros agentes
inform-if	Informação sobre a veracidade de determinada informação
inform-ref	Informação sobre um dado valor
not-understood	Indicação de que uma dada mensagem não foi percebida
propagate	Pedido de propagação a um conjunto de agentes de uma dada mensagem
propose	Envio de proposta, por exemplo, como resposta a uma mensagem cfp
proxy	Permite enviar uma mensagem que vai ser reenviada a um conjunto de agentes
query-if	Pedido de informação sobre a veracidade de determinada informação
query-ref	Pedido de informação sobre um dado valor
refuse	Recusa de executar determinada acção
reject-proposal	Recusa de uma proposta efectuada no contexto de uma dada negociação
request	Uma das performativas mais importantes da *FIPA ACL*. Consiste num pedido a um dado agente para executar determinada acção
request-when	Pedido para executar uma dada acção quando uma determinada condição for verdadeira.
request-whenever	Pedido para executar uma dada acção sempre que uma determinada condição seja verdadeira.
Subscribe	Pedido para ser informado acerca das alterações relacionadas com determinado facto ou informação

Tabela 5: Performativas providenciadas pela linguagem FIPA ACL.

Como é visível analisando a tabela 5, a diferença principal entre a *FIPA ACL* e o *KQML* está relacionada com as performativas providenciadas por cada linguagem. A linguagem *FIPA ACL* procura providenciar uma semântica mais compreensível do que a linguagem *KQML*. Uma das vantagens mais significativas da *FIPA ACL* relativamente ao *KQML* consiste na disponibilização de performativas mais adequadas à execução de processos de negociação.

3.4.3.4 Ontologias

Como foi analisado anteriormente, para que seja possível a interacção entre os agentes que compõem um Sistema Multi-Agente (SMA) é necessário que exista uma plataforma de comunicação, uma linguagem de comunicação e que os agentes possuam um vocabulário comum bem definido. No entanto, é usual que diferentes agentes possuam diferentes terminologias para o mesmo significado, ou idêntica terminologia para significados diferentes. Este problema pode ser resolvido se eles partilharem uma ontologia comum. Uma ontologia não é mais do que: "a representação do conhecimento de um dado domínio, disponibilizada a todos os outros componentes de um sistema de informação" [Huhns e Singh, 1997b]. A utilização de uma ontologia comum garante a consistência (uma mesma expressão tem o mesmo significado para qualquer agente) e compatibilidade (um conceito qualquer é designado, pela mesma expressão, por qualquer agente) da informação presente no sistema [Macedo, 2001].

Existem diversas definições de ontologia propostas pelos investigadores da área. Jim Hendler define uma ontologia como uma definição formal de um corpo de conhecimento [Hendler, 2001]. Grüninger e Fox [Grüninger e Fox, 1995] definem uma ontologia como uma descrição formal de entidades, propriedades, relacionamentos, restrições, comportamentos. Weiss [Weiss, 1999] define ontologia como uma especificação dos objectos, conceitos e relacionamentos de uma dada área de interesse. Na sua definição, uma ontologia vai além de especificar uma taxonomia de classes (ou tipos), pois deve também descrever os relacionamentos. Segundo Gruber [Gruber, 1993], uma ontologia é uma especificação da conceitualização, ou seja, dos conceitos e relacionamentos que podem existir para um agente ou para uma comunidade de agentes. Noy e McGuinness afirmam que uma ontologia é uma descrição explícita formal de conceitos de um domínio do discurso (classes), propriedades desses conceitos descrevendo as suas características (*slots*), atributos desses conceitos (designados por papéis ou propriedades), e restrições sobre esses atributos (nomeadamente restrições aos valores que podem tomar) [Noy e McGuinness, 2001]. Para estes autores, o desenvolvimento de uma ontologia inclui [Noy e McGuinness, 2001]: definir um conjunto de classes; organizar as classes numa hierarquia taxonómica; definir as propriedades (*slots*); descrever os valores permitidos para estas propriedades; preencher os valores das propriedades com instâncias.

No contexto de um Sistema Multi-Agente, uma ontologia é uma representação formal de conceitos, características e relacionamentos num dado domínio específico, permitindo o

entendimento comum da área de conhecimento entre pessoas e agentes de *software*. Isto permite a uma máquina usar o conhecimento disponibilizado por humanos ou por outras máquinas, múltiplas máquinas compartilharem o seu conhecimento e ainda, a reutilização de conhecimento [Malucelli, 2002]. Uma ontologia é então uma especificação dos objectos, conceitos e relacionamentos de uma área de interesse. Os conceitos poderão ser representados por predicados unários de lógica de primeira ordem. Os relacionamentos serão representados por predicados de aridade superior.

As ontologias, para além do campo dos Sistemas Multi-Agente, são aplicadas em muitos campos de investigação, dos quais se destaca o desenvolvimento de bases de dados, na componente da sua especificação, com a utilização de modelos como o Relacional ou Entidade-Associação ou ainda o UML (*Unified Modelling Language*).

Existem diversos tipos de ontologias, entre os quais de destacam:

- **Ontologias genéricas.** Definem termos suficientemente genéricos e frequentemente usados como base para permitir a definição de outros termos compostos. Podem ser consideradas como meta-ontologias;
- **Ontologias de domínio.** Ontologias baseadas nas genéricas mas especializadas para uma determinada área. Os termos e conceitos definidos nestas ontologias dizem respeito à área seleccionada;
- **Ontologias de modelização e representação.** Disponibilizam as primitivas que são utilizadas nas ontologias genéricas e de domínio;
- **Ontologias de aplicação.** Contêm as definições necessárias à modelização do conhecimento numa área específica.

O grau de formalização de uma ontologia permite também efectuar a sua classificação em *ontologias informais* (expressas em linguagem natural), *ontologias semi-informais* (expressas numa forma restrita e controlada da linguagem natural), *ontologias semi-formais* (expressas numa linguagem formal, artificialmente definida para o efeito) e *ontologias formais* (expressas numa linguagem lógica com uma semântica definida formalmente).

Existem diversas linguagens de representação do conhecimento utilizadas para modelização de ontologias. De entre estas destacam-se a *XML (eXtensible Markup Language)* [XML, 2001], *UML (Unified Modelling Language)* que é uma linguagem padrão para especificar, visualizar, documentar e construir sistemas computacionais, o *KIF* (abordado na secção 3.4.3.1), o *DAML (DARPA Agent Markup Language)* [DAML, 2001] que foi desenvolvido como uma extensão do *XML*, o *OIL (Ontology Inference Layer)* e o *CycL* (baseado na lógica de primeira ordem).

As ontologias são cada vez mais essenciais para o desenvolvimento e utilização de sistemas inteligentes assim como para a interoperação de sistemas heterogéneos. Permitem comunicar aos agentes presentes num sistema (pessoas ou agentes de *software*), o

vocabulário que está disponível sobre um dado domínio importante para o funcionamento do sistema. Acrescentam a este vocabulário, o significado que é atribuído no âmbito desse mesmo sistema para os diversos termos desse vocabulário. Com a generalização da sua utilização e o alargamento dos Sistemas Multi-Agente, contendo agentes desenvolvidos por diferentes pessoas ou organizações, as necessidades de comunicação com vocabulários comuns entre estes agentes será cada vez mais relevante. Desta forma, as ontologias vão desempenhar num futuro próximo um papel central no desenvolvimento de Sistemas Multi-Agente.

3.5 Aprendizagem em SMA

Embora neste trabalho não sejam utilizadas metodologias de aprendizagem, tendo em conta a importância que a aprendizagem em geral possui nos Agentes e Sistemas Multi-Agente, nesta secção será realizada uma breve análise deste tema. Será dada particular relevância à distinção entre aprendizagem individual e aprendizagem multi-agente e às metodologias de aprendizagem utilizadas especificamente em SMA.

A aprendizagem em Sistemas Multi-Agente é um campo de investigação bastante recente mas, no entanto, muito importante dentro da IAD. Embora este tema tenha sido negligenciado durante bastante tempo, preferindo os investigadores da área analisar tópicos tais como arquitecturas, comunicação, coordenação, cooperação ou negociação multi-agente[15], é hoje uma convicção geral que este tema merece particular atenção. Duas motivações podem ser apontadas para este facto:

- Providenciar a Sistemas Multi-Agente heterogéneos, compostos por múltiplas unidades com conhecimentos e capacidades distintas, a capacidade para aprender e ajustar o seu comportamento individual e de grupo, automaticamente;
- Perceber melhor os mecanismos e processos naturais de aprendizagem em grupos ou equipas (por exemplo um grupo de agentes cooperando entre si e com humanos).

Os SMA possuem habitualmente uma complexidade estrutural e de funcionamento considerável, sendo usualmente impossível determinar à partida, o conjunto de comportamentos e as actividades concretas que irão ser executadas pelo sistema. Desta forma assume particular importância o desenvolvimento de agentes capazes de aprenderem e se adaptarem de forma a melhorarem o seu funcionamento no contexto do SMA.

---

[15] O autor deste trabalho não escapa incólume a esta afirmação pois, nesta tese, o tema da aprendizagem em SMA não é abordado numa perspectiva prática.

3.5.1 Tipos de Aprendizagem em SMA

A aprendizagem em SMA é mais do que uma mera magnificação da aprendizagem de agentes individuais. Engloba todas as complexidades da aprendizagem de agentes isolados e é possível construir agentes no contexto de um SMA, que aprendam isoladamente, de forma completamente independente dos outros agentes. No entanto, a aprendizagem em SMA vai mais além e pressupõe que os agentes aprendam em conjunto e que a sua aprendizagem seja influenciada pela troca de informações, assunções partilhadas, pontos de vista comuns sobre o ambiente, convenções sociais, normas, etc. [Weiss, 1996]

Podem ser considerados dois tipos principais de aprendizagem multi-agente [Weiss, 1996]:

- **Aprendizagem Interactiva.** Situações em que os agentes de forma colectiva procuram atingir os seus objectivos de aprendizagem comuns;
- **Aprendizagem Individual.** Situações em que cada agente procura atingir os seus próprios objectivos de aprendizagem mas em que a sua aprendizagem é afectada por outros agentes, os seus conhecimentos, crenças, intenções, etc.

3.5.2 Determinação do Crédito ou Culpa

Um problema sempre presente na aprendizagem multi-agente é a determinação e atribuição do crédito ou culpa pela mudança (positiva ou negativa) no comportamento global do sistema. É necessário decompor este crédito pelos vários agentes que contribuíram para essa mudança global no comportamento. Este problema pode ser decomposto em dois subproblemas [Weiss, 1996]:

- A atribuição do crédito ou culpa pela mudança global do comportamento às acções externas de cada agente. Na aprendizagem em SMA este problema é complexo pois as acções de vários agentes podem, simultaneamente contribuir para a alteração do comportamento global do sistema.
- A atribuição do crédito ou culpa da execução de uma acção às decisões internas correspondentes. Este problema é praticamente idêntico em aprendizagem individual e aprendizagem SMA. Consiste na determinação das decisões que levaram à execução da acção ou acções relevantes na mudança de comportamento do sistema.

Esta decomposição conceptual não é no entanto clara em grande parte dos problemas [Sen e Weiss, 1999]. Grande parte das abordagens à aprendizagem em SMA não diferenciam explicitamente os dois subproblemas e focam-se unicamente num deles, simplificando consideravelmente o outro [Sen e Weiss, 1999].

3.5.3 Características Importantes na Aprendizagem

As duas categorias de aprendizagem descritas (i.e. aprendizagem interactiva e individual) cobrem um largo espectro de tipos de aprendizagem mais específicos que podem ocorrer no contexto de um SMA. Na figura 17, é apresentado um diagrama que resume os principais tópicos de investigação no âmbito da aprendizagem em SMA.

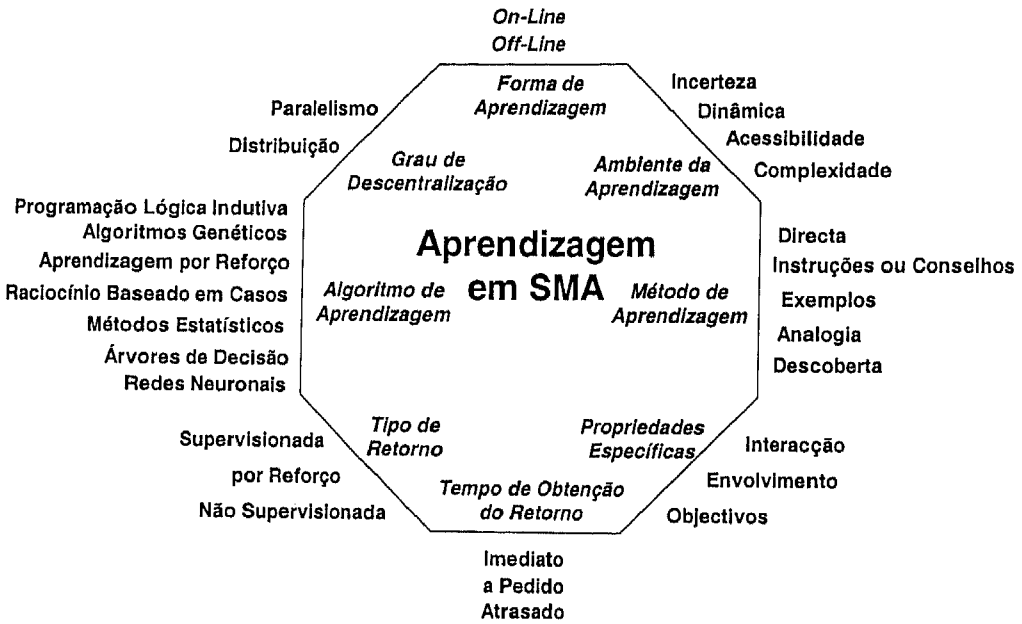

Figura 17: Características da Aprendizagem em SMA

As abordagens à aprendizagem em SMA podem ser classificadas com maior precisão utilizando um conjunto de características auxiliares [Sen e Weiss, 1999]:

- **Grau de Descentralização.** Nomeadamente no que diz respeito ao grau de distribuição e paralelismo do processo de aprendizagem;
- **Forma de Aprendizagem.** De acordo com a disponibilidade dos dados necessários para o processo de aprendizagem, podem classificar-se os métodos de aprendizagem como "em-linha" (*"online"*) (aprendizagem realizada durante a operação normal do agente) e *"off-line"* (aprendizagem realizada posteriormente analisando dados guardados durante a operação normal do agente).
- **Ambiente de Aprendizagem.** O ambiente pode possuir todas as características definidas na secção 2.2. Desta forma é necessário ter em consideração a dinâmica, incerteza, acessibilidade, determinismo e complexidade do ambiente. A aprendizagem será mais complexa em ambientes multi-agente dinâmicos, contínuos, não determinísticos e inacessíveis.
- **Propriedades Específicas da Interacção.** Estas propriedades incluem o nível da interacção (observação, argumentação, negociação, etc.), persistência da interacção (curto prazo a longo prazo), frequência da interacção, estruturação da

interacção (hierárquica, sem qualquer estruturação, etc.), variabilidade da interacção, etc. Desta forma podem existir situações em que o processo de aprendizagem seja extremamente rápido e simples (por exemplo observando um outro agente durante um curto espaço de tempo) e situações em que este processo seja demorado e extremamente complexo (por exemplo negociação utilizando um protocolo elaborado com um elevado número de agentes);

- **Propriedades Específicas do Envolvimento.** Este envolvimento pode ser caracterizado pela sua relevância (essencial ou dispensável no processo de aprendizagem colectiva) e pelo papel desempenhado pelo agente (generalista ou especialista numa dada actividade);
- **Propriedades Específicas dos Objectivos.** Estas propriedades podem incluir o tipo de melhoria esperada pela utilização do processo de aprendizagem (individual ou colectiva) e a compatibilidade dos objectivos de aprendizagem dos vários agentes (complementares ou conflituosos).
- **Método de Aprendizagem.** Os métodos de aprendizagem são uma das áreas mais importantes e com maior volume de investigação realizada dentro da aprendizagem em SMA. De entre os métodos mais relevantes destacam-se:
  - **Aprendizagem Directa.** Implantação directa do conhecimento ou capacidades sem qualquer tipo de inferência complementar por parte do agente;
  - **Aprendizagem por Instruções ou Conselhos.** Fusão do conhecimento ou capacidades adquiridas com conhecimento e capacidades prévias do agente;
  - **Aprendizagem por Exemplos.** Extracção e refinamento de conhecimento e capacidades tais como padrões ou conceitos gerais, a partir de exemplos positivos e negativos;
  - **Aprendizagem por Analogia.** Transformação de conhecimento utilizado na resolução de um problema semelhante de forma a resolver um novo problema análogo;
  - **Aprendizagem por Descoberta.** Recolha de novo conhecimento e capacidades através da realização de observações, experiências, teste de hipóteses e teorias e análise de resultados experimentais.
- **Tipo de Retorno da Aprendizagem.** O retorno da aprendizagem indica a melhoria obtida até um dado momento. Pode ser fornecido pelo ambiente, calculado pelo próprio agente ou fornecido por um outro agente. É usual a decomposição dos métodos de aprendizagem relativamente ao retorno, em três classes:
  - **Aprendizagem Supervisionada.** O retorno especifica a actividade

desejada do agente aprendiz e o objectivo final da aprendizagem é obter um resultado o mais próximo possível deste retorno;

- **Aprendizagem por Reforço.** O retorno especifica unicamente a utilidade da acção ou actividade actual do agente aprendiz. O objectivo consiste em maximizar esta utilidade;
- **Aprendizagem Não Supervisionada.** Neste tipo de aprendizagem não existe retorno. Desta forma, o objectivo consiste em encontrar acções e actividades úteis, tendo como base um sistema de tentativa e erro.

- **Tempo de Obtenção do Retorno.** No caso de existência de retorno, a forma de obtenção desse retorno da aprendizagem pode ser:
  - **Imediata.** Todas as acções têm um *"feedback"* imediato que permite ao agente avaliar a qualidade dessas acções;
  - **A Pedido.** O agente aprendiz pode requisitar em qualquer instante o retorno de uma dada acção ou conjunto de acções passado.
  - **Atrasada.** O retorno é enviado ao agente mas não de uma forma imediata e/ou constante. Os instantes de tempo em que o retorno é enviado não são controlados pelo agente aprendiz.

Utilizando estas características é possível caracterizar o processo de aprendizagem a diversos níveis. Tendo em conta o elevado número de combinações possível, é fácil concluir que a determinação da metodologia de aprendizagem mais adequada a cada caso específico é, em si, também um problema significativo neste contexto.

Por vezes a aprendizagem *"off-line"* não é possível devido a restrições impostas, como por exemplo a impossibilidade de guardar informação suficiente que a permita efectuar. Este problema pode ainda ser mais grave no caso de as acções necessárias para efectuar a aprendizagem afectarem directamente, de forma destrutiva, o ambiente ou os seus outros utilizadores. Neste caso, será muito difícil a um agente realizar um processo de aprendizagem, pois esse processo de aprendizagem afecta o próprio comportamento do sistema. Desta forma, o agente estará a aprender a adaptar-se a um ambiente adulterado, o que poderá não ser útil no ambiente real.

3.5.4 Algoritmos de Aprendizagem

Um elevado número de algoritmos foi proposto na literatura da especialidade para permitir realizar a aprendizagem em agentes ou outras entidades computacionais. De entre os mais estudados e utilizados destacam-se:

- Métodos Estatísticos (aprendizagem não supervisionada);
- Raciocínio Baseado em Casos (*"Case Based Reasoning"*);

- Árvores de Decisão;
- Redes Neuronais (e outros algoritmos conexionistas);
- Aprendizagem por Reforço;
- Programação Lógica Indutiva;
- Algoritmos Genéticos (e outros algoritmos evolucionários).

Devido a limitações de espaço e ao facto deste trabalho não utilizar metodologias de aprendizagem na sua componente mais prática, não apresentamos aqui uma descrição mais detalhada destes algoritmos. Para uma análise mais detalhada dos algoritmos de aprendizagem referidos, sugere-se a leitura de [Mitchell, 1997] ou [Silva, 2000b]. No que diz respeito à análise da aprendizagem no contexto de um Sistema Multi-Agente, estudos mais aprofundados podem ser encontrados em [Sen e Weiss, 1999], [Weiss, 1996], [Stone e Veloso, 1996] e [Nunes e Oliveira, 2002].

3.5.5 Dificuldades Específicas da Aprendizagem em SMA

Embora a aprendizagem em Sistemas Multi-Agente, como foi analisado ao longo desta secção, possua algumas características distintivas relativamente à aprendizagem individual, de facto, a utilização de aprendizagem em SMA é realizada na maioria dos casos utilizando a forma tradicional de aprendizagem individual. Desta forma, a interacção entre os agentes durante o processo de aprendizagem e os benefícios que esta interacção pode trazer para o processo de aprendizagem colectiva não são usualmente explorados.

3.6 Coordenação em SMA

O conceito de coordenação é intuitivo e é muito simples possuir uma noção do que significa a palavra coordenação. Por exemplo, quando se participa numa conferência bem organizada, se analisa uma linha de montagem que funciona suavemente, se lecciona numa faculdade bem gerida, ou se assiste a um jogo de futebol entre duas equipas capazes de realizar excelentes jogadas colectivas, é fácil perceber que existem acções de grupo bem coordenadas. No entanto, em grande parte dos casos, a boa coordenação é quase invisível e é mais fácil perceber o que é coordenação quando esta falha [Malone e Crowston, 1994]. Exemplos de ausência de coordenação verificam-se quando: se espera horas para embarcar num avião porque não existe uma porta para realizar o embarque, se chega a um hotel onde se tinha uma reserva e o hotel está cheio, se chega a uma sala para leccionar uma aula e esta se encontra ocupada ou, se assiste a um jogo de futebol com muitos passes errados e jogadores mal posicionados no campo.

A coordenação pode ser definida como "o acto de trabalhar em grupo de forma harmoniosa" [Malone e Crowston., 1991]. No entanto, tal como acontece com a definição do conceito de agente, também a definição do conceito de coordenação não é partilhada por todos os investigadores da área. Desta forma, diversos autores propuseram definições distintas mas interrelacionadas do termo coordenação.

3.6.1 Definição de Coordenação

Diversos autores propuseram definições distintas para o conceito de coordenação, destacando-se:

- "Ajustamento ou interacção harmoniosa" [AHD, 2000]
- "O acto de trabalhar em grupo de forma harmoniosa" [Malone et al., 1990]
- "Processo pelo qual um agente raciocina acerca das suas acções locais e das acções (antecipadas) dos outros para tentar assegurar que a comunidade actue de maneira coerente" [Jennings, 1996]
- "Processo de gestão das interdependências entre actividades" [Malone e Crowston, 1994]
- "A operação de sistemas complexos compostos de componentes" [NSF-IRIS, 1989]
- "O esforço conjunto de actores independentes no sentido de atingirem objectivos mutuamente definidos" [NSF-IRIS, 1989]
- "A integração e ajustamento harmonioso dos esforços individuais no sentido de alcançar um objectivo mais amplo" [Singh, 1992]
- "O acto de trabalhar em conjunto" [Malone e Crowston, 1991]

Na maioria destas definições destaca-se a noção de grupo, trabalho conjunto, harmonia e objectivo comum. Desta forma, a definição de coordenação que adoptámos neste trabalho é a seguinte:

> *"O acto de trabalhar em conjunto de forma harmoniosa no sentido de atingir um acordo ou objectivo comum" [Reis, 2002e]*

Nas secções anteriores analisamos a forma como agentes podem partilhar o mesmo ambiente e comunicar entre si. No entanto, de forma a que os agentes possam trabalhar em conjunto de forma harmoniosa é necessário mais do que a capacidade de comunicarem entre si. Duas dificuldades principais surgem neste contexto [Wooldridge, 2002]:

- Num Sistema Multi-Agente os agentes são muitas vezes implementados por projectistas distintos com objectivos distintos. Desta forma podem não partilhar objectivos comuns e como tal podem ter de negociar ou argumentar com os outros agentes de forma a persuadi-los a efectuarem a coordenação;

- Devido aos agentes serem autónomos, efectuando as suas decisões dinamicamente em *"run-time"*, têm de ser capazes de coordenar, também dinamicamente, as suas actividades, com os outros agentes presentes no ambiente.

Estas duas dificuldades levaram à proposta de muitas metodologias de coordenação distintas por parte de diversos investigadores da comunidade dos SMA. No entanto, existe uma grande dificuldade em estabelecer uma teoria da coordenação unanimemente aceite por todos os investigadores da área. De forma a efectuar a coordenação é, no entanto, nítido que os agentes têm de ser capazes de partilhar informação e tarefas dinamicamente (i.e. em *"run-time"*) coordenando as suas actividades no contexto do Sistema Multi-Agente.

A definição de boas estratégias de coordenação permite que grupos de agentes executem tarefas cooperativas de forma eficiente através de decisões conjuntas sobre que agentes devem executar uma determinada tarefa e quando e a quem devem comunicar os seus resultados e conhecimento em cada instante.

3.6.2 A Necessidade de Coordenar Agentes

Sichman e Demazeau [Sichman, 1994] [Sichman e Demazeau, 1995] afirmam que a necessidade de coordenação resulta da existência de relações de dependência que estão relacionadas com o facto de um agente necessitar do outro de forma a poder satisfazer os seus objectivos. Definem quatro relações de dependência básicas:

- **Independência.** Não existe qualquer tipo de dependência entre os agentes;
- **Unilateral.** Um agente depende do outro mas a relação inversa não se verifica;
- **Mútua.** Ambos os agentes dependem do outro para atingirem os seus próprios objectivos;
- **Dependência recíproca.** Um agente depende do outro para um dado objectivo e o outro agente depende do primeiro para um outro objectivo (não necessariamente o mesmo).

Wooldridge [Wooldridge, 2002] denota que estas relações podem ser qualificadas como localmente ou mutuamente "acreditadas". Uma relação é localmente acreditada se um agente acredita que a relação existe mas não acredita que o outro agente acredita que ela existe. Uma relação mutualmente acreditada existe quando um agente acredita que a relação existe e acredita também que o outro agente a reconhece.

Malone e Crowston [Malone e Crowston, 1994] analisaram as relações de dependência numa outra perspectiva e identificaram nove tipos de dependência entre actividades. Para cada tipo de dependência sugeriram processos para as gerir como demonstra a tabela 6.

Dependência	*Processos de Coordenação para Gerir a Dependência*
Recursos Partilhados	*"First come/First serve", Ordenação por Prioridades, Orçamentos, Decisão da Gestão, Negociação e Ofertas*
Alocação de Tarefas	*"First come/First serve", Ordenação por Prioridades, Orçamentos, Decisão da Gestão, Negociação e Ofertas*
Relações Produtor/Consumidor	*Notificação, Sequenciamento, Seguimento, Gestão de Stocks, Just in Time, Quantidade Económica de Encomenda, Normalização, Projecto Participatório, Interacção com os Utilizadores*
Restrições de Pré-Requisitos	*Notificação, Sequenciamento, Seguimento*
Transferência	*Gestão de Stocks, Just in Time, Quantidade Económica de Encomenda*
Usabilidade	*Normalização, Projecto Participatório, Interacção com os Utilizadores*
Projecto para produção	*Engenharia Concorrente*
Restrições de Simultaneidade	*Escalonamento, Sincronização*
Tarefas e Sub-Tarefas	*Selecção de Objectivos, Decomposição de Tarefas*

Tabela 6: Tipos de Dependência entre Actividades [Malone e Crowston, 1994]

Jennings [Jennings, 1996] afirma que existem três razões principais para a necessidade de coordenar as acções de múltiplos agentes:

- **Existem dependências nas acções dos agentes.** Interdependência ocorre quando as acções necessárias para atingir os objectivos dos agentes individuais estão relacionadas. Isto pode acontecer devido a decisões locais de agentes terem impacto nas decisões de outros agentes ou devido à possibilidade de existência de interacções destrutivas entre os vários agentes;
- **Existe a necessidade que o conjunto de agentes respeite restrições globais.** Estas restrições podem situar-se a nível de custos, tempo, recursos, etc. Se os agentes agissem individualmente não conseguiriam respeitar estas restrições. Unicamente se coordenarem as suas actividades podem respeitá-las;
- **Nenhum agente individualmente tem recursos, informação ou capacidade suficiente para executar a tarefa ou resolver o problema completo.** Grande parte dos problemas necessitam de conhecimentos distintos para serem resolvidos, que só podem ser conseguidos por diferentes agentes. No entanto, os conhecimentos distintos dos vários agentes têm de ser combinados de forma a produzir o resultado desejado. Os diversos agentes podem também ter recursos distintos (capacidade de processamento, memória, etc.) que têm de ser utilizados de forma coordenada para resolver o problema. Os agentes podem também possuir informação distinta, por exemplo, devido a possuírem sensores distintos ou estarem geograficamente em locais distintos. Podem ainda ter as capacidades

de posicionamento diferentes, podendo posicionar-se em zonas distintas. Desta forma as suas capacidades de percepção e acção sobre o meio serão distintas.

Mesmo quando os agentes individuais possam trabalhar independentemente dos restantes, a coordenação pode aumentar a eficiência do sistema. Desta forma, Nwana et al. [Nwana, et al., 1996], adicionam duas razões para a necessidade de coordenar agentes num sistema:

- **Eficiência.** Através da troca de informação ou divisão de tarefas, a coordenação pode aumentar a eficiência do sistema. Mesmo sendo ambos capazes de executar as tarefas que lhes estão atribuídas, dois agentes poderão trocar as suas tarefas se ambos forem capazes de executar de forma mais eficiente a nova tarefa. Um outro exemplo, poderá ser um agente enviar informação ou permitir a utilização dos seus recursos computacionais de forma a aumentar a eficiência de uma tarefa executada por outro agente;
- **Prevenir a anarquia e o caos.** A coordenação é necessária ou desejável porque, devido à descentralização dos sistemas multi-agente, a anarquia pode-se estabelecer facilmente. Em sistemas com uma dimensão ou complexidade razoáveis, os agentes possuem unicamente uma visão parcial do mundo e conhecimento e objectivos locais que podem entrar em conflito com os dos outros agentes. No entanto, estão habilitados a interagir com os outros agentes do sistema e, tal como em qualquer sociedade, estas interacções podem provocar grande confusão, desordem e desorganização, conduzindo à anarquia. Nenhum agente possui uma visão global e autoridade sobre o sistema pelo que será necessário que os agentes coordenem as suas actividades voluntariamente com os restantes agentes do sistema.

A coordenação por seu lado, pode requerer cooperação. No entanto, é importante destacar que a coordenação pode ser atingida por outros meios não cooperativos. A competição ou negociação são processos de coordenação para agentes antagonistas. Para que os agentes se coordenem, usualmente têm de comunicar. No entanto, é possível a coordenação sem comunicação, nomeadamente nos casos em que os agentes possuem modelos do comportamento dos restantes agentes. Outra possibilidade, consiste em definir previamente uma estrutura organizacional, atribuindo papéis, responsabilidades, cadeias de comando ou hierarquias entre os agentes.

Neste contexto, uma conclusão importante é que nem sempre é útil realizar acções de coordenação pois isto pode implicar o dispêndio de recursos úteis na realização de outras tarefas. Lesser [Lesser, 1998], afirma que de forma a desenvolver estratégias eficientes de coordenação que funcionem numa grande variedade de ambientes, os agentes devem explicitamente medir os benefícios e custos da coordenação na situação corrente, de forma quantificável. A situação corrente inclui os objectivos (e a sua importância) que um agente possui a curto-prazo, os métodos ou acções disponíveis a esse agente e aos outros agentes, os requisitos que a aplicação desses métodos coloca ao agente e aos outros agentes, o

estado do ambiente do agente e outras restrições que sejam impostas à actividade do agente. Lesser denota ainda que efectuar decisões de coordenação é um problema complexo, multi-nível de optimização [Lesser, 1999] que se baseia em verificar (usualmente numa perspectiva estatística) em que medida as acções de coordenação contribuem a alto-nível para o atingir da execução das acções que conduzem ao objectivo do agente.

3.6.3 SMA Cooperativos vs. Competitivos

Podemos distinguir duas abordagens na construção de Sistemas Multi-Agente: sistemas compostos por agentes cooperativos e sistemas compostos por agentes competitivos, muitas vezes designados por egoístas ("*self-interested*").

Os Sistemas Multi-Agente cooperativos são usualmente projectados por uma única, ou eventualmente por múltiplas, mas no entanto interdependentes, entidades ou pessoas. Neste tipo de sistema, os agentes comportam-se de forma a incrementar a utilidade global do sistema e não a sua utilidade pessoal. Desta forma existe uma preocupação com o aumento do desempenho global do sistema e não do desempenho individual dos agentes.

Nos Sistemas Multi-Agente compostos por agentes competitivos, cada agente é projectado por um projectista distinto. Cada agente tem a sua própria agenda e motivação. Os agentes não estão usualmente interessados no bem da comunidade mas sim, em obter a sua satisfação pessoal. Em cenários que envolvam aquisição de bens ou serviços, tais como o comércio electrónico na Internet, este tipo de Sistema Multi-Agente é muito mais realista.

3.6.4 Coordenação de Agentes Competitivos

Um problema óbvio na coordenação e cooperação de agentes está relacionado com o atingir de acordos em sociedades de agentes competitivos. Este tipo de sociedade competitiva é bastante comum entre os humanos. Todos os dias interagimos com agentes com quem não partilhamos objectivos comuns e em casos mais extremos a única forma de lucrarmos pode ser à custa dos nossos oponentes[16]. No entanto, em cenários mais realistas é típico que os agentes partilhem pelo menos alguns objectivos, e existe geralmente alguma margem para que se possam atingir acordos que sejam mutuamente benéficos. Aliás, em geral, a capacidade para atingir acordos sem que exista uma terceira parte a ditar os termos, é essencial em grande parte das aplicações de agentes inteligentes autónomos. Desta forma, capacidades de negociação e argumentação são centrais na criação de agentes autónomos capazes de funcionar em sociedades de agentes que não partilhem necessariamente objectivos comuns. Sendo a negociação uma forma de coordenação por

[16] Este tipo de encontro designa-se por encontro de soma-zero.

excelência para atingir acordos entre agentes competitivos, o capítulo seguinte irá referir com maior detalhe a investigação realizada nesta área.

Embora a negociação desempenhe um papel primordial na coordenação em Sistemas Multi-Agente compostos por agentes competitivos, nas situações em que um conjunto de agentes partilham objectivos comuns, a negociação não é o mecanismo de coordenação mais aconselhável. Nestes casos, a coordenação dos agentes pode ser realizada em grande parte por metodologias de cooperação uma vez que o conjunto de agentes partilha os mesmos objectivos e existem vantagens mútuas em cooperarem para os atingirem.

3.6.5 Coordenação de Agentes Cooperativos

O interesse em aplicações de agentes em que agentes cooperativos trabalham em conjunto como equipas tem crescido significativamente nos últimos anos. Cada vez mais agentes são construídos com o fim específico de trabalharem como membros de equipas, em domínios tais como o treino virtual [Tambe, 1995], ambientes interactivos ou o futebol robótico [Kitano et al., 1997].

Diversas metodologias foram propostas na literatura da especialidade de forma a permitirem efectuar a coordenação de agentes cooperativos que realizam uma determinada tarefa em conjunto. De entre estas, destacam-se a resolução distribuída e cooperativa de problemas [Durfee e Lesser, 1987], a coordenação por partilha de tarefas e resultados, coordenação por contratação [Smith e Davis, 1980], planeamento global parcial [Durfee, 1988] [Decker, 1995], intenções conjuntas [Cohen et al., 1990], planos partilhados [Grosz e Kraus, 1996] e *locker-room agreement* [Stone e Veloso, 1999]. Outras metodologias de coordenação mais gerais foram desenvolvidas e aplicadas em diversos domínios por diferentes autores. Exemplo disto são a coordenação por organização estrutural, normas e leis sociais, conhecimento *a priori* e modelização mútua.

Um dos principais problemas da maioria das metodologias de coordenação propostas é que a sua aplicação só é possível em domínios relativamente simples. De facto, poucas são as metodologias aplicadas com sucesso em domínios inacessíveis, dinâmicos, não determinísticos e contínuos. Estes tipos de domínios abertos tornam-se ainda mais complexos quando existe uma necessidade de coordenação espacial dos agentes envolvidos, ou seja, quando estes possuem capacidades de mobilidade, tal como em cenários de guerra ou no futebol robótico. Desta forma, a criação de novas metodologias de coordenação de alto-nível de equipas de agentes é um tópico ainda pouco explorado dentro da investigação em coordenação em SMA.

3.7 Conclusões

Independentemente de serem colaboradores ou competidores, os agentes que formam um Sistema Multi-Agente irão interagir uns com os outros. Esta interacção não é indesejada mas sim intrínseco ao próprio conceito de agente que pressupõe a sociabilidade como forma de um agente atingir os seus próprios objectivos. As interacções pressupõem que os agentes conheçam os outros agentes presentes no ambiente ou, pelo menos, que estejam a par da sua existência, ou seja, que os agentes sejam projectados como membros de uma sociedade multi-agente. No entanto, estas interacções têm de ser convenientemente coordenadas pois caso contrário a sociedade de agentes pode degenerar numa sociedade completamente descoordenada e caótica.

Os dois capítulos seguintes analisam as metodologias de coordenação, separando a coordenação de agentes competitivos das restantes metodologias de coordenação aplicáveis essencialmente a agentes cooperativos. Esta divisão prende-se com o facto de a negociação em SMA ser uma metodologia aplicável essencialmente em SMA compostos por agentes competitivos (egoístas), sendo assim bastante diferente das restantes metodologias de coordenação aplicáveis a agentes cooperativos.

Capítulo 4

4. Coordenação de Agentes Competitivos

A negociação automática é uma das áreas mais activas de investigação no contexto dos sistemas multi-agente e é, por excelência, a metodologia de coordenação para agentes competitivos. Em geral, a negociação entre agentes tem como objectivo permitir a modificação dos planos locais de agentes autónomos, no caso de interacções entre eles, e a identificação de situações onde essas interacções são possíveis. A negociação é por vezes utilizada, mais especificamente, para efectuar a alocação de tarefas e recursos, a resolução de conflitos, a resolução de diferenças nos objectivos de agentes e a determinação da organização estrutural numa dada situação [Muller, 1996].

Embora os processos de negociação possam ter muitos fins, uma ideia comum à maioria dos investigadores de IAD e SMA é que a negociação entre agentes tem dois objectivos essenciais: a resolução de conflitos entre agentes e a coordenação de agentes competitivos. No entanto, a negociação por vezes é também vista como um problema de linguagem, onde a sintaxe e a semântica das suas mensagens são analisadas. Em outras áreas de investigação, mais ligadas à psicologia, sociologia e gestão, a negociação é por vezes investigada numa perspectiva de análise dos comportamentos individuais e sociais dos negociadores. Desta forma, e tendo em conta as diversas perspectivas, não existe uma definição comum de negociação e uma delimitação clara do que é investigação em negociação.

No âmbito desta tese, a negociação é estudada numa perspectiva essencialmente direccionada para a coordenação de agentes autónomos competitivos. Desta forma, o capítulo inicia-se com uma breve discussão do processo de resolução de conflitos e da forma como este processo leva à realização de negociações entre humanos e/ou agentes. De seguida são apresentados os princípios da negociação computacional, sendo dado especial ênfase aos conceitos de espaço de negociação e protocolo de negociação. Dada a importância que os leilões possuem no âmbito da negociação entre múltiplos agentes, é efectuada uma descrição do conceito de leilão e dos diversos tipos de leilões. Os leilões são ainda comparados no que diz respeito às suas vulnerabilidades, vantagens e

desvantagens relativas. De seguida são analisados alguns modelos de negociação e discutidos os conceitos de estratégia e táctica de negociação. O capítulo encerra-se com uma descrição da negociação multi-atributo e de algumas aplicações práticas de negociação em SMA.

4.1 Negociação em SMA

Uma parte significativa da investigação realizada em coordenação de Sistemas Multi-Agente analisa a coordenação de agentes egoístas ("*self-interested*") e pode ser classificada sob o título de Negociação [Nwana et al., 1996]. A negociação, de um modo geral, é um processo que permite a dois ou mais participantes a obtenção de algo necessário a cada uma das partes e que o oponente possui, após um período de discussão que poderá permitir, no final, um acordo satisfatório para ambos os participantes. Outras definições de negociação podem ser:

- "Processo que procura resolver o conflito de tal modo que a solução se torne satisfatória para ambas as partes implicadas" [Serrano, 1996];
- "Processo de tomada de decisão no qual duas ou mais partes tomam conjuntamente decisões para resolver os interesses em conflito, no que respeita à distribuição de recursos escassos" [Pruitt, 1981];
- "Processo através do qual duas ou mais partes tentam reduzir ou fazer desaparecer o conflito que existe entre elas. Uma negociação com sucesso é, então, aquela que permite a ambas as partes atingir um compromisso, mesmo que para tal as partes façam concessões, comuniquem e persistam" [Wertheim, 2000];
- "Processo de tomada de decisão num contexto de interacção estratégica ou de interdependência" [Jesuino, 1998];
- "Processo para resolução de conflitos entre duas ou mais partes de tal forma que ambas ou todas modificam os seus requisitos para atingir um compromisso mutuamente aceitável" [Benson et al., 1987];
- "Processo que envolve partes, onde cada uma tem qualquer coisa que a outra quer, e que chegam a um acordo de troca, após um período de discussão" [Hindle, 1998].

Grande parte das negociações resulta da existência de um conflito entre duas ou mais partes. Da tentativa de resolução desse conflito emerge um processo de negociação multi-agente. Apesar da importância da negociação na resolução do conflito, esta não se pode entender como a solução para todo o tipo de problemas, uma vez que existe baixa probabilidade de alterar significativamente as crenças e objectivos de cada um dos agentes, através de um processo de negociação [Cunha, 1999].

4.2 Resolução de Conflitos

Os seres humanos desde sempre se viram obrigados a lidar com conflitos, enquanto indivíduos que fazem parte de uma colectividade de maior ou menor dimensão e sujeitos integrantes de estruturas do mais variado cariz, independentemente do tempo e do espaço, podendo argumentar-se que "o universo complexo das relações humanas e da interacção social é sempre potencialmente instável e conflituoso" [Friedberg, 1995].

Existem dificuldades quanto à definição do conceito de conflito social, pois as perspectivas e modelos explicativos do mesmo são abundantes e nenhuma definição deste conceito aparece como predominante [Lewicky et al., 1992]. Diferentes autores, observam e definem o conflito de diferentes formas. Deutsch [Deutsch, 1973] afirma que existe conflito quando se observa qualquer tipo de actividade inconciliável. Pruitt [Pruitt, 1981] define o conflito como um facto, onde uma das partes tenta exercer influência sobre a outra e esta lhe resiste. Para Mark e Snyder [Mark e Snyder, 1957], o conflito é um processo de interacção social, onde os valores são incompatíveis. Finalmente, Férnandez-Ríos [Férnandez-Ríos, 1986] qualifica uma situação ou relação conflituosa quando existe uma interacção entre dois ou mais participantes, com intenção de causar prejuízo ao outro, havendo assim condutas incompatíveis para obter recursos limitados, utilizando directa ou indirectamente o poder com inexistência ou ineficácia de normas. Na figura 18, é apresentado um diagrama geral das causas e consequências do conflito.

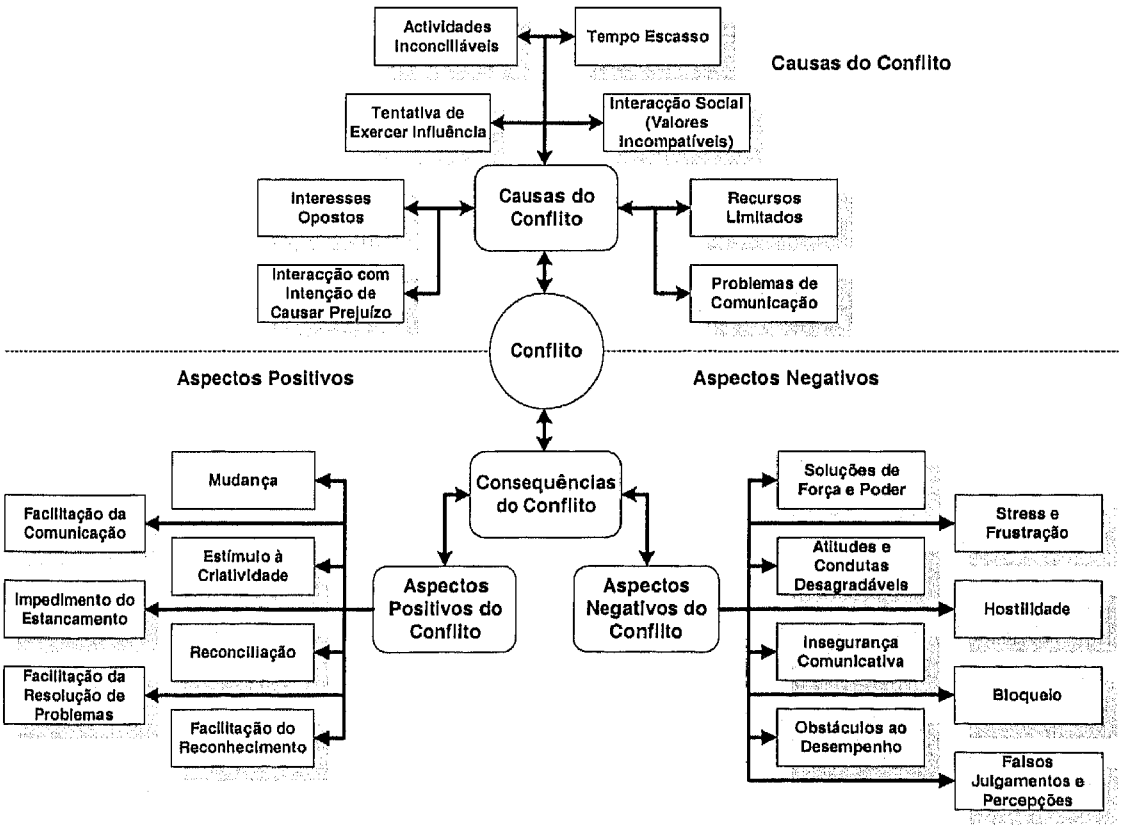

Figura 18: Diagrama das Causas e Consequências do Conflito.

A existência de interesses opostos é a principal razão para se gerar o conflito, mas a esta podem-se juntar:

- **Competição por Recursos Escassos.** Isto leva muitas vezes à competição sem regras bem definidas, uma vez que esses recursos podem ser fundamentais para qualquer das partes;
- **Tempo Escasso**. Em muitos casos, determinada negociação sobre determinado assunto, com aquele conjunto de objectivos só faz sentido num determinado espaço temporal. Isto pode levar as partes a terem de se confrontar com sérias restrições temporais e ansiedade, o que pode conduzir a uma maior susceptibilidade não havendo, portanto, espaço para a tolerância e para uma negociação aberta e fundamentada nos objectivos iniciais;
- **Problemas de Comunicação.** Muitas vezes basta existir um pequeno mal entendido no que diz respeito ao significado de uma mensagem proferida por uma das partes, para que logo ali surja um impasse ou uma situação de conflito.

Sendo estas as principais ou mais frequentes, não a totalidade, de causas para a ocorrência de uma situação de conflito, será necessário que as partes tenham a habilidade de contornar ou ultrapassar estas situações.

Deutsch [Deutsch, 1973], estabeleceu que o conflito possui aspectos negativos e positivos. Os aspectos negativos incidem no fomentar a insegurança comunicativa, a estimulação de soluções de força e poder, o desenvolvimento de atitudes hostis e a criação de falsos julgamentos e percepções. Por outro lado, os aspectos positivos podem constituir facilidade de comunicação, promoção do reconhecimento da legitimidade e interesses e facilitação da reconciliação dos interesses dos indivíduos através da sua resolução. Entre os aspectos mais destrutivos do conflito, Pruitt e Rubin [Pruitt e Rubin ,1986] salientam o escalamento, isto é, as pessoas lidam com o conflito através da rivalidade, onde cada indivíduo tenta sair-se bem à custa do outro indivíduo. Esta noção de escalamento do conflito é bastante relevante para os agentes de *software*.

Os aspectos negativos que derivam das situações conflituais, estimulam soluções de força, percepções, atitudes e condutas desagradáveis entre indivíduos ou grupos [Serrano, 1996b]. Robbins [Robbins, 1987], chega à conclusão que um nível muito baixo ou excessivamente elevado de conflito cria obstáculos ao desempenho. O nível excessivo pode provocar *stress*, frustração, hostilidade e bloqueio das aspirações de cada uma das partes. Contrariamente, o nível óptimo, pode produzir mudança, estimulando a criatividade e impedindo a estagnação.

Deutsch [Deutsch, 1990], afirma que os conflitos não podem ser reprimidos por muito tempo nem postos de parte. Assim, mostra um conjunto de ideias para o desenvolvimento de conhecimentos e aptidões na resolução construtiva de conflitos:

- **Analisar o tipo de conflito.** Analisar se o conflito é de soma nula, onde uma

parte ganha e a outra perde; de motivos mistos, onde ambas as partes ganham ou perdem; ou cooperativo em que ambas podem ganhar ou ambas podem perder. É importante ter conhecimento do tipo de conflito, pois os diferentes tipos requerem diferentes estratégias e tácticas;

- **Encarar o conflito de uma forma razoável e flexível.** Facilitando desta forma o confronto entre as partes envolvidas, respeitando os seus próprios interesses e os da outra parte;
- **Considerar que as posições podem ser opostas, mas os interesses não.** Fisher refere que quando as pessoas revelam os seus interesses, existe uma maior possibilidade de encontrar uma solução que satisfaça as partes [Fisher et al., 1993];
- **Explorar os interesses partilhados por ambas as partes.** Isto aumenta a empatia e ajuda a solucionar os problemas. Se existe muita desconfiança, é útil dispor de uma terceira parte, designada mediador, para ajudar no processo de exploração;
- **Definir bem o conflito a ser resolvido de modo cooperativo.** Primeiro deve-se diagnosticar-se claramente o problema, seguindo-se a procura criativa de novas opções de gestão do conflito;
- **Analisar bem as propostas dos oponentes.** Devem interpretar-se correctamente e analisar com cuidado as propostas dos oponentes;
- **Comunicar de forma compreensível com os oponentes.** Comunicar com clareza, evitando interpretações subjectivas, de modo a ser bem compreendido. A análise correcta das comunicações dos oponentes e a comunicação compreensível com esses oponentes é essencial para a resolução de conflitos;
- **Estar atento às distorções perceptivas, aos juízos erróneos, às tendências para o enviesamento e a metodologias de raciocínio estereotipado.** Estas desenvolvem-se em ambas as partes no desenrolar do conflito. É necessário possuir capacidade de adaptação no decurso do processo de resolução de conflitos;
- **Saber como responder em diferentes tipos de situações conflituais.** A este respeito, Deutsch [Deutsch, 1990] define cinco dimensões para responder a uma situação de conflito, que são: afastamento do conflito/excessivo envolvimento; duro/suave; rígido/à vontade; intelectual/emocional; em espiral/minimizador.

Deutsch [Deutsch, 1990] frisa que a perseverança de um indivíduo que está habilitado a preocupar-se com os oponentes e a ser justo, poderá ser um impedimento à resolução. É necessário perceber que os oponentes são entidades egoístas ("*self-interested*") e que não se pode assumir que possuam um comportamento cooperativo ou uma noção de bem social.

Como anteriormente foi referido, o escalamento do conflito é um fenómeno que tende a reproduzir-se de forma gradual, "... é frequentemente o resultado de uma espiral de conflito (uma forma de círculo vicioso) na qual cada parte reage litigiosamente à recente acção litigiosa da outra parte" [Pruitt e Rubin, 1986]. Pruitt e Gahagan [Pruitt e Gahagan, 1974] propõe três modelos que englobam a maior parte das teorias de escalamento de conflitos: o modelo da espiral de conflito, o modelo da agressão-defesa e o modelo da mudança estrutural.

Sucintamente, o modelo da espiral de conflito sustenta que o fenómeno em análise resulta de um círculo vicioso de acção-reacção, em que cada lado responde ao outro, criando assim um afrontamento [Pruitt e Rubin, 1986]. Consideram-se dois tipos de espiral de conflito: a defensiva (onde cada lado em litígio toma a defesa para se auto-proteger de possíveis ameaças resultantes das acções de protecção da outra parte) e a retaliatória (quando existem acções opostas em que cada parte pune a outra). Este modelo flui em ambas as direcções, onde cada parte reage às acções da outra, e cada reacção é mais intensa e severa do que a anterior.

O segundo modelo, o da agressão-defesa, contrariamente ao anterior, flui apenas numa direcção. As duas partes, agressor-defensor, são distintas, sendo o agressor que usualmente inicia o processo de conflito com a outra parte, devido à insistência de concretização do seu objectivo ou conjunto de objectivos [Pruitt e Rubin, 1986]. Como o agressor é uma parte que quer mudar as coisas no rumo dos seus objectivos ou interesses e o defensor é a outra parte que tenta não aceder a essa mudança, o primeiro inicia o processo com estratégias brandas de competição, mas, se estas não resultam, aplica estratégias mais rígidas até conseguir atingir os seus interesses. Por outro lado, o defensor reage unicamente, escalando os seus esforços como resposta, até que o agressor ganhe ou desista [Pruitt e Rubin, 1986].

O modelo da mudança estrutural argumenta que o conflito escalado produz resíduos (na forma de mudanças nas partes e comunidades às quais pertencem) que encorajam o afrontamento e diminuem os esforços para alcançar uma solução [Pruitt e Rubin, 1986].

Frequentemente verifica-se o fracasso das estratégias de afrontamento, isto porque os conflitos paralisam a acção e mantêm-se de forma inútil, criando novas situações que incrementam os custos económicos, sociais e/ou pessoais [Cunha, 1999].

Bercovitch [Bercovitch, 1984], refere que os conflitos serão resolvidos, de um modo geral, se se alcançou um resultado apreciável; o comportamento conflituoso terminou; ou se foi atingido um acordo satisfatório de valores e recursos. Refere ainda, que existem três tipos de resultados do conflito: afastamento ou retirada; conquista, dominação ou imposição; e acordo de compromisso.

Na óptica de Greenhalgh, existem três alternativas que são capazes de resolver o conflito [Greenhalgh, 1987]: resolução (reajustamento cognitivo acompanhado por necessidade do acordo), dominação (ignorar as vontades da outra parte, sendo dominador no acordo) e

negociação (implicando a obrigatoriedade de alcançar o acordo em conjunto sobre o assunto em disputa). O mesmo autor considera que a resolução é uma alternativa não praticável no conflito interpessoal de longa duração, assim como a dominação é uma abordagem instável, ao contrário da negociação, que é o meio mais distinto de lidar com o conflito, principalmente nas organizações e em sistemas compostos por múltiplos indivíduos em que não existe uma relação hierárquica evidente entre eles, como é o caso usual dos Sistemas Multi-Agente.

De uma forma mais abrangente podemos afirmar que existem cinco formas de resolução do conflito, que são:

- **Competição.** Neste tipo de resposta ao conflito, para cada uma das partes, apenas importa a satisfação das suas necessidades e não a realização das necessidades da outra parte;
- **Colaboração.** Aqui as partes têm em consideração as necessidades de ambas, ou seja, é importante que ambos os intervenientes saiam satisfeitos do processo de negociação;
- **Compromisso.** A satisfação das necessidades de ambas as partes é moderadamente importante;
- **Afastamento.** Este tipo de resolução de conflito enuncia que as partes não se interessam nem por satisfazer as suas necessidades nem as dos outros, ou seja, simplesmente evitam o conflito;
- **Acomodação.** Resulta da constatação que não é tão importante para o agente a satisfação das necessidades relacionadas com o conflito em causa como é importante para a outra parte. Desta forma, a acomodação não conduz a grande prejuízo e o benefício que gera na outra parte pode ser a médio prazo recompensador.

Em geral, os negociadores de sucesso assumem uma posição de colaboração, de forma a que ambas as partes tenham a sensação que ganharam, uma vez que a negociação tende a correr melhor se ambos acharem que estão numa situação do tipo ganhar-ganhar, ou seja, tentam satisfazer as necessidades de ambas as partes.

Independentemente do seu tipo ou causas, a existência de um conflito implica sempre a necessidade de coordenação entre os agentes envolvidos no mesmo de forma a que possa ser resolvido. A metodologia de coordenação utilizada mais frequentemente na resolução de conflitos, sobretudo em conflitos entre agentes autónomos, é a negociação. Neste processo, ambas as partes procuram encontrar um acordo (solução) que possa ser mutuamente aceite e resolva o conflito original.

4.3 Princípios da Negociação Computacional

Tal como acontece com os humanos, quando existe a necessidade de chegar a acordo sobre vários atributos de um produto ou conseguir alguma coisa que uma outra parte possui, também os agentes fazem uso da negociação. Em sistemas multi-agente a negociação é uma das melhores formas de interacção que permite a um conjunto de agentes chegar a um acordo mútuo, no sentido de conciliar uma crença, satisfazer um objectivo ou plano [Beer et al., 1998]. Numa visão mais automática do processo, complementando as definições apresentadas anteriormente, por negociação automática pode entender-se:

- "Processo em que participam dois ou mais agentes que tentam atingir objectivos que não conseguem, ou preferem não conseguir, atingir por sua conta" [Parsons et al., 1998];
- "Processo pelo qual é tomada uma decisão conjunta por duas ou mais partes. Primeiro as partes verbalizam ofertas contraditórias e depois avançam para o acordo" [Faratin et al., 1999];
- "Processo pelo qual grupos de agentes comunicam uns com os outros para tentar chegar a um acordo mútuo aceitável para todos sobre um determinado assunto" [Lomuscio et al., 2000].

A negociação fortalece as tentativas de cooperação e coordenação, tanto entre agentes como entre humanos. Permite atingir acordos e cooperação entre agentes egoístas, ou seja, agentes que não possuem uma noção de bem social. Para que um agente consiga influenciar um outro participante, este precisa de ser convencido a actuar de uma determinada forma. Assim sendo, a melhor maneira de atingir este estado é utilizando a negociação, ou seja, fazer propostas, oferecer concessões, opções de troca, e acreditar que se chegará a um acordo aceitável [Parsons et al., 2001].

Pode considerar-se que a negociação automática é composta por três constituintes principais [Parsons et al., 2001]:

- **Protocolos de Negociação.** Conjuntos de regras que governam a interacção entre os agentes participantes na negociação. Existem vários protocolos de negociação (por exemplo, na forma de leilão existem os leilões Inglês, Holandês, Vickrey, entre outros), mas não existe um protocolo melhor do que outro. O que acontece, é que o protocolo é escolhido de acordo com o que os participantes acharem que melhor se adequa ao tipo de negociação que pretendem fazer;
- **Objectos de Negociação.** Conjuntos de atributos sobre os quais se pretende chegar a acordo. O objecto de negociação pode conter um único atributo, como o preço, por exemplo, mas pode também conter uma variedade de atributos, como o preço, a quantidade, os prazos, condições de entrega, entre outros. Num nível

superior, os agentes têm a flexibilidade de alterar os valores dos atributos do objecto de negociação, isto é, os participantes podem fazer contrapropostas para conseguir que o acordo final se ajuste melhor aos seus objectivos em negociação;

- **Modelos de Tomada de Decisão do Agente.** Ferramentas de tomada de decisão que os participantes utilizam, de modo a actuar de acordo com o protocolo de negociação, tendo em vista atingir os objectivos propostos. O refinamento do modelo, assim como o âmbito das decisões que têm de ser tomadas, são influenciados pelo protocolo utilizado, pela natureza do objecto de negociação, e pelo número de operações que pode ser realizado sobre ele.

A importância destes três pontos varia de acordo com o tipo de negociação, os agentes participantes e o ambiente em que esta está inserida.

4.3.1 Tipos de Negociação

Existem diversos métodos de negociação, que se podem dividir em três tipos principais (figura 19) [Wertheim, 2000] [Reis, 2002d]:

- **Negociação Distributiva.** Este tipo de negociação é também designada de competitiva, ou ganhar-perder. Neste tipo de negociação uma parte ganha e a outra perde, ou seja, quanto mais uma parte obtém menos sobra para a oponente. Aqui, os interesses da outra parte não são importantes e existe um completo conflito de interesses. Apenas importa o que o próprio agente pode obter, tendo como estratégia a competição e a tentativa de manipulação do oponente. Esta negociação também se apelida de "reclamação de valor" uma vez que o seu objectivo é aumentar o próprio valor do agente diminuindo o do oponente. A negociação é vista como um jogo de soma nula, onde os ganhos de um agente correspondem às perdas do outro, havendo um esforço em maximizar ganhos e minimizar perdas;
- **Negociação Integrativa.** Este tipo de negociação é também designado por colaborativa, ou ganhar-ganhar, em que ambas as partes podem ganhar e a principal preocupação é a maximização dos resultados para ambas as partes. Este tipo de negociação favorece o surgimento de novas ideias, de novas propostas e a troca de informação entre os participantes. Existe simultaneamente uma preocupação pelos interesses próprios e pelos interesses da outra parte, não sendo a negociação uma mera troca de concessões. Desta forma, as estratégias dominantes são a cooperação entre as partes, a partilha de informação e a resolução conjunta dos problemas. A este tipo de negociação também se pode chamar de "criação de valor", uma vez que o objectivo é que as partes terminem a negociação com a sensação de que adquiriram mais valor do que no início. A negociação é vista como um jogo de soma positiva, ou seja, existe a crença de

que há uma forma de conceder benefícios à outra parte sem que tal constitua uma perda para si próprio, mas antes também um beneficio;

- **Negociação Mista.** Os dois tipos de negociação anterior não vivem isolados um do outro, pelo que podem existir situações em que se encontram simultaneamente elementos da negociação distributiva e da negociação integrativa. A este tipo de negociação dá-se o nome de negociação mista. Como exemplo temos a negociação de um preço entre uma empresa e um cliente sobre um determinado produto ou serviço onde, em alguns níveis, os interesses da empresa se confrontam com o do cliente. Por exemplo, a empresa pode querer um preço elevado e o cliente deseja obter um preço baixo. No entanto, noutros níveis os interesses podem coincidir, ou seja, ambas as partes querem que as suas necessidades sejam satisfeitas e que, tanto a empresa como o cliente, saiam felizes da negociação. Por exemplo, pode interessar a ambos a realização de futuros negócios pelo que o bem da empresa interessa ao cliente e vice-versa.

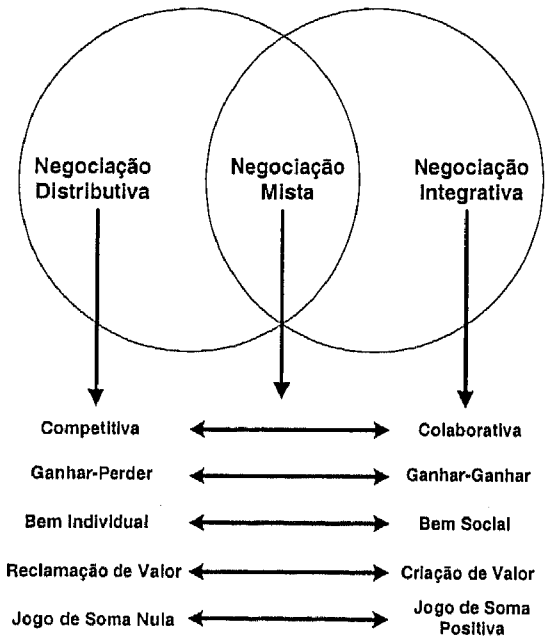

Figura 19: Tipos de Negociação

Os principais pontos para se conseguir levar a cabo uma negociação integrativa ou ganhar-ganhar, podem ser resumidos como:

- Planear a negociação e ter uma estratégia concreta, ou seja, ser claro naquilo que é importante para o agente;
- Separar os oponentes dos problemas, não sobrevalorizando pequenas questões que não contribuem de forma alguma para o objectivo da negociação;
- Dar ênfase a soluções que possam conduzir a situações ganhar-ganhar, como sendo a criação de opções para um ganho mútuo gerando uma variedade de possibilidades antes de decidir o que fazer;

- Considerar a situação da outra parte, ou seja, cada uma das partes deve tentar colocar-se no lugar do oponente (criando e actualizando um modelo do comportamento desse oponente) e pensar como ele irá agir, quais são os seus objectivos, necessidades e até mesmo interesses subjacentes;
- Cada parte deve conhecer a sua melhor alternativa no caso de fracasso da negociação. É importante perceber qual será o resultado para o agente no caso de não conseguir atingir todos os seus objectivos iniciais;
- Utilizar toda a informação disponível durante o processo de negociação. Analisar correctamente o fluxo de informação, uma vez que a negociação é um processo iterativo, composto por uma série de eventos como são a troca de propostas e contrapropostas. Esta análise poderá conduzir a retirar o máximo de informação da outra parte e perceber quais poderão ser os seus objectivos e estratégia.

É sem dúvida importante que ambas as partes desenvolvam um processo de negociação que se apoie nas ideias defendidas pela negociação integrativa, uma vez que, com este tipo de negociação, todos ganham alguma coisa. A utilização de princípios da negociação integrativa pode constituir uma boa base para a definição de estratégias para as negociações distributivas.

4.3.2 Fases do Processo de Negociação

As fases do processo de negociação, embora resultantes no mesmo tipo de processo, numa metodologia que permite aos participantes a concretização dos seus objectivos e resolução dos conflitos existentes, diferem no seu nome de acordo com os diferentes autores.

Segundo Douglas [Douglas, 1962], a primeira fase do processo de negociação consiste na definição do conteúdo e limites da negociação, seguida de uma fase denominada dinâmica negociadora que consiste fundamentalmente na apresentação das propostas e contrapropostas e, por último, uma fase terminal de resolução e acordos.

Outra definição das fases da negociação é apresentada por Benson et al. [Benson et al., 1987] que divide o processo em 4 fases: Preparação, Argumentação, Proposta, Encerramento (figura 20).

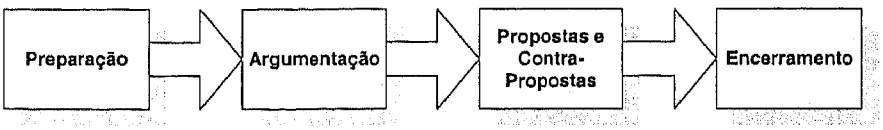

Figura 20: Fases do Processo de Negociação

A fase de preparação corresponde à recolha de informações sobre a negociação e sobre os seus participantes. A análise de processos de negociação semelhantes, em que os agentes participantes tenham entrado, revela-se bastante útil neste contexto. Um agente mal

preparado pode apenas reagir aos acontecimentos despoletados pelo processo, ou seja, acaba por nunca os proporcionar, e mais cedo ou mais tarde acaba por demonstrar a sua má preparação para a negociação. Um bom agente negociador deverá saber o que é que tem de ser realizado, como atingir o(s) objectivo(s) pré-definido(s), conhecer os oponentes e as suas possíveis estratégias e objectivos e possuir flexibilidade suficiente que lhe permita aplicar diferentes tácticas durante o processo de negociação. É exactamente na fase de preparação que se definem quais os objectivos a atingir, assim como a forma de chegar até eles. Bastante útil neste contexto poderá ser a elaboração de uma lista definindo quais os objectivos que têm necessariamente de ser satisfeitos para a negociação poder ser considerada satisfatória e uma lista contendo os objectivos que não são essenciais mas que são desejáveis. Estes últimos objectivos serão a base para avaliar a qualidade da solução obtida.

A fase da argumentação caracteriza-se pela exposição, por ambas as partes, das suas razões e tentativa de persuasão do oponente. A argumentação não se fica apenas pelos momentos iniciais do contacto das partes, mas será um processo iterativo. Esta fase é uma oportunidade, não um obstáculo, uma vez que pode dar aos intervenientes o acesso a todo o tipo de informação relacionada com os objectivos do oponente, as suas obrigações e intenções, assim como explorar os pontos que separam uma parte da outra, as atitudes, os interesses e inibições do oponente. Pode-se deduzir que um interesse leva o oponente a dizer "sim", assim como uma inibição o leva a dizer "não". É então nesta fase que os intervenientes têm a oportunidade de testar todas as suposições calendarizadas na fase de preparação, acerca da outra parte interveniente na negociação. É portanto, nesta fase, importante utilizar a argumentação como meio de conhecimento da outra parte, onde uma troca de informação pode revelar pontos fulcrais para os benefícios daquela negociação, como sendo, possíveis ajustamentos que a outra parte está disposta a fazer para o respectivo atingir dos objectivos.

A fase das propostas e contrapropostas tem como finalidade a apresentação de um ponto ou conjunto de pontos a ser negociado, vindo estes sob a forma de propostas, já que não se pode negociar as necessidades, interesses ou inibições da outra parte, mas sim e apenas as propostas. Quando se chega a esta fase, as partes podem começar por declarar quais as pré-condições, que permitirão enfraquecer o poder de negociação do oponente, a serem abrangidas para que se obtenha um bom resultado do processo de negociação. A fase de propostas serve, também, para fazer a negociação avançar, uma vez que se pode perder muito tempo na fase de argumentação e acabar por se perder o controlo do tempo e do que realmente está a ser negociado. Daí que, se uma das partes fizer uma proposta, avança-se no sentido do final da negociação, uma vez que já se ambiciona o acordo e a respectiva realização dos objectivos. Habitualmente, esta fase é composta por um conjunto de propostas e contrapropostas que, em negociações electrónicas, seguem um determinado protocolo de negociação pré-estabelecido.

A fase final da negociação é o encerramento. Esta fase permite aos intervenientes avaliar se o resultado é favorável ou não para os seus interesses. Aqui ainda se troca informação e definem-se propostas com concessões mútuas que permitam acordos definitivos. Esta fase durará até que as partes cheguem a um acordo, e então fazem-se os últimos acertos, nomeadamente no que diz respeito a pormenores específicos da negociação, como sejam a quantidade, preço, data de entrega, data de pagamento, etc. Poderá ainda haver uma ruptura na negociação, caso, por exemplo, os objectivos mínimos não sejam satisfeitos. Neste caso, dá-se por terminado o processo ou eventualmente retorna-se para a fase de argumentação e começa-se um novo ciclo de negociação.

4.3.3 Espaço de Negociação

A definição do espaço de negociação é importante porque permite identificar os possíveis parâmetros que podem ser utilizados para classificar um determinado mecanismo de negociação [Lomuscio et al., 2000]. Esta classificação é um passo importante para desenvolver sistemas de negociação automática mais evoluídos, uma vez que define e delimita o espaço designado para as interacções dos agentes. Desta forma, o espaço de negociação é o conjunto de todos os possíveis jogos de negociação [Lomuscio et al., 2000].

Existem três componentes básicos que são importantes quando se projecta um sistema de negociação automática: protocolo de negociação, objecto de negociação e estratégia de negociação. O protocolo de negociação, como foi analisado anteriormente, apresenta as "regras de combate" entre os participantes na negociação, ou seja, define as circunstâncias sobre as quais as interacções entre os agentes acontecem: quais os acordos que podem ser realizados e em que momentos e com que ordenação podem ser efectuadas as ofertas. O objecto de negociação são os atributos sobre os quais se pretende chegar a acordo.

A estratégia de negociação dos agentes consiste na especificação de planos que vão definir a sequência das acções que o agente planeia realizar durante a negociação, e as condições de efectivação das mesmas. Estas acções podem incluir argumentações, propostas e respostas. Existem diversas estratégias compatíveis com um determinado protocolo e é usual que uma estratégia que funcione bem, utilizando um dado protocolo, e um conjunto de adversários, não tenha a mesma eficácia para outro protocolo e/ou conjunto de adversários. Da união do protocolo de negociação com o objecto de negociação e a estratégia de negociação surge o mecanismo de negociação.

4.3.4 Parâmetros do Espaço de Negociação

O projecto de um cenário de negociação apropriado depende, em grande parte, de um conjunto de parâmetros, o qual pode servir para gerar um espaço de possíveis cenários de negociação [Lomuscio et al., 2000]. Os parâmetros que caracterizam o processo de

negociação encontram-se dentro das seguintes características (figura 21): cardinalidade da negociação; características do agentes; ambiente e características dos produtos/bens em negociação; parâmetros dos eventos; parâmetros da informação e parâmetros de alocação. O mecanismo de negociação resulta da combinação e definição de todas estas características.

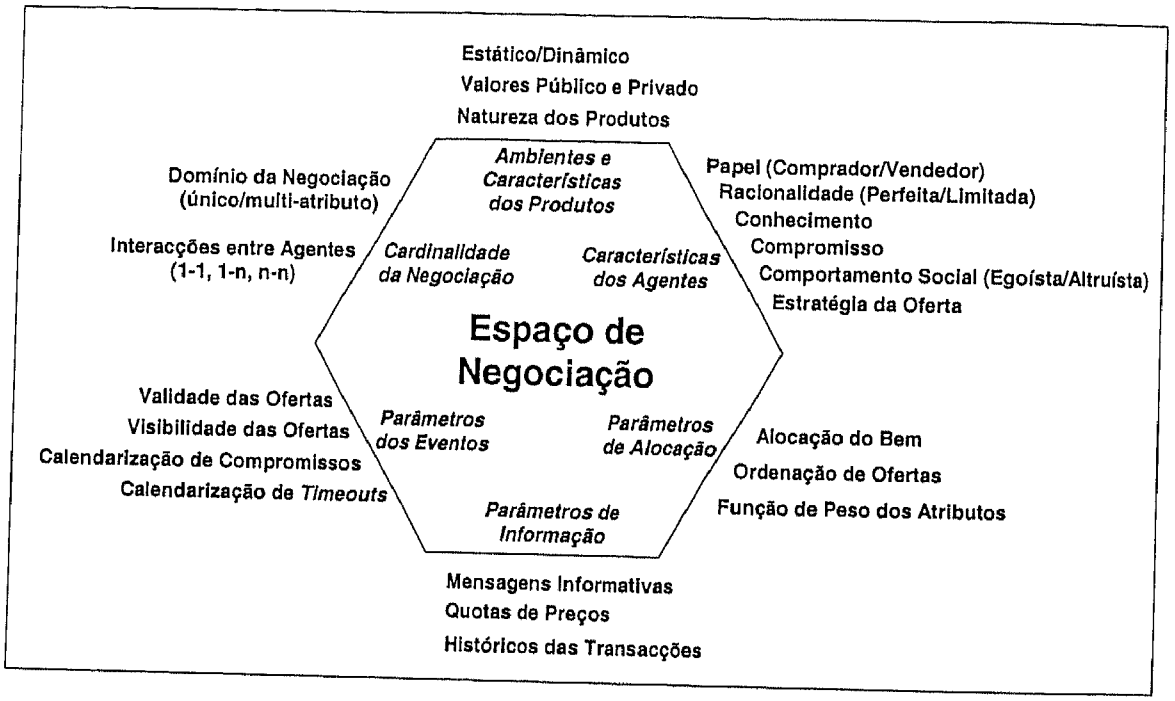

Figura 21: Parâmetros do Espaço de Negociação [Reis, 2002d]

4.3.4.1 Cardinalidade da Negociação

A cardinalidade pode ser dividida em dois tipos: segundo o domínio da negociação (único-atributo ou multi-atributo) e segundo as interacções entre agentes (um-para-um, muitos-para-um ou muitos-para-muitos).

O domínio da negociação é o conjunto de tuplos sobre o qual os agentes negoceiam. Segundo Beer [Beer et al., 1898] estes tuplos são os objectos da negociação, isto é, representam o conjunto de atributos sobre o qual se deve chegar a acordo. Os elementos desses objectos da negociação correspondem a atributos como o preço, a qualidade, a quantidade, as garantias, datas de entrega, entre outros.

Quando se trata de um processo de negociação único-atributo, onde, por exemplo, o atributo em causa será o preço, os objectos da negociação são singulares. No que diz respeito à negociação multi-atributo, os diferentes atributos podem estar relacionados por uma função de utilidade pública, anteriormente acordada. Esta função torna o processo de formulação de ofertas mais fácil, uma vez que cada agente é capaz de estimar a função de utilidade de cada um dos agentes envolvidos no processo de negociação. No entanto, usualmente não existe a possibilidade de definição de uma tal função e como tal a negociação multíplo-atributo é muito complexa.

As interacções entre os agentes podem ser classificadas em relação ao número de agentes intervenientes na negociação. Na negociação um-para-um, um agente negoceia exactamente com um outro agente, enquanto que na negociação muitos-para-um, vários agentes negoceiam com apenas um agente. Neste tipo de interacção um agente desempenha usualmente papel de vendedor, enquanto que os restantes agentes desempenham o papel de compradores ou vice-versa.

Por último, a negociação muitos-para-muitos, onde vários agentes negoceiam com vários outros agentes, é o cenário mais complexo. Neste tipo de cenário podem utilizar-se por exemplo, os leilões duplos (secção 4.4).

Apesar desta classificação, nem sempre se pode determinar a cardinalidade das interacções apenas pelo número de agentes, isto porque é possível ter um vasto conjunto de agentes e manipulá-los de modo a que o processo de negociação assente numa cardinalidade do tipo um-para-um. A estruturação do Sistema Multi-Agente e o protocolo de negociação influenciam deste modo a cardinalidade da negociação.

4.3.4.2 Características dos Agentes

Os agentes que participam no processo de negociação devem ser capazes de classificar as suas preferências, por isso eles podem avaliar e escolher entre diferentes negócios. A caracterização dos agentes pode ser efectuada segundo as vertentes:

- **Papel.** Um agente pode desempenhar, numa negociação, o papel de vendedor, comprador ou simultaneamente os dois papéis.
- **Racionalidade.** Pode dividir-se em dois tipos: perfeita ou limitada. A suposição de racionalidade perfeita leva à suposição da capacidade dos agentes realizarem, arbitrariamente, um elevado número de cálculos num tempo constante. Na maioria dos cenários, os agentes não têm a capacidade ou poder computacional para executar esses cálculos, sendo então forçados a realizar a oferta ou desistir com base em cálculos efectuados num tempo limitado. Desta forma, os modelos de negociação que utilizam racionalidade limitada são mais realistas.
- **Conhecimento.** Os agentes têm conhecimento sobre os bens que lhes interessam e pelos quais estão dispostos a ofertar, assim como podem ter algum conhecimento sobre o valor que os outros agentes atribuem a esses e outros bens. De qualquer forma, a existência de informação privada, é muito importante para a criação da estratégia de oferta dos agentes. Desta forma, a existência de objectivos individuais e a avaliação da função de utilidade do oponente, podem ser pontos importantes para a definição da estratégia do agente.
- **Compromisso.** Podem existir vários níveis de compromisso. Por exemplo, depois de efectuar uma oferta, o agente pode ou não ser obrigado a adquirir o produto, no caso do vendedor aceitar a oferta. Se tal estiver definido no protocolo de

negociação, um dado agente pode também ser obrigado a parar as ofertas para bens similares até que seja recebida uma mensagem de aceitação ou uma contraproposta.

- **Comportamento Social.** Os agentes podem agir como entidades egoístas e individuais, como entidades altruístas de uma sociedade, ou mantendo um equilíbrio entre estes dois comportamentos. Existe também a possibilidade dos agentes formarem coligações ou se associarem em equipas, que podem ser uma ferramenta importante na obtenção de melhores acordos. Qualquer que seja a atitude do agente, o importante é que, ou como entidades distintas ou como uma equipa, eles consigam atingir a solução mais conveniente de acordo com os objectivos propostos.
- **Estratégia de Oferta.** A estratégia de ofertas de um agente é a componente que decide sobre o estabelecimento ou aceitação de ofertas, realização de contrapropostas ou abandono da negociação.

Embora os parâmetros sejam relativamente independentes, existe uma certa relação prática entre eles. Por exemplo, embora a estratégia da oferta seja conceptualmente independente dos outros parâmetros, é razoável relacioná-la com a racionalidade, o conhecimento, o compromisso e o comportamento social de um agente. De um modo geral, supõe-se que, sendo um agente individualmente racional, a sua estratégia de ofertas será não aceitar compromissos que o façam abandonar a negociação com uma satisfação inferior do que no início do processo [Lomuscio et al., 2000].

4.3.4.3 Ambientes e Características dos Produtos

Tal como discutido na secção 2.2, e neste caso relativo aos ambientes de negociação, estes podem ser estáticos ou dinâmicos. Um ambiente estático é aquele em que as variáveis, como o preço, se mantêm constantes ao longo do tempo, e um ambiente dinâmico é aquele em que essas mesmas variáveis mudam os seus valores ao longo do tempo. Deste modo, um agente inserido num ambiente dinâmico poderá ver a sua função de utilidade ser afectada. A função de utilidade de um agente representa as suas preferências, daí que, enquanto num ambiente estático se imagina um agente que não aprende durante o processo e mantém a sua função de utilidade até ao fim, num ambiente dinâmico este comportamento seria o menos adequado para atingir resultados positivos.

As características dos bens também influenciam a definição do protocolo de negociação. Essas características são:

- Valor público e valor privado dos produtos;
- Natureza dos produtos.

Os agentes podem avaliar de forma diferente um bem. Podem avaliar o bem de acordo com o seu valor privado ou com o valor que acreditam que os outros agentes atribuem a

esse bem. Normalmente, o valor público e o valor privado funcionam em conjunto, isto porque, quando se trata da compra de um bem para uso privado, algumas vezes pode existir a intenção de voltar a vender o bem. Resulta deste facto que é importante conhecer a avaliação pública desse mesmo bem. Um exemplo típico é a aquisição de um veículo automóvel em que, embora o valor privado seja muito importante na aquisição do bem, usualmente o comprador irá vender o automóvel após alguns anos, pelo que o valor público é também importante.

Quanto à natureza do objecto da negociação, ou seja o conjunto de atributos dos bens sobre os quais se pretende chegar a acordo [Jennings et al., 2000], este pode ser um conjunto de bens discreto ou contínuo. As negociações mais usuais realizam-se sobre bens discretos. Uma negociação com prazos, como por exemplo, prazos de entrega ou prazos de contratos, é uma negociação sobre um número não definido de objectos, isto porque, não se sabe quantos contratos se irão negociar, num dado limite de tempo.

4.3.4.4 Parâmetros dos Eventos

O protocolo de negociação é, em grande parte, influenciado pelo rumo que as ofertas e outros eventos levam, durante o processo de negociação [Lomuscio et al., 2000]. Os eventos podem ser distinguidos através de:

- **Validade das Ofertas.** Especifica uma parte importante do protocolo de negociação: o critério para validar as ofertas. Para que as ofertas sejam válidas, têm de ser feitas no instante de tempo correcto e o seu valor, por vezes, deve satisfazer algumas restrições. Estas regras prendem-se com o protocolo de negociação que está a ser utilizado.
- **Visibilidade das Ofertas.** As ofertas podem ser transmitidas por mensagens privadas entre vendedor e comprador, ou transmitidas para todos os agentes. Caso seja possível a formação de coligações, as ofertas podem ser transmitidas em mensagens enviadas unicamente para um subconjunto de agentes.
- **Calendarização de Compromissos.** Um compromisso é o evento que permite, temporariamente, um encontro entre o comprador e o vendedor. Os compromissos podem ser calendarizados aleatoriamente, ou sucedendo a outros eventos, por exemplo, a primeira oferta que iguala o pedido do comprador.
- **Calendarização de *Timeouts*.** Determina o encerramento da negociação e transforma os compromissos assumidos em compromissos definitivos.

Os compromissos e os *timeouts* também dependem dos parâmetros de alocação, que serão caracterizados em seguida.

4.3.4.5 Parâmetros de Informação

No decurso do processo de negociação, para além do envio de mensagens contendo ofertas, são usualmente trocadas, pelos participantes, mensagens contendo informação. Essa informação pode auxiliar o comprador e o vendedor a atingirem um acordo. Os participantes podem ainda veicular informação, no sentido de reduzir as interferências produzidas por outros agentes sendo desta forma benéficas, no que diz respeito, por exemplo, à redução do tempo computacional dos agentes.

Existem vários tipos de mensagens informativas, tais como quotas de preço e históricos das transacções. As quotas são geradas por futuros compradores pedindo um preço indicador ao vendedor, antes de se iniciar o processo de negociação. Podem ser úteis para ambas as partes e permitir reduzir o tempo de negociação. O histórico de transacções similares pode também ser pedido pelos compradores, ou ser facultativamente fornecido pelo vendedor, de modo a dar maior credibilidade à informação fornecida aos potenciais agentes compradores. Este histórico das transacções, que é de grande relevância nas negociações entre humanos, pode formar a base para a negociação entre agentes computacionais baseada em processos de argumentação.

4.3.4.6 Parâmetros de Alocação

Os parâmetros de alocação apenas se aplicam em negociações do tipo muitos-para-um e muitos-para-muitos [Lomuscio et al., 2000]. Estes parâmetros decidem qual o agente a quem será alocado o bem (e que irá, desta forma, adquirir o bem) quando mais do que um agente se mostra interessado pelo mesmo. É importante referir que, se a negociação for multi-atributo, para que os parâmetros de alocação possam funcionar, terá de existir uma função aceite por todos os participantes na negociação, que "pese" os diferentes atributos, produzindo um valor de utilidade para cada um. Sem isto, não seria possível ordenar as ofertas em conformidade com a política de alocação [Lomuscio et al., 2000].

4.3.5 Protocolos de Negociação

O protocolo de negociação é o conjunto de regras que governam o processo de negociação, ou seja, a coordenação do comportamento dos agentes durante a negociação é forçada pelas regras normativas do protocolo de negociação [Faratin et al., 1999]. Este conjunto de regras compreende [Parsons et al., 2001]:

- A definição dos tipos de participantes autorizados a entrar no processo de negociação;
- Os estados da negociação admissíveis;
- Os eventos que causam as mudanças de estado, como por exemplo: envio de uma oferta, aceitação de uma proposta, rejeição de uma proposta, etc.; e

- As acções válidas dos participantes num determinado estado, ou seja, que mensagens podem ser enviadas, por quem, para quem e em que estado.

O protocolo de negociação pode também ser definido como um diagrama de transição de estado, o qual apresenta os vários estados possíveis que um agente pode representar durante a negociação, assim como, as transições possíveis entre estados que um agente está autorizado a realizar [Parsons et al., 1998].

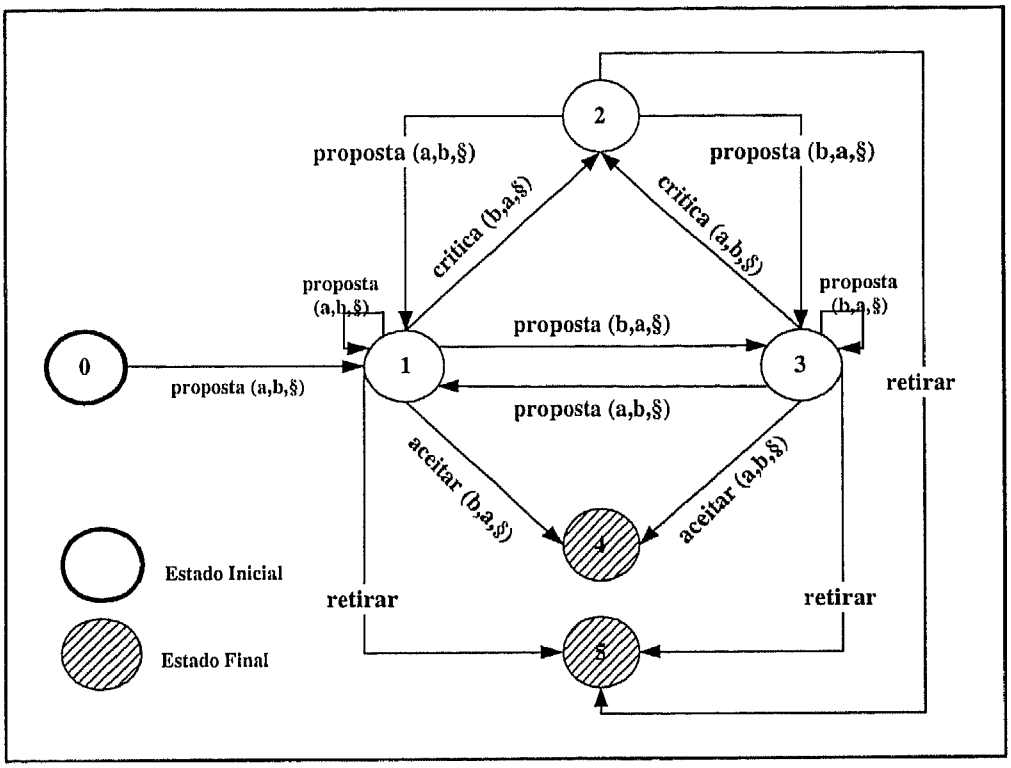

Figura 22: Protocolo de Negociação para Dois Agentes [Faratin et al., 1999]

A Figura 22 esquematiza um possível protocolo de negociação contendo mensagens do tipo proposta, crítica, aceitação, retirada. O protocolo contém ainda 5 estados possíveis para a negociação. O processo começa (estado 0) quando um agente *a* faz uma proposta ao outro participante (agente *b*), designada por *proposta (a,b,§)*. O símbolo § designa a proposta a ser efectuada e a sua eventual explicação. No estado 1, o agente *a* pode fazer uma segunda proposta, sem esperar pela resposta da primeira proposta, ou, por outro lado, o agente *b* pode actuar, tanto para aceitar a proposta (conduzindo ao estado 4), como para fazer uma crítica (estado 2), uma contra-proposta (estado 3) ou simplesmente retirar-se da negociação (estado 5). No caso do agente aceitar a proposta, pode novamente dar uma explicação se o desejar, e no caso de fazer uma crítica, esta também pode ser sustentada por uma razão. A realização de críticas ou contra-propostas pode continuar indefinidamente num processo iterativo. Este protocolo utiliza duas alocuções básicas, *aceitar* e *retirar*, que permitem a qualquer dos agentes finalizar a negociação obtendo ou não um acordo [Parsons et al., 1998].

BIBLIOTECA F.E. PORTO

4.3.5.1 Características do Protocolo de Negociação

Segundo Parsons et al. [Parsons et al., 2001], é possível projectar protocolos de negociação nos quais existem algumas propriedades que são consideradas desejadas. Essas propriedades são [Zlotkin e Rosenchein, 1994] [Sandholm, 1999]:

- **Sucesso Garantido.** Um protocolo garante o sucesso se assegurar que, no final, num tempo finito, um acordo é certamente atingido;
- **Maximizar o Bem-Estar Social.** Um protocolo maximiza o bem-estar social se assegurar que qualquer resultado maximiza a soma das utilidades dos participantes da negociação;
- **Eficiência de *Pareto*.** O resultado de uma negociação diz-se "*Pareto* eficiente", se não houver outro resultado global que melhore a utilidade para um dos agentes, sem ao mesmo tempo prejudicar o resultado obtido por qualquer dos outros agentes;
- **Racionalidade Individual.** A participação de um agente na negociação é individualmente racional, se o ganho da participação desse agente na negociação não for menor do que o ganho que o agente teria, se não participasse no processo. Isto significa que, deve ser do próprio interesse dos participantes entrar na negociação. A racionalidade individual dos protocolos é essencial porque sem ela não haveria incentivo para os agentes participarem na negociação;
- **Estabilidade.** Um protocolo é estável se provir todos os agentes com um incentivo para estes se comportarem de um determinado modo. O tipo de estabilidade mais conhecido é o *Nash Equilibrium* (equilíbrio de Nash[17]) [Nash, 1950]. Duas estratégias, s e s' dizem-se estar em equilíbrio de Nash se se supuser que se um agente está a usar s, então o outro agente não pode fazer melhor do que usar s', e vice-versa;
- **Simplicidade.** Um protocolo dito simples é aquele que torna a estratégia apropriada óbvia para os participantes na negociação. Isto significa que um protocolo é simples se, com a sua utilização, um participante consegue facilmente determinar a estratégia óptima;
- **Distribuição.** Um protocolo deve, idealmente, ser projectado para assegurar que não existe um único ponto de falha, como por exemplo um único árbitro, assim como, idealmente, minimiza a comunicação entre os agentes participantes na negociação.

---

[17] John Nash. Matemático Americano que, após uma vida de luta contra a esquizofrenia, venceu o prémio Nobel da Economia em 1994 essencialmente devido à sua tese de 27 páginas "Non-Cooperative Games" [Nash, 1950], escrita em 1950, onde é descrito o equilíbrio de Nash.

Estas características contribuem em grande parte para o sucesso da negociação, daí que seja importante a presença de todas elas no protocolo de negociação que for utilizado.

4.3.5.2 Esquema Genérico para Negociação Electrónica

Com este ponto pretende fazer-se uma abordagem genérica ao que pode ser um processo de negociação entre vários agentes. A negociação pode ser vista como uma pesquisa distribuída por um espaço de potenciais acordos [Parsons et al., 2001]. Numa dada negociação, os participantes são os componentes activos que determinam essa mesma pesquisa. Para melhor compreender o processo, observe-se a Figura 23.

No início do processo cada agente tem uma porção de espaço, no qual tenta fazer acordos, onde, normalmente, cada agente tem alguma forma de avaliar os pontos no seu espaço e alguma forma de utilizar essa avaliação para determinar os actuais acordos que efectua. A negociação prossegue com os participantes sugerindo pontos específicos, ou regiões, no espaço de negociação como sendo potencialmente aceitáveis. Durante o processo de negociação, os espaços de negociação de cada agente, assim como as suas funções de avaliação, podem ser alterados, isto é, podem expandir-se, contrair-se, deslocar-se, quando por exemplo, o seu ambiente muda ou porque os agentes são persuadidos para alterar os seus pontos de vista [Teixeira, 2002].

A pesquisa, ou seja a negociação, termina quando o conjunto de agentes participantes no processo, encontra um ponto mútuo aceitável no espaço de acordos ou quando, por qualquer razão, o protocolo determina que a pesquisa deve ser terminada, sem que os agentes cheguem a fazer algum tipo de acordo [Teixeira, 2002].

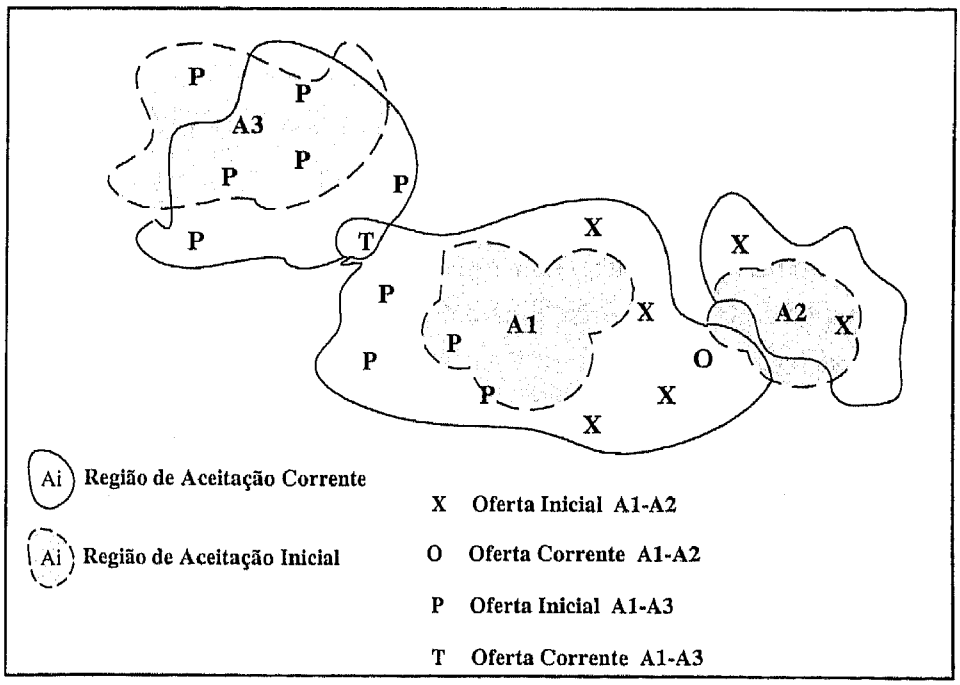

Figura 23: Espaço de Acordos de Negociação

O exemplo ilustrado na Figura 23 representa uma negociação entre o agente A1 e os agentes, A2 e A3. A oferta corrente na negociação entre A1 e A3 (que corresponde à letra T) está na área de sobreposição dos dois agentes, o que significa que esta oferta pode representar um potencial acordo entre eles. Por outro lado, no caso da negociação entre os agentes A1 e A2, a oferta corrente (que corresponde à letra O) não levará os agentes a chegar a acordo, uma vez que está fora do espaço de acordos do agente A2. Na verdade, nesta negociação entre os agentes A1 e A2, não haverá acordo sem que os agentes alterem os seus espaços de negociação, isto porque não existem áreas de intersecção entre os dois espaços de acordos.

Segundo esta análise, pode-se dizer que o requisito mínimo de um agente de negociação é a capacidade de fazer e responder a propostas. Uma proposta é uma solução para o problema em negociação [Parsons et al., 2001]. De um modo geral, as propostas podem ser realizadas de acordo com o histórico da negociação ou independentemente das propostas dos outros agentes participantes no processo de negociação.

Neste tipo de negociação específico, o facto dos agentes apenas poderem aceitar ou rejeitar propostas, pode tornar a negociação morosa e ineficiente. Isto deve-se a que os agentes participantes não possuem qualquer mecanismo que permita justificar o porquê de uma dada proposta ser inaceitável. Desta forma, os agentes podem não estar próximos de atingir um acordo e não possuem conhecimento da direcção do espaço de acordos para que se podem dirigir de forma a atingir um acordo. Assim sendo, para aumentar a eficiência do processo de negociação, é desejável que o protocolo seja capaz de tornar o "*feedback*" das respostas mais útil. Este "*feedback*" pode ter a forma de críticas ou contrapropostas. O agente deve ser capaz de aproveitar o "*feedback*" recebido para o utilizar na geração de propostas que sejam mutuamente mais benéficas de modo a permitir atingir mais rapidamente um acordo.

4.4 Leilões

Um leilão é um mercado especial com um sistema de regras explicitamente definido, determinando a alocação de recursos e preços, que são definidos tendo como base as ofertas realizadas pelos participantes. Este paradigma de negociação, devido à sua simplicidade e clareza, é utilizado em diversos sistemas de negociação automática [Brandt et al., 2000], [Oliveira et al., 1999] [Oliveira, 2001]. No entanto, possui também algumas desvantagens, nomeadamente o possível aparecimento de "*shills*" (compradores fictícios), especulação e coligações.

Antes de efectuar a descrição dos vários tipos de leilões será importante fazer algumas considerações gerais, úteis para compreender o funcionamento dos diversos tipos de leilões.

Normalmente existem duas razões para as pessoas participarem em leilões [Agorics, 1996]. A primeira é a aquisição de bens para consumo particular, como por exemplo, vinho ou flores e, nestes casos, o comprador utiliza a sua própria avaliação do bem vendido (*avaliação privada).* A segunda razão para participar num leilão é comprar bens que serão posteriormente usados para revenda ou uso comercial. Neste caso o bem em causa vale o mesmo para todos os compradores, apesar de cada um ter a sua própria avaliação privada, mas o valor exacto do bem é desconhecido. A isto chama-se o *valor comum* de um bem, que por outras palavras significa, que o valor de um bem é determinado por outros bens semelhantes já anteriormente leiloados. Um exemplo deste caso, é a compra de uma habitação ou terreno, onde cada possível comprador tem uma diferente avaliação privada mas cada um deve adivinhar qual o preço que o imóvel terá no final. Um outro termo comum na literatura da especialidade é o "*castigo do vencedor*", que é o nome atribuído para quando o vencedor paga por um item mais do que ele realmente vale. O vencedor é confrontado com a ideia que os outros participantes avaliaram o item por um baixo valor e quando realmente pensa que ganhou ao comprar o bem, engana-se porque os seus lucros baixaram. O vencedor é o licitador que fez o maior erro positivo na avaliação do bem.

Existe uma taxonomia básica para classificar os leilões, semelhante à utilizada para classificar os processos de negociação genéricos. Esta taxonomia classifica os processos específicos de leilão relativamente ao objecto da transacção, número de participantes, visibilidade da proposta e cálculo do preço final:

- **Objecto da Transacção.** O objecto da transacção pode ser um item único (leilão de *item único*) ou múltiplos itens (leilão de *múltiplos itens*);
- **Número de Participantes.** Os leilões podem ser *"one sided"*, quando apenas ofertas são permitidas (leilão de um-para-muitos), e *"double sided"* onde é permitido, aos compradores e vendedores, fazerem ofertas simultaneamente (leilão de muitos-para-muitos);
- **Visibilidade da Proposta.** As propostas podem ser visíveis a todos os participantes do leilão (leilão *aberto*) ou visíveis unicamente ao leiloeiro (leilão *fechado* ou de *propostas seladas*);
- **Cálculo do Preço Final.** Os leilões podem ser do tipo *1º preço* ou *2º preço*, caso o bem seja vendido pelo valor exacto da oferta mais elevada, ou caso o vencedor pague a segunda melhor oferta pelo bem para venda.

Existem diversos tipos de leilões aplicáveis em situações distintas e com vantagens e desvantagens relativas. Entre os mais utilizados incluem-se o Leilão Inglês, Leilão Holandês (ou de preço descendente), Leilão de Propostas Seladas – 1º Preço (ou discriminatório), Leilão de Vickrey – 2º Preço, Leilão de Pagamento Generalizado, Leilão Duplo [Reis, 2002d]. No anexo 1 é apresentada uma descrição de cada um destes tipos de leilões e das suas vantagens e desvantagens relativas.

Os diversos tipos de leilões podem ser vulneráveis a diversos tipos de factores, incluindo [Sandholm, 1999]: coligações entre compradores, compradores mentirosos, revelação indesejada de informação privada, leiloeiros mentirosos e anúncios fictícios. No anexo 2 é apresentada uma descrição detalhada das vulnerabilidades de cada tipo de leilão.

4.5 Modelos de Negociação

4.5.1 Negociação Baseada na Teoria dos Jogos

A negociação baseada na teoria dos jogos utiliza um método semelhante às estratégias utilizadas na teoria económica. Os conceitos básicos nesta abordagem são: funções de utilidade; espaços de acordos; estratégias e protocolos de negociação. A utilidade é definida como a diferença entre o ganho por atingir o objectivo e o custo em atingi-lo. Cada agente possui a sua própria função de utilidade que pode depender de um conjunto de parâmetros. Quando dois ou mais agentes entram num processo de negociação, uma matriz de pagamentos ("*payoff matrix*") é construída baseada nas funções de utilidade de cada um dos agentes. A matriz representa a utilidade resultante para cada agente do resultado das suas ofertas e das ofertas dos outros participantes. Esta matriz é construída a partir de conhecimento comum a todos os participantes. Partindo deste conhecimento, os agentes vão realizar ofertas e contra-ofertas com o objectivo de maximizarem a sua utilidade e com a assunção de que cada um dos outros agentes irá também maximizar a sua utilidade.

A teoria dos jogos funciona bastante bem como forma de prever o efeito que uma dada oferta terá num oponente e no efeito que terá à posteriori no próprio agente. No entanto, esta teoria, quando aplicada à negociação entre agentes, tem diversas limitações [Nwana et al., 1996]. Uma primeira limitação surge do facto de assumir que todos os agentes se vão comportar de forma racional, procurando maximizar a sua própria utilidade e como tal não deixar margem para comportamento irracional. Uma segunda limitação, ainda mais importante, consiste no facto de esta teoria se basear na assunção que todos os agentes vão revelar completamente a sua própria estratégia aos outros agentes de forma a permitir construir a matriz de pagamentos. Desta forma, todos os agentes possuem conhecimento completo sobre as preferências e crenças de todos os outros agentes. No entanto, isto não é o que se passa no mundo real e como tal também não será realista em ambientes abertos de negociação entre agentes tais como a Internet. Os agentes têm de raciocinar sabendo de antemão que possuem unicamente um conhecimento parcial das preferências e crenças dos outros. Por outro lado, em sistemas com centenas ou milhares de agentes, a matriz de pagamentos ficaria também demasiado grande para poder ser tratada. Uma limitação final do modelo está relacionada com o facto de lidar unicamente com informação actual e não entrar em linha de conta com o histórico da negociação ou predições do futuro da mesma.

4.5.2 Negociação Baseada em Planos

A negociação baseada em planos descreve a negociação como um processo dividido em duas fases. Na primeira fase, o agente cria um plano composto por um conjunto de acções que ele pensa conduzir aos seus objectivos. Na segunda fase, o agente tenta coordenar o seu plano com os dos outros agentes de forma a encontrar e resolver os possíveis conflitos. Este tipo de negociação encontra-se já na fronteira entre as metodologias de cooperação e de negociação. Grande parte dos tópicos que emergem da sua utilização, estão centrados no modo de coordenação entre os diversos agentes que possuem planos parciais incompatíveis.

Os agentes podem adoptar uma estratégia de planeamento centralizada onde um dos agentes assume o papel de coordenador, recebe os planos dos outros agentes e resolve os conflitos entre eles de forma a encontrar um plano multi-agente que possa satisfazer as necessidades de todos os participantes. Este papel de coordenador pode ser assumido por um dos agentes participantes ou por um agente externo. Se existirem relações hierárquicas entre os agentes é usual que o agente coordenador seja o agente que se encontra no topo da hierarquia. Se não existirem hierarquias bem definidas, a coordenação centralizada é bastante mais complexa e a existência de um agente coordenador torna-se difícil de implementar.

Em contraste com o planeamento centralizado surge o planeamento distribuído. Neste tipo de planeamento, cada agente tenta construir um modelo dos restantes agentes e dos seus planos. Os agentes negoceiam depois uns com os outros de forma a alterarem os seus planos e os modelos dos planos dos outros agentes. A negociação termina quando os conflitos com os planos dos restantes agentes se encontrarem resolvidos. Um dos principais problemas do planeamento multi-agente está relacionado com a quantidade de comunicação necessária para efectuar a negociação conducente à resolução dos conflitos entre os agentes e consequente elaboração de um plano global. Por outro lado, os problemas resultantes da existência de um único agente coordenador, que acaba por ser um "*bottleneck*" (estrangulamento) no planeamento, não se verificam. O planeamento distribuído, coloca no entanto problemas de investigação no que diz respeito aos métodos de coordenação dos diversos agentes participantes. Como os agentes não têm um conhecimento completo dos planos, objectivos ou intenções dos outros agentes, a sua coordenação é bastante complexa. Métodos complexos são necessários para criar um único plano global a partir dos diversos sub-planos parciais. Os diversos métodos de planeamento distribuído propostos por diferentes autores serão abordados com maior detalhe no capítulo 5, quando forem tratadas as questões relativas à coordenação de agentes cooperativos.

4.5.3 Negociação Baseada em Técnicas Humanas

As negociações entre humanos introduzem diversas características que não estão presentes na negociação entre agentes. Conceitos tais como arbitragem, mediação e relaxação de propostas surgem neste contexto. A criação de modelos baseados na negociação humana implica outros conceitos difíceis de modelizar tais como o *stress*, frustração, hostilidade, desconfiança, bloqueio, insegurança, timidez, falsos julgamentos, rivalidades, distorções preceptivas, juízos erróneos, enviesamento cognitivo, metodologias de raciocínio estereotipado, estratégias de afrontamento, experiência, etc. A maioria das abordagens deste tipo de negociação baseiam-se na observação da forma como os humanos efectuam a negociação. Os humanos baseiam os seus processos de negociação na sua experiência e no seu conhecimento específico no âmbito da negociação em curso. Usualmente na negociação entre humanos aplicam-se as quatro fases de Benson et al. [Benson et al., 1987]: preparação (onde se estuda o tipo de negociação e os adversários); argumentação (onde os participantes trocam argumentos e fazem as primeiras propostas), série de propostas e contra-propostas (com vista a resolver os conflitos das primeiras propostas), encerramento (onde se estabelece o negócio final). Na fase de propostas e contra-propostas, os agentes relaxam as suas restrições e procuram influenciar as crenças dos outros agentes de forma a atingirem o melhor acordo possível. No entanto, esta noção de relaxação e a sua forma de utilização no processo de negociação são complexas. É relativamente simples entrar num processo de negociação em que os agentes continuamente trocam propostas e contra-propostas sem que se atinja um acordo. Esta situação é usualmente designada por *livelock*. [Nwana et al., 1996].

Uma outra forma de negociação baseada na negociação humana, consiste no uso da mediação como método de resolução de conflitos no decurso da negociação. Segundo Jesuino [Jesuino, 1998], a mediação está muitas vezes associada aos conflitos intergrupais, nas organizações, envolvendo responsáveis do mesmo nível hierárquico. Os mediadores não dispõem de poder formal, não podendo impor uma solução. A sua acção usualmente limita-se ao aprofundamento dos pontos discordantes, à identificação de áreas de compromisso potencial e ao encorajamento das partes na realização de concessões. O mediador, no mínimo, tem o papel "efeito de presença", ou seja, assiste, regista as ocorrências e por vezes poderá encorajar as partes a serem mais conciliatórias.

Pruitt [Pruitt, 1981], refere algumas tácticas de mediação: equilíbrio do poder, o mediador pode intervir no sentido de igualar ou equilibrar o poder das partes; pressão do tempo, pode fixar prazos limites, contribuindo para o reconhecimento da necessidade da ordenação; atitudes recíprocas, pode ajudar a melhorar as atitudes das partes em relação umas às outras, prevenindo contra estereótipos e instruindo sobre os efeitos de comportamentos rígidos; comunicação, pode melhorar a comunicação entre os negociadores, beneficiando o desenvolvimento de soluções integrativas. O mediador pode ainda intervir na identificação de soluções concretas. Convém distinguir entre as situações

onde existe uma solução que satisfaz os interesses de ambas as partes e as situações onde essa alternativa não existe, porque o papel do mediador vai ser diferente num e noutro caso. Se existe uma alternativa viável, mas os negociadores não a identificam ou não a aceitam, o papel do mediador deve ser de identificar as causas do bloqueio e, em seguida, recorrer às tácticas adequadas para as neutralizar e superar.

As funções essenciais do mediador são: identificar alternativas integrativas; encorajar os participantes a clarificar as suas necessidades e prioridades e desenvolverem uma atitude favorável à procura de soluções e a desenvolver a comunicação; diagnosticar as situações, analisando se os pontos em negociação são susceptíveis de compensação recíproca, ou de uma solução de compromisso ou ainda se há margem para soluções criativas; identificar possibilidades de conciliação, devendo o mediador exercer pressão no sentido de cada uma das partes alterar os seus respectivos limites mínimos, utilizando tácticas do tipo distributivo, como a persuasão ou o recurso a promessas e ameaças.

A introdução de mediação no processo de negociação entre agentes, embora possa resolver problemas relacionados com bloqueios, introduz no entanto uma certa centralização no processo e um estrangulamento ("*bottleneck*") que é o agente mediador. Uma alternativa (ou mesmo complemento) à mediação consiste na utilização de um agente com poder de decisão que realize uma espécie de arbitragem do processo. Os agentes apresentam as suas posições a este agente que, após escutar ambas as partes, toma uma decisão sobre o acordo a adoptar. Os árbitros podem pronunciar-se sobre uma ou várias questões e podem ter poder vinculativo ou consultivo sobre os agentes. No procedimento de arbitragem distinguem-se dois métodos: *método convencional*, onde o árbitro tem a liberdade de recomendar qualquer acordo por ele considerado apropriado; *método da oferta final*, em que o árbitro tem que escolher entre uma das soluções finais apresentadas pelas partes. A arbitragem é muito utilizada em processos humanos de negociação tais como a resolução de conflitos laborais ou no sistema judicial. A principal vantagem deste processo é permitir uma solução clara para os conflitos, reduzindo os custos das negociações prolongadas e sem saída [Jesuino, 1998]. No entanto, a utilização de arbitragem pode introduzir vários inconvenientes como sejam o efeito de congelação (em que as partes antecipam o fracasso e não se empenham na obtenção de um acordo), a possível parcialidade do árbitro (beneficiando uma das partes) e a possível dificuldade em fazer com que as partes aceitem a decisão final do árbitro.

4.6 Negociação Multi-Atributo

Embora grande parte da discussão realizada anteriormente diga respeito a negociação sobre um único atributo (usualmente o preço), muitas vezes a negociação é realizada sobre múltiplos atributos simultaneamente, como sejam: o preço, qualidade, prazo de entrega, prazo de pagamento, etc.

A consideração de múltiplos atributos na negociação é muitas vezes simplificada através da definição de uma função de utilidade global que corresponde a uma combinação linear das utilidades dos vários atributos do produto negociado. Este procedimento transforma a negociação multi-atributo novamente numa negociação de atributo único (a utilidade global). Por exemplo, Oliveira et al. [Oliveira et al., 1999] [Macedo, 2001] utilizam este método na compra de automóveis com múltiplos vendedores e diversos atributos em negociação. Outros exemplos deste mecanismo de conversão de negociações multi-atributo em atributo simples podem ser encontrados em Vulkans e Jennings [Vulkans e Jennings, 1998], [Matos e Sierra, 1998] e [Cardoso e Oliveira, 2000].

Abordagens mais elaboradas utilizam ordens preferenciais para os diversos atributos do produto em negociação. Exemplos disto podem-se encontrar em [Macedo, 2001], [Pereira, 2000] e [INSPIRE, 2001]. Neste contexto, Macedo[18] [Macedo, 2001], propõe um protocolo original de negociação multi-atributo com vista à formação de empresas virtuais. Utiliza uma negociação iterativa multi-atributo, com adaptação e resolução de dependências distribuídas, apropriada ao contexto de uma actividade comercial [Oliveira e Rocha, 2000].

4.7 Estratégias e Tácticas de Negociação

As estratégias de negociação, podem ser entendidas como uma forma global de encarar a negociação, tendo em conta todos os factores a partir dos quais surge essa mesma negociação [Cunha, 1999]. Consistem em planos globais, para as diversas fases do processo de negociação, que devem ser o mais detalhados e flexíveis possível. Reis et al. [Reis et al., 2001] [Reis e Lau, 2001a] definem uma estratégia para agentes como sendo composta por um conjunto de tácticas que possuem condições de activação e especificam o comportamento completo de um agente ou grupo de agentes.

As tácticas de negociação são o conjunto de comportamentos utilizados pelas partes envolvidas no processo de negociação, para conseguir os seus objectivos. Cada estratégia aplicada leva à aplicação de tácticas distintas, ou seja para cada tipo de estratégia existe uma conduta que mais se adequa à situação em causa [Cunha, 1999] [Reis et al., 2001]. Usualmente os bons negociadores, sejam eles humanos ou agentes, são capazes de alterar a sua táctica ao longo da negociação em face do tipo de negociação, dos oponentes e do decurso do processo de negociação.

Necessário será dizer que estas estratégias e tácticas têm, sem dúvida, um papel importante para uma boa negociação, mas também é verdade que seria um desperdício de esforço se se levasse com rigidez o cumprimento de todas elas. É por isso importante

[18] Ana Paula Rocha Macedo cujo nome científico é Ana Paula Rocha.

possuir estratégias flexíveis compostas por tácticas também elas flexíveis e condições de activação também flexíveis que tenham em consideração a evolução da negociação e os comportamentos dos diversos participantes. No entanto, esta tripla flexibilidade não é nada simples de alcançar.

As estratégias e tácticas podem ser divididas em dois grupos, de acordo com o tipo de negociação: estratégias e tácticas integrativas e competitivas. No âmbito de negociações integrativas, Pruitt [Pruitt, 1981] refere que existem vários tipos de acordos integrativos:

- **Expansão dos recursos.** Muitos conflitos resultam da escassez de recursos ou do seu anulamento mútuo, mas a sua expansão pode não ser viável, tanto pelos custos que envolvem como pela situação que pode constituir uma oportunidade única no tempo;
- **Compensação não especifica.** Uma das partes, de modo a obter o que pretende, compensa a outra de forma não directamente relacionada com a negociação dos custos por ela incorridos. Para que este tipo de acordo resulte é necessária informação sobre o que a outra parte valoriza;
- **Compatibilização de prioridades.** Cada uma das partes cede nos pontos de baixa prioridade para si mas que são de alta prioridade para a outra;
- **Redução de custos.** Uma das partes obtém o que deseja e, em compensação, os custos para a outra parte são reduzidos ou mesmo eliminados; para tal, é necessário ter conhecimento das preferências das partes;
- **Alternativa super-ordenada.** Identificação de uma nova opção que satisfaça os interesses mais importantes de ambas as partes, pela descoberta de soluções originais e criativas. Segundo Fisher et al. [Fisher et al., 1993], esta negociação deve deslocar-se das posições para os interesses. Jesuino [Jesuino, 1998], refere um exemplo clássico desta táctica que consiste em duas senhoras numa biblioteca publica que entram em conflito de posições: uma quer a janela aberta e a outra quer a janela fechada. Os interesses são: uma delas é que haja ventilação devido ao seu problema de asma, e a outra, é evitar corrente de ar pois receia constipar-se. A solução que se poderia atribuir, neste caso, era abrir a janela do compartimento contíguo, o que permite o ar circular sem que haja perigo de corrente de ar. A descoberta de soluções originais pode ser objecto de técnicas de grupo como, por exemplo, o *brainstorming* ou os *grupos nominais* [Jesuino, 1998]. O *brainstorming* é uma técnica de grupo onde se pede aos indivíduos para criarem o máximo de ideias possíveis, sem haver discussões, as quais são registadas e fixadas num quadro num local visível. Seguidamente é efectuada uma análise das ideias pelo grupo, retendo-se as válidas para a resolução do problema. Por vezes é difícil aceitar certas ideias emitidas por determinados sujeitos, porque estes adoptam posturas competitivas. Os *grupos nominais*, consistem em grupos em que cada um dos seus membros trabalha em separado, o

que evita os inconvenientes das inibições e reservas. As ideias resultantes são reunidas e os seus respectivos autores ignorados (anonimato). Isto permite reduzir o número de opções e facilitar o posterior processo de selecção e avaliação de soluções.

Jesuino [Jesuino, 1998] referencia que no processo integrativo há que trocar informações e ensaiar propostas alternativas. A troca de informação pode ter forma explícita ou implícita (indirecta). Na forma explícita, os negociadores podem inclusivamente revelar totalmente as suas preferências e crenças, transformando a negociação numa resolução conjunta do problema. Na comunicação implícita, temos a argumentação utilizada para defender a sua própria posição que revela quase sempre algo sobre a natureza dos motivos e interesses subjacentes; o desenvolvimento de estratégias cognitivas para determinar os interesses reais do seu oponente; propostas que implicitamente revelam quais as possíveis alternativas integrativas; perguntas tácticas, como por exemplo, "como alteraria a minha proposta para a tornar mais aceitável?", etc.

Relativamente às estratégias e tácticas competitivas, estas incluem [Jesuino, 1998]:

- **Táctica da lisonja.** Tem como objectivo diminuir a capacidade de resistência do oponente, mas tem de ser utilizada com subtileza, senão é tomada como manipuladora. Esta táctica visa criar uma atmosfera de confiança, favorecendo o diálogo e a cedência por parte dos outros agentes;
- **Tácticas de persuasão.** A persuasão tem o objectivo de convencer o interlocutor que as propostas são razoáveis, válidas e até vantajosas. A táctica de persuasão é um processo que se exerce a partir da comunicação. A credibilidade da fonte é imprescindível para a persuasão, e é fácil de estabelecer quando os interlocutores se conhecem mutuamente ou quando a identificam através de informações secundárias. O estatuto e competência de um agente, poderá permitir-lhe influenciar um outro agente de forma subtil;
- **Promessas e ameaças.** No processo de negociação, as promessas e ameaças são tácticas utilizadas com o objectivo de obter concessões do oponente. As promessas são uma táctica mais ligeira e positiva. As ameaças têm um cariz mais pesado e negativo. Ambas derivam do poder: poder de recompensa ou poder de punição, respectivamente;
- **Decisões irreversíveis.** As decisões irreversíveis são ameaças que constituem ruptura do processo negocial e devem ser evitadas. Por exemplo, um aviso público de greve é mais poderoso do que uma "simples" ameaça de greve. Neste caso está a transferir-se a responsabilidade do resultado da negociação para a outra parte. A greve pode ser ou não desconvocada, dependendo desse mesmo resultado. Existem várias formas para dar credibilidade às decisões irreversíveis, tais como: tornar pública a posição (por exemplo uma greve); demonstrar que a posição defendida resulta de exigências irredutíveis; a aliança com terceiros (por

exemplo, solidariedade sindical).

No âmbito das tácticas para negociação computacional entre agentes merece referência o estudo realizado por Robert Axelrod em 1980. Axelrod organizou um torneio entre programas de computador que jogavam o *Dilema do Prisioneiro* de forma iterativa durante 200 rondas. O *Dilema do Prisioneiro* é um cenário típico da teoria dos jogos que consiste no seguinte: "Dois homens são acusados de um crime. Não podem comunicar um com o outro e sabem que se um confessar o crime e o outro não, o confessor será libertado e o outro preso por 3 anos. Se ambos confessarem serão ambos presos por 2 anos. Se nenhum confessar serão ambos presos por um ano". Neste exemplo, o *Equilíbrio de Nash*, correspondente à acção a seleccionar seguindo a teoria dos jogos, estaria na não cooperação mútua, ou seja, ambos confessarem. Se ambos os agentes actuarem de modo racional irão não cooperar, obtendo desta forma um castigo de 2. Embora isto pareça um desperdício, pois no caso de ambos cooperarem o castigo seria de 1, o que se passa é que se um agente assumir que o outro será cooperativo, então porquê cooperar? Se o outro agente for cooperativo e ele cooperar terá um castigo de 1, mas se não cooperar terá um castigo de 0. Desta forma, o comportamento racional numa única ronda do jogo do prisioneiro será não cooperar. No entanto, na competição de Axelrod, o jogo era composto por múltiplas rondas, jogadas de forma sucessiva, sendo possível identificar o comportamento do oponente e verificar a sua vontade ou não de cooperar.

No torneio de Axelrod, em cada uma das 200 rondas, cada um dos programas podia seleccionar uma de duas acções: cooperar, i.e. não confessar, ou não cooperar, i.e. confessar. Cada programa jogava 5 jogos completos contra cada um dos outros programas e o resultado final era a soma dos castigos totais, vencendo o programa com o menor castigo global.

De entre as estratégias mais simples utilizadas pelos participantes destacaram-se:

- Realizar sempre a mesma acção, cooperando em todas as rondas ou não cooperando em todas as rondas.
- Aleatória, ignorando as acções do adversário, seleccionando cooperar ou não cooperar, com 50% de probabilidade cada.
- *Tit-for-tat.* Na primeira ronda cooperar. Nas rondas seguintes executar a mesma acção que o oponente realizou na ronda anterior.

As estratégias mais complexas baseavam-se em variações do *Tit-for-tat* ou tentativas de explorar adversários mais fracos procurando-os forçar a cooperar mas por vezes realizando acções não cooperativas como forma de ganhar vantagem. O torneio foi vencido pela estratégia *Tit-for-tat* básica que se superiorizou a estratégias bastante mais complexas.

No seu estudo, Axelrod [Axelrod, 1984] concluiu que o *Tit-for-tat* venceu pois teve a oportunidade de competir contra outros programas também inclinados para cooperar.

Conclui também que, em cenários deste tipo, é boa ideia começar por cooperar de modo a que o oponente coopere também. Por outro lado, é útil castigar o oponente por não cooperar exactamente no mesmo grau mas, ter capacidade de perdoar e cooperar, no caso do oponente se mostrar a determinada altura cooperativo.

Do estudo de Axelrod saíram as bases para a definição de estratégias e tácticas mais complexas em negociações entre agentes que se processam durante iterações ou rondas sucessivas.

Rosenschein e Zlotkin, introduziram um protocolo e estratégia de negociação designada por *Protocolo Monótono de Concessão* [Rosenschein e Zlotkin, 1994]. Nesta estratégia, inicialmente, ambos os agentes participantes na negociação enviam uma oferta (Of_1 e Of_2), dos seus conjuntos de ofertas admissíveis (C_1 e C_2). Um acordo é atingido se um dos agentes considera a proposta do outro melhor ou igual à sua proposta, ou seja, $Utilidade_1(Of_2) \geq Utilidade_1(Of_1)$ ou $Utilidade_2(Of_1) \geq Utilidade_2(Of_2)$. Se ambos os casos se verificarem simultaneamente, então uma das ofertas (Of_1 ou Of_2) é seleccionada aleatoriamente. Se um acordo não é atingido, então a negociação continua com uma nova ronda de propostas simultâneas. Nenhum agente pode realizar uma proposta que seja pior para o outro agente do que a realizada na ronda anterior. Se nenhum agente realiza uma concessão numa determinada ronda então a negociação termina sem acordo. Este protocolo garante que se o conjunto de acordos possível for finito, a negociação termina (com ou sem acordo) num número finito de rondas. No entanto, devido à possível elevada dimensão do conjunto de acordos possíveis, este protocolo não garante que a negociação termine num tempo razoável.

O *Protocolo Monótono de Concessão* deixa em aberto diversas questões importantes como sejam: Qual deverá ser a primeira proposta dos agentes? Que agente deve conceder em cada ronda? Quanto deve cada agente conceder em cada ronda? Em princípio, a primeira proposta deverá, logicamente, ser o acordo preferido do agente. No entanto, a resposta às duas perguntas restantes não são tão simples. Rosenschein e Zlotkin, propuseram a *Estratégia de Zeuthen* que se baseia em definir a concessão em cada ronda como função do nível de risco que o agente está disposto a assumir. No seu trabalho definem que a vontade de arriscar um conflito para o agente i, na ronda t, será medida como [Rosenschein e Zlotkin, 1994]:

$$risk_i^t = \begin{cases} 1 & se \quad Utilidade_i(Of_i^t) = 0 \\ \dfrac{Utilidade_i(Of_i^t) - Utilidade_i(Of_j^t)}{Utilidade_i(Of_i^t)} & se \quad Utilidade_i(Of_i^t) \neq 0 \end{cases}$$

O valor do risco será então um valor entre 0 e 1 calculado dividindo a utilidade que o agente i perderia se aceitasse a proposta do agente j na ronda t, pela utilidade que i perderia não concedendo e causando o conflito. Valores elevados de risco (perto de 1) indicam que i tem pouco a perder com o conflito. Valores reduzidos de risco (perto de 0)

indicam que *i* terá muito a perder com o conflito e como tal deverá conceder. A estratégia Zeuthen propõe que o agente que deve conceder na ronda *t*, deverá ser aquele com o valor menor da função de risco. Se ambos tiverem o mesmo valor de risco então ambos devem conceder. A estratégia propõe ainda que, em cada ronda, cada agente deverá efectuar a menor concessão necessária para mudar o balanço do risco, para que na ronda seguinte seja o outro agente a conceder.

Embora a *Estratégia de Zeuthen* não garanta o sucesso da negociação, garante a sua terminação num tempo finito. Não garante também a maximização do bem social mas garante que a solução atingida será *Pareto Óptima*. É também racional individualmente para os agentes pois se um acordo for atingido, esse acordo será melhor para os agentes do que a inexistência de acordo. A estratégia está ainda em *Equilíbrio de Nash* pelo que, se um agente a usar, o outro não terá melhor solução do que a usar também. No entanto, a estratégia assume a negociação bilateral, veracidade dos agentes e que os valores individuais de cada agente são revelados no início da negociação. Em domínios realistas isto não se verifica pelo que a sua aplicação neste tipo de domínios será muito limitada.

A definição de tácticas para negociação entre agentes foi estudada numa perspectiva mais geral, incluindo negociações entre vários agentes e informação privada, por Faratin et al. [Faratin et. al., 1998]. No seu estudo propõe que as tácticas correspondam ao modo de geração de uma proposta ou contra-proposta a partir de um objecto ou item de negociação, e utilizando um único critério (tempo, recurso, etc.). Propõe as seguintes tácticas genéricas de negociação [Faratin et. al., 1998]:

- *Tácticas dependentes do tempo.* Os agentes variam as suas propostas de acordo com o tempo que dispõem para concluir a negociação. Se existir uma *deadline* em que os agentes têm de atingir um acordo, será provável que os agentes concedam mais quando essa *deadline* se aproxima. Estas tácticas utilizam uma função concessão dependente do tempo cuja forma pode ser parametrizada convenientemente;
- *Tácticas dependentes de recursos.* Os agentes variam as suas propostas com base na disponibilidade de um determinado recurso ou conjunto de recursos. Modelizam a pressão para chegar a um acordo que os recursos limitados (tais como largura de banda, dinheiro, memória, etc.) e o ambiente (por exemplo, o número de agentes que concorrem à utilização do recurso) impõem ao comportamento do agente. As funções de concessão utilizadas são semelhantes às dependentes do tempo, exceptuando que o seu domínio é a quantidade de recursos disponível e não o tempo;
- *Tácticas dependentes do comportamento.* Os agentes tentam imitar o comportamento dos seus oponentes ou responder convenientemente a este comportamento. Diferentes tipos de imitações podem ser realizadas, baseando-se na conduta da negociação de um oponente na sequência das suas propostas.

Estas tácticas básicas são depois combinadas numa função linear, utilizando pesos, construindo deste modo as propostas ou contra-propostas. Por exemplo, uma proposta poderá ser construída através de uma função linear que pesa os valores obtidos por uma táctica dependente do tempo e uma táctica dependente de um dado recurso [Faratin et. al., 1998].

Baseada nesta noção simples de táctica, Faratin et. al. [Faratin et. al., 1998] constroem estratégias que definem, em cada instante, qual a combinação apropriada de tácticas a utilizar para gerar uma proposta ou contra-proposta.

4.8 Utilização Prática da Negociação

A utilização de metodologias de negociação electrónica e de leilões em particular, é cada vez mais o mecanismo preferencialmente utilizado na coordenação de agentes egoístas ("*self-interested*"). De facto, com o crescimento exponencial da Internet e do número de operações de comércio electrónico realizadas utilizando este meio, a negociação assume cada vez maior importância no contexto das metodologias de coordenação entre agentes. Este facto motivou o aparecimento de diversos sistemas de utilização prática da negociação utilizando a Internet, em problemas específicos. Nesta secção analisam-se resumidamente alguns dos sistemas relevantes neste contexto, nomeadamente: *Kasbah* [Chavez e Maes, 1996], *SMACE* [Cardoso, 1999], *MACIV* [Fonseca, 2000], *ForEV* [Macedo, 2001] e *AuctionBot* [Wurman et. al., 1998].

4.8.1 O Mercado Kasbah

O *Kasbah* [Chavez e Maes, 1996] é um sistema que permite a criação de um mercado composto por agentes autónomos, que representam compradores e vendedores, e transaccionam produtos de diversos tipos. Todos os elementos presentes neste sistema - agentes, produtos e o próprio mercado - são representados como objectos locais ao sistema. A comunicação entre os agentes é realizada por invocação de métodos, tendo os agentes obrigatoriamente de suportar um conjunto de métodos pré-definidos. O comportamento dos diferentes agentes na negociação (compradores e vendedores) varia de acordo com parâmetros especificados pelo projectista do agente que incluem: a data limite para compra/venda do produto, preço desejado e mínimo/máximo aceitável.

Quando um agente é adicionado ao mercado, é notificado de quais os agentes que estão interessados em comprar/vender os produtos que este novo agente vende/compra. Quando um agente sai do mercado, os outros agentes são notificados desse facto. Desta forma, cada agente possui o conhecimento de todos os agentes que constituem possíveis parceiros de negociação.

A negociação é realizada de forma directa entre compradores e vendedores. Os agentes realizam propostas de compra ou venda aos outros agentes, que as podem ou não aceitar.

As negociações são realizadas em múltiplas rondas, em que os agentes são supostos relaxar as suas ofertas. A negociação termina quando existe um acordo ou é atingido o valor limite da negociação de todos os agentes, caso em que já não será possível obter acordo. O sistema disponibiliza agentes pré-definidos com três estratégias de negociação possíveis: ansioso, calmo e moderado (correspondem a funções de alteração do valor das ofertas ao longo do tempo: linear, quadrática ou cúbica).

Este sistema possui como características mais negativas: não ser um sistema distribuído (pois os agentes são representados como objectos e a comunicação é efectuada por invocação de métodos) e não permitir uma grande flexibilidade na definição de agentes (os agentes, embora parametrizáveis, são pré-definidos). As suas principais vantagens relativamente a outros sistemas estão relacionadas com o facto de permitir a negociação directa entre agentes vendedores e compradores (sem mediação por parte de um leiloeiro) e disponibilizar aos agentes presentes no mercado, informação detalhada sobre os possíveis parceiros de negócio.

4.8.2 O Sistema SMACE

O *SMACE* [Cardoso, 1999] [Cardoso e Oliveira, 2000] é um Sistema Multi-Agente de negociação no âmbito do Comércio Electrónico em que os utilizadores delegam intenções de compra ou venda de produtos em agentes que se encarregam de negociar de forma autónoma com outros agentes de modo a atingir os seus objectivos. Funciona como um mercado aberto onde agentes compradores e vendedores negoceiam entre si, de modo a chegar a acordo sobre os bens que pretendem comprar ou vender. Os agentes compradores e vendedores podem residir em máquinas com localizações distintas, interagindo com o mercado através da Internet.

O modelo de negociação do *SMACE* é multi-lateral (diversos compradores e vendedores negoceiam simultaneamente) e multi-atributo (pois a negociação ocorre sobre um conjunto de atributos tais como o preço, garantia, tempo de entrega, qualidade, etc.) [Cardoso e Oliveira, 2000].

O sistema, embora aberto à participação de agentes exteriores, disponibiliza um conjunto de agentes predefinidos que podem ser criados automaticamente [Cardoso, 1999]: *Multiple Tactic Agent* (que pode combinar até três tácticas na geração das suas propostas); *Adaptive Behaviour Agent* (que permite combinar tácticas e inclui capacidades de *aprendizagem por reforço)*; *Enhanced Adaptive Behaviour Agent (*semelhante ao anterior, mas com um algoritmo de aprendizagem superior).

O *SMACE* apresenta uma interface com o utilizador amigável que permite criar agentes compradores e vendedores e visualizar a sua acção e desempenho. O sistema foi desenvolvido em linguagem Java utilizando a plataforma *JATLite* [Jatlite, 2001]. Os agentes trocam mensagens no formato *KQML*.

O sistema *SMACE* apresenta um modelo abstracto semelhante ao do *Kasbah* em que a negociação é efectuada directamente entre compradores e vendedores. No entanto possui algumas potencialidades que lhe transmitem uma muito maior flexibilidade. Uma das potencialidades está relacionada com as características de negociação multi-atributo do sistema. Por outro lado, a negociação é realizada utilizando a Internet e a linguagem standard para agentes *KQML*. Isto permite a investigadores externos, o desenvolvimento dos seus próprios agentes, transmitindo uma maior heterogeneidade ao sistema. Finalmente, os agentes pré-definidos no *SMACE* são também mais flexíveis em termos das tácticas disponíveis do que os agentes *Kasbah*. A última vantagem está relacionada com o facto do sistema possuir potencialidades básicas de aprendizagem. O sistema foi avaliado usando sistemas adaptativos (algoritmo de aprendizagem – Q) e com tácticas fixas.

4.8.3 O Sistema MACIV

Os principais objectivos do projecto *MACIV – Multi Agent System for Resource Management in Civil Construction,* consistiram no desenvolvimento de metodologias para modelizar a gestão de recursos para tarefas complexas em cenários espacialmente distribuídos e caracterizados por alterações ao longo do tempo [Fonseca, 2000] [Oliveira et al., 1997]. Desta forma, no âmbito deste projecto, os autores procuraram aplicar a tecnologia dos agentes de *software* com ênfase nas metodologias de negociação com formação de coligações e aprendizagem, para a resolução de problemas de gestão de recursos distribuída. O cenário escolhido para a aplicação foi a gestão distribuída de recursos em ambientes de construção civil. Este cenário apresenta como principais características: distribuição geográfica; decomposição simples de tarefas; multiplicidade de funções; diversidade de recursos; incerteza e dinâmica.

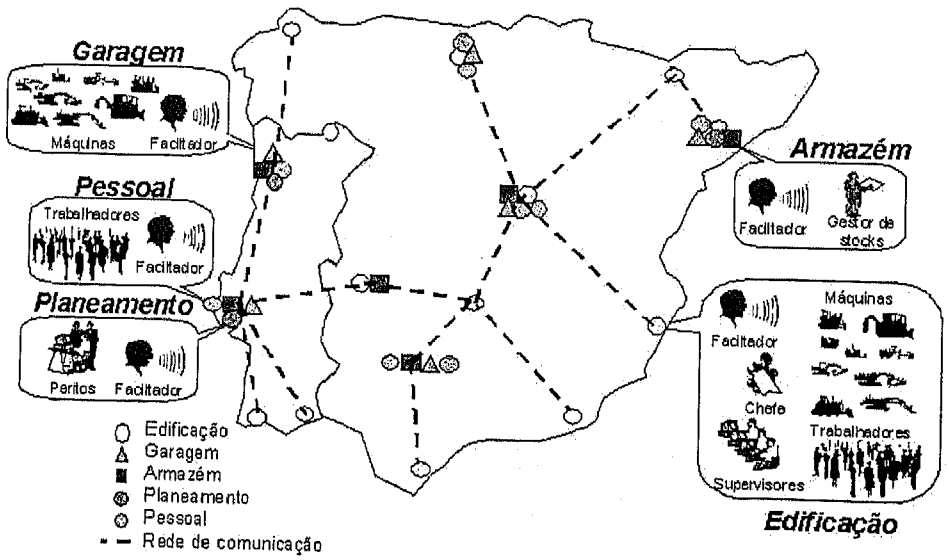

Figura 24: Diagrama do Sistema MACIV [Fonseca, 2000]

O sistema modeliza cada entidade física como um agente de *software* e agrupa os agentes básicos em organizações designadas por macro agente (agrupamento de agentes numa determinada localização geográfica). Na figura 24, podem observar-se cinco tipos de macro-agentes: garagem, pessoal, planeamento, armazém e edificação. Cada um destes macro-agentes tem características e responsabilidades distintas. O sistema utiliza comunicação baseada em facilitadores (em cada macro-agente) de forma a reduzir o volume de comunicação.

As características mais interessantes (no âmbito deste trabalho) do *MACIV* estão relacionadas com os seus protocolos de negociação interactiva. Dado que os diversos agentes representam recursos de diferentes companhias, com interesses antagonistas, comportam-se de forma competitiva. Os autores do projecto [Oliveira et al., 1999] desenvolveram protocolos de negociação iterativa do tipo leilão, para efectuar a alocação de recursos. No entanto, dado que muitas das tarefas deste tipo de cenários (construção civil) são facilmente decomponíveis podendo ser executadas por grupos de agentes, foi dada especial relevância neste trabalho à formação temporária de coligações de agentes. Os protocolos de negociação desenvolvidos tiveram em conta não só a negociação entre agentes individuais como também a negociação entre coligações e a negociação dentro de cada uma das coligações [Fonseca, 2000].

Embora a negociação seja muitas vezes limitada à discussão do preço de cada um dos bens ou serviços negociados, no sistema *MACIV*, os autores implementaram um sistema de negociação multi-atributo em que cada negociador possui a sua forma de avaliação pessoal de cada produto baseada nos seus múltiplo atributos [Fonseca, 2000]. Os agentes do sistema incluem ainda capacidades de aprendizagem por reforço, adaptando-se automaticamente, em tempo real às condições que se verificam em cada momento.

Embora o sistema *MACIV* se destine a um problema e domínio particular, as suas metodologias de negociação, formação de coligações e aprendizagem são facilmente generalizáveis para outros domínios. Por outro lado, o sistema em si, constitui uma plataforma de negociação multi-atributo através de leilões com continuidade temporal e formação de coligações.

4.8.4 FOREV - Formação de Empresas Virtuais

O sistema *ForEV (**Formação de Empresas Virtuais**)* é uma plataforma de negociação computacional com o objectivo principal de permitir a formação de empresas virtuais compostas por empresas independentes [Macedo, 2001] [Oliveira e Rocha, 2000]. Estas empresas virtuais formam-se para responder aos requisitos do mercado aberto de forma a, em conjunto, possuírem as características necessárias que não possuem individualmente para realizar um determinado negócio. A figura 25 apresenta um diagrama da formação de uma empresa virtual no contexto do sistema *ForEV*.

O sistema utiliza três tipos de agentes básicos [Macedo, 2001]: agente mercado, agente registo e agentes empresa. O agente de mercado modeliza a alto-nível os clientes e as suas necessidades que irão originar a formação das empresas virtuais. O agente registo é responsável pela criação e manutenção do local onde os agentes mercado e empresas irão interagir. Os agentes empresa modelizam as diversas empresas que potencialmente poderão formar a empresa virtual.

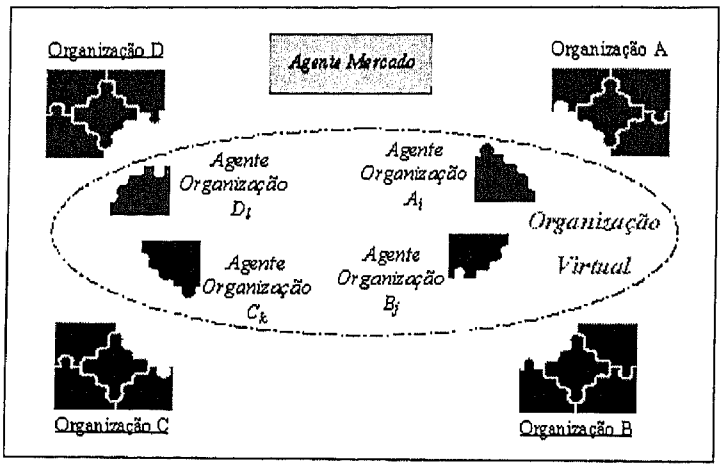

Figura 25: Formação de uma Empresa Virtual no Sistema FOREV [Macedo, 2001]

O processo de formação de empresas virtuais é modelizado como um processo de negociação iterativa multi-atributo com preocupações na manutenção da privacidade da informação de cada empresa e inclusão de características de aprendizagem. O algoritmo de negociação desenvolvido – Negociação Q, é um processo de negociação iterativa, adaptativa, multi-critério, utilizando argumentação qualitativa [Rocha e Oliveira, 2001a] [Rocha e Oliveira, 2001b]. Neste processo, os agentes utilizam o resultado da aprendizagem, realizada ao longo de um conjunto de rondas de negociação, de forma a gerarem novas propostas mais eficazes. O algoritmo de aprendizagem utilizado no sistema é a aprendizagem por reforço: *aprendizagem-Q* mas com reforços qualitativos e não quantitativos

O sistema *FOREV* não impõe restrições ao número máximo de agentes nem à complexidade do domínio. Embora a aplicação prática do sistema tenha sido realizada num domínio bastante simples (negociações com um número reduzido de agentes, num subdomínio da indústria têxtil com um único bem – casaco – composto por três componentes: tecido, botão e máquina de costura), os seus princípios parecem generalizáveis para domínios bem mais complexos e cenários mais realistas. O sistema apresenta diversas vantagens relativamente a outros tais como a sua capacidade de implementação de negociações distribuídas com adaptabilidade e manutenção de privacidade da informação dos agentes.

4.8.5 O *AuctionBot* e a *Trading Agent Competition*

O *AuctionBot* resultou de um projecto desenvolvido pelo laboratório de Inteligência Artificial da universidade de Michigan, e é essencialmente um sistema que permite a realização de leilões [Wurman et al., 1998]. Os participantes no mercado inscrevem-se no sistema através do preenchimento de uma ficha de inscrição. Cada um dos participantes no mercado pode criar novos leilões para a negociação de um dado produto, especificando o tipo de leilão e parâmetros respectivos. Para tal, o sistema disponibiliza todos os tipos de leilões mais comuns discutidos na secção 4.4 (e analisados em detalhe no anexo 1). A negociação utilizando o sistema é realizada através da Internet, não existindo comunicação directa entre os participantes. Cada participante pode consultar periodicamente o mercado, verificar quais as transacções em curso, e realizar novas propostas de compra ou venda. O sistema analisa as propostas recebidas tendo em conta o tipo e parâmetros do leilão, vai determinando o resultado provisório do leilão e notificando os participantes.

O sistema *AuctionBot* [Wurman et al., 1998] deu origem ao aparecimento de uma competição de agentes de negociação – *TAC – Trading Agent Competition* que teve a sua primeira edição em Julho de 2000, organizada por um grupo de investigadores das universidades de Michigan e North Carolina State [TAC, 2001]. Trata-se de uma competição Multi-Agente, onde os múltiplos agentes competem entre si. Os agentes são entidades autónomas que negoceiam, em diferentes tipos de leilão, a compra e venda de múltiplos bens relacionados.

Em cada jogo da TAC, são colocados oito agentes a negociar em simultâneo. Cada um dos agentes simula um agente de viagens que possui 8 clientes virtuais, que pretendem viajar da cidade *TACtown* para a cidade Tampa e voltar, num período de 5 dias. Cada cliente é caracterizado por um conjunto de preferências (determinadas de modo aleatório) para datas de partida e chegada, quartos de hotel, e tipos de entretenimento. De modo a obter utilidade para o seu cliente, o agente TAC tem de construir um pacote de viagem completo que inclui bilhete de avião de ida e volta para *TACtown*, reservar quarto de hotel para o período correspondente, e adquirir bilhetes de entretenimento apropriados para esse cliente durante o seu período de estadia, i.e. respeitando as preferências do cliente. A pontuação de um agente TAC num jogo é contabilizada pela diferença entre a soma das utilidades dos seus clientes para os pacotes que recebem, e a despesa total do agente na aquisição das viagens, hotéis e bilhetes de divertimentos.

Os agentes TAC negoceiam num mercado que é gerido por um servidor colocado na universidade de Michigan [Wurman et al., 1998]. Os agentes TAC compram e vendem voos, quartos de hotel e bilhetes de entretenimento em diferentes leilões, enviando as suas propostas para o servidor. O servidor actualiza o mercado convenientemente, executa as transacções e comunica numa linguagem baseada em *XML* com os agentes, enviando-lhes informação actualizada.

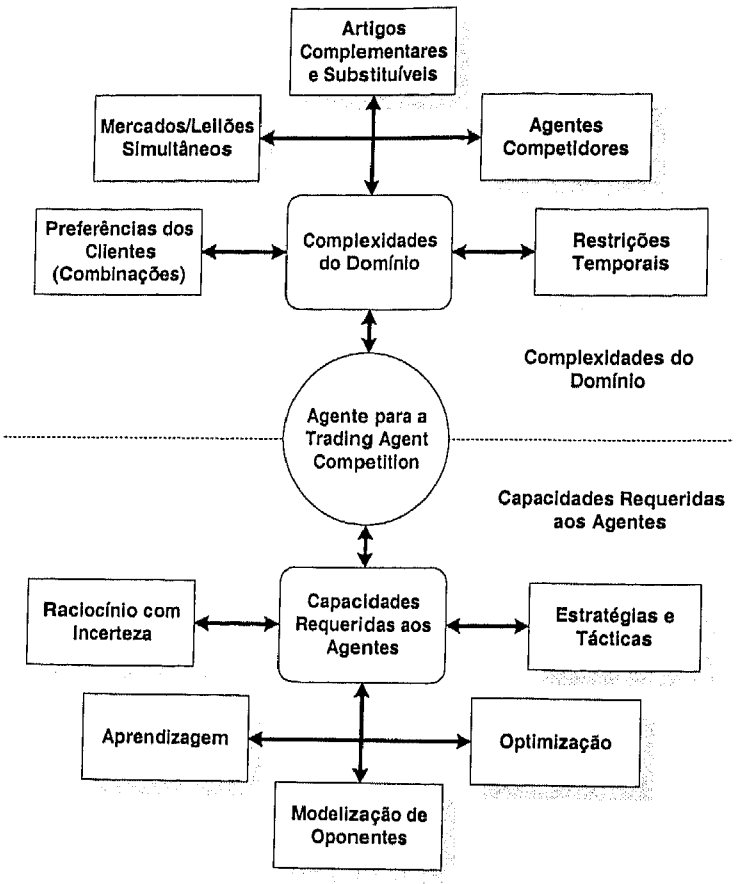

Figura 26: Diagrama da TAC - Trading Agent Competition

Cada jogo tem a duração de 12 minutos, e inclui um total de 28 leilões de três tipos diferentes: 8 leilões para os voos, 8 leilões para os quartos de hotel, e 12 leilões para os bilhetes de entretenimento. Todos os jogos são diferentes, devido não só ao facto do mercado ser imprevisível e dinâmico, mas também aos diferentes perfis de clientes atribuídos inicialmente aos agentes *TAC*, e ainda ao diferente número de bilhetes de cada um dos tipos de entretenimento (atribuição aleatória). Todos os clientes possuem valores preferenciais para datas de chegada e partida, hotel e valores de entretenimento de cada um dos tipos.

Este domínio de aplicação abrange campos de investigação interessantes. Os agentes negoceiam num ambiente muito complexo e dinâmico, com adversários desconhecidos. Para sobreviver em tal ambiente, os agentes podem usar estratégias de negociação complexas, incluindo tácticas, modelização dos oponentes, raciocínio com incerteza, capacidades de aprendizagem e capacidades de resolução de problemas de optimização (figura 26). Uma característica interessante a realçar no desempenho dos agentes *TAC* é o facto de estes serem obrigados a resolver problemas combinatórios, onde os bens disponíveis para o agente são agrupados em pacotes, e entregues como tal aos clientes.

De realçar nesta competição o facto de em 2001 [TAC, 2001] as equipas classificadas em 1º lugar - *Livingagents* (Clemens Fritschi e Klaus Dorer – Living Systems AG) e 2º lugar - *ATTac* (Peter Stone, Robert Schapire et al. – ATT Labs) na competição, terem sido

construídas pelos autores das equipas que se classificaram em 2° lugar (*Magma Freiburg* – Klaus Dorer) e em 1° lugar (*CMUnited99* – Peter Stone) no RoboCup 99 [Veloso et al. 2000]. Este facto demonstra que metodologias semelhantes tais como a aprendizagem ou o raciocínio estratégico podem ser aplicadas com sucesso em ambos os domínios.

4.9 Conclusões

A negociação desempenha um papel central na investigação realizada em sistemas multi-agente. Nomeadamente, a negociação continua a desempenhar um papel crucial entre as metodologias de coordenação, sobretudo para agentes competitivos. A investigação em coordenação por negociação, continua em grande expansão como demonstra o recente aparecimento (no ano de 2000) e imediato sucesso da competição de investigação *TAC* para agentes negociadores. Os investigadores da área, procuram analisar cenários cada vez mais complexos e realistas, incluindo mercados abertos, protocolos iterativos mais realistas de negociação, negociações multi-atributo e negociações interrelacionadas simultâneas.

Da discussão realizada ao longo desta secção, torna-se claro que muita investigação é ainda necessária nas áreas da coordenação de agentes competitivos, resolução de conflitos, negociação convencional, negociação automática e leilões. Esta investigação inclui a extensão de alguns dos métodos e sistemas apresentados mas também o desenvolvimento de novas metodologias e sistemas. Diversas áreas de investigação neste contexto, parecem estar ainda em aberto. Por exemplo, a forma de representação do conhecimento de um dado humano que permita a um agente representá-lo convenientemente durante um processo complexo de negociação. Este tipo de conhecimento deverá incluir as estratégias e tácticas de negociação a adoptar mas também, deve definir o grau de liberdade que o agente terá no decurso da negociação. No entanto, a maioria das metodologias analisadas não utilizam meta-raciocínios complexos necessários em domínios mais realistas.

A resolução de conflitos é um tema intimamente ligado à negociação. Como foi analisado ao longo do capítulo, na comunidade de sociologia e psicologia é usual que a negociação seja vista como um processo essencialmente destinado à resolução de conflitos. Estes conflitos podem surgir por diversos motivos e os seus processos de resolução podem também ser variados. No entanto, é importante realçar que a existência de um conflito não possui unicamente aspectos negativos. Aspectos positivos do conflito incluem, entre outros, impedir o imobilismo, estimulando a comunicação, criatividade e mudança, conduzindo à resolução de problemas e reconciliação. No entanto, para que tal se verifique, mecanismos adequados de resolução de problemas têm de se encontrar disponíveis, tais como protocolos adequados de negociação.

Na negociação computacional colocam-se diversas questões que derivam directamente da negociação convencional. Existem dois tipos de negociação (distributiva e integrativa) que no entanto não vivem isolados. Aliás, é muito frequente encontrar processos de

negociação mistos que possuem características de ambos os tipos de negociação. No âmbito da negociação é importante analisar e utilizar o conceito de espaço de negociação e protocolo de negociação. É muito importante definir um protocolo de negociação adequado ao tipo de negociação e realizar.

Os leilões devido à sua simplicidade e clareza são muito utilizados na negociação automática. Diversos tipos de leilões foram propostos baseados essencialmente nos tipos principais de leilão vulgarmente utilizadas pelos humanos. No entanto, os diversos tipos de leilões são vulneráveis a coligações entre compradores, compradores mentirosos, leiloeiros mentirosos, etc. Desta forma, torna-se necessário escolher o tipo de leilão adequado ao ambiente, agentes participantes e produtos ou serviços vendidos.

Por outro lado, para além de definir um protocolo de negociação eficiente e adequado ao tipo de negociação a realizar, é também necessário dotar os agentes de capacidades de negociação adequadas de modo a poderem realizar as suas propostas e contra-propostas. Diversos modelos baseados na teoria dos jogos, planeamento ou técnicas humanas foram propostos neste âmbito. Neste contexto assume particular relevância a negociação multi-atributo, não só pelos problemas de investigação adicionais que coloca mas sobretudo pelo seu realismo e proximidade com as negociações humanas. No entanto a investigação realizada sobre este tipo de negociação é ainda reduzida. Em grande parte dos casos, as relações entre os diversos atributos não são exploradas e funções lineares de avaliação são utilizadas baseadas numa soma pesada dos diversos atributos.

Intimamente relacionada com negociações mais realistas e com a negociação multi-atributo estão os conceitos de táctica e estratégia que emergiram nos últimos anos na comunidade de negociação em SMA. Uma estratégia de negociação corresponde a uma forma global de encarar a negociação. É essencialmente um plano global flexível para atingir o objectivo da negociação. Diversos autores constroem estratégias compostas por tácticas de negociação que são activadas de acordo com o decurso da negociação e o comportamento dos oponentes. No entanto, o modo de definição das tácticas e o seu agrupamento com vista a criar uma estratégia coerente de negociação, estão longe de ser temas consensuais entre os investigadores da área. Desta forma, este campo de investigação está ainda muito em aberto mas devido à sua importância irá constituir uma das áreas preferenciais de investigação em negociação em SMA, ao longo dos próximos anos.

Capítulo 5

5. Coordenação de Agentes Cooperativos

Nos capítulos anteriores foram discutidos os conceitos de Agente Autónomo e Sistema Multi-Agente e analisadas as principais metodologias de coordenação de agentes competitivos ("*self-interested*"). No entanto, embora tenha sido analisado como colocar agentes a comunicar de forma compreensível, resolver conflitos e chegar a acordos com outros agentes, não foi ainda discutida a forma como se podem coordenar conjuntos de agentes para realizarem trabalho cooperativo, em equipa.

Embora a negociação seja uma metodologia útil também na coordenação de agentes cooperativos, diversas metodologias mais adequadas à coordenação deste tipo de agentes foram desenvolvidas pelos investigadores da área. Neste capítulo são descritos modelos de coordenação com o objectivo de permitir a agentes cooperativos trabalhar em conjunto, de forma harmoniosa, em ambientes complexos. Ao longo da discussão será dado ênfase particular às metodologias adequadas à coordenação de equipas de agentes que operam em ambientes abertos e multi-objectivo.

Inicialmente são distinguidas as áreas da coordenação de agentes cooperativos e trabalho em equipa ("*teamwork*"). Em seguida é apresentada a investigação realizada em resolução distribuída e cooperativa de problemas. Diversos tipos de coordenação propostos na literatura da especialidade são descritos e comparados, incluindo: coordenação por troca de informação e resultados, coordenação por planeamento multi-agente, coordenação por modelização mútua e coordenação por organização estrutural. São também analisados modelos de coordenação mais específicos propostos por diversos autores, incluindo as intenções conjuntas, planos partilhados, *Locker Room Agreement*, o esquema *TAEMS* e o modelo *STEAM*. Propõem-se ainda tipos de coordenação distintos dos usuais, incluindo a coordenação por percepção inteligente, controlo parcialmente hierárquico e coordenação estratégica. O capítulo conclui-se com uma análise crítica da investigação realizada na área e com alguns apontadores para a forma como alguns dos conceitos apresentados são utilizados e estendidos no âmbito do trabalho prático realizado no contexto desta tese.

5.1 Objectivos da Coordenação de Agentes Cooperativos

No âmbito de uma comunidade de agentes cooperativos, coordenação pode ser definida como o processo pelo qual os agentes raciocinam acerca das suas acções locais e antecipam o efeito das acções dos outros de modo a assegurar que a comunidade actue de maneira coerente [Jennings, 1996].

Sem coordenação, os benefícios da resolução distribuída de problemas ou execução distribuída de tarefas desaparecem e a comunidade de agentes pode degenerar numa colecção caótica e incoerente de agentes individuais.

Lesser e Corkill afirmam que os objectivos da coordenação de agentes são [Lesser e Corkill, 1987]:

- Garantir que todas as partes necessárias do problema são resolvidas por pelo menos um agente ou que todas as partes da tarefa cooperativa são executadas;
- Garantir que todos os agentes interagem de modo a permitir que as suas actividades sejam executadas como parte da execução da tarefa cooperativa e desta forma integradas na solução global;
- Garantir que os membros da equipa agem de maneira a atingir os objectivos globais e de forma globalmente consistente;
- Garantir que os três objectivos anteriores são alcançados dentro dos limites computacionais e dos recursos disponíveis.

Um modelo de coordenação providencia um esquema formal, pelo qual as formas de interacção entre os agentes podem ser expressas. Assim, este modelo irá lidar com a criação e destruição de agentes, actividades de comunicação dos agentes, distribuição e mobilidade espacial dos agentes, sincronização, distribuição e monitorização das acções dos agentes ao longo do tempo (figura 27).

Um modelo de coordenação pode ser visto como consistindo em três elementos [Ciancarini, 1996]:

- **Coordenáveis.** As entidades cujas interacções mútuas vão ser regidas pelo modelo de coordenação, i.e., os agentes no Sistema Multi-Agente;
- **Meios de Coordenação.** As abstracções utilizadas que permitem as interacções entre os agentes e o núcleo em torno do qual os agentes se encontram organizados. Isto inclui meios de interacção físicos, canais de comunicação, etc;
- **Leis de Coordenação.** Definem o modo de utilização dos meios de coordenação nos eventos de interacção. Estas leis podem ser definidas de diversas formas como por exemplo através de uma linguagem de comunicação (que define a sintaxe utilizada para troca de dados) e uma linguagem de coordenação (que

define as primitivas de interacção e a sua semântica).

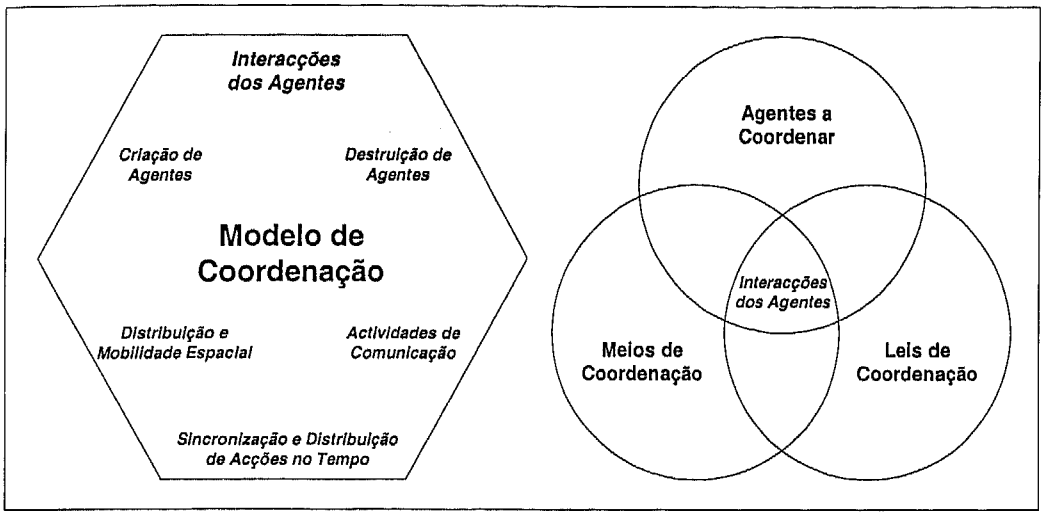

Figura 27: Modelo de Coordenação

5.2 Formação de Equipas de Agentes

Uma outra designação frequente para a coordenação de equipas de agentes cooperativos é a denominação de *trabalho de equipa* ("*teamwork*"). *Trabalho de equipa* pode ser definido como um esforço cooperativo dos membros de uma equipa para atingir um objectivo comum [AHD, 2000].

Uma *equipa* pode ser definida como um conjunto distinguível de dois ou mais indivíduos que interagem de forma dinâmica, interdependente e adaptativa de modo a atingirem objectivos comuns, partilhados e especificados.

Tambe [Tambe, 1997] adopta a designação – *trabalho de equipa* – no seu trabalho relacionado com a coordenação de equipas de agentes e afirma que o *trabalho de equipa* é cada vez mais crítico em ambientes multi-agente, tais como: treino virtual [Tambe, 1995] [Rao et al., 1993], educação interactiva, integração de informação na Internet [Williamson et al., 1996], futebol robótico simulado [Kitano et al., 1997], entretenimento interactivo [Hayes-Roth et al., 1995] e missões multi-robô.

Embora a coordenação e o trabalho de equipa estejam relacionados de forma muito próxima, pode-se afirmar que o trabalho de equipa é uma subárea da coordenação no sentido de que algumas técnicas de coordenação não podem ser consideradas trabalho de equipa. Por exemplo, o trânsito normal de veiculos, embora coordenado não pode ser considerado trabalho de equipa. Por oposição, conduzir em caravana de veículos é claramente trabalho de equipa. A diferença entre as duas situações é que embora em ambos os casos exista coordenação, o trabalho de equipa envolve objectivos comuns e cooperação entre os membros de equipa, o que se verifica na condução em caravana, mas

não no trânsito normal, onde existem objectivos globais que excedem os objectivos próprios ou comuns da equipa [Tambe, 1997].

5.3 Resolução Distribuída e Cooperativa de Problemas

A investigação em resolução distribuída de problemas iniciou-se com os trabalhos de Smith e Davis [Smith e Davis, 1980] e mais tarde de Durfee e Lesser [Durfee e Lesser, 1987]. Durfee e Lesser [Durfee e Lesser, 1987], estudaram sistemas compostos por múltiplas entidades, cada qual possuindo conhecimento distinto mas no entanto interrelacionado com o das restantes que, em conjunto, pudessem resolver problemas complexos. A resolução distribuída e cooperativa de problemas (RDCP), estuda o modo como um conjunto de entidades computacionais, ligadas em rede, pode trabalhar em conjunto de maneira a resolver problemas que são demasiado complexos para resolver com as suas capacidades individuais [Durfee et al., 1989]. Cada entidade computacional possui capacidade de efectuar a resolução de problemas complexos dentro do seu domínio de especialidade e é capaz de operação autónoma e independente. No entanto, o problema global a ser resolvido por este conjunto de entidades, não é passível de ser resolvido completamente sem cooperação. A cooperação é necessária devido ao facto de nenhuma entidade computacional ter conhecimento e recursos suficientes para resolver o problema completo. No entanto, cada entidade computacional, possui conhecimento e capacidades para resolver partes do problema global [Durfee et al., 1989].

A resolução distribuída de problemas é composta por três fases [Smith e Davis, 1980] que podem ocorrer iterativamente:

- **Decomposição do Problema.** O problema original é decomposto em diversos subproblemas de dimensão e complexidade menor. Cada um dos subproblemas é alocado a um ou mais agentes para efectuar a sua resolução;
- **Resolução dos Subproblemas.** Os subproblemas são resolvidos pelos diversos agentes envolvidos no processo;
- **Síntese da Solução**. A solução final é obtida através da fusão das respostas obtidas pelos diferentes agentes para cada um dos subproblemas.

No trabalho de Durfee e Lesser, as entidades computacionais ou agentes são consideradas benevolentes, no sentido em que todos os agentes do sistema supostamente partilham um objectivo global, e como tal não existe a necessidade de resolver conflitos entre eles. Os agentes são totalmente cooperativos e nesse sentido procuram auxiliar os outros agentes do sistema se tal for benéfico para o conjunto, mesmo que tal implique custos individuais. Esta assunção de benevolência e de que os agentes estão unicamente preocupados com a utilidade do sistema só poderá ser válida se os agentes forem todos projectados pelo

mesmo indivíduo ou organização [Wooldridge, 2002]. Ao contrário, em Sistemas Multi-Agente que contêm agentes competitivos, não pode ser assumido que todos os agentes partilham o mesmo objectivo pois muitas vezes eles são projectados e implementados por diferentes indivíduos ou organizações de forma a os representarem nesse mesmo sistema. Os conflitos irão naturalmente emergir neste tipo de sociedade, tal como emergem em sociedades humanas, sem que, no entanto, isto signifique que a cooperação não seja útil [Wooldridge, 2002]. Aliás, a criação de sociedades de agentes e consequente definição de metodologias de coordenação e cooperação, que permitam a interacção desses agentes na sociedade assumem-se como algumas das áreas de investigação com maior crescimento nos últimos anos, no âmbito da àrea dos Sistemas Multi-Agente.

A implementação de uma sociedade de agentes capaz de resolver problemas complexos, coloca a questão de como avaliar a solução obtida e quais os critérios a considerar para avaliar o sucesso da implementação. Uma possibilidade consiste na utilização da coerência como medida desse sucesso. Neste contexto, Wooldridge, propõe que a coerência pode ser medida em termos da qualidade, eficiência, utilização de recursos, clareza e tolerância a incerteza e/ou falhas [Wooldridge, 1994].

A Resolução Distribuída Coerente de Problemas, realizada de maneira satisfatória através de um SMA, implica a resposta a múltiplas questões. Wooldridge afirma que as principais preocupações e tópicos de investigação na RDCP são [Wooldridge, 2002]:

- Como pode um problema ser decomposto em tarefas menos complexas?
- Como podem essas tarefas ser distribuídas entre os agentes do sistema?
- Como pode a solução global do problema ser sintetizada a partir das soluções parciais dos vários subproblemas?
- Como podem as actividades de resolução de problemas dos agentes ser optimizadas de forma a produzir uma solução que optimize a coerência?
- Que metodologias devem ser utilizadas de modo a coordenar a actividade dos agentes, evitar interacções destrutivas e maximizar a eficiência das interacções positivas?

Diversos autores prosseguiram o trabalho de Durfee e Lesser e procuraram desenvolver metodologias para responder a estas questões. As secções seguintes apresentam algumas das metodologias mais relevantes desenvolvidas neste contexto.

5.3.1 Coordenação por Partilha de Tarefas e de Resultados

Uma das primeiras metodologias de coordenação propostas para agentes autónomos foi sugerida por Smith e Davis [Smith e Davis, 1980]. No seu trabalho sugeriram a separação do problema de resolução distribuída cooperativa de problemas nas três fases apresentadas

na secção anterior: decomposição do problema, resolução dos subproblemas e síntese da solução.

Na primeira fase, o problema global é decomposto em problemas de dimensão inferior. Esta decomposição é tipicamente hierárquica para que os subproblemas possam por sua vez ser decompostos em problemas ainda menores e assim sucessivamente. A decomposição sucessiva continua até se atingirem problemas de uma dimensão tratável por agentes individuais. No entanto, é importante ter em consideração que os subproblemas gerados devem ter o mínimo de interdependências possível para que não seja complexa a sua resolução separada e a integração das suas soluções individuais. A decomposição do problema deve, idealmente, ter em consideração o conhecimento sobre os agentes que irão resolver os subproblemas gerados.

Na segunda fase do processo, os subproblemas identificados na primeira fase são resolvidos individualmente. Esta fase envolve uma alocação prévia dos problemas aos agentes e a troca de informação entre os agentes durante a resolução. Quanto maior for o número de interdependências entre os subproblemas maior será a necessidade de comunicação e troca de informação entre os agentes durante a sua resolução. Esta troca de informação pode ter lugar de forma reactiva ou pró-activa e consiste, usualmente, no envio de informação que um dado agente considere útil para a(s) tarefa(s) que um outro agente esteja a executar.

A fase final consiste na integração das soluções dos subproblemas individuais numa solução global. Tal como na primeira fase, esta fase pode ser realizada de uma forma hierárquica. A principal dificuldade deste processo resulta usualmente da possibilidade de existência de soluções incompatíveis geradas por diferentes agentes.

A resolução distribuída e cooperativa de problemas (RDCP) apresenta dois cenários possíveis (figura 28):

- **Partilha de Tarefas.** A partilha de tarefas ocorre quando um dado problema é decomposto e as suas diferentes partes alocadas a agentes distintos. A principal dificuldade consiste na definição do método de alocação dos problemas aos agentes. Se os agentes possuírem todos capacidades, conhecimentos e recursos idênticos, i.e. se forem homogéneos, então a tarefa é simples. No entanto, se os agentes tiverem capacidades distintas e forem autónomos podendo inclusivamente recusar a execução de tarefas, o problema torna-se muito mais complexo. Em domínios que incluam agentes competitivos, as metodologias de coordenação discutidas no capítulo anterior (resolução de conflitos, negociação, leilões, etc.) são de grande importância na partilha de tarefas;
- **Partilha de Resultados.** A partilha de resultados implica a troca de informação entre os agentes que seja relevante para a resolução do problema global. Esta informação pode ser trocada de forma pró-activa (um agente decide autonomamente enviar informação a outros agentes que pensa lhes poder ser útil)

ou reactiva (o envio de informação é realizado como resposta a pedido de um dado agente).

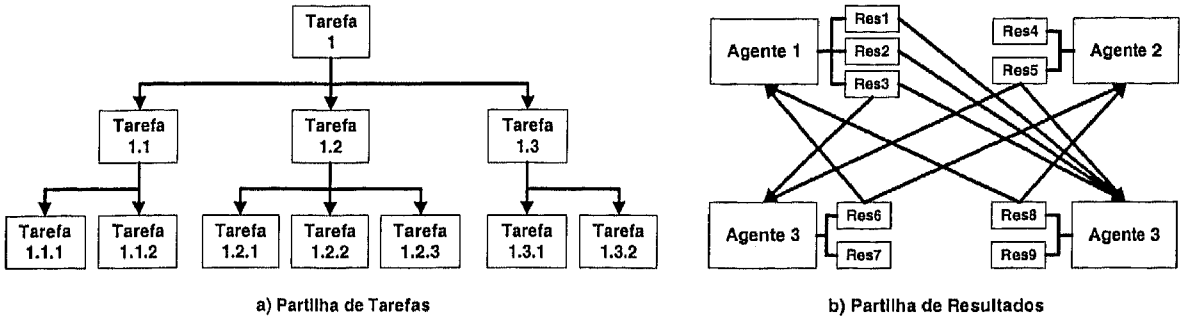

Figura 28: Tarefas Cooperativas na RDPC: a) Partilha de Tarefas e b) Partilha de Resultados

5.3.2 Coordenação por Partilha de Tarefas – Protocolo da Rede Contratual

O protocolo da rede contratual ("*Contract-Net*") destina-se a permitir a coordenação através da partilha de tarefas entre agentes capazes de efectuar a resolução de problemas [Smith, 1977] [Smith, 1980a] [Smith, 1980b] [Smith e Davis, 1980]. É essencialmente um protocolo de alto-nível que se baseia, tal como o nome indica, em efectuar a contratação de tarefas. Para efectuar a definição deste protocolo, Smith inspirou-se no modo de executar a contratação usada em empresas reais.

O protocolo *Contract-Net* pode ser considerado como um processo de contratação iniciado por um dado agente A_1, que pretende ver realizada uma dada tarefa T_1. Através do conhecimento sobre os outros agentes, esse agente A_1 verifica que existe mais do que um agente com capacidade para realizar a tarefa T_1. Este processo baseia-se num protocolo específico que consiste na troca de um conjunto de mensagens entre o agente A_1 e os outros agentes com capacidade de realização de T_1, com vista à incumbência de um desses agentes para a realização dessa tarefa T_1. O resultado da negociação pode ser a incumbência de um dado agente para a realização da tarefa pretendida T_1, mas pode também ser a verificação de que nenhum agente consegue ou está disposto a realizar a tarefa T_1, dentro das restrições pretendidas. Neste último caso, o agente organizador da negociação (A_1), conclui que não consegue obter ajuda para a realização dessa tarefa T_1 dentro das restrições impostas, e tem de tomar uma decisão sobre a alteração da sua actividade (por exemplo, tentar um plano alternativo para a obtenção da acção pretendida ou relaxar as restrições impostas para a realização da tarefa).

O protocolo é extremamente simples (figura 29). Inicia-se pelo envio, por parte do agente organizador, de um anúncio geral da tarefa a executar ou, caso o organizador possua conhecimento das capacidades dos outros agentes, de um anúncio limitado destinado ao conjunto de agentes que ele pensa poderem executar essa tarefa (obedecendo a um

conjunto de restrições pré-especificado). Os agentes receptores (ou respondentes) após receberem e interpretarem o anúncio, realizarão uma avaliação do mesmo no que diz respeito às suas possibilidades de execução da tarefa solicitada. Se considerarem que conseguem e estão dispostos a realizar a tarefa, enviam uma proposta de execução ao agente organizador. O organizador pode receber diversas propostas de execução e, como tal, deverá seleccionar o agente mais apropriado à execução da tarefa com base na informação dessas mesmas propostas. A selecção é comunicada a esse agente e a rejeição das respectivas propostas é comunicada aos restantes agentes. O protocolo pode ainda incluir a confirmação por parte do agente vencedor de que assume o compromisso de realmente executar a tarefa.

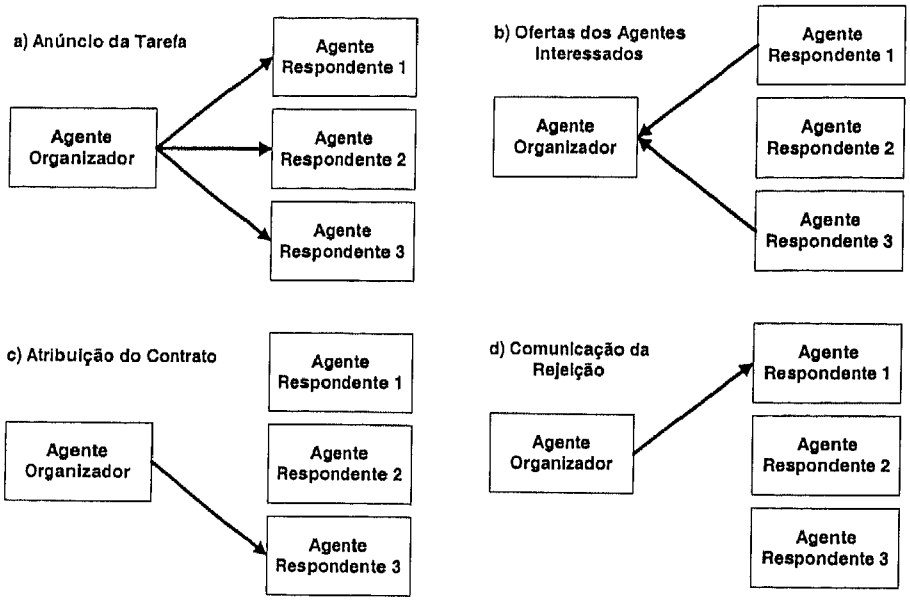

Figura 29: Protocolo Contract-Net – Rede de Contratual

O protocolo rede contratual destina-se a determinar a possibilidade de realização ou não de cooperação e em caso afirmativo a forma concreta de como a cooperação se vai realizar (qual o agente ao qual vai ser pedido para efectuar a tarefa). Apesar de ser muito simples (e eventualmente por isso mesmo), a rede contratual tornou-se no mais implementado e mais estudado método de coordenação por partilha de tarefas.

5.3.3 Coordenação por Comunicação – Partilha de Resultados

A partilha de resultados, e mais genericamente a partilha de informação e conhecimento, é essencialmente um método de coordenação por comunicação. Na literatura da especialidade, este tipo de coordenação é usualmente apresentado em conjunto com a partilha de tarefas, no âmbito da descrição de agentes que efectuam a resolução distribuída de problemas. Nesta perspectiva, os resultados partilhados correspondem a soluções de problemas menores que são progressivamente compostas de modo a formarem a solução

global do problema completo. Durfee [Durfee, 1999] sugere que os agentes podem melhorar o seu desempenho através da partilha de resultados das seguintes formas:

- **Confiança.** As soluções calculadas por diferentes agentes de modo independente podem ser analisadas em conjunto, verificando a existência de erros e aumentando a confiança na solução global;
- **Completude.** Os agentes podem comunicar as suas visões locais do problema de modo a atingir uma visão global melhor;
- **Precisão.** Os agentes podem trocar resultados de modo a aumentar a precisão da solução global;
- **Tempo.** O tempo necessário para atingir a solução pode diminuir pois, mesmo que cada agente possa resolver as suas tarefas sozinho, a disponibilidade de informação adicional pode acelerar o seu próprio processo de resolução.

Habitualmente, a partilha de resultados é combinada com a partilha de tarefas. Os agentes partilham de forma reactiva e/ou pró-activa informação parcial que possuam, de forma a auxiliar os restantes agentes na resolução dos seus problemas individuais.

Um dos principais problemas na resolução distribuída de problemas está relacionado com o aparecimento de inconsistências entre os diversos agentes presentes no sistema. Os agentes podem apresentar inconsistências quanto a:

- **Crenças (informação sobre o estado do mundo).** Este tipo de incoerência surge devido à inacessibilidade do ambiente (e consequente visão parcial do mundo disponível a cada agente), ao dinamismo do ambiente e aos eventuais erros na percepção dos agentes;
- **Objectivos (intenções que desejam atingir).** A incoerência relativa aos objectivos surge devido à autonomia dos agentes, que não partilham exactamente o mesmo conjunto de objectivos.

Quando o sistema possui uma dimensão razoável, os agentes são autónomos e o ambiente é inacessível e dinâmico, as inconsistências são inevitáveis. Para lidar com este problema, diversos autores propuseram soluções distintas incluindo:

- Ignorar as inconsistências [Durfee et al., 1989];
- Metodologias de revisão de crenças para inconsistências sobre crenças [de Kleer, 1986] [Malheiro, 1999] [Malheiro e Oliveira, 2000],
- Tratar as inconsistências como conflitos e resolvê-las através de negociação; e
- Construir sistemas suficientemente robustos capazes de raciocinar na presença de inconsistências [Lesser e Corkill, 1981].

Esta última abordagem é a mais adequada em domínios dinâmicos e inacessíveis pois será evidente que o estado do mundo dos agentes será sempre incompleto, sujeito a erros e

desactualizado. Neste contexto, Lesser e Corkill sugeriram as seguintes características para um sistema tolerante a informação inconsistente e/ou incorrecta [Lesser e Corkill, 1981]:

- **Independência Relativa aos Eventos.** Resolução de problemas não depende directamente da sequência de eventos. A resolução de problemas é executada de forma oportunística e incremental, de acordo com a disponibilidade de informação e não numa ordem pré-especificada.
- **Troca de Informação de Alto-Nível.** Os agentes trocam informação de alto-nível relacionada com os resultados intermédios e não dados em bruto. Isto permite uma muito maior autonomia e independência no comportamento dos agentes.
- **Resolução Contínua de Inconsistências.** As incertezas, incorrecções e inconsistências vão sendo resolvidas ao longo do processo de resolução de problemas e não no início e no final do mesmo.
- **Metodologia de Resolução Flexível.** A metodologia de cálculo da solução deve ser flexível e não deve estar restringida a um único caminho. A disponibilidade de múltiplos planos para atingir a solução aumenta consideravelmente a robustez do sistema, eliminando possíveis estrangulamentos nesta resolução.

A coordenação por comunicação pode também ser tratada a mais alto-nível, como no trabalho de Gasser, que propõe a troca de informação de controlo de meta-nível acerca das prioridades e acções actuais de cada agente entre os agentes do sistema [Gasser, 1992].

5.4 Coordenação por Planeamento Multi-Agente

Um tema sujeito a grande investigação na área dos Sistemas Multi-Agente é o planeamento multi-agente (planeamento das actividades de um conjunto de agentes). Neste tipo de planeamento, é necessário ter em consideração que as actividades dos diferentes agentes presentes no sistema podem interferir umas com as outras. Desta forma, é necessário coordenar tais actividades. Existem essencialmente três possibilidades para o planeamento multi-agente [Durfee, 1999]:

- **Planeamento Centralizado de Planos Distribuídos.** Um sistema centralizado de planeamento desenvolve planos para um conjunto de agentes em que a divisão e alocação de tarefas se encontra especificada. O agente central distribui os planos parciais aos restantes agentes que funcionam num nível hierarquicamente inferior (como escravos) e têm a responsabilidade da sua execução;
- **Planeamento Distribuído de um Plano Global.** Um grupo de agentes coopera de forma a gerar um plano global de actividades. Tipicamente, os agentes componentes do sistema vão ser especialistas em diferentes aspectos necessários

para o desenvolvimento do plano global. É usual que os agentes que geram o plano não sejam os mesmos que são responsáveis pela sua execução;

- **Planeamento Distribuído de Planos Distribuídos.** Um grupo de agentes coopera de forma a gerar planos individuais de acção, coordenando dinamicamente as suas actividades ao longo desse planeamento. Os agentes podem ser egoístas e como tal, os conflitos resultantes podem ser resolvidos através das metodologias de negociação analisadas no capítulo anterior.

Claro que também será possível o planeamento centralizado de um plano global, mas neste caso não estaríamos perante um sistema multi-agente. Em geral, o planeamento centralizado é mais simples do que o planeamento distribuído, pois o agente central terá uma visão global do problema e pode definir as relações de coordenação que necessitar. No entanto, este planeamento global nem sempre é possível e muitas vezes não é desejável, seja devido à elevada dimensão do problema, à necessidade de especialistas distintos, à distribuição geográfica dos agentes, ou à necessidade de manutenção de privacidade da informação. O caso mais complexo é o terceiro, em que um plano global pode nunca existir. Neste caso, os agentes produzirão em conjunto, dinamicamente, os planos parciais que serão executados por cada um dos agentes.

5.4.1 Planeamento Global Parcial

O *Planeamento Global Parcial – PGP* está intimamente ligado com a coordenação por partilha de tarefas e resultados e assenta no princípio que agentes cooperativos realizam troca de informação de forma a chegar a uma solução global para um dado problema. O planeamento para atingir a solução é *parcial* no sentido em que nenhum agente é capaz de gerar um plano completo que permita resolver o problema. É no entanto *global*, no sentido em que os agentes formam planos não-locais através da partilha de planos locais de forma a atingir a resolução do problema global. Este planeamento divide-se em três fases distintas [Durfee e Lesser, 1987] [Durfee, 1988] [Durfee, 1996]:

- **Geração de Objectivos.** Cada agente decide quais os seus objectivos e define planos de curto-prazo que lhe permitam atingir esses objectivos;
- **Troca de Informação.** Os agentes trocam informação relativamente a planos, objectivos e soluções. A troca de informação é essencial para determinar os pontos onde os planos locais dos agentes interactuam com os de outros agentes;
- **Alteração de Planos Locais.** De forma a coordenarem as suas actividades, os agentes alteram os seus planos locais de acordo com a informação disponível.

De forma a evitar a incoerência nos estados locais dos agentes, o *PGP* propõe a utilização de uma estrutura de meta-informação que define quais os agentes que devem trocar informação, qual a informação trocada, quais as pré-condições para esta troca, etc.

O *Planeamento Global Parcial* baseia-se na estrutura designada por *Plano Parcial Global*[19]. Esta é uma meta-estrutura que é gerada cooperativamente por todos os agentes e que contém:

- **Objectivos.** Os objectivos partilhados pela comunidade de agentes e que o plano gerado pelo sistema deve resolver;
- **Mapas de Actividades.** Representando as actividades actuais de cada agente e os resultados esperados dessas actividades;
- **Grafo de Construção da Solução.** Representando a forma como os agentes devem interagir e a informação que devem trocar de forma a gerar a solução final.

O *PGP* foi utilizado num dos primeiros simuladores para teste de Sistemas Multi-Agente: a Bancada de Monitorização Distribuída de Veículos[20]. Este simulador foi desenvolvido com o objectivo de permitir comparar diferentes abordagens à resolução distribuída de problemas. Permitia o teste de agentes de resolução distribuída de problemas, utilizando como problema *standard* o domínio da detecção e monitorização distribuída de veículos. O objectivo consistia em detectar e seguir um conjunto de veículos que passava por uma dada região, monitorizada por um conjunto de sensores distribuídos espacialmente. O domínio de aplicação era essencialmente apropriado para agentes reactivos, pois sempre que um novo veículo aparecia no sistema despoletava um processo de detecção e seguimento. A rapidez de processamento era essencial, uma vez que o domínio era dinâmico e exigia que os agentes rapidamente determinassem as trajectórias dos veículos presentes. Este domínio conduziu à criação do *PGP* que Durfee e Lesser propuseram, de forma a coordenar as actividades dos seus agentes de resolução distribuída de problemas [Durfee e Lesser, 1987] [Durfee, 1988] [Durfee, 1996].

5.4.2 Planeamento Global Parcial Generalizado

O *Planeamento Global Parcial* foi posteriormente estendido por Keith Decker que propôs o mecanismo designado por *Planeamento Global Parcial Generalizado*. Este mecanismo utiliza cinco novas metodologias na coordenação das actividades dos agentes [Decker, 1995]:

- **Comunicação de Informação.** A comunicação de informação é realizada com o objectivo de actualizar perspectivas não-locais. Como os agentes possuem unicamente perspectivas locais em ambientes inacessíveis, a partilha de informação pode auxiliá-los a atingir uma visão mais completa do ambiente. A

[19] Do inglês *Partial Global Plan.*

[20] Do inglês *Distributed Vehicle Monitoring Testbed.*

quantidade de informação que é comunicada com este fim pode variar de acordo com a política de comunicação definida. Desta forma, é possível comunicar: toda a informação disponível em cada instante (o que dependendo do domínio pode sobrecarregar a rede ou mesmo ser impossível); comunicar uma parte seleccionada da informação (colocando-se a questão de qual o mecanismo a utilizar na sua selecção); ou não comunicar qualquer tipo de informação.

- **Comunicação de Resultados.** Os agentes podem comunicar resultados de vários modos, similares aos modos de comunicação de informação: todos os resultados obtidos; resultados que o agente acredite interessarem a outros agentes; ou unicamente os resultados essenciais para o planeamento;
- **Tratamento da Redundância.** A redundância verifica-se quando dois ou mais agentes resolvem o mesmo problema ou subproblema. No entanto a redundância pode ser deliberada, ou seja, pode ser desejável colocar mais do que um agente a resolver um determinado problema. Este facto é útil para incrementar a fiabilidade e a qualidade da solução obtida. Em geral, a redundância conduz a um desperdício de recursos e, como tal, deve ser limitada. Decker e Lesser [Decker e Lesser, 1995] adoptam um esquema bastante simples no tratamento de redundâncias, baseado em selecção aleatória para efectuar a sua alocação, sendo esta decisão comunicada aos restantes agentes que devem cancelar a execução da actividade em causa;
- **Tratamento de Relações Rígidas de Coordenação.** As relações rígidas de coordenação ocorrem quando a execução de uma dada acção por parte de um agente interfere com a execução de outra acção por parte de outro agente. Um exemplo de uma relação deste tipo poderá ser *enables*(T_1,T_2) que indica que o final de uma tarefa T_1 é necessário para que T_2 possa ser executada [Decker, 1995]. Este tipo de relações é evitado através do reescalonamento de acções conflituosas.
- **Tratamento de Relações Flexíveis de Coordenação.** As relações flexíveis de coordenação ocorrem quando a interferência entre acções conflituosas não é crítica para o sucesso das acções (embora possa alterar significativamente a eficiência ou qualidade de execução). O reescalonamento não é aqui obrigatório embora possa ser altamente desejável, dependendo do grau de interferência entre as acções.

Em comparação com o *PGP*, o *Planeamento Parcial Global Generalizado,* acrescenta ainda o escalonamento de tarefas com *"deadlines"*, heterogeneidade nos agentes e a comunicação a múltiplos níveis de abstracção. Estas extensões tornam este mecanismo muito mais flexível e utilizável na prática. O mecanismo foi implementado completamente no simulador *TAEMS* [Decker, 1996] analisado na secção 5.5.4.

5.5 Modelos de Coordenação

5.5.1 Intenções Conjuntas ("*Joint Intentions Framework*")

A *Teoria das Intenções Conjuntas (Joint Intentions Framework)* foi proposta por Cohen et al. [Cohen et al., 1990] como uma metodologia para a cooperação em Sistemas Multi-Agente. O trabalho relacionado com as intenções conjuntas iniciou-se com a formalização lógica do que são *Intenções Individuais* realizada por Cohen e Levesque [Cohen e Levesque, 1990] baseada na definição de *Intenção* dada por Bratman [Bratman, 1990]. Esta formalização foi designada *Objectivos Persistentes ("Persistent Goals")* e descreve a forma como as *intenções* de um agente se relacionam com as suas *crenças*, *compromissos* e *acções*.

A *Teoria das Intenções Conjuntas* está focada no estado mental conjunto de uma equipa de agentes, designado por *Intenção Conjunta* [Cohen e Levesque, 1991]. Uma equipa T deseja conjuntamente uma determinada acção de equipa, se todos os seus membros estiverem comprometidos a completar essa acção e simultaneamente acreditam que a estão a realizar. Um compromisso conjunto é definido neste contexto como um *objectivo conjunto persistente (OCP)*. Um dado *OCP* para executar uma dada acção para atingir o objectivo p, é denotado por *OCP(p)*. O *OCP* da equipa para atingir p, pode ser denotado por *OCP(T,p,q)* onde q é uma condição de irrelevância que permite à equipa abandonar a execução do *OCP* se acreditar que q é falso. O objectivo conjunto *OCP* permanecerá em execução se todos os membros da equipa acreditarem que existe uma motivação comum M para realizar p, esse objectivo p ainda não foi atingido e p ainda é realizável.

Um *OCP* é então um compromisso colectivo para realizar a execução de um determinado objectivo. O estado mental de uma equipa de agentes que possui um dado *OCP* pode ser descrito como:

- Inicialmente todos os agentes acreditam que:
  - É possível realizar o objectivo p;
  - O objectivo p ainda não foi realizado;
  - A motivação (justificação) M para a realização de p encontra-se presente.
- Todos os agentes da equipa possuem o objectivo p até que uma condição de terminação Z seja verdadeira.
- A condição de terminação Z consiste em que seja mutuamente acreditado que uma das seguintes condições se verifica:
  - O objectivo p está satisfeito;
  - O objectivo p é impossível de satisfazer;

- A motivação *M* para realizar *p* já não se encontra presente.

- Até que a condição *Z* seja satisfeita, cada agente:
  - Se acreditar que *p* foi atingido, então terá o objectivo de que isto se torne mutuamente acreditado e manterá este objectivo até que *Z* se verifique;
  - Se acreditar que *p* é impossível, então terá o objectivo de que isto se torne mutuamente acreditado e manterá este objectivo até que *Z* se verifique;
  - Se acreditar que a motivação *M* já não é válida, então terá o objectivo de que isto se torne mutuamente acreditado e manterá este objectivo até que *Z* se verifique;

A *Teoria das Intenções Conjuntas* baseia-se essencialmente no comportamento de equipas humanas. Quando os humanos trabalham em equipa, os estados mentais semelhantes a intenções desempenham um papel importante no comportamento da equipa [Levesque et al., 1990] [Cohen e Levesque, 1991]. Neste contexto, Levesque estudou a distinção entre acções coordenadas cooperativas e acções coordenadas não-cooperativas [Levesque et al., 1990]. Do seu estudo surgiu o conceito de responsabilidade para com os outros membros da equipa. Para além do objectivo global da equipa, partilhado por todos os agentes, surgem os objectivos de cooperação com os outros agentes da equipa. Estes objectivos consistem em informar os restantes elementos: do atingir dos objectivos globais, da impossibilidade de realizar os objectivos globais ou da não validade das justificações para conseguir atingir os objectivos globais.

Jennings estendeu o trabalho de Levesque et al. e distinguiu *compromissos* de *convenções* [Jennings, 1993] [Jennings, 1995]. *Compromissos* são promessas para realizar objectivos enquanto *convenções* são meios de monitorizar o compromisso. As *convenções* especificam as circunstâncias em que os *compromissos* podem ser abandonados e o modo como os agentes se devem comportar se tal se verificar (localmente e relativamente aos restantes membros da equipa).

Quando um grupo de agentes se comporta como uma equipa, deve possuir um *compromisso conjunto* para o objectivo global e *compromissos individuais* para as suas tarefas individuais. Quer o *compromisso conjunto*, quer os *compromissos individuais* devem ser persistentes. O *compromisso conjunto* difere dos individuais, no sentido em que é distribuído por todos os membros da equipa. Uma convenção social (partilhada por todos os agentes) identifica as condições nas quais o *compromisso conjunto* pode ser abandonado e como os agentes se devem comportar para com os seus companheiros de equipa [Jennings, 1993]. A existência de uma convenção social partilhada por todos, permite aos agentes possuir um modelo de referência relativamente ao comportamento cooperativo dos colegas de equipa, que pode ser de extrema utilidade para a definição do seu próprio comportamento.

5.5.2 Planos Partilhados ("*Shared Plans*")

O modelo dos *Planos Partilhados (Shared Plans)* [Grosz e Kraus, 1996] [Grosz e Sidner, 1990], baseia-se numa atitude mental distinta do modelo das *intenções conjuntas*: ter a intenção de realizar uma dada acção [Bratman, 1992] [Grosz, 1996]. Esta intenção é muito similar à intenção de um agente individual realizar uma acção. No entanto, é uma intenção individual que é dirigida para as acções dos restantes elementos da equipa ou para uma acção conjunta global da equipa. A intenção é definida através de um conjunto de axiomas que guia os indivíduos na selecção e execução de acções, incluindo acções de comunicação que facilitem ou permitam aos seus colegas a realização de um conjunto de tarefas [Grosz e Kraus, 1996].

Um plano partilhado pode consistir num *plano completo partilhado (PCP)* ou num *plano parcial partilhado (PPP)*. Um *plano completo partilhado* para realizar uma dada tarefa conjunta T, consiste numa representação em que todos os aspectos da actividade conjunta T estão especificados. Isto incluí uma crença e acordo mútuo numa receita R_T para realizar T. A receita R_T especifica um conjunto de acções A, que quando executadas tendo em conta restrições específicas, constituem a realização de T. $PCP(P, G, T, t_p, t_T, R_T)$ denota um plano conjunto P de um grupo G, no instante de tempo t_p para fazer uma acção conjunta T, até ao tempo t_T, utilizando a receita R_T. O *plano completo partilhado* $PCP(P, G, T, t_p, t_T, R_T)$ será executado se:

- Todos os agentes que pertencem ao grupo G acreditam que R_T é uma receita válida para executar T;
- Todos os agentes que pertencem ao grupo G acreditam que todos desejam que o grupo execute T até ao instante de tempo t_T;
- Para cada acção A_i pertencente ao conjunto A de acções da receita R_T:
  - Existe um subgrupo G_i que possui um *PCP* para A_i utilizando uma receita R_{Ai} ou existe um único agente que possui um plano completo individual para realizar essa acção;
  - Os restantes membros do grupo G acreditam que existe uma receita disponível ao grupo G_i que permita a este grupo executar A_i utilizando um dado plano;
  - Os restantes membros do grupo G desejam que o grupo G_i consiga executar A_i utilizando um qualquer plano;

A teoria dos planos partilhados procura deste modo descrever a rede de intenções de toda a equipa no decurso da execução da tarefa cooperativa. Habitualmente, os grupos só possuem planos parciais, o que torna necessário a comunicação e o planeamento, de modo a obter um *PCP*. A diferença principal entre os *PCP* e os *PPP* é que nos segundos as receitas podem não estar completamente especificadas. Tal facto torna-se relevante em

ambientes dinâmicos, como é o caso do futebol robótico, onde a equipa tem de ser reactiva às novas situações que lhe aparecem ao longo do tempo, implicando também a necessidade de alterar estas receitas de execução.

5.5.3 Acordo de Balneário ("*Locker Room Agreement*")

Um dos principais problemas das intenções conjuntas e dos planos partilhados é que a equipa é vista como um conjunto de agentes que negoceia ou contrata um plano conjunto a ser executado. A equipa forma-se dinamicamente e comunica livremente até que a execução do seu plano esteja concluída [Stone e Veloso, 1999]. Diversos domínios exigem um esforço de equipa persistente no sentido de atingir objectivos comuns num ambiente com comunicação limitada.

Stone e Veloso [Stone e Veloso, 1999] [Stone, 1998] introduziram o conceito de *locker-room agreement* (*acordo de balneário*) como mecanismo de coordenação de alto-nível útil em domínios com reduzida comunicação. Embora alguns autores [Tambe, 1997] afirmem que a definição de acções pre-determinadas não é suficientemente flexível e robusta a falhas em domínios dinâmicos, o sucesso do *locker-room agreement* num domínio tão dinâmico como os das competições da liga de simulação do futebol robótico veio provar exactamente o contrário[21].

O mecanismo baseia-se na definição de uma estrutura flexível de equipa baseada em [Stone, 1998]:

- Papéis (*roles*) que especificam o comportamento de um agente individual e mecanismos para realizar a sua troca entre agentes;
- Formações compostas por colecções de papéis e condições para a activação destas formações;
- Planos multi-agente (*set-plays*) para execução em situações específicas.

Esta estrutura, partilhada à partida por todos os membros da equipa, permite-lhes actuar em domínios dinâmicos e com comunicação restringida ou mesmo inexistente. Os papéis especificam o comportamento de um agente individual de modo flexível. As formações decompõem o espaço de tarefas num conjunto de papéis que são posteriormente atribuídos aos vários agentes. Numa equipa de n agentes, $T = \{a_1, a_2, a_3, ..., a_n\}$, onde a_i representa o agente i, uma formação assume a forma:

$$F = \{R, \{U_1, U_2, ..., U_k\}\}$$

[21] A equipa CMUnited, baseada no *locker-room agreement*, venceu os campeonatos do mundo de futebol robótico simulado de 1998 e 1999, marcando um total de 176 golos, sem sofrer nenhum, nessas duas competições.

Onde $R = \{r_1, r_2, r_3, \ldots, r_n\}$ tal que $i \neq j \Rightarrow r_i \neq r_j$ representa um conjunto de papéis.

No *locker-room agreement,* o número de papéis é igual ao número de agentes que compõem a equipa, sendo no entanto possível definir papéis redundantes. Cada unidade U_i é um subconjunto de R: $U_i = \{r_{i1}, r_{i2}, \ldots r_{ik}\}$ tal que $r_{ia} \in R$, $a \neq b \Rightarrow r_{ia} \neq r_{ib}$ e r_{i1} é o capitão ou líder da unidade. O mapeamento dos agentes nos papéis, $T \rightarrow R$, não é fixo, ou seja, os papéis podem ser preenchidos por diferentes agentes homogéneos. Cada papel pode fazer parte de qualquer formação ou unidade. As unidades são utilizadas para lidar com a resolução de problemas locais, em vez de utilizar toda a equipa num dado subproblema. Os capitães são elementos da equipa com capacidades especiais e são responsáveis por dirigir os membros de uma dada unidade.

O *locker-room agreement* especifica uma formação inicial, um mapeamento inicial dos agentes aos papéis e condições para a realização da mudança automática de formação. Cada agente mantém em memória qual a formação actual da equipa e qual o mapeamento dos agentes aos papéis nessa formação. Como os agentes podem dispor de mapeamentos inexactos $T \rightarrow R$ os autores desenvolveram comportamentos individuais robustos, que não dependem da exactidão da informação relativa ao mapeamento dos papéis aos agentes [Stone, 1998]. Quando a comunicação não for possível ou for bastante restringida, os agentes procuram utilizar a largura de banda disponível para informar os restantes agentes dos seus papéis, de forma a sincronizar a equipa. Em situações em que seja possível comunicar sem restrições, a equipa sincroniza-se facilmente sendo partilhada a informação relativa a quais os agentes que executam cada um dos papéis [Stone e Veloso, 1999].

Como parte do *locker-room agreement*, a equipa pode definir planos multi-agente compostos por vários passos que serão executados em momentos apropriados. Quando ocorrem frequentemente situações repetitivas, é útil possuir planos pré-definidos para lidar com tais situações. A definição de um plano multi-agente (*set-play*) neste contexto é composta por: uma condição que indica quando o plano deve ser activado; um conjunto de papéis do plano, $R_p = \{p_{r1}, p_{r2}, \ldots, p_{rm}\}$ $m \leq n$, que define as acções a serem executadas pelos participantes no plano. Cada papel do plano, p_{ri}, inclui o comportamento a ser executado pelo agente e uma condição de término que indica o estado em que o agente deverá cessar a execução do papel do plano e regressar ao seu comportamento normal. Os planos são definidos como parte do *locker-room agreement* de modo a serem conhecidos *a priori* por todos os agentes da equipa.

O *locker-room agreement* foi aplicado com amplo sucesso por Stone et al. ao futebol robótico simulado [Stone et al., 2000a]. Na aplicação prática deste método de coordenação a um domínio tão complexo como o futebol robótico, foi visível que parte dos conceitos teóricos não foi utilizada na prática. Por exemplo, a troca dinâmica de papéis, embora

constitua um método de coordenação potencialmente útil no futebol robótico simulado, não foi aplicada na prática no âmbito da equipa CMUnited[22].

Uma das principais limitações do *locker-room agreement* está relacionada com a definição de um número de papéis idêntico ao número de agentes. Este facto, não permite definir papéis reutilizáveis por diversos membros da equipa. Por outro lado, no trabalho de Stone e Veloso [Stone e Veloso, 1999], a formação é definida como um mapeamento de agentes a papéis, não correspondendo ao conceito tradicional de formação, encontrada nos desportos colectivos ou aplicações militares, isto é, uma distribuição espacial dos agentes. Se por um lado, este facto permite a utilização deste conceito em domínios que não impliquem a distribuição espacial dos agentes, por outro lado a flexibilidade do conceito original é perdida. Torna-se impossível definir formações sem associar a cada posição espacial um comportamento predefinido. Em domínios que impliquem a utilização de um elevado número de agentes, como por exemplo cenários de guerra, a definição de uma formação implica a especificação de centenas ou milhares de papéis, cada um dos quais definindo completamente o comportamento do agente (incluindo o seu posicionamento espacial).

5.5.4 *TAEMS Framework*

O *TAEMS – Task Analysis, Environment Modeling and Simulation,* é um esquema para modelizar ambientes com tarefas computacionais complexas nas quais podem ser utilizadas abordagens baseadas em agentes [Decker, 1996]. O esquema permite representar, de modo independente do domínio, problemas complexos de coordenação de agentes [Prasad e Lesser, 1999]. Permite analisar e simular quantitativamente o comportamento de um Sistema Multi-Agente no que diz respeito a características relevantes do ambiente computacional de que fazem parte. O *TAEMS* é deste modo, não só uma linguagem que permite especificar as características de um SMA mas também um sistema de simulação que permite demonstrar graficamente as tarefas, acções dos agentes e dados estatísticos.

O *TAEMS* suporta uma descrição em camadas das características do ambiente e da tarefa [Decker, 1995] [Decker, 1996], sendo estas camadas as seguintes:

- **Nível Objectivo.** A parte objectiva ou real descreve o essencial da estrutura de tarefas e ambiente. É essencialmente uma descrição formal da tarefa sem informação sobre os agentes em particular;

---

[22] De facto, no trabalho de Stone e Veloso, não foram especificados protocolos que permitam a troca ou atribuição dinâmica de papéis aos agentes. Acresce a este facto que este método não foi utilizado em competições oficiais ou incluído nos binários da equipa CMUnited disponibilizados pelos autores [Stone et al., 1999].

- **Nível Subjectivo.** Descreve a forma como os agentes vêem e interagem com a tarefa a executar ao longo do tempo. Inclui o modo como os agentes se apercebem do estado do mundo.
- **Nível Generativo.** Descreve as características estatísticas necessárias para gerar a informação objectiva e subjectiva num dado domínio. Este nível permite gerar diferentes problemas com as mesmas características evitando instâncias únicas.

O nível subjectivo é essencial para a avaliação do raciocínio dos agentes, pois estes realizam as suas decisões baseados unicamente em informação subjectiva. O desempenho do sistema em geral e dos agentes em particular pode ser aferido, tendo como base informação objectiva. O nível objectivo descreve a estrutura essencial de um dado problema ao longo do tempo. Está centrado na forma como as relações e dependências entre as tarefas afectam dinamicamente a qualidade e duração de cada tarefa.

Um problema de coordenação é representado no *TAEMS*, como um conjunto de grupos de tarefas. Cada grupo de tarefas *GT* representa um conjunto de acções interrelacionadas e possui um tempo de chegada *Ar(T)* e um tempo limite (*deadline*) *D(T)*. A estrutura de um grupo de tarefas é representada como uma árvore em que cada nó corresponde a uma tarefa *T*. A figura 30, representa um exemplo de uma estrutura de tarefas *TAEMS*. A árvore de tarefas é composta por uma tarefa inicial (tarefa raíz) que se decompõe sucessivamente em subtarefas. As tarefas que se encontram nas folhas da árvore são os métodos executáveis *M*. Embora a estrutura de tarefas seja semelhante a uma árvore, devido à possível existência de relações arbitrárias entre as tarefas, esta estrutura é na realidade um grafo.

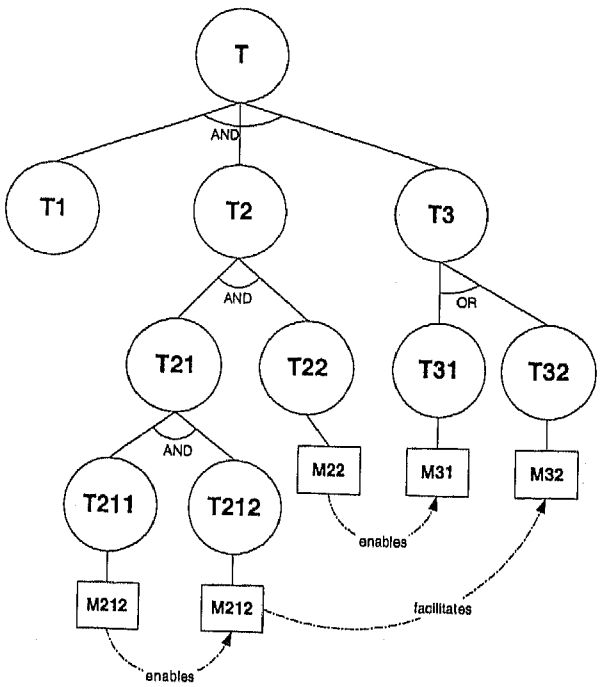

Figura 30: Exemplo de uma Estrutura de Tarefas TAEMS

A utilização do esquema *TAEMS* na modelização de ambientes com tarefas complexas, inicia-se através da construção de um modelo do ambiente. Decker [Decker, 1995] [Decker, 1996] sugere a construção inicial de uma instância do problema utilizando os níveis objectivo e subjectivo, utilizando-a então como base para a construção do nível generativo. O esquema TAEMS providencia diversas funções (*max, min, mean* e *sum*), efeitos predefinidos para tarefas (*enables, precedes, facilitates* e *hinders*) e recursos (*uses* e *limits*). No entanto, podem ser definidas novas funções, efeitos e recursos, estendendo este conjunto. De seguida, as acções dos agentes devem ser especificadas de modo formal. O *TAEMS* disponibiliza uma meta-estrutura para as transições de estado dos agentes, dividida em quatro partes: controlo, recolha de informação, comunicação e execução de métodos. A parte do controlo define o modo como as outras partes (recolha, comunicação execução) são executadas ao longo do tempo. Diversas acções predefinidas de recolha de informação e comunicação são disponibilizadas. Por exemplo, as acções de comunicação predefinidas incluem a transmissão do resultado corrente de um método ou tarefa, partes da estrutura de uma tarefa e comunicações de meta-nível. Relativamente à execução dos métodos, o *TAEMS* suporta execução sequencial simples, execução com monitorização, suspensão, interrupção temporária, etc.

A avaliação do desempenho no âmbito do *TAEMS* é baseada no tempo de execução das tarefas e na qualidade dos resultados dessa execução [Decker, 1995]. A qualidade dos resultados é definida propositadamente de forma vaga, de maneira a que possa ser utilizada na avaliação de qualquer ambiente, utilizando critérios distintos [Decker, 1995]. Consiste num valor escalar que define a completude, utilidade, precisão ou exactidão de um dado resultado, ou outro aspecto desejável para efectuar a avaliação de uma tarefa.

5.5.5 *STEAM – simply a Shell for Teamwork*

O modelo de coordenação, ou trabalho de equipa como os autores [Tambe, 1997] preferem chamar-lhe, *STEAM (simply, a Shell for Teamwork)* é baseado na teoria das intenções conjuntas [Levesque et al., 1990] [Cohen e Levesque, 1991] e na teoria dos planos conjuntos [Grosz, 1996, Grosz e Kraus, 1996]. Utiliza como blocos básicos do trabalho em equipa as intenções conjuntas. Tal como na teoria dos planos partilhados, os agentes possuem uma descrição hierárquica, multi-resolução e complexa, de intenções conjuntas, intenções individuais e crenças acerca das intenções dos outros [Tambe, 1997].

Uma das características relevantes do modelo *STEAM* consiste no vasto e complexo conjunto de capacidades de trabalho de equipa em que se baseia e no tipo de domínios em que foi aplicado com sucesso. Os domínios analisados, ao contrário da grande maioria em que as metodologias anteriores foram aplicadas, possuem como principais características:

- Equipas numerosas, compostas por um vasto conjunto de agentes;
- Domínios abertos (dinâmicos, inacessíveis, não-determinísticos e contínuos);

- Organizações de equipa complexas e hierárquicas;
- Distribuição espacial dos agentes;
- Comunicação restringida e com reduzida largura de banda entre os agentes.

Muito bons resultados foram relatados pelos autores desta metodologia em diversos domínios, de entre os quais se destacam: cenários de batalha, transporte multi-robô e futebol robótico simulado. Relativamente ao domínio do futebol robótico simulado, verifica-se que o modelo obteve resultados dignos de registo unicamente em 1997, numa altura em que os jogos se disputavam ainda sem cooperação visível entre os elementos das equipas. Na competição RoboCup, de 1997 até 1999, verificou-se que todas as equipas, incluindo as equipas vencedoras, desenvolviam as suas acções de ataque através de investidas individuais dos seus agentes sem que se verificassem passes ou outro tipo de movimentos colectivos. Pode-se argumentar que os resultados obtidos pela equipa baseada no modelo *STEAM*[23] tiveram como causa fundamental as capacidades individuais dos seus agentes e não a adequação do modelo de coordenação e trabalho de equipa ao domínio do futebol robótico simulado[24].

5.6 Coordenação sem Comunicação

5.6.1 Coordenação por Modelização Mútua

Uma outra abordagem à coordenação de agentes baseada em técnicas de coordenação de humanos corresponde à coordenação por modelização mútua. Cada agente cria um modelo de cada um dos outros agentes da sua equipa. Este modelo permite que o agente se coloque no lugar de um outro agente e possa prever quais serão as suas acções em determinada circunstância. Do modelo devem constar as crenças, intenções, desejos, capacidades, e qualquer outra informação relevante sobre cada um dos outros agentes a modelar.

A coordenação por modelização mútua foi sugerida inicialmente por Genesereth et al., e é baseada em modelos derivados da teoria dos jogos, no seu trabalho sob o tema "cooperação sem comunicação"[25] [Genesereth et al., 1986]. No seu trabalho, cada agente

---

[23] A equipa *ISIS*, baseada no modelo STEAM, obteve o terceiro lugar em 1997 e o quarto lugar em 1998 na competição RoboCup.

[24] No entanto, os autores argumentam o contrário baseados exactamente nestes mesmos resultados [Marsella et al., 2001] mas ocultando o facto de não existir real cooperação entre os seus agentes no desenvolvimento dos movimentos ofensivos da equipa.

[25] "Cooperation without Communication"

dispunha da informação relativa às preferências dos outros agentes, e como tal podia estimar qual a acção racional que estes deveriam executar em cada instante.

Les Gasser utilizou também um tipo de cooperação semelhante, no âmbito do seu sistema *MACE* [Gasser et al., 1987]. O sistema *MACE* foi um dos primeiros ambientes de teste desenvolvidos para Sistema Multi-Agente. Contém cinco componentes principais:

- Agentes de aplicação;
- Agentes de sistema predefinidos (por exemplo interfaces com o utilizador);
- Serviços disponíveis a todos os agentes;
- Base de dados (com descrições dos agentes); e
- Conjunto de *kernels* que tratam das comunicações físicas em cada máquina a executar o sistema.

No seu estudo, Gasser identificou três aspectos essenciais dos agentes: contêm conhecimento; têm percepção do ambiente; e possuem capacidade para executar acções nesse ambiente. Dividiu também o conhecimento de cada agente em dois tipos [Gasser et al., 1987]: conhecimento local e especializado do domínio; e conhecimento sobre os outros agentes (*acquaintance knowledge*). Este conhecimento dos outros agentes permite a execução de cooperação por modelização mútua e inclui os seguintes tópicos:

- **Classe.** Os agentes *MACE* encontram-se organizados em grupos designados por classes;
- **Nome.** Cada agente possui um nome único na classe. O par (classe, nome) identifica cada um dos agentes;
- **Papéis.** Descrevem as funções dos agentes no interior de cada classe;
- **Capacidades (*Skills*).** Capacidades dos outros agentes, em termos das funções e tarefas que são capazes de executar ou recursos que são capazes de providenciar;
- **Objectivos (*Goals*).** Objectivos dos outros agentes;
- **Planos.** Perspectiva do agente sobre a forma como os outros agentes irão atingir os seus objectivos. É descrita como uma lista de capacidades (*skills*) e operações, que o agente irá executar para conseguir os seus objectivos.

A utilização de modelos dos outros agentes na construção de um SMA foi também utilizada no âmbito do projecto *ARCHON (ARchitecture for Cooperative Heterogeneous ON-line systems)* [Wittig et al., 1994] [Oliveira et al., 1992]. O projecto *ARCHON* foi o maior projecto europeu que se realizou na área da Inteligência Artificial Distribuída. No seu âmbito foi definida uma arquitectura genérica de agentes (figura 31) e uma metodologia adequada à definição de sistemas baseados em IAD para um elevado número de problemas industriais reais. Duas destas aplicações (gestão da distribuição de energia eléctrica e controlo de um acelerador de partículas) foram aplicadas com sucesso nas

organizações reais para que foram desenvolvidas (respectivamente Iberdrola no norte de Espanha e CERN – Centro Europeu para a Investigação em Física de Alta-Energia em Genebra).

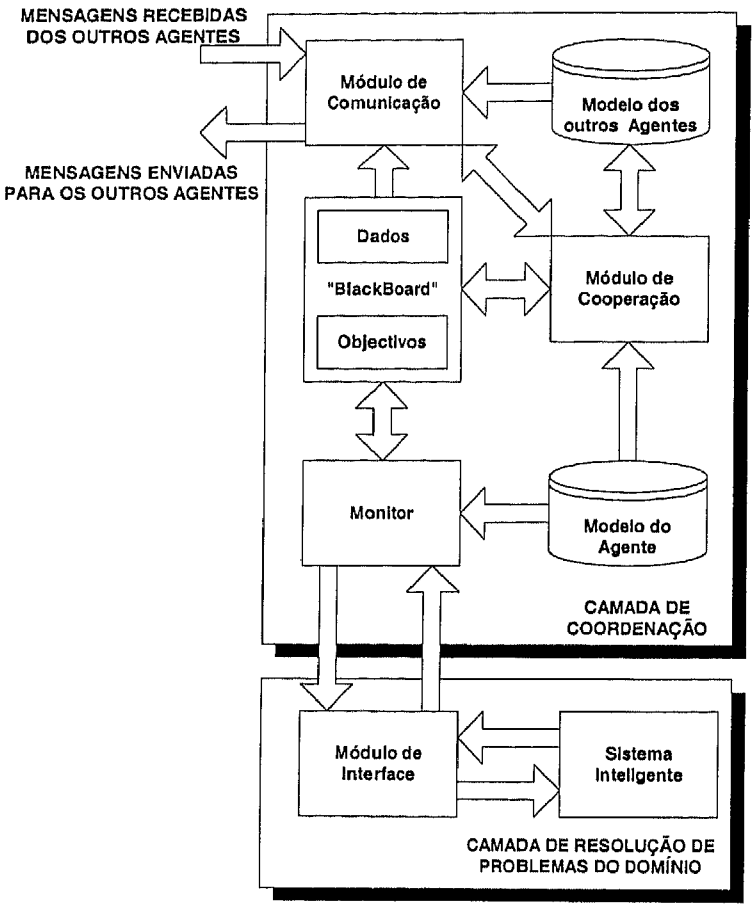

Figura 31: Arquitectura dos Agentes ARCHON

Uma das características mais interessantes do projecto *ARCHON* foi o desenvolvimento de uma arquitectura genérica para agentes semi-autónomos de *software* [Wittig et al., 1994]. Uma comunidade de agentes *ARCHON* não possuía deste modo qualquer tipo de controlo centralizado e cada agente era responsável por decidir a sua própria agenda e gerir as interacções com os outros agentes da comunidade. Para o fazer, a arquitectura possuía os componentes necessários para realizar coordenação por modelização mútua. De facto, cada agente, para além de possuir informação e conhecimento de resolução de problemas num dado domínio, possuía também um modelo próprio e um modelo de todos os outros agentes presentes na comunidade (*acquaintance models*) [Wittig et al., 1994] [Mouta, 1996]. O módulo de planeamento e coordenação dos agentes era então responsável por utilizar esta informação e decidir as acções e comunicações a executar em cada instante.

5.6.2 Coordenação por Percepção Inteligente

Diversas metodologias de utilização inteligente dos sensores de um agente foram propostas por investigadores da área. Por exemplo, os conceitos de visão activa e percepção activa existem já desde a década de 80 [Bajcsy, 1988] [Aloimonos et al., 1998]. Outros autores introduziram metodologias distintas com o objectivo de efectuar uma utilização inteligente dos sensores de um agente, como é o caso das metodologias de fusão sensorial (em que informações provenientes de diferentes tipos de sensores são fundidas de forma inteligente).

A utilização de percepção para a coordenação de equipas de agentes é uma metodologia relativamente recente. Baseia-se numa extensão dos trabalhos de coordenação sem comunicação de Balch e Arkin [Balch e Arkin, 1994]. No seu trabalho, Balch e Arkin analisam situações em que a comunicação pode não ser particularmente útil para a cooperação entre agentes e investigam três tipos de tarefas multi-agente:

- *Foraging*, em que os agentes procuram resgatar objectos de uma arena;
- *Consuming*, onde os agentes encontram objectos e utilizam-nos no local onde foram encontrados; e
- *Grazing*, em que as trajectórias seguidas pelos agentes têm de cobrir completamente a área da arena.

Neste estudo foram usados três níveis de comunicação: nenhuma comunicação; comunicação do estado interno; comunicação dos objectivos. Uma das conclusões obtidas é que o tipo de tarefa cooperativa a realizar afecta consideravelmente o desempenho dos diversos esquemas de comunicação. Nomeadamente na tarefa de *grazing*, a comunicação do estado interno ou dos objectivos não melhora significativamente o desempenho da equipa. Este resultado é atribuído ao facto de existir uma *comunicação implícita*, que é provocada pela mudança do estado do mundo, gerada pelas acções dos agentes. Como neste domínio os agentes deixavam um rasto nos locais onde passavam, existe uma comunicação implícita na geração deste rasto e na sua visualização pelos outros agentes [Balch e Arkin, 1994].

Estendendo o trabalho de Balch e Arkin, é possível incrementar a coordenação através da utilização inteligente dos sensores. Os sensores podem ser utilizados no seguimento das acções realizadas por outros agentes sobre o mundo. A coordenação por percepção inteligente baseia-se exactamente neste princípio. Consiste na utilização de acções de controlo dos sensores do agente de forma a analisar melhor as acções realizadas pelos outros agentes e facilitar a execução de acções cooperativas. Este tipo de coordenação foi sugerido por Reis e Lau [Reis e Lau, 2001a] e aplicado no contexto do controle inteligente dos sensores de um agente do futebol robótico simulado. O objectivo consiste em utilizar as acções de percepção do agente de forma a aumentar a coordenação da equipa, maximizando as hipóteses de cooperação com os restantes elementos da equipa. No seu

trabalho, Reis [Reis, 2001] [Reis, 2001c] [Reis, 2002f] concluiu que a utilização deste mecanismo em conjunto com a coordenação por modelização mútua e com alguns mecanismos seleccionados de coordenação por organização estrutural, permite coordenar uma equipa de agentes sem utilizar qualquer mecanismo de comunicação. Reis propõe ainda uma arquitectura de agente capaz de suportar a coordenação por percepção inteligente e uma implementação deste mecanismo adequada ao controlo de um pescoço flexível no futebol robótico simulado, designada por *SLM – Strategic Looking Mechanism* [Reis e Lau, 2001a] [Reis, 2002f]. No capítulo nove, este mecanismo de coordenação e a sua aplicação ao futebol robótico será descrito com maior detalhe e resultados experimentais, do seu impacto no desempenho de uma equipa, serão analisados.

5.7 Coordenação por Organização Estrutural

No contexto de Sistemas Multi-Agente uma estrutura organizacional pode ser vista como a definição do fluxo de informação e das relações de controlo entre os diversos agentes. Estas relações de controlo, independentemente do seu tipo, são responsáveis por definir a autoridade relativa dos agentes e formatar os tipos de interacção social que possam ocorrer [Jennings, 1996]. Por exemplo, na modelização de uma determinada empresa de distribuição utilizando um Sistema Multi-Agente, o projectista pode optar por atribuir a distribuição de diferentes produtos a diferentes agentes ou pode optar por atribuir diferentes regiões espaciais de distribuição a diferentes agentes. O tipo de interacções entre os agentes resultantes das duas opções é completamente distinto. As relações especificadas pelas estruturas organizacionais fornecem informação de longo-prazo sobre a comunidade de agentes e a sua forma de operação como um sistema. Assim, define-se uma coordenação por organização estrutural especificando que, acções são executadas pelos diferentes agentes e que tipo de comunicação existe entre eles.

Diversos autores utilizam uma terminologia distinta, nomeadamente através da utilização do termo *estratégias* para designar a especificação do comportamento global de um agente ou grupo de agentes [Reis et al., 2001] [Singh, 1990] e do termo *papéis* para designar o comportamento de um agente individual [Werner, 1989] [Stone e Veloso, 1999] [Reis e Lau, 2001a] [Reis et al., 2000]. Por vezes, estes conceitos encontram-se interrelacionados como no trabalho de Reis [Reis, 2002f] que define *estratégias* compostas por *tácticas* que por sua vez são compostas por *formações* que atribuem a cada agente um *posicionamento* e um *papel*.

Shoham e Tennenholtz [Shoham e Tennenholtz, 1992] utilizam uma forma de organização distinta que designam por *lei social*. Na sua abordagem, a sociedade adopta um conjunto de leis (por exemplo, leis de trânsito) que especificam como é que os diferentes agentes se devem comportar. Os agentes são projectados de modo a obedecerem a essas leis sociais.

O processo de projecto dos agentes é simplificado uma vez que é assumido que todos irão respeitar essas leis sociais.

Embora as organizações estruturais sejam assumidas como estruturas de longo-prazo, por vezes é necessário alterá-las frequentemente. Malone [Malone, 1987] demonstrou que diferentes organizações estruturais são adequadas a diferentes situações, o que torna importante o conceito de situação e a alteração da organização estrutural em face da situação. A situação tem de envolver toda a comunidade e, como tal, tem de ser reavaliada frequentemente, bem como, tem de ser acreditada por todos os elementos da comunidade. De acordo com a situação, a comunidade pode rearranjar a sua estrutura de forma a melhor se adaptar à execução da sua tarefa cooperativa na nova situação. Trabalhos relacionados com a reorganização estrutural dinâmica de sociedades ou equipas de agentes incluem face de novas situações incluem [Ishida et al., 1990] e [Reis et al., 2001].

5.7.1 Coordenação por Normas e Leis Sociais

A coordenação por normas e leis sociais inspira-se em modelos humanos e deriva da disciplina da sociologia. Uma *norma* corresponde a um padrão de comportamento preestabelecido. Uma *lei social* tem um significado semelhante mas inclui em si própria uma noção de autoridade. Na sociedade, os exemplos de normas e leis sociais são abundantes. Por exemplo, uma *norma* consiste na regra assumida por todos de que: "numa paragem de autocarro, quem chega primeiro será o primeiro a entrar no autocarro" [Wooldridge, 2002]. Não existe qualquer autoridade que verifique esta norma pelo que são os próprios agentes a regularem os seus comportamentos de modo a obedecerem à norma. Uma *lei social* pode, por exemplo, corresponder à necessidade de pagar a conta no final de um jantar no restaurante. Existe uma autoridade, no caso a polícia, que tem o poder de verificar se a lei é obedecida e aplicar sanções ao agente em caso contrário. Neste mesmo exemplo, uma *norma* poderá consistir em dar uma gorjeta ao empregado (habitual em vários países, incluindo Portugal).

As *normas* e *leis sociais* providenciam aos agentes um modo de coordenarem as suas acções, através da estruturação dessas mesmas acções, fornecida pelo modelo da norma ou lei. Representam restrições de comportamento que podem limitar a liberdade dos agentes na escolha das suas acções. Implicam um balanceamento entre a execução de acções, de modo a atingir os seus objectivos, e execução de acções de modo a respeitar as leis sociais. Como todas as restrições, as leis sociais permitem ao agente limitar o número de cursos de acção possível, simplificando o seu processo de decisão.

As normas e leis sociais podem existir dentro de um Sistema Multi-Agente de duas formas:

- **Projectadas *off-line*.** Nesta abordagem as leis sociais são definidas à-priori e implantadas no interior dos agentes;

- **Emergentes *on-line*.** Nesta abordagem as leis sociais emergem do funcionamento do próprio sistema e são adquiridas no decurso da sua operação pelos agentes;

Evidentemente que a primeira abordagem é mais fácil de implementar e permite ao projectista do sistema possuir um maior controlo sobre o seu modo de operação. A aquisição de leis sociais *on-line* possui a grande vantagem de permitir adquirir leis sociais desconhecidas ou inexistentes à partida, sendo deste modo adequada para a utilização em ambientes abertos tais como a Internet ou ambientes derivados do mundo-real.

5.7.2 Coordenação por Conhecimento à-Priori

A coordenação por conhecimento à-priori engloba grande parte da investigação realizada no âmbito da coordenação por normas e leis sociais projectadas *off-line*. O conceito de coordenação por conhecimento à-priori é mais abrangente pois este conhecimento não se limita a normas e leis sociais, podendo incluir estruturas mais complexas como sejam tácticas, formações espaciais, modelos de outros agentes (colegas e/ou agentes hostis), planos flexíveis, etc.

Neste contexto, Stone e Veloso [Stone, 1998] [Stone e Veloso, 1999] propuseram a utilização de um esquema de coordenação designado por *acordo de balneário* (*locker-room agreement*) descrito anteriormente na secção 5.5.3.

Reis et al. [Reis et al., 2001] [Reis e Lau, 2001] propõem a utilização de conhecimento à-priori para definir a estratégia de uma equipa de futebol (robótico) e baseiam a coordenação da equipa FC Portugal de futebol robótico neste princípio. No seu trabalho, Reis define que o conhecimento necessário para uma equipa disputar uma partida de futebol inclui [Reis, 2002f]:

- **Regiões.** Tendo em conta que o futebol implica a movimentação espacial dos agentes é importante que estes possuam uma definição comum das diversas regiões do campo;
- **Situações.** Contendo a definição lógica de situações que possa implicar uma mudança de comportamento dos agentes;
- **Formações.** Contendo distribuições espaciais e formas de movimentação de todos os agentes que compõe a equipa;
- **Modelos dos Colegas.** Modelos dos colegas de equipa de forma a permitir a realização de coordenação por modelização mútua;
- **Modelos dos Adversários.** Modelos dos jogadores adversários de forma a permitir a realização de coordenação por modelização mútua;
- **Modelo da Equipa Adversária.** Este modelo permite uma definição do comportamento esperado da equipa oponente que é utilizada na definição da

estratégia da equipa e no decurso do jogo na alteração da táctica em utilização;

- **Planos Flexíveis.** Utilizados em situações de jogo e que permitem a execução de uma determinada tarefa colectiva;
- **Estratégias de Comunicação.** Utilizadas para permitir diferentes tipos de comunicação entre os agentes que compõem a equipa;
- **Papéis.** Contendo comportamentos individuais de cada agente. Cada agente possui uma biblioteca de papéis e em cada instante desempenha um único papel;
- **Tácticas.** Contendo a definição do comportamento global da equipa e a utilização das formações, papéis, planos e estratégias de comunicação no seu âmbito. Cada táctica possui determinadas condições de activação e implica a utilização de formações distintas para situações distintas.

Reis e Lau [Reis e Lau, 2001] aplicaram com sucesso este tipo de coordenação no futebol robótico no âmbito da equipa FC Portugal e estenderam o conceito para aquilo que designaram de coordenação estratégica. Uma descrição mais detalhada deste trabalho pode ser encontrada no capítulo 9.

Outros trabalhos, relacionados com a coordenação por conhecimento *a priori*, incluem a definição de linguagens que permitam descrever informação e conhecimento que serão partilhados inicialmente por um conjunto de agentes e que serão úteis ao longo da execução da tarefa cooperativa para coordenar esses mesmos agentes. No âmbito do futebol robótico, Reis e Lau definiram uma linguagem COACH UNILANG que permite a definição de uma estratégia para uma equipa que vai disputar um jogo de futebol [Reis e Lau, 2002]. Esta linguagem para além de permitir definir tácticas de futebol, permite também a coordenação por controlo parcialmente hierárquico, como será analisado na secção seguinte e no capítulo nove.

No domínio do escalonamento na gestão universitária, Reis e Oliveira [Reis e Oliveira, 2001] desenvolveram uma linguagem, SCHEDULING UNILANG que permite a definição de problemas de escalonamento, incluindo não só os seus dados mas também restrições, funções de avaliação e soluções. Esta linguagem foi aplicada com sucesso no âmbito do sistema *UNIPS – University Planning and Scheduling System* na definição de problemas de horários e problemas associados no âmbito de um Sistema Multi-Agente [Reis, 1999].

5.8 Coordenação por Controlo Parcialmente Hierárquico

As metodologias de coordenação definidas anteriormente pressupõem que os agentes envolvidos são completamente autónomos, no sentido da inexistência de uma hierarquia

entre eles. No entanto, em grande parte dos sistemas sociais humanos tal não se verifica. Estes sistemas são normalmente compostos por agentes a diferentes níveis de hierarquia, que possuem uma dose considerável de autonomia. Suponha-se um exemplo de uma sociedade humana em que um empregado de uma determinada organização recebe uma ordem do seu patrão para escrever um determinado relatório. Provavelmente, a sua reacção será aceitar a ordem e escrever tal relatório desde que possua capacidade, tempo e recursos suficientes para o fazer. Suponha-se um outro exemplo em que o mesmo empregado recebe uma ordem do patrão para incendiar o edifício onde se encontram. Provavelmente, embora possua capacidade, tempo e recursos suficientes para o fazer, o empregado não o realizará. Este facto deve-se a que este agente possui autonomia suficiente para tomar as suas próprias decisões e a ordem recebida provavelmente não conduzirá ao cumprimento dos seus objectivos individuais na empresa (salário elevado, promoções, etc.) nem dos objectivos globais da empresa (crescimento da companhia, lucro, etc.).

Para não violar a noção de autonomia, as ordens recebidas de agentes hierarquicamente superiores não podem ser tomadas como ordens absolutas por parte de agentes autónomos. Sobretudo, se estas ordens ou instruções estão em conflito com as intenções do agente, coloca-se a questão de como pode ser integrada a informação enviada pelo agente hierarquicamente superior, de modo consistente, no conjunto de crenças e intenções do agente.

A ideia da coordenação parcialmente hierárquica não é em si inovadora. McCarthy sugeriu há mais de 40 anos a ideia de integração de instruções provenientes de uma fonte externa [McCarthy, 1958]. No entanto, poucas metodologias foram desenvolvidas para permitir este tipo de coordenação entre agentes com posições hierarquicamente distintas. De entre os trabalhos relevantes na área destacam-se [Grosof, 1995] e [Carpenter et al., 2003]. Grosof analisou o processo de gestão de instruções conflituosas e definiu quatro critérios para determinar a forma de resolução do conflito [Grosof, 1995]:

- **Especificidade.** As instruções mais restritivas são seguidas com maior prioridade;
- **Novidade.** As instruções mais recentes são seguidas com maior prioridade;
- **Autoridade.** As instruções de agentes hierarquicamente superiores são seguidas com maior prioridade;
- **Fiabilidade.** As instruções que conduzem a uma probabilidade de sucesso são seguidas com maior prioridade.

Carpenter et al. [Carpenter et al., 2003] estenderam o trabalho de Grosof e propuseram uma metodologia para integrar instruções de um treinador numa arquitectura de selecção de acções e aplicam esta metodologia experimentalmente no contexto do seu treinador (*OWL*) e da sua equipa (*ChaMeleons2001*) de futebol robótico simulado. Esta equipa

utiliza uma arquitectura hierárquica de comportamentos que contém três componentes principais na selecção de acções: comportamentos individuais de baixo-nível; gestor de comportamentos; e árbitro de comportamentos. No seu trabalho, utilizam um conjunto de comportamentos organizados hierarquicamente, em que cada comportamento possui um conjunto de condições de aplicabilidade e uma probabilidade de sucesso. O gestor de comportamentos, sempre que recebe instruções de uma fonte externa, integra essas instruções no local apropriado da árvore de comportamentos. O árbitro de comportamentos é uma função que selecciona em cada instante um único comportamento, de entre os possíveis, para execução.

Embora os resultados experimentais publicados em [Carpenter et al., 2003] tenham sido impossíveis de comprovar experimentalmente no nosso laboratório[26], este trabalho não deixa por isso de ser relevante na definição de metodologias de controlo parcialmente hierárquico, nomeadamente na perspectiva de como um agente deve lidar com instruções pouco fiáveis provenientes de diversas fontes.

A um nível de abstracção mais elevado, Reis definiu um modo de coordenação por controlo parcialmente hierárquico, no âmbito da utilização do treinador *on-line* no futebol robótico [Reis e Lau, 2001] [Reis e Lau, 2002]. No âmbito do seu trabalho definiu uma linguagem que permite a esse treinador de futebol robótico simulado coordenar uma equipa de agentes autónomos, através de um controlo parcialmente hierárquico realizado através do envio de instruções aos jogadores [Reis e Lau, 2001] [Reis e Lau, 2002]. As instruções referem-se a tácticas, formações e outras definições de alto-nível do comportamento individual de um jogador ou colectivo da equipa.

5.9 Coordenação Estratégica

A coordenação estratégica é baseada na coordenação por conhecimento *a priori*, podendo ser considerada uma instanciação desta. Baseia-se na definição de uma estratégia que é composta por um conjunto de tácticas, que implicam diferentes formas de actuação da equipa como um todo. Cada táctica utiliza um conjunto de formações. Dependendo da situação global, a formação mais adequada é seleccionada em cada instante. Cada formação atribui a cada agente um papel e, no caso de domínios espaciais, um esquema de posicionamento dinâmico.

Na aplicação da coordenação estratégica torna-se essencial:

- A criação de uma biblioteca de tácticas e, no caso de domínios espaciais,

[26] Ao contrário do que os autores afirmam em [Carpenter et al., 2003], as experiências realizadas no nosso laboratório provaram que o treinador *OWL* piora o comportamento de todas as equipas referidas em [Carpenter at al., 2003].

formações adequadas ao domínio;

- O conceito de situação e a definição de situações facilmente identificáveis pelos agentes que permitam efectuar a troca dinâmica de tácticas e formações;
- O conceito de papel como forma de definição do comportamento individual de um jogador como parte da equipa;
- A definição de papéis adequados à tarefa cooperativa a realizar, e de mecanismos que permitam aos agentes uma troca dinâmica de papéis.
- A definição de mecanismos que permitam a degradação suave de tácticas e formações, no caso de as perspectivas distintas dos agentes os levarem a identificações distintas da situação;
- A definição de mecanismos de posicionamento flexíveis, dinâmicos e adequados ao domínio;

Podem ser enumerados três tipos de coordenação essenciais à aplicação da coordenação estratégica a domínios dinâmicos:

- Coordenação Situacional (Temporal);
- Coordenação Espacial por Posicionamento Estratégico;
- Coordenação por Definição e Troca de Papéis.

Claro que a coordenação espacial por posicionamento estratégico só faz sentido em domínios em que os agentes possuem uma distribuição espacial ou geográfica tais como o futebol robótico (simulado ou não), o resgate e salvamento, cenários de guerra, limpeza de minas, etc. Uma análise mais detalhada da coordenação estratégica, seus domínios de aplicação e sua aplicação prática ao futebol robótico simulado, encontra-se no capítulo nove.

5.9.1 Coordenação Situacional (Temporal)

A coordenação situacional consiste na definição de situações que podem facilmente ser identificadas através do estado do mundo incompleto de agentes que operam em ambientes inacessíveis, dinâmicos e com comunicação limitada e insegura. Estas situações podem posteriormente ser utilizadas como eventos de coordenação, nomeadamente, no âmbito de um esquema de coordenação estratégica para realizar a transição entre tácticas e formações. Exemplos de situações no domínio do futebol são o ataque, defesa, marcação de um livre ou pontapé de baliza.

Reis [Reis, 2002f] propõe uma metodologia para definir situações no âmbito do futebol robótico simulado e uma forma simples de degradação da situação de forma a evitar transições bruscas nas tácticas e formações utilizadas pelos agentes constituintes de uma dada equipa.

5.9.2 Coordenação Espacial por Posicionamento Estratégico

Uma das principais carências da maioria dos tipos e modelos de coordenação analisados, consiste no facto de não tratarem convenientemente a coordenação espacial de agentes, que possuem capacidades de movimentação no espaço. A movimentação de agentes surge em muitos domínios, como sejam, a maioria dos domínios que incluem robôs móveis, domínios de guerra (ou simulação de batalhas), futebol robótico (real e simulado), resgate e salvamento, etc.

A coordenação espacial por posicionamento consiste em definir um esquema flexível de movimentação de uma equipa de agentes no espaço que lhes permita executar a sua tarefa cooperativa com facilidade. Este esquema implica, no entanto, a necessidade de balancear as acções dos agentes destinadas a manter o seu posicionamento estratégico com as acções destinadas a executar as outras tarefas cooperativas.

Reis et al. [Reis et al., 2001] propõem a distinção entre situações activas e estratégicas como forma de permitir a uma equipa utilizar uma metodologia de coordenação estratégica, sem que tal signifique a perda de capacidade de decisão individual ou perda de reactividade. Com base nesta distinção, é usado um esquema de posicionamento designado por *Posicionamento Estratégico Baseado em Situações* (*SBSP – Situation Based Strategic Positioning*) como forma de coordenar espacialmente uma equipa de futebol robótico simulado. O esquema *SBSP* tornou-se o esquema de posicionamento mais utilizado na competição RoboCup nos dois anos seguintes à sua proposta[27].

5.9.3 Coordenação por Definição e Troca de Papéis

A coordenação por definição e troca de papéis é uma extensão do conceito de coordenação por decomposição e alocação de tarefas. Baseia-se igualmente no princípio de efectuar a decisão de quem faz o quê. No entanto, é realizada a um nível bastante mais elevado. Em vez de alocar uma ou mais tarefas a cada agente, aloca papéis aos agentes. Esses papéis definem completamente o comportamento individual de cada agente. Para além da definição de papéis para cada um dos agentes que compõem a equipa, é útil realizar a troca de papéis entre esses agentes em determinadas circunstâncias. Nomeadamente, a troca de papéis entre dois agentes A_i e A_j deverá claramente ser realizada se num determinado instante:

- Um determinado agente A_i tem atribuído o papel P_i e um agente A_j tem atribuído um papel P_j;

---

[27] Em 2001, 6 dos 8 finalistas da liga de simulação do RoboCup utilizaram o *SBSP* ou um esquema de posicionamento directamente baseado neste.

- O agente A_i é capaz de desempenhar o papel P_j melhor do que o agente A_j;
- O agente A_j é capaz de desempenhar o papel P_i melhor do que o agente A_i.

Este esquema de troca de papéis é, no entanto, muito rígido e diversas metodologias foram propostas por diferentes autores para permitir uma troca mais flexível e dinâmica de papéis dentro de uma dada equipa de agentes. Stone e Veloso. [Stone e Veloso, 1999] sugeriram efectuar a troca de papéis sempre que tal fosse útil para a equipa, embora não tenham especificado o modo de definir este conceito de utilidade.

Reis et al. [Reis e Lau, 2001] [Reis et al., 2001] estenderam o trabalho de Stone e Veloso e propuseram a realização de trocas de papéis e posicionamentos para agentes heterogéneos, baseadas em utilidades no contexto do futebol robótico simulado. Na sua metodologia, a troca de papéis de dois agentes A_i e A_j é realizada baseada em:

- Distâncias relativas das posições $Pos(A_i)$ e $Pos(A_j)$ dos agentes A_i e A_j às posições estratégicas $PosEst(P_i)$ e $PosEst(P_j)$ dos papéis P_i e P_j;
- Adequação dos agentes A_i e A_j aos papéis P_i e P_j, ou seja: $Adeq(A_i, P_i)$, $Adeq(A_i, P_j)$, $Adeq(A_j, P_i)$ e $Adeq(A_j, P_j)$;
- Importâncias $Imp(P_i)$ e $Imp(P_j)$ dos papéis P_i e P_j para a equipa.

O seu mecanismo de troca dinâmica de posições, designado por *DPRE – Dynamic Positioning and Role Exchange,* ao contrário do mecanismo de Stone e Veloso, define completamente o conceito de utilidade da troca de posicionamento e papel de dois agentes e foi utilizado com sucesso em competições RoboCup. Os resultados experimentais obtidos em competições oficiais e no laboratório, confirmaram que o *DPRE* melhora significativamente o comportamento global de uma equipa de agentes para o futebol robótico simulado. No capítulo nove, os princípios básicos do *DPRE*, a sua implementação no futebol robótico simulado e resultados, demonstrando a sua influência no desempenho de uma equipa, serão analisados com maior detalhe.

5.10 Conclusões

O campo dos Sistemas Multi-Agente enfrenta nos nossos dias um elevado número de desafios, destacando-se o de criar metodologias que permitam a equipas de agentes operarem harmoniosamente em conjunto (que permitam coordenar os diversos agentes de modo eficiente).

Um número crescente de investigadores na área dos agentes procura construir equipas de agentes para domínios complexos e dinâmicos tais como a educação, integração de informação, gestão fabril, cenários de guerra, futebol simulado ou robótica cooperativa. Infelizmente, a incerteza nestes domínios dinâmicos e complexos, impede um trabalho de

equipa coerente, em particular devido ao facto de os diferentes membros da equipa possuírem visões parciais e frequentemente inconsistentes do seu ambiente.

Mecanismos de coordenação e comunicação flexíveis e de alto-nível, são a chave para fazer face ao dinamismo, incerteza e complexidade dos domínios. É necessário avaliar o desempenho da equipa e ter capacidade para reorganizar a equipa dinamicamente de acordo com as contingências. No entanto, a maioria dos Sistemas Multi-Agente implementados não são capazes de providenciar esta flexibilidade nos mecanismos de coordenação. Por outro lado, a maioria das metodologias de coordenação de agentes cooperativos propostas na literatura, não estão preocupadas com a mobilidade espacial dos agentes e, como tal, não são aplicáveis a tarefas cooperativas realizadas por equipas de agentes robóticos em ambientes do mundo-real ou simulações destes.

Neste capítulo foram analisadas as principais metodologias de coordenação de agentes cooperativos propostas pelos investigadores da área, foram sugeridas extensões de algumas dessas metodologias e propostas novas metodologias de coordenação de alto-nível de equipas que permitam fazer face ambientes complexos dinâmicos e inacessíveis. A coordenação estratégica proposta, permite coordenar uma equipa de forma dinâmica e flexível e inclui metodologias de análise situacional, posicionamento espacial dinâmico através de formações e alocação e troca dinâmica de papéis. A definição formal da coordenação estratégica e das restantes metodologias de coordenação propostas e a sua aplicação a domínios multi-agente que impliquem a realização de tarefas complexas através de equipas de agentes, serão analisadas em maior detalhe nos capítulos seguinte deste trabalho.

Capítulo 6

6. Problemas de Escalonamento na Gestão Universitária

Embora todas as Instituições de Ensino Superior enfrentem os mesmos tipos de problemas que têm de ser resolvidos no decurso do seu funcionamento, a definição exacta destes problemas está intimamente relacionada com a organização interna da Instituição e com o seu modo de funcionamento. De entre os problemas que as instituições de ensino enfrentam, destacam-se os problemas de escalonamento, que consistem em efectuar a determinação da melhor distribuição temporal de recursos a tarefas, com o objectivo de garantir a execução correcta dessas tarefas mas minimizando os custos e a utilização de recursos na sua execução.

Os principais problemas de escalonamento no contexto do funcionamento usual de uma Universidade ou outra instituição de ensino superior incluem a *Distribuição de Serviço Docente*, as diversas variantes da *Geração de Horários*, a *Calendarização de Exames* e a *Alocação de Salas*. Da diversidade de formas em que uma Instituição de Ensino Superior se pode organizar decorrem problemas e processos de resolução distintos de *Distribuição de Serviço Docente*, *Geração de Horários*, *Calendarização de Exames* e *Alocação de Salas*.

No início deste capítulo caracterizam-se as Instituições Universitárias do ponto de vista organizacional e apresentam-se os principais problemas de escalonamento que se colocam na sua gestão. De seguida, analisam-se resumidamente os problemas de escalonamento da gestão universitária mais relevantes, nomeadamente a *Distribuição de Serviço Docente*, *Geração de Horários* do tipo Escolar e Universitário e *Calendarização de Exames*. Para cada problema é apresentada a sua descrição, solução desejada, restrições associadas à obtenção dessa solução e principais variantes.

As diversas técnicas, propostas pelos investigadores da área, para resolver os problemas de escalonamento referidos, são descritas resumidamente. Entre estas técnicas destacam-se as meta-heurísticas, nomeadamente o arrefecimento simulado, pesquisa tabu e algoritmos genéticos. Outras metodologias, aplicadas com sucesso na resolução deste tipo de problemas, são a programação em lógica e a programação em lógica com restrições.

Discute-se também resumidamente a aplicabilidade de metodologias baseadas em agentes na resolução deste tipo de problemas.

De modo a concluir a necessidade de criação de um sistema integrado de resolução de problemas de gestão universitária, são discutidas as soluções comerciais disponíveis actualmente no mercado para este tipo de problemas.

O capítulo conclui-se com uma análise das metodologias propostas pelos investigadores da área para representarem problemas de geração de horários e outros problemas relacionados e com uma análise detalhada dos desafios colocados pelos problemas de escalonamento da gestão universitária aos investigadores da área dos Agentes e Sistemas Multi-Agente.

6.1 Organização e Gestão Universitária

Os modelos de organização e gestão das Universidades são fortemente influenciados pelas disposições normativas emanadas pelos Estados [Durão, 1998]. De facto, uma das formas de gestão e controlo dos Governos sobre os sistemas de ensino superior consiste em regulamentar e tornar congruentes os modelos de organização e gestão das instituições que os compõem.

A organização interna e forma de gestão da Universidade são fortemente influenciadas pelo facto de esta ser uma organização caracterizada pela burocracia profissional [Dias, 1999]. Este formato é usual em organizações que têm um conjunto estável e definido de tarefas a realizar, as quais são bastante complexas, exigindo, por isso, um controlo directo por parte de quem as desempenha. Assim, a estrutura deste tipo de organizações tem por base a descentralização da autoridade e a normalização das capacidades como método de organização das actividades [Mintzberg, 1979].

É comum que uma Universidade assuma uma estrutura baseada numa de duas estruturas organizativas: a estrutura Integrada ou a estrutura Federativa. No entanto, é típica a existência de Universidades com estruturas mistas.

A Universidade do tipo integrado é caracterizada pela gestão centralizada e pela estrutura de suporte ao seu funcionamento, muito semelhante à estrutura de suporte das organizações empresariais. Deste modo, não existe qualquer razão para que as estruturas de suporte universitárias não sejam geridas de igual forma às suas congéneres empresariais, sendo a adequação dos seus serviços assegurada pela coordenação de pessoal académico nomeado para essa função.

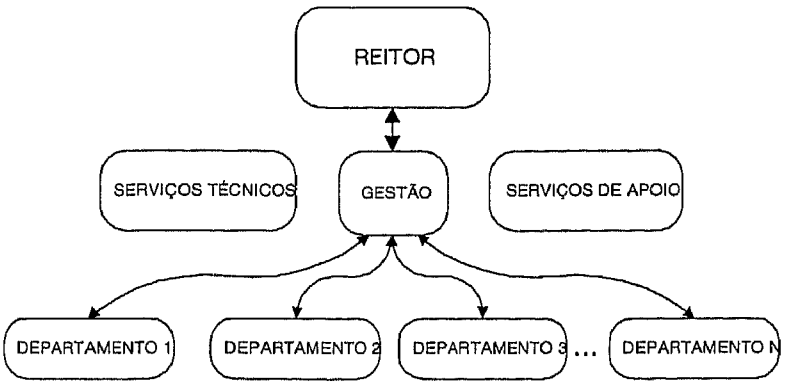

Figura 32: Estrutura da Universidade Integrada

Esta estrutura, representada esquematicamente na figura 32, apesar de compartimentada em Departamentos ou Secções Autónomas, não dá a estes o poder de se auto gerirem completamente. De resto, é comum que os Departamentos não tenham alocados a si quaisquer recursos tais como salas ou docentes com vista a facilitar a sua gestão centralizada.

A Universidade do tipo federativo é caracterizada, em geral, por uma estrutura departamental de base, organizada em Faculdades, Institutos ou Escolas com uma grande autonomia e que desenvolvem a sua própria estrutura de suporte. A figura 33 apresenta um esquema da estrutura Federativa.

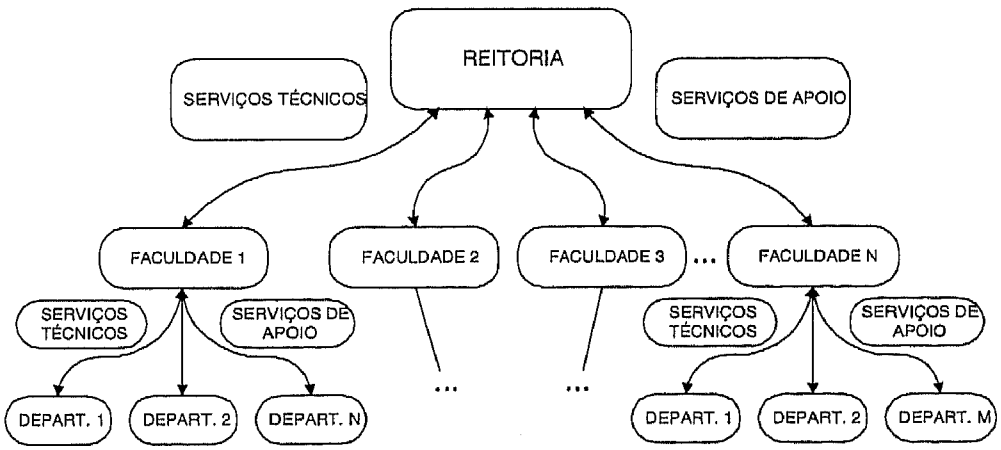

Figura 33: Estrutura da Universidade Federativa

Como as características das Faculdades são usualmente distintas entre si, estas têm a liberdade de se organizarem da maneira mais adequada a cada realidade específica. Assim, o número de Departamentos e o modo como estes são geridos são bastante variáveis. A dificuldade de modelização da estrutura federativa está relacionada precisamente com o seu carácter diverso.

O grau de descentralização orgânica das Universidades (estrutura Integrada ou Federativa) depende de um conjunto complexo de condições ligado à dimensão da organização, ao seu

contexto histórico e ao âmbito científico de actuação. No entanto, uma característica geralmente comum a todas as Universidades é a compartimentação do conhecimento científico em Departamentos, de acordo com as Áreas Científicas. Assim, pode-se entender que a unidade básica da Universidade é, normalmente, o Departamento.

O Departamento é uma unidade formada por um conjunto de docentes com interesses e especializações numa mesma Área Científica. A responsabilidade e função essencial dos Departamentos são a organização e desenvolvimento das competências científicas do seu corpo docente, o qual representa, o recurso mais importante da Universidade. Ao Departamento cabe também a gestão e organização das células de actividade básicas, as quais são o curso e a disciplina na actividade de ensino e o projecto na actividade de investigação.

Em certas situações, o maior número e aumento da dimensão dos Departamentos conduziu à criação de estruturas intermédias autónomas que conseguissem articular as actividades desenvolvidas em Áreas Científicas afins. Estas estruturas são vulgarmente designadas por secções ou áreas disciplinares.

O grau de autonomia de um Departamento depende da sua responsabilidade em gerir cursos e/ou disciplinas, a quantidade de recursos que lhe pertence e da sua própria dimensão. Um Departamento é, normalmente, responsável por um determinado número de cursos da estrutura onde está inserido (Universidade, Faculdade ou Instituto), e pela maioria das disciplinas que compõe esses cursos, bem como, pelos docentes que as leccionam. No entanto, em Portugal é usual que determinadas licenciaturas ou mestrados sejam leccionados por diversos departamentos ou mesmo faculdades, sem que exista um departamento ou faculdade que possa ser apontado como o seu responsável.

6.2 Problemas de Optimização Combinatória

Os problemas de escalonamento da gestão universitária são muitas vezes modelizados como problemas de optimização combinatória e resolvidos de forma automática utilizando metodologias típicas de resolução destes problemas.

Grande parte dos problemas de optimização está preocupada com a escolha da melhor configuração para um conjunto de parâmetros de forma a atingir um determinado conjunto de objectivos. Dividem-se naturalmente em duas categorias: problemas com variáveis contínuas e problemas com variáveis discretas. Dentro deste último tipo, encontramos os problemas que se incluem na classe vulgarmente designada por problemas de optimização combinatória. Um problema de optimização combinatória $POC = (S, f)$ pode então ser definido por:

- Um conjunto de *Variáveis:* $X = \{x_1, x_2, ..., x_n\}$;
- Um conjunto de *Domínios* para essas *Variáveis*: $D_1, D_2, ...D_n$;

- Um conjunto de *Restrições* entre essas *Variáveis*: $R = \{R_1, R_2, ..., R_{nrest}\}$;
- Uma *Função Objectivo* a ser minimizada (ou maximizada) f: $D_1xD_2x...xD_n \rightarrow R$;

O conjunto S de todas as soluções admissíveis do problema será:

$S = \{s=\{(x_1,v_1), (x_2,v_2),..., (x_n,v_n)\} | v_i \in D_i, i=1, ..., n \wedge s \text{ satisfaz todas as restrições}\}$

S é usualmente designado por *Espaço de Pesquisa* (ou de soluções) pois cada elemento deste conjunto pode ser visto como uma possível solução do problema. O modo de resolução de um problema de optimização combinatória, consiste em encontrar uma solução $s^* \in S$ com um valor mínimo da função de optimização f, ou seja $f(s^*) \leq f(s), \forall s \in S$. s^* é designada como a solução óptima para o problema P.

Exemplos de problemas de optimização combinatória incluem o problema do caixeiro-viajante, os problemas de satisfação de restrições em geral e os problemas de escalonamento. Devido à enorme importância prática dos problemas de optimização combinatória, diversos algoritmos para a sua solução foram propostos. Esses algoritmos podem ser classificados como algoritmos completos ou aproximados. Os algoritmos completos garantem atingir a solução óptima para uma instância finita de um problema de optimização combinatória. No entanto, a maioria dos problemas de optimização combinatória encontrados na prática, são NP-Completos [Even et al., 1976] [Garey e Johnson, 1979] e como tal, métodos completos necessitam de um tempo exponencial, no pior caso, para encontrar a solução óptima. Mesmo para instâncias de problemas reais, relativamente pequenas, estes algoritmos podem necessitar de um tempo demasiado elevado para serem utilizados na prática. Desta forma, a quantidade de investigação científica dedicada aos métodos aproximados, ao longo dos últimos anos tem crescido consideravelmente. Neste tipo de métodos, a obtenção de uma solução óptima é sacrificada de forma a ser garantida a obtenção de uma solução boa, num intervalo de tempo razoável.

De entre os métodos aproximados é usual distinguir entre os métodos construtivos e os métodos reparativos. Os métodos construtivos baseiam-se em algoritmos que geram as soluções de raiz, adicionando a uma solução inicialmente vazia, componentes até a solução estar completa. Os métodos reparativos partem de uma solução inicial completa, eventualmente inválida (ou seja violando as restrições rígidas) construída aleatoriamente ou utilizando um método construtivo. Essa solução é depois iterativamente melhorada, de forma a deixar de violar as restrições rígidas e obter o melhor valor possível na função de avaliação.

De entre os problemas de optimização mais frequentes destacam-se os problemas de escalonamento[28]. Estes problemas têm como objectivo efectuar a determinação da melhor

[28] Os problemas de escalonamento são designados vulgarmente por *"scheduling problems"* na literatura anglo-saxónica.

distribuição de recursos a tarefas, com o objectivo de garantir a execução dessas tarefas mas minimizar os custos na sua execução. Problemas de decisão relativos ao escalonamento de tarefas e problemas relacionados tais como a geração de horários, a calendarização de exames ou a distribuição de serviço docente estão entre os mais difíceis dentro daqueles que pertencem à classe dos problemas NP-Completos.

6.3 Problemas de Escalonamento na Gestão Universitária

Neste trabalho pretendem-se aplicar metodologias de Inteligência Artificial à resolução de problemas de escalonamento da gestão universitária. Como domínio de aplicação foram seleccionados os problemas de escalonamento mais importantes que são vulgarmente encontrados na gestão universitária [Reis e Oliveira, 2001]:

- **Distribuição de Serviço Docente.** Atribuir a cada disciplina docentes responsáveis pela leccionação das suas diversas aulas (teóricas, práticas, etc.) às turmas que as frequentarem, tentando respeitar as preferências dos docentes relativas às disciplinas a leccionar, as horas máximas, mínimas e desejáveis a atribuir a cada docente em cada período lectivo (semestre, trimestre, etc.). Por vezes, a distribuição de serviço docente é realizada em várias fases. Inicialmente são averiguadas as preferências dos docentes. Em seguida, são designados os responsáveis das disciplinas. Na terceira fase é definido o serviço docente de todos os docentes (total de turmas de cada tipo e cada disciplina atribuídas a cada docente). Na fase final, as turmas individuais são atribuídas aos docentes.
- **Geração de Horários Escolares.** Este é um problema típico na maioria das escolas mas também frequente em universidades portuguesas. Consiste em alocar um tempo (ou conjunto de tempos) para cada aula de cada disciplina na instituição de ensino. A unidade de escalonamento é o grupo de alunos (turma) que se supõe ter um plano curricular comum. É considerado que a atribuição dos docentes às disciplinas já se encontra efectuada antes da resolução deste problema. A atribuição de salas pode ser efectuada em conjunto com a geração de horários ou as salas podem ser utilizadas como uma restrição e a sua atribuição individual ser realizada após a conclusão do processo de geração de horários.
- **Geração de Horários Universitários.** Escalonamento de todas as aulas de um conjunto de cursos numa Universidade ou Faculdade, minimizando a sobreposição de aulas com alunos em comum e evitando que sejam atribuídas duas aulas ao mesmo tempo aos docentes. Neste caso, os estudantes são tratados individualmente e não como pertencendo a um grupo (ao contrário da Geração de Horários Escolares). Os docentes, usualmente, estão já atribuídos às disciplinas, embora alguma flexibilidade nesta atribuição seja também comum. Os alunos

inicialmente encontram-se já inscritos nas disciplinas a frequentar embora alguma flexibilidade possa também ser admitida nesta inscrição. A definição de qual a turma a frequentar por cada aluno é usualmente parte do processo de geração de horários e não se encontra já tomada à partida. Em Portugal, é usual que mesmo os horários das Universidades sejam do tipo Horários Escolares[29], ou seja definidos por grupos de alunos.

- **Calendarização de Exames.** Escalonamento dos exames de um conjunto de cursos numa Universidade evitando a sobreposição de exames com alunos comuns e espalhando ao máximo (para os alunos) os exames. A alocação de salas e de vigilantes para os exames pode ser realizada em conjunto com o processo de calendarização de exames ou após a conclusão deste.
- **Alocação de Salas.** Quase todos os problemas de geração de horários ou calendarização de exames implicam a realização de um processo de alocação de salas. A cada evento (aula ou exame) tem de ser alocado espaço suficiente e adequado. Desta forma, a uma sala ou um conjunto de salas com capacidade e características adequadas à realização do evento deve ser atribuída. Problemas com eventos com múltiplas salas, partilha de salas por eventos e distâncias entre salas, são comuns no processo de alocação de salas.

Diversos outros problemas de escalonamento são resolvidos no dia a dia de cada Universidade. De entre estes, destacam-se a definição do número de turmas de cada disciplina, a atribuição dos alunos às turmas, a alocação de vigilantes ou a marcação de reuniões. Embora não sejam centrais para a execução deste trabalho, ao longo da tese, estes problemas serão também descritos resumidamente.

Devido à referida diversidade de estruturas universitárias, uma das questões mais importantes na gestão das mesmas é a decisão relativa à resolução centralizada destas tarefas ou à distribuição da sua resolução pela Unidades Autónomas (Departamentos, Secções) da Universidade [Dias, 1999]. Saber onde e por quem vão ser realizadas, depende de factores como: a dimensão da Universidade, o número de cursos, docentes e alunos afectos a si, a localização geográfica de todos os seus componentes e o seu contexto histórico. O contexto histórico é particularmente relevante neste processo uma vez que, após definido um modo de funcionamento, será muito difícil proceder à sua alteração numa determinada Universidade.

---

[29] Exceptuam-se a esta afirmação os horários da maioria dos mestrados e de cursos compostos maioritariamente por disciplinas opcionais em Universidade Portuguesas.

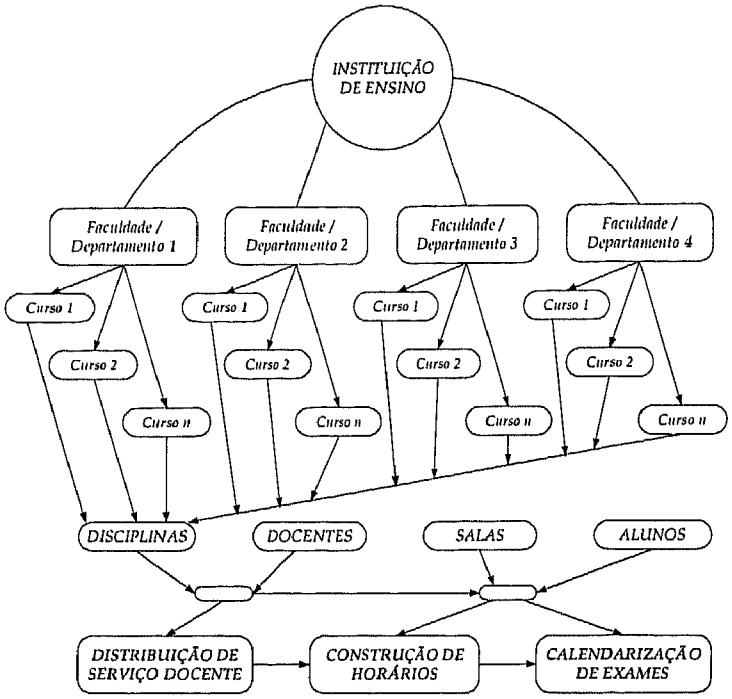

Figura 34: Problemas de Escalonamento numa Instituição de Ensino

Quando a Universidade é de reduzida dimensão e o número de cursos, docentes e discentes está de acordo com essa dimensão então as tarefas são, normalmente, exercidas pelos responsáveis gerais da Instituição, por vezes com o apoio de uma divisão criada para executar uma determinada tarefa de um modo semi-profissional. Esta divisão, dependendo da sua inserção na organização universitária e das suas funções, pode assumir designações diversas tais como Gabinete de Horários, Gabinete de Estudos e Planeamento, Gabinete de Gestão de Informação, etc.[30].

Quando a Universidade atinge uma dimensão considerável ao nível das suas instalações, do seu corpo docente e do número de discentes, então os Departamentos detêm, genericamente, o poder de desenvolver as suas próprias tarefas como se de uma Universidade ou Faculdade se tratasse. Entre outras tarefas, os Departamentos, têm o poder de formular planos curriculares dos cursos afectos a si, executar a distribuição do serviço dos seus docentes, gerar os seus próprios horários, construir a calendarização de exames das suas disciplinas e gerir as suas próprias salas de aula e laboratórios. No

[30] Na Universidade Fernando Pessoa, o autor deste trabalho fundou e coordenou durante os seu primeiros seis meses de existência, em 1996, o GEP – Gabinete de Estudos e Planeamento que tinha como funções: apoio às direcções de departamento na distribuição de serviço docente; análise de planos curriculares e manutenção da sua coerência; geração de horários; calendarização de exames; e gestão da circulação de informação na Universidade. Este gabinete permanece hoje em funcionamento da referida Universidade com funções acrescidas tais como a gestão dos sumários electrónicos e das páginas Web da Universidade.

entanto, esta autonomia e independência dos Departamentos não são totais pois existem sempre recursos e tarefas partilhados com outros departamentos da mesma Universidade.

Embora a compartimentação das Universidades ou Faculdades em departamentos permita separar o problema global em diversos subproblemas de mais fácil resolução, esta separação nunca é total, uma vez que existem sempre responsáveis globais hierárquicos, recursos partilhados e tarefas cuja responsabilidade não está totalmente definida. A coordenação dos processos de decisão de cada unidade autónoma com as outras unidades autónomas é essencial para a formulação de boas soluções finais para os diversos problemas. No entanto, é usual que esta coordenação seja menosprezada e que os departamentos resolvam os seus próprios problemas de forma autónoma e só no final, tentem tornar as suas decisões consistentes com as decisões dos outros departamentos.

6.3.1 Distribuição de Serviço Docente

6.3.1.1 Descrição do Problema

O problema da distribuição do serviço docente consiste em atribuir aos docentes aulas de disciplinas, (que usualmente podem ser teóricas, teórico-práticas, práticas ou laboratoriais) para as quais estão habilitados, tendo em conta as preferências e habilitações dos próprios para as leccionar.

Este processo de distribuição do serviço docente está ligado ao processo de geração de horários, pelo que por vezes existe uma grande pressão na elaboração destes dois processos, visto ser conveniente que a distribuição de serviço docente esteja completamente definida antes do processo de geração de horários.

No processo de distribuição do serviço docente ocorrem diversas etapas. A primeira etapa consiste na recolha dos planos curriculares e responsabilidades (a nível de secção, departamento ou faculdade) na leccionação de cada disciplina e na averiguação das preferências dos docentes em leccionar as diversas disciplinas. Esta recolha de informação é normalmente efectuada através de formulários, onde os docentes indicam um número restrito de disciplinas que tencionam leccionar, utilizando uma escala qualitativa. No entanto, a recolha verbal de preferência é também comum em instituições de ensino portuguesas, nomeadamente nas de menor dimensão. Nesta fase, algumas instituições aproveitam também para averiguar as cargas horárias desejadas pelos docentes e suas distribuições pelos períodos lectivos anuais (usualmente semestres) e quais as preferências de horário da cada docente.

A segunda etapa consiste na distribuição do serviço docente de todas as disciplinas (teóricas, teórico-práticas e práticas). O decisor (habitualmente o director da faculdade, departamento ou secção) utiliza nesta fase o seu conhecimento próprio sobre as capacidades e habilitações dos docentes e a ajuda dos formulários contendo a informação

relativamente às preferências dos docentes pelas disciplinas. O responsável pela distribuição escolhe os docentes que acha serem indicados para reger e leccionar as diversas aulas de cada disciplina, respeitando as cargas lectivas totais a atribuir a cada docente, de acordo com os seus tipos de contrato e categorias profissionais. Por vezes, é necessária a interacção com os docentes nesta fase do processo, de modo a permitir obter uma solução global que respeite todas as restrições. Existem, inclusivamente, instituições de ensino que realizam reuniões gerais, por área específica de conhecimento, para efectuar ou concluir a resolução do problema da distribuição de serviço docente.

A fase final consiste em optimizar esta distribuição do serviço docente usando o "feedback" dos docentes (reclamações e sugestões). Por vezes o processo subsequente de geração de horários pode implicar a alteração da distribuição de serviço docente. Um exemplo em que esta situação ocorre, consiste na atribuição de aulas de uma dada disciplina com horário nocturno a um docente que indique nas suas preferências de horário uma total indisponibilidade para leccionar aulas à noite.

6.3.1.2 Solução do Problema

A solução de um problema de distribuição de serviço docente pode ser definida como:

> ***Para cada disciplina a leccionar na faculdade/universidade, definir o(s) docente(s) regente(s) da mesma e para cada turma de cada tipo a leccionar de cada disciplina, definir o docente responsável pela sua leccionação, satisfazendo um determinado conjunto de restrições***

6.3.1.3 Restrições Rígidas

No processo de distribuição de serviço docente existe um elevado número de restrições rígidas que se deve respeitar de modo a obter uma adequada distribuição do serviço docente. Existem também outras restrições mais flexíveis cuja não violação, embora não seja essencial, contribui para o incremento da qualidade geral da distribuição de serviço. As restrições rígidas incluem:

- Para todas as disciplinas têm de haver um regente e docentes alocados para todas as turmas respectivas;
- Os docentes devem leccionar um número anual de horas com limites mínimos e máximos definidos pelas suas categorias profissionais, tipos de contratos e eventuais reduções de serviço lectivo devido a outros cargos que ocupem na instituição de ensino;
- Limite máximo de disciplinas distintas que um dado docente pode leccionar;
- Limite máximo de horas semanais por período (usualmente semestre) que os docentes podem leccionar;

- Nenhum docente deverá leccionar uma disciplina para a qual não esteja habilitado;
- Impossibilidades de determinados docentes leccionarem em determinados períodos de tempo (semestres ou trimestres).

6.3.1.4 Restrições Flexíveis

As restrições flexíveis do problema da distribuição de serviço docente (cujo cumprimento embora não seja indispensável é essencial para avaliar a qualidade de uma solução) incluem:

- Preferências dos docentes relativamente às disciplinas a regerem e leccionarem em cada período;
- Docentes não devem ter um número de disciplinas distintas para leccionar inferior/superior a limites fixados;
- Docentes não devem ter um número de horas de aulas semanais para leccionar em cada período (usualmente semestre) inferior/superior a limites fixados;
- Preferências relativas à distribuição da carga horária dos docentes pelos diversos períodos do ano;
- Preferências relativas à formação de grupos de docentes para leccionar determinadas disciplinas;
- Incompatibilidades de docentes para leccionar disciplinas em conjunto com outros docentes;
- Preferências do coordenador do curso relativas aos docentes regerem ou leccionarem cada disciplina;
- Respeito pelas categorias profissionais e hierarquia interna dos docentes na distribuição do serviço;
- Preferências ou outro tipo de directivas da direcção do departamento/faculdade ou reitoria da Universidade;
- Preferência de determinado docente em leccionar no período da tarde/manha/noite (que pode condicionar a atribuição de algumas disciplinas);
- Preferências de docentes relativas à leccionação de disciplinas a turmas específicas.

Por vezes, a solução de um problema de distribuição de serviço docente, implica mesmo a alteração dos planos curriculares de cursos, não leccionando uma dada disciplina num ano porque um dado docente se encontra indisponível, ou alterando o semestre de leccionação devido às preferências de distribuição de carga de um docente.

6.3.2 Geração de Horários

6.3.2.1 Descrição do Problema

Um problema de geração de horários pode ser entendido como a fixação, no espaço e no tempo, de uma sequência de encontros entre docentes e alunos, num determinado período de tempo (tipicamente uma semana), tentando satisfazer as diversas restrições associadas.

Um horário é, basicamente, uma tabela que descreve a alocação de recursos, no tempo, para a realização de um conjunto de tarefas. Entenda-se que neste processo de geração de horários, os recursos são considerados como sendo as salas, turmas e docentes que estão alocados a tempos lectivos (dias ou períodos de tempos). Por tarefas, a serem realizadas, considera-se o conjunto de aulas que será leccionado em cada semana.

O processo de geração de horários é uma tarefa complicada, difícil e que se pode tornar demorada, não só devido à diversidade de restrições que se tem que ter em conta, mas também devido ao facto de envolver muitas pessoas (alunos, docentes, administração da universidade, responsáveis pelas faculdades, departamentos e cursos). Segundo Bloomfield e McSarry, [Bloomfield e McSarry, 1979], dependendo do tamanho e da diversidade de ofertas de disciplinas, o tempo necessário para efectuar a construção dos horários pode variar de uma simples tarde até um intensivo mês de trabalho árduo. Na figura 35 é apresentado um exemplo de um horário.

UNIVERSIDADE FERNANDO PESSOA

GABINETE DE ESTUDOS E PLANEAMENTO

Horário de Turma - 1º Semestre

Licenciatura: **Ciências da Comunicação**

Horário: *Ano:* 3 *Turma:* A

	2ª Feira			3ª Feira			4ª Feira			5ª Feira			6ª Feira			Sábado		
8																TXTJOR2	RJP	304
9																TXTJOR2	RJP	304
10																TXTPUB2	RRM	304
11										MKTIND	MOM	203				ING3	SPM	304
12										MKTIND	MOM	203						
13	ING3	SPM	306				GC3	MCP	102				THRP	JNP	304			
14	ING3	SPM	306	SEMCOM2	PCR	304	GC3	MCP	102	THRP	JNP	102	THRP	JNP	304			
15	TXTJOR2	RJP	306	SEMCOM2	PCR	304	GC3	MCP	102	THRP	JNP	102	MKTIND	MOM	304			
16	TXTJOR2	RJP	306	CEM	AGO	304	CEM	AGO	306	TXTPUB2	RRM	102	MKTIND	MOM	304			
17				CEM	AGO	304	CEM	AGO	306	TXTPUB2	RRM	102	SEMCOM2	PCR	304			
18																		
19																		
20																		
21																		
22																		

Código	Disciplina	u.c.	Tip.	H	1ºS	2ºS	Docente	
CEM	Comunicação Empresarial e Mercadologia	7	AN	8	4	4	AGO	António Garria Oliveira
GC3	Gramática da Comunicação III	6	AN	6	3	3	MCP	Mª Helena Patrão
ING3	Inglês III	6	AN	6	3	3	SPM	Susana Teixeira
MKTIND	Marketing Industrial	7	AN	8	4	4	MOM	Maria Olinda Marques
SEMCOM2	Semiótica da Comunicação II	6	AN	6	3	3	PCR	Pedro Reis
THRP	Teoria e História das Relações Públicas	7	AN	8	4	4	JNP	João Nuno Patrício
TXTJOR2	Texto Jornalístico II	7	AN	8	4	4	RJP	Ricardo Jorge Pinto
TXTPUB2	Texto Publicitário II	6	AN	6	3	3	RRM	Rui Ribeiro de Melo

Figura 35: Exemplo de um Horário

A geração de horários para universidades, consiste na definição de horários para todas as aulas das diversas disciplinas, de todos os anos da universidade, minimizando a sobreposição de aulas com alunos em comum, evitando a sobreposição de salas. O problema consiste em atribuir aulas a períodos em que nenhum docente, turma ou sala esteja envolvido em mais do que uma aula ao mesmo tempo, evitando assim as sobreposições dos recursos existentes na universidade e relevantes para o bom funcionamento deste processo. É necessário ter em conta, no processo de geração de horários universitários, a carga horária dos docentes e dos alunos, da mesma forma, que se deve respeitar as restrições e preferências pois estas tornarão o horário mais eficaz, tanto para os docentes como para os alunos. Este problema, como foi referido anteriormente, é unicamente usual em Mestrados e disciplinas opcionais em Portugal. Usualmente os problemas de geração de horários em Portugal são tratados como problemas de geração de horários escolares (agrupando os alunos em turmas).

6.3.2.2 Solução do Problema

A solução de um problema de geração de horários pode ser definida como:

> ***Para cada aula de cada turma de cada disciplina leccionada em cada período numa dada instituição de ensino, atribuir-lhe um tempo (dia e hora) de início e uma sala com características apropriadas, impedindo que os docentes, turmas e salas sejam utilizados para duas aulas simultaneamente e obedecendo a um conjunto de outras restrições.***

6.3.2.3 Restrições Rígidas

Um dos principais problemas na construção de um horário está relacionado com o elevado número de restrições que deve ser respeitado, de modo a obter um bom horário. Devido a este elevado número de restrições que pode existir nas diversas Instituições de Ensino Superior, muitas delas não podem ser totalmente respeitadas. Neste sentido, pode-se atribuir um grau de preferência a cada uma delas, permitindo assim obter prioridades na resolução do processo de geração de horários.

As restrições rígidas de um problema de geração de horários incluem:

- Nenhum docente pode leccionar duas ou mais aulas em simultâneo;
- Nenhum aluno ou grupo de alunos (turma) pode ter duas ou mais aulas em simultâneo;
- Na mesma sala não pode haver duas ou mais aulas leccionadas em simultâneo;
- O número de alunos numa sala não deverá exceder a sua capacidade máxima;
- Cada aula de uma disciplina deve ser leccionada numa sala do tipo apropriado (anfiteatro, laboratório de informática, laboratório de química, etc.);

- Impossibilidades lectivas dos docentes por períodos de tempo (dia/hora);
- Impossibilidades lectivas das turmas por períodos de tempo (dia/hora). Por exemplo, existem turmas cujos horários têm de ser obrigatoriamente nocturnos;
- Indisponibilidade de salas por períodos de tempo (dia/hora);
- Os alunos/turmas e os docentes não devem ter mais do que um certo número de aulas (ou horas de aulas) consecutivas;
- Os alunos/turmas e os docentes não devem ter mais do que um certo número de aulas (ou horas de aulas) num dado dia;
- Inclusão de tempo disponível para almoço (ou jantar) nos horários das turmas e dos docentes.

6.3.2.4 Restrições Flexíveis

As restrições flexíveis que determinam a qualidade de um dado horário incluem:

- As aulas para os alunos e docentes devem ser agrupadas em blocos de média dimensão (i.e. não devem existir aulas isoladas nem grupos de aulas com longa duração);
- Os alunos e os docentes não devem ter "furos" (tempos sem aulas precedidos e seguidos por tempos com aulas) nos horários;
- Preferências de horário dos docentes (relativamente a leccionar em cada período de tempo lectivo);
- Preferências de horário das turmas (relativamente a terem aulas em cada período de tempo lectivo);
- Preferências de utilização das salas em cada período de tempo lectivo;
- Preferências de leccionação de disciplinas em cada período de tempo lectivo (por exemplo, determinadas disciplinas não devem ser leccionadas ao início ou ao fim da tarde);
- No caso de universidades com uma elevada dimensão, os alunos e docentes devem ter tempo para se deslocarem de uma aula para a outra;
- Aulas da mesma disciplina devem ser leccionadas em dias diferentes e espaçadas o mais possível no horário;
- Garantia de dias ou partes de dias (manhãs, tardes ou noites) livres para docentes, turmas ou salas;
- Docentes que devem ter horários compatíveis (mesmos dias de aulas e horários semelhantes) ou incompatíveis (não leccionarem ao mesmo tempo);
- Aulas que devem ocorrer exactamente antes de outras ou num dia anterior a

outras aulas do horário;

- Disciplinas que não podem ser leccionadas no mesmo dia ou em tempos consecutivos a outras disciplinas;
- Disciplinas que têm de ser leccionadas simultaneamente (ou no mesmo dia) que outras disciplinas.

Dada a complexidade do problema, diversas outras restrições podem ser incluídas. Diferentes instituições de ensino utilizam restrições distintas na geração dos seus horários.

6.3.3 Calendarização de Exames

6.3.3.1 Descrição do Problema

O problema da calendarização de exames requer a atribuição de blocos de tempo a um determinado número de exames, um por disciplina, dentro de um espaço de tempo limitado e satisfazendo um conjunto de restrições. A construção de calendários de exames é uma tarefa comum a todas as universidades, cuja resolução manual envolve muito esforço e se revela normalmente insatisfatória para os alunos.

Segundo Andrea Schaerf [Schaerf, 1995], um calendário de exames consiste no escalonamento dos exames de um conjunto de disciplinas de uma instituição de ensino evitando a sobreposição de exames das disciplinas com alunos em comum, e dispersando os exames dos alunos, o mais possível. Com base nesta definição, conclui-se que se deve evitar que determinado aluno tenha dois ou mais exames distintos na mesma hora, garantindo, preferencialmente, que estes sejam espalhados o mais possível pelo calendário. Neste contexto os exames devem também estar divididos por diferentes dias, de forma a minimizar o número de exames no mesmo dia, combinando os dias entre si para minimizar o número de exames em dias consecutivos.

O problema pode ser dividido em duas fases. A primeira consiste na elaboração de um calendário que respeite determinadas condições rígidas que nos conduz a um calendário aplicável. Para se obter um calendário válido deve-se ter em conta que todos os exames só devem ser calendarizados uma só vez, e que se dois exames forem calendarizados para o mesmo período de tempo, estes não devem entrar em conflito, ou seja, devem estar alocados a diferentes salas e não devem conter os mesmos docentes ou alunos. A segunda fase consiste na introdução de factores de optimização do calendário que permitem melhorar a sua qualidade final.

O objectivo final da calendarização de exames, é o de garantir que todos os alunos possam realizar todos os exames a que se propõem, sem que o calendário afecte significativamente as suas hipóteses de obter aprovação nas disciplinas, ou pelo menos, que tenha um efeito semelhante em todos os estudantes [Weare, 1995]. Na figura 36 é apresentado um exemplo de um calendário de exames.

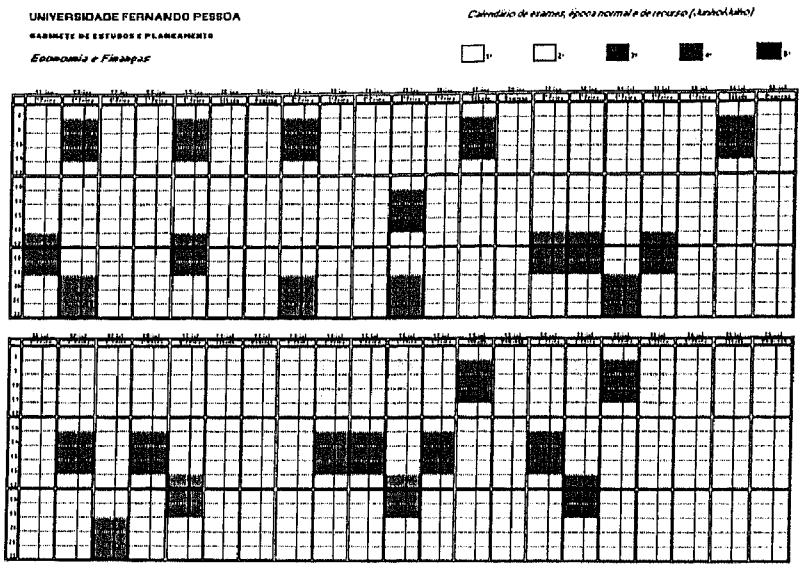

Figura 36: Exemplo de um Calendário de Exames

O problema da calendarização de exames é muito semelhante ao problema da geração de horários e é difícil fazer uma distinção clara entre os dois [Schaerf, 1995]. As diferenças exactas dependem dos problemas específicos. No entanto, algumas diferenças gerais são geralmente aceites [Barber 1997], [Reis 1999a]:

- Só existe um exame para cada disciplina por período (e usualmente diversas aulas);
- O período de avaliação é tipicamente maior do que uma semana, e o número de períodos pode variar;
- Os estudantes são normalmente tratados individualmente na calendarização de exames;
- Na maioria dos problemas de calendarização de exames pode existir mais do que um exame na mesma sala;
- Um exame pode ser dividido por múltiplas salas;
- A disposição das mesas e a distribuição dos lugares pelos alunos podem ser importantes na marcação de exames;
- A condição de conflito para os alunos é estrita nos exames, enquanto na geração de horários, por vezes, se aceita que um aluno possa ter sobreposições de aulas, não é geralmente aceitável que tenha sobreposição de exames;
- Os exames, por aluno, devem ser espalhados pelo calendário ao contrário dos horários de aulas em que é aconselhável agrupar as disciplinas em blocos.

6.3.3.2 Solução do Problema

A solução de um problema de calendarização de exames consiste em:

Para cada exame em cada período de cada disciplina de uma dada instituição de ensino, atribuir-lhe um tempo (dia e hora) de início e um conjunto de salas respeitando um determinado conjunto de restrições. Associada à resolução deste problema, por vezes, está também a atribuição de um conjunto de vigilantes para cada exame.

6.3.3.3 Restrições Rígidas

A calendarização de exames deve respeitar determinadas restrições rígidas:

- Nenhum aluno pode ter dois ou mais exames em simultâneo;
- Nenhum docente pode ter dois ou mais exames em simultâneo;
- O período de avaliação deve ter uma dimensão suficiente para se realizarem todos os exames;
- Cada exame deve ter um número de vigilantes suficiente;
- Indisponibilidades de docentes, turmas/alunos e salas nos diversos períodos de tempo;
- Cada exame deve ter alocado um número de salas com uma capacidade total suficiente para todos os alunos que o realizam;
- Cada exame deve ser alocado em salas do tipo apropriado;
- Todos os docentes da disciplina devem ser alocados como vigilantes.

6.3.3.4 Restrições Flexíveis

As restrições que permitem avaliar a qualidade de um calendário de exames são:

- Preferências dos docentes relativamente a dar exames em cada dia/hora;
- Preferências das turmas/alunos relativamente a terem exames em cada dia/hora;
- Preferências de utilização das salas em cada período de tempo;
- Exames que têm de ocorrer antes ou depois de outros exames;
- Exames que devem ocorrer em períodos de tempo ou dias consecutivos a outros exames;
- Exames que têm de ocorrer simultaneamente com outros exames;
- Exames que devem ocorrer entre duas datas pré-determinadas;
- Nenhum aluno deve ter exames em períodos de tempo ou dias consecutivos;
- Exames calendarizados na mesma sala devem ter a mesma duração (pode ocorrer o caso de existir mais do que um exame na mesma sala);

- Exames para os alunos devem ser espalhados no calendário;
- Distância entre as salas utilizadas para cada exame deve ser mínima;
- Preferências dos docentes relativamente aos exames que desejam vigiar;
- Deve existir um dado tempo ou número de dias de intervalo entre dois exames para o mesmo aluno;
- Utilização das salas (em número de *salas*tempos*) deve ser minimizada, i.e. as salas com maior capacidade devem ser utilizadas preferencialmente e a capacidade total das salas alocadas a um exame deve ser aproximadamente a necessária;
- Número de vigilantes a utilizar deve ser minimizado;
- Número de vigilâncias alocadas a cada docente deve ser o indicado para a sua categoria profissional ou tipo de contrato.

Da análise das restrições apresentadas é visível que o problema da calendarização de exames, embora à partida pudesse parecer mais simples do que a geração de horários, introduz uma série de complexidades, relacionadas essencialmente com o tratamento individual dos estudantes e alocação de conjuntos de salas, que o torna de muito difícil resolução.

6.3.4 Alocação de Salas

A alocação de salas consiste em atribuir salas com capacidade suficiente e características aconselháveis à realização de um conjunto de eventos lectivos, aulas ou exames, de uma Universidade ou Faculdade.

O processo de alocação de salas é normalmente realizado em conjunto com os processos de geração de horários ou calendarização de exames. Esta resolução conjunta implica, no entanto, a introdução de uma variável adicional na resolução dos problemas, aumentando desta forma consideravelmente o espaço de pesquisa. Desta forma, em problemas em que exista um conjunto de salas com características semelhantes, é vantajoso utilizar as salas inicialmente como uma restrição e realizar a sua alocação específica, unicamente após ter sido concluído o processo de geração de horários ou calendarização de exames.

6.3.5 Visão Integrada

Da análise realizada aos diversos problemas de escalonamento da gestão universitária, é visível que estes se encontram interrelacionados e, que a sua resolução individual de forma totalmente separada não é a mais aconselhável. No entanto, as diferentes abordagens à resolução destes problemas encontradas na literatura tratam cada um dos problemas de forma separada dos restantes, sem ter em contra as possíveis interacções entre as soluções

de cada problema. No capítulo 7 será analisada a forma de integrar todos estes problemas de escalonamento através da definição de uma linguagem de representação – *SCHEDULING UNILANG* – que os permite representar a todos de forma integrada, permitindo desta forma efectuar a sua resolução conjunta.

6.4 Técnicas de Resolução dos Problemas de Escalonamento

Ao longo do tempo, variadas técnicas de construção de horários, calendarização de exames e resolução dos problemas associados, foram aparecendo, sendo as primeiras baseadas na simulação do método humano de resolução do problema. A ideia geral destas técnicas era a de ir definindo o horário, para cada aula, começando com as aulas que apresentavam o maior número de restrições. As diferentes técnicas diferiam apenas nos critérios usados para definir os tipos de restrições. Mais tarde, começaram-se a aplicar técnicas automáticas de resolução destes problemas, como sejam a *Programação Inteira* [Lawrie, 1969] e a *Coloração de Grafos* [Welsh e Powell, 1967]. Outros autores, aplicaram mais tarde com sucesso *Meta-Heurísticas* na resolução deste tipo de problemas, como sejam o *Arrefecimento Simulado* [Abramson, 1993] [Schaerf e Schaerf, 1995], *Pesquisa Tabu* [Hertz, 1992] e *Algoritmos Genéticos* [Colorni et al., 1992]. Embora a utilização de algoritmos de geração de horários (ou calendarização de exames) automáticos seja desejável, existem também diversas vantagens na geração de horários manual ou simplesmente assistida por computador. Isto levou diversos autores [Verbraeck, 1990] a defender a utilização de *Métodos Interactivos* de geração de horários ou baseados em *Sistemas de Apoio à Decisão*. Mais recentemente têm sido utilizadas técnicas derivadas da Inteligência Artificial como a *Programação em Lógica* [Fahrion e Dollanski, 1992] [Kang e White, 1992], *Sistemas Periciais* [Gudes et al., 1990] e a *Programação e Lógica com Restrições* [Gueret et al., 1995] [Stamatopolus et al., 1998] [Reis e Oliveira, 1999a].

Os diferentes métodos para resolver, com sucesso, problemas reais de construção de horários, são em quase todos os casos, usados somente numa escola ou instituição em particular, e raramente são devidamente comparados entre si. No entanto, uma comparação apurada é vital para determinar quais os melhores métodos computacionais, na presença de variados tipos de dados para construção de horários. Quando as técnicas usadas são relativamente simples, os dados são facilmente reproduzidos. Mas essa tarefa é cada vez mais complicada, devido ao desenvolvimento de técnicas de construção de horários mais sofisticadas, em resposta, em parte, à necessidade de lidar com um maior e mais variado número de restrições.

6.4.1 Coloração de Grafos

O algoritmo de *Coloração de Grafos* é um dos métodos clássicos de construção de horários que se foi impondo ao longo do tempo, existindo diversa literatura sobre o problema. A grande divulgação desta técnica de geração de horários originou o aparecimento de muitos autores com diversas variantes do problema.

De uma forma geral, o algoritmo de *Coloração de Grafos* é um método de pesquisa onde cada vértice, no grafo, representa o conjunto (aula, docente, sala). Dois vértices estão ligados se não conseguem ser colocados no horário simultaneamente, isto é, se têm uma aula, um docente ou uma sala em comum, para o mesmo período de tempo. A resolução do problema reduz-se a encontrar uma coloração do grafo, em que dois vértices que estejam ligados não tenham a mesma cor.

É possível representar facilmente problemas simples de construção de horários como sendo um grafo. Considerando que temos **n** eventos ($ev_1...ev_n$), alguns dos quais não podem ser escalonados ao mesmo tempo devido a determinados conflitos (quaisquer que sejam as razões do conflito). Isto pode ser representado como um grafo, que consiste num conjunto V de vértices $V = \{v_1, ..., v_n\}$, onde v_i corresponde ao evento ev_i. Para completar o grafo necessitamos de um conjunto de ramos que representam os conflitos entre os eventos. Isto pode ser representado como $E = \{i, j \in N \bullet conflito(ev_i, ev_j) \bullet (v_i, v_j)\}$ onde $conflito(ev_i, ev_j)$ é um predicado que será verdadeiro se, e só se, os eventos ev_i e ev_j não puderem ser escalonados em simultâneo. Seria importante escalonar todos estes eventos dentro de um número limite de períodos de tempo, para que eventos em conflito sejam escalonados em períodos de tempo diferentes. Isto pode ser relacionado com o modelo de grafos, colorindo cada vértice do grafo para que dois vértices ligados por determinado ramo não tenham a mesma cor. Aqui, cada cor corresponde a um período de tempo no horário. Fica claro que este problema seria facilmente resolvido se tivéssemos n cores disponíveis e simplesmente atribuíssemos a cada vértice uma única cor, mas poderão haver limitações práticas no número de cores disponível.

A utilização deste modelo ajuda à resolução de um problema bem estruturado mas tem como limitação que apenas considera a atribuição de eventos a períodos sem conflitos. É possível, no entanto, melhorar o modelo para descrever uma maior variedade de restrições. Outra limitação está relacionada com o facto de só ser considerada a atribuição de tempos aos eventos e não a atribuição de salas.

6.4.2 Programação Inteira

Um problema de *Programação Inteira* não é mais do que um problema de programação linear, com variáveis inteiras. Winston [Winston, 1991], define programação linear da seguinte forma: "Um problema de programação linear é um problema de optimização, para o qual é necessário tentar maximizar (ou minimizar) uma função linear que

representa as variáveis de decisão. Essa função é chamada função objectivo do problema. Os valores das variáveis de decisão devem satisfazer um conjunto de restrições. Cada restrição deve ser uma equação linear ou uma desigualdade linear. Um sinal de restrição é associado a cada variável. Para cada variável x_i, o sinal da restrição específica que, o valor de x_i deve ser não negativo ou que x_i tenha o sinal não restringido".

As soluções baseadas em programação inteira assentam em formulações matemáticas dos problemas a resolver. Existem várias formulações matemáticas, de diversos autores [deWerra, 1985], sobretudo para o problema da geração de horários. Uma formulação matemática é apresentada por [Stamatopoulos et al., 1998] que assume existirem N disciplinas, M docentes e L salas. Considera, também, um conjunto de dias de trabalho, ***Dias***, bem como um conjunto de períodos de tempo, ***Períodos***, por cada dia. Por exemplo, poderá ser o caso de ***Dias = {seg, ter, ..., sex}*** e ***Períodos = {09-10, 10-11, ..., 19-20}***.

Para cada disciplina i ($1 \leq i \leq N$) existe um dado número de parâmetros:

v_i: *número aproximado de alunos que assiste à disciplina i*

l_i: *número de aulas da disciplina i*

p_{ij}: *número de sessões (períodos de leccionação) da aula j da disciplina i* ($1 \leq j \leq l_i$)

t_i: *número de docentes da disciplina i*

n_{ij}: *docente j da disciplina i* ($1 \leq j \leq t_i$, $1 \leq n_{ij} \leq M$)

d_i: *número de disciplinas disjuntas (sem conflitos) da disciplina i*

s_{ij}: *disciplina disjunta j com a disciplina i* ($1 \leq j \leq d_i$, $1 \leq s_{ij} \leq N$, $s_{ij} \neq i$)

us_i: *número de slots indisponíveis (u-slots) da disciplina i*

dus_{ij}: *dia do j u-slot da disciplina i* ($1 \leq j \leq us_i$, $dus_{ij} \in$ ***Dias***)

pus_{ij}: *período de leccionação do j u-slot da disciplina i* ($1 \leq j \leq us_i$, $pus_{ij} \in$ ***Períodos***)

Para cada docente i ($1 \leq i \leq M$) definem-se os seguintes parâmetros:

ut_i: *número de* ***u-slots*** *do docente i*

dut_{ij}: *dia do j* ***u-slot*** *do docente i* ($1 \leq j \leq ut_i$, $dut_{ij} \in$ ***Dias***)

put_{ij}: *período de leccionação do j* ***u-slot*** *do docente i* ($1 \leq j \leq ut_i$, $put_{ij} \in$ ***Períodos***)

Finalmente, os parâmetros para cada sala i ($1 \leq i \leq L$) são:

c_i: *capacidade da sala i*

uc_i: *número de* ***u-slots*** *da sala i*

duc_{ij}: *dia do j* ***u-slot*** *da sala i* ($1 \leq j \leq uc_i$, $duc_{ij} \in$ ***Dias***)

puc_{ij}: *período de leccionação do j* ***u-slot*** *da sala i* ($1 \leq j \leq uc_i$, $puc_{ij} \in$ ***Períodos***)

O que é necessário considerar é o tempo e espaço de todas as sessões das aulas de cada disciplina que têm de ser escalonadas, ou seja:

x_{ijk}: *dia da sessão k da aula j da disciplina i*

y_{ijk}: *período de leccionação da sessão k da aula j da disciplina i*

z_{ijk}: *sala da sessão k da aula j da disciplina i*

onde $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$, $x_{ijk} \in$ *Dias*, $y_{ijk} \in$ *Períodos e* $1 \leq z_{ijk} \leq L$.

As restrições que a solução tem de satisfazer são as seguintes:

1) Duas sessões não podem ter lugar ao mesmo tempo e na mesma sala:

$x_{ijk} \neq x_{i'j'k'} \vee y_{ijk} \neq y_{i'j'k'} \vee z_{ijk} \neq z_{i'j'k'}$

$\forall i, j, k, i', j', k'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$, $1 \leq i' \leq N$, $1 \leq j' \leq l_{i'}$, $1 \leq k' \leq p_{i'j'}$ *e* $i \neq i'$ *ou* $j \neq j'$ *ou* $k \neq k'$.

2) As sessões de uma aula têm de ter lugar em períodos de leccionação consecutivos, no mesmo dia e na mesma sala:

$x_{ijk} = x_{ij(k-1)}$

$y_{ijk} = próximo(y_{ij(k-1)})$

$z_{ijk} = z_{ij(k-1)}$

$\forall i, j, k$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$ *e* $2 \leq k \leq p_{ij}$ *onde próximo(tp) é o período de leccionação imediatamente a seguir ao período tp (ex: próximo(10-11) = (11-12)).*

3) As aulas de uma disciplina têm de ter lugar em dias diferentes:

$x_{ijk} \neq x_{i'j'k'}$

$\forall i, j, j', k, k'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq j' \leq l_i$, $j \neq j'$, $1 \leq k \leq p_{ij}$ *e* $1 \leq k' \leq p_{ij'}$.

4) As capacidades das salas têm de ser respeitadas:

$v_i \leq c_{zijk}$

$\forall i, j, k$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$ *e* $1 \leq k \leq p_{ij}$.

5) Nenhum docente pode leccionar duas aulas ao mesmo tempo:

$x_{ijk} \neq x_{i'j'k'} \vee y_{ijk} \neq y_{i'j'k'}$

$\forall i, j, k, i', j', k'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$, $1 \leq i' \leq N$, $1 \leq j' \leq l_{i'}$, $1 \leq k' \leq p_{i'j'}$, $i \neq i'$ *e* $\{n_{ij''} \mid 1 \leq j'' \leq t_i\} \cap \{n_{i'j'''} \mid 1 \leq j''' \leq t_{i'}\} \neq \emptyset$.

6) Os requisitos de indisponibilidade dos docentes têm de ser respeitados:

$dut_{nijj'} \neq x_{ij''k} \vee put_{nijj'} \neq y_{ij''k}$

$\forall i, j, j', j'', k$ *com* $1 \leq i \leq N$, $1 \leq j \leq t_i$, $1 \leq j' \leq ut_{nij'}$, $1 \leq j'' \leq l_i$ *e* $1 \leq k \leq p_{ij''}$.

7) Os requisitos de indisponibilidade das salas têm de ser respeitados:

$duc_{zijkj'} \neq x_{ijk} \vee puc_{zijkj'} \neq y_{ijk}$

$\forall i, j, k, j'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$ *e* $1 \leq j' \leq uc_{zijk}$.

8) Os requisitos de indisponibilidade das disciplinas têm de ser respeitados:

$dus_{ij} \neq x_{ij'k} \vee pus_{ij} \neq y_{ij'k}$

$\forall i, j, j', k \text{ com } 1 \leq i \leq N, 1 \leq j \leq us_i, 1 \leq j' \leq l_i \text{ e } 1 \leq k \leq p_{ij'}$.

9) Os requisitos de não conflitualidade entre as disciplinas têm de ser respeitados:

$x_{ijk} \neq x_{sij'j''k'} \vee y_{ijk} \neq y_{sij'j''k'}$

$\forall i, j, k, j', j'', k' \text{ com } 1 \leq i \leq N, 1 \leq j \leq l_i, 1 \leq k \leq p_{ij}, 1 \leq j' \leq d_i, 1 \leq j'' \leq l_{sij'} \text{ e } 1 \leq k' \leq p_{sij'j''}$.

De referir que na formulação anterior não se encontram codificados os tipos de disciplinas. É assumido que essa informação já foi pré-processada e o resultado é representado por pares de disciplinas disjuntas. O objecto é aqui encontrar uma solução possível e estamos perante um problema de pesquisa e não de optimização da solução. A formulação de um problema de geração de horários como um problema de optimização implica conjunto adicional de restrições flexíveis.

6.4.3 Heurísticas

As heurísticas utilizam conhecimento específico sobre o problema e a sua forma de resolução para obter soluções satisfatórias de uma forma eficiente. É uma forma de abordagem muito utilizada na resolução de problemas combinatórios de resolução complicada. Normalmente, as heurísticas directas preenchem o horário completo com uma aula (ou um grupo de aulas) por período de tempo, desde que não existam conflitos. Nessa altura, começam-se a efectuar trocas e reescalonamento das aulas já alocadas, de forma a acomodar outras aulas.

Um exemplo típico da utilização bem conseguida de heurísticas é o sistema *Schola* [Uhlemann et al., 1969]. O sistema é baseado nas três estratégias seguintes:

- A) Atribuir a aula mais "urgente" ao período de tempo mais "favorável" para essa aula;
- B) Quando um período de tempo só pode ser utilizado para uma aula, então é atribuído a essa aula;
- C) Mover uma aula já escalonada, para um período de tempo livre, de forma a libertar esse período, para que seja possível tentar escalonar determinada aula que se esteja nesse momento a tratar.

Uma aula é "*urgente*" quando tem associada muitas restrições. Por exemplo, quando um docente (e a turma) tem poucos períodos de tempo disponíveis para alocação e muitas aulas para escalonar. Um período de tempo é "*favorável*" quando um pequeno número de outras aulas pode ser escalonado nesse período de tempo, baseado na disponibilidade dos outros docentes e turmas. O sistema *Schola* tenta escalonar as aulas, alternando a utilização das estratégias A e B. Quando mais nenhuma aula pode ser escalonada desta forma, então o sistema passa a usar a estratégia C. A estratégia A é o núcleo do sistema e é utilizada em quase todos os sistemas, com diferentes formas de definição do que é

"urgente" e do que é "favorável". A utilização da estratégia B pode prevenir que a estratégia A entre em situações sem alternativa. A estratégia C permite, de uma forma limitada, "*backtracking*" de forma a recuperar dos "erros" da estratégia A.

6.4.4 Meta-Heurísticas

Técnicas de pesquisa baseadas na Inteligência Artificial e Investigação Operacional são designadas por *Meta-Heurísticas* e foram propostas nas décadas de 70 e 80. As *Meta-Heurísticas* são técnicas inteligentes de optimização, que constituem uma estrutura geral para construir heurísticas para resolver problemas tipicamente NP-Completos. Por outras palavras, são um conjunto de conceitos que pode ser utilizado para definir métodos heurísticos que podem ser aplicados a um vasto conjunto de problemas distintos.

As *Meta-Heurísticas* podem ser aplicadas a problemas combinatórios, em que uma solução pode ser calculada em tempo polinomial, mas tal não é prático. As *Meta-Heurísticas* são uma estratégia de alto nível que guia outras heurísticas na pesquisa de soluções admissíveis em domínios onde esta tarefa é complexa. Uma analogia é a utilização de uma meta-linguagem para explicar uma linguagem [Greenberg, 1999]. Por exemplo, para explicar uma linguagem de computador, utilizam-se símbolos, tais como parênteses, na meta-linguagem, para denotar as propriedades da linguagem a ser descrita, tais como parâmetros que são opcionais.

As *Meta-Heurísticas* mais conhecidas são:

- *Arrefecimento Simulado* [Kirkpatrick et al., 1983];
- *Pesquisa Tabu* [Glover e Laguna, 1993];
- *Algoritmos Genéticos* [Holland, 1975].

Estas Meta-Heurísticas, sendo uma generalização dos procedimentos de pesquisa local (heurísticas de melhoria iterativa), constituem métodos de pesquisa genérica aplicáveis a grandes conjuntos de problemas de optimização.

6.4.4.1 Arrefecimento Simulado

O *Arrefecimento Simulado* impôs-se no início da década de 80, pela sua simplicidade de implementação e ampla aplicabilidade. A utilização de um algoritmo de *Arrefecimento Simulado* foi proposto por [Kirkpatrick et al., 1983]. O algoritmo, baseia-se numa analogia com a termodinâmica, especificamente com a forma como os líquidos congelam e cristalizam, ou metais arrefecem. Esta é uma técnica padronizada depois do processo físico de arrefecimento de um sólido no ramo da Engenharia Metalúrgica. A essência do processo é um arrefecimento lento, permitindo tempo suficiente para a redistribuição dos átomos, conforme vão perdendo mobilidade. É essencial garantir que, após o arrefecimento, é atingido um estado de baixa energia.

A nível funcional, o *Arrefecimento Simulado* é um método de pesquisa local em que se pretende, de forma iterativa e seguindo uma determinada estratégia de pesquisa, minimizar uma função de custo sobre um espaço de soluções. Este método assume uma estrutura de problema que pode ser facilmente optimizado, para servir diferentes tipos de problemas. O algoritmo típico de arrefecimento simulado aceita uma nova solução se o custo é menor do que o custo da solução corrente. Mesmo que o custo da nova solução seja maior, existe a probabilidade desta solução ser aceite, que depende da temperatura actual do sistema. Com este critério de aceitação, é possível abandonar óptimos locais da função de avaliação. Conforme a temperatura vai diminuindo, esta probabilidade de aceitação vai diminuindo para que, no final do processo, só as soluções melhores sejam aceites, convergindo a solução para o óptimo local mais próximo.

Um exemplo de aplicação de Arrefecimento Simulado ao Problema de Construção de Horários foi desenvolvido por Abramson [Abramson, 1991] [Abramson, 1993]. No seu trabalho Abramson considera a possibilidade que duas turmas diferentes tenham alunos em comum. Uma solução é descrita por uma lista de conjuntos de aulas (uma lista por cada período). Dada uma solução, a escolha da solução vizinha é efectuada pela escolha aleatória de um período de tempo e uma aula no período seleccionado, e movendo a aula para um período diferente, escolhido de forma aleatória. A função objectivo *f*, a ser minimizada, é uma soma pesada do número de conflitos entre turmas, entre docentes e entre salas. [Abramson, 1991] efectuou um estudo comparativo de diversas taxas de arrefecimento (utilizando como critérios de terminação um número de iterações da ordem dos três milhões), concluindo que o melhor valor era de 0.9.

O *Arrefecimento Simulado* apresenta como principais vantagens a sua simplicidade e robustez [Kirkpatrick et al., 1983] [Faria, 2000]. Exige no entanto longos tempos de computação, comparativamente a outros métodos, de forma a permitr um arrefecimento lento e a obtenção de boas soluções.

6.4.4.2 Pesquisa Tabu

A *Pesquisa Tabu* [Glover e Laguna, 1993] é uma técnica de pesquisa local que funciona mantendo uma lista das últimas n soluções testadas ou das última alterações efectuadas à solução (conhecida como *Lista Tabu*). O processo de escolha de uma alternativa da vizinhança passa pela avaliação da qualidade de cada opção que é considerada escolhendo-se a opção mais vantajosa, mesmo que a qualidade seja inferior à situação actual. Se a nova solução se encontrar na *Lista Tabu*, então é considerada *Tabu* e opta-se pela segunda melhor opção e assim por diante.

A *Pesquisa Tabu* tenta "escapar" aos mínimos locais, escolhendo para solução seguinte, a melhor solução x, pertencente a $V(x_n)$ (x_n é a solução corrente), ou a uma sub-vizinhança $V'(x_n) \subseteq V(x_n)$, se $V(x_n)$ for demasiado grande para ser totalmente explorada, mesmo que seja pior que a solução corrente, isto é, mesmo que $F(x) > F(x_n)$. Existe a possibilidade de

a pesquisa entrar em ciclo, isto é, ao fim de i iterações, x_n tornar a ser a solução corrente. A partir daqui, o algoritmo seria incapaz de evoluir noutro sentido que não fosse repetir indefinidamente o percurso que o traz de volta a x_n. Para impedir a ocorrência de ciclos, guarda-se na *Lista Tabu* as soluções já visitadas (embora esta seja uma estratégia muito redutora, uma vez que pode haver interesse em voltar a soluções já visitadas e explorar novas direcções de pesquisa). Pode-se também limitar o tamanho da *Lista Tabu* às últimas k soluções visitadas, eventualmente com k variável ao longo da pesquisa. Consegue-se assim, evitar ciclos de tamanho não superior a k e permite-se voltar a soluções já visitadas, ao fim de k iterações (sempre que uma nova solução é visitada, entra para a *Lista Tabu*, saindo a mais antiga). Mesmo assim, podem ocorrer ciclos, havendo necessidade de um critério de paragem, baseado na melhoria da função objectivo nas últimas N iterações, ou num número total, pré-determinado, de iterações, para garantir que o algoritmo termina.

É possível tornar a *Lista Tabu* menos restritiva, mantendo a sua função de evitar ciclos na pesquisa. O que se pretende evitar não é o tornar a visitar uma solução já visitada, mas sim não tornar a efectuar o movimento da passagem da solução x_n, para a solução x. A alternativa será, a *Lista Tabu* conter, não as soluções já visitadas (x_n), mas sim os pares de soluções (x_n, x) que estão relacionados com os movimentos já efectuados. Desta forma, a pesquisa tabu pode ser vista como um método de pesquisa, com vizinhanças de tamanho variável (em cada iteração a vizinhança é redefinida de modo a excluir as soluções que obrigariam a movimentos *Tabu*). Uma *Lista Tabu*, de pares de soluções, apresenta problemas de ordem prática aquando da sua implementação. Normalmente, não é viável: manter uma descrição completa dos pares de soluções correspondentes a movimentos tabu; verificar se um par de soluções pertence à lista tabu (torna-se demasiado demorado). A alternativa passa por atribuir um estado *tabu*, não aos pares de soluções (x_n, x), mas sim a algum atributo ou característica da transformação da solução x_n, na solução x (variáveis que mudam de valor, elementos que trocam de posição, elementos que mudam de um conjunto para outro, etc.), isto é, às m modificações que transformam a solução x_n, na solução x.

A atribuição do estado *Tabu* às modificações, em vez das soluções, é demasiado restritivo[31]. Dever-se-á utilizar um mecanismo que permita passar por cima do estado *tabu* de uma certa modificação, através do que se designa por *nível de aspiração* (se a solução gerada, a partir de uma modificação *Tabu*, é suficientemente boa, então ela é aceite). Um *nível de aspiração* evidente, e que usualmente faz parte de qualquer implementação de algoritmos de *Pesquisa Tabu*, é que a solução gerada seja melhor, que a melhor solução encontrada até ao momento.

---

[31] Exemplo: se o movimento de uma solução (uma sequência de valores) para outra é gerado trocando os elementos das posições 2 e 7, será inserida na *Lista Tabu* a modificação que consiste em trocar de posição os elementos 2 e 7, independentemente da sequência (solução) de partida ou de chegada.

Quanto maior for a *Lista Tabu*, melhor se evita que o algoritmo entre em ciclo, mas tal, envolve maior tempo de processamento para comparar as diferentes soluções. A utilização da *Lista Tabu* permite ao algoritmo "fugir" de óptimos locais e desenvolver uma pesquisa mais global do espaço de solução. Em problemas de geração de horários, uma solução vizinha pode ser determinada considerando o movimento de um único evento para um novo período ou trocando dois eventos de período. Como o tamanho da vizinhança, utilizando este critério, pode ser enorme em problemas práticos de geração de horários, somente uma fracção da vizinhança é calculada, proporcional ao número de eventos.

De acordo com o tamanho de representação de uma solução, e desta forma o tempo envolvido na comparação de dois horários, o estado completo de soluções previamente visitadas não pode ser guardado na *Lista Tabu*. Em vez disso, é usual a *Lista Tabu* ser armazenada em termos de um conjunto de pares (evento, período), representando a posição dos eventos anteriores a um dado movimento de vizinhança. De seguida, não é permitido ao evento ser escalonado no período dado até o par (evento, período) sair da *Lista Tabu*. Enquanto isto, inevitavelmente, acelera o processo, o que é muito importante em grandes problemas, por outro lado nega alguns movimentos que resultarão num estado prévio ainda não visitado (pois os eventos não podem ser reescalonados num período de tempo que ainda esteja na *Lista Tabu*).

Na pesquisa a efectuar não existem quaisquer garantias de convergência, mesmo que teóricas (ao contrário da pesquisa por arrefecimento simulado). O algoritmo limita-se a explorar algumas regras simples para escapar aos mínimos locais, procurando imitar o comportamento humano. Apesar da falta de fundamentação teórica, a pesquisa Tabu tem sido aplicada com sucesso na resolução de numerosos problemas de optimização combinatória. A característica mais importante centra-se na utilização da memória (distingue-a de uma pesquisa local simples). O passado, recente ou mais afastado, vai influir de forma determinante no processo de pesquisa, guiando-a em função das soluções já visitadas.

O conjunto de decisões a tomar para implementar um algoritmo de *Pesquisa Tabu* é maior do que o equivalente num algoritmo de Pesquisa por *Arrefecimento Simulado*. Estas decisões são menos normalizadas e menos suportadas por resultados teóricos.

De realçar que este algoritmo é muito pouco baseado em factores aleatórios (ao contrário do Arrefecimento Simulado, que utiliza descida aleatória, múltiplos começos, entre outros), mas sim no pressuposto de que a pesquisa inteligente constitui uma melhor forma de orientação [Glover e Laguna, 1993]. A Pesquisa Tabu depende grandemente das estruturas de memória utilizadas para representar os seus conceitos e exige um maior esforço de "engenharia", na sua modelação e parametrização.

6.4.4.3 Algoritmos Genéticos

Algoritmos genéticos são uma ferramenta de optimização, baseada na teoria da evolução de Darwin [Holland, 1975] [Davis, 1991]. Têm a capacidade de produzir soluções de boa qualidade mesmo quando as dimensões do problema são elevadas. Por esta razão, os algoritmos genéticos têm vindo a ser aplicados com sucesso a uma enorme variedade de problemas [Newall, 1999].

Estes algoritmos operam sobre uma população de soluções representada por alguma codificação. Cada membro da população consiste num número de genes, sendo cada qual uma unidade de informação. Novas soluções são obtidas pela combinação de genes a partir de diferentes membros da população para produzir descendentes (cruzamento) ou pela alteração de membros da população (mutação). Uma simulação de uma "selecção natural" tem lugar de seguida, pela avaliação da qualidade de cada solução, seleccionando as que apresentam melhores condições de sobrevivência para a próxima geração. Os *Algoritmos Genéticos* lidam com populações em vez de soluções individuais: as soluções interactuam, cruzam-se e produzem filhos, dos quais se espera que retenham as boas características dos seus pais. Baseiam-se na pesquisa local, sendo explorada a vizinhança de uma população inteira. Estes algoritmos não operam directamente sobre o espaço das soluções: estas têm de ser codificadas como *strings* de tamanho finito, sobre um alfabeto finito (nem sempre a codificação mais evidente é a que conduz a melhores resultados).

Os *Algoritmos Genéticos* começam com uma população inicial com N soluções e evoluem gerando sucessivas populações da mesma dimensão, em cada iteração. São utilizados operadores (selecção / cruzamento / mutação) para a geração, exploração da vizinhança de uma população e selecção de uma nova geração. A codificação do algoritmo passa pela descrição de um indivíduo (solução) da população, designado por cromossoma. Cada posição da *string* que constitui o cromossoma é designada por gene, que pode tomar um valor de entre um alfabeto finito, pré-determinado pela codificação utilizada. Originalmente, apenas se consideravam codificações binárias (cada gene tomava o valor 0 ou 1). No entanto, implementações mais recentes utilizam um conjunto mais alargado de símbolos.

A obtenção da geração $(n+1)$ a partir da geração n ($X^{(n)}$) processa-se da seguinte forma: os melhores indivíduos (soluções) de $\mathbf{X}^{(n)}$ são seleccionados; são efectuadas operações de cruzamento sobre pares desses indivíduos, gerando-se descendentes dessas soluções; estes descendentes irão substituir os "maus" indivíduos da população corrente; a uma pequena parte dos filhos é efectuada uma mutação (alterando ligeiramente de forma aleatória as suas características).

O processo de avaliação dos indivíduos (soluções) é realizado de forma a que cada solução da população corrente seja avaliada pela sua adaptação ao meio envolvente (*fitness*). Esta pode, simplesmente, ser o valor da função objectivo do problema (maximização, por exemplo). A selecção dos indivíduos de uma população é baseada na sua "adaptação",

mas não de uma forma determinística: as x_i soluções são retiradas aleatoriamente, com reposição da população corrente, com uma probabilidade ***pi***, que aumenta com a sua "adaptação". O cruzamento (*"crossover"*) é realizado utilizando pares formados pelos indivíduos seleccionados (com determinada probabilidade). Existem diversas possibilidades de definir o operador *crossover*, sendo o mais comum o *crossover* de 2 pontos. Os "maus" indivíduos são também, aleatoriamente seleccionados, de acordo com a sua "adaptação", mais correctamente, com a sua "desadaptação", de uma forma semelhante à da de selecção dos "bons" indivíduos, mas com uma probabilidade decrescente com o aumento da sua "adaptação". Normalmente, o algoritmo termina após a geração de um número pré-definido de novas populações.

Os Algoritmos Genéticos têm sido aplicados a diversos problemas, nomeadamente ao Problema da Construção de Horários. Um exemplo de aplicação a este problema é apresentado por Colorni et al. [Colorni et al., 1992]. No seu trabalho, horários inviáveis (que não respeitam as restrições globais do problema) são considerados no espaço de pesquisa deste tipo de algoritmos. A função objectivo engloba um factor correspondente ao número de violações das retrições rígidas e restrições flexíveis. De forma a desviar a pesquisa no sentido da obtenção de horários viáveis, é atribuído um grande peso às violações das restrições rígidas consideradas na referida função objectivo. Uma solução é representada através de uma matriz M_{mxp}, tal que a linha i de M representa o horário do docente t_i. Em particular, cada entrada m_{ik} contém o nome da turma que o docente encontra no período k. O operador de *crossover* é aplicado a dois horários T_1 e T_2, da seguinte forma: existe uma função local de adaptação (*fitness*) que calcula a adaptação do horário de um docente em particular. As linhas de T_1 são ordenadas de forma a decrescer a adaptação local e as k melhores linhas são colocadas juntas com as outras m-k linhas retiradas de T_2. A segunda descendência é obtida a partir das linhas inutilizadas de T_1 e T_2. O valor de k é determinado com base na adaptação local de ambos T_1 e T_2. O operador de mutação utiliza h genes contíguos e troca-os com outros h genes contíguos não sobrepostos, pertencentes à mesma linha. O algoritmo inclui também uma fase de pesquisa local final que move a solução gerada para o seu óptimo local.

Na aplicação de Algoritmos Genéticos é de grande importância a definição correcta dos seus parâmetros, como sejam: o tamanho da população; a taxa de substituição de soluções (dos pais pelos filhos); a probabilidade de *crossover*; a probabilidade de mutação e a função de avaliação das soluções (que permite seleccionar as melhores soluções). Existem também escolhas estruturais que é necessário efectuar, nomeadamente: codificação das soluções; operador de *crossover*; operador de mutação. Existem alguns cuidados importantes que são necessários, como: manter a diversidade da população; explorar as vizinhanças de diversos máximos locais e não apenas de um deles; escolher criteriosamente a codificação e operadores, para posterior afinação cuidadosa dos parâmetros do algoritmo.

Algumas dificuldades na implementação de algoritmos genéticos clássicos, aplicados ao problema de construção de horários e a facilidade com que estes algoritmos podem ser conjugados (tornam-se em algoritmos híbridos) com outros métodos pesquisa, leva a que a maioria das aplicações de geração de horários utilizando algoritmos genéticos, inclua a utilização simultânea ou posterior de outros métodos de resolução de problemas de optimização.

6.4.5 Programação em Lógica

A Programação em Lógica ganhou notoriedade dentro da comunidade científica desde meados dos anos setenta, altura em que surgiu o *PROLOG* (***Programation et Logique***), originário das ideias de Colmerauer [Colmerauer, 1975], em Marselha e Kowalski [Kowalski, 1979], em Edimburgo. Esta linguagem depressa se tornou naquela que melhor representava a nova classe de linguagens declarativas. A programação em *PROLOG* difere da programação convencional, na medida em que utiliza uma lógica declarativa na formulação dos problemas e a dedução para a sua resolução.

Um programa em Prolog pode ser subdividido em dois componentes, que podemos designar por competência e desempenho, ou lógica e controlo [Kowalski, 1979]. O primeiro componente encontra-se relacionado com a descrição da informação factual intimamente ligada ao problema em si, isto é, relações e declarações que devem ser manipuladas e combinadas, no sentido da obtenção de soluções. O segundo componente, como o próprio nome indica, lida com a estratégia e o controlo da manipulação das relações anteriormente referidas. Desta forma, conclui-se que a lógica é responsável pela consistência do programa, enquanto que o controlo é responsável pela sua eficiência. Uma metodologia ideal deverá, primeiramente resolver a competência do programa (ou seja, o quê) e só depois o desempenho (ou seja, o como).

Kang e White [Kang e White, 1992] propuseram a utilização da programação em lógica na resolução do problema da construção de horários. Utilizaram o *PROLOG* como a linguagem de implementação do seu programa de construção de horários. A principal vantagem desta abordagem reside na possibilidade de expressar, de uma forma declarativa, as restrições envolvidas no problema. A capacidade total de "*backtracking*" oferecida pelo *PROLOG* sobrepõe-se a heurísticas que permitem apenas, de forma limitada, o reescalonamento de atribuições que criam conflitos. Dessa forma, quando não é possível escalonar uma aula, o procedimento procura uma aula equivalente, já escalonada, e coloca-a num período diferente. No caso de não existir uma aula equivalente que possa ser mudada para um período diferente, então é colocada numa lista de aulas as quais serão escalonadas manualmente, mais tarde.

Os principais problemas da *Programação em Lógica* em geral e da linguagem *PROLOG* em particular encontram-se relacionados com a sua dificuldade em lidar com objectos pertencentes a domínios inteiros, reais ou racionais, e com o simples mas ineficiente

método de pesquisa standard do *Prolog*: a pesquisa em profundidade com *backtracking*. Este mecanismo de controlo é basicamente um método de geração e teste, cuja performance decresce largamente em problemas de grande dimensão. Desta forma a utilização de *Programação em Lógica* na resolução de problemas de optimização combinatória tais como a geração de horários tem vindo a ser substituída pela *Programação em Lógica com Restrições* que combina a declaratividade do *Prolog* com a eficiência do mecanismo de resolução de restrições.

6.4.6 Programação em Lógica com Restrições

A *Programação em Lógica com Restrições (PLR)* (*Constraint Logic Programming – CLP*) [Jaffar e Lassez, 1987], surgiu numa tentativa de ultrapassar as referidas limitações de linguagens de *Programação em Lógica*. A sua ideia básica consiste em complementar o núcleo computacional do sistema lógico, a unificação, com um manipulador de restrições, num determinado domínio [Azevedo e Barahona, 1994]. Por se basear em restrições, a PLR sofreu a influência da investigação proveniente de um paradigma bastante mais antigo, denominado Problema de Satisfação de Restrições (PSR). Esta influência resultou num melhoramento significativo do processo de pesquisa do sistema lógico básico, presente na programação lógica.

Uma restrição é uma relação lógica entre um conjunto de variáveis, cada uma tomando um valor de um determinado domínio, ou seja, uma restrição limita os valores possíveis que uma variável pode tomar. Desde os anos sessenta e setenta [Waltz, 1972] que os PSR aparecem em diversas áreas como a inteligência artificial e a investigação operacional. Exemplos disto são as áreas da visão, alocação de recursos, escalonamento de tarefas, construção de horários e raciocínio temporal [Reis, 1997].

Um PSR é um problema que tem por dados um conjunto finito de variáveis, cada uma com o seu domínio finito associado, e um conjunto de restrições que limitam a combinação de valores que um conjunto de variáveis pode tomar simultaneamente [Guerét et al., 1995]. Uma solução para um PSR é a atribuição de valores, a todas as variáveis, respeitando as restrições e o domínio das variáveis.

A PLR é a fusão dos dois paradigmas apresentados: a programação em lógica e a satisfação de restrições. A ideia básica consiste em complementar o núcleo computacional do sistema lógico e a unificação, com um manipulador de restrições, num determinado domínio. Este esquema foi proposto, pela primeira vez, por [Jaffar e Lassez, 1987] e foi denominado {[PLR(x)] [CLP(x)]}. A variável x pode tomar vários valores, que representam outros tantos domínios de computação. Exemplos destes domínios são: R – para reais, Q – para racionais, FD – para domínios finitos e B – para *booleanos*, entre outros.

Embora seja um campo de investigação relativamente recente, a PLR tem progredido ao longo dos últimos anos, em diversas direcções distintas. Os seus conceitos iniciais têm sido adaptados de forma a servir as mais diversas aplicações [Jaffar e Lassez, 1994].

[Yoshikawa et al., 1994] propuseram a utilização de um solucionador do problema de construção de horários, baseado na relaxação de restrições. Neste tipo de abordagem, é atribuída uma determinada penalidade a cada restrição e o objectivo é encontrar uma atribuição das variáveis do problema, que minimizem a penalidade total. A linguagem de restrições permite ao utilizador expressar diversos tipos de restrições (por exemplo, definir as indisponibilidades de um determinado docente). O método combina um algoritmo, que pesquisa uma solução inicial, com um procedimento tipo *hill-climbing*, a utilizar na fase de optimização.

Reis e Oliveira [Reis e Oliveira, 1999a] [Reis e Oliveira, 1999b] [Reis e Oliveira, 2000b] propuseram um método de resolução automática de problemas de horários baseado numa definição completa das restrições rígidas e flexíveis do problema. Apresentam diversas formulações para problemas de geração de horários e calendarização de exames e utilizam o sistema de PLR *ECLiPse* na resolução dos mesmos. Muito bons resultados foram obtidos utilizando a PLR na resolução de problemas de calendarização de exames reais [Reis et al., 1999] [Reis e Oliveira, 2000b].

A evolução da PLR deve-se ao poder de resolução que estas linguagens têm, para aqueles problemas combinatórios extremamente complexos, encontrados, por exemplo, no escalonamento da produção, calendarização, afectação ou transportes.

6.4.7 Sistemas Periciais

Os problemas de escalonamento são habitualmente de resolução bastante complexa. Muitas organizações resolvem os seus problemas específicos, utilizando peritos humanos, que possuem grande experiência e intuição na resolução dos seus próprios problemas específicos. Muitas vezes, as restrições num problema alteram-se de forma tão frequente, que chegar a uma solução é suficiente, em vez de tentar obter uma solução óptima. Por outro lado, os especialistas na resolução destes problemas utilizam heurísticas e regras práticas, as quais são, normalmente, muito difíceis de formalizar matematicamente. Mesmo quando é tentada uma formulação analítica do problema (como, por exemplo, no mapeamento do problema de calendarização de exames para um problema de coloração de grafos), muitas restrições complexas não são formalizadas e somente restrições muito simples são usadas. Nos casos em que uma solução não é encontrada, os peritos muitas vezes utilizam o julgamento humano no relaxamento de algumas das restrições, algo que não é possível na formulação analítica. Peritos humanos, com pleno conhecimento do sistema que utilizam, podem efectuar pequenas alterações pontuais, evitando dessa forma que o sistema tenha de refazer todo o processo.

Um Sistema Pericial é uma aplicação computacional que representa e raciocina com conhecimento especializado, tendo por objectivo dar conselhos ou resolver problemas de forma semelhante à que o faria um especialista humano no domínio. O Sistema Pericial tenta simular o raciocínio humano, raciocinando sobre uma representação dinâmica do conhecimento humano, sendo esta representação expressa em algum tipo de linguagem específica para o efeito e mantido na denominada Base de Conhecimento. Apresentam também capacidades de explicação e justificação dos resultados obtidos, ou recomendações, no sentido de, sendo utilizados como assistentes, convencer o utilizador que as soluções encontradas são realmente correctas. O Sistema Pericial deve também ser capaz de interagir com o utilizador de forma amigável: "Capacidades dedutivas, alteração interactiva dos conhecimentos, explicabilidade, linguagem amigável de interacção, são algumas das características fundamentais dos Sistemas Periciais" [Oliveira, 1984].

Gudes et al. [Gudes et al., 1990] afirmam que a principal ênfase a dar num sistema pericial é a generalidade. Nos seus trabalhos é permitida uma forma muito geral de restrições, e as suas estratégias de controlo e *backtracking* podem ser controladas por regras e meta-regras. A grande vantagem da utilização de *backtracking* é a de permitir alterações locais, que são uma necessidade de todos os sistemas da vida real. Finalmente, os problemas de alocação de recursos são conhecidos por serem impossíveis de resolver pela máquina, através de uma pesquisa exaustiva. Isto é verdade mesmo se o problema lida com um número relativamente pequeno de recursos e períodos de tempo, como é o caso dos horários universitários.

Uma aplicação de *Sistemas Periciais* à Geração de Horários foi realizada por Reis e Rocha [Reis e Rocha, 1993]. No seu trabalho definiram um sistema baseado em conhecimento capaz de efectuar a geração de horários para o segundo ciclo do ensino básico português e apresentar explicações das alocações efectuadas. O sistema era também capaz de efectuar realocações de aulas e apresentar as respectivas explicações.

6.4.8 Sistemas Interactivos

Nos *Sistemas Interactivos* de *Construção de Horários* existe interactividade entre o sistema e um utilizador, que terá de tomar diversas decisões de forma a construir os horários pretendidos [Faria, 2000]. Ou seja, para que o referido utilizador tome as melhores decisões possíveis (de entre as centenas ou milhares que tem de tomar), deverá ser apoiado pelo sistema que o conduzirá nesse sentido. Como tal, compreende-se que os *Sistemas Interactivos* sejam baseados em *Sistemas de Apoio à Decisão.*

Segundo Gorry e Morton [Gorry e Morton, 1971], um *Sistema de Apoio à Decisão* pode ser definido como: "A mistura eficaz de inteligência humana, tecnologia da informação e *software*, que interactuam de uma forma chegada, para resolver problemas complexos.". Para Turban e Meredith [Turban e Meredith 1994], um *Sistema de Apoio à Decisão* pode ser definido como: "Um sistema informático interactivo, que utiliza regras de decisão e

modelos associados a uma base de dados "completa" e aos pontos de vista do decisor, conduzindo a decisões específicas e implementáveis, para a resolução de problemas não tratáveis pelo recurso exclusivo a modelos de optimização". Finalmente, segundo Laudon e Laudon [Laudon e Laudon, 1997], um *Sistema de Apoio à Decisão* pode ser definido como: "Sistema interactivo controlado pelo utilizador, que combina informação, modelos analíticos sofisticados e *software* de fácil utilização, num único sistema que apoia o utilizador (decisor) no processo de decisões semi-estruturadas e não estruturadas".

Um *Sistema de Apoio à Decisão* é composto por 3 componentes: Sistema de Gestão da Base de Dados (a informação a utilizar é normalmente extraída de bases de dados relevantes); O Sistema de Gestão da Base de Modelos (contendo modelos de resolução do problema completo ou de partes do problema); O Sistema de Gestão da Interface Gráfica com o Utilizador (que proporciona a facilidade de interacção entre o seu utilizador, a sua base de dados e a sua base de modelos).

Os *Sistemas de Apoio à Decisão* proporcionam apoio a decisões interdependentes ou sequenciais, acompanhando, se necessário, todo o processo de tomada de decisão. São capazes de testar diferentes estratégias, rápida e objectivamente (análise "*what-if*"). O decisor tem controlo absoluto sobre todos os passos do processo de tomada de decisão de um problema, pois o objectivo destes sistemas é apoiar, e não, substituir o decisor. Desta forma, é possível perceber que quando se fala de *Sistemas Interactivos*, tal é indissociável do conceito de *Sistemas de Apoio à Decisão*. Estes sistemas permitem a um utilizador construir o seu próprio sistema, tornando-o personalizado e intuitivo, e como tal, encará-lo como uma extensão do conhecimento do respectivo utilizador.

Grande parte dos Sistemas comerciais de Geração de Horários são essencialmente Sistemas de Apoio à Decisão com alguns métodos (usualmente muito simples tecnicamente) de geração automática de horários. De entre estes destacam-se o *GP-Untis* [GP-Untis, 2001], *Mimosa* [Mimosa, 2001], *ADE* [ADE, 2001], o *GestHor* [GestHor, 2001] e o *Celcat* [Celcat, 2001]. Estes sistemas comerciais serão objecto de uma análise mais detalhada na secção 6.5.

6.4.9 Sistemas Multi-Agente

Os problemas de escalonamento da gestão universitária reais são resolvidos na prática de forma distribuída, utilizando o conhecimento e informação de diversos agentes presentes no processo educativo. Estes agentes encontram-se na prática a diversos níveis hierárquicos. No nível superior pode ser identificado o reitor da universidade que interactua essencialmente na sua actividade, com os directores das diversas faculdades ou departamentos da universidade. No nível imediatamente inferior, os directores de departamento interactuam com os directores de curso, directores de áreas científico-pedagógicas (ou secções) e eventualmente com os directores de recursos. Os directores de área (secção), comunicam no nível inferior com os docentes dessa área (secção) enquanto

os directores de curso comunicam com representantes das turmas e docentes que leccionam as disciplinas desses cursos.

Para modelizar a resolução de problemas de escalonamento como um SMA, vários outros agentes podem ainda ser identificados, como por exemplo, agentes escalonadores tais como agentes para a distribuição de serviço docente, geração de horários ou calendarização de exames. Estes agentes possuem capacidade de resolução dos problemas em causa, podendo interactuar com os agentes docentes, turmas e directores, de forma a atingir uma resolução global do problema.

Embora as metodologias baseadas em agentes sejam adequadas à resolução de problemas de escalonamento universitários, poucos trabalhos de investigação foram publicados até hoje nesta área. Destaca-se o trabalho de Silva [Silva, 2000a] que tratou o problema da geração de horários utilizando uma abordagem baseada num sistema Multi-Agente. O sistema construído (GHES – Gerador de Horários no Ensino Secundário) baseia-se na representação das entidades envolvidas na geração de horários através de agentes de *software* [Silva, 2000a]. Foi desenvolvido utilizando a linguagem Java baseada na plataforma JINI [JINI, 2001] para implementar a comunicação.

Os agentes computacionais utilizados representavam turmas, docentes e salas. Cada agente possui uma arquitectura que pretende reflectir, tanto quanto possível, as capacidades dos agentes reais neste processo. Cada agente individual tem objectivos, restrições individuais e comportamentos parametrizáveis que condicionam a geração dos horários pela comunidade de agentes.

No sistema GHES, todos os agentes são cooperativos, relaxando sucessivamente as suas restrições com o objectivo de cooperação global. No entanto, estão também interessados na obtenção de horários que satisfaçam os objectivos, as impossibilidades e respeitem tanto quanto possível, as preferências próprias neles delegadas. A coordenação global do sistema é descentralizada e os agentes negoceiam a atribuição dos recursos através da troca de mensagens entre si [Silva, 2000a]. Cada agente avalia os resultados da negociação baseado em custos de utilização dos recursos num determinado espaço de tempo. Utilizando o sistema foram analisados problemas fictícios de reduzida e média dimensão, concluindo-se que é possível resolver problemas de geração de horários utilizado um SMA.

6.5 Sistemas Comerciais de Geração de Horários

Dada a complexidade do problema de geração de horários e problemas associados e o elevado número de instituições de ensino que o têm de resolver anualmente, um enorme número de empresas de desenvolvimento de *software* foram atraídas para o desenvolvimento de aplicações comerciais para efectuar a sua resolução. Desta forma, existem imensas aplicações comerciais que permitem efectuar a geração automática ou

assistida de horários e por vezes resolver alguns dos problemas associados tais como a alocação de salas e a calendarização de exames.

Os pacotes de *software* disponíveis no mercado nacional e internacional para a resolução deste tipo de problema diferem significativamente, quer a nível dos seus objectivos quer a nível das suas potencialidades. Os objectivos diferem de aplicação para aplicação: alguns direccionados para instituições de ensino superior, outros para instituições do ensino básico/secundário e ainda alguns destinados às empresas. Alguns dos pacotes de software são essencialmente ferramentas de apoio à decisão enquanto outros são ferramentas de geração automática de horários. A maioria das aplicações existentes no mercado apresenta pelo menos um processo semi-automático de construção de horários. A apresentação gráfica das aplicações é diversa, dependendo da metodologia adoptada na construção dos horários.

Neste sentido foi desenvolvido um estudo sobre algumas aplicações comerciais de geração de horários. As aplicações seleccionadas obedeceram a cinco critérios de escolha:

- **Implantação no mercado internacional.** Foram seleccionadas algumas das ferramentas mais utilizadas internacionalmente em Escolas e Universidades;
- **Implantação no mercado nacional.** Seleccionaram-se as ferramentas mais utilizadas em escolas e instituições de ensino superior nacionais;
- **Produtos nacionais.** Pacotes de *software* desenvolvidos por investigadores ou empresas nacionais;
- **Qualidade da investigação.** Foram seleccionados programas com qualidade, i.e. que demonstrem através das suas potencialidades, a realização de investigação de qualidade para o seu desenvolvimento;
- **Disponibilidade de informação.** Infelizmente a informação relativa a alguns pacotes de *software* que obedecem a uma parcela significativa dos quatro critérios anteriores, é extremamente escassa. Desta forma foram analisados primariamente pacotes de *software* cujos produtores disponibilizam informação detalhada.

As aplicações seleccionadas foram estudadas de acordo com diversos critérios, incluindo a resolução do problema a que se destinam (geração de horários em escolas ou universidades, calendarização de exames, etc.), as capacidades de introdução e validação de dados, construção de horários (de forma automática e assistida), restrições rígidas e flexíveis consideradas, visualização de horários, impressão de horários e auxílio *on-line*.

Das aplicações analisadas, utilizando os critérios de selecção referidos anteriormente, destacam-se cinco das mais utilizadas a nível nacional e internacional:

- ***Mimosa***: O *Mimosa* é um programa de geração de horários de origem Finlandesa [Mimosa, 2001]. Destina-se a todo o tipo de instituições com necessidades de

escalonamento e criação de horários, podendo inclusivamente ser usado na programação de conferências. Foi desenvolvido utilizando a linguagem *Borland Delphi*. Existe no mercado desde 1986 e encontra-se disponível unicamente para o sistema Windows, sendo usado em todos os continentes. Trata-se de uma aplicação com utilização internacional que usa o Inglês como idioma original. Apesar disto, os utilizadores podem mudar o texto dos relatórios gerados e tudo o que se imprime, para qualquer idioma nativo. Esta aplicação distingue-se pela sua funcionalidade multi-utilizador, permitindo mais de 255 utilizadores simultâneos, efectuando a validação dos ficheiros usados. É ainda compatível com qualquer rede, suportando conexões telefónicas com *SMS WAP* (*Mimosa para WAP*). Informações sobre os algoritmos em que se baseiam os métodos de atribuição automática de recursos não estão disponíveis. A empresa unicamente indica que grande parte das abordagens é realizada através de heurísticas.

- ***Celcat***: O *CELCAT* existe no mercado desde 1979 [Celcat, 2001]. É comercializado pela empresa com a mesma denominação, sediada em Inglaterra. Nesta análise foi utilizada a versão 5.6 deste sistema. O *CELCAT* é sobretudo usado nos países Anglo-Saxónicos (Austrália, Canadá, EUA e Reino Unido), sendo inclusivamente, a ferramenta de construção de horários mais usada em Inglaterra[32]. Este produto destina-se à resolução de todo o tipo de necessidades de escalonamento e criação de horários em interactividade com o utilizador. Para o utilizador final a linguagem é o inglês. Tecnicamente, o programa foi desenvolvido em *Borland C++* e está disponível para a plataforma *MS-Windows*. O produto está disponível comercialmente em três versões *Standard* (normal), *Pro* e *Viewer* (visualização). A versão normal permite a leitura, escrita e alteração de dados. A versão *Pro* distingue-se pelas funcionalidades de rede permitindo o acesso multi–utilizador com diferentes níveis de acesso à informação. A versão Viewer permite efectuar exclusivamente a leitura e visualização dos dados. Infelizmente a informação disponibilizada pela empresa e as versões de demonstração não permitem efectuar um estudo mais aprofundado das diferenças entre as três versões. A empresa disponibiliza também, um serviço (*TurnKey*) que consiste, em eles próprios construírem os horários para os clientes. Este serviço permite à empresa que o adquira, a libertação de recursos humanos, poupança de tempo e investimento em software.
- ***ADE***: O *ADE* é um sistema de apoio à decisão construído em França, flexível e eficaz, que possui também ferramentas de construção automática de horários [ADE, 2001]. Este produto destina-se a todo o tipo de necessidades de

[32] Seguindo a informação dos folhetos publicitários disponibilizados pela empresa e disponível em diversos sites na Internet.

escalonamento e criação de horários em interactividade com o utilizador. É utilizado por instituições como a France Telecom, Royal Air Force, Marinha Francesa, entre outras. Para o utilizador final a linguagem é o inglês ou opcionalmente o francês. Permite a instalação em rede e o acesso multi–utilizador com diferentes níveis de acesso à informação. Os métodos de atribuição automática de recursos baseiam-se em meta heurísticas, como o algoritmo de "arrefecimento simulado". A utilização do programa é bastante simples e intuitiva existindo basicamente duas fases: entrada de dados e restrições e geração de horários.

- ***Gp-Untis***: É um produto concebido pela empresa *Gruber & Petters-Austria* [GPUntis, 2001], cuja primeira versão tem já mais de 30 anos. É actualmente usado por mais de 15.000 estabelecimentos de ensino espalhados por todo o mundo, com particular incidência na Europa [GP-Units, 2001], e por cerca de 400 estabelecimentos em Portugal [Chavarria, 2002]. A nível nacional bem como em todos os países de língua oficial portuguesa, a Cardita – Análise Informática, Lda possui o exclusivo da sua distribuição. Este produto destina-se a todas as instituições com necessidades de elaboração de horários, desde o ensino básico e secundário ao ensino superior, podendo haver adaptações para outras instituições específicas. Possui versões em diversas línguas, incluindo o Alemão, Espanhol, Inglês e Português. Todos os anos é lançada[33] uma nova versão do programa com novas funcionalidades e melhorias do desempenho de rotinas já existentes, baseada nas conclusões dum seminário que anualmente se realiza na Áustria [GPUntis, 2001], com todos os parceiros interessados e onde se dá conta das necessidades mais prementes dos utilizadores. O ajustamento do programa à realidade portuguesa tem sido efectuado com o colaboração de personalidades, desde sempre ligadas à elaboração de horários e com o contributo de Escolas Piloto que, normalmente, com um ano de antecedência, utilizam o programa, com a finalidade de o testar, despistar eventuais erros, e também colaborar nas acções de formação e promoção. O produto está disponível comercialmente numa versão com vários módulos opcionais. Tecnicamente o programa é desenvolvido em *MS-Visual C++*, para o sistema *Windows*. O programa inclui algoritmos de geração automática de horários baseados em meta heurísticas, nomeadamente algoritmos genéticos. Durante o cálculo o programa informa sobre o estado da geração de horários em três janelas: qualidade dos horários; erros nos dados e advertências úteis para mudar os dados básicos e os horários.
- **GestHor:** Trata-se de uma aplicação desenvolvida em Portugal [GestHor, 2001] específica para o ensino português, que possui por defeito os programas

[33] Informação publicitária disponibilizada pela empresa.

curriculares oficiais do ensino básico e secundário nacionais, mas que possui também a possibilidade de alterar ou estender os programas curriculares. É um produto da empresa *Cronológica* e está disponível comercialmente em quatro versões distintas, quer para o ensino secundário quer para o ensino universitário: *GestHor* e *GestHor Pro*. Estas versões diferem essencialmente nos algoritmos adoptados para a resolução do problema. A versão Standard utiliza o algoritmo *"Iterative Hill-Climbing"*, enquanto a versão *Pro* distingue-se pela utilização dos algoritmos *"Iterative Hill-Climbing"*, *"algoritmo A*"* e algoritmos genéticos.

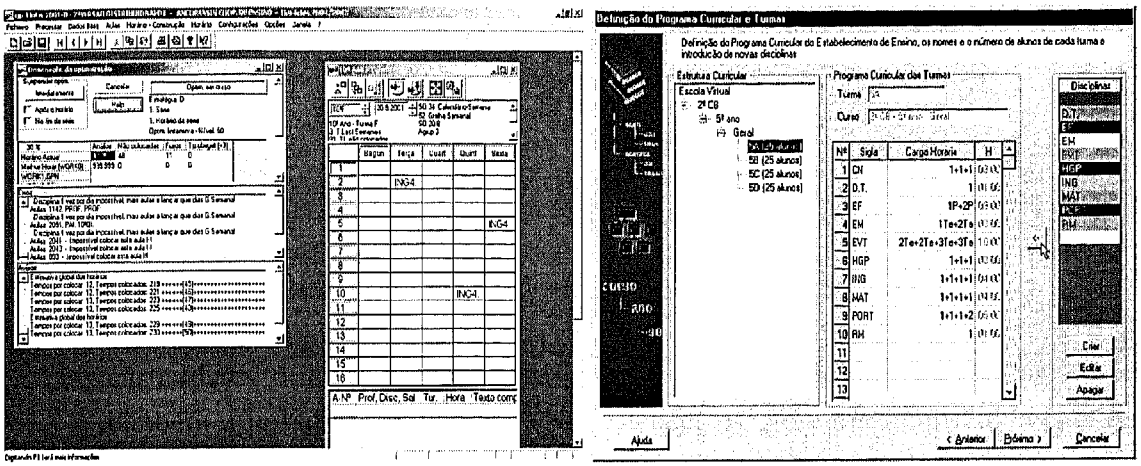

Figura 37: Janela de visualização da geração automática do GPUntis (esquerda) e definição de planos curriculares do GestHor (direita)

Na tabela seguinte é apresentado um comparativo entre as principais funcionalidades dos sistemas analisados:

Critérios	Mimosa	Celcat	ADE	GPUntis	GestHor
Construção de Horários					
Construção automática de horários	X		X	X	X
Construção assistida de horários	X	X	X	X	X
Construção mista (automática / assistida)	X		X	X	X
Construção de Horários indiscriminada (por turma, professor ou sala)	X	X	X	X	X
Resolução de Outros Problemas de Escalonamento					
Construção de Calendários de Exames					
Distribuição de Serviço Docente				X	X
Definição do Número de Turmas					
Alocação dos Alunos a Turmas					
Alocação de Salas	X	X	X	X	X
Construção Automática de Horários					
Múltiplos parâmetros	X		X	X	X
Controlo por Tempo Decorrido					X
Controlo por percentagem de Aulas Marcadas	X		X	X	X
Controlo por Qualidade dos Horários	X		X	X	X
Não alteração das aulas "Fixas"	X			X	X
Métodos Construtivos na Construção de Horários	X		X	X	X

Métodos Reparativos na Construção de Horários	X		X	X	X
Edição/Visualização de Horários					
Janela Global de Horários	X	X	X	X	X
Janela de Horários	X	X	X	X	X
Indicação das aulas sem marcações (Incompletas)	X	X	X	X	X
Visualização de horários de Turmas, Professores e Salas	X	X	X	X	X
Marcação de aulas por "drag & drop"	X	X	X		X
Garantia de não-sobreposição de aulas	X	X	X	X	X
Impressão dos Horários					
Horários de Turmas, Professores e Salas	X	X	X	X	X
Impressão Personalizável	X	X		X	X
Impressão para formato HTML (Internet)	X	X	X	X	X
Exportação dos Horários para Word/Excel ou outros	X	X	X	X	X
Critérios de Qualidade – Restrições					
Satisfação de preferências, pausas, Indisponibilidades	X	X	X	X	X
Furos de Docentes e Turmas	X		X	X	X
Tempos Consecutivos	X			X	X
Tempos Máximos por dia ou outros Períodos	X	X	X	X	X
Tempos por Semana	X	X	X	X	X

Tabela 7: Comparativo entre as aplicações de construção de horários.

Esta análise permite-nos concluir que as aplicações que disponibilizam o modo automático de construção de horários, embora não resolvam por completo o problema oferecem, no entanto, mais-valias à organização administrativa escolar. Claramente o *Gp-Untis* e o *GestHor* apresentam algumas vantagens em relação às outras aplicações, uma vez que se encontram adaptadas para o ensino português.

Analisando a tabela 7, verifica-se que quase todas as aplicações possuem capacidades de construção manual, assistida e automática de horários. No entanto, relativamente à resolução de outros problemas de escalonamento universitários, as aplicações apresentam unicamente capacidade de efectuar a alocação de salas (integrada no processo de geração de horários). As aplicações *GPUntis* e *GestHor* apresentam potencialidades muito simples de auxílio na distribuição de serviço docente (permitindo manter a alocação final dos docentes às disciplinas em aberto até muito próximo da conclusão do processo de geração de horários). Nenhuma aplicação possui no entanto uma visão integrada do processo de geração de horários e consequente apoio a nível de definição do número de turmas, alocação dos alunos às turmas, auxílio real no processo de distribuição de serviço docente ou calendarização de exames.

Um estudo mais detalhado destas e de outras aplicações de construções de horários e os resultados de inquéritos elaborados às escolas secundárias e universidades nacionais sobre a construção de horários podem ser encontrados em [Reis, 1999] [Chavarria, 2002].

6.6 Representação de Problemas de Escalonamento Universitários

O grande crescimento que se verificou ao longo dos últimos anos na área da construção de horários tornou-se em si próprio um problema que gerou diversos desafios de investigação. A variedade de formatos de dados actualmente em uso e a existência de diversas variedades do problema da geração de horários (e outros problemas relacionados) tornam a comparação de resultados e a troca de ideias de investigação e de dados relativos a problemas reais extremamente difíceis. Diversas tentativas foram realizadas ao longo do tempo no sentido de encontrar um *standard* para a representação de problemas de construção de horários mas, actualmente, não existe ainda uma linguagem universal aceite para descrever estes problemas. Foram realizadas diversas tentativas no sentido de tentar encontrar esse formato de informação *standard*, que conduziram, no entanto, a linguagens incompletas em diversos aspectos.

6.6.1 A Linguagem TTL de Cooper e Kingston

Cooper e Kingston [Cooper e Kingston, 1993] propuseram uma especificação formal do problema baseada na TTL, uma linguagem de especificação de horários. Uma instância TTL consiste num grupo de tempo, alguns grupos de recursos e alguns eventos. Um grupo de tempo define os nomes dos tempos disponíveis para a alocação dos eventos, seguidos por uma especificação da forma em que os tempos são distribuídos ao longo dos dias da semana. Para especificar esta distribuição é proposto um formato, baseado na utilização de parênteses (para representar os dias), ponto e vírgula (que significam intervalos), pontos (para tempos desejados) e vírgulas (para dias indesejados). Grupos de recursos podem conter subgrupos, que são subconjuntos do conjunto de recursos que define funções que podem ser executadas. Um recurso pode encontrar-se em qualquer número de subgrupos.

Nesta linguagem de especificação, eventos são colecções de "*slots*" (conjunto dia/hora) aos quais serão atribuídos elementos de diversos grupos de recursos, sujeitos a certas restrições. Somente um conjunto básico de restrições é definido na especificação da linguagem.

6.6.2 A Linguagem Proposta por Cumming e Paechter

Cumming e Paechter [Cumming e Paechter, 1995] propuseram um formato de dados *standard* num artigo de discussão apresentado na conferência PATAT'95, mas não submetido formalmente na conferência ou impresso nos *proceedings*. As suas propostas foram bastante criticadas na conferência pela falta de generalização, mas originaram enormes discussões acerca do assunto, no qual, a dificuldade de criar um *standard* se tornou evidente. Nesse artigo de discussão, [Cumming e Paechter, 1995] propuseram

BIBLIOTECA F.E. PORTO

alguns princípios e requisitos como guias para a criação do standard. O referido standard pretendia representar horários completos e incompletos e ainda preferências. Os componentes usados são tempos (com uma representação *dia:hh:mm*), eventos, docentes, alunos e salas. Não faziam distinção entre docentes e alunos argumentando que em determinados casos, alunos poderão leccionar aulas. Uma lista de palavras-chave com diferentes parâmetros foi proposta como standard. Por exemplo, *oferta.sala* com parâmetros evento e sala, significa que determinado evento deverá ser atribuído a uma sala e que a referida sala é uma opção. Propõem, também, uma convenção cruzada (mas não como parte do *standard*) que pode ser ligada a qualquer palavra-chave e que originam um produto cartesiano entre os argumentos dessa palavra-chave (que neste caso são listas).

Cumming e Paechter tentaram representar as restrições flexíveis (*soft constraints*) utilizando funções de custo, mas concluíram que a avaliação de um horário é provavelmente a parte mais difícil de *standardizar* [Cumming e Paechter, 1995]. Algumas omissões importantes do trabalho estão relacionadas com a disponibilidade de recursos, repartição de eventos, grupos de recursos, semanas e outros tipos de períodos, tipos de salas, definição de qual o problema para resolver e como representar a solução.

6.6.3 O GATT de Collingwood, Ross e Corne

Um trabalho interessante relacionado com formatos standard para a representação de dados de problemas de construção de horários encontra-se incluído no sistema de horários – GATT, desenvolvido por Collingwood[34] et al. [Collingwood et al., 1996]. O *GATT (Genetic Algorithm Time Tabler)* utiliza um formato de ficheiro para descrever problemas de construção de horários. O formato pretende ser capaz de descrever qualquer *GELTP (General Exam/Lecture Timetabling Problem)*, bem como, problemas de horários não educacionais. O formato é descritivo e utiliza como componentes principais: eventos, "slots" de tempo, salas, estudantes e docentes. O formato é essencialmente destinado para problemas de calendarização de exames. Dessa forma, existem algumas omissões importantes que incluem semanas e outros períodos (úteis por exemplo para a distribuição de serviço docente em que os semestres têm de ser considerados), tipos de salas, ofertas de eventos (e duração das secções dos eventos), diversas restrições, tais como, as de continuidade (úteis para atingir bons horários para docentes e alunos), grupos de alunos (essencial em horários escolares) e grupos de docentes.

6.6.4 A Linguagem de Burke, Kingston e Pepper

Um artigo mais recente escrito por Burke et al. [Burke et al., 1998], propõe um tipo de standard diferente para instâncias de horários. Burke et al., incluem uma simples, mas

[34] Ema Collingwood cujo nome científico antes do seu casamento era Ema Hart.

incompleta descrição dos tipos de dados, palavras-chave e sintaxe da linguagem e apresentam algumas novas funcionalidades a desenvolver. Estas incluem a possibilidade de expressar instâncias completamente, como sejam recursos, reuniões, restrições (rígidas e flexíveis) complexas e soluções propostas (de forma a possibilitar a avaliação da solução de acordo com os critérios da instância) e que a tradução entre o formato standard definido e outros já existentes seja possível. O funcionamento da linguagem proposta é natural, de forma a permitir, o mais simples possível, a definição do problema de construção de horários, na sua forma matemática. Os tipos de dados utilizados no formato são: classes, conjuntos, sequências, inteiros, reais, *booleanos*, caracteres e *strings*. Um tipo de dados de grande utilidade deste formato são os conjuntos, pois permite representar conjuntos de alunos, docentes ou salas, típicos em problemas de geração de horários.

Em adição a um conjunto comum de operadores de conjuntos (*member, union, intersection, subset,* etc.), incluem também operadores como *forall, exists, sum* e *prod.* Isto transforma a sua linguagem de representação numa linguagem de programação, permitindo a definição de restrições de forma semelhante à de uma linguagem de programação com restrições.

6.6.5 A Linguagem STTL de Kingston

Kingston [Kingston, 1999], propôs a STTL, uma linguagem de especificação e avaliação de problemas de construção de horários, instâncias e soluções. Trata-se de uma linguagem orientada aos objectos, contendo os meios naturais de definição de entidades como sejam tempos, recursos e encontros, e funções que permitem que requisitos arbitrários sejam claramente expressos. Um problema é considerado como uma questão geral, para a qual é necessário encontrar uma resposta, como por exemplo um Problema de Construção de Horários ou o Problema de Calendarização de Exames. Uma instância de um problema é um caso particular desse problema. A solução de uma instância é um conjunto de decisões que resolvem essa instância (muitas vezes, não da melhor forma). No entanto, a linguagem não é completamente declarativa, existindo operadores de controlo que permitem a definição de expressões, mais ou menos complexas, dos parâmetros do problema. Deste modo, é necessária alguma aprendizagem da estrutura léxical, estrutural e funcional da linguagem, de forma a ser possível a definição de um problema.

6.6.6 A Linguagem Unilang

Mais recentemente, Reis e Oliveira [Reis e Oliveira, 2000b] [Reis e Oliveira, 2001] propuseram a linguagem *Scheduling Unilang* como uma linguagem genérica para a representação de problemas de construção de horários, calendarição de exames e problemas relacionados. Esta linguagem permite, não só representar os dados dos problemas, como também representar restrições, funções de avaliação, soluções parciais e

soluções completas dos mesmos. A linguagem *Scheduling Unilang* será objecto de uma descrição mais detalhada no capítulo 7.

6.7 Conclusões

Neste capítulo foi apresentado o problema da geração de horários e os problemas que lhe estão associados (distribuição de serviço docente, calendarização de exames e alocação de salas). Foram ainda apresentados os principais métodos de resolução de problemas de geração de horários e sistemas comerciais com vista à sua resolução. Concluiu-se que as metodologias de resolução e os sistemas comerciais de geração de horários não permitem resolver problemas de escalonamento da gestão universitária de forma integrada. Usualmente permitem efectuar a geração de horários ou a calendarização de exames sem preocupações com a ligação entre a fase de distribuição de serviço docente e estas fases, nem permitindo apoiar o processo de distribuição de serviço docente.

O domínio do escalonamento da gestão universitária coloca um vasto conjunto de desafios no que diz respeito à aplicação das metodologias de Inteligência Artificial Distribuída, nomeadamente os Sistemas Multi-Agente. As características do domínio e desafios associados incluem:

- **Resolução de Problemas NP-Completos.** Os problemas de escalonamento a resolver no âmbito de um SMA para a gestão universitária são NP-Completos [Schaerf, 1995] e como tal, não existe qualquer método exacto que os permita resolver em tempo polinomial. O tempo necessário para a sua resolução cresce exponencialmente com a dimensão do problema. Acresce a este facto a elevada dimensão que é usual numa Universidade ou Faculdade. Em Portugal, uma Faculdade de média dimensão possui habitualmente diversas licenciaturas e departamentos, mais de 200 docentes, 3000 alunos e 100 salas de aula.
- **Organização Interna Complexa da Universidade.** Uma Universidade típica em Portugal, organiza-se em diversas faculdades e/ou departamentos e secções. Possui várias licenciaturas e cursos de mestrado com responsabilidades, que por vezes são partilhadas pelos departamentos. Os planos curriculares incluem tradicionalmente disciplinas optativas e aulas de diversos tipos (teóricas, práticas, laboratoriais, etc.). As universidades possuem ainda salas de diversos tipos e capacidades.
- **Necessidade de Resolução Constante de Problemas.** Os problemas de escalonamento ocorrem essencialmente no início dos períodos lectivos anuais da Universidade (início do ano lectivo ou início dos semestres ou trimestres). No entanto, devido ao dinamismo do ambiente e dos agentes envolvidos no processo educativo, é necessário realizar alterações constantes ao longo do ano lectivo nos horários, distribuição de serviço docente e alocação de salas.

- **Agentes Heterogéneos.** Numa Universidade ou Faculdade existem diversos agentes reais com capacidades distintas (docentes, alunos, funcionários, etc.). De forma a criar um modelo de uma organização universitária, é necessário criar agentes computacionais que os representem, surgindo desta forma agentes docentes, agentes alunos ou turmas (que representam grupos de alunos) e diversos tipos de agentes escalonadores (capazes de resolver problemas de distribuição de serviço docente, geração de horários, calendarização de exames, etc.).
- **Necessidade de Interacção entre Agentes e Humanos.** Sendo o domínio da gestão universitária, um domínio real, é necessário que os agentes projectados para operarem neste domínio, possuam capacidades de resolução de problemas mas também de interacção com os humanos que irão representar.
- **Diversos Problemas Interrelacionados.** A resolução de problemas de escalonamento na gestão universitária, implica a necessidade de resolução de diversos problemas interrelacionados. No entanto, os decisores responsáveis pela resolução de cada uma dos problemas são usualmente distintos e têm de comunicar e cooperar para que seja possível efectuar a resolução do problema global.
- **Distribuição Espacial dos Agentes.** Os agentes presentes na Universidade encontram-se geograficamente distribuídos. Diferentes departamentos, usualmente encontram-se situados em edifícios distintos. É típico em Portugal que as universidades não possuam um *campus* único e que as suas Faculdades se encontrem distribuídas por vários pólos, localizados em sítios distintos de uma ou várias cidades[35].
- **Visão Parcial do Problema.** Os agentes escalonadores possuem unicamente uma visão parcial do problema e necessitam por vezes de solicitar informação adicional aos restantes agentes durante a sua resolução do problema.
- **Ambiente Multi-Objectivo, Parcialmente Cooperativo e Parcialmente Adverso.** Os agentes presentes no ambiente possuem múltiplos objectivos complexos, i.e. muito difíceis de explicitar em funções de utilidade. Por exemplo, relativamente aos docentes, estes possuem como objectivos principais, leccionar as disciplinas que preferem com um bom horário e em salas apropriadas. No entanto, os agentes não desejam leccionar demasiadas horas semanais. Uma questão que se coloca é como se podem balancear os objectivos relativos à

[35] Exemplos típicos disto são as Universidades do Porto e do Minho. No primeiro caso, a Universidade encontra-se distribuída por 3 pólos na cidade do Porto, enquanto no segundo caso, encontra-se inclusivamente distribuída por duas cidades (Braga e Guimarães) que distam entre si de cerca de 20 km.

distribuição de serviço? Por um lado, os agentes docentes não pretendem leccionar muitas horas mas até podem preferir leccionar mais horas, desde que estas sejam das disciplinas que preferem. Neste contexto, a distribuição de serviço e os horários podem implicar a competição entre os agentes docente, quer pela regência ou aulas de disciplinas, quer na construção de bons horários. No entanto, os agentes não desejam prejudicar os seus colegas nem os alunos, ou seja, os agentes são parcialmente cooperativos. Por vezes, é necessário formar equipas de agentes que permitam, por exemplo, leccionar correctamente determinadas disciplinas.

- **Necessidade de Balancear Processos de Resolução de Problemas e de Negociação.** De forma a construir horários, distribuições de serviço docente ou calendários de exame, é necessário resolver problemas de escalonamento NP-Completos. No entanto, a resolução destes problemas não é realizada por um único agente com uma visão global do problemas, mas sim por múltiplos centros de decisão que possuem unicamente uma visão parcial do mesmo. Desta forma, os agentes não possuem um controlo total sobre os recursos que irão utilizar na resolução dos seus problemas locais e necessitam de encetar processos de negociação com outros agentes (nomeadamente com outros agentes escalonadores ou com agentes responsáveis por gerir cada um dos recursos) de forma a utilizarem correctamente os recursos nas suas resoluções.
- **Hierarquias entre Agentes.** O domínio da gestão universitária é rico em relações hierárquicas pouco rígidas entre agentes. Por exemplo, numa Faculdade é usual a existência de um director, vários departamentos com directores, licenciaturas com respectivos coordenadores ou directores, secções e respectivos coordenadores, etc. No entanto, é usual a existência de alguma flexibilidade nas relações hierárquicas definidas e nas funções atribuídas a cada agente.

No capítulo sete desta tese, estes desafios serão analisados com maior detalhe e algumas metodologias de coordenação, adequadas à resolução de problemas no domínio da gestão universitária, serão propostas. Deve, no entanto, ser realçado que o domínio da gestão universitária foi utilizado nesta dissertação de forma bastante simplificada, uma vez que as metodologias de coordenação desenvolvidas implicam na sua maioria a movimentação espacial de agentes. Desta forma, o domínio principal de teste das metodologias de coordenação desenvolvidas foi o futebol robótico simulado apresentado nos capítulos oito e nove.

Capítulo 7

7. Coordenação em SMA na Gestão Universitária

O projecto UNIPS teve como objectivo construir um sistema que permitisse efectuar a construção de horários e resolução de problemas associados de forma integrada. Os problemas de escalonamento principais associados aos horários incluem: a distribuição de serviço docente, a calendarização de exames e a alocação de salas. Como foi já analisado anteriormente, normalmente estes problemas são resolvidos de forma separada ou pelo menos não totalmente interligada. É usual efectuar a distribuição de serviço docente completa e só depois começar a resolução do problema de geração de horários. No entanto, devido a existirem interligações profundas entre estes problemas, a sua resolução de forma integrada é muito desejável.

A resolução integrada destes problemas tem de ter em consideração que a Universidade encontra-se usualmente dividida em faculdades, departamentos ou secções e a resolução real dos problemas de DSD, GH ou CE é efectuada de forma distribuída por essas unidades. Estas unidades não são totalmente autónomas pois:

- Partilham recursos: docentes, salas e/ou alunos (turmas);
- Não existe uma definição clara dos limites das responsabilidades ou recursos alocados a cada departamento ou unidade.

A complexidade na resolução distribuída do problema é ainda maior se pensarmos que grande parte dos recursos geridos (i.e. docentes, alunos, etc.), são também em si, unidades com um elevado grau de autonomia e com objectivos e vontades próprias. Desta forma, no âmbito do projecto UNIPS, procurou-se modelizar cada entidade envolvida no processo de geração de horários (e processos associados) como um agente inteligente que se encontra inserido no âmbito de um Sistema Multi-Agente. O processo de resolução implementado para o problema, procura também simular o processo real de construção de horários numa Universidade. Este processo centraliza a resolução de uma parcela do problema, existindo no entanto, negociação e cooperação entre os intervenientes para tentar resolver conflitos ou melhorar partes da solução global do problema que afectem mais do que um agente.

7.1 Tipos de Coordenação Aplicáveis ao Domínio

A geração de horários e a resolução de outros problemas de escalonamento em universidades, permite a aplicação de diversas metodologias de coordenação, nomeadamente:

- **Coordenação por Comunicação.** Utilização de protocolos de comunicação que permitam aos agentes envolvidos no processo de resolução dos problemas de escalonamento trocar informação e outros eventos úteis para coordenação;
- **Coordenação por Percepção Inteligente.** Definição de algoritmos que permitem dar atenção preferencial a agentes que possuem informação útil para a resolução actual do problema;
- **Coordenação por Organização Estrutural.** Definição de uma estrutura organizativa com uma alocação correcta de responsabilidades, que permita a resolução mais simples dos problemas de escalonamento nas universidades;
- **Coordenação por Decomposição e Alocação de Tarefas.** Decomposição da tarefa global em sub-tarefas e sua alocação dinâmica aos diferentes agentes do sistema;
- **Coordenação por Modelização Mútua.** Definição de metodologias que permitam a construção de modelos dos restantes agentes do sistema que permitam auxiliar os processos de decisão dos agentes individuais;
- **Coordenação Parcialmente Hierárquica.** Utilização de agentes hierarquicamente superiores e definição de metodologias que permitam a agentes autónomos hierarquicamente inferiores, integrarem os conselhos ou ordens dos agentes superiores, com as suas regras locais de decisão;
- **Coordenação por Conhecimento *a Priori*.** Definição de uma linguagem que permita a definição de problemas completos de escalonamento universitário, as suas soluções parciais e métodos de avaliação, que sejam partilhadas por todos os agentes do sistema.

Estes mecanismos de coordenação foram implementados unicamente de forma parcial para este domínio, pois o domínio de teste principal das metodologias definidas no âmbito desta tese foi o futebol robótico simulado.

Desta forma, a metodologia principal proposta para este domínio é a linguagem *SCHEDULING UNILANG*, que é um mecanismo que permite efectuar a Coordenação por Conhecimento *a Priori*, na resolução de problemas de escalonamento na Gestão Universitária.

7.2 Formalização de Problemas de Escalonamento Universitário

Como foi analisado no capítulo seis, o problema da *Distribuição de Serviço Docente* consiste em definir os docentes responsáveis por cada uma das disciplinas leccionadas numa dada Universidade ou Faculdade e atribuir a leccionação de turmas de disciplinas de cada tipo (teórica, prática, teórico-prática, laboratorial) aos docentes qualificados para as leccionar, tendo em atenção diversos critérios e restrições. As restrições incluem o respeito pelas capacidades dos docentes, pelas preferências manifestadas pelos mesmos em leccionar as diferentes disciplinas e o respeito pelas cargas horárias que podem ser atribuídas em cada período a cada docente.

Em muitas instituições de ensino, a acumulação de tarefas no período que antecede o início de um novo ano lectivo, e a pressão existente para produzir os horários definitivos antes do início das aulas, provoca que o processo de distribuição do serviço docente seja, por vezes, executado rapidamente, ficando o seu resultado final longe de ser óptimo. Os principais lesados com esta resolução pouco cuidada do problema, são os docentes e alunos, bem como, os responsáveis pelos processos de *Construção dos Horários* lectivos.

Embora a *Distribuição de Serviço Docente* possa ser realizada de diversas formas distintas, este processo pode ser genericamente dividido em três fases fundamentais:

- **Recolha de Preferências dos Docentes.** A primeira fase inicia-se bastante antes do arranque do ano lectivo e consiste em averiguar as disciplinas que os docentes pretendem leccionar (preferências). A forma mais comum de os responsáveis pelo processo realizarem esta tarefa, consiste em entregar ao corpo docente um formulário onde os docentes poderão indicar um número restrito de disciplinas que pretendam leccionar e qual a sua preferência (numa dada escala pré-definida) em leccionar cada disciplina. Em determinadas Instituições de Ensino Superior aproveita-se esta fase para averiguar também as preferências de horário dos docentes e o número de horas que estes pretendem leccionar (no caso de docentes com contrato a tempo parcial, em que este valor não se encontra especificado). Pode-se ainda verificar a disponibilidade dos docentes para serviços docentes pouco balanceados, i.e. com carga docente concentrada num dos semestres.
- **Definição de Regências e Docentes que irão Leccionar as Disciplinas.** A segunda fase do processo é a mais trabalhosa e consiste na *Distribuição de Serviço Docente* propriamente dita. Para cada disciplina podem existir diversos docentes interessados em leccioná-la. Pode também não existir qualquer docente interessado, caso em que será necessário encontrar um docente adequado para a leccionar. Os responsáveis pela distribuição de serviço devem escolher o docente

mais adequado para ser o regente[36] de cada disciplina. Além do regente responsável pela disciplina, são também definidos todos os docentes necessários para garantir todas as aulas dos diversos tipos (teórica, prática, teórico-prática, laboratorial) a todas as turmas dessa disciplina. É necessário garantir que todas as aulas são atribuídas a um docente, tendo o cuidado de não sobrecarregar qualquer docente com demasiadas aulas. Os responsáveis por realizar o processo de distribuição de serviço docente e o conjunto de restrições e critérios utilizados nesta fase são bastante variáveis de instituição para instituição. Os responsáveis mais usuais são o reitor da Universidade, director da Faculdade, directores dos departamentos e/ou secções ou directores das licenciaturas ou cursos de mestrado. É usual, que não exista um processo bem definido para a execução desta distribuição e que diversos centros de decisão estejam envolvidos no processo e tenham de interactuar de forma a atingir uma solução final satisfatória.

- **Ajustes Iterativos da Distribuição de Serviço.** A terceira fase do processo consiste em ajustar a distribuição de serviço usando o "*feedback*" dos docentes e de outros agentes envolvidos no processo (reclamações e sugestões). Este processo pode consistir em diversas rondas de negociação e usualmente termina quando é atingida uma solução aceitável para a grande maioria dos envolvidos ou quando o tempo disponível para finalizar o processo se esgota. Por vezes, são os próprios docentes a colaborar no processo de distribuição de serviço, através da proposta de soluções alternativas e trocas de serviço entre eles, que permitam melhorar a qualidade global da solução. De notar, que não é estritamente necessário, mas deveras aconselhável, que este processo esteja concluído antes do processo de *Construção dos Horários*. As alterações na *Distribuição do Serviço Docente*, após o início do processo de *Construção dos Horários*, devem ser mínimas e realizadas apenas em casos especiais.

Constata-se que em diversas instituições de ensino superior nacionais, durante o processo de *Distribuição de Serviço Docente*, não é utilizado qualquer apoio informático, ao contrário do processo de *Construção de Horários*, no qual diversas instituições têm esse apoio nas diferentes fases do processo. Geralmente, toda a informação resultante do referido processo é armazenada em folhas de papel ou numa folha de cálculo *Excel*[37], utilizando-se o computador apenas para validar os dados e como ferramenta de edição de texto (sobretudo para a construção dos relatórios finais).

---

[36] O regente é o responsável pela disciplina. Usualmente lecciona as aulas teóricas da disciplina e eventualmente uma parcela ou a totalidade das aulas práticas ou teórico-práticas da mesma. No entanto, pode acontecer que o regente de uma dada disciplina, não leccione a totalidade das aulas teóricas da disciplina, leccione unicamente aulas práticas ou não leccione aulas dessa disciplina.

[37] Aplicação do tipo folha de cálculo, integrada do pacote Microsoft Office

A relação existente entre a distribuição de serviço docente e a geração de horários, não é frequentemente explorada na execução deste processo. Deste facto resulta, que são por vezes, atribuídas disciplinas cujo horário é nocturno a docentes que colocam nas suas preferências de horários uma impossibilidade total de leccionar à noite. A resolução do subsequente problema de horários torna-se impossível, considerando todas as restrições, e torna-se necessário alterar a distribuição de serviço ou relaxar algumas restrições na geração de horários.

Devido ao elevado número de restrições dificilmente explicitáveis e formas de avaliação da solução, muito subjectivas, não é fácil formalizar o conceito de problema de distribuição de serviço docente. Simplificando algumas restrições, podemos formalizar o conceito de problema de *Distribuição de Serviço Docente* como sendo um problema de optimização combinatória *PDSD* [Reis, 2001d]:

Suponha-se a existência de:

- Um conjunto $Docentes = \{Doc_1, Doc_2, ..., Doc_{ndoc}\}$ de docentes que leccionam na Universidade;
- Um conjunto $Disciplinas = \{D_1, D_2, ..., D_{ndisc}\}$ de disciplinas leccionadas na Universidade;
- Um conjunto $Turmas = \{T_1, T_2, .., T_{ntur}\}$ representando as turmas da Universidade;
- Um conjunto $Aulas = \{A_1, A_2, ..., A_{naulas}\}$ de aulas das diferentes disciplinas leccionadas na Universidade;
- Um conjunto $Períodos = \{P_1, P_2, ..., P_{nper}\}$ de períodos lectivos distintos (semestres, trimestres, etc.);
- Um conjunto $Tipos = \{T_1, T_2, ..., T_{ntipos}\}$ de tipos de aulas (teóricas, teórico-práticas, práticas, laboratoriais, etc.);

É conhecido ainda:

- $DuraçãoAulas = \{DA_1, DA_2, ..., DA_{naulas}\}$ correspondendo à duração de cada uma das *naulas* aulas leccionadas na Universidade;
- $PeríodoDisciplina = \{PD_1, PD_2, ..., PD_{ndisc}\}$ atribuindo a cada disciplina o período em que esta será leccionada;
- $AulaDisciplina = \{ADisc_1, ADisc_2, ..., ADisc_{naulas}\}$ em que $ADisc_j \in Disciplina$, associando a cada uma das *naulas* aulas a disciplina respectiva;
- $Importância_i$, $\forall i \in Docentes$, correspondendo às importâncias dos *ndoc* docentes na Universidade;
- $Preferências_i = \{Pref_1, Pref_2, ..., Pref_{ndisc}\}$, $\forall i \in Docentes$, correspondendo às preferências dos *ndoc* docentes relativamente à leccionação de cada uma das

ndisc disciplinas;

- $Capacidades_i = \{Cap_1, Cap_2, ..., Cap_{ndisc}\}$, $\forall i \in Docentes$, correspondendo às capacidades dos *ndoc* docentes relativamente à leccionação de cada uma das *ndisc* disciplinas;
- $HorasMax_i$, $\forall i \in Docentes$, atribuindo a cada docente um máximo de horas anual;
- $HorasMin_i$, $\forall i \in Docentes$, atribuindo a cada docente um mínimo de horas anual;
- $HorasDes_i$, $\forall i \in Docentes$, atribuindo a cada docente um valor desejado de horas anual;
- $HorasPeríodoMax_i = \{HPMax_1, HPMax_2, ..., HPMax_{nper}\}$, $\forall i \in Docentes$, atribuindo a cada docente um máximo de horas em cada um dos períodos anuais;
- $HorasPeríodoMin_i = \{HPMin_1, HPMin_2, ..., HPMin_{nper}\}$, $\forall i \in Docentes$, atribuindo a cada docente um máximo de horas em cada um dos períodos anuais;
- $HorasPeríodoDes_i = \{HPDes_1, HPDes_2, ..., HPDes_{nper}\}$, $\forall i \in Docentes$, atribuindo a cada docente um máximo de horas em cada um dos períodos anuais;

As Variáveis de Decisão são:

- $Regente = \{RD_1, RD_2, ..., RD_{ndisc}\}$ em que $RD_i \in Docentes$, correspondendo aos regentes das diversas disciplinas leccionadas na Universidade;
- $Serviço = \{SA_1, SA_2, ..., SA_{naulas}\}$ em que $SA_i \in Docentes$, correspondendo aos docentes atribuídos a cada uma das aulas das diversas disciplinas leccionadas na Universidade;

Variáveis Auxiliares:

- $Lecciona_{i,j} = \begin{cases} 1 \text{ se } Serviço_j = i \\ 0 \text{ caso contrário} \end{cases}$ indicando, que um dado docente lecciona uma aula;
- $DiscPer_{i,j} = \begin{cases} 1 \text{ se } PeríodoDisciplina_j = i \\ 0 \text{ caso contrário} \end{cases}$ indicando, que uma disciplina é leccionada num dado período;
- $AulaPer_{i,j} = DiscPer_{i,AulaDisciplina_j}$ indicando, que uma dada aula de uma disciplina é leccionada num dado período;

Um *PDSD* consiste então na definição dos valores atribuídos às variáveis dos conjuntos *Regentes* e *Serviço*, de forma a:

$$Minimizar\ f = \Sigma w_{carga} * f_{carga_docente} + w_{adeq} * f_{adequação_docente}$$

Sendo

- w_{carga} e w_{adeq} os pesos relativos dados ao respeito pelas cargas lectivas desejadas para os docentes e pela adequação dos docentes à leccionação das disciplinas.
- $f_{carga_docente}$ uma função que mede o respeito pela carga lectiva dos docentes.
- $f_{adequação_docente}$ uma função que mede o respeito pelas preferências e adequações dos docentes a leccionar as disciplinas.

Respeitando as seguintes restrições rígidas do problema:

- $\sum_{j=1}^{naulas} DA_j * (SA_j = i) \leq HorasMax_i \quad \forall i \in [1, ndoc]$
  - Nenhum docente pode leccionar um número de horas superior ao máximo no total dos vários períodos
- $\sum_{j=1}^{naulas} DA_j * (SA_j = i) \geq HorasMin_i \quad \forall i \in [1, ndoc]$
  - Nenhum docente pode leccionar um número de horas inferior ao mínimo no total dos vários períodos
- $\sum_{j=1}^{naulas} DA_j * (SA_j = i \wedge PD_{ADisc_{SAj}} = p) \leq HorasMax_{i,p} \quad \forall i \in [1, ndoc], \forall p \in [1, nper]$
  - Nenhum docente pode leccionar um número de horas superior ao máximo no total de cada um dos períodos
- $\sum_{j=1}^{naulas} DA_j * (SA_j = i \wedge PD_{ADisc_{SAj}} = p) \leq HorasMax_{i,p} \quad \forall i \in [1, ndoc], \forall p \in [1, nper]$
  - Nenhum docente pode leccionar um número de horas inferior ao mínimo no total de cada um dos períodos

Claro que esta formulação não está totalmente completa pois não inclui tópicos, tais como:

- Associações de disciplinas, modelizando o facto de ser vantajoso um dado docente leccionar uma disciplina no caso de também leccionar uma outra;
- Grupos de docentes, modelizando o facto de determinados docentes leccionarem melhor em conjunto do que outros. É usual em instituições de ensino superior portuguesas, surgirem restrições do tipo: "não lecciono esta disciplina se for em conjunto com o docente *X*" ou "só lecciono esta disciplina se as aulas práticas forem leccionadas pelo docente *Y*".
- Possibilidade de mudança de semestre (ou período) de uma dada disciplina. Por vezes é possível mudar disciplinas de semestre, permitindo um melhor balanceamento da carga dos docentes. Isto pode ser modelizado incluindo uma matriz de custo de mudança de período de cada uma das disciplinas. Nesta

matriz, por exemplo, uma disciplina como Análise Matemática 2, teria um custo muito elevado de mudança de semestre (devido às relações de precedência com disciplinas como Análise Matemática 1 ou Análise Matemática 3). Uma disciplina sem relações de precedência poderia ter um custo reduzido de mudança de semestre.

- Benefício para a leccionação de um número reduzido de disciplinas por parte de cada docente. É benéfico que cada docente não leccione um número muito elevado de disciplinas. Isto poderia ser facilmente incluído na função de avaliação, introduzindo um valor de número de disciplinas desejado (e eventualmente máximo e mínimo também) para cada docente (ou geral);
- Benefício para a leccionação por parte de pelo menos dois docentes de cada uma das disciplinas. É benéfico em termos pedagógicos que cada disciplina seja leccionada por pelo menos dois docentes. Deste modo, em caso de doença ou outro tipo de impossibilidade, por parte de um dos docentes, a qualidade de ensino não sofrerá um decréscimo considerável. No entanto, este objectivo é um pouco contraditório com o objectivo anterior;
- O cuidado em não prejudicar em demasia um único indivíduo. Na resolução de problemas de distribuição de serviço docente reais, por vezes é melhor uma solução global de pior qualidade mas que não prejudique em demasia qualquer indivíduo em particular (por exemplo, atribuindo a um docente uma carga muito elevada de aulas, de disciplinas fora das suas preferências e para as quais não possua habilitações).

Um problema evidente desta formulação é que parte da informação utilizada é relativamente subjectiva e de carácter qualitativo. Por exemplo, é muito complicado definir um valor numérico relativo à importância dos docentes. Pode-se utilizar uma conversão directa da categoria do docente para uma escala numérica (por exemplo: Professor Catedrático – 5, Professor Associado – 4, Assistente Estagiário –1), mas parte da informação utilizada pelos decisores humanos na resolução deste problema é perdida. Da mesma forma, o tratamento de preferências, ou adequações de docentes a disciplinas é relativamente complicado.

Formalizações semelhantes à apresentada foram também desenvolvidas para os problemas de geração de horários e calendarização de exames, que serviram de base para a construção dos respectivos agentes de resolução.

7.3 A Linguagem *SCHEDULING UNILANG*

Neste capítulo será apresentada uma linguagem (*SCHEDULING UNILANG*) definida como parte da aplicação de metodologias de coordenação por conhecimento *a priori* no

domínio do escalonamento universitário. Esta linguagem foi definida tendo como base as formalizações que permitem a representação de problemas completos de construção de horários e outros problemas associados e pode facilmente ser lida e entendida por informáticos e administradores de escolas e universidades. A linguagem *SCHEDULING UNILANG* é baseada na definição de oito sub-problemas do problema completo de construção de horários [Reis e Oliveira, 2000a]. Esta linguagem foi projectada no âmbito desta tese e utilizada extensivamente em diversas partes do projecto UNIPS associado a esta tese.

7.3.1 Sub-Problemas da Construção de Horários

A construção de horários pode ser vista como um problema multidimensional de atribuições [Carter e Laporte, 1996], no qual alunos e docentes (ou vigilantes) são atribuídos a disciplinas, exames, ofertas de disciplinas ou turmas e eventos (encontros individuais entre docentes e alunos) são atribuídos a salas e tempos. Isto indica que não temos um único problema designado por construção de horários. Podemos ter problemas de atribuições de alunos, atribuições de docentes (ou vigilantes), alocação de salas e alocação de tempos, todos incluídos num problema de construção de horários global. A descrição de um dado problema deve ser pré-processada de forma a executar testes de validade e decomposição do problema original nos seus subproblemas associados. Os subproblemas depois de resolvidos, utilizando algoritmos apropriados, permitem a construção da solução final do problema de geração de horários. É possível visualizar, na figura 38 uma representação genérica de um problema de construção de horários.

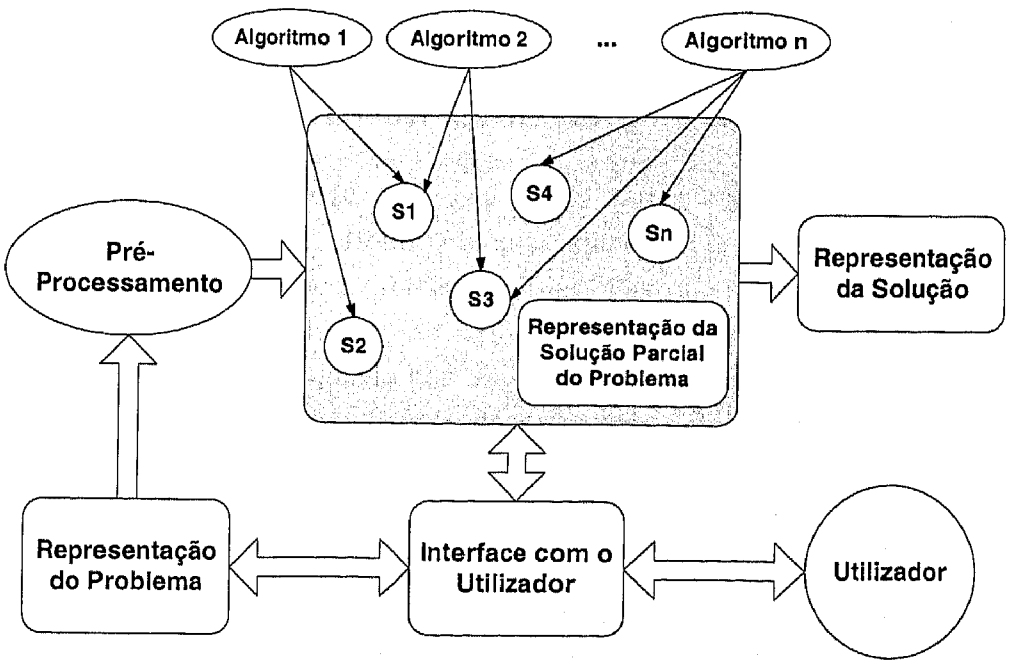

Figura 38: Representação genérica de um problema de construção de horários

Estendendo a definição de Carter e Laporte [Carter e Laporte, 1996], dos sub-problemas de horários, Reis e Oliveira [Reis e Oliveira, 2000a] propuseram a definição das oito classes seguintes de sub-problemas de horários inter-relacionados:

- **HTD – Horários Escolares (Horários Turma-Docente):** Este é um problema comum na maior parte das escolas e instituições de ensino superior portuguesas. Consiste em atribuir um tempo (ou conjunto de tempos) a cada aula de cada módulo na escola (ou universidade). A unidade de escalonamento é o grupo de alunos (turma) que tem um plano curricular comum. É considerado que a atribuição dos docentes e turmas aos eventos já se encontra realizada previamente. Normalmente, as salas são usadas como restrições neste problema ou então a alocação de salas é executada simultaneamente com o horário de turma-docente;
- **HU – Horários Universitários (Horários de Disciplinas):** O escalonamento de todas as aulas de um conjunto de módulos de um conjunto de cursos de uma universidade, minimizando as sobreposições de aulas com alunos em comum e evitando que os docentes tenham mais do que uma aula ao mesmo tempo. Neste caso, os alunos são tratados individualmente e não é assumido que pertençam a algum grupo, como acontece nos horários escolares. Os docentes, normalmente, encontram-se já atribuídos (no entanto, é incluída alguma flexibilidade na atribuição). Por vezes, os alunos não são atribuídos às ofertas de eventos antes da resolução deste problema. Os estudantes encontram-se normalmente, atribuídos aos eventos previamente, mas em algumas universidades isto pode também ser falso;
- **CE – Calendarização de Exames:** Escalonar (no tempo) os exames de um conjunto de disciplinas de uma universidade, evitando sobreposição de exames que contenham alunos em comum e espalhando os exames de cada aluno o mais possível. A atribuição de salas e atribuição de vigilantes poderão ser realizadas antes ou após a etapa CE;
- **DO – Definição de Ofertas:** Esta necessidade ocorre dentro de todos os problemas de construção de horários. Consiste em, para cada módulo, definir o número de ofertas (turmas) existente. Um problema similar ocorre na construção dos calendários de exame se considerarmos a divisão de exames (em múltiplos períodos no tempo). Nesse caso, o número de ofertas para cada exame deverá ser determinado;
- **EA – Escalonamento de Alunos:** Este problema ocorre quando os módulos são leccionados em múltiplas ofertas. Após os alunos terem realizado a seleccção dos módulos, devem ser atribuídos a secções de módulos, tentando-se obter horários (para os alunos) sem conflitos e balanceando os tamanhos das ofertas;
- **AD – Alocação de Docentes:** Atribuir docentes a diferentes módulos, tentando

respeitar as suas preferências, as suas horas desejadas de tempo de aulas, o balanceamento desejado entre horas de aulas durante os períodos do ano (semestres ou trimestres), bem como, outras restrições. Por vezes, isto é realizado em dois ou três passos em separado. O primeiro passo consiste na selecção dos docentes responsáveis por cada um dos módulos (no entanto, nas universidades públicas mais antigas, este problema é resolvido efectuando somente pequenas alterações de ano para ano). No segundo passo, os docentes que vão leccionar os diversos tipos de aulas de cada um dos módulos, são seleccionados. Um terceiro passo consiste na atribuição de secções individuais (turmas) a docentes. Por vezes, isto mantém-se em aberto (ou, pelo menos, muito flexível) até à construção dos horários;

- **AV – Alocação de Vigilantes:** Este é um problema habitual associado com o escalonamento de exames. Cada exame necessita de um ou mais vigilantes. O número de vigilantes é normalmente definido de acordo com o número de alunos que realizam o exame e o número de salas usadas;
- **AS – Alocação de Salas:** Habitualmente, todos os problemas reais de construção de horários têm uma fase de atribuição de salas. Os eventos deverão ser atribuídos a salas específicas (ou conjuntos de salas), satisfazendo o tamanho e tipo de necessidades do evento. Podem surgir problemas com a separação de eventos, salas partilhadas e distâncias entre salas.

Outros sub-problemas poderiam ser incluídos nesta classificação, mas a importância não será a mesma daqueles acima considerados. Por exemplo, após a construção dos horários de exames, algumas instituições poderão agregar pequenos exames em conjuntos, que serão escalonados ao mesmo tempo, utilizando a mesma sala e os mesmos vigilantes. Outras instituições usam módulos com diferentes cargas horárias ao longo do ano, tornando os horários diferentes todas as semanas.

Para descrever um problema específico de geração de horários, é necessário conhecer em detalhe, quais dos sub-problemas irão ser resolvidos e a ordem pela qual isso será efectuado.

7.3.2 Requisitos da Linguagem *SCHEDULING UNILANG*

Muitos formatos utilizados por aplicações de construção de horários têm sido desenvolvidos com objectivos de minimizar o espaço usado para armazenamento de dados ou facilitar o rápido processamento dos mesmos. O formato usado nesta linguagem tenta obter um compromisso entre generalidade e simplicidade. É suficientemente geral para permitir a representação das variantes mais comuns dos problemas de construção de

horários, incluindo escolas, universidades e calendarização de exames, tendo uma sintaxe bastante simples. Os principais requisitos na elaboração deste formato foram:

- Ser independente dos detalhes de implementação e estratégias de construção de horários;
- Ser fácil de estender, com a inclusão de novos conceitos e restrições;
- Os problemas existentes devem ser facilmente traduzidos para este formato;
- O formato deve ser facilmente lido pela máquina (e qualquer sistema de horários), bem como, pelo utilizador humano;
- A informação que não está directamente relacionada com o problema de escalonamento não deve ser incluída neste formato. Como tal, informação comum acerca da administração da escola, como seja, por exemplo, os nomes dos estudantes e respectivas moradas, serão omitidos do formato;
- O formato deverá ser suficientemente geral para permitir a representação de problemas de horários escolares, universitários e calendarização de exames;
- Deverá permitir a representação de problemas completos e subproblemas simples relacionados (como atribuições de vigilâncias, definição de ofertas e atribuição de salas);
- Deverá permitir a definição clara de todas as restrições associadas com os problemas;
- Deverá permitir a definição de todas as restrições rígidas e flexíveis deste tipo de problemas;
- Uma função de avaliação permitindo a medida da qualidade do horário deverá ser fácil de incluir no formato. Isto permite avaliar directamente qualquer solução proposta, de acordo, com os critérios definidos no problema;
- A dimensão dos ficheiros criados deverá ser a mais reduzida possível;
- O formato deverá permitir a representação de soluções completas e incompletas de problemas dos diversos tipos;
- Deverá ser robusto permitindo a validação de dados simples e a eliminação de erros comuns.

7.3.3 Componentes da Linguagem

A linguagem permite a definição de diferentes períodos compostos por um conjunto de semanas (ou outro período de tempo) com escalonamentos iguais ou similares. Cada semana é composta por períodos de tempo, localizados em determinado dia (nessa semana) e tempo. No modelo definido, é preferível separar a definição de recursos em três

classes principais: alunos, docentes e salas. Embora existam algumas semelhanças entre as três classes (por exemplo, todos os recursos têm restrições de disponibilidade), as diferenças são também evidentes em qualquer problema de construção de horários. Por exemplo, as salas podem suportar (em determinados problemas) diversos eventos ao mesmo tempo, ter restrições de capacidade, tipos e distâncias entre elas. Alunos e docentes podem ser agrupados em grupos (com objectivos diferentes) e ter restrições de tipo e carga de trabalho diária. Os docentes podem ter um número mínimo e máximo de restrições de atribuição de eventos e restrições de habilitações (indicando a sua capacidade ou desejo de leccionar determinado assunto). Os alunos (e grupos de alunos) podem também ter restrições de espalhamento (no caso da calendarização de exames, que se pretendem espalhados ao longo do período) e continuidade (no caso de horários escolares, que se pretendem sem furos).

Os componentes considerados são:

- Períodos (Semestres/Trimestres)
- Tempos (Dia/Hora)
- Recursos:
  - Salas
  - Docentes
  - Alunos
- Eventos (Aulas/Exames/Reuniões).

Um evento é um encontro entre um docente (ou conjunto de docentes) e um conjunto de alunos (ou grupo de alunos) que tem lugar numa sala (ou num conjunto de salas) num determinado tempo (ou conjunto de tempos). Não são considerados outros recursos, como o equipamento, porque estes não são particularmente importantes em problemas de construção de horários.

Como, por vezes, a distinção entre dados e restrições não é completamente clara, este modelo permite a especificação de dados e restrições num formato comum [Reis e Oliveira, 2000a]. Alguns dos tipos de restrições incluídos têm duas versões diferentes. Uma versão forte na qual a restrição deverá ser respeitada (*hard constraint*) e uma versão fraca na qual dever-se-á tentar respeitar a restrição (*soft constraint*). O conjunto de *soft constraints* incluído, permite a definição de uma função de avaliação para medir a qualidade do horário. Cada uma das *soft constraints* incluída tem uma preferência numérica, a qual é o custo da não satisfação dessa restrição.

Uma solução (parcial ou global) para um problema de construção de horários, consiste em alocar a cada um dos eventos considerados: um conjunto de tempos ("slots"), um conjunto de salas, um conjunto de docentes e eventualmente, um conjunto de alunos (ou grupo de alunos). No entanto, na maior parte dos problemas de construção de horários encontrados

na literatura da especialidade, alunos e docentes encontram-se já atribuídos aos eventos (ou ofertas dos eventos). Este modelo pretende ser suficientemente geral para resolver esses subproblemas de atribuição que são: atribuição de docentes, atribuição de vigilantes (no caso de exames) e escalonamento dos alunos.

Para que o formato fosse o mais robusto possível, seguindo a ideia de Collingwood et al. [Collingwood et al., 1996] no sistema GATT, pode ser incluída uma lista de sinónimos para cada um dos conceitos (palavras chave) usados, independentemente da língua utilizada. A lista de sinónimos incluída por defeito na linguagem é:

```
[Default | All | Every | Any],
[Year | Problem | File],
[Schedule | Timetable],
[Event | Module | Lecture | Lesson | Exam | Examination | Tutorial],
[Teacher | Teachers | Invigilator | Supervisor | Lecturer],
[Student | Students],
[Day | Days],
[Time, | Times | Hour | Hours | Minute | Minutes],
[Slot | Slots | Time Slot],
[Place | Places | Room | Rooms],
[Preference | Weight | Priority | Penalty],
[Consecutive | Continuos],
[Simultaneously | Concurrently | Together | At the same time],
[Teaches | Invigilates | Supervises],
[Hold | Holds | Have capacity | Has capacity | Capacity],
[Specify | Specifies | Require | Requires | Need | Needs],
[Last | Lasts],
[Duration | Length],
[Contains | Comprises | Has | Have],
[Is | Are],
[At least | No less than | No fewer than],
[At most | No more than],
[Exactly | Precisely]
[Group_teachers | Area | Department]
[Group_students | Class | Student_type],
[Room_type | Room_Group | Buildings]
[Double_bookings | Clashes | Conflicts]
```

A definição de sinónimos permite a definição de ficheiros de entrada com uma leitura mais fácil para os utilizadores. É também possível que os utilizadores construam as suas próprias listas de sinónimos num ficheiro separado, e com a ajuda de um simples "pré-processador" o ficheiro possa ser convertido para esta linguagem de especificação.

7.3.4 Representação do Tempo

Cada ficheiro contém um problema relacionado com determinado período de tempo, que normalmente corresponde a um ano lectivo. Como tal, o ficheiro deve incluir a identificação desse ano:

```
this is year <NAME>
```

Cada ano pode ser dividido em períodos de tempo (semestres, trimestres, períodos de exame, etc.). A linguagem *SCHEDULING UNILANG* permite a definição do número de períodos usado (para o problema) e os nomes desse períodos. Cada período pode ter uma

data inicial, dias da semana inicial e final e, é composto por um conjunto de semanas. O conceito de semana não é o mesmo de uma semana típica. Aqui interessa um conceito de semana como um período básico de escalonamento. Assim, por exemplo, num problema de calendarização de exames, uma "semana" pode durar 15 ou 20 dias, enquanto num problema de construção de horários numa universidade ou numa escola uma semana dura 5 ou 6 dias.

```
periods are {<PE>}
period {<PE>} contains weeks {<N>}
period {<PE>} begins on date <DATE>
period {<PE>} begins|ends on day <D>
```

É assumido que cada semana é composta por um conjunto de dias consecutivos e que cada dia é composto por um conjunto de tempos (*time slots*) disjuntos e consecutivos. Assim, em cada semana, temos um número finito de tempos. Cada tempo pertence a determinado dia e começa em determinado tempo.

```
slots are {<S>}
days are {<D>}
times are {<T>}
slot <S> is on day <D> at time <T>
```

Alguns tempos não podem ser usados para eventos no horário. Exemplos comuns são tempos aos Domingos, Sábados e tempos à noite. Esta linguagem permite duas formas de definir estas impossibilidades. A primeira é baseada simplesmente na omissão dos tempos impossíveis. A segunda é baseada na definição dos tempos e no estabelecimento explícito das impossibilidades de alocação. Isto permite medidas de diferenças de tempo coerentes entre tempos, que podem ser necessárias para algumas aplicações de construção de horários [Reis e Oliveira, 2000a].

```
slots {<S>} are unusable
```

Podem ser definidos períodos de tempo (como manhãs, tardes, horas de almoço, etc.) utilizando o conceito de período de tempo (*time period*). Cada período de tempo é composto por um conjunto de tempos numa semana. Esta sintaxe inclui o termo *default* que permite a atribuição de valores a todos os elementos de uma dada classe.

```
time_periods are {<TP>}
time_period {<TP>}|default contains slots {<S>}
```

Utilizando a palavra chave *default* as restrições são aplicadas a todos os elementos pertencentes à classe dada. Isto pode também ser aplicado a qualquer restrição que lida com *slots* de tempo, salas, docentes, alunos e grupos (de docentes ou alunos).

Para cada tempo, pode ser definida uma capacidade em termos de eventos e lugares. Estas capacidades são importantes para permitir a definição de problemas de construção de horários, que não tenham associados problemas de alocação de salas. Por outro lado, se a

alocação de salas é uma parte do problema completo a resolver, não é necessário definir explicitamente a capacidade dos *slots*.

```
slots {<S>}|default have capacity <N> seats|events
```

Para definir que o problema da alocação de tempo para os eventos deve ser resolvido, foi definida a seguinte sintaxe:

```
solve class-teacher timetabling
solve course timetabling
solve examination timetabling
```

Isto permite a resolução das três principais classes dos problemas de construção de horários. Desta forma, qualquer sistema pode facilmente seleccionar as restrições apropriadas e medidas de qualidade para um dado problema de construção de horários.

7.3.5 Representação do Espaço

A representação do espaço é baseada na ideia de que habitualmente cada evento necessita de uma sala, mas em alguns casos necessita de mais do que uma. Dado um número de salas disponível, cada qual com a sua identificação própria, poderá ser representado da seguinte forma:

```
rooms are {<R>}
```

Cada sala pode suportar um dado número de alunos num determinado tempo. Da mesma forma, cada sala pode suportar somente um determinado número de eventos (normalmente um) por cada tempo. Se a preferência for omitida, as restrições tornam-se rígidas (*hard constraints*). Se a preferência for incluída, então as restrições podem ser violadas com um custo P.

```
room {<R>}|default holds <N> students|events [preference <P>]
```

O número de alunos e eventos que uma sala pode suportar simultaneamente pode ser diferente ao longo dos tempos semanais:

```
room {<R>}|default holds <N> students|events in slots {<S>}
[preference <P>]
```

Normalmente, não é permitido que as salas possuam sobreposição de eventos e podem encontrar-se indisponíveis ou com preferência de disponibilidade em alguns *slots*:

```
room {<R>}|default cannot have double_bookings [preference <P>]
room {<R>}|default specify|excludes in slots {<S>} [preference <P>]
```

Esta especificação permite, também, a definição de tipos de salas. Cada sala possui um determinado tipo ou conjunto de tipos:

```
room_types are {<RT>}
room_type {<RT>}|default has rooms {<R>}|default [preference <P>]
```

Foi introduzido também o conceito de distância entre as salas. Cada sala pode ser ligada (se a preferência for omitida) ou pode encontrar-se próxima de outra sala, com uma determinada medida de proximidade:

```
room {<R>} close to room {<R>} [preference <P>]
```

Para definir que o problema a resolver inclui a atribuição de salas a eventos, temos a seguinte sintaxe:

```
solve room assignment
```

7.3.6 Descrição de Eventos

O conceito de evento é crucial nesta especificação. Um evento pode ser uma aula, um exame, almoço, etc. Tem uma dada duração (em termos de *slots* de tempo), pode ou não necessitar de espaço (salas) e recursos como professores (ou vigilantes) e alunos (ou grupos de alunos).

O primeiro passo é definir os eventos (identificadores de eventos) e, eventualmente, alguns grupos de eventos (como cursos, anos de cursos, programas, etc.).

```
events are {<E>} [in period <PE>]
groups_events are {<GE>}
```

Os eventos podem ter durações, por defeito, mas pode ser atribuída uma duração individual diferente e os alunos que podem/vão assistir ao evento têm de ser definidos:

```
event {<E>}|default lasts <N> slots
event {<E>} has students {<ST>} [preference <P>]
```

Além do número de alunos esperado, um evento pode ter alguns (anónimos) alunos extra. O número total de alunos do evento pode também ser definido directamente. Isto pode ser útil, nos casos em que os alunos não vão ser tratados individualmente. Alguns eventos podem necessitar de um determinado número de docentes ou salas (anónimos). Isto pode ser útil, se não estamos preocupados com os problemas de alocação de docentes ou salas.

```
event {<E>} has <N> students|extra_students
events {<E>}|default requires <N> teachers [preference <P>]
events {<E>}|default requires <N> rooms [preference <P>]
```

Os eventos podem pertencer a grupos de eventos e podem ser separados em diferentes partes e durações (em termos de tempos), que irão ocorrer em dias diferentes da semana.

```
group_events {<GE>} has events {<E>} [preference <P>]
event {<E>}|default has <N> event_parts of duration {<N>}
```

Normalmente, um evento necessita de um determinado tipo de sala, que pode ser obrigatório ou flexível:

```
events {<E>}|default requires room_type {<RT>} [preference <P>]
```

Uma restrição típica, relacionada com partes de eventos é que elas não devem ter lugar no mesmo dia. Os eventos podem ser divididos em múltiplas ofertas, o que significa que, diferentes docentes podem repeti-las a diferentes alunos, durante a semana. Por exemplo, um exame pode ser dividido em duas partes (uma parte teórica e uma prática) e em três ofertas (uma para a turma A, uma para a turma B e outra para a turma C). Um módulo é normalmente dividido em aulas (partes) que são leccionadas em dias diferentes da semana e podem conter múltiplas ofertas.

```
event {<E>}|default minimum/maximum <N> students
event {<E>}|default has <N> event_sections
```

Se as ofertas de eventos não estiverem definidas e o problema da definição de secções tiver de ser resolvido, temos então:

```
solve section definition
```

Existem dois tipos diferentes de preferências temporais dos eventos: observando tempos e períodos de tempo (manhãs, tardes, etc.). Estas preferências incluem pré-alocações de especificação e exclusão ou tentativa de evitar (com preferência).

```
event {<E>}|default specify|excludes slots {<S>} [preference <P>]
event {<E>}|default specify|excludes time_period {<TP>} [preference
<P>]
```

As preferências de espaço dos eventos são semelhantes às preferências de tempo. Temos dois tipos: preferências de sala e preferências de tipo de sala.

```
event {<E>}|default specify|excludes rooms {<R>} [preference <P>]
event {<E>}|default specify|excludes room_types {<RT>} [preference
<P>]
```

7.3.7 Turmas e Alunos

O número e nomes (identificadores) dos alunos e dos grupos de alunos podem ser definidos exactamente da mesma forma que os nomes dos *slots*, salas e eventos:

```
students are {<ST>}
group_students are {<GST>}
```

Os alunos podem ser agrupados em grupos (com preferências de escalonamento ou planos curriculares comuns (ou pelo menos similares):

```
group_students {<GST>} have students {<ST>}|default [preference <P>]
group_students {<GST>} has <N> students
```

Um aluno pode também ser um docente. Este é o caso de alunos de pós-graduação que também são docentes de colegas com formação inferior.

```
student <ST> is teacher <TE>
```

Alunos ou grupos de alunos podem estar indisponíveis em alguns *slots*, ao longo da semana. Isto pode ser considerado uma restrição rígida ou flexível:

```
student|group_students {<ST>|<GST>} default specify|excludes slots
{<S>}|default [preference <P>]
```

Alunos e grupos de alunos podem assistir a eventos:

```
event {<E>}|default has students|group_students {<ST>|<GST>}|default
[preference <P>]
```

Um aluno ou grupo de alunos, para além de se encontrar inscrito num evento pode também estar inscrito numa das suas ofertas. As preferências podem também ser usadas na alocação de alunos e grupos de alunos a eventos:

```
event_section {<ES>} has students {<ST>} [preference <P>]
event_section {<ES>} has group_students {<GST>} [preference <P>]
```

No sentido de evitar que alunos (ou grupos de alunos) tenham sobreposições de eventos, podemos dizer que:

```
student|group_student {<ST>|<GST>}|default cannot have double_bookings
[preference <P>]
```

Isto pode ser uma restrição flexível (com um dado custo de violação <P>) ou rígida (se a palavra-chave não for usada). Se o problema de alocação de alunos a ofertas de eventos for considerado, isso deverá ser definido usando:

```
solve student scheduling
```

7.3.8 Docentes e Vigilantes

Neste modelo, os docentes podem ser vistos também como vigilantes (e a palavra-chave *teaches* pode também significar vigia). Os docentes podem ser agrupados em áreas ou grupos de docentes. Isto permite a definição de departamentos e áreas científicas.

```
teachers are {<TE>}
group_teachers are {<GTE>}
group_teachers {<GTE>}|default has teachers {<TE>}|default
```

Cada docente pode, previamente, ser escalonado para leccionar um dado número de eventos (pré-alocações). A informação de cada docente é também associada com a sua capacidade de leccionar um dado evento.

```
teacher {<TE>} teaches|cannot_teach events {<E>} [preference <P>]
```

Para permitir a modelação de problemas de alocação de docentes e problemas de atribuição de vigilâncias é necessário um número mínimo e máximo de eventos (ou tempos) para os docentes.

```
teacher {<TE>}|default maximum|minimum <N> events|times [in period
<PE>]
```

Os docentes ou grupos de docentes podem ter preferências de horário. Um docente (ou grupo) pode excluir (ou evitar com alguma preferência) alguns tempos da semana:

```
teacher|group_teachers {<TE>|<GTE>}|default specify|excludes slots
{<S>}|default [preference <P>]
```

Normalmente, os docentes não podem ter sobreposições de aulas e, por vezes, dependendo do significado de grupos de docentes, estes não podem, também, ter sobreposições de aulas.

```
teacher|group_teachers{<TE>|<GTE>}|default cannot have double_bookings
[preference <P>]
```

Se o problema inclui a atribuição de docentes a eventos (ex: alocação de docentes, atribuição de vigilâncias na calendarização de exames, etc.) então, deverá ser incluída a seguinte linha:

```
solve teachers assignment
solve invigilator assignment
```

7.3.9 Restrições de Carga de Trabalho, Espalhamento e Ordenação

As restrições de carga de trabalho e espalhamento, inicialmente, estão relacionadas com a utilização de docentes e alunos ao longo do período de escalonamento. Os alunos e grupos de alunos podem ter restrições carga de trabalho, estabelecendo que eles não podem ter mais do que um dado número de eventos ou tempos de trabalho consecutivos ou num dado dia. Normalmente, isto aplica-se à maioria das restrições exactamente (*exactly*) e pelos menos (*at least*) e a algumas restrições que podem também ser usadas em alguns problemas menos frequentes. Estas restrições podem também ser aplicadas a docentes:

```
students|student_group{<ST>|<GST>}|default have exactly|atleast|atmost
<N> [consecutive] events|times in a day [preference <P>]
```

```
teacher {<TE>}|default have exactly|atleast|atmost <N> [consecutive]
events|times in a day [preference <P>]
```

Restrições de espalhamento são típicas em problemas de *Calendarização de Exames*. Estas restrições reflectem a preocupação de que os alunos (ou grupos de alunos) tenham tempo suficiente entre eventos. Normalmente, são restrições "*at least*".

```
students|student_group{<ST>|<GST>}|default have exactly|atleast|atmost
<N> times|days between events [preference <P>]
```

Por vezes, restrições de espalhamento estabelecem que os alunos têm no máximo um dado número de eventos (ou tempos ocupados) em cada número de dias (ou tempos):

```
students|student_group{<ST>|<GST>}|default have exactly|atleast|atmost
<N> events|times in each <N> times|days [preference <P>]
```

As restrições de ordenamento estão relacionadas com a ordem de determinados eventos no período de escalonamento. Um evento pode ser escalonado exactamente, pelo menos ou no máximo, um dado número de tempos (ou dias) antes de outro evento. Outro tipo habitual de restrição estabelece que existem "exactamente"/"pelo menos"/"no máximo" um dado número de tempos ou dias de intervalo entre os dois eventos (sem qualquer preocupação com o evento que surge primeiro).

```
events {<E>} exactly|atleast|atmost <N> times|days before|interval
```

Capítulo 8

8. Análise do Domínio de Aplicação: Futebol Robótico

Dos domínios de aplicação seleccionados para testar as estratégias de coordenação de agentes desenvolvidas, o Futebol Robótico com ênfase na liga de simulação, é aquele a que vamos dar maior relevância. A iniciativa RoboCup [Kitano et al., 1995] [Kitano, 1997] é um projecto de investigação e educação internacional com o objectivo de promover a investigação em Inteligência Artificial (Distribuída) e Robótica Inteligente [RoboCup, 2001]. O projecto baseia-se na utilização de um problema standard – o futebol robótico – onde um elevado conjunto de tecnologias é necessário para possibilitar a construção de uma equipa de Robôs reais ou virtuais capaz de participar num desafio de futebol, jogado de acordo com um determinado conjunto de regras pré-especificado.

O futebol robótico inclui diversas ligas que se dividem em dois tipos: ligas robóticas (utilizando robôs pequenos, médios, cães e humanóides) e a liga de simulação. Cada liga coloca um conjunto de desafios de investigação próprio e tem ênfase em determinados tópicos necessários para efectivamente colocar equipas de robôs a disputar uma partida de futebol. Por exemplo, na liga de simulação, a ênfase é colocada na coordenação em SMA, enquanto na liga de robôs pequenos, a ênfase é colocada no controlo rápido e preciso dos robôs e na liga de robôs médios os tópicos mais importantes incluem a visão computacional, projecto electromecânico e auto-localização dos robôs.

Existem também outros desafios associados ao futebol robótico que a iniciativa RoboCup promove. De entre estes destacam-se o *RoboCup Rescue* [Kitano et al., 1999] e o *RoboCup Júnior* [Sklar et al., 2002]. O *Rescue* tem como objectivo estimular a aplicação da investigação realizada no futebol robótico, a domínios socialmente mais úteis, no caso, missões de salvamento e resgate em grandes catástrofes [Kitano et al., 1999] [Rescue, 2001]. Divide-se em duas ligas: robótica e simulação. O *RoboCup Júnior* surgiu como uma forma de estimular os mais jovens a participar no *RoboCup* [Sklar et al., 2002]. Providencia as condições que permitem a jovens em idade escolar, construir e colocar em funcionamento os seus robôs para realizar diversas tarefas.

Neste trabalho um destaque especial é dado à liga de simulação do *RoboCup*. Esta liga é baseada no sistema de simulação *soccerserver* [Noda, 1995] [Noda et a., 1998] que permite a duas equipas de agentes autónomos disputar um jogo de futebol simulado. O *soccerserver* providencia um domínio muito realístico, no sentido em que inclui muitas complexidades do futebol real e dos sistemas robóticos tais como erros nos sensores e actuadores e energia limitada [Noda e Stone, 2002].

Este capítulo é importante para perceber o trabalho apresentado no capítulo nove relativo à proposta de metodologias de coordenação para equipas de agentes. A complexidade geral do RoboCup e em particular do *soccerserver* implica a necessidade de uma descrição prévia do concurso e da liga de simulação, introduzindo os desafios de investigação e terminologia própria do sistema *soccerserver* para que possa ser compreendido o trabalho realizado utilizando-o como plataforma de teste.

Este capítulo está organizado da seguinte forma. A secção 8.1 apresenta uma descrição da iniciativa *RoboCup*, dos seus objectivos e história. Na secção 8.2 é apresentada uma panorâmica sobre as diversas ligas do *RoboCup* e os desafios de investigação que colocam. Na secção 8.3 é apresentado o futebol robótico simulado e é detalhado o funcionamento do simulador (*soccerserver*) que permite realizar jogos de futebol robótico simulado. A secção seguinte descreve resumidamente a investigação realizada por diferentes grupos de investigação utilizando os problemas colocados pelo RoboCup. Em seguida é apresentado o trabalho relacionado e são descritas outras aplicações da investigação realizada no âmbito da liga de simulação do *RoboCup*. O capítulo conclui-se com a discussão das vantagens e perigos da realização de competições científicas como o RoboCup e com algumas conclusões sobre este domínio.

8.1 Introdução

8.1.1 Objectivos do RoboCup

O objectivo da iniciativa *RoboCup* é promover a investigação em Robótica Inteligente e Inteligência Artificial através de um desafio que seja simultaneamente estimulante do ponto de vista científico, colocando um vasto conjunto de problemas científicos aos investigadores da área mas, ao mesmo tempo, extremamente atraente para o público em geral e para os meios de comunicação social. Com esta vertente, pretende-se que o *RoboCup* seja capaz de chamar a atenção do público e comunicação social, não só para o *RoboCup* como competição por si só, mas também para a investigação realizada em Inteligência Artificial e Robótica Inteligente pelos diversos laboratórios participantes.

Como forma de promover a investigação na área, foi lançado um objectivo de longo prazo:

"No ano de 2050, uma equipa de robôs autónomos humanóides, ser capaz de vencer a equipa campeã do mundo de futebol, num encontro disputado de acordo com as regras da FIFA." [Kitano, 1997]

Este objectivo é actualmente partilhado como um dos grandes desafios da comunidade, pelos investigadores da área da Inteligência Artificial e Robótica a atingir durante os próximos 50 anos. Embora este desafio, à luz da ciência e tecnologia actuais, pareça altamente ambicioso, a colocação de objectivos científicos bem definidos de longo prazo, tem sido, ao longo dos anos, uma forma de estimular o desenvolvimento científico [RoboCup, 2001]. Um exemplo marcante disto surgiu com o projecto *Apollo* e com o seu objectivo de longo prazo:

"Aterrar um homem na Lua e fazê-lo regressar, são e salvo à Terra" [Kennedy, 1961].

Embora o alcance deste objectivo de longo prazo tenha constituído um marco na história da humanidade, por si só não ofereceu um grande impacto económico ou social. As tecnologias desenvolvidas para conseguir alcançar este objectivo formaram uma base sólida para enormes progressos económicos e sociais. No caso do projecto *Apollo*, a motivação dada pela ambição e abrangência do objectivo final, deu origem a avanços tecnológicos e científicos que foram a base das fundações humanas e tecnológicas da poderosa indústria Americana dos dias de hoje [RoboCup, 2001].

No projecto *Apollo*, embora o objectivo fosse aterrar um homem na Lua, a sua abrangência era muito maior. Neil Armstrong após aterrar na Lua, afirmou: *"É um pequeno passo para o homem, mas um salto gigante para a humanidade!"*. Esta frase mostrou o cumprimento do objectivo inicial, mas os objectivos do projecto *Apollo* iam muito para além de aterrar um humano na Lua: Desenvolver tecnologia que permitisse estabelecer interesses Americanos no espaço; dar a proeminência no Espaço aos EUA; desenvolver um programa alargado de pesquisa na Lua; desenvolver a capacidade humana para trabalhar na Lua e em outros planetas, etc.

Embora o objectivo de longo prazo do *RoboCup* seja o desenvolvimento de uma equipa capaz de vencer o campeão do mundo humano de futebol, outros objectivos mais modestos podem ser estabelecidos com base neste, como por exemplo, conseguir criar uma equipa de robôs que jogue tal como uma equipa de humanos ou, criar robôs capazes de jogar um jogo de futebol contra um equipa de humanos. Para além disto, o objectivo final da iniciativa *RoboCup*, embora impossível no curto-prazo em face da ciência e tecnologia actuais, irá criar um conjunto de sub-objectivos que levarão até ao objectivo final. Entre estes sub-objectivos, destaca-se a criação de equipas de robôs reais e virtuais, de diversos tipos e com características distintas, que sejam capazes de jogar razoavelmente com regras modificadas, no sentido de as simplificar, relativamente às da FIFA [FIFA, 2001].

Um outro desafio semelhante colocado aos investigadores em Inteligência Artificial no decurso das últimas 4 décadas, consistiu em construir um agente (programa) que fosse

capaz de vencer o campeão mundial de Xadrez utilizando as regras oficiais da Federação Internacional de Xadrez. Este desafio para além de promover a investigação em Inteligência Artificial, mostrou a importância da existência de problemas *standard* em que diferentes metodologias e avanços científicos podem ser comparados. Esta faceta de problema *standard* foi extremamente importante no Xadrez. Diversos algoritmos de pesquisa, arquitecturas de computadores e metodologias científicas foram desenvolvidos para este domínio. Em Maio de 1997, o computador Deep Blue da IBM [Deep Blue, 1997] derrotou Gary Kasparov (o campeão humano de Xadrez), utilizando as regras oficiais do Xadrez [Deep Blue, 1997]. Com esta vitória, o desafio de 40 anos da Inteligência Artificial utilizando o Xadrez como domínio de aplicação ficou muito próximo de um final com sucesso.

Uma das características que tornou o Xadrez computorizado um problema *standard* foi a facilidade que este domínio ofereceu para a comparação de abordagens distintas, e para a avaliação do progresso científico global realizado no domínio. No entanto, com a chegada ao final do desafio associado ao Xadrez, no mundo da IA, novos domínios e problemas *standard* mais complexos e estimulantes tornaram-se necessários. Foi neste contexto que se desenvolveu o desafio do futebol robótico como um problema *standard* para a IAD e robótica inteligente.

Uma característica essencial do RoboCup consiste em visualizá-lo também como um problema *standard* em que várias teorias, arquitecturas, metodologias e algoritmos podem ser avaliadas e facilmente comparadas entre si. O domínio preenche inteiramente as características necessárias para ser o sucessor do Xadrez como problema *standard* da IA e domínio para a comparação de diversas metodologias da IA e Robótica [Reis, 2001a]. As diferenças principais entre o domínio do Xadrez e o *RoboCup* podem ser visualizadas na tabela 9.

Domínio / *Característica*	***Xadrez***	***RoboCup***
Ambiente	Estático	Dinâmico
Mudança de Estado	Por turnos	Tempo-Real
Acessibilidade ao Estado do Mundo	Completa	Incompleta
Resultado das Acções	Determinístico	Não Determinístico
Leitura dos Sensores	Discreta (Simbólica)	Contínua (não simbólica)
Utilização dos Actuadores	Discreta (Simbólica)	Contínua (não simbólica)
Controlo	Centralizado	Distribuído

Tabela 9: Diferenças entre as características dos domínios do RoboCup e Xadrez.

Analisando a tabela 9 é visível que o *RoboCup* constitui um domínio que inclui características bastante mais complexas que o Xadrez. Uma das diferenças principais reside no facto do seu ambiente ser dinâmico. Enquanto os agentes (jogadores virtuais ou robóticos) decidem a próxima acção a executar, a bola encontra-se em movimento, assim como os agentes colegas de equipa e os agentes adversários. No Xadrez, todos os objectos estão estacionários enquanto os agentes decidem a próxima acção a executar.

No Xadrez a mudança de estado é realizada por turnos (jogadas alternadas de ambos os jogadores). No *RoboCup* todos os jogadores (de ambas as equipas) podem agir ao mesmo tempo no mundo.

A acessibilidade ao estado do mundo é completa no Xadrez. No *RoboCup* a acessibilidade ao estado do mundo é dada unicamente através dos sensores de cada robô real ou virtual. A capacidade destes sensores é limitada, pelo que só uma parcela muito reduzida do estado do mundo se encontra acessível a cada momento aos robôs. Por exemplo, no caso da visão, quer o seu alcance quer o seu ângulo de abertura são limitados, conduzindo a que unicamente uma parcela do campo, usualmente muito reduzida, seja visível em cada instante. Para obter uma perspectiva global do campo e do posicionamento dos diversos objectos neste, é necessário procurar construir um estado do mundo global (representação interna do conhecimento sobre o estado do mundo, que integre os valores recebidos dos sensores no passado com os valores actuais) partindo das visões parciais obtidas em cada instante.

Outra complexidade adicional do ambiente do *RoboCup* consiste no facto de as acções efectuadas pelos agentes não terem um efeito único e garantido, ou seja, o ambiente não ser determinístico. No xadrez a realização de uma dada jogada tem um efeito único e garantido e não existe qualquer possibilidade de falha. Ao contrário, no futebol robótico, a realização de uma dada acção, por exemplo chutar a bola com uma dada potência numa dada direcção, não tem um efeito totalmente previsível. Pode acontecer que o agente nem sequer consiga chutar a bola. Mesmo no caso de o chuto ser bem sucedido, a potência e direcção do chuto efectivamente aplicadas à bola não são totalmente previsíveis.

O estado do mundo, a leitura dos sensores e a utilização dos actuadores são compostos por variáveis discretas no Xadrez. No entanto, no *RoboCup*, todas estas variáveis são contínuas, tornando assim este domínio bastante mais complexo. O número de percepções possível, estados admissíveis do mundo e acções possíveis de executar são, em cada instante, virtualmente infinitas no RoboCup. Contribuindo também para este aumento de complexidade, encontra-se o facto de os valores serem lidos pelos sensores, incluindo erros consideráveis e as ordens enviadas aos actuadores serem também executadas com erros significativos.

A diferença final entre os domínios consiste no facto de o controlo no *RoboCup* ser em geral distribuído. Dependendo da modalidade, um número elevado de agentes autónomos

(que pode variar entre 4 e 11 nas ligas de futebol robótico), tem de agir autonomamente e coordenarem-se de forma a atingir um objectivo comum.

O *RoboCup* foi desta forma projectado de forma a colocar num mundo limitado, um conjunto elevado de complexidades do mundo real, mantendo no entanto o custo, complexidade global e dimensão do problema, acessível aos grupos de investigação em Robótica e Inteligência Artificial. Os problemas de investigação colocados pelo *RoboCup* de uma forma integrada, cobrem uma vasta área dos domínios da IA e Robótica, incluindo: coordenação, cooperação e comunicação multi-agente, arquitecturas de agentes inteligentes, aprendizagem, planeamento em tempo-real, decisão estratégica e táctica, comportamento reactivo, visão por computador, processamento e análise de imagem, sistemas de locomoção e actuação, sistemas sensoriais, fusão sensorial em tempo-real, navegação, controlo inteligente robótico, etc.

8.1.2 História do RoboCup

A ideia do futebol robótico foi introduzida por Alan Mackworth da Universidade de British Columbia, Canadá em 1992[41] [Mackworth, 1993]. De forma independente, um grupo de investigadores Japoneses de IA, organizaram um *Workshop* intitulado "Os Grandes Desafios da IA" em Outubro de 1992 em Tóquio. Neste *Workshop* foram discutidas diversas questões sobre a utilização do futebol na promoção da investigação realizada em Robótica e IA, realizado um estudo financeiro, tecnológico e do impacto social desta utilização. Foram também definidos protótipos de robôs futebolistas e um primeiro projecto de um simulador de futebol. O resultado do estudo foi o lançamento em Junho de 1993 de uma competição robótica intitulada *Robot J-League*[42] que após as excelentes reacções de grupos de investigação fora do Japão, nomeadamente grupos de investigação Americanos, foi estendida para um projecto internacional conjunto com o nome *Robot World Cup Initiative – RoboCup.*

Paralelamente a esta discussão, diversos investigadores estavam já a utilizar o futebol robótico como domínio da sua investigação. Itsuki Noda no ElectroTechnical Laboratory (ETL) do Japão estava a desenvolver um simulador de jogos virtuais de futebol que, posteriormente daria origem ao *soccerserver* [Soccerserver, 2001], o simulador oficial da liga de simulação do RoboCup. Minoru Asada e seus colaboradores na Universidade de Osaka e Manuela Veloso e Peter Stone na Carnegie Mellon University estavam também envolvidos na criação de robôs futebolistas.

---

[41] O artigo original "On Seeing Robots", embora datado de 1992, foi publicado posteriormente em 1993 no livro Computer Vision: Systems, Theory and Applications, pp. 1-13.

[42] A J-League é a liga profissional de futebol no Japão

Em Setembro de 1993, o primeiro anúncio oficial desta iniciativa foi realizado e regras específicas foram criadas para diversas ligas de futebol robótico. Itsuki Noda no ETL lançou a versão 0 do *SoccerServer* (desenvolvida em LISP), e depois a versão 1.0, desenvolvida em C++. A primeira demonstração pública deste sistema foi realizada na conferência IJCAI'95[43], que teve lugar em Agosto de 1995 em Montreal, Canadá. No decurso desta conferência foi também realizado o anúncio oficial da organização do primeiro *RoboCup – Robot World Cup Soccer Games and Conferences*, em conjunção com a IJCAI'97 em Nagoya no Japão. Ao mesmo tempo foi decidido realizar um Pre-RoboCup em 1996 para identificar e avaliar os possíveis problemas da organização do RoboCup97.

O Pre-RoboCup96 realizou-se durante a IROS'96[44] em Osaka em Novembro de 1996 com oito equipas na modalidade de simulação e uma demonstração de robôs médios. A primeira competição RoboCup realizou-se em Nagoya em 1997 com mais de 40 equipas participantes. Mais de 5 mil espectadores assistiram aos jogos [Robocup, 2001] num dos maiores eventos científicos de sempre.

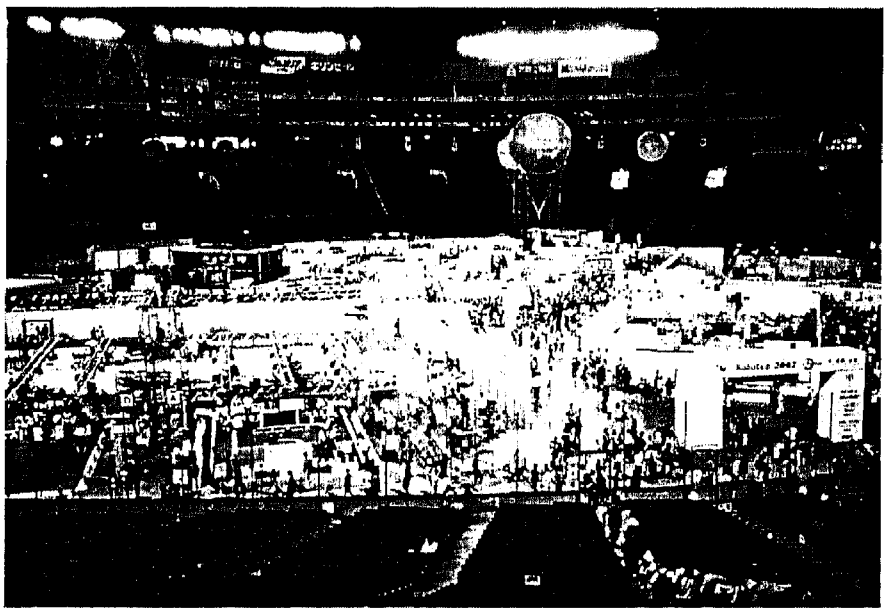

Figura 45: Aspecto Geral do Fukuoka Dome – RoboCup2002

As cinco edições seguintes do *RoboCup* disputaram-se em Paris 1998, Estocolmo 1999, Melbourne 2000, Seattle 2001 e Fukuoka 2002. Para além destas competições, diversas outras competições regionais ou nacionais tiveram lugar. De entre estas destacam-se o *Japan Open* (disputado em 2000, 2001 e 2002), o campeonato Europeu (disputado unicamente em 2000 em Amesterdão), o *German Open* (2001 e 2002) e o *China Open* (2000, 2001 e 2002). As últimas edições do *RoboCup* transformaram-se em grandes

[43] IJCAI – International Joint Conference on Artificial Intelligence

[44] IROS – International Conference on Intelligence Robotics and Systems

eventos científicos com centenas de participantes e milhares de espectadores. No RoboCup 2002 em Fukuoka participaram 288 equipas, com mais de 1000 investigadores envolvidos e assistiram ao certame um total de mais de 110 mil espectadores (figura 45).

8.1.3 A Federação do RoboCup

A federação do *RoboCup* (*RoboCup Federation*) é uma associação internacional, registada na Suíça, que tem como principiais objectivos organizar o esforço internacional para a promoção da ciência e Tecnologia, utilizando jogos de futebol com robôs e agentes de *software* [RoboCup, 2001].

As funções principais da Federação *RoboCup* estão ligadas à organização anual de um campeonato do mundo e à coordenação dos esforços globais de todos os investigadores da área. Os membros da Federação são todos investigadores activos da área e representam um alargado número de Universidades e grandes empresas. Como o número de investigadores envolvido no projecto é muito alargado e encontra-se espalhado por todo o globo, diversos comités regionais estão formados para promover a investigação do *RoboCup* em áreas geográficas localizadas.

A federação é composta por um Presidente, um conjunto de *Trustees*, um comité executivo, um comité de aconselhamento e comités técnicos (compostos por sete elementos cada) para cada uma das ligas. Comités específicos são designados pela Federação para os eventos organizados tais como o campeonato mundial de *RoboCup*.

Os comités técnicos das diversas ligas têm a incumbência de definir o futuro de cada uma das ligas. Têm também como função iniciar e guiar a discussão referente à alteração das regras da respectiva liga, efectuada entre os membros da comunidade. Para além desta função, os comités técnicos são também responsáveis por tomar as decisões finais relativas às regras a utilizar, em cada liga, em cada ano. Embora inicialmente os comités técnicos das várias ligas fossem constituídos por membros designados pelo comité executivo, em 2001, a eleição dos membros dos comités foi democratizada. Desta forma, cada comité técnico passou a ser constituído por sete membros, quatro dos quais designados pelo comité executivo e três eleitos democraticamente pela comunidade de investigação na liga respectiva do *RoboCup*[45].

[45] O autor é desde Novembro de 2001, membro do comité técnico da Liga de Simulação do RoboCup. É o único elemento do comité técnico da liga de simulação que foi eleito em todas (duas) as eleições realizadas (2001 e 2002) para este órgão da federação RoboCup.

8.2 As Ligas do RoboCup

O desafio colocado pelo *RoboCup* aos investigadores mundiais das áreas da IA e Robótica subdivide-se em 3 categorias principais (figura 46), sendo que, o Futebol Robótico é claramente a categoria mais relevante:

- ***RoboCup Soccer*** – Futebol Robótico, incluindo uma liga de simulação e três ligas robóticas principais com regras distintas;
- ***RoboCup Rescue*** – Aplicação da investigação realizado no futebol robótico ao domínio da busca e salvamento em grandes catástrofes. A categoria divide-se numa modalidade de simulação e uma modalidade robótica;
- ***RoboCup Júnior*** – Utilização do *RoboCup* na educação. Crianças e jovens utilizando uma infra-estrutura simples, criam equipas de robôs para jogar futebol (2 contra 2), dançar ou resgatar vítimas.

Para além destas actividades e da organização anual do campeonato mundial de futebol robótico, a iniciativa RoboCup conta ainda com a realização de conferências especializadas, programas educacionais e desenvolvimento de infra-estruturas diversas.

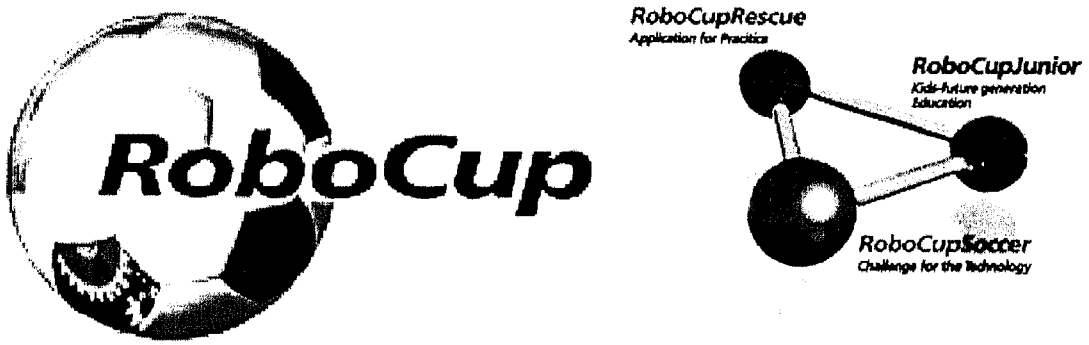

Figura 46: As Três Vertentes do RoboCup

O Futebol robótico é sem dúvida o pilar central do *RoboCup*. Divide-se em quatro ligas principais e diversos desafios associados [RoboCup, 2001]:

- **Liga de Simulação** – Equipas de 11 agentes de *software* competem num ambiente simulado, jogando um desafio de futebol virtual, utilizando o sistema de simulação *soccerserver*;
- **Liga de Robôs Pequenos** – *Small-Size* – Equipas de 5 robôs de reduzidas dimensões, com controlo e visão centralizados, competem num campo com dimensões semelhantes às de uma mesa de *ping-pong*;
- **Liga de Robôs Médios** – *Middle-Size* – Equipas de 4 robôs totalmente autónomos competem num campo de 5x10 metros;
- **Liga de Robôs com Pernas** – *Sony Legged* – Equipas de 4 robôs cães Aibo da

Sony [Yamamoto e Fujita, 2000] jogam num campo de reduzidas dimensões.

Para além das ligas principais do *RoboCup*, existem diversos desafios associados:

- **Liga de Robôs Humanóides** – Desenvolvimento de robôs humanóides capazes de a médio prazo, jogar um desafio de futebol. Para já, têm existido unicamente demonstrações e competições de velocidade de deslocamento bípede e *penalties*.
- **Treinador da Simulação** – Desenvolvimento de sistemas de análise de jogo que sejam capazes de alterar o comportamento de uma equipa, através de instruções para os seus constituintes, tendo em consideração o decurso do jogo;
- **Comentador Inteligente** – Desenvolvimento de sistemas de análise de jogo e comentadores inteligentes com capacidade de geração de linguagem natural;
- **Visualizador 3D** – Criação de visualizadores tridimensionais para jogos de simulação, incluindo animações e sons realistas;

Todos os anos, de forma a aferir o progresso científico realizado pelos vários grupos de investigação, nos diversos tópicos mencionados, a Federação *RoboCup* organiza o *Robot World Cup Soccer Games and Conferences*, vulgarmente conhecido por *RoboCup*. Competições relacionadas com as diversas ligas e desafios, demonstrações e um congresso científico [Kitano, 1998] [Asada e Kitano, 1999] [Veloso et al., 2000] [Stone et al., 2001a] [Birk et al., 2002] (descrito no anexo 3) compõem o programa deste evento. Existem ainda diversas competições e eventos regionais relacionados com o *RoboCup* tais como o campeonato Europeu, o *German Open*, o *Japan Open*, o *China Open*, etc.[46].

As equipas portuguesas têm obtido muito bons resultados no âmbito do RoboCup, destacam-se a vitória num campeonato do Mundo e dois campeonatos da Europa da liga de simulação por parte da equipa FC Portugal, a vitória num campeonato, a vitória num campeonato da Europa da liga de robôs pequenos por parte da equipa 5dpo [Costa et al., 2000] e dois terceiros lugares em campeonatos da Europa da liga de robôs médios por parte da equipa 5dpo [Moreira et al., 2002].

8.2.1 A Liga de Simulação

A liga de simulação é baseada no sistema de simulação de utilização livre *soccerserver* [Noda et al., 1998] [SoccerServer, 2001]. Este sistema simula um campo contendo 2 equipas e uma bola onde se realiza um jogo 2D de futebol. Cada equipa é composta por 11 jogadores (e eventualmente 1 treinador) que se conectam ao simulador, numa arquitectura cliente-servidor, através de *sockets* UDP. O simulador aceita comandos de baixo-nível dos

[46] Em 2003 surgiram diversas novas competições regionais incluindo o Australian Open, American Open, Iranian Open e o festival Robótica 2003 em Portugal cujo tema principal foi o futebol robótico.

jogadores, executa-os de forma imperfeita e envia informação (também imperfeita) de percepção aos jogadores.

O *soccerserver* possibilita a realização de investigação de alto-nível em coordenação, aprendizagem e planeamento multi-agente, enquanto se espera pelo desenvolvimento de hardware suficientemente robusto e apropriado, que permita efectuar este tipo de investigação com robôs reais. Os avanços na liga de simulação incluem a utilização de aprendizagem para optimizar as habilidades de baixo-nível [Stone, 1998] [Riedmiller et al., 2000], técnicas de cooperação tais como tácticas [Reis e Lau, 2003a], formações [Reis, et al., 2001] [Stone e Veloso, 1999] e papéis [Reis et al., 2001] [Stone, 1999], formalizações de estratégias de equipa complexas [Prokopenko e Buttler, 1999] [Reis et al., 2001] [Reis e Lau, 2003a], planeamento flexível [Stone, 1999], sistemas de comunicação inteligente [Stone, 1999] [Prokopenko e Buttler, 1999] [Reis e Lau 2001a, 2001], novas arquitecturas de agentes [Stone, 1998] [Reis e Lau 2001a, 2001], linguagens de comunicação para agente e para treinadores [Reis e Lau, 2002] [SoccerServer, 2001] e ferramentas de *debug* e análise de agentes [Stone, 1999] [Reis et al., 2001].

A aprendizagem tem sido um dos temas mais proeminentes de investigação no âmbito da liga de simulação [Stone et al., 2001c]. Grande parte das melhores equipas na liga de simulação tais como o CMUnited [Stone, 2000a] e os Karlsruhe Brainstormers [Riedmiller et al., 2000] têm desenvolvido os seus *skills* básicos (e até comportamentos individuais e cooperativos) com o auxílio de aprendizagem por reforço [Stone, 1998] [Riedmiller et al., 2000]. No entanto, outras equipas bem sucedidas têm demonstrado a possibilidade de desenvolver muito bons comportamentos de baixo-nível sem o recurso a aprendizagem. Exemplos disto são a equipa FC Portugal que utiliza optimização *on-line* nos seus *skills*, Tsinghuaeolus [Cai et al., 2002] e YowAI que utilizam abordagens analíticas no desenvolvimento destes *skills*.

A investigação em estratégias de equipa complexas e em mecanismos de coordenação para Sistemas Multi-Agente tem sido outra das áreas de maior destaque na liga de simulação, ao longo dos últimos anos. Técnicas de coordenação como formações e papéis foram introduzidas por Stone e Veloso [Stone, 1998] [Stone e Veloso, 1999] no *RoboCup* e conduziram à criação da equipa CMUnited que foi campeã do mundo em 1998 e 1999. Reis, Lau e Oliveira estenderam o trabalho de Stone e Veloso e criaram protocolos flexíveis para efectuar a troca de papéis e posicionamentos de agentes heterogéneos [Reis et al., 2001], introduziram o conceito de táctica no *RoboCup* [Reis et al., 2001] e criaram um mecanismo de posicionamento que permite a definição de formações muito semelhantes às utilizadas no futebol real [Reis et al., 2001]. Baseada nestes conceitos foi desenvolvida a equipa FC Portugal que foi campeã do mundo do RoboCup em 2000 [Reis e Lau, 2001a]. Este trabalho foi ainda estendido de forma a criar uma formalização genérica para o conceito de estratégia para uma competição contra um oponente com objectivos contrários [Reis et al., 2001] [Reis e Lau, 2003a]. Este modelo, aplicável também a outros domínios, constitui uma das bases para o desenvolvimento da

flexibilidade da equipa FC Portugal e para o seu sucesso no *RoboCup* [Lau e Reis, 2000] [Lau e Reis, 2002] [Reis e Lau, 2003a].

A liga de simulação, o funcionamento do sistema de simulação e a investigação realizada no âmbito desta liga vai ser objecto de uma análise mais profunda ao longo das diversas secções que compõem este capítulo. A investigação realizada no âmbito do projecto FC Portugal na liga de simulação será objecto de análise no capítulo 9.

8.2.2 A Liga de Robôs Pequenos – *Small-Size*

A liga de robôs pequenos (*small-size*) é também conhecida como a liga F-180. Cada jogo disputa-se entre equipas compostas por 5 robôs de reduzidas dimensões, que jogam num campo com alcatifa verde e marcações brancas com dimensões ligeiramente superiores às de uma mesa de *ping-pong*. As regras do jogo permitem a utilização de visão global e controlo centralizado dos robôs. Desta forma, durante o jogo, o sistema de controlo dos robôs, geralmente externo a estes, recebe a informação proveniente da câmara localizada sobre o campo, processa esta informação determinando as posições, orientações e velocidades dos robôs e da bola, decide o comando mais apropriado a ser executado por cada robô e envia este comando através de comunicação rádio a cada um dos robôs. Embora a maioria das equipas utilize visão e controlo centralizado, algumas equipas têm utilizado visão e decisão locais, o que embora dificulte a sua capacidade para jogar, torna os problemas de investigação relacionados com a visão e localização muito mais ricos.

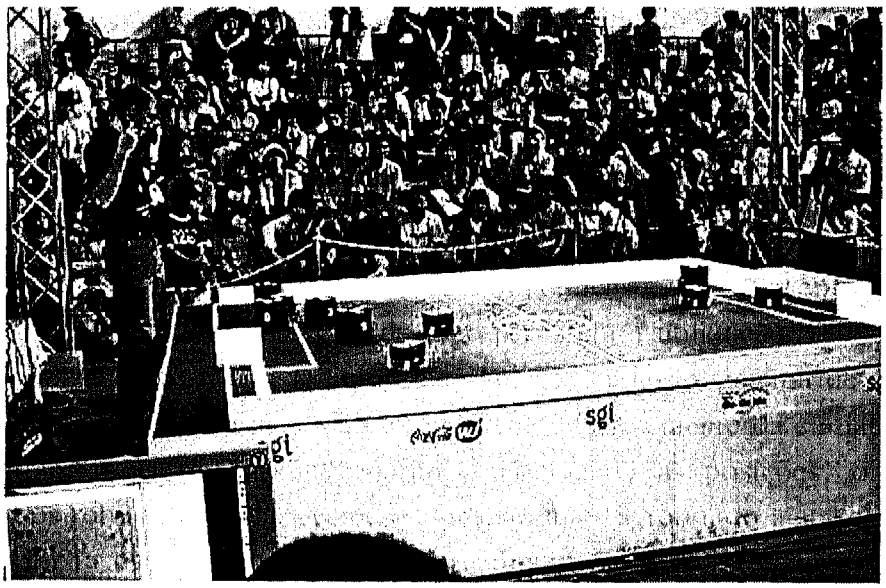

Figura 47: A liga de Robôs Pequenos (Small-Size)

A superfície de jogo é forrada com alcatifa verde delimitada por paredes brancas. Uma das balizas está pintada de azul e a outra de amarelo. Os robôs possuem marcas próprias no topo de forma a poderem ser identificados pelo sistema de visão global. Os robôs da liga de *small-size* podem ter até 18 cm segundo qualquer diagonal e ocupar, no máximo, 180

cm^2. Cada equipa pode ter um máximo de 5 robôs que, no entanto, podem apresentar características distintas. Cada jogo desta liga consiste em duas partes de 10 minutos cada. Durante o jogo nenhuma interferência humana é permitida com o sistema de controlo dos robôs.

A necessidade de alta velocidade e de um controlo preciso deram a esta liga a reputação de "a liga de engenharia" [Stone et al., 2001b]. Os desenvolvimentos científicos realizados no âmbito desta liga incluem [Stone et al., 2001b] novos sistemas electromecânicos, de controlo de sistemas, de electrónica digital e comunicações sem fios. A maioria das equipas tem investido os seus esforços de investigação nestes tópicos e unicamente algumas equipas, até 2000 (tais como os campeões em 1999 e 2000 - Cornell BigRed), demonstraram comportamentos de equipa tais como passes explícitos, papéis ou formações. No entanto, os avanços verificados ao longo dos últimos anos no hardware e a relativa estabilidade das regras do jogo contribuem para uma necessidade urgente da realização de investigação em metodologias de coordenação para ser bem sucedido nesta liga, ao longo dos próximos anos [Reis, 2001a].

8.2.3 A Liga de Robôs Médios – *Middle-Size*

A liga de robôs médios (*middle-size*) é também conhecida como a liga F-2000 e coloca um vasto conjunto de problemas de investigação que despertou o interesse de mais de 50 grupos de investigação espalhados pelo mundo. Grande parte dos problemas de investigação colocados na liga de simulação, tais como autonomia dos agentes, visão local e na liga de robôs pequenos, tais como processamento de imagem complexo, necessidade de controlo preciso em tempo-real, encontra-se simultaneamente na liga de robôs médios.

Três factores influenciam primariamente o projecto das equipas e dos seus robôs:

- O ambiente de jogo, nomeadamente o campo;
- As restrições impostas ao projecto dos robôs;
- A tarefa cooperativa a realizar, isto é, jogar futebol.

O ambiente de jogo foi projectado cuidadosamente para que os problemas relacionados com a percepção e locomoção sejam suficientemente simples mas simultaneamente constituam desafios estimulantes, que permitam despertar investigação séria no seio desta liga [Veloso et al., 2000] [RoboCup, 2001]. O campo é forrado a alcatifa verde e possui 10x5m de dimensão e duas balizas coloridas respectivamente de amarelo e azul. Até 2001 o campo estava rodeado de paredes brancas (com 50 cm de altura). Em 2002 as regras foram alteradas, retirando as paredes que rodeavam o campo e colocando postes coloridos nos quatro cantos, de forma a permitir uma localização mais simples aos robôs. Diversas linhas brancas são utilizadas para marcar o campo, incluindo uma linha que delimita o campo, linhas de golo, linhas que marcam as áreas, uma linha central e uma linha que delimita o círculo central. A iluminação do campo deve estar compreendida entre 500 e

1500 lux. De forma a serem regulamentares os robôs devem possuir um corpo negro e possuir marcas de identificação da equipa (*magenta* ou *cyan*). Diversas restrições elaboradas às dimensões, peso e forma dos robôs são colocadas de forma a garantir alguma uniformidade nas características dos mesmos.

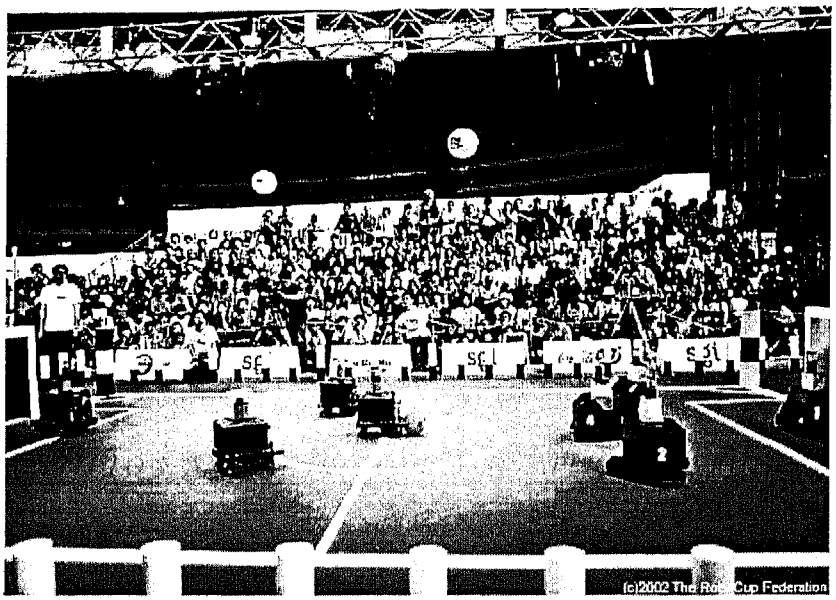

Figura 48: A liga de Robôs Médios (Middle-Size)

Os jogos de robôs médios são disputados entre duas equipas compostas por quatro robôs cada. Um árbitro humano e um assistente encarregam-se de fazer cumprir as regras do jogo. Cada jogo é composto por duas partes com 10 minutos cada.

8.2.4 A Liga de Robôs com Pernas – Cães da Sony

A liga de robôs com pernas utiliza equipas de 4 robôs AIBO ERS210 construídos pela Sony. Os robôs AIBO têm 20 graus de liberdade e sete tipos de sensores: imagem, áudio, temperatura, proximidade, aceleração, pressão e vibração. Nesta liga, uma vez que não é possível efectuar modificações no hardware dos robôs, a equipa com o *software* mais apropriado normalmente vence os jogos.

Cada jogo na liga de robôs com pernas é composto por duas metades de 10 minutos cada. Os robôs têm de ser totalmente autónomos durante os jogos. Controlo remoto, comunicação sem fios, ou visão global não foram permitidos até 2001 (embora comunicação visual e sonora pudessem ser utilizadas). A partir de 2002 foi permitida a realização de comunicação sem fios entre os robôs dentro do campo (embora este tipo de comunicação permanecesse proibida com o exterior do campo)

Os robôs jogam num campo de reduzidas dimensões (aproximadamente 4x3 metros) que contém marcadores bem definidos com determinadas cores. O campo tem também um rebordo branco que evita que a bola saia do mesmo. A bola é laranja e as balizas são azuis e amarelas. Os marcadores coloridos são utilizados para auxiliar os robôs a localizarem-se.

Nove cores são utilizadas no campo: cor de rosa e verde (para os marcadores), amarelo e azul claro (para os marcadores e balizas), vermelho e azul escuro (para os equipamentos dos robôs), cor de laranja (para a bola), verde claro (para o campo) e branco (para as linhas e outras marcações no campo). Os robôs necessitam de ter capacidade para detectar e discriminar estas cores de forma a serem capazes de reconhecer os objectos presentes no campo.

A investigação realizada no âmbito da liga de robôs com pernas está actualmente centrada em visão por computador, localização e locomoção robótica. Para estes três desafios de investigação, diferentes grupos desenvolveram diferentes abordagens. Com a recente disponibilização de comunicação sem fios e o aumento do número de elementos de cada equipa de 3 para 4, a capacidade de coordenação e comportamento de equipa dos robôs passou a ser crucial a partir de 2002.

A visão é um problema muito complexo nesta liga. Na liga de robôs pequenos a visão é obtida por um sistema centralizado através de uma (ou mais) câmaras colocadas em cima do campo. Através deste sistema é relativamente simples identificar as posições da bola e de todos os robôs que se encontram sobre o campo. Na liga de robôs médios a visão é bastante mais complexa pois os robôs têm de ser totalmente autónomos e como tal carregar os seus sistemas de visão totalmente a bordo. Este facto provoca que cada robô só tenha uma visão parcial do campo. Acresce que, habitualmente robôs de diferentes tamanhos e formas estão presentes no campo, parcialmente ocultos, o que torna o reconhecimento de objectos um problema complexo.

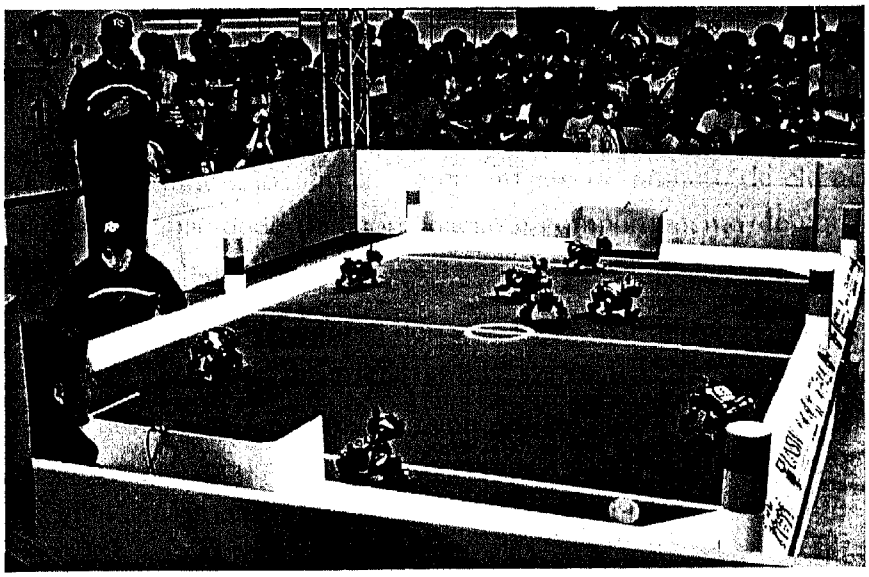

Figura 49: A liga de Robôs com Pernas da Sony

Na liga de robôs com pernas da Sony, a visão é ainda um problema mais complexo. Os robôs têm um ângulo de visão muito limitado e devido à locomoção com pernas, um acentuado efeito de oscilação está presente nas imagens. A cabeça dos robôs onde a câmara está montada possui três graus de liberdade o que permite a implementação de

algoritmos de percepção visual inteligente. Embora a Sony disponibilize, em conjunto com a plataforma robótica, algoritmos de detecção de cores e processamento de imagem, a maioria das equipas desenvolve os seus próprios algoritmos de processamento e análise de imagem. Por exemplo, Bruce et al. [Bruce et al., 2000] desenvolveu uma biblioteca de processamento e análise de imagem disponível utilizando a licença GPL designada CMVision [Bruce et al., 2001].

A localização é também um problema extremamente complexo na liga de robôs com pernas. De modo a permitir aos robôs efectuarem decisões baseadas em objectivos, a posição e direcção dos robôs no campo é extremamente importante [Wendler, 2001]. Os robôs têm três graus de liberdade para movimentar a câmara e, deste modo, é possível desenvolver algoritmos de localização baseados no movimento da cabeça de forma a localizar os marcadores posicionados em torno do campo que permitem aos robôs localizarem-se. A qualidade das imagens obtidas é crucial para definir algoritmos de localização robustos. É necessário desenvolver algoritmos de processamento e análise de imagem que permitam não só reconhecer os objectos presentes na imagem, mas também medir com precisão as dimensões, distâncias e direcções dos objectos presentes na imagem. A informação visual disponível neste ambiente, extremamente dinâmico, contém grandes erros, incluindo potenciais erros catastróficos (tais como a detecção de objectos fantasmas). Diferentes equipas desenvolveram diferentes algoritmos para a localização tais como métodos probabilísticos ou raciocínio baseado em casos[47].

Embora algoritmos para efectuar a movimentação básica dos robôs sejam disponibilizados pela Sony em conjunto com os robôs AIBO, estes não são suficientemente poderosos para desenvolver uma equipa competitiva nesta liga. As equipas têm de desenvolver melhores algoritmos de actuação a baixo-nível, incluindo algoritmos para andar rápido, rodar rápido, driblar, chutar ou defender a bola. Para além disso, a selecção de comportamentos de alto-nível é também muito importante. Por exemplo, os campeões de 2000 e 2001, UNSW da University de New South Wales em Sidney, Austrália, utilizaram robôs com diferentes papéis, cada qual com uma selecção de comportamentos distinta.

Outros tópicos de investigação relevantes nesta liga incluem a definição de arquitecturas de agentes de *software* apropriadas, comunicação multi-agente (*wireless* ou por áudio) entre robôs, e algoritmos para comportamento cooperativo em equipa. Embora todas as equipas utilizem um conjunto de módulos de *software* típico na sua arquitectura (tais como a visão, localização e locomoção), algumas equipas optaram por arquitecturas de agente distintas, utilizando, por exemplo, princípios BDI.

[47] Case-based reasoning

8.2.5 O *RoboCup Rescue:* Salvamento em Catástrofes

O salvamento em situações de desastre é um dos problemas sociais mais relevantes da actualidade, envolvendo um elevado número de agentes heterogéneos que actuam num ambiente hostil. O projecto *RoboCup Rescue* foi lançado em 1999 com a intenção de promover investigação e desenvolvimento neste domínio de grande impacto social a diversos níveis, envolvendo a coordenação do trabalho de equipas de agentes múltiplos, agentes físicos para busca e salvamento, infra-estruturas de informação, assistentes digitais pessoais e um simulador realista. Utilizando estes sistemas pretende-se efectuar a avaliação de estratégias e sistemas robóticos para busca e salvamento, sendo futuramente todos estes subsistemas integrados de uma forma abrangente.

Este projecto assenta na experiência de sucesso do projecto RoboCup na sua vertente de futebol de forma a permitir aplicar as metodologias desenvolvidas neste domínio em outras áreas, socialmente mais úteis. O domínio foi escolhido, em parte também, pelas suas características semelhantes ao futebol, como sejam o ambiente dinâmico, informação incompleta e com ruído, agentes heterogéneos e necessidade de coordenação de equipas de agentes na execução de tarefas complexas. A similaridade dos domínios leva os investigadores da área a crer que as metodologias definidas para as ligas de futebol do RoboCup, poderão em grande parte ser aplicadas também no *RoboCup Rescue* [Kitano et al., 1999].

Os objectivos principais do projecto podem ser detalhados da seguinte forma [Stone et al., 2001c]:

- Desenvolvimento e aplicação de tecnologias avançadas de robótica inteligente e inteligência artificial, para resposta a emergências e desastres no âmbito de sistemas sociais mais seguros;
- Introdução de novos problemas de importância social, como desafios para a robótica e IA, indicando direcções de investigação interessantes;
- Proposta de infra-estruturas de sistemas futuros baseados essencialmente em robótica e IA;
- Aceleração da investigação e desenvolvimento em busca e salvamento através da introdução da competição RoboCup.

Dois projectos e respectivas ligas com competições anuais estão actualmente activos: *Projecto de Simulação, Projecto de Robótica e Infra-estrutura.* Destes dois projectos resultam duas ligas com competições anuais: *Liga de Simulação de Rescue* e *Liga de Robôs de Rescue.* Os participantes dos dois projectos afirmam que a integração destas actividades conduzirá à criação das equipas de agentes robóticos heterogéneos de busca e salvamento do futuro [Rescue, 2001].

O *Projecto de Simulação RoboCup Rescue* iniciou-se com a construção de um ambiente genérico de simulação de desastres urbanos baseado em redes de computadores e na tecnologia dos agentes inteligentes. À semelhança do simulador *soccerserver*, o simulador de *Rescue*, inclui um módulo de simulação e um módulo de visualização. Diversos visualizadores foram construídos, incluindo visualizadores tridimensionais (figura 50).

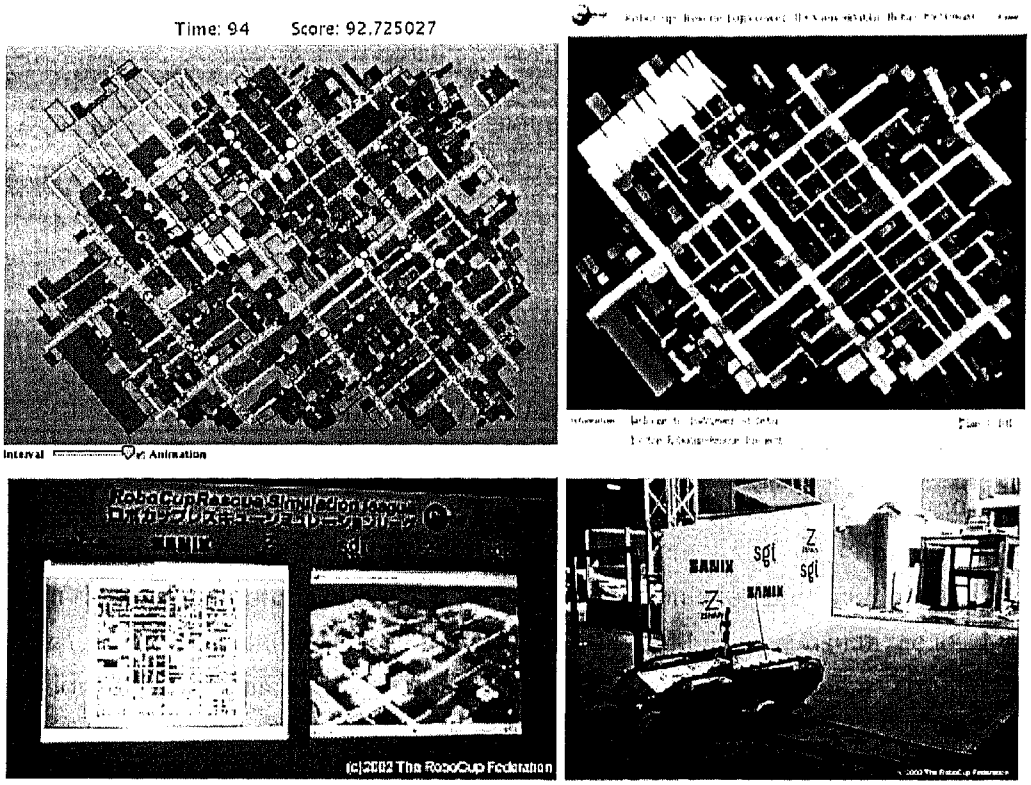

Figura 50: RoboCup Rescue (Visualizadores – em cima e Competição – em baixo)

O simulador modeliza um mundo virtual (cidade virtual) e a ocorrência de uma catástrofe nesse mundo. Para tal, simula o colapso de edifícios, bloqueio de estradas, espalhamento de incêndios, fluxo de trânsito e os seus efeitos mútuos. Diversos agentes heterogéneos inteligentes, tais como bombeiros, polícias, vítimas, voluntários, etc., conduzem acções de busca e salvamento neste mundo virtual. O objectivo das equipas de agentes consiste em minimizar os desastres da catástrofe e evitar a perda de vidas de humanas.

8.2.6 O RoboCup Júnior

O *RoboCup Júnior* é uma iniciativa onde as crianças do ensino básico e secundário podem experimentar na prática tópicos avançados de robótica. É essencialmente um projecto educacional com o objectivo de providenciar um ambiente que permita às crianças aprender ciência e tecnologia em geral através da robótica [Kitano, 1999] [Sklar et al., 2002]. Devido ao objectivo emblemático do RoboCup (que em 2050 uma equipa de robôs vença os campeões humanos de futebol), torna-se necessário que as gerações mais jovens se envolvam e como tal o *RoboCup Jr.* surge nesse contexto.

O *RoboCup Jr.* foi projectado de forma a que a excitação do jogo de futebol utilizando robôs reais em vez de criaturas virtuais possa estimular os mais novos, dando-lhes a conhecer uma parte da complexidade que é realmente programar robôs para desempenharem tarefas complexas. Ao contrário das outras ligas, que foram projectadas para permitir investigação ao mais alto nível por parte dos laboratórios de investigação, universidades e empresas, o *RoboCup Júnior* procura utilizar hardware robótico genérico e de fácil programação, tal como os MindStorms da Lego, para permitir a crianças terem uma experiência totalmente prática de realmente construir e programar um robô.

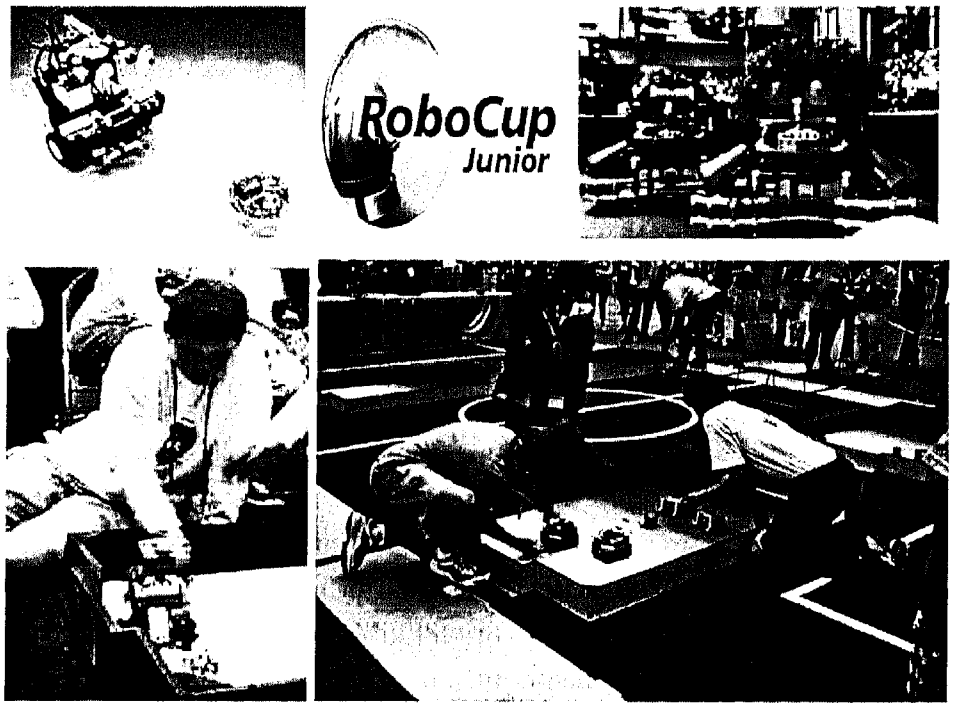

Figura 51: RoboCup Júnior (Modalidades de Futebol e Dança)

O *RoboCup Júnior* oferece diversos desafios com ênfase em aspectos cooperativos e competitivos. Para as crianças, esta iniciativa providencia uma oportunidade para trabalharem em grupo enquanto são introduzidos aos princípios básicos da robótica, electrónica, *hardware* e *software*. Os desafios actualmente oferecidos (figura 51) são três [RoboCup, 2001]:

- **Futebol.** Equipas de 2 robôs autónomos competem entre si num campo rectangular de reduzidas dimensões (figura 51).
- **Dança.** Um ou mais robôs executam habilidade de dança ao som da música. Este desafio procura estimular a criatividade na construção dos robôs e na definição dos seus movimentos.
- **Salvamento.** Os robôs executam corridas para salvar vítimas de um suposto desastre. O campo varia em complexidade, desde o simples seguimento de uma linha numa superfície plana, até ao seguimento de um caminho com obstáculos e inclinações.

Actualmente encontra-se em fase de construção um simulador oficial para o *RoboCup Júnior*, com o objectivo de permitir envolver ainda mais estudantes pré-universitários nesta iniciativa.

8.2.7 Outros Desafios Associados

8.2.7.1 Treinador *On-Line*

O treinador on-line é um agente especial na liga de simulação que possui informação global e sem erros sobre o jogo realizado. Como não está sujeito às restrições relacionadas com o raciocínio em tempo-real como estão os agentes que controlam os diversos jogadores (o treinador não controla qualquer jogador), este agente é utilizado para efectuar análises mais complexas, como as análises estratégicas ou tácticas do jogo.

Um dos desafios de investigação relacionados com o treinador *on-line* consiste na definição de uma linguagem de comunicação que permita a este agente comunicar de forma simples com a sua equipa de forma a alterar a táctica de jogo de acordo com a sua análise estratégica. Reis e Lau desenvolveram a primeira linguagem de alto-nível para o treinador, COACH UNILANG [Reis e Lau, 2002] no âmbito da equipa FC Portugal. Baseada nesta linguagem e nos contributos de diversos investigadores foi desenvolvida uma linguagem mais simples e menos expressiva, designada CLang, com o objectivo de permitir modificar o comportamento individual de agentes da liga de simulação e efectuar uma competição entre treinadores. Esta linguagem foi incorporada no *soccerserver* e designada "linguagem standard". No entanto, unicamente 4 equipas participaram no RoboCup 2001 na primeira competição de treinadores que a linguagem deveria permitir realizar, tendo as outras equipas concluído que a linguagem era demasiado "baixo-nível" e preocupado com o comportamento individual para permitir efectuar alterações no comportamento global da equipa. Em 2002 diversos conceitos da linguagem COACH UNILANG foram introduzidos na linguagem CLang tornando-a bastante mais expressiva e adequada à realização de uma competição de treinadores.

8.2.7.2 Visualizadores 3D

O desenvolvimento de visualizadores 3D para as diversas competições RoboCup tem sido um desafio de investigação estimulado ao longo dos últimos anos. O domínio preferencial da aplicação dos visualizadores tem sido a liga de simulação de futebol robótico. No entanto, nos últimos anos, após o aparecimento do *RoboCup Rescue*, visualizadores 3D para este domínio de simulação têm também sido desenvolvidos. Uma análise mais detalhada dos sistemas de visualização 3D disponíveis, dos projectos de investigação realizados nesta área, e do visualizador 3D desenvolvido no âmbito do projecto FC Portugal será apresentada na secção 9.9.

8.2.7.3 Comentador Inteligente

O desenvolvimento de um sistema de análise do jogo e comentário inteligente em linguagem natural, é um dos desafios associados ao RoboCup. O sistema tem de ser capaz de efectuar tarefas idênticas às efectuadas pelas equipas tais como detectar a posição da bola e dos jogadores de cada equipa no campo. No entanto, adicionalmente, tem de ser capaz de calcular informação de mais alto-nível a partir da evolução temporal das posições dos jogadores e da bola. Esta informação pode consistir em informação estatística do jogo, informação relativa ao comportamento colectivo de cada uma das equipas e informação relativa ao comportamento individual de cada jogador. A informação estatística relevante inclui[48] o resultado (número de golos marcados por cada equipa), o tempo de posse de bola de cada equipa em cada região do campo, o número de remates efectuados por cada equipa, informação relativa às perdas de bola e recuperações de bola, passes correctos e errados de cada equipa em cada região, etc. Esta análise pode implicar um sistema de visão centralizado, no caso do comentador ser aplicado às ligas robóticas (*small-size, middle-size* ou *legged*).

Para além da análise ao jogo, o comentador deve ser capaz de interpretar a informação e gerar comentários em linguagem natural baseados na análise efectuada e nos acontecimentos que se estão a passar no jogo. Isto implica a realização de investigação em tópicos tais como o processamento de linguagem natural e síntese de voz.

Embora este desafio seja amplamente encorajado pela Federação RoboCup (essencialmente devido à atracção que propicia para os espectadores), poucos sistemas de análise e comentário foram desenvolvidos até ao momento. De entre os sistemas desenvolvidos destacam-se o MIKE e o Isaac.

O sistema Isaac foi desenvolvido por Tambe et al. [Tambe et al., 1999] com o objectivo de efectuar análises à posteriori de jogos efectuados na liga de simulação. O sistema utiliza o algoritmo C4.5 para gerar árvores de decisão que demonstram como as equipas se comportaram num dado jogo. No entanto as análises que realiza são em geral muito simples e longe das análises realizadas por um analista de futebol real.

O sistema MIKE foi desenvolvido originalmente em 1998 pelo laboratório ETL no Japão para a liga de simulação [Frank, 2001]. É capaz de gerar comentários em texto ou utilizando síntese de voz em Inglês, Francês ou Japonês. Em conjunto com o sistema foi desenvolvido um sistema de cálculo de estatísticas de jogo, o *Proxy Server* capaz de se conectar ao simulador durante o jogo e gerar diversas estatísticas simples de jogos realizados na liga de simulação.

---

[48] Uma análise mais detalhada da informação estatística relevante na liga de simulação será efectuada no capítulo 9, durante a análise da linguagem de treinador COACH UNILANG.

O sistema MIKE foi entretanto adaptado com sucesso para a liga de *small-size*. Esta adaptação foi efectuada utilizando um sistema de visão global que envia a posição dos robôs e da bola para o sistema MIKE e na sua associação ao robô humanóide SIG que se encarregava posteriormente de realizar algumas animações e efectuar o comentário [Frank, 2001].

8.3 O Futebol Robótico Simulado

A liga de simulação do RoboCup é baseada no simulador designado por *SoccerServer* [Noda et al., 1998] que simula um jogo virtual de futebol disputado entre duas equipas compostas por 11 agentes autónomos cada. O simulador foi construído como um ambiente de simulação multi-agente, incerto e com funcionamento em tempo-real, capaz de permitir a competição entre equipas de jogadores virtuais, cada qual controlado separadamente por um agente autónomo.

Esta secção pretende oferecer uma panorâmica geral do funcionamento do simulador que é utilizado como base para as competições de futebol robótico simulado e para diversos outros trabalhos de investigação e desafios associados – o *soccerserver*. O conteúdo da secção é baseado no manual da versão actual do simulador – *soccerserver* [SoccerServer, 2001] e na experiência de utilização do sistema do autor. São apresentadas essencialmente as características mais relevantes do simulador para a compreensão do trabalho realizado no âmbito desta tese.

8.3.1 Aplicações Constituintes do Simulador

O sistema de simulação de futebol é composto por 3 módulos principais: o servidor (*soccerserver*); o visualizador (*soccer monitor*) e o vídeo (*logplayer*).

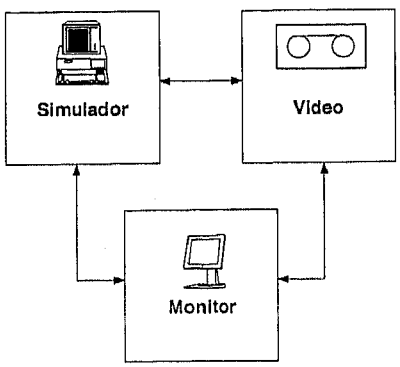

Figura 52: Aplicações Constituintes do Sistema de Simulação

O simulador é o responsável por realizar a simulação de jogos utilizando para tal uma arquitectura cliente-servidor. Providencia um campo de futebol virtual e simula o movimento de todos os objectos no mesmo, controlando o jogo segundo um conjunto de regras pré-especificado. Os clientes (jogadores) conectam-se ao simulador através de

sockets UDP. O simulador recebe os comandos dos clientes, executa-os simulando o movimento de todos os objectos no campo e envia aos clientes informação sensorial.

O monitor é a ferramenta que permite visualizar os jogos virtuais realizados, com o auxílio do simulador. Comunica com o simulador também através de *sockets UDP*, recebendo deste as posições de todos os objectos no campo virtual. Efectua depois o *display* gráfico desta informação de forma a permitir aos humanos a visualização do jogo.

O vídeo (*Logplayer*) é uma aplicação que permite a visualização de jogos pré-gravados (ficheiros log) à semelhança do que acontece com um gravador de vídeo. A sua utilização é associada à utilização de um monitor de forma a poder visualizar os jogos.

8.3.1.1 O Simulador – *SoccerServer*

O simulador (servidor) é o sistema que possibilita a realização de jogos simulados de futebol entre duas equipas compostas por 11 agentes autónomos. Uma vez que funciona numa arquitectura cliente-servidor, permitindo aos jogadores comunicarem através de *sockets UDP*, não impõe qualquer limitação às linguagens ou sistemas operativos utilizados para o desenvolvimento das equipas. A única restrição é que as equipas suportem comunicação via *UDP/IP*.

Cada equipa pode ter até 11 jogadores (clientes) e eventualmente um treinador (cliente com capacidades e privilégios especiais). Cada cliente é um processo separado que se conecta ao servidor através de uma porta especificada. Para participar num jogo, o cliente deve enviar uma mensagem para a porta 6000 do servidor que como resposta deverá aceitar a sua conexão atribuindo-lhe uma porta específica que irá ser utilizada a partir desse momento para efectuar a transferência de informação entre o simulador e o cliente.

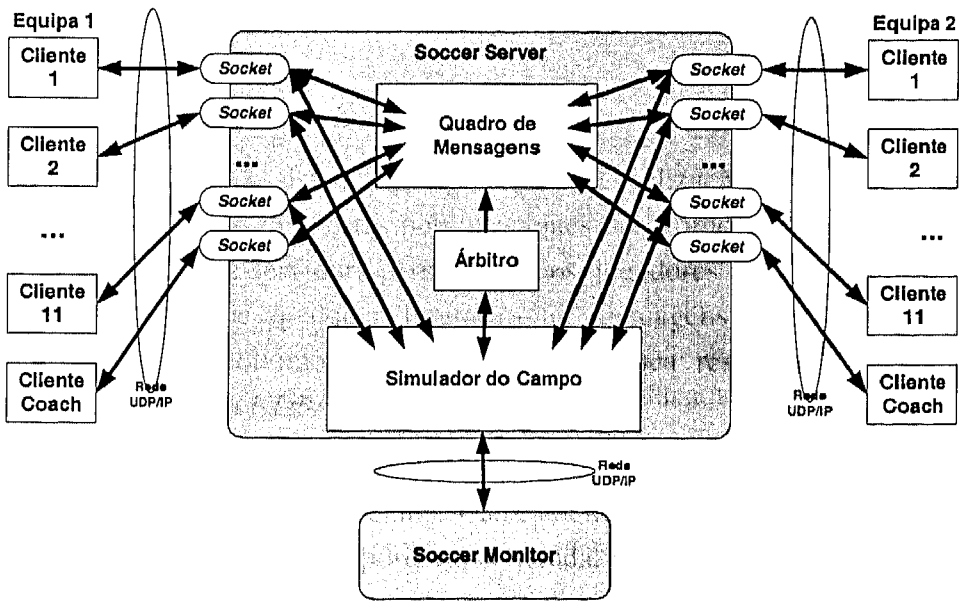

Figura 53: Arquitectura do Sistema de Simulação SoccerServer

Os jogadores enviam pedidos ao simulador sobre as acções a realizar pelo jogador que controlam (rodar, acelerar, chutar a bola, etc.). O servidor recebe as suas mensagens, trata de executar os seus pedidos (deste que estes sejam válidos[49]) e actualiza o ambiente em consonância. Em adição, o servidor envia aos jogadores informação sensorial, i.e., informação visual relativa aos jogadores e objectos no campo de visão do jogador, informação auditiva composta pelas mensagens que o jogador ouve, etc.

O simulador é um sistema de tempo real que no entanto trabalha com intervalos de tempo discretos, ou ciclos. Em cada ciclo (que tem uma duração por defeito de 100ms), o simulador aguarda por comandos dos jogadores e envia (se for o caso), informação sensorial aos jogadores. No final do ciclo, o simulador actualiza o ambiente virtual tendo em conta as acções que cada jogador deseja executar e a dinâmica do sistema em geral.

8.3.1.2 O Monitor – *Soccer Monitor*

O *soccer monitor* é uma ferramenta de visualização que permite visualizar os jogos realizados utilizando o simulador *soccerserver*. Recebe do simulador, através de *sockets UDP*, a informação respeitante aos diversos objectos e suas posições no campo e aos eventos do jogo. A visualização desta informação é realizada através de uma interface gráfica. Permite ainda comunicar em sentido inverso com o simulador (servidor), permitindo desta forma ao utilizador, enviar comandos úteis como sejam o início ou interrupção de um jogo ou a marcação de um livre numa dada região do campo a favor de uma das equipas.

Vários monitores podem ser conectados ao mesmo servidor no decurso de um jogo. Esta possibilidade permite visualizar em diversos terminais o mesmo jogo ao mesmo tempo utilizando para tal, diferentes ferramentas de visualização com potencialidades distintas (por exemplo um monitor 3D e um sistema de análise de jogo). No entanto, para realizar um jogo não é necessário conectar qualquer monitor ao simulador.

Monitores com capacidade de conexão ao simulador e de visualização gráfica dos jogos foram construídos por diferentes grupos de investigação. De entre estes, destacam-se os monitores tridimensionais que permitem uma visualização 3D dos jogos realizados e visualizadores direccionados para a análise de jogos e cálculo de estatísticas de jogo.

O monitor tradicional do *soccerserver* permite configurar as cores e dimensões com que são visualizados todos os objectos (jogadores e bola). Permite ainda configurar os tipos de mensagens visualizados e os respectivos textos e as dimensões e fontes destes. Embora desde a criação da liga de simulação, o monitor utilizado em jogos oficiais tenha sido o

---

[49] Exemplos de pedidos inválidos são: chutar a bola quando ela não se encontra suficientemente próxima do agente; acelerar o agente quando este não possui energia para o fazer; agarrar a bola quando o agente não é o guarda-redes da equipa.

Soccer Monitor tradicional, o monitor oficial do futebol simulado é actualmente, desde Outubro de 2001, o *FrameView* [Merke, 2002], construído pela universidade de Karlsruhe na Alemanha. Para além das funções do monitor tradicional, permite ainda analisar diversas características do jogo, úteis para a sua análise. Entre estas incluem-se a visualização de: ampliações de regiões do campo, da energia dos jogadores, da visão dos jogadores e das características dos jogadores heterogéneos[50] presentes no campo.

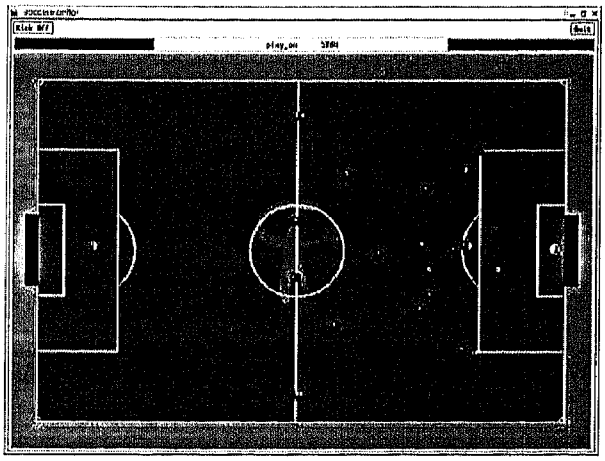

Figura 54: Monitor Tradicional para Linux do Soccer Server.

A arquitectura modular do simulador permite a separação entre o módulo de simulação e o módulo de visualização. Isto permite não só a utilização de diferentes monitores associados ao mesmo simulador, mas ainda a construção de monitores em sistemas operativos distintos. Um exemplo significativo é o visualizador para Windows construído por Klaus Dorer na universidade de Freiburg [Dorer, 2000] e continuado pela universidade de Tsinghua, China [Cai et al., 2002].

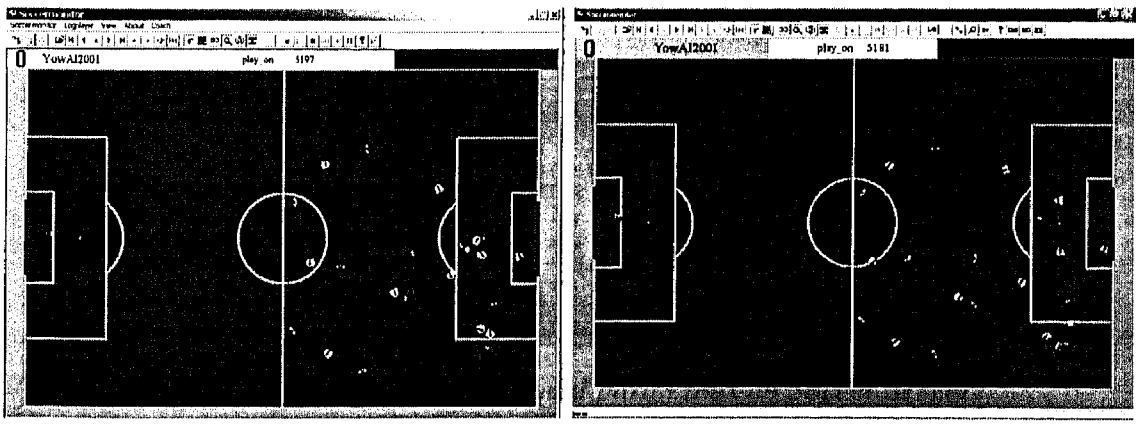

Figura 55: Monitores para Windows do Soccer Server (Freiburg e Tsinghua).

[50] Os jogadores heterogéneos foram introduzidos na liga de simulação em 2001. Consistem em jogadores com capacidades físicas distintas dos jogadores tradicionais. Por exemplo, um jogador pode ter uma maior velocidade máxima mas cansar-se mais rapidamente ou então ter uma maior potência de chuto mas uma menor precisão.

8.3.1.3 O Vídeo - *LogPlayer*

O *LogPlayer* é uma aplicação que permite a visualização de jogos pré-gravados à semelhança do que acontece com um gravador de vídeo. O servidor está equipado com opções que permitem, enquanto um jogo é realizado, gravar a informação mais relevante do mesmo para o disco do computador. O ficheiro gerado (ficheiro log) pode depois ser utilizado para visualizar o jogo com o auxílio do *Logplayer*. A utilização do *Logplayer* é usualmente combinada com a utilização de um Monitor de forma a visualizar o jogo gravado. A utilização principal do *LogPlayer* situa-se ao nível da análise da estratégia das várias equipas e dos seus pontos fracos e fortes. À semelhança do que se passa com um gravador de vídeo, o *LogPlayer* está equipado com botões de avanço e recuo rápido e paragem, dispondo também da possibilidade de avançar directamente para um dado ciclo do jogo.

8.3.2 O Mundo Simulado e as Regras do Jogo

Nesta secção descreve-se a forma de execução de um jogo de futebol robótico simulado, incluindo as regras do jogo e a dinâmica do mundo simulado. Para uma melhor compreensão do conteúdo desta e das seguintes secções, é necessário possuir conhecimentos básicos sobre futebol real e sobre as regras deste jogo.

8.3.2.1 O Campo e os Objectos

Os jogos de futebol robótico simulado disputam-se num campo virtual com 105*68m que contém diversas linhas e objectos para auxiliar os jogadores a se localizarem. A figura 56 contém uma representação do campo virtual e das linhas e marcos de localização que o mesmo contém.

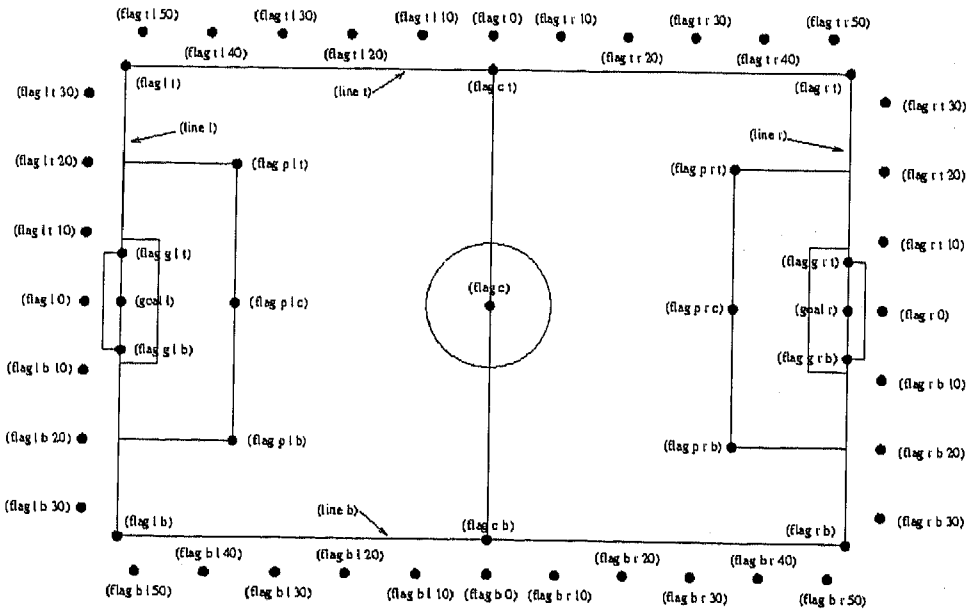

Figura 56: Localização e nomes dos marcos de orientação no Soccer Server.

As bandeiras *(goal r)* e *(goal l)* correspondem a bandeiras virtuais localizadas no centro das balizas direita e esquerda. *(flag g l b)*, *(flag g l t)*, *(flag g r b)* e *(flag g r t)* correspondem aos postes virtuais das duas balizas. *(flag c)* corresponde a uma bandeira virtual no centro do campo. Existem ainda bandeiras colocadas nos limites do campo e das áreas. As restantes bandeiras estão colocadas fora do campo e a 5 metros do limite deste, espaçadas entre elas de 10 metros. Por exemplo, *(flag b l 30)* está 5 metros abaixo da linha lateral inferior do campo, 30 metros à esquerda do centro do campo. Existem ainda 4 linhas visíveis que limitam o campo. Estas correspondem a *(line t)*, *(line b)*, *(line r)* e *(line l)* que limitam o campo respectivamente em cima, em baixo, à direita e à esquerda. Cada bandeira é identificada por uma etiqueta que é usada para a sua identificação durante a comunicação simulador-jogador.

Cada jogo de futebol robótico simulado tem uma duração de 10 minutos divididos em duas partes de 5 minutos, correspondentes a 3000 ciclos de simulação cada[51]. Em caso de empate no final do tempo regulamentar, seguem-se prolongamentos sucessivos de 5 minutos (3000 ciclos) seguindo a regra do golo dourado[52]. Em competições oficiais, dependendo do tipo de jogo disputado, o jogo pode ser terminado no final do tempo regulamentar, com um empate, mesmo sem recurso a prolongamento.

8.3.2.2 Os Árbitros Artificial e Humano

Durante cada partida existe um determinado conjunto de regras que é garantido pelo árbitro automático contido no servidor que comunica com os jogadores através de mensagens, e por um árbitro humano designado para supervisionar o jogo.

O árbitro artificial consegue arbitrar o jogo totalmente sem necessidade de auxílio do árbitro humano e em geral, em jogos entre equipas de boa qualidade e com *fair-play*, o árbitro humano é dispensável. No entanto, faltas tais como obstruções ou anti-jogo são muito difíceis de ser julgadas pelo árbitro artificial pois dizem respeito a intenções dos jogadores. Para resolver este tipo de situações, o *soccer monitor* providencia uma interface que permite a um árbitro humano efectuar duas operações parametrizáveis de supervisão do jogo: marcar uma falta a favor de uma das equipas num dado ponto do campo e "lançar a bola ao solo" num dado ponto do campo. No anexo 4 são apresentadas as principais características do árbitro artificial, as regras que este é responsável por garantir, as mensagens que este agente pode enviar aos jogadores e a sua respectiva sintaxe.

---

[51] Isto considerando os parâmetros por defeito actuais do simulador, nomeadamente ciclos de simulação de 100 mili-segundos.

[52] Também designada vulgarmente por morte-súbita, isto é, a equipa que marcar o primeiro golo no prolongamento vence de imediato o jogo.

Todos os anos, as capacidades do árbitro artificial são estendidas de forma a torná-lo cada vez mais autónomo e evitar a necessidade de existir intervenção humana durante o jogo. Neste sentido, em 2001, o árbitro passou a realizar a marcação de bolas ao solo no caso de uma equipa não recolocar rapidamente a bola em jogo ou de o guarda-redes de uma equipa efectuar vários *moves* em sequência antes de repor a bola. Em 2002, o árbitro foi estendido de forma a marcar livres no caso de atrasos de bola para o guarda-redes ou marcação de livres em que um jogador passa a bola para si mesmo e prossegue com ela controlada.

Ao longo dos próximos anos é previsível que o árbitro artificial seja ainda mais desenvolvido de forma à intervenção humana na arbitragem dos jogos ser totalmente desnecessária. É previsível também a separação deste módulo do servidor num agente autónomo com capacidades de análise e controlo do jogo. Isto permitirá aumentar a modularidade do sistema e permitirá que diferentes grupos de investigação desenvolvam os seus próprios árbitros. Desta forma, os diferentes jogos oficiais poderão ser arbitrados por árbitros artificiais construídos por diferentes universidades e países, à semelhança do que se passa no futebol real.

8.3.3 Protocolos de Comunicação com o Servidor

De forma a poderem participar no jogo simulado, os clientes têm de possuir capacidades que lhes permitam seguir um determinado conjunto de protocolos de comunicação com o servidor. De entre estes protocolos destacam-se os protocolos de conexão, acção e percepção.

8.3.3.1 Protocolo de Conexão dos Clientes

O protocolo de conexão (tabela 10) permite a um cliente conectar-se, desconectar-se ou reconectar-se ao simulador.

Conexão (Cliente para o Servidor)	*Conexão (Resposta do Servidor)*
(**init** *<NomeEquipa> [(version <NumVer>)] [(goalie)]*) *<NomeEquipa> ::= (-|_|a-z|A-Z|0-9)* *<NumVer> ::= versão do protocolo de comunicação a utilizar (p.e. 7.0)*	(**init** *<Lado> <Unum> <ModoJogo>*) *<Lado> ::= l|r* *<Unum> ::= 1-11* *<ModoJogo> ::= um dos modos de jogo válido* (**error** *no_more_team_or_player_or_goalie*)
Reconexão (Cliente para o Servidor)	*Reconexão (Resposta do Servidor)*
(**reconnect** *<NomeEquipa> <Unum>*) *<NomeEquipa> ::= (-|_|a-z|A-Z|0-9)*	(**init** *<Lado> <Unum> <ModoJogo>*) *<Lado> ::= l|r* *<Unum> ::= 1-11* *<ModoJogo> ::= um dos modos de jogo válido* (**error** *no_more_team_or_player*) (**error** *reconnect*)
Disconexão (Cliente para o Servidor)	*Disconexão (Resposta do Servidor)*
(**bye**)	

Tabela 10: Protocolo de Conexão dos Clientes ao Simulador

Como o *soccerserver* é um sistema em constante evolução, permite a conexão de clientes que utilizam versões do protocolo distintas, permitindo desta forma que clientes desenvolvidos para versões anteriores do *soccerserver* se possam conectar ao sistema. No caso do cliente se conectar com um protocolo superior ou igual ao 7.0, então receberá adicionalmente a informação relativa aos parâmetros do servidor[53] e aos parâmetros dos jogadores heterogéneos disponíveis[54].

8.3.3.2 Protocolo de Percepção dos Clientes

A percepção dos clientes está dividida em três partes distintas: percepção auditiva (***hear***), visual (***see***) e sensorial (***sense_body***). A tabela 11 descreve em detalhe o protocolo utilizado pelo servidor para enviar estes três tipos de percepção aos clientes.

Percepção (Servidor para o Cliente)
(**hear** <*Tempo*> <*Emissor*> <*Mensagem*>) <*Tempo*> ::= *ciclo de simulação do simulador* <*Emissor*> ::= *online_coach_left* | *online_coach_right* | *referee* | *self* | <*Direcção*> <*Direcção*> ::= *-180..180* <*Mensagem*> ::= *[string]*
(**see** <*Tempo*> <*InfoObj*>) <*Tempo*> ::= *ciclo de simulação do simulador* <*InfoObj*> ::= (<*NomeObj*> <*Distância*> <*Direcção*> <*DistVar*> <*DirVar*> <*DirCorpo*> <*DirCabeça*>) | (<*NomeObj*> <*Distância*> <*Direcção*> <*DistVar*> <*DirVar*>) | (<*NomeObj*> <*Distância*> <*Direcção*>) | (<*NomeObj*> <*Direcção*>) <*NomeObj*> ::= (p <*NomeEquipa*> [<*Unum*> [*goalie*]]]) | (*b*) | *g* [*l*|*r*] | (*l* [*l*|*r*|*t*|*b*]) | (*f c*) | (*f* [*l*|*c*|*r*] [*t*|*b*]) | (*f p* [*l*|*r*] [*t*|*c*|*b*]) | (*f g* [*l*|*r*] [*t*|*b*])| (*f* [*l*|*r*|*t*|*b*] *0*) | (*f* [*t*|*b*] [*l*|*r*] [*10*|*20*|*30*|*40*|*50*]) | (*f* [*l*|*r*] [*t*|*b*] [*10*|*20*|*30*]) <*Distância*> ::= *Real positivo* <*Direcção*> ::=*[-180.0 .. 180.0]* <*DistVar*> ::= *Real* <*DirVar*> ::= *Real* <*DirCorpo*> ::= *[-180.0, 180.0]* <*DirCabeça*>::= *[-180.0, 180.0]* <*NomeEquipa*> ::= *[string]* <*Unum*> ::= *[1..11]*
(**sense_body** <*Tempo*> (view_mode *[high|low] [narrow|normal|wide]*) (stamina <*Energia*> <*Esforço*>) (speed <*ValorVel*> <*DirVel*>) (head_angle <*DirCabeça*>) (kick <*ContagemKicks*>) (dash <*ContagemDashes*>) (turn <*ContagemTurns*>) (say <*ContagemSays*>) (turn_neck <*ContagemTurnNecks*>) (catch <*ContagemCatchs*>) (move <*ContagemMoves*>) (change_view <*ContagemChangeViews*>)) <*Tempo*> ::= *ciclo de simulação do simulador*

[53] Antes da versão 7.0 os clientes tinham de possuir um ficheiro de configuração com os parâmetros utilizados pelo servidor.

[54] Antes da versão 7.0 os jogadores eram homogéneos.

<Energia> ::=[0..4000]
<Esforço> ::=[0..1.0]
<ValorVel> ::= real positivo
<DirVel> ::=[-180.0 .. 180.0]
<DirCabeça> ::=[-180.0 .. 180.0]
<Contagem> ::= inteiro positivo (para todas as contagens)*

Tabela 11: Protocolo de Percepção dos Clientes

O protocolo de percepção dos clientes é relativamente simples, pois não é previsto que os clientes enviem qualquer tipo de resposta ao servidor após a recepção de informação sensorial.

8.3.3.3 Protocolo de Acção dos Clientes

O protocolo de acção dos clientes especifica a sintaxe das mensagens de comando que os clientes estão autorizados a enviar para o servidor e as possíveis respostas do servidor a estas mensagens. Este protocolo é em geral muito simples, pois a maioria das mensagens enviadas pelos clientes não implicam a ocorrência de resposta por parte do servidor. O protocolo encontra-se representado na tabela 12.

Acção (Cliente para o Servidor)
(**dash** *<Potência>*) *<Potência> ::= [-100, 100]*
(**turn** *<Momento>*) *<Momento> ::= [-180, 180]*
(**move** *<PosX> <PosY>*) *<PosX> ::= [-52.5, 52.5]* *<PosY> ::= [-34.0, 34.0]*
(**kick** *<Potência> <Direcção>*) *<Potência> ::= [-100, 100]* *<Direcção> ::= [-180, 180]*
(**tackle** *<Potência>*) *<Potência> ::= [0, 100]*
(**catch** *<Direcção>*) *<Direcção> ::= [-180, 180]*
(**turn_neck** *<Direcção>*) *<Direcção> ::= [-180, 180]*
(**change_view** *<Abertura> <Qualidade>*) *<Abertura> ::= narrow | normal | wide* *<Qualidade> ::= high | low*
(**attentionto** *<Equipa> <Unum>*) | *(attentionto off)* *<Equipa> ::= opp | our | l | r | left | right | <Nome_Equipa>*
(**say** *<Mensagem>*) *<Mensagem> ::= "Texto"*
(**pointto** *<Distância> <Direcção>*) | *(pointo off)* *<Distância> ::= Inteiro* *<Direcção> ::= [-180, 180]*

Acção (Resposta do Servidor)
(**error** *unknown_command*) (**error** *ilegal_command_form*) (**score** <*Tempo*> <*GolosEquipa*> <*GolosOponente*>) *para comandos* (**score**) (**sense_body** <*Tempo*> (view_mode *(high|low)* *(narrow|normal|wide)*) (stamina <*Energia*> <*Esforço*>) (speed <*ValorVel*> <*DirVel*>) (head_angle <*DirCabeça*>) (kick <*ContagemKicks*>) (dash <*ContagemDashes*>) (turn <*ContagemTurns*>) (say <*ContagemSays*>) (turn_neck <*ContagemTurnNecks*>) (catch <*ContagemCatchs*>) (move <*ContagemMoves*>) (change_view <*ContagemChangeViews*>)) *para comandos* (**sense_body**)

Tabela 12: Protocolo de Acção dos Clientes

Os clientes enviam comandos que o simulador executa sem enviar qualquer resposta aos clientes. No caso do comando ser desconhecido ou os seus parâmetros serem ilegais, o simulador responde com uma mensagem de erro. Só existem dois casos em que o simulador responde ao comando enviado pelos clientes que corresponde ao comando (***score***) a que o simulador responde com o tempo e resultado do jogo e o comando (***sense_body***) a que o simulador responde com informação sensorial física para o agente.

8.3.4 Percepção dos Agentes

No futebol robótico simulado, como foi referido anteriormente, os agentes têm três tipos de sensores. O sensor auditivo detecta mensagens enviadas pelo árbitro, treinador, colegas de equipa e adversários. O sensor visual detecta informação visual disponível que inclui distâncias e direcções dos objectos e jogadores que se encontram no campo de visão do agente. O sensor físico detecta o estado do agente, incluindo a sua energia, velocidade e ângulo do pescoço relativamente ao corpo. Utilizando correctamente estes três sensores, o agente deve ser capaz de construir uma imagem clara das ocorrências no campo de jogo.

8.3.4.1 Informação Visual

A informação visual desempenha um papel primordial no futebol robótico simulado. O sensor visual dos agentes permite-lhes obter informação sobre a direcção e distância dos objectos que se encontram no seu campo visual em cada instante. O sensor visual desempenha também o papel de sensor de proximidade, dando aos agentes informação sobre os objectos que se encontram muito próximos mas atrás do agente. A informação visual é enviada automaticamente pelo servidor periodicamente cada *send_step* ou *send_step/2* dependendo do modo de visualização seleccionado pelo jogador.

É importante perceber que a informação enviada a cada agente é relativa à perspectiva desse agente, e como resultado disto um dado agente não se pode aperceber directamente da sua posição global ou da posição global dos outros jogadores ou da bola. Para tal, a informação relativa recebida deve ser convertida em informação absoluta. Como auxílio à realização desta conversão, conforme demonstrado na figura anterior, diversos marcos e

linhas encontram-se colocados no campo. Combinando as posições globais conhecidas destes marcos com as suas posições relativas que se encontram incluídas nas mensagens de visualização do agente, é possível ao agente aferir a sua posição global no campo e consequentemente determinar as posições relativas dos outros agentes e da bola.

Um jogador pode controlar a frequência, alcance e qualidade da informação visual que lhe é enviada pelo simulador. A frequência por defeito de envio da informação visual é controlada pelo parâmetro *send_step* que corresponde a 150 ms. No entanto, o agente pode trocar frequência por qualidade da informação visual e abertura do seu cone de visão. Para o fazer, pode ajustar a sua qualidade de visualização e a sua abertura de visualização, *ViewQuality* e *ViewWidth* respectivamente. A tabela 13 demonstra a frequência visual dos agentes tendo em conta o modo visual seleccionado. Esta frequência visual é calculada da seguinte forma:

frequência_visual = *send_step* * ***factor_qualidade*** * ***factor_abertura***

em que **factor_qualidade** é 0.5 se *ViewWidth* for ***narrow***, 1 se for ***normal*** e 2 se for ***wide*** e **factor_abertura** é 1 se *ViewQuality* for ***high*** e 0.5 se for ***low***. O cone de visão do agente é determinado pelo parâmetro do servidor *visible_angle* que representa a abertura em graus do cone de visão normal do agente (que corresponde a 90° actualmente). O ângulo de visualização é calculado da seguinte forma:

ângulo_visualização = *visible_angle* * ***factor_abertura***

Um agente pode ainda ver (sentir) objectos que estão até *visible_distance* do seu corpo embora só perceba o seu tipo (bola, jogador, baliza, bandeira, etc.) sem receber informação acerca da sua identificação (qual o jogador, qual a bandeira, etc.).

Width / *Quality*	*Narrow*	*Normal*	*Wide*
High	*send_step/2* *75 ms*	*send_step* *150 ms*	*send_step*2* *300 ms*
Low	*send_step/4* *37.5 ms*	*send_step/2* *75 ms*	*send_step* *150 ms*

Tabela 13: Frequência visual dos agentes de acordo com o seu modo visual

A informação visual é enviada pelo servidor aos jogadores no formato seguinte:

(**see** *<Tempo> <InfoObj>*)

onde:

<Tempo> ::= Inteiro (e corresponde ao ciclo do simulador)

<InfoObj> ::= (<NomeObj> <Distância> <Direcção> [<DistVar> <DirVar>
[<DirCorpo> <DirCabeça>]]) | (<NomeObj> <Direcção>)

dependendo do tipo de visão seleccionado e onde:

<NomeObj> ::= (p <NomeEquipa> [<UNum> [goalie]]) |

(b) | g [l|r] | (l [l|r|t|b]) |

(f c) | (f [l|c|r] [t|b]) | (f p [l|r] [t|c|b]) | (f g [l|r] [t|b])|

(f [l|r|t|b] 0) | (f [t|b] [l|r] [10|20|30|40|50]) | (f [l|r] [t|b] [10|20|30])

<Distância> ::= Real positivo

<Direcção> ::=[-180.0 .. 180.0]

<DistVar> ::= Real

<DirVar> ::= Real

<DirCorpo> ::= [-180.0 .. 180.0]

<DirCabeça>::= [-180.0 .. 180.0]

<NomeEquipa> ::= [string]

<Unum> ::= [1 .. 11]

Esta informação visual é fornecida para todos os objectos visíveis, isto é, aqueles que estão no cone de visão do jogador. A quantidade de informação fornecida para cada objecto, depende do tipo e distância do objecto e da qualidade de visão seleccionada pelo jogador. Quando a qualidade de visão for ***low***, as únicas informações fornecidas são o nome e direcção do objecto. Se a qualidade for ***high***, a distância ao objecto é também fornecida e dependendo do tipo de objecto outra informação pode ainda ser fornecida como seja a direcção do corpo ou do pescoço no caso de um jogador.

A informação relativa a *<Distância>*, *<Direcção>*, *<DistVar>* e *<DirVar>* é calculada da seguinte forma [Soccerserver, 2001]:

$p_{rx} = p_{xt} - p_{xo}$

$p_{ry} = p_{yt} - p_{yo}$

$v_{rx} = v_{xt} - v_{xo}$

$v_{ry} = v_{yt} - v_{yo}$

$<Distância> = \sqrt{p_{rx}^2 + p_{ry}^2}$

$<Direcção> = arctan\ (p_{ry}/p_{rx}) - Dir_o$

$e_{rx} = p_{rx} / <Distância>$

$e_{ry} = p_{ry} / <Distância>$

$<DistVar> = (v_{rx} * e_{rx}) + (v_{ry} * e_{ry})$

$<DirVar> = ((-(v_{rx} * e_{rx}) + (v_{ry} * e_{ry}))/<Distância>)*(180/\pi)$

Onde (p_{xt}, p_{yt}) é a posição absoluta do objecto visualizado, (p_{xo}, p_{yo}) é a posição absoluta do agente que está a visualizar o objecto, (v_{xt}, v_{yt}) é a velocidade absoluta do objecto visualizado, (v_{xo}, v_{yo}) é a velocidade absoluta do agente que está a visualizar o objecto e Dir_o é a direcção absoluta para a qual o agente se encontra direccionado. Desta forma, (p_{rx}, p_{ry}) e (v_{rx}, v_{ry}) são respectivamente a posição relativa e a velocidade relativa do objecto visualizado e (e_{rx}, e_{ry}) é um vector unitário que é paralelo ao vector (p_{rx}, p_{ry}).

Os parâmetros *<DirCorpo> e <DirCabeça>* só são incluídos no caso do objecto observado ser um jogador e correspondem à direcção relativa do corpo e cabeça do jogador, observado relativamente ao observador.

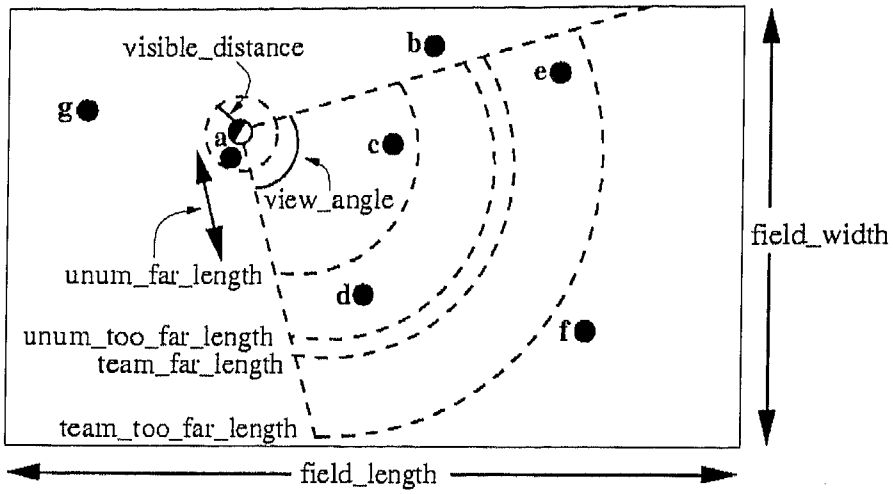

Figura 57: Percepção Visual dos agentes [Stone, 1998].

A figura 57 demonstra a percepção visual dos agentes. O Agente cuja percepção visual está a ser ilustrada é representado através de dois semicírculos (em que o semicírculo claro corresponde à frente do agente). Os círculos negros correspondem a outros jogadores no campo. Só os objectos dentro do ângulo de visão (*view_angle*) ou a uma distância inferior a *visible_distance* são visíveis. Os parâmetros *unum_far_length, unum_too_far_length, team_far_length* e *team_too_far_length* afectam a precisão e tipo de informação visual enviada ao agente.

A informação visual enviada, pelo servidor pode ser sumariada da seguinte forma:

- **Objectos Estáticos (bandeiras, linhas e balizas):** nome do objecto, distância relativa e direcção. No caso das linhas, a distância corresponde à distância ao ponto da linha onde o bissector do cone de visão do jogador cruza essa linha e a direcção é o ângulo entre a linha e esse bissector.
- **Jogadores:** A informação depende da distância do observador ao jogador da forma seguinte:

  Se *<Distância>* $\leq$ *unum_far_length*, então quer o número quer a equipa do jogador são incluídas. Os valores de *<Distância>, <Direcção>, <DistVar>, <DirVar>, <DirCorpo>* e *<DirCabeça>* são todos incluídos.

  Se *unum_far_length* < *<Distância>* $\leq$ *unum_too_far_length* = *team_far_length*, então a equipa do jogador e os valores de *<Distância>, <Direcção>*, serão visíveis. No entanto, a probabilidade de que o número do jogador seja visualizado decresce linearmente de 1 par 0 entre *unum_far_length* e *unum_too_far_length.* O mesmo acontece com a probabilidade de serem incluídos os valores de *<DistVar>, <DirVar>, <DirCorpo>* e *<DirCabeça>.*

Se *unum_too_far_length* = *team_far_length* < <*Distância*> ≤ *team_too_far_length*, então a informação relativa ao número do jogador e os valores de <*DistVar*>, <*DirVar*>, <*DirCorpo*> e <*DirCabeça*> nunca serão dados. A probabilidade de a equipa do jogador ser fornecida na informação visual, decresce linearmente de 1 a 0 entre *team_far_length* e *team_too_far_length*.

Se <*Distância*> > *team_too_far_length* então o jogador será identificado como um anónimo sem qualquer informação do seu número ou equipa.

- **Bola:** Para a bola a situação é similar aos jogadores, isto é:

  Se <*Distância*> ≤ *unum_far_length*, então os valores de <*Distância*>, <*Direcção*>, <*DistVar*>, <*DirVar*> são todos incluídos.

  Se *unum_far_length* < <*Distância*> ≤ *unum_too_far_length* = *team_far_length*, então os valores de <*Distância*>, <*Direcção*> serão visíveis e a probabilidade de que <*DistVar*> e <*DirVar*> sejam visíveis decresce linearmente de 1 para 0 entre *unum_far_length* e *unum_too_far_length*.

  Se <*Distância*> > *unum_too_far_length* = *team_far_length*, então a informação relativa a <*DistVar*> e <*DirVar*> nunca será dada.

8.3.4.2 Informação Auditiva

A informação auditiva é de extrema importância, pois todas as mensagens enviadas pelo árbitro são deste tipo. Desta forma, um jogador não pode desempenhar correctamente o seu papel na equipa sem ser capaz de interpretar correctamente as mensagens auditivas.

Para além das mensagens do árbitro, os jogadores recebem também por via auditiva, mensagens do seu treinador (caso este exista) e dos outros jogadores (colegas de equipa e adversários). O formato para as mensagens auditivas enviadas pelo simulador é:

(**hear** <*Tempo*> <*Direcção*> <*Equipa*> [<*Emissor*>] <*Mensagem*>) ou

(**hear** <*Tempo*> <*Equipa*> [<*Emissor*>])

onde:

<*Tempo*> ::= *Inteiro*

<*Direcção*> ::= *Real*

<*NomeEquipa*> ::= *string*

<*Emissor*> ::= *self* | *referee* | *online_coach_left* | *online_coach_right* | <*Direcção*>

<*Mensagem*> ::= *string*

em que <*Tempo*> indica o tempo corrente. <*Emissor*> pode ser *self* – próprio jogador, *referee* – árbitro, *online_coach_left* ou *online_coach_right* – um dos treinadores ou <*Direcção*> – direcção relativa do jogador que enviou a mensagem.

Diversos parâmetros permitem configurar esquemas de comunicação diferentes no simulador. Os parâmetros principais permitem configurar a distância máxima de

comunicação (*audio_cut_dist*), o número máximo de mensagens que é possível ouvir num dado número de ciclos (*hear_max*, *hear_inc* e *hear_decay*) e a dimensão máxima das mensagens (*say_msg_size*).

Para além da limitação relativa à capacidade auditiva, o servidor implementa também uma restrição relativa ao alcance espacial das mensagens. Uma mensagem só pode ser escutada por jogadores que estejam a uma distância máxima de *audio_cut_dist* do emissor. A distância máxima utilizada em competições oficiais tem sido de 50 metros tornando desta forma, por exemplo, muito difícil a um guarda-redes, escutar qualquer mensagem enviada por jogadores que estejam na metade oposta do campo. As mensagens enviadas pelos treinadores e árbitro não têm limitações de alcance.

Diversas alterações têm sido introduzidas no modelo de comunicação ao longo dos anos. A tabela 14 sumaria os parâmetros de configuração do modelo de comunicações e as alterações realizadas nestes parâmetros de 2001 para 2002.

Parâmetro / *Competição*	*Dimensão Máxima*	*Alcance Máximo*	*Instante de Recepção*	*Frequência Auditiva*	*Identificação do Emissor*
Antes de 2001	*512 Bytes*	*50 metros*	*No mesmo ciclo*	*1 mensagem cada 2 ciclos*	*Equipa e direcção do emissor*
A partir de 2002	*10 Bytes*	*50 metros*	*Ciclo seguinte à emissão*	*1 mensagem por ciclo*	*Equipa e número do emissor*

Tabela 14: Alterações no Modelo de Comunicações de 2001 para 2002

Na tabela é visível que a dimensão máxima das mensagens sofreu um decréscimo considerável de 2001 para 2002 (512 bytes para unicamente 10 bytes) e as mensagens passaram unicamente a ser ouvidas pelos agentes no ciclo seguinte à sua emissão. Como compensação para este decréscimo foi duplicada a frequência auditiva das mensagens e foi introduzida a identificação do emissor (simulando o reconhecimento de vozes humano), ou seja:

<Emissor> ::= self | referee | online_coach_left | online_coach_right | <Number>

Estes novos parâmetros de comunicação tornaram muito mais complicada a decisão de quando e o quê comunicar. Exigiram também a implementação de algoritmos de compressão da informação a comunicar por parte das equipas.

8.3.4.3 Informação Física

Para além da percepção visual e auditiva, o simulador envia também aos agentes percepção física, correspondente ao estado do agente. Esta informação é automaticamente enviada aos jogadores a cada *sense_body_step*, actualmente definido como 100 milisegundos. O formato da mensagem de envio de informação física é:

(**sense_body** *Tempo*
(view_mode *Qualidade Abertura*) (stamina *Energia Esforço*) (speed *ValorVel DirVel*)

(head_angle *DirCabeça*) (kick *ContagemKicks*) (dash *ContagemDashes*)
(turn *ContagemTurns*) (say *ContagemSays*) (turn_neck *ContagemTurnNecks*)
(catch *ContagemCatchs*) (move *ContagemMoves*) (change_view *ContagemCViews*))

A *Qualidade* e *Abertura* indicam respectivamente a qualidade (*high* ou *low*) e abertura (*narrow*, *normal* ou *wide*) actuais da visão do agente.

Os parâmetros *Energia* e *Esforço* referem-se à energia (*Stamina*) e esforço actuais do agente. Indicações da velocidade absoluta e respectiva direcção e do ângulo relativo entre a cabeça e o corpo do agente são fornecidos nos parâmetros seguintes (*ValorVel DirVel* e *DirCabeça*).

Os 8 parâmetros finais dizem respeito à contagem de comandos enviados pelo agente que foram efectivamente executados pelo simulador. Por exemplo, *ContagemKicks* = 50, indicará que o agente executou até ao momento, com sucesso, 50 comandos do tipo *kick*. Esta indicação é útil para detectar problemas de rede, problemas de sincronização com o servidor ou de falta de capacidade de processamento.

8.3.5 Acção dos Agentes

Os agentes da liga de simulação do RoboCup têm à sua disposição um vasto leque de comandos parametrizáveis, tornando o número de acções possível a cada agente em cada instante neste domínio, virtualmente infinito. Estas acções básicas encontram-se sumariadas na tabela 15.

Comando	*Parâmetros*	*Limitações*
(**dash** <*Potência*>)	<*Potência*> ::= *[-100, 100]*	*Clientes podem enviar unicamente um comando do tipo dash, turn, move, kick, tackle ou catch em cada ciclo* *Potências negativas consumem o dobro da energia*
(**turn** <*Momento*>)	<*Momento*> ::= *[-180, 180]*	*Clientes podem enviar unicamente um comando do tipo dash, turn, move, kick, tackle ou catch em cada ciclo*
(**move** <*PosX*> <*PosY*>)	<*PosX*> ::= *[-52.5, 52.5]* <*PosY*> ::= *[-34.0, 34.0]*	*Este comando só pode ser executado no início de cada metade do jogo ou após um golo de uma equipa ou pelo guarda-redes após agarrar a bola.*
(**kick** <*Potência*> <*Direcção*>)	<*Potência*> ::= *[-100, 100]* <*Direcção*> ::= *[-180, 180]*	*A bola deve estar suficientemente próxima do jogador para que este a possa chutar*
(**tackle** <*Potência*>)	<*Potência*> ::= *[0, 100]*	*O comando tackle só pode ser executado em servidores a partir da versão 8.x* *Após executar um comando tackle, o jogador fica imóvel durante alguns ciclos*
(**catch** <*Direcção*>)	<*Direcção*> ::= *[-180, 180]*	*O comando catch só pode ser executado pelo guarda-redes.* *Para o guarda-redes conseguir agarrar a bola, esta tem de estar suficientemente próxima.*

(**turn_neck** <*Direcção*>)	<*Direcção*> ::= *[-180, 180]*	
(**change_view** <*Abertura*> <*Qualidade*>)	<*Abertura*> ::= *narrow* | *normal* | *wide* <*Qualidade*> ::= *high* | *low*	
(**attentionto** <*Equipa*> <*Unum*>) | (**attentionto** *off*)	<*Equipa*> ::= *opp* | *our* | *l* | *r* | *left* | *right* | <*Nome_Equipa*>	
(**say** <*Mensagem*>)	<*Mensagem*> ::= *"Texto"*	*Texto com uma dimensão máxima de 10 bytes. Antes da versão 8.x, a dimensão máxima do texto era de 512 bytes.*
(**pointto** <*Distância*> <*Direcção*>) | (**pointo** *off*)	<*Distância*> ::= *Inteiro* <*Direcção*> ::= *[-180, 180]*	

Tabela 15: Acções Possíveis para os Agentes na Liga de Simulação do RoboCup

As acções podem-se decompor em quatro tipos principais: movimento (***dash***, ***turn*** e ***move***); interacção com a bola (***kick***, ***tackle*** e ***catch***); controle da percepção (***turn_neck***, ***attentionto***, ***change_view***) e comunicação (***say***, ***pointto***). Uma restrição importante consiste no facto de os agentes poderem enviar unicamente um comando do tipo movimento ou interacção com bola por ciclo. Desta forma é impossível, por exemplo, acelerar e chutar a bola ao mesmo tempo. No caso de um agente enviar mais do que um comando primário (movimento ou interacção com a bola) para o servidor, este selecciona um comando aleatoriamente para ser executado.

Embora as acções básicas disponíveis aos agentes na liga de simulação do RoboCup tenham permanecido relativamente estáveis, ao longo dos anos algumas modificações têm sido introduzidas. De entre estas destacam-se a introdução do pescoço flexível e consequente comando ***turn_neck*** em 1999; a introdução da acção ***tackle*** em 2002; a introdução do comando de controle de percepção ***attentionto*** em 2002 e a inclusão do comando ***pointto*** também em 2002.

Uma outra alteração importante relativamente à capacidade de acção dos agentes foi introduzida em 2001 com a inclusão de jogadores heterogéneos. Estes jogadores possuem actuadores com características diferentes e consequentemente capacidades de acção distintas.

8.3.5.1 Movimento – Move

O simulador permite aos agentes enviar três comandos com o objectivo de movimentar os jogadores que controlam. Estes comandos são:

(**dash** <*Potência*>)

(**turn** <*Momento*>)

(**move** <*PosX*> <*PosY*>)

em que <*Potência*> $\subset$ *[minpower, maxpower]*, <*Momento*> $\subset$ *[-180, 180]*, <*PosX*> $\subset$ *[-52.5, 52.5]* e <*PosY*> $\subset$ *[-34.0, 34.0]*.

O Comando ***move*** pode ser utilizado para posicionar directamente a equipa no campo. Permite a um jogador posicionar-se exactamente num dado ponto *(<PosX>, <PosY>)* do campo. No entanto não pode ser utilizado, em condições normais, durante o jogo. Só pode ser utilizado durante os modos *before_kick_off, goal_l* e *goal_r*, ou seja antes do início de uma das partes do jogo e após um golo de uma das equipas. Durante estes modos, os jogadores podem ser posicionados nas suas metades do campo ($x<0$) e podem ser movimentados utilizando este comando, o número de vezes desejado.

O segundo objectivo do comando ***move*** consiste em permitir ao guarda-redes movimentar-se na sua grande área, após ter agarrado a bola (utilizando um comando ***catch***). O guarda-redes pode movimentar-se em conjunto com a bola *goalie_max_moves* (cujo valor por defeito é 2), antes de chutar a bola.

8.3.5.2 Movimento – Aceleração e Energia

O comando ***dash*** é utilizado para acelerar um jogador na direcção do seu corpo com uma dada potência. *Potência* é utilizado como parâmetro, sendo os seus valores mínimo e máximo parâmetros do servidor, actualmente configurados para –100 e 100 respectivamente. O simulador evita que os jogadores possam movimentar-se sempre à máxima velocidade (*player_speed_max*), atribuindo-lhes uma energia limitada. Esta energia (*stamina*) é consumida sempre que o jogador utiliza um comando ***dash***[55]. No início (e no intervalo) de cada jogo, a energia é colocada ao máximo (*stamina_max*, actualmente 4000). Sempre que um jogador acelera no sentido positivo (*Potência* > *0*), a sua energia é diminuída de *Potência*. Acelerar no sentido negativo (*Potência* < *0*), é mais dispendioso em termos de energia, consumindo *2*Potência*.

A energia de cada jogador é modelada utilizando três parâmetros:

- ***Energia***: representa o valor da energia (*stamina*) corrente do jogador e deve ter um valor compreendido entre *[0, stamina_max]*. Um comando ***dash*** não pode ter uma *Potência* superior ao valor de *Energia*;
- ***Esforço***: representa a eficiência dos movimentos do jogador e tem um valor compreendido entre *[effort_min, effort_max]*. Uma baixa capacidade de esforço significa que a *Potência_Real* dos comandos executados pelo jogador vai ser reduzida, ou seja: *Potência_Real = Esforço * Potência*;
- ***Recuperação***: Define a taxa de recuperação de energia dos jogadores. Está contida no intervalo *[recover_min, 1.0]*.

---

[55] Embora actualmente, o comando dash seja o único a consumir energia, tem sido sugerido por diversos investigadores a extensão deste princípio a outros comandos tais como o kick e o turn.

A energia de um jogador é decrementada sempre que este executa um comando ***dash*** mas é restaurada ligeiramente no final de cada ciclo de simulação. Todos os ciclos, se a *Energia* do jogador for inferior a um determinado limiar, o simulador decrementa o *Esforço* e a capacidade de *Recuperação* do jogador. Se o valor da *Energia* for superior a um outro limiar, o *Esforço* é incrementado. A *Recuperação* nunca é incrementada durante o decurso normal do jogo[56]. O algoritmo 1 demonstra em detalhe o modelo completo de cálculo da energia aplicado pelo simulador em cada ciclo.

```
// Redução da energia de acordo com a Potência de dash
SE (comando dash recebido) ENTÃO
        SE Potência > 0 ENTÃO Energia = Energia - Potência
                        SENÃO Energia = Energia - 2*|Potência|
// Redução da capacidade de recuperação quando a energia está abaixo do limiar
de decremento da recuperação
SE Energia <= recover_dec_thr * stamina_max ENTÃO
        SE Recuperação > recover_min ENTÃO Recuperação = Recuperação - recover_dec
        Recuperação = Max(recover_min, Recuperação)
// Redução da capacidade de esforço quando a energia está abaixo do limiar de
decremento do esforço
SE Energia <= effort_dec_thr * stamina_max ENTÃO
        SE Esforço > effort_min ENTÃO Esforço = Esforço - effort_dec
        Esforço = Max(effort_min, Esforço)
// Incremento da capacidade de esforço quando a energia está acima do limiar de
incremento do esforço
SE Energia >= effort_inc_thr * stamina_max ENTÃO
        SE Esforço < effort_max ENTÃO Esforço = Esforço + effort_inc
        Esforço = Min(effort_max, Esforço)
// Restauro de energia de acordo com o valor da Recuperação
Energia = Min (stamina_max, Energia + Recuperação * stamina_inc_max)
```

Algoritmo 1: Algoritmo do Modelo de Energia do Simulador

Quando um agente executa um comando ***dash***, o simulador utiliza a Potência Real do comando de forma a calcular a aceleração a aplicar ao jogador na transição do ciclo actual de simulação para o ciclo seguinte:

$$\left(a_x^t, a_y^t\right) = Potência_Real * dash_power_rate * (\cos(\theta^t), \sin(\theta^t))$$

onde *dash_power_rate* é um parâmetro do servidor utilizado para determinar a dimensão do vector aceleração aplicado ao jogador e θ^t é a direcção do corpo do jogador no ciclo de simulação *t*. O vector de aceleração é então normalizado para um comprimento absoluto máximo de *player_accel_max*, após o que é adicionado vectorialmente à velocidade corrente do jogador. A velocidade resultante é então normalizada para um máximo de

[56] Embora seja restaurada a 1.0 no início de cada parte do jogo.

player_speed_max e é adicionada vectorialmente à posição do jogador para calcular a sua nova posição. Desta forma, *player_speed_max*, corresponde aproximadamente à máxima distância que um jogador pode percorrer num ciclo de simulação[57]. O algoritmo 2 demonstra com maior detalhe o modelo de movimento do simulador:

```
// Normalização da aceleração
        |a_t| = Max(|a_t|, player_accel_max)
// Cálculo da velocidade
        v_t = v_t + a_t
// Normalização da velocidade
        |v_t| = Max(|v_t|, player_speed_max)
// Adição de Ruído à velocidade
        v_t = v_t + Ruído
// Cálculo da nova posição do Jogador
        p_t+1 = p_t + v_t
// Redução da Velocidade do jogoador e colocação da aceleração a 0
        v_t+1 = v_t * player_decay
        a_t+1 = 0
```

Algoritmo 2: Algoritmo do Modelo de Movimento dos Jogadores do Simulador

No caso de o jogador ser de um tipo heterogéneo, o valor de *player_accel_max*, *player_speed_max* ou de outros parâmetros do algoritmo, podem ser diferentes.

8.3.5.3 Movimento – Rotação

Enquanto o comando ***dash*** é utilizado para acelerar um jogador na direcção do seu corpo, o comando ***turn*** é utilizado para alterar a direcção do corpo do jogador. O argumento do comando corresponde ao *Momento* de rotação que pode ter valores compreendidos entre *[minmoment, maxmoment]*, actualmente definidos como –180 e 180 graus respectivamente:

(**turn** <*Momento*>)

Se o jogador se encontra parado no campo, o ângulo de rotação do jogador será igual ao *Momento* incluído no comando ***turn***. No entanto, de forma a tornar a simulação mais real, foi introduzida a noção de inércia. Desta forma, o ângulo de rotação do jogador não é igual ao momento mas sim dependente deste e da velocidade de movimento do jogador. Quando um jogador se desloca com maior velocidade, é mais difícil realizar a rotação devido ao efeito de inércia:

$$\hat{A}ngulo_Real = Momento * (1.0 + Ruído) / (1.0 + inertia_moment * |v_t|)$$

$$Ruído = Random\ (-player_rand,\ player_rand)$$

[57] Desprezando os efeitos do ruído e do vento (caso esteja activado).

Onde *inertia_moment* é um parâmetro do simulador (actualmente com o valor 5.0). *Ruído* é um valor aleatório retirado de uma distribuição uniforme entre *[-player_rand, player_rand]*. $|v_t|$ corresponde ao valor absoluto da velocidade do jogador. Como um jogador só pode executar um comando por ciclo, ou seja não pode realizar um ***dash*** e um ***turn*** simultaneamente no mesmo ciclo, a sua velocidade máxima quando executar um ***turn*** será de *player_speed_max * player_decay*, o que com os parâmetros actuais do servidor, corresponde a um valor de 0.4. Desta forma, o máximo que um jogador pode rodar a esta velocidade será de $\pm 180*1.0/(1.0+5.0*0.4) = \pm 60$[58].

As complexidades introduzidas no modelo de movimento dos jogadores, tais como a adição de ruído, energia limitada, inércia, etc., tornam problemas que à partida poderiam parecer de muito simples resolução, tais como a definição de trajectórias de movimento ou o planeamento de intercepções de bola, em problemas extremamente complexos de resolver.

8.3.5.4 Controlo da Bola – Chuto (*Kick*)

A segunda classe de comandos básicos está relacionada com a interacção dos agentes com a bola. Actualmente[59], existem três comandos deste tipo:

(**kick** <*Potência*> <*Direcção*>)

(**catch** <*Direcção*>)

(**tackle** <*Potência*>)

O comando ***kick*** serve para impulsionar a bola com uma determinada *Potência*, numa dada *Direcção*. O Comando ***catch*** só pode ser executado pelo guarda-redes e serve para o mesmo se "lançar para o chão" numa dada *Direcção* e agarrar a bola. O comando ***tackle*** serve para um jogador se "lançar para o chão" e chutar a bola com uma dada *Potência* na direcção do seu corpo. É útil para desarmar adversários que tenham a bola em sua posse.

Quando um jogador emite um comando ***kick***, deve fornecer dois parâmetros: a *Potência* do chuto, que vai determinar a aceleração dada à bola e que deve ter um valor entre *[minpower, maxpower]* e a *Direcção* do chuto que deve ser dada em graus e estar contida entre *[minmoment, maxmoment]*. Quando o comando ***kick*** chega ao servidor, só será executado se a bola estiver a uma distância do jogador inferior à máxima distância de chuto que é definida por *ball_size + player_size + kickable_margin*, ou seja, se a distância entre o exterior da bola e o exterior do jogador for inferior a *kickable_margin*. Jogadores heterogéneos podem ter diferentes capacidades relativamente aos parâmetros de chuto, nomeadamente diferentes *kickable_margin*.

---

[58] Desprezando os efeitos do ruído.

[59] Nas regras oficiais das competições em 2002 – Versão 8.x do simulador.

A eficiência com que o comando é executado depende da posição relativa da bola ao jogador. Se a bola estiver próxima e à frente do jogador, o comando será executado com maior eficiência do que se a bola estiver mais afastada e atrás do jogador:

$$PotFinal = Potência * kick_power_rate * (1 - 0.25*Dir_Bola/180 - 0.25*Dist_Bola/kickable_margin)$$

Em que *Dir_Bola* é a direcção da bola relativamente ao corpo do jogador e *Dist_Bola* é a distância da bola relativamente ao corpo do jogador. Desta forma se a bola estiver atrás do jogador (*Dir_Bola = 180*) e afastada do mesmo (*Dist_Bola = kickable_margin*) a eficciência do chuto será unicamente 50%.

Existe também um parâmetro *kick_rand* que permite adicionar ruído ao chuto dos jogadores. Para os jogadores normais este parâmetro tem o valor 0, podendo no entanto ter valores distintos para jogadores heterogéneos.

```
// Para cada comando kick válido aplicado por um jogador acelerar a bola na
direcção correspondente ao comando kick e com a potência final
      PARA ComandoKick = 1 ATÉ NComandosKick FAÇA
            SE Valido(ComandoKick) ENTÃO
                  Ba_t = Ba_t + PotFinal * DirecçãoKick
// Para cada comando tackle válido e bem sucedido aplicado por um jogador
acelerar a bola na direcção ao corpo do jogador e com a potência correspondente
      PARA ComandoTackle = 1 ATÉ NComandosTackle FAÇA
            SE Valido(ComandoTackle) E Sucesso(ComandoTackle) ENTÃO
                  Ba_t = Ba_t + PotTackle * DirecçãoPlayer
// Limitação da aceleração da bola
      |Ba_t| = Max(|Ba_t|, ball_accel_max)
// Cálculo da velocidade da bola adicionando a aceleração
      Bv_t = Bv_t + Ba_t
// Limitação da velocidade da bola ao máximo admissível
      |v_t| = Max(|v_t|, ball_speed_max)
// Adição de Ruído à velocidade da bola
      Bv_t = Bv_t + Ruído_Bola(ball_rand)
// Cálculo da nova posição da bola
      Bp_t+1 = Bp_t + Bv_t
// Redução da Velocidade da bola utilizando o "decay" e colocação da aceleração
a 0
      Bv_t+1 = Bv_t * ball_decay
      Ba_t+1 = 0
```

Algoritmo 3: Algoritmo do Modelo de Movimento da Bola

O efeito de um comando *kick* será acelerar a bola na direcção correspondente ao ***kick***. O algoritmo 3 demonstra (de forma simplificada) o processo utilizado no simulador para calcular a velocidade e posição da bola em cada ciclo, tendo em conta os comandos ***kick*** e ***tackle*** emitidos pelos diversos jogadores.

Parâmetro	*Valor*
Minpower	*-100*
maxpower	*100*
minmoment	*-180*
Maxmoment	*180*
kick_power_rate	*0.027*
kick_rand	*0*

Parâmetro	*Valor*
ball_size	*0.085*
player_size	*0.3*
kickable_margin	*0.7*
ball_accel_max	*2.7*
ball_speed_max	*2.7*
ball_decay	*0.94*
ball_rand	*0.05*

Tabela 16: Parâmetros que afectam a acção kick (e valores para competições em 2002).

Embora o modelo de chuto da bola não tenha sofrido alterações nos últimos anos, devido à alteração dos parâmetros (aumento de *kick_power_rate* de 0.016 para 0.027), o problema do chuto da bola foi simplificado e, em muitos casos, não é agora necessário múltiplos ***kicks*** para acelerar a bola para a sua velocidade máxima. Na tabela 16 são apresentados os parâmetros actuais.

8.3.5.5 Controlo da Bola – Agarrar a Bola (*Catch*)

O guarda-redes é o único agente de cada equipa que tem capacidade para executar um comando de ***catch*** de forma a agarrar a bola. O comando ***catch*** toma como único parâmetro a *Direcção* que tem como valores válidos o intervalo *[minmoment, maxmoment]*. Quando o comando ***catch*** chega ao servidor, só será executado se as seguintes pré-condições forem obedecidas: o agente que enviou o comando ser do tipo guarda-redes; o modo de jogo ser *play_on*; a bola estar dentro da grande-área; a bola estar na área coberta pelo ***catch***, ou seja, num rectângulo com comprimento *catchable_area_l* e largura *catchable_area_w* na *Direcção* em que foi efectuado o ***catch*** (figura 58). Neste caso, o guarda-redes terá uma probabilidade de *catch_probability* de agarrar a bola.

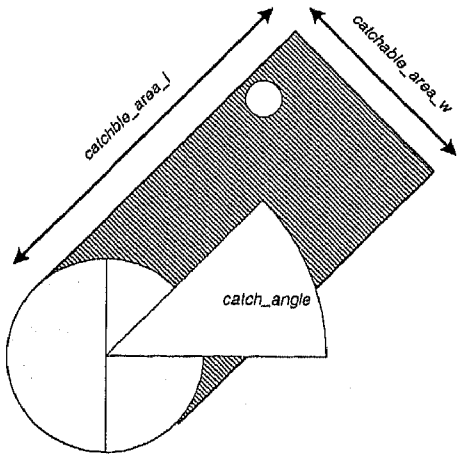

Figura 58: Modelo de Catch do Simulador

Se o comando de ***catch*** não for bem sucedido, demora *catch_ban_cycle* ciclos até que o guarda-redes possa enviar um novo comando de ***catch***[60]. No caso do comando ser bem sucedido, o modo de jogo muda imediatamente para *goalie_catch_ball_[l|r]* e logo após, muda para *free_kick[l|r]*. O guarda-redes pode então utilizar o comando ***move*** para se reposicionar na área e recolocar a bola em jogo. Os parâmetros do servidor importantes para a acção de ***catch*** encontram-se detalhados na tabela 17.

Parâmetro	*Valor*
Minmoment	*-180*
Maxmoment	*180*
catchable_area_l	*2.0*
catchable_area_w	*1.0*

Parâmetro	*Valor*
catch_probability	*1.0*
catch_ban_cycle	*5*
goalie_max_moves	*2*

Tabela 17: Parâmetros que afectam a acção catch.

8.3.5.6 Controlo da Bola - *Tackle*

A introdução do novo comando de ***tackle***, efectuada em 2002[61], tem como principal objectivo aumentar o realismo providenciando uma forma realista de desarmar jogadores adversários que se encontram na posse da bola. O formato do comando é:

(**tackle** <*Potência*>)

O efeito do tackle corresponde ao chuto da bola na direcção em que se encontra virado com a potência especificada no comando. O sucesso do ***tackle*** não é no entanto garantido sendo determinado probabilisticamente baseado na posição da bola relativamente ao jogador que efectua o ***tackle***. Quanto mais próxima estiver a bola, maior será a probabilidade de sucesso do ***tackle***. Se a bola estiver a mais de 2 metros de distância segundo o eixo dos xx ou 1 metro de distância segundo o eixo dos yy do jogador, esta probabilidade será nula

Após tentar efectuar um ***tackle*** um jogador fica imóvel durante 10 ciclos. A informação visual contém uma indicação se um dado jogador (visível) efectuar um ***tackle***. Um jogador que efectuar um ***tackle*** recebe informação sobre o número de ciclos que ainda tem de esperar até poder efectuar um novo comando.

8.3.5.7 Controlo da Percepção – Rotação do Pescoço Flexível

Utilizando o comando ***turn_neck***, os jogadores podem rodar o seu pescoço flexível, obtendo um importante controlo sobre a percepção. A sintaxe do comando é:

---

[60] Pode no entanto, neste intervalo, enviar comandos de kick para chutar a bola.

[61] O comando tackle encontra-se disponível a partir do servidor 8.x

(**turn_neck** <*Ângulo*>)

O comando ***turn_neck*** modifica a direcção do pescoço do jogador e consequentemente do seu ângulo de visão relativamente ao seu corpo de *Ângulo* graus. O pescoço pode ser rodado entre *[minneckmoment, maxneckmoment]*, correspondendo a *[-180, 180]* graus na versão actual do simulador, relativamente ao corpo do jogador. No entanto, o valor final da direcção do pescoço relativamente ao corpo tem de assumir um valor compreendido entre *[minneckang, maxneckang]*, correspondendo a *[-90, 90]* graus. Quando um comando ***turn_neck*** conduz a um valor ilegal deste ângulo, o comando é adaptado de forma a que a direcção do pescoço relativamente ao corpo, permaneça dentro dos limites. No controlo da direcção de visão utilizando este comando é necessário ter em conta que o ângulo de visão varia também com o comando ***turn***, mesmo que nenhum comando ***turn_neck*** seja executado.

Contrariamente ao comando ***turn***, não é adicionado ruído ao comando ***turn_neck*** nem a eficiência deste depende da velocidade do jogador. Desta forma, um comando *(**turn_neck** 80)* conduzirá a que o pescoço do jogador rode exactamente de 80 graus no sentido positivo relativamente ao corpo do jogador, independentemente da velocidade a que o jogador se desloque.

8.3.5.8 Controlo da Percepção – Configuração da Visão

A configuração da percepção visual de um agente pode ser efectuada utilizando o comando ***change_view*** com a seguinte sintaxe:

(**change_view** <*Abertura*> <*Qualidade*>)

Este comando toma dois parâmetros: *Abertura* do cone de visão (*narrow, normal* ou *wide*) e a *Qualidade* da informação visual (*high* ou *low*). Os jogadores podem utilizar o comando ***change_view*** de forma a trocarem a *Abertura* ou *Qualidade* da informação visual por uma maior frequência do envio da mesma por parte do servidor. Uma maior qualidade visual e um cone de visão mais largo conduzem a informação visual menos frequente.

8.3.5.9 Controlo da Percepção – Configuração da Audição

A introdução em 2002 do comando ***attentionto*** introduziu novos desafios no controle da percepção dos agentes da liga de simulação. Utilizando este comando é agora possível efectuar uma escolha selectiva de quais os agentes que são prioritários ouvir em cada instante. A sintaxe do comando é:

(**attentionto** <*Equipa*> <*Unum*>) | (**attentionto** *off*)

onde <*Equipa*> ::= *opp* | *our* | *l* | *r* | *left* | *right* | <*Nome_Equipa* > permite indicar qual a equipa a que pertence o jogador e <*Unum*> é um inteiro que permite especificar o número desse jogador. Cada comando ***attentionto*** sobrepõe-se a um anterior.

A primeira forma do comando especifica o jogador ao qual deve ser dada atenção. As mensagens deste jogador deverão ser ouvidas preferencialmente às dos outros jogadores. Se esse jogador enviar várias mensagens no mesmo ciclo, uma é escolhida ao acaso. No caso de esse jogador não enviar qualquer mensagem, uma mensagem de um outro jogador é escolhida aleatoriamente.

A segunda forma do comando permite desligar a atenção dada a um determinado jogador de forma a que as mensagens a escutar sejam escolhidas aleatoriamente de qualquer jogador.

8.3.5.10 Comunicação - Say

O modelo de comunicação do *soccerserver* baseia-se na possibilidade de envio de mensagens para o ambiente, que são distribuídas pelos agentes que estão próximos. Este modelo de comunicação sofreu em 2002 a introdução de uma série de restrições à comunicação auditiva e a adição de um novo comando que permite a um agente apontar numa determinada direcção.

Sempre que um agente deseje enviar uma mensagem aos restantes jogadores, poderá emitir um comando say com a forma:

(**say** <*Messagem*>)

O parâmetro único deste comando *Messagem* corresponde à mensagem que o jogador deseja emitir. O tamanho desta mensagem está limitado pelo parâmetro *say_msg_size* (correspondente a 512 bytes até 2001 mas reduzido para apenas 10 bytes em 2002). Os caracteres da mensagem têm de provir do conjunto [0..9a..zA..Z ().+-*/?_<>] (sem os parênteses), o que implica a disponibilidade de 74 símbolos unicamente para efectuar a comunicação. A explicação detalhada dos parâmetros de comunicação e respectivos significados e alterações ao longo do tempo foi já realizada na secção 8.3.4.2.

8.3.5.11 Comunicação - Pointto

Desde 2002 os jogadores têm também a capacidade de comunicar visualmente com os restantes jogadores apontando para pontos específicos do campo. A sintaxe deste novo comando é:

(**pointto** <*Distância*> <*Direcção*>) | (**pointto** *off*)

Um jogador após apontar para uma dada direcção permanecerá durante um mínimo de 5 ciclos e um máximo de 20 ciclos ou outro comando pointto emitido, a apontar nessa direcção. Os jogadores permanecerão a apontar para a mesma direcção independentemente do seu movimento no campo.

Os restantes jogadores recebem na sua informação visual a direcção em que um dado jogador está a apontar, na forma:

(p <Equipa> <Jogador>) <Distância> <Direcção> <DistVar> <DirVar> <DirCorpo> <DirCabeça> <DirAponta> ou

(p <Equipa>) <Distância> <Direcção> <DirAponta>

Ruído dependente da distância a que um jogador se encontra do jogador que está a apontar é adicionado.

8.3.6 Agentes Heterogéneos

Os agentes heterogéneos foram introduzidos em 2001 como forma de estimular a investigação realizada em diversos domínios, com ênfase para a alocação dinâmica de recursos. No início de cada jogo são sorteados aleatoriamente seis tipos de agentes heterogéneos que possuem características distintas dos jogadores normais. Estas características incluem, entre outras, velocidade, aceleração, recuperação de energia e momento de inércia distintos. A tabela 18 apresenta um resumo dos parâmetros que podem sofrer alterações nos jogadores heterogéneos.

Parâmetro	*Valor*	*Alcance*	*Descrição*
player_speed_max	*1.0*	*[1.0, 1.2]*	*Jogador com maior velocidade máxima*
stamina_inc_max	*45.0*	*[25.0, 45.0]*	*Jogador que recupera menos energia em cada ciclo*
player_decay	*0.4*	*[0.4, 0.6]*	*Jogador cuja velocidade decresce mais lentamente*
inertia_moment	*5.0*	*[5.0, 10.0]*	*Jogador que roda com mais dificuldade em movimento*
dash_power_rate	*0.006*	*[0.006, 0.008]*	*Jogador que acelera mais rapidamente*
player_size	*0.3*	*[0.1, 0.3]*	*Jogador com um corpo mais pequeno*
kickable_margin	*0.7*	*[0.7, 0.9]*	*Jogador com maior área de chuto*
kick_rand	*0.0*	*[0.0, 0.1]*	*Jogador com menor precisão de chuto*
extra_stamina	*0.0*	*[0.0, 100.0]*	*Jogador com alguma energia extra*
effort_max	*1.0*	*[0.8, 1.0]*	*Jogador que entra mais dificilmente em esforço*
effort_min	*0.6*	*[0.4, 0.6]*	*Jogador que sai mais facilmente de esforço*

Tabela 18: Parâmetros dos Jogadores Heterogéneos

Estes parâmetros têm sido utilizados como base para a definição de *"tradeoffs"* em cada competição de forma a garantir que embora os jogadores heterogéneos possuam algumas características físicas melhores que os jogadores normais, possuem também no entanto algumas características piores. Exemplos de *"tradeoffs"* utilizados em competições incluem a troca de aceleração por recuperação de energia (correspondendo a um jogador que acelera rapidamente mas também se cansa rapidamente) ou a troca de área de chuto por precisão de chuto (correspondendo a um jogador com pernas longas capaz de cortar bolas aos adversários com maior facilidade mas que têm dificuldades na realização de passe precisos).

No início do jogo, o treinador de cada equipa é informado de quais os tipos de jogadores heterogéneos que dispõe para o jogo. Pode então substituir os jogadores normais por jogadores de tipos heterogéneos até a um máximo de 3 de cada tipo. No decurso do jogo, o treinador dispõe de três substituições que poderá utilizar para refrescar a equipa (os

jogadores que entram possuem a energia e capacidade de recuperação ao máximo) e alterar os tipos de jogadores (introduzindo jogadores heterogéneos de tipos distintos).

8.3.7 O Treinador

O treinador é um agente privilegiado que pode ser utilizado por uma equipa para auxiliar os seus jogadores. O simulador disponibiliza dois tipos de treinadores: O treinador *on-line* e o treinador *off-line*. Ambos os treinadores recebem informação global e livre de erros relativa a todos os objectos presentes no campo. Para além disto, o treinador recebe ainda todas as mensagens enviadas pelos jogadores e pelo árbitro e pode enviar mensagens audíveis por todos os jogadores da sua equipa.

Diversas restrições são incluídas neste agente de forma a que não possa ser utilizado como um controlador centralizado da equipa. No entanto, é também neste cliente que se situam importantes capacidades para o controlo global da equipa tais como a capacidade para seleccionar os jogadores heterogéneos a utilizar e efectuar substituições durante o jogo (de jogadores cansados ou de forma a efectuar a troca do tipo de jogadores).

8.3.7.1 Capacidades do Treinador *off-line*

A diferença essencial entre o treinador *off-line* e o treinador *on-line* é que o primeiro não pode ser utilizado durante jogos oficiais, servindo exclusivamente para auxiliar na fase de desenvolvimento da equipa. O treinador *off-line* pode exercer um muito maior controlo sobre a equipa relativamente ao treinador *on-line*. Ele pode controlar o modo de jogo desactivando o árbitro artificial se necessário. Pode ainda movimentar os jogadores de ambas as equipas e a bola para qualquer ponto do campo. Pode também atribuir velocidades e orientações aos jogadores o que se revela muito útil para ensaiar determinadas situações durante os treinos. Com o auxílio deste agente é muito simples criar situações de treino semelhantes às que os treinadores reais de futebol criam durante os treinos de equipas reais de futebol.

8.3.7.2 Capacidades do Treinador on-line

O treinador *on-line* é utilizado durante os jogos para fornecer informação adicional e enviar instruções aos jogadores. As suas capacidades são muito limitadas pois não pode controlar o modo de jogo ou os jogadores. Adicionalmente, para que ele não possa ser utilizado como um controlador centralizado, as suas capacidades de comunicação com os jogadores são fortemente restringidas. O treinador só pode comunicar com os seus jogadores quando o jogo está parado ou no decurso normal do jogo, mas utilizando mensagens codificadas numa linguagem especial e que chegam aos jogadores com atraso relativamente ao envio.

Embora as restrições às capacidades do treinador sejam severas, dado que este agente recebe informação global e livre de erros e não tem solicitações em tempo-real, pode gastar o seu tempo realizando análises mais complexas acerca do comportamento do adversário e da estratégia, tácticas e formações a utilizar pela sua equipa. Constitui desta forma uma ferramenta valiosa para analisar as forças e fraquezas do oponente, analisar o decurso do jogo e decidir quais os conselhos tácticos a fornecer aos seus jogadores.

8.3.7.3 Linguagem "Standard" de Comunicação

De forma a permitir a um treinador artificial treinar uma equipa de futebol robótico simulado é necessário definir uma linguagem de comunicação apropriada. Esta linguagem será tanto mais apropriada quanto genérica e de compreensão geral for. Desta forma, uma nova área de investigação surgiu na comunidade RoboCup com o objectivo de criar uma linguagem standard que permita a um dado treinador artificial ser capaz de treinar diferentes equipas de futebol robótico simulado.

A primeira linguagem criada neste sentido foi COACH UNILANG [Reis e Lau, 2002] definida como uma linguagem standard de alto-nível para treinar uma equipa de futebol (robótico). COACH UNILANG é uma linguagem baseada em conceitos comuns do futebol real e robótico, tais como: regiões, períodos de tempo, situações, tácticas, formações, tipos e comportamentos de jogadores. Inclui construtores que permitem definir tácticas genéricas de futebol de uma forma directa e muito simples. Uma descrição mais detalhada da linguagem pode ser encontrada no capítulo 9 deste trabalho ou em [Reis e Lau 2002].

A especificação da linguagem COACH UNILANG foi tornada pública em Janeiro de 2001 e recolheu um amplo apoio da comunidade do RoboCup. No entanto a federação RoboCup optou por utilizar como linguagem *"standard"* no simulador oficial uma linguagem de baixo-nível designada CLang, importando unicamente alguns conceitos básicos da linguagem COACH UNILANG. Esta linguagem *"standard"* foi utilizada como base para uma competição de treinadores no RoboCup 2001 em Seattle. No entanto, unicamente 4 equipas participaram nesta competição. Adicionalmente, foi concluído pela maioria dos participantes e por grande parte dos assistentes da competição que a linguagem standard não era suficientemente expressiva e era demasiado baixo-nível para treinar uma boa equipa de futebol robótico simulado. Desta forma, nenhum dos participantes utilizou com sucesso esta linguagem para treinar a sua própria equipa na competição oficial e adicionalmente foi concluído que na competição de treinadores, as quatro equipas jogavam melhor sem treinador do que com um treinador comunicando em CLang.

Estas primeiras experiências levaram à conclusão da necessidade de uma linguagem de alto-nível mais preocupada com o nível da equipa, do que com o comportamento individual dos jogadores. Bons jogadores artificiais de futebol simulado já possuem

capacidades de decisão individual apuradas e necessitam sobretudo de orientação sobre como funcionarem como elementos de uma equipa. No entanto, para o ano de 2002, contra a opinião da comunidade, o comité técnico decidiu-se pela manutenção da linguagem Clang como base da competição de treinadores, importando no entanto para esta linguagem uma parcela significativa dos conceitos introduzidos na linguagem COACH UNILANG[62] [Reis e Lau, 2002].

8.4 Abordagens mais significativas ao Futebol Robótico Simulado

Uma das principais motivações das competições RoboCup consiste na disponibilização de um problema standard composto por inúmeros subproblemas cuja resolução pode ser efectuada através da utilização de diferentes metodologias. Desta forma, diferentes equipas utilizam metodologias distintas para a resolução do mesmo problema. Por exemplo, no problema de chutar a bola, algumas equipas utilizam abordagens analíticas [Stone et al., 1999], outras utilizam aprendizagem por reforço [Riedmiller et al., 2000] e outras metodologias de optimização *on-line* [Reis e Lau, 2001a].

As principais metodologias desenvolvidas pelas principais equipas do RoboCup ao longo dos últimos anos foram [Reis, 2001a] [Reis, 2001b]:

- **Formações e Papéis.** Metodologias que permitem definir formações espaciais e atribuir aos jogadores papéis e eventualmente formas de efectuar a troca desses papéis;
- **Estratégias e tácticas.** Metodologias que permitem definir a alto-nível a estratégia para um dado jogo e alterar o comportamento da equipa globalmente durante esse jogo.
- **Modelização de Oponentes.** Criação de modelos da equipa oponente e dos seus jogadores de forma a poder adequar o comportamento da equipa a esse oponente.
- **Metodologias de aprendizagem.** Utilização de metodologias de aprendizagem tais como a aprendizagem por reforço de forma a desenvolver *low-level skills* (chuto, drible, intercepção, etc.) ou métodos de decisão individual (passar, chutar, driblar) ou mesmo de decisão colectiva.
- **Técnicas de Optimização.** Aplicação de metodologias de optimização para o desenvolvimento de *low-level skills* para os jogadores.

---

[62] A equipa FC Portugal embora tenha manifestado a sua oposição ao formato e linguagem utilizada na competição, participou e venceu a mesma. Na competição ficou mais uma vez provado que a linguagem CLang tem um efeito detrimental nas equipas treinadas.

- **Planeamento Flexível.** Desenvolvimento de metodologias que permitam criar, durante o jogo, planos flexíveis para a execução colectiva de determinadas tarefas durante o jogo.
- **Representação do conhecimento do futebol real.** Desenvolvimento de metodologias que permitam efectuar a recolha, representação e utilização de conhecimento recolhido através do contacto com especialistas da área (treinadores, jogadores e jornalistas).
- **Utilização do treinador *on-line*.** Definição de metodologias que permitam ao treinador analisar o jogo e decidir, em cada instante, qual a melhor táctica para a sua equipa.
- **Construção de Ferramentas de Análise e Debug.** Construção de ferramentas que permitam analisar jogos de futebol robótico simulado *on-line* ou *off-line* e que permitam analisar a informação disponível aos agentes e os seus processos de raciocício ao longo dos jogos.

8.5 Trabalho Relacionado e Aplicações do RoboCup

Existem diversas aplicações interessantes das tecnologias desenvolvidas no contexto da liga de simulação do RoboCup. Algumas delas encontram-se já em progresso, enquanto outras possuem um grande potencial para serem realizadas num futuro próximo. De entre estas aplicações destacam-se [Reis, 2001a] [Reis, 2002g]:

- *Routing* de pacotes de rede;
- Coordenação de pilotos de helicópteros sintéticos e análise de cenários de guerra;
- Simulação de combate aéreo e de Veículos (aéreos) inteligentes e autónomos;
- Coordenação automática de pessoas;
- Combate a catástrofes tais como incêndios florestais ou no âmbito do *RoboCup Rescue*;
- Realidade virtual;
- Educação;
- Análise de jogos de futebol;
- Coordenação de *AGVs* (*Automated Guided Vehicles*);
- Robótica industrial e controlo de processos industriais;
- Controlo de satélites;

- Limpeza de minas e de lixo radioactivo;
- Controlo de robôs hospitalares;
- Controlo Inteligente de Câmaras.

Stone et al. [Stone et al., 2000] na Carnegie Mellon University, aplicaram com sucesso diversas metodologias desenvolvidas no âmbito da liga de simulação do RoboCup ao domínio do *routing* de pacotes de rede. De entre estas metodologias destacam-se o *TPOT-RL – Team Partitioned, Opaque Transition Reinforcement Learning* [Stone et al., 2000], a aprendizagem máquina hierárquica em camadas (*Layered Learning*) [Stone et al., 2000], as formações adaptativas [Stone et al., 1999], a comunicação em canal único com reduzida largura de banda [Stone et al. 1999], os *Predictive, Locally Optimal Skills (PLOS)* e o posicionamento utilizando atracções e repulsões (SPAR) [Stone, 1998]. Embora os resultados publicados por Stone et al. na aplicação a este domínio não sejam tão extensos como os de aplicação ao futebol robótico, os resultados demonstram que muitas das metodologias desenvolvidas no RoboCup podem ser generalizadas para outros domínios[63].

Milind Tambe e Gal Kaminka na University of South Califórnia partiram de uma perspectiva inversa, aplicando ao RoboCup simulado metodologias desenvolvidas originalmente para outros domínios. O modelo de trabalho em equipa STEAM foi desenvolvido originalmente para a coordenação de pilotos de helicópteros em situações militares complexas e de larga escala [Tambe, 1997]. O RoboCup foi um domínio adicional onde esta metodologia foi demonstrada sem que, no entanto, tenha obtido resultados muito significativos. A mesma tecnologia formou a base do TEAMCORE, uma arquitectura de integração de aplicações distribuídas que foi demonstrada em diversos outros domínios tais como a coordenação automática de humanos [Tambe, 2000]. Gal Kaminka desenvolveu também, na mesma Universidade, um trabalho focalizado na monitorização e diagnóstico de equipas, que foi avaliado em três domínios distintos: futebol simulado, simulação militar e coordenação de humanos [Kaminka, 2000].

No domínio da detecção distribuída de incêndios florestais destaca-se o trabalho desenvolvido por Mikhail Prokopenko no CSIRO, Austrália [Prokopenko, 2002].

Uma aplicação relevante das metodologias desenvolvidas no âmbito do RoboCup, consiste na análise de jogos de futebol real. Diversos sistemas comerciais, tais como *Match Analysis* [MatchAnalysis, 2002] existem exactamente com a função de analisar jogos de futebol, providenciando estatísticas de alto-nível sobre o jogo para treinadores e profissionais da comunicação social. As equipas técnicas de uma percentagem muito significativa das principais equipas de futebol mundiais utilizam sistemas de análise automática de jogos. As metodologias desenvolvidas no âmbito da liga de simulação do

---

[63] O trabalho de Stone na sua tese de doutoramento e a sua aplicação nesta área, levou a que este cientista fosse contratado pela AT&T Research.

RoboCup, nomeadamente no que diz respeito ao cálculo automático de estatísticas de jogo, analisadores de jogo e comentadores automáticos têm, num futuro muito próximo, aplicação directa no domínio do futebol real. No âmbito do projecto FC Portugal, baseado no contacto com profissionais da área do futebol (jogadores, treinadores e jornalistas) foi desenvolvido um sistema de análise estatística de jogos de futebol, uma metodologia de definição de uma estratégia para um jogo de futebol e uma linguagem standard para comunicação treinador – equipa. Encontra-se actualmente em estudo a aplicação destas metodologias ao domínio do futebol real em Portugal.

O *RoboCup Rescue* é um dos domínios mais promissores para aplicação das metodologias desenvolvidas no âmbito do RoboCup. Esta nova iniciativa da Federação RoboCup está preocupada com o planeamento e execução de operações de resgate e salvamento em desastres de larga escala tais como terramotos e incêndios.

A robótica Industrial é outra aplicação futura do RoboCup que assume grande importância. O envio de robôs autónomos e inteligentes para situações perigosas diminui o número de vidas humanas colocadas em risco. Tarefas complexas em que as tecnologias principais desenvolvidas no RoboCup podem ser aplicáveis, incluem a exploração de terreno, gestão de armas nucleares, exploração lunar, limpeza de lixo radioactivo e a limpeza de minas.

8.6 O RoboCup como Competição Científica

As competições científicas têm em geral um grande potencial para acelerar o progresso científico num dado domínio. No entanto, a realização de competições como o RoboCup tem também diversos perigos para o avanço da ciência.

Os principais perigos das competições científicas [Stone, 1998] estão centrados na obsessão por vencer, com o desenvolvimento de soluções dependentes do domínio e com o custo do equipamento necessário para ter uma equipa competitiva.

Especialmente quando prémios monetários estão envolvidos, existe um certo incentivo a manter secretas as técnicas "vencedoras" de ano para ano, sem que o seu desenvolvimento permita avançar a ciência em geral. No RoboCup, a atribuição do "*scientific challange award*" procura diminuir o risco do secretismo, incentivando e premiando a publicação de trabalho científico de qualidade. Por outro lado, em diversas ligas, nomeadamente na liga de simulação, a disponibilização do binário de anos anteriores é condição necessária para a admissão à competição no ano seguinte. A disponibilização de código fonte é também largamente incentivada. Os campeões da liga de simulação de diversos anos (CMUnited, FC Portugal e Tsinghuaeolus) e outras equipas bem sucedidas (tais como os Karlsruhe Brainstormers, UvaTrilearn ou YowAI) têm disponibilizado partes ou mesmo a totalidade do seu código fonte na Internet.

O desenvolvimento de soluções dependentes do domínio é outro perigo das competições, em particular em domínios complexos como o RoboCup. Alguns autores consideram [Stone, 1999] que para produzir uma equipa de futebol robótico bem sucedida, a investigação em metodologias de coordenação e cooperação não é suficiente. No entanto, é convicção do autor que a investigação que leva realmente a obter bons resultados no RoboCup simulado situa-se a um nível elevado. Os detalhes de baixo nível muito ligados ao domínio, tais como desenvolver bons algoritmos de intercepção de bola, chuto, drible, etc. são de muito menor importância do que o desenvolvimento de algoritmos de coordenação, tais como alocação e troca de papéis, posicionamento estratégico, formalizações do conceito de estratégia, formas de definir e implementar tácticas e formações, etc.

O custo para ter uma equipa competitiva é muito relevante sobretudo nas ligas robóticas. A utilização de tecnologias caras já existentes pode conduzir a um grande benefício na competição sem que conduza a um avanço da ciência. Uma solução que a federação RoboCup tem utilizado nas ligas robóticas consiste em impor restrições às dimensões, sensores e actuadores dos robôs, tais como limites na potência de chuto ou na capacidade de controlo de bola.

Peter Stone afirma que um perigo da competição RoboCup está relacionado com a dificuldade em entrar nela, pois as equipas antigas têm uma considerável vantagem [Stone, 1999]. No entanto, ao longo dos últimos anos esta conclusão tem-se demonstrado errada no âmbito do RoboCup. Em 2000 e 2001, a liga de simulação foi vencida por equipas que se estrearam na competição (FC Portugal e Tsinghuaeolus). Factores que contribuem para este facto são a disponibilidade de código fonte (de diversas fontes) na Internet e as alterações às regras introduzidas de ano para ano pelo comité técnico da simulação. A equipa FC Portugal que venceu o RoboCup 2000 utilizou parte do código fonte (contendo os *low-level skills* e comunicação com o servidor) disponibilizado pelo campeão anterior (CMUnited99). A equipa Tsinghuaeolus que venceu o campeonato do mundo em 2001, utilizou o código fonte disponibilizado pelo FC Portugal em 2000 (contendo a definição de tácticas, formações e posicionamento estratégico).

Um outro perigo da competição consiste numa avaliação incorrecta dos resultados. Evidentemente que um único jogo entre duas equipas, não é uma experiência que permita retirar conclusões científicas válidas. Não se pode concluir que se a equipa *X* venceu a equipa *Y*, então todas as metodologias utilizadas pela equipa *X* serão mais bem sucedidas do que as utilizadas pela equipa *Y* [Stone, 1998]. Para além disso, o resultado do jogo é um dos muito factores a ser considerado numa perspectiva científica de análise. Factores como o tempo de posse de bola por região do campo, número de remates à baliza, cruzamentos, passes bem sucedidos, defesas do guarda-redes, etc. são muito relevantes na análise do jogo. Experiências controladas no laboratório têm de ser efectuadas para validar os resultados de investigação e as contribuições de investigação de cada equipa [Stone, 1999].

Embora existam diversos perigos na realização de competições científicas, os benefícios de as realizar superam largamente os problemas que advêm dessa realização. Os benefícios principais, no caso da liga de simulação, situam-se ao nível da existência de uma plataforma comum que permita comparar soluções, trocar ideias, aferir do progresso realizado e realizar competições com a natural motivação dada pela vontade de vencer. A competição em si própria, constitui uma forte inspiração para a investigação. A vontade de vencer, criando para tal, uma equipa competitiva e como tal criando soluções inovadoras para os diversos problemas, que é necessário resolver para criar essa equipa, tem constituído ao longo dos anos um dos motores para o progresso científico no âmbito do RoboCup.

A realização de competições RoboCup permite juntar um vasto conjunto de investigadores que realizam investigação em diversos domínios relacionados com Sistemas Multi-Agente e que utilizam o futebol robótico como domínio de aplicação. As equipas têm de resolver os mesmos problemas no mesmo domínio e utilizam para tal metodologias distintas. As diferentes soluções podem, através da competição, ser comparadas de forma directa. Como a competição RoboCup é realizada todos os anos e os binários das melhores equipas do ano anterior são disponibilizados, verifica-se que todas as equipas antes de participarem no campeonato procuram desenvolver metodologias que lhes permitam vencer as melhores equipas do ano anterior. Esta motivação de vencer os melhores contribui decisivamente para o progresso da investigação na área.

A existência do *soccerserver* e a disponibilização de uma enorme quantidade de equipas na Internet, permite ainda aos investigadores realizarem testes no seu laboratório e terem uma noção imediata do seu progresso. Por exemplo, de forma a testar a utilidade de uma determinada metodologia é possível realizar testes no laboratório jogando contra uma outra equipa, seleccionada de entre a grande variedade de equipas disponibilizadas na Internet, activando ou não a técnica em estudo.

O entusiasmo que o RoboCup consegue induzir aos professores e aos estudantes a todos os níveis é outro benefício evidente. É mais fácil envolver estudantes em investigação séria utilizando uma aplicação motivadora como o RoboCup [Stone, 1999]. Usualmente as equipas incluem desde professores até estudantes de pós-graduação e graduação, todos trabalhando em conjunto. Adicionalmente, as competições são muito úteis para leccionar cursos da área. Diversos cursos quer ao nível da graduação, quer ao nível de pós-graduação, utilizam o simulador do RoboCup como base para aplicação das metodologias e usualmente incluem uma mini-competição entre os estudantes.

Outro benefício da realização de competições RoboCup consiste no facto de impor prazos bem definidos para a criação de um Sistema Multi-Agente totalmente funcional. Usualmente, o desenvolvimento de agentes realmente funcionais é muitas vezes desprezado quando se realiza investigação científica. No entanto, para competir no RoboCup não basta colocar um ou dois módulos dos agentes em funcionamento e escrever

um artigo explicando como esses módulos um dia irão funcionar. É necessário desenvolver agentes completos que realmente funcionem do ponto de vista individual e como componentes do Sistema Multi-Agente.

8.7 Conclusões

O RoboCup veio estimular consideravelmente a investigação realizada em Inteligência Artificial Distribuída e Robótica Inteligente. Mas mais do que isso, veio chamar a atenção da comunicação social e do público em geral para esta investigação. As suas ligas colocam problemas de investigação distintos mas ao mesmo tempo interrelacionados. A liga de simulação analisada com maior detalhe neste trabalho, inclui complexidades encontradas nos sistemas robóticos (tais como erros na percepção e acção) mas ao mesmo tempo utiliza um cenário realista do ponto de vista da tarefa cooperativa a executar pelas equipas de agentes – jogar um desafio de futebol.

A liga de simulação coloca um vasto conjunto de desafios aos investigadores da área da Inteligência Artificial e Sistemas Multi-Agente. As características do domínio e desafios associados mais importantes colocados pelo simulador incluem [Reis, 2001a]:

- **Simulação em tempo-real.** O simulador actualiza o mundo em intervalos de tempo discretos (ciclos de simulação), cada qual com a duração de 100ms. Neste intervalo de tempo, os agentes recebem diversos tipos de informação sensorial e têm de enviar pedidos de execução de acções para o simulador. Isto requer aos agentes, capacidade de raciocínio em tempo-real.
- **Diversas fontes de Informação Sensorial e disponibilidade de sensores configuráveis.** Os agentes possuem três tipos de sensores: visual, auditivo e físico. O sensor visual fornece aos agentes informação (distâncias, direcções, etc.) sobre os objectos que se encontram no seu campo de visão. Funciona também como um sensor de proximidade fornecendo informação sobre objectos que estão muito próximos. A informação visual é bastante incompleta (os agentes possuem cones de visão reduzidos), é fornecida numa perspectiva relativa ao agente e com erros. No entanto, a disponibilidade de marcadores ao longo do campo permite ao agente converter esta informação em coordenadas absolutas. O sensor auditivo fornece aos agentes a informação relativa às mensagens emitidas pelos outros agentes, pelo árbitro e pelo treinador. Tem no entanto um alcance e capacidade muito limitados. O sensor físico fornece informação sobre a velocidade e energia do jogador e acções executadas pelo agente. Os sensores são configuráveis, permitindo a sua utilização inteligente por parte dos agentes. O sensor visual é configurável no que diz respeito à qualidade/frequência da informação e ângulo relativo ao corpo do agente. O sensor auditivo é configurável relativamente à prioridade atribuída às mensagens de diferentes agentes.

- **Modelo energético realista.** O simulador evita que os agentes corram continuamente no campo à máxima velocidade, atribuindo a cada jogar energia limitada. Sempre que o jogador executa um comando *dash*, gasta energia. No entanto, a energia é também restaurada em cada ciclo de acordo com a capacidade de recuperação do jogador. No caso de os jogadores permanecerem cansados durante muito tempo perdem capacidade de recuperação e eficiência de movimentos. Este modelo energético torna necessário que os agentes raciocinem em termos de consumo dos seus recursos energéticos, tendo em conta a importância das acções a executar.
- **Comunicação pouco fiável e com baixa largura de banda.** Ao contrário do que acontece na maioria das aplicações de SMA, neste domínio, os agentes não podem comunicar directamente uns com os outros. As comunicações são efectuadas através do simulador usando os protocolos *say* e *hear* que restringem a comunicação de diversas formas. Acresce a este facto que os 22 agentes utilizam o mesmo canal de comunicação que possui reduzida largura de banda e é muito pouco fiável.
- **Percepção e acção assíncronas.** Os agentes podem executar uma acção em cada 100ms. No entanto, a informação visual (com os parâmetros visuais por defeito), chega aos agentes de 150 em 150ms. Desta forma, e uma vez que é crucial aos agentes executarem uma acção em cada ciclo, torna-se necessário que os agentes sejam capazes de enviar comandos para o servidor mesmo sem receber informação sensorial.
- **Ambiente multi-objectivo, parcialmente cooperativo e parcialmente adverso.** Os agentes possuem dois objectivos de alto-nível contraditórios: defender para evitar sofrer golos e atacar para marcar golos. O ambiente é parcialmente cooperativo e os agentes têm de se comportar como membros de uma equipa. É também parcialmente adverso, no sentido que existe uma outra equipa de agentes exactamente com os objectivos opostos.
- **Vasto conjunto de acções de baixo-nível disponíveis.** Os agentes possuem acções de duas categorias: primárias (*kick, dash, turn, tackle, catch* e *move*) e secundárias (*turn_neck, change_view, attentionto, pointto* e *say*). Em cada ciclo, unicamente uma das acções primárias pode ser executada, embora diversas acções secundárias possam ser executadas concorrentemente. Se o agente enviar mais do que uma acção primária, o servidor escolhe aleatoriamente qual das acções enviadas vai executar, ignorando todas as outras. Não existe nenhuma garantia que uma dada acção seja executada pelo servidor, pelo que é necessário confirmar a execução das acções utilizando os sensores. Ruído é adicionado às acções de diversas formas.
- **Necessidade de transformar as acções de baixo-nível em comandos de alto-**

nível. De forma a executar uma determinada acção simples, tal como chutar a bola numa dada direcção, driblar com a bola ou interceptar uma bola em movimento, é necessário executar uma sequência de acções elementares de baixo-nível. Por exemplo: várias acções *kick* são habitualmente necessárias para chutar a bola numa dada direcção; vários *dash* e *turn* intercalados são necessários para interceptar a bola; vários *dash, turn* e *kick* são necessários para driblar com a bola numa dada direcção. No entanto, os erros contidos na percepção e acção, o ruído que o simulador introduz na movimentação dos objectos em campo e as acções imprevisíveis dos restantes agentes, implicam a necessidade de construir planos flexíveis para realizar tais acções. Torna-se necessário efectuar replaneamentos em cada ciclo, em face de nova informação sensorial disponível.

- **Necessidade de criar acções colectivas complexas.** A complexidade da tarefa cooperativa a executar, implica a necessidade de criar em tempo-real, acções colectivas complexas tais como passes, desmarcações, jogadas colectivas (implicando a troca de bola entre vários jogadores), movimentos defensivos colectivos (implicando dois ou mais jogadores executarem acções defensivas em conjunto, tais como cortar a bola a um adversário ou marcarem um conjunto de avançados), definição de formações, etc.
- **Agentes heterogéneos.** A introdução de agentes heterogéneos em 2001 trouxe desafios adicionais para o domínio. Cada equipa pode escolher os seus jogadores de entre sete tipos gerados aleatoriamente antes do início do jogo. Os diferentes tipos de jogadores possuem capacidades distintas, baseadas num conjunto de *tradeoffs* sobre um número limitado de parâmetros. Torna-se necessário seleccionar os melhores tipos de jogadores e efectuar um mapeamento desses tipos aos agentes de acordo com as suas funções.
- **Agente treinador com visão global do mundo.** A disponibilidade de um agente privilegiado que possui uma visão global e sem erros do mundo, permite analisar as capacidades e fraquezas do oponente de uma forma muito mais segura. O treinador é uma ferramenta ideal para estudar a estratégia do adversário e as capacidades dos seus jogadores e, em face desta análise, decidir qual a táctica a utilizar pela sua equipa no decurso do jogo. O treinador é também responsável por seleccionar e substituir os jogadores heterogéneos. Tendo em conta que este agente só pode comunicar quando o jogo se encontra parado (ou a intervalos regulares, durante o jogo, mas com atraso adicionado às suas mensagens), torna-se necessário que a informação comunicada seja de alto-nível.

No capítulo nove desta tese, estes desafios serão analisados com maior detalhe e novas metodologias de coordenação adequadas ao domínio do futebol robótico simulado serão propostas.

Capítulo 9

9. Coordenação em SMA no Futebol Robótico

O Futebol Robótico em geral e a liga de simulação em particular, constituem domínios particularmente adequados à aplicação da investigação realizada em metodologias de coordenação em Sistemas Multi-Agente, nomeadamente no que diz respeito à coordenação para agentes cooperativos.

A escolha do futebol robótico como domínio principal de teste para a maioria das metodologias desenvolvidas nesta tese, prende-se com as características do ambiente proporcionado pelo simulador *soccerserver*. O ambiente é inacessível, dinâmico, não determinístico, contínuo e impõe elevadas limitações às capacidades de comunicação dos agentes. Para além disso, a definição clara de duas equipas e uma noção inicial exacta da composição das equipas e da tarefa cooperativa a realizar por cada uma, tornam este ambiente um dos domínios por excelência para a aplicação das metodologias de coordenação de agentes cooperativos desenvolvidas, nomeadamente para a aplicação com sucesso da coordenação estratégica.

O facto de existir um agente privilegiado – o treinador – com uma visão global do ambiente, mas que não pode interferir regularmente no jogo, torna este domínio ideal para a aplicação das metodologias de coordenação por controlo parcialmente hierárquico desenvolvidas.

A disponibilidade de sensores configuráveis, nomeadamente de uma câmara virtual accionada por um pescoço flexível simulado, permite a utilização de metodologias de percepção inteligente.

A baixa largura de banda disponibilizada no domínio e as restrições impostas à comunicação, tornam o futebol robótico simulado particularmente atractivo para a aplicação de metodologias de coordenação por comunicação e de metodologias de coordenação por modelização mútua. Como não é possível comunicar uma parcela significativa da informação disponível, torna-se necessário criar modelos dos colegas de equipa e raciocinar utilizando a predição das suas acções.

Grande parte da investigação descrita neste capítulo foi desenvolvida no âmbito do projecto FC Portugal. Este é um projecto de cooperação entre a Universidade do Porto[64] e a Universidade de Aveiro[65] com o objectivo de desenvolver uma equipa de futebol robótico simulado capaz de participar nas competições RoboCup. Como resultado foi desenvolvido um sistema Multi-Agente totalmente funcional – FC Portugal – que participou com muito bons resultados em diversas competições internacionais. Este capítulo descreve o SMA FC Portugal[66] e a investigação que proporcionou a sua construção.

Os tópicos principais de investigação do projecto FC Portugal incluem:

- Metodologias de coordenação e cooperação em SMA;
- Metodologias de comunicação e percepção inteligentes em SMA;
- Arquitecturas de agentes inteligentes;
- Conceito de estratégia para uma competição;
- Representação de conhecimento
- Modelização de agentes e equipas de agentes;
- Definição de um treinador para competições de equipas de agentes;
- Optimização em SMA;
- Sincronização de processos em arquitecturas distribuídas;
- Desenvolvimento de ferramentas de depuração ("*debug*") e análise de agentes.

A investigação descrita dá particular ênfase aos tópicos cuja investigação foi maioritariamente realizada pelo autor na Universidade do Porto[67], nomeadamente ao desenvolvimento de metodologias eficientes de coordenação de equipas de agentes.

---

[64] LIACC - Laboratório de Inteligência Artificial e Ciência de Computadores da Universidade do Porto – NIAD&R - Núcleo de Inteligência Artificial Distribuída e Robótica da Faculdade de Engenharia da Universidade do Porto.

[65] IEETA – Instituto de Engenharia Electrónica e Telecomunicações da Universidade de Aveiro.

[66] A descrição é centrada nos avanços de investigação realizados no âmbito do projecto, até Setembro de 2001. No entanto, alguns desenvolvimentos realizados durante o ano de 2002 são também resumidamente descritos.

[67] No entanto, contribuições de ambas as Universidades existem em quase todas as partes do projecto, sendo muito difícil separar as contribuições de cada Universidade.

9.1 Tipos de Coordenação Aplicáveis ao Domínio

O domínio do futebol robótico é, como foi discutido no capítulo anterior, um domínio extremamente complexo, dinâmico, parcialmente cooperativo e parcialmente adverso. As metodologias de coordenação a aplicar, de forma a construir boas equipas de futebol robótico simulado têm de reflectir as características do domínio.

Os tipos de coordenação aplicados ao domínio do futebol robótico no âmbito desta tese são as seguintes:

- **Coordenação por Comunicação.** Utilização de um protocolo de comunicação – ADVCOM – que permite coordenar a equipa através da partilha de informação e eventos úteis para a coordenação;
- **Coordenação por Percepção Inteligente.** Definição de um algoritmo de visão estratégica – *SLM – Strategic Looking Mechanism*;
- **Coordenação por Modelização Mútua.** Criação de modelos dos colegas de equipa e adversários e sua utilização no processo de coordenação;
- **Coordenação Estratégica.** Definição de uma organização da equipa que permite definir uma estratégia para um dado jogo composta por tácticas com condições específicas de activação. Neste contexto, foram objecto de estudo:
  - Distinção entre situações activas e estratégicas no contexto do futebol robótico;
  - Metodologias para definir situações facilmente reconhecidas por todos os agentes;
  - Definição de papéis (tipos de jogadores) especificando o comportamento individual e colectivo dos jogadores a diversos níveis;
  - Definição de um mecanismo de posicionamento estratégico baseado em situações – SBSP – *Situation Based Strategic Positioning*;
  - Metodologias para efectuar a troca dinâmica de papéis e posicionamentos baseadas em utilidades – DPRE – *Dynamic Positioning and Role Exchange*;
- **Coordenação Parcialmente Hierárquica.** Utilização do agente treinador e de metodologias de integração dinâmica de instruções nos agentes jogadores. Neste âmbito foi também desenvolvida uma linguagem – COACH UNILANG – que permite a um agente treinador comunicar a estratégia para um determinado jogo à sua equipa e alterar essa estratégia no decurso do jogo;

Embora estes mecanismos de coordenação constituam a parte mais significativa da investigação realizada no âmbito do projecto FC Portugal. É de notar que muitas outras áreas foram abordadas neste projecto, de forma a constituir uma equipa de agentes

BIBLIOTECA F.E. PORTO

totalmente funcional capaz de participar nas competições internacionais RoboCup. Grande parte dos desenvolvimentos nesta área foram influenciados pelo futebol real e definidos tendo como base o contacto com especialistas em futebol real (treinadores, jogadores e jornalistas).

9.2 Influência do Futebol no Futebol Simulado

O futebol real tem uma grande influência no futebol robótico simulado e como tal a construção de uma boa equipa de futebol robótico simulado implica a inclusão de conhecimento específico do domínio. As metodologias de coordenação e decisão individual dos jogadores foram em grande parte definidas ou ajustadas por influência de conhecimento adquirido através do contacto com profissionais de futebol real: jornalistas, treinadores e jogadores. Este contacto levou à definição de 30 medidas para criar uma equipa de futebol robótico simulado [Reis, 2000] que constituíram a base para a definição das metodologias de coordenação e decisão individual utilizadas no projecto. A tabela 19 apresenta as 30 medidas definidas no início do projecto.

Linhas de Orientação para Criar uma Equipa de Futebol Simulado	
1) Colocar-se sempre numa posição do campo útil para a equipa.	16) Permitir que a bola trabalhe para ti e não o oposto.
2) Saber sempre com precisão a posição e velocidade da bola	17) Manter uma formação da equipa compacta.
3) Quando a bola estiver bem localizada e não for provável a alteração da sua trajectória, olhar para o campo localizando os colegas de equipa e adversários relevantes para a situação.	18) Usar o espaço livre para atacar e limitá-lo ao defender.
4) Não comunicar a não ser que saiba algo de importância para os colegas.	19) Quando a posse de bola é do adversário, todos os jogadores devem pensar "defensivamente".
5) Comunicar sempre e de imediato eventos e informação importantes se achar que é um dos jogadores da equipa que é capaz de os descrever melhor.	20) Os avançados (e também os outros jogadores) devem trocar de posições sempre que tal for útil ou para confundir a defesa adversária.
6) Decidir acções prevendo o seu desenlace (imaginar possíveis cenários que advenham da sua execução).	21) Os extremos ofensivos da equipa devem permanecer abertos para atrair os defesas para longe do centro da área.
7) Em cada momento reduzir as opções possíveis dos oponentes e aumentar as opções possíveis dos colegas de equipa.	22) Desafiar (limitando-lhe o espaço de acção do oponente que tem a posse da bola rapidamente) para que não possa progredir de forma rápida no campo.
8) Para realizar diferentes acções colectivas no campo (atacar, defender, etc.) um comportamento de equipa diferente é necessário.	23) Limitar as opções do adversário, limitando-lhe o espaço e cobrindo as suas possíveis linhas de passe.
9) Não desperdiçar energia a realizar tarefas que não sejam importantes para a equipa.	24) Quanto mais próximo está a bola da baliza da equipa, mais apertada deve ser a defesa.
10) Tentar colocar-se no interior do cérebro dos colegas e adversários e tentar imaginar o que eles estão a pensar. Utilizar essa informação para decidir as acções.	25) Como defesa não fazer passes arriscados nem tentar intercepções arriscadas. Deixar o atacante tomar a iniciativa.
11) Se puder marcar, chutar! Senão, passar a alguém que possa marcar. Senão, passar a alguém que possa passar a alguém que possa marcar.	26) O que se faz sem bola é tão importante como o que se faz com bola. Definir formas de movimentação sem bola adequadas.

Senão...	
12) Não segurar a bola a não ser que não exista outra acção para fazer.	27) Utilizar jogadores com diferentes capacidades para diferentes posições no campo.
13) Não passar a bola para trás.	28) Mudar o estilo de jogo de acordo com o oponente e com o curso do jogo.
14) Sempre que possível passar a bola para jogadores que a possam receber atrás da defesa do adversário.	29) Ser flexível e adaptativo. Não utilizar regras rígidas para nada.
15) Cruzar a bola dos flancos para o centro da área sempre que possível.	30) Treinar contra adversários com capacidades e estratégias distintas e analisar em profundidade os jogos.

Tabela 19: Trinta Linhas de Orientação para criar uma equipa de futebol simulado

As duas primeiras medidas são essenciais e permitem por si só desenvolver uma equipa competitiva de futebol simulado. Aliás, baseado nestas duas medidas foi desenvolvido o código FCP Agent [Reis e Lau, 2001d] que, unicamente com 250 linhas de código (que assentam sobre o código de baixo nível do CMUnited99 [Stone et al., 1999]), permite construir uma equipa competitiva para a liga de simulação. Estas 250 linhas de código implementam um esquema de posicionamento muito simples (baseado numa simplificação do posicionamento estratégico utilizado pelo FC Portugal) que implementa a medida 1. Implementam também um esquema de comunicação muito simples (em que é comunicada sempre a posição e velocidade da bola por jogadores que a saibam) que implementa a medida 2. No entanto, para desenvolver uma equipa capaz de jogar futebol de forma semelhante à de uma equipa de futebol real, as outras 28 linhas de orientação são também necessárias.

Analisando as medidas propostas é evidente que algumas são extremamente complexas de alcançar utilizando uma equipa de agentes autónomos. Mesmo com uma equipa de futebol real é extremamente difícil treinar os seus jogadores para que cumpram com eficácia todas estas medidas. Diversas metodologias de inteligência artificial, agentes inteligentes e principalmente de Sistemas Multi-Agente foram desenvolvidas de forma a permitir construir uma equipa de agentes que siga de forma eficaz as 30 medidas sugeridas.

9.3 Arquitectura dos Agentes

9.3.1 A Arquitectura *Strategic Agent*

Para construir agentes adequados à execução de uma determinada tarefa é necessário definir uma arquitectura apropriada para esses agentes. No âmbito do futebol robótico simulado e tendo em conta a discussão apresentada em secções anteriores, relativamente às características do domínio, da equipa de agentes e da tarefa cooperativa a realizar, é necessário construir agentes capazes de balancear a reactividade com a capacidade de deliberação individual e social. Por outro lado, a inclusão de capacidades de coordenação avançadas e conhecimento específico do futebol nestes agentes, leva à necessidade de

inclusão de módulos capazes de lidar com estas metodologias de coordenação e conhecimento específico. A arquitectura dos agentes do FC Portugal designada por *Strategic Agent* – Agente Estratégico, encontra-se representada na figura 59.

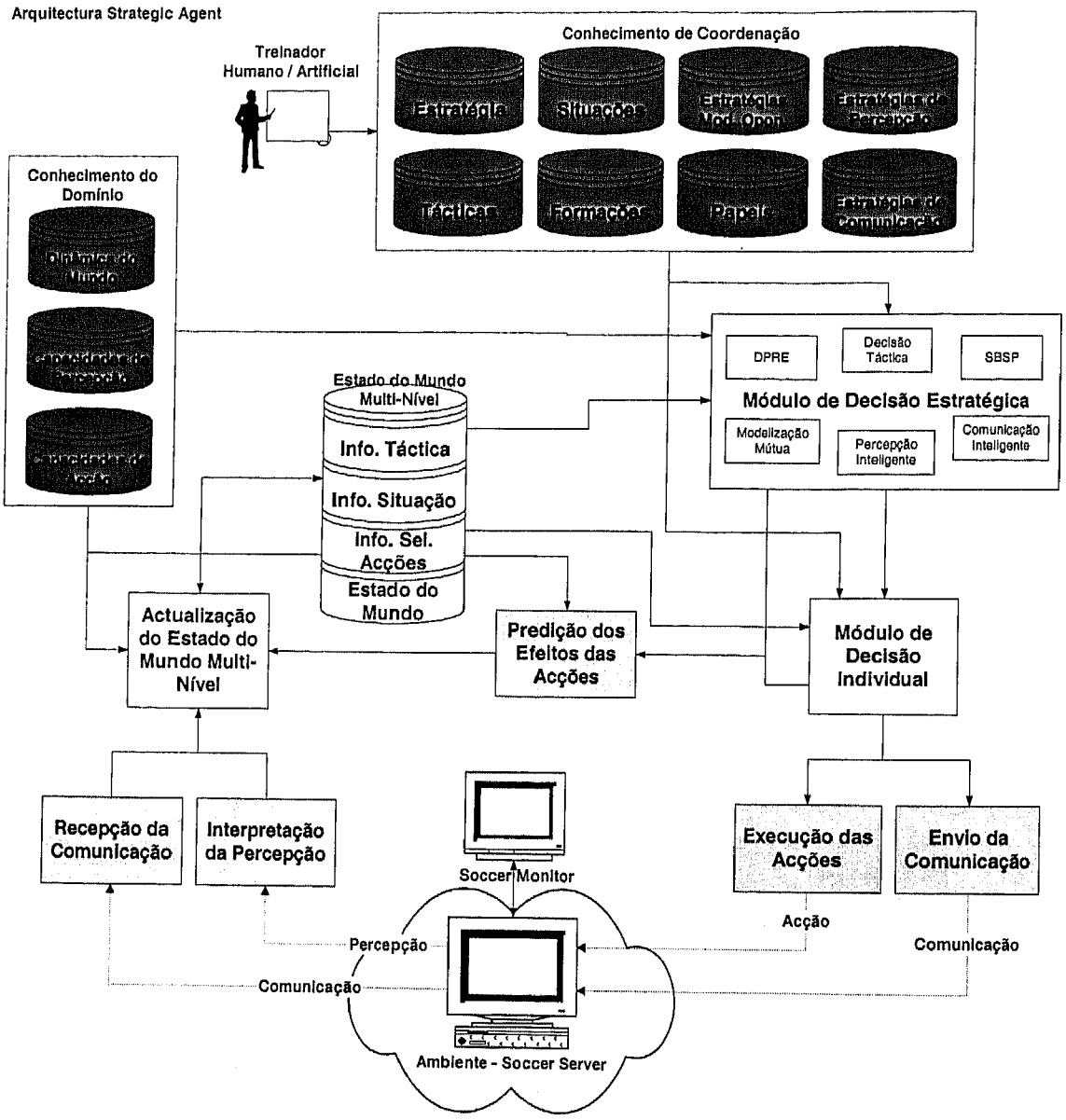

Figura 59: Arquitectura Strategic Agent dos Agentes FC Portugal

A arquitectura *Strategic Agent* centra-se na separação do conhecimento necessário em duas estruturas independentes: o conhecimento de coordenação e o conhecimento do domínio. No conhecimento do domínio é incluída a dinâmica do mundo, mais precisamente a parte desta dinâmica conhecida à partida pelo agente e as capacidades de percepção e acção do agente. No conhecimento de coordenação é incluída a estratégia a seguir, as tácticas disponíveis, as situações, formações e papéis que irão permitir utilizar a coordenação estratégica. São ainda incluídas estratégias de modelação de oponentes, de

comunicação e de percepção inteligente que irão permitir utilizar coordenação por modelização mútua, comunicação e percepção inteligente.

O estado do mundo de um agente estratégico é definido como uma estrutura multi-nível contendo desde informação de baixo-nível (incluindo as posições, velocidades e orientações dos objectos presentes no mundo) até informação de nível estratégico (incluíndo informação temporal estratégica que irá permitir a selecção de tácticas a utilizar). Este estado do mundo é actualizado utilizando a informação proveniente da percepção do agente, das comunicações recebidas, da dinâmica do mundo e da predição dos efeitos das suas acções e das acções dos outros agentes.

O módulo de decisão estratégica é responsável por seleccionar, em cada instante, a táctica a utilizar e definir qual a situação de jogo, e de acordo com estas informações, definir qual a formação a utilizar e o consequente papel do agente e posicionamento estratégico. Este módulo é também responsável pelas decisões de comunicação e utilização dos sensores configuráveis do agente (percepção inteligente). Finalmente, é ainda este módulo que realiza a predição das acções dos colegas de equipa e dos oponentes, de acordo com a informação disponível no estado do mundo e com a informação disponível no conhecimento de coordenação relativo à modelização de colegas e oponentes.

Quando um agente se encontra em situação activa, a decisão do módulo estratégico é complementada com a decisão individual. Este módulo de decisão individual parameteriza a decisão estratégica utilizando conhecimento específico do domínio e informação contida no estado do mundo de baixo nível do agente. É também responsável pelas decisões específicas do domínio que, no caso do domínio em análise, estão relacionadas, por exemplo, com a decisão de a quem passar a bola, como driblar com a bola, para onde a chutar, como interceptar a bola, entre outras.

As decisões individuais são depois transformadas em comandos a enviar para os actuadores, que no domínio em análise correspondem aos comandos a enviar para o simulador. Esta transformação envolve, entre outras, questões de sincronização com o simulador, encriptação e compressão de informação e gestão física e monitorização dos meios de comunicação disponíveis.

9.3.2 Estado do Mundo Multi-Resolução

Para que um agente se comporte de forma inteligente, é muito importante que este possua um modelo do estado do mundo que descreva o estado corrente do seu ambiente. O agente pode então utilizar este modelo para raciocinar sobre a melhor acção a executar em cada instante.

9.3.2.1 Características da Simulação

No futebol robótico simulado, os agentes recebem, no início do jogo, um conjunto de informações do servidor relacionado com a dinâmica do ambiente e com as capacidades de percepção e acção de cada um dos tipos de agente envolvidos no jogo. Esta informação corresponde aos parâmetros utilizados pelo *soccerserver* para executar a simulação (ver capítulo oito), incluindo: as dimensões do campo e das balizas, localização dos marcadores do campo (que irão permitir aos agentes realizar a sua auto-localização), características físicas e dinâmica da bola, características do terreno e vento no recinto, características da competição (número de partes, tempo máximo de jogo, ocorrência ou não de prolongamento e *penalties* em caso de empate) e características físicas dos jogadores (dimensões, momento máximo do pescoço, área de chuto, ângulo de visão, gestão da energia, etc.).

Os agentes recebem também informação relativa aos parâmetros dos jogadores heterogéneos, incluindo: intervalos de variação dos parâmetros e definições específicas das características de cada um dos tipos de jogador heterogéneo. Todas as informações descritas são armazenadas em estruturas próprias (objectos, vectores e matrizes de diversos tipos) contidas no conhecimento específico do domínio e não são alteradas no decurso do jogo. Nestas estruturas são também armazenados os dados referentes ao número do jogador, equipa em que joga (*left* ou *right*) e nome das equipas.

9.3.2.2 Estado do Mundo

Durante o jogo, os jogadores recebem informação unicamente através de mensagens, mensagens estas que contêm percepção visual, sensorial e auditiva (proveniente do árbitro, treinador ou restantes jogadores). Esta informação, em conjunto com o conhecimento específico do domínio, permite a criação e actualização da estrutura contendo o estado do mundo multi-nível.

O modelo criado para os agentes FC Portugal implementados no âmbito desta tese, inclui uma representação probabilística do estado do mundo, baseada na especificação dos parâmetros de simulação e no historial de percepções do agente. Relativamente a cada jogador das outras equipas, os agentes mantêm no seu estado do mundo as posições *(x, y)*, velocidades *(vx, vy)*, orientações dos corpos e pescoços e valores relativos à confiança nesses valores [0..1] e precisão [0..1] esperada dos mesmos (figura 60).

A confiança está directamente relacionada com o instante de tempo em que o jogador foi visualizado pela última vez (o valor do tempo incluído na mensagem com informação visual, que continha esse jogador). Se a estimativa da posição do objecto foi realizada com base em percepções recentes, a confiança será elevada. O estado do mundo dos agentes é actualizado sempre que nova informação visual é disponibilizada. A posição e velocidade dos objectos e jogadores que não se encontram no campo visual do agente é também actualizada, tendo como base o seu movimento previsto ou esperado. Para estes, a

confiança na sua posição e velocidade é diminuída em cada ciclo em que não se encontrem visíveis.

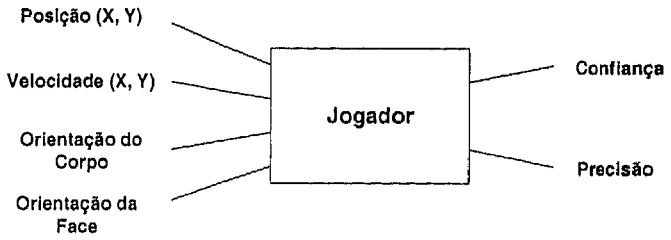

Figura 60: Informação no Nível Inferior do Estado do Mundo Relativa aos Outros Jogadores

A precisão dos valores está relacionada com a qualidade da observação dos mesmos tendo em conta o modelo do ambiente no que diz respeito aos erros de percepção. Observações de objectos que se encontram próximos do agente serão mais precisas do que observações de objectos distantes. Estimativas da posição ou orientação de objectos são também consideradas menos precisas do que observações directas desses objectos. Informação comunicada possui a precisão atribuída originalmente pelo agente que a comunicou.

As informações relativas às posições dos diversos objectos no campo são armazenadas num esquema de coordenadas absolutas *(x, y)*, como é indicado na figura 61. O centro do sistema de coordenadas corresponde ao centro do campo, com eixos *x* e *y*, ao longo do comprimento do campo (i.e. através do centro das balizas) e ao longo da linha central do campo. O sistema de coordenadas é definido de forma a que o sentido positivo do eixo dos *xx* seja no sentido atacante da equipa, independentemente de a mesma jogar da esquerda para a direita ou da direita para a esquerda. Os ângulos estão definidos no intervalo *[-180, 180]* de forma a que o ângulo *0°* se encontra na direcção positiva do eixo dos *x*.

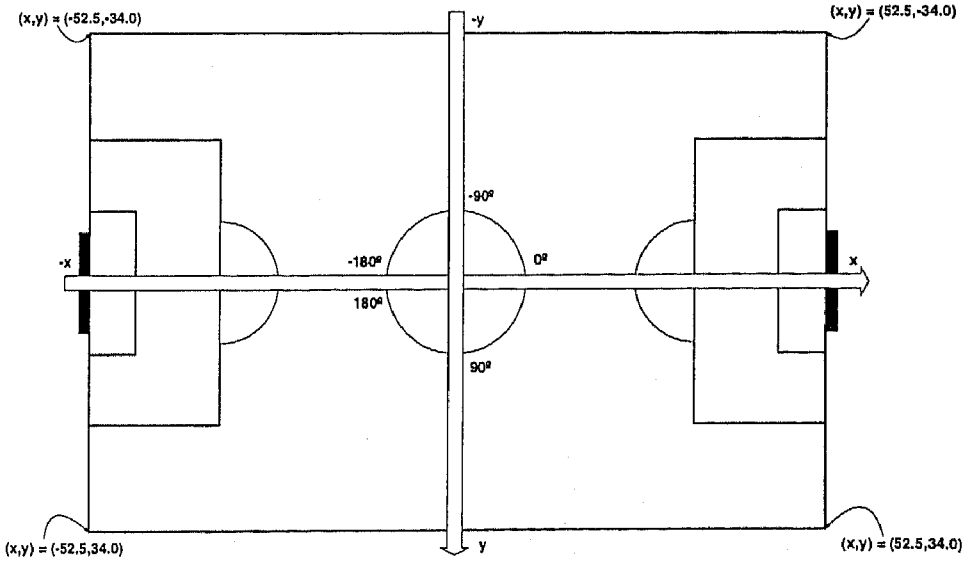

Figura 61: Coordenadas Absolutas e Ângulos do Campo de Jogo

Um aspecto importante a ter em consideração é que a informação que o agente recebe por via da sua percepção visual, é sempre relativa à perspectiva do agente no campo. Como resultado, o agente não pode determinar directamente, por observação, a sua posição absoluta no campo. A informação relativa tem de ser convertida para coordenadas absolutas no campo, mas esta conversão só será válida se a informação relativa ao próprio agente (sua posição, velocidade, orientação do corpo e pescoço) for válida e precisa.

Os agentes usam também a comunicação com o intuito de trocar informação sobre os seus estados do mundo, aumentando a precisão dos estados do mundo dos colegas de equipa. Embora em 2000 e 2001, fosse possível efectuar a troca de grandes quantidades de informação (512 bytes por mensagem), as novas regras introduzidas em 2002 reduziram a dimensão das mensagens para 10 *bytes*, não permitindo desta forma uma troca de informação completa e precisa acerca dos estados do mundo dos diferentes agentes. O protocolo de comunicação utilizado para este fim será analisado na secção 9.7.1.

A conversão da informação visual, relativa ao agente, para coordenadas absolutas é realizada com o auxílio dos diversos marcadores (bandeiras, linhas e balizas) colocados no campo e cuja posição é fixa e conhecida *a priori* (ver figura 56). Estes marcadores podem ser utilizados para permitir ao agente efectuar a sua auto-localização. Combinando as posições relativas dos marcadores visíveis, que o agente recebe como informação visual, com as suas posições absolutas conhecidas *a priori*, é possível ao agente determinar a sua própria localização *(x,y)* e orientação. O método utilizado para realizar esta tarefa foi adaptado de [Stone et al., 1999]. Consequentemente, a posição e orientação do agente pode ser utilizada de forma a determinar as posições e orientações absolutas de todos os outros agentes.

9.3.2.3 Estrutura do Estado do Mundo Multi-Resolução

Uma das características mais relevantes da arquitectura apresentada na secção anterior consiste na inclusão de um estado do mundo multi-nível (figura 62).

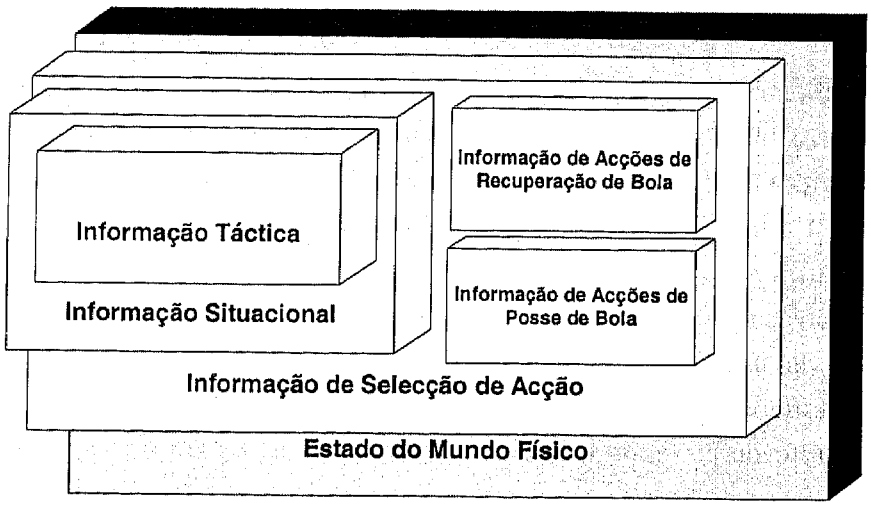

Figura 62: Estado do Mundo Multi-Nível dos Agentes FC Portugal

Como é visível na figura 62, o estado do mundo é separado em quatro níveis de abstracção [Reis et. al., 2001]:

- **Informação Táctica:** Informação de alto-nível necessária para seleccionar e configurar a melhor táctica em cada momento. Para além do tempo e resultado de jogo, inclui o comportamento global da equipa adversária (qualidade, capacidades colectivas e individuais, agressividade, etc.) e estatísticas de alto-nível do jogo (remates, assistências, posse de bola por região do campo, recuperações e perdas de bola por região do campo);
- **Informação Situacional:** Informação relevante para a selecção da situação de jogo e correspondente formação e ainda para a configuração do mecanismo de posicionamento estratégico baseado em situações. Esta informação está essencialmente relacionada com o estado actual do jogo e tem como objectivo ser suficiente para permitir realizar uma análise de alto-nível do momento actual;
- **Informação de Selecção de Acção:** Informação relevante para seleccionar uma acção apropriada, tal como distâncias à bola e possibilidades (tempos) de intercepção de bola de todos os jogadores, opções de movimentação (dos jogadores) e situações de "fora-de-jogo". Inclui ainda a informação relativa a todas as acções com bola possíveis a um dado jogador, tais como os parâmetros relativos a cada passe, chuto ou drible possíveis.
- **Estado do Mundo Físico:** Informação de baixo-nível, incluindo o estado do agente e as posições, velocidades e orientações de todos os objectos presentes no ambiente.

A informação do estado do mundo multi-nível é actualizada, em cada ciclo, partindo da informação visual, sensorial e auditiva disponível e da predição dos efeitos das acções dos outros agentes. Inicialmente, é actualizado o estado do mundo físico e, baseado neste, é actualizada a informação de selecção de acção, informação situacional e informação táctica.

Devido à extensão da informação contida no estado do mundo multi-nível (cerca de 350 itens de informação distintos) a sua descrição completa não é possível neste trabalho. A descrição detalhada do conteúdo de cada um dos níveis do estado do mundo multi-nível do FC Portugal pode ser encontrada em [Reis e Lau, 2001d].

9.3.2.4 Actualização do Estado do Mundo

A actualização do estado do mundo é realizada utilizando a informação proveniente dos vários sensores do agente. Estes incluem a audição (mensagens dos colegas de equipa incluindo informação relativa às posições, velocidades, etc., dos diversos agentes e da bola), a visão e a percepção *sense_body*. É também utilizada informação proveniente da

predição dos efeitos das acções do agente sobre o ambiente e dos efeitos das acções esperadas dos outros agentes sobre o ambiente.

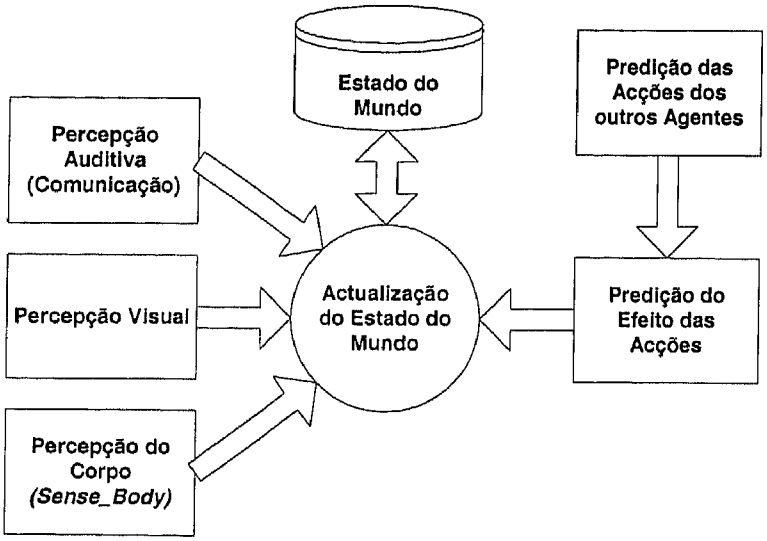

Figura 63: Actualização do Estado do Mundo

A actualização do estado do mundo multi-nível dos agentes é realizada da forma indicada no algoritmo 4. Consiste na actualização sucessiva das diversas camadas do estado do mundo multi-nível baseada na informação actualizada das camadas anteriores. Inclui ainda a criação e actualização de um histórico do estado do mundo.

```
ALGORITMO RefreshWorldState(Perceptions, MultiLevelWS, WSHistory)
    RETORNA MultiLevelWS, WSHistory
PARÂMETROS
    Perceptions - Percepção visual, auditiva e física do jogador
    MultiLevelWS - Estado do Mundo Multi-Nível do agente
    WSHistory - Historial do estado do mundo
{
// Decompõe o MultiLevelWorldState nas suas componentes
    (WorldState, ActionWorldState, SituationWorldState, TacticalWorldState) = MultiLevelWS
// Actualiza o estado do mundo de baixo-nível contido em MultiLevelWorldState
    WorldState = RefreshLowLevelWorldState(WorldState, Perceptions, WSHistory)
// Actualiza o estado do mundo de selecção de acções (com e sem bola)
    ActionWorldState = RefreshActionWorldState(ActionWorldState, WorldState, WSHistory)
// Actualiza o estado do mundo situacional
    SituationWorldState = RefreshSituationWorldState(SituationWorldState,
            ActionWorldState, WorldState, WSHistory)
// Actualiza o estado do mundo táctico
    TacticalWorldState = RefreshTacticalWorldState(TacticalWorldState,
            SituationWorldState, ActionWorldState, WorldState, WSHistory)
// Actualiza o histórico do estado do mundo contendo a informação do ciclo anterior
    WorldStateHistory = UpdateWorldStateHistory(MultiLevelWS, WSHistory)
    RETORNA MultiLevelWS, WSHistory
}
```

Algoritmo 4: Actualização do Estado do Mundo Multi-Nível

O estado do mundo de baixo nível é actualizado (algoritmo 5) baseia-se na utilização da informação sensorial e da predição dos efeitos das acções executadas pelo agente para

actualizar a camada inferior do estado do mundo (contendo as posições, velocidades e orientações de todos os objectos do campo). Inclui também uma filtragem do estado do mundo de forma a evitar variações bruscas inesperadas dos valores do mesmo.

```
ALGORITMO RefreshLowLevelWorldState(WorldState, Perceptions, WSHistory)
    RETORNA WorldState
PARÂMETROS
    Perceptions - Percepção visual, auditiva e física do jogador
    WorldState - Estado do Mundo do Agente
{
//Actualiza o estado do mundo considerando as percepções disponíveis e a predição do
efeito das acções do agente
    WorldState = RefreshLowLevelWorldState_SenseBody(WorldState, Perceptions)
    WorldState = RefreshLowLevelWorldState_Perceptions(WorldState, Perceptions)
    WorldState = RefreshLowLevelWorldState_Communications(WorldState, Perceptions)
    WorldState = RefreshLowLevelWorldState_Prediction(WorldState, Perceptions)
//Filtra o estado do mundo utilizando a informação dos ciclos anteriores
    WorldState = FilterWorldState (WorldState, WSHistory)
    RETORNA WorldState
}
```

Algoritmo 5: Actualização do Estado do Mundo de Baixo-Nível

A actualização do estado do mundo para decisão das acções a executar é realizada tendo em consideração se o agente controla a bola ou não. O algoritmo 6 descreve o método utilizado para fazer esta actualização. No anexo 5 são apresentados os algoritmos utilizados para actualizar a informação do estado do mundo quando o agente tem a posse de bola e quando não tem.

Se o agente tem o controlo de bola, a informação específica para cada acção possível de executar (remate, passe, drible, etc.) é actualizada. Se o agente não controla a bola, as informações de alto-nível relativa ao jogador e à bola, são actualizadas. Por exemplo, para a bola, esta informação de alto-nível inclui, entre outros itens: se a bola mudou de posição repentinamente, se a bola foi chutada, se a bola se desloca rapidamente, se a bola está livre no campo, se a bola foi centrada para a área, se foi chutada para a baliza ou se a bola está controlada por algum jogador.

A actualização do estado do mundo de acção prossegue com o cálculo de distâncias, tempos e pontos de intercepção de todos os jogadores à bola. Em seguida, é fundida informação de alto-nível comunicada pelos restantes elementos da equipa (por exemplo, relativa a tempos de intercepção, posse de bola, etc.). O algoritmo finaliza, através da análise de informação regional como sejam as linhas de fora de jogo e congestionamento[68] de regiões específicas do campo.

[68] Congestionamento corresponde à ocupação da região do campo por jogadores adversários. O número de jogadores e a sua proximidade à região considerada incrementam a sua congestão.

```
ALGORITMO RefreshActionWorldState(ActionWorldState, WorldState, WSHistory)
    RETORNA ActionWorldState
PARÂMETROS
    WorldState - Estado do Mundo do Agente
    ActionWorldState - Estado do Mundo para Selecção de Acções do Agente
{
// Actualiza parâmetros do estado do mundo necessários quer o agente tenha ou não a posse de bola
    ActionWorldState = RefreshCommonActionWorldState (ActionWorldState, WorldState)
// Se o agente puder controlar a bola então actualiza o estado de mundo relacionado com a posse de bola senão actualiza o estado do mundo de recuperação da bola
    SE PosseBola ENTÃO
        ActionWorldState = RefrescaBallPossessionWS (ActionWorldState, WorldState)
        SENÃO ActionWorldState = RefrescaBallRecoveryWS (ActionWorldState, WorldState)
// Aplica operações de filtragem ao estado do mundo, tendo em conta a informação histórica do mesmo de forma a reduzir a influência dos erros neste
    ActionWorldState = FilterActionWS (ActionWorldState, WorldState, WSHistory)
    RETORNA ActionWorldState
}
```

Algoritmo 6: Actualização do Estado do Mundo de Acção

A descrição completa deste algoritmo está para além do âmbito deste trabalho pois o algoritmo completo de actualização do estado do mundo da equipa FC Portugal, implementado em linguagem C++ possui cerca de 3500 linhas de código. Uma descrição mais detalhada do algoritmo de actualização do estado do mundo multi-nível pode ser encontrada em [Reis e Lau, 2001d].

9.3.2.5 Predição dos Efeitos das Acções

Os métodos de predição destinam-se a prever o estado futuro do ambiente baseado em percepções passadas. Dividem-se em sete grupos principais:

- **Métodos para prever o estado do agente após executar uma determinada acção.** Por exemplo a execução de um *dash*, implicará o agente alterar a sua velocidade e um *turn* alterar a sua orientação absoluta.
- **Métodos para prever o estado futuro da bola.** No caso da bola não ser chutada a previsão é simples bastando diminuir sucessivamente a sua velocidade e alterando a sua posição em conformidade. A execução de um *kick* terá como efeito alterar a velocidade da bola.
- **Métodos para prever o estado futuro dos jogadores.** Supondo conhecidos os comandos a executar por um dado jogador, prever o seu deslocamento no campo.
- **Métodos para prever o tempo que um agente demora a atingir uma dada posição.** Supondo que o agente irá executar a sequência óptima de comandos, prever o tempo (ciclos) que o agente demorará a atingir uma determinada posição.

- **Métodos para prever o tempo que um agente demora a interceptar a bola.** Supondo que um agente irá executar a sequência óptima de comandos, prever em quantos ciclos consegue interceptar uma bola em movimento e em que ponto do campo será efectuada essa intercepção.
- **Métodos para prever as acções executadas por um dado agente nos próximos ciclos.** Estes métodos permitem prever quais as acções que cada agente irá executar no campo e alterar a informação desse agente utilizando os métodos descritos anteriormente que prevêem o efeito dessas acções.
- **Métodos para prever o posicionamento estratégico dos agentes de uma dada equipa.** Tendo em conta as posições ocupadas pelos agentes em situações anteriores do jogo, prever a posição dos jogadores adversários em cada situação.

Estes métodos constituem a base da criação do estado do mundo de alto-nível, baseado no estado do mundo físico. A sua descrição mais detalhada pode ser encontrada em [Reis e Lau, 2001d].

9.4 Comportamentos de Baixo-Nível *(Low-Level Skills)*

A disponibilização de metodologias de coordenação poderosas, não significa por si só a criação de uma equipa poderosa de futebol robótico simulado. As percepções e acções básicas disponíveis aos agentes são de muito baixo-nível, pelo que é necessário criar não só formas de transformar a percepção num estado do mundo de alto-nível (como foi analisado na secção anterior), como também formas de transformar decisões de alto nível em comandos básicos a enviar para o servidor. Este processo designa-se vulgarmente por criação de *low-level skills* (comportamentos de baixo-nível).

No âmbito do projecto FC Portugal, foram inicialmente utilizados os *low-level skills* disponibilizados no código fonte da equipa CMUnited, disponibilizado na Internet [Stone et al., 1999]. Aliás, o FC Portugal utilizou esses mesmos *low-level skills*, sem qualquer alteração significativa, no primeiro campeonato da Europa em que participou (Amesterdão 2000). Entretanto, após ter-se verificado que estes *low-level skills* já não estavam ao nível das melhores equipas mundiais, *low-level skills* completamente diferentes foram desenvolvidos no âmbito do FC Portugal para o campeonato do mundo disputado em Melbourne em 2000. Infelizmente, com a alteração das regras da competição e do simulador, o desenvolvimento de *low-level skills* tem de ser um processo em constante execução. Devido ao ênfase da investigação do FC Portugal não ser neste tópico, actualmente a equipa não possui *low-level skills* competitivos com as melhores equipas mundiais.

Nesta secção são descritas, de forma resumida, as principais especificidades dos *low-level skills* criados no âmbito do FC Portugal. Uma descrição mais detalhada de todos os *low-level-skills* pode ser encontrada em [Reis e Lau, 2001d].

9.4.1 Movimentação Evitando Adversários

Uma das habilidades básicas consiste no movimento de um agente de um ponto inicial até um ponto destino, evitando colidir com os adversários que se encontrem no caminho. No anexo 6 é apresentado o algoritmo de movimentação, evitando colidir com adversários, utilizado no decurso normal do jogo, pelos agentes implementados no âmbito desta dissertação.

O algoritmo baseia-se em calcular a distância e ângulo para o ponto de destino. Se o ponto de destino tiver sido atingido então o agente roda para a direcção final desejada e finaliza. Se for colidir com um jogador, modifica o destino momentaneamente de forma a contorna-lo. Dependendo da distância ao destino e da situação de jogo e posição no campo calcula o erro máximo no ângulo de orientação do jogador para o ponto de destino. O erro admissível será tanto maior quanto maior for a proximidade do jogador ao ponto de destino. Se o ângulo para o ponto de destino é maior do que o erro máximo admissível então roda para o ponto de destino senão acelera para o ponto tendo cuidado com a energia.

O algoritmo descrito encontra-se bastante simplificado relativamente ao utilizado na prática pelo FC Portugal (que contém diversas heurísticas específicas do domínio), apresentando-se unicamente a estratégia de alto-nível utilizada na movimentação. Detalhes sobre a implementação prática incluindo as funções para detectar colisões com adversários, tornear adversários, calcular a direcção final desejada para o jogador, gerir a energia do jogador e ajustar de forma automática, dependendo da situação de jogo e tipo de movimento, o ângulo de erro, podem ser encontradas em [Lau e Reis, 2001d].

Para o guarda-redes foi desenvolvida uma habilidade básica de movimentação mais complexa utilizando não só movimentos para a frente mas também movimentos para trás e mantendo sempre (que possível) o contacto visual com a bola [Lau e Reis, 2001d].

9.4.2 Intercepção da Bola

A intercepção da bola é uma das habilidades críticas no futebol robótico simulado. No âmbito do projecto FC Portugal foram desenvolvidos algoritmos de intercepção segura de bola de modo a possibilitarem a elaboração de metodologias de decisão individual de mais alto-nível [Lau e Reis, 2002]. No anexo 7 é apresentada a metodologia seguida para realizar a intercepção de bola pelos agentes da equipa FC Portugal [Lau e Reis, 2001]. Este algoritmo verifica se é impossível a intercepção ou se esta foi já concluída, casos em que termina. Se a bola estiver parada então o jogador movimenta-se para o ponto onde esta

se encontra. Senão, testa diversas formas de intercepção utilizando comandos *dash* (preferencialmente) e *turn* e retorna qual o próximo comando a realizar. Detalhes sobre a implementação prática dos algoritmos de intercepção do FC Portugal podem ser encontrados em [Lau e Reis, 2001].

9.4.3 Drible Inteligente

Em 2001, a equipa chinesa Tsinghuaeolus venceu o campeonato do mundo de futebol robótico tendo como principal vantagem competitiva um drible inteligente e rápido. Isto demonstra a utilidade de desenvolver algoritmos de drible eficientes e seguros. O drible do FC Portugal embora não seja tão poderoso como o da equipa *Tsinghuaeolus*, permite que o agente, se desloque rapidamente no campo, controlando a bola com segurança e evitando adversários. O ângulo de drible relativamente ao opositor mais próximo é calculado de forma inteligente procurando sempre manter a bola fora do alcance deste. O algoritmo 7 descreve o drible inteligente implementado.

```
ALGORITMO SmartDribble(DesiredDir, ActionWorldState)
    RETORNA DribbleState - Estado actual do drible
PARÂMETROS
    DesiredDir - Direcção desejada de drible no campo
    ActionWorldState - Estado do mundo de acção
{
    DribbleAngle = CalculateDribbleSmartAngle(ActionWorldState)
    DribbleDir = CalculateDribbleSmartDir(DesiredDir, ActionWorldState)
    DribbleState = DribbleExecution(DribbleDir, DribbleAngle, ActionWorldState)
    RETORNA DribbleState
}
```

Algortimo 7: Drible Inteligente evitando Adversários

Para ser possível fintar os adversários enquanto o agente se desloca numa determinada direcção, o algoritmo de drible baseia-se em calcular a melhor posição da bola relativamente ao corpo do agente (colocando a bola o mais longe possível dos adversários de forma a que estes não possam chutar a bola) e a direcção de movimentação do agente tendo em conta o estado do mundo de acção (contornando os adversários mais próximos). Tendo em conta as direcções calculadas, o melhor comando para enviar ao simulador é seleccionado.

O cálculo do ângulo de drible é realizado construindo um vector com os melhores ângulos de drible para driblar cada um dos adversários mais próximos e atribuindo pesos a esses adversários tendo em atenção a sua proximidade e ângulo relativamente à bola.

A direcção de drible é calculada tendo em conta a direcção desejada de drible e a informação contida no estado do mundo de acção. Se não existirem oponentes nessa direcção, o agente movimenta-se com a bola na direcção desejada. No entanto, habitualmente não é este o caso. Assim, a direcção desejada é alterada de forma a contornar os adversários. O cálculo baseia-se em construir um vector com os pesos

atribuídos a cada mudança de direcção possível. Para cada adversário, que se encontre próximo do agente e aproximadamente na direcção de drible desejada, um peso é adicionado à posição do vector correspondente à melhor direcção para o evitar. No final, a melhor direcção é seleccionada efectuando uma filtragem ao vector construído.

No anexo 8 são descritos os algoritmos de cálculo do ângulo e direcção do drible. O algoritmo de execução do drible é demasiado extenso para descrever totalmente, mesmo a alto-nível. Este algoritmo recebe como parâmetros o ângulo e direcção de drible e baseia-se na decisão de um comando *dash*, *kick* ou *turn* dependendo da posição e velocidades relativas da bola, dos oponentes mais próximos, da diferença entre o ângulo de drible desejado e actual e da diferença entre a direcção de drible desejada e actual.

9.4.4 Chuto Optimizado (*"Optimization Kick"*)

A probabilidade de sucesso das acções dos jogadores depende da sua capacidade para manipular a bola correctamente, nomeadamente da sua capacidade para chutar a bola com precisão e com elevada velocidade numa dada direcção.

O modelo de chuto do *SoccerServer*, permite aos jogadores executarem sucessivos comandos *kick*, acelerando e alterando a direcção de movimento da bola, quando esta se encontra dentro de um círculo com um dado raio centrado no jogador. Esta possibilidade pode ser explorada de forma a obter, utilizando uma sequência de *kicks* correctamente calculada, uma maior velocidade de chuto da bola do que utilizando um único comando *kick*. No entanto, encontrar esta sequência de *kicks* para acelerar sucessivamente a bola, não é um problema de resolução trivial uma vez que, a eficiência de um dado chuto depende da distância e direcção relativa do jogador à bola e enormes erros estão associados à percepção, acção e movimento no *SoccerServer*. A figura 64 ilustra o problema do chuto utilizando *kicks* múltiplos, evitando adversários.

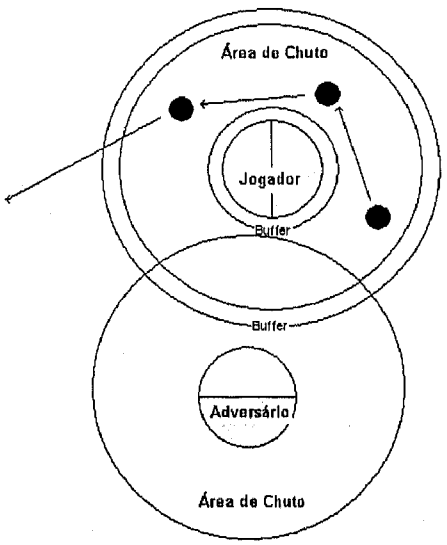

Figura 64: Chuto utilizando Vários Kicks, evitando adversários

Diversas abordagens foram utilizadas pelos investigadores da área na resolução deste problema, incluindo soluções analíticas [Stone et al., 2000a] e a utilização de aprendizagem por reforço [Riedmiller et al., 2000].

O *Optimization Kick* consiste numa metodologia baseada em técnicas de simulação e optimização, para resolver o problema do chuto e definir as acções a executar de forma a conseguir chutar a bola com uma determinada potência numa dada direcção. Tendo como base a dinâmica do *SoccerServer*, foi definida uma classe que permite simular todo o processo de chuto, prevendo a posição e velocidade da bola ao longo de uma sequência de chutos, permitindo assim utilizar algoritmos de pesquisa no sentido de encontrar uma boa solução para o problema.

As condições iniciais do problema de optimização do chuto definem as posições e velocidades iniciais do jogador e da bola e dos adversários mais próximos:

- *RelBallX, RelBallY*: Posição da bola relativamente ao jogador;
- *BallVellX, BallVelY:* Velocidade da bola relativa ao jogador;
- *PlayerVelX, PlayerVelY:* Velocidade absoluta do jogador;
- *OpponentX, OpponentY:* Posição relativa do oponente mais próximo;
- *OpponentVelX, OpponentVelY:* Velocidade relativa do oponente mais próximo;
- *OpponentRelDir:* Direcção relativa do oponente mais próximo.

O problema do chuto é definido como um problema de optimização em que a função a optimizar é:

$$f(|KickDir - KickDesDir|, |KickSpeed - KickDesSpeed|, NKicks, SafetyKickable, SafetyOpp)$$

Em que:

- *KickDir:* direcção de chuto obtida;
- *KickDesDir:* direcção de chuto desejada;
- *KickSpeed:* velocidade de chuto obtida;
- *KickDesSpeed:* velocidade desejada para o chuto;
- *NKicks:* é o comprimento (número de acções) da sequência de chutos;
- *SafetyKickable:* Probabilidade de a sequência de *Kicks* ser efectuada sem que a bola saia da área de controle do jogador nem colida com o mesmo;
- *SafetyOpp:* Probabilidade de durante a sequência de *Kicks* a bola não entrar na área de controle de um oponente.

A solução para o problema pode assumir quatro formas distintas:

- *(Pow1, Dir1):* Solução obtida com um único *kick*;
- *(Pow1, Dir1) (Pow2, Dir2):* Solução obtida com uma sequência de dois *kicks*;
- *(Pow1, Dir1) (Pow2, Dir2) (Pow3, Dir3):* Solução obtida com uma sequência de três *kicks*;
- *(Pow1, Dir1) (Pow2, Dir2) (Pow3, Dir3) (Pow4, Dir4):* Solução obtida com uma sequência de quatro *kicks*.

Embora seja possível construir soluções com um número arbitrário de passos, os resultados experimentais obtidos comprovaram que, embora não seja possível obter a solução óptima em termos de precisão na direcção e velocidade, em todos os casos, é sempre possível obter uma solução sub-óptima de muito boa qualidade, utilizando quatro *kicks* na sequência[69]. Desta forma e tendo em conta que na qualidade da solução interessa também minimizar o número de chutos, não são consideradas soluções com mais de quatro *kicks* no *Optimization Kick*.

Os parâmetros considerados no algoritmo são:

- *BufferKickable, BufferColision, BufferOpponent:* Buffers para evitar que a bola saia da área de chuto do jogador, colida com o corpo do jogador ou entre na área de chuto do oponente;
- *DirRandom, PowerRandom:* Factores de aleatoriedade na fase inicial para a direcção e para a potência;
- *PowerMin2:* Potência mínima utilizada a partir do segundo chuto;
- *AngleTol2:* Tolerância máxima no ângulo do chuto relativamente ao ângulo desejado, utilizado a partir do segundo chuto;
- *DirStep, PowerStep:* Passos utilizados na segunda fase do algoritmo para a direcção e potência;
- *MaxIterations1, MaxIterations2:* Número de iterações máximo na primeira e segunda fase;

[69] Esta conclusão era válida em 2000. Em 2001 a potência de chuto disponível aos jogadores foi aumentada (de 0.016 para 0.027) tornando possível a obtenção de soluções sub-óptimas de muito boa qualidade utilizando unicamente três *kicks* e na maioria dos casos, utilizando unicamente dois *kicks*. Esta alteração às regras limitou também consideravelmente o interesse competitivo do *Optimization Kick*.

```
ALGORITMO OptimizationKick(KickDesDir, KickDesSpeed, KickDesDist, BallRelPos, BallVel,
            PlayerVel, OppInfo)
    RETORNA KickSeq - Sequência de Kicks (Potência, Direcção) que permite chutar a bola
com a direcção e velocidade desejadas
PARÂMETROS
    KickDesDir - Direcção desejada Velocidade Desejada e distância desejada de chuto
    BallRelPos - Posição relativa da bola (x,y) em relação ao corpo do jogador
    BallVel, PlayerVel - Velocidade (x,y) da bola e jogador
    OppInfo - Informação dos oponentes mais próximos
{
//Pesquisa semi-aleatória seguida de uma pesquisa tabu da solução
    InitKickSeq = RandomSearch(KickDesDir, KickDesSpeed, KickDesDist, BallRelPos,
            BallVel, PlayerVel, OppInfo)
    KickSeq = TabuSearch(InitKickSeq, KickDesDir, KickDesSpeed, KickDesDist, BallRelPos,
            BallVel, PlayerVel, OppInfo)
    RETORNA KickSeq
}
```

Algoritmo 8: Optimization Kick

O algoritmo do *Optimization Kick* (algoritmo 8) encontra-se dividido em duas fases:

- **Primeira Fase: Pesquisa Semi-Aleatória da Solução.** Com o objectivo de encontrar uma boa sequência (ou diversas boas sequências) de chutos, *kicks* semi-aleatórios são executados até a bola sair da área de chuto do jogador ou colidir com o mesmo. Nesta altura, o último chuto é substituído por um chuto calculado analiticamente de forma a minimizar o erro da velocidade de chuto na direcção pretendida. Diversas heurísticas são utilizadas para conduzir esta fase para uma boa solução: só o primeiro chuto é completamente aleatório, sendo os seguintes chutos potentes (*PowerMin2>50*); a partir do segundo chuto, o ângulo da direcção do chuto com a direcção final é inferior a *(Angle_Tol2<45)*. Cada sequência de chuto calculada é avaliada (utilizando a função de avaliação referida) e a melhor é passada para a segunda fase do algoritmo.
- **Segunda Fase: Melhoria Iterativa da Solução.** A solução obtida na primeira fase é melhorada sucessivamente de forma a encontrar um máximo local da função de optimização. O algoritmo seleccionado para esta fase foi a Pesquisa Tabu [Glover e Laguna, 1993]. Em cada iteração, soluções vizinhas da solução anterior são obtidas através da alteração de dois dos parâmetros da sequência de chuto (direcções ou potências) sendo a melhor (não contida na lista tabu) seleccionada como nova solução. Vizinhanças na mesma direcção do último movimento efectuado são exploradas preferencialmente na iteração seguinte do algoritmo. O processo termina no final de um conjunto de iterações pré-determinado.

O algoritmo 9 descreve a primeira fase do processo. Baseia-se na execução de um conjunto de iterações calculando sequências de chuto aleatórias, substituindo o último chuto por um chuto calculado analiticamente. Os chutos calculados são avaliados, sendo o melhor seleccionado, para ser objecto de melhoria iterativa na segunda fase do processo.

```
ALGORTIMO RandomSearch(KickDesDir, KickDesSpeed, KickDesDist, BallRelPos, BallVel,
                PlayerVel, OppInfo)
    RETORNA BestKickSeq - Solução inicial: sequência de Kicks (Potencia, Direcção)
{
// Inicializa o melhor valor para um valor muito baixo
    BestValue = InitializeBestValue()
// Executa um conjunto de iterações calculando sequências de chuto aleatórias,
substituíndo o último chuto por um chuto calculado analiticamente. Avalia os chutos e
selecciona o melhor
    PARA Iteration = 1 ATÉ MaxIterations1 FAÇA
    {
        KickSeq = CreateSemiRandomKickSeq(RelBallPos, BallVel, PlayerVel)
        KickSeq = ReplaceLastKickAnaliticaly(KickSeq, KickDesDir, KickDesSpeed,KickDesDist)
        KickSeqValue = KickSeqEvaluationFunction(KickSeq, KickDesDir, KickDesSpeed,OppInfo)
        SE KickSeqValue > BestValue ENTÃO {
            BestValue = KickSeqValue
            BestKickSeq = KickSeq
            }
    }
    RETORNA BestKickSeq
}
```

Algoritmo 9: Optimization Kick – Pesquisa Semi-Aleatória

A pesquisa tabu (algoritmo 10) processa-se calculando em cada iteração uma vizinhança da melhor solução utilizando um algoritmo de vizinhança direccional preferencial. A melhor solução da vizinhança é seleccionada como a nova solução e a direcção preferencial, a lista tabu e o valor da solução actual são actualizados. O algoritmo termina ao fim de um número de iterações especificado ou se não for possível obter uma melhor solução.

```
ALGORTIMO TabuSearch(InitKickSeq, KickDesDir, KickDesSpeed, KickDesDist, BallRelPos,
                BallVel, PlayerVel, OppInfo)
    RETORNA KickSeq - Solução final: sequência de Kicks (Potência, Direcção)
PARÂMETROS
    InitKickSeq - Solução inicial a melhorar: sequência de Kicks (Potência, Direcção)
{
// KickSeq é initializado para a melhor sequência da fase anterior. A lista tabu é
inicializada para uma lista vazia. A última direcção de movimento é inicializada para um
valor nulo
    KickSeq = InitKickSeq
    TabuList = InitializeTabuList()
    LastMoveDir = InitializeLastMoveDir()
    BestValue = KickSeqEvaluationFunction(InitKickSeq, KickDesDir, KickDesSpeed, OppInfo)
// Em cada iteração é calculada uma vizinhança da solução e o melhor elemento dessa
vizinhança. Caso esse elemento seja melhor do que a melhor solução actual adiciona a
solução à lista tabu e a nova solução passa a ser a solução calculada
    PARA Iteration = 1 ATÉ MaxIterations2 FAÇA
    {
// vizinhança que altera dois parâmetros da sequência de chuto (potência ou direcção). São
consideradas preferencialmente soluções na direcção do último movimento na solução
determinada.
        Neighborhood = CalculateNeighborhood(KickSeq, LastDir)
//A nova sequência de chuto é a melhor contida na vizinhança que não esteja na lista tabu.
        NewKickSeq = CalculateBestKick(TabuList, Neighborhood)
        NewKickSeqValue = KickSeqEvaluationFunction(NewKickSeq, KickDesDir, KickDesSpeed,
                        OppInfo)
//Se a nova sequência de chuto não for melhor do que melhor solução, termina
```

```
        SE KickSeqValue < BestValue ENTÃO RETORNA KickSeq
// Senão actualiza a lista tabu, a direcção do último movimento, o melhor valor e a
solução actual
        LastDir = CalculateDir(KickSeq, NewKickSeq)
        TabuList = UpdateTabuList(TabuList, NewKickSeq)
        BestValue = NewKickSeqValue
        KickSeq = NewKickSeq
        }
    RETORNA KickSeq
}
```

Algoritmo 10: Optimization Kick – Pesquisa Tabu

A função de avaliação do *Optimization Kick* (algoritmo 11) é uma soma pesada dos erros na direcção e velocidade, tempo de execução do chuto (em número de comandos) e segurança evitando colisões e oponentes[70].

```
ALGORITMO KickSeqEvaluationFunction(KickSeq, KickDesDir, KickDesSpeed, OppInfo)
    RETORNA Value - Avaliação de uma sequência de kicks
{
//Determina os parâmetros correspondentes ao resultado da aplicação da sequência de kicks,
nomeadamente a direcção e velocidade finais, o comprimento (em número de acções da
sequência, a segurança de manter a bola na área chutável do jogador sem colidir com o
corpo do mesmo e a segurança de manter a bola fora do alcance dos adversários no decurso
da execução do kick completo.
    KickDir = GetFinalDir(KickSeq)
    KickSpeed = GetFinalSpeed(KickSeq)
    NKicks = GetLength(KickSeq)
    SafetyKickable = GetKickableSafety(KickSeq)
    SafetyOpp = GetOpponentSafety(KickSeq, OppInfo)
//Calcula os erros na distância e velocidade finais.
    ErrorDir = abs(KickDir-KickDesDir)
    ErrorSpeed = abs(KickSpeed-KickDesSpeed)
// Calcula a soma pesada de todos os factores de avaliação.
    Value = WDir * ErrorDir + WSpeed * ErrorSpeed + WKicks * (NKicks-1) +
       WSafKick * SafetyKickable + WSafOpp * SafetyOpp
    RETORNA Value
}
```

Algoritmo 11: Função de Avaliação do Optimization Kick

A figura 65 demonstra os resultados obtidos pelo algoritmo em 2000 experiências, considerando unicamente a fase de pesquisa semi-aleatória e uma função de avaliação baseada na velocidade (com velocidade objectivo de 2.7metros/ciclo) e direcção de chuto final. É visível que o valor da mediana e média da avaliação obtida nas experiências cresce monotonicamente, atingindo um valor próximo dos 2.40 (para um máximo de 2.70) perto

[70] Os valores dos parâmetros utilizados em competições oficiais em 2000 foram: WDir = 0.5; WSpeed = 10; WKicks = 3; WSafKick = 20; WSafOpp = 5. Em 2001 os valores para WKicks e WSafOpp foram aumentados respectivamente para 10 e 15 pois o aumento na potência de chuto permite obter mais facilmente a velocidade de chuto desejada e os adversários eram de maior qualidade, pelo que se tornava necessário ter mais cuidado com eles.

das 1500 iterações[71]. O valor mínimo obtido nesta fase do algoritmo é reduzido situando-se abaixo de 2.00.

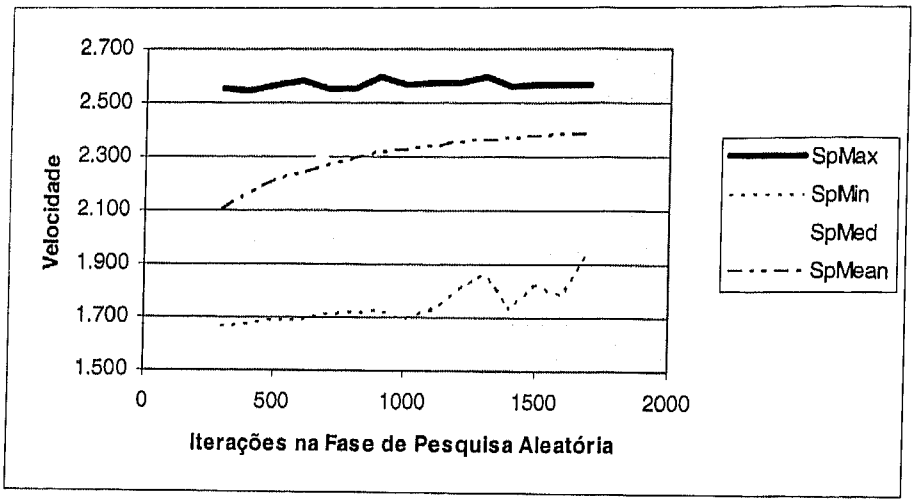

Figura 65: Resultados Obtidos na Primeira Fase do Optimization Kick em 2000 experiências (com kick_power_rate = 0.016)

A figura 66 contém os resultados obtidos através da segunda fase do algoritmo considerando unicamente 500 iterações na fase semi-aleatória e 150 iterações na fase de melhoria iterativa. É visível que o *Optimization Kick*, mesmo com um número muito reduzido de iterações, permite obter uma velocidade final com mediana de 2.52 metros/ciclo (muito próxima da velocidade máxima – 2.7). Este número de iterações corresponde a um tempo de execução inferior a 1 ms para todo o processo (utilizando um processador Pentium 700Mhz com 256Mb de memória) o que permite a utilização do método *on-line* sem problemas de desempenho.

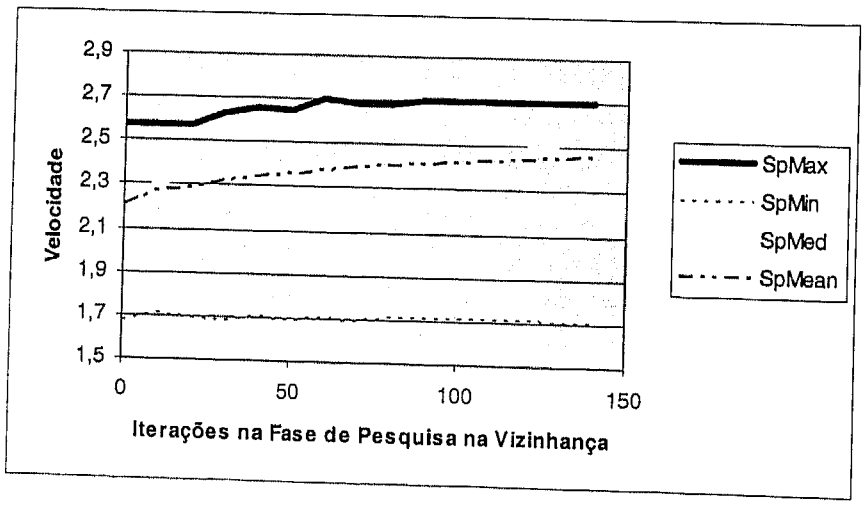

Figura 66: Resultados Obtidos na Segunda Fase do Optimization Kick em 2000 experiências (com kick_power_rate = 0.016)

[71] Com os parâmetros do servidor em 2000, i.e., *kick_power_rate = 0.016*

Diversas experiências utilizando algoritmos genéticos, arrefecimento simulado e outros métodos de optimização e sua comparação experimental com soluções analíticas do problema, foram realizadas no âmbito deste estudo. Uma descrição mais detalhada do algoritmo e das experiências referidas pode ser encontrada em [Lau e Reis Rel FCP, 2002].

9.5 Módulo de Decisão de Alto Nível

O funcionamento do módulo de decisão de alto-nível do FC Portugal encontra-se representado na figura 67.

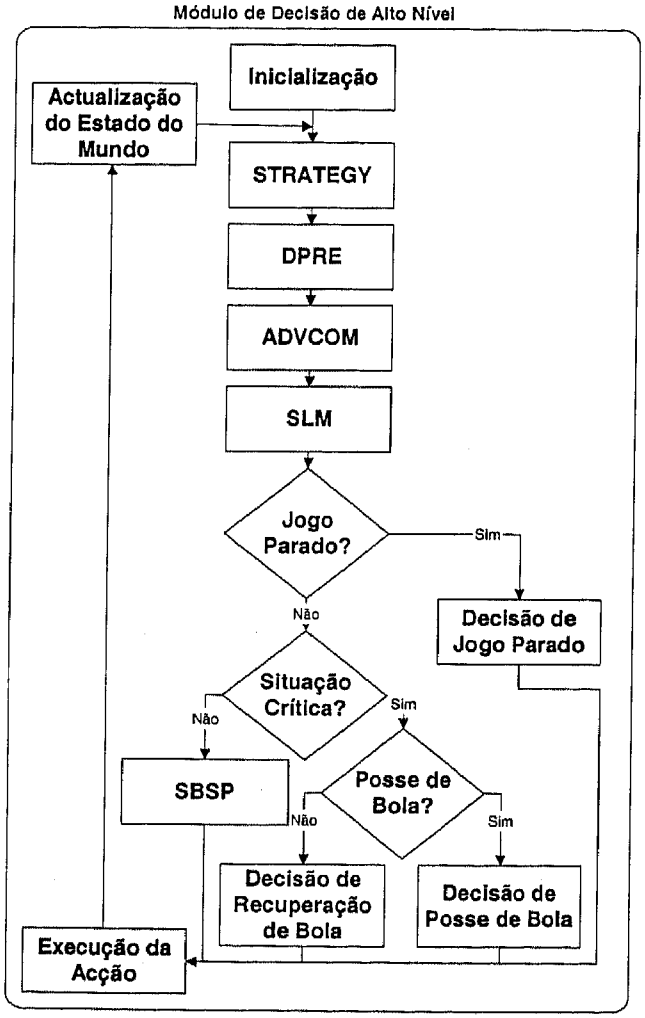

Figura 67: Fluxograma do Módulo de Decisão de Alto-Nível

O processo começa com a inicialização dos agentes, em que são executadas diversas tarefas relacionadas com a leitura dos ficheiros de configuração e inicialização do estado do mundo multi-nível dos agentes.

A fase designada STRATEGY consiste numa análise táctica do jogo. Tendo em conta a informação táctica disponível (tempo, resultado, posse de bola por regiões do campo,

acções individuais dos jogadores e seus resultados, entre outras) e a eventual disponibilidade de instruções do treinador, cada agente vai decidir qual a táctica mais apropriada a utilizar.

As instruções do treinador relativas ao comportamento individual, são fundidas com as regras de comportamento individual que o jogador possuía previamente. As instruções relativas ao comportamento colectivo (tácticas, parcelas de tácticas e formações) são consideradas como ordens e sobrepõe-se às definições previamente existentes. Nesta fase, tendo em conta o estado do mundo de alto-nível situacional e o seu conhecimento, o jogador vai também decidir qual a situação em que o jogo se encontra e decidir qual a formação a adoptar, em consonância.

A fase seguinte consiste na decisão do interesse colectivo na execução de uma troca dinâmica de posição e comportamento com um colega de equipa. Nesta fase, é analisado iterativamente, o interesse que existe em que pares de jogadores realizem uma troca de posição. O algoritmo termina quando se atinge a estabilidade, i.e. quando numa determinada iteração não exista interesse em realizar qualquer troca de posições.

A fase seguinte designa-se por ADVCOM – Comunicação Avançada e encarrega-se das decisões relacionadas com a comunicação, ou seja, quando e o quê comunicar. De acordo com a situação, com o seu conhecimento individual do mundo e com uma estimativa do conhecimento do mundo dos colegas, os agentes decidem, em cada ciclo, se comunicam ou não e qual o conteúdo de uma eventual comunicação.

Tendo em conta a situação actual e a informação contida no estado do mundo, os agentes decidem em seguida o modo de visão a utilizar (abertura e qualidade da visão) e a direcção para onde vão direccionar o seu pescoço flexível no ciclo seguinte.

Se o jogo estiver parado, por exemplo na execução de livres ou lançamentos, os agentes utilizam um módulo de decisão com jogo parado que os procura colocar nas posições correctas para o reinício do jogo. Neste módulo, é necessário considerar a eventual existência de regiões do campo inacessíveis ao agente (por exemplo no caso de um livre a favor da equipa adversária, o agente não se pode aproximar mais do que 9m da bola). De acordo com o tipo de táctica que se encontra em execução, o agente pode também optar por uma reposição muito rápida de bola (sem descansar e sem se preocupar com a finalização da colocação dos colegas) ou muito lenta (descansando ao máximo e esperando até ao limite máximo de tempo permitido para movimentar a bola).

A decisão com jogo a decorrer divide-se em duas possibilidades: situação crítica (activa) e situação estratégica (passiva). Em situação crítica, ou seja, quando o agente acha que possui o controle da bola ou pensa poder recuperá-lo rapidamente, os agentes utilizam os módulos de decisão com ou sem bola. Em situação estratégica, os agentes posicionam-se de acordo com o algoritmo *SBSP – Posicionamento Estratégico Baseado em Situações.*

A fase final consiste na definição das acções de baixo-nível a enviar para o servidor, utilizando os comportamentos descritos anteriormente. O módulo reinicia-se com a actualização do estado do mundo multi-nível utilizando a percepção visual recebida (no caso de ter sido recebida nova informação visual no ciclo), a comunicação recebida ao longo do ciclo anterior, proveniente de colegas de equipa e a predição de acções.

9.6 Coordenação Estratégica no Futebol Robótico

A coordenação estratégica foi introduzida por Reis et al. [Reis et al., 2001] e consiste na definição de uma estratégia composta por um conjunto de tácticas que implicam formas de actuação distintas de uma equipa de agentes. Em cada instante, uma única táctica encontra-se activa. Cada táctica utiliza um conjunto de formações que implicam não só distribuições espaciais distintas dos agentes mas também a atribuição de papéis distintos, definindo o comportamento de cada agente. Em cada instante, e dada a situação de jogo (análise de alto-nível do mesmo) é seleccionada uma dada formação da táctica activa nesse instante.

9.6.1 Definição Formal da Estratégia da Equipa

Na figura 68, encontra-se representada a definição de uma *Estratégia da Equipa*. Na figura é visível que a *Estratégia da Equipa* é composta por um conjunto de *Tácticas* que possuem um conjunto de *Parâmetros Tácticos* associados. Cada *Táctica* define também qual a *Formação* a utilizar em cada *Situação* de jogo. Cada *Formação* contém um conjunto de *Posicionamentos* que atribui a cada *Agente* uma movimentação dinâmica no campo e um *Papel* (tipo de jogador). O *Papel* define o comportamento de um jogador a três níveis: *Estratégico*, *Posse de Bola* e *Recuperação de Bola*.

Formalizando o conceito de *Estratégia de Equipa* dos agentes, apresentado na figura 68, esta será:

> *Estratégia_Equipa = (Tácticas, Agentes, Papéis, Formações, Situações, Est_Com, Est_Vis, Est_Mod, Planos)*

Em que *Tácticas*, consistem num conjunto de *ntácticas* aplicáveis à equipa

> $Tácticas = \{Táctica_1, Táctica_2, ..., Táctica_{ntácticas}\}$

Agentes representam os *nagentes* que constituem a equipa que irá executar a tarefa cooperativa:

> $Agentes = \{Agente_1, Agente_2, ..., Agente_{nagentes}\}$

Papéis são um conjunto de *npapéis* possíveis para os agentes definindo comportamentos individuais distintos.

$Papéis = \{Papel_1, Papel_2, ..., Papel_{npapéis}\}$

Formações são um conjunto de *nformações*, que permitem a aplicação de esquemas de posicionamento distintos aos agentes:

$Formações = \{Formação_1, Formação_2, ..., Formação_{nformações}\}$

Situações são um conjunto de *nsituações* que permite a aplicação de diferentes esquemas de posicionamento aos *nagentes*:

$Situações = \{Situação_1, Situação_2, ..., Situação_{nsituações}\}$

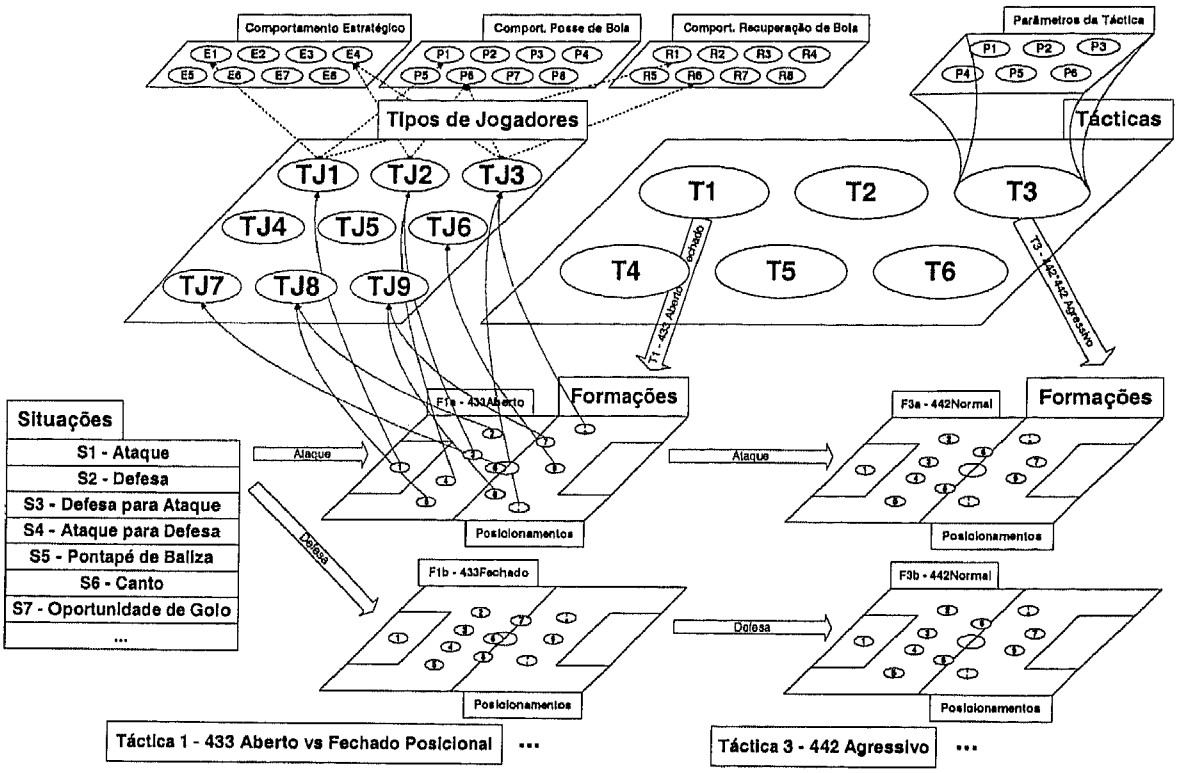

Figura 68: Diagrama da Estratégia do FC Portugal

A estratégia inclui também *Estratégias de Comunicação*, *Estratégias de Visão* e *Estratégias de Modelação de Oponentes* (e colegas de equipa) e *Planos Flexíveis* pré-concebidos:

$Est_Com = \{ECom_1, ECom_2, ..., ECom_{necom}\}$

$Est_Vis = \{EVis_1, EVis_2, ..., EVis_{nevis}\}$

$Est_Mod = \{EMod_1, EMod_2, ..., EMod_{nemod}\}$

$Planos = \{Plano_1, Plano_2, ..., Plano_{nplanos}\}$

As *Tácticas* são utilizadas para indicarem à equipa um conjunto de *Parâmetros Tácticos* globais, quais as *Formações* e *Planos Flexíveis* que devem ser utilizados em cada *Situação* de jogo (de entre as *Formações* e *Planos* pré-definidos).

$Táctica_i = (Param_Táctica_i, Form_Sit_i, Plan_Sit_i, Reg_Act_Táctica_i) \ \forall i = 1..ntácticas$

Cada *Plano Flexível* contém *Regras de Activação*, *Regras de Terminação* e a Evolução dos *Posicionamentos* e *Papéis* dos *Agentes* participantes ao longo do tempo. Contém ainda as *Acções Preferenciais* a executar por cada jogador ao longo da sua execução[72].

As *Tácticas* são seleccionadas de acordo com um conjunto de *Regras de Activação Tácticas (Reg_Act_Táctica)* que utiliza a *Informação Táctica* disponível na camada superior do estado do mundo multi-nível. Por exemplo, quando a equipa está a perder próximo do final de um jogo, mas se estiver a ter domínio do jogo (em termos de posse de bola e oportunidades para marcar golos) uma *Táctica* mais atacante, arriscada e agressiva poderá ser seleccionada. Se for claro das estatísticas de jogo e informação de modelização do oponente (disponíveis na informação táctica do estado do mundo multi-nível), que a equipa adversária está a dominar totalmente o jogo, então uma *Táctica* mais defensiva, utilizando formações mais recuadas, que tente minimizar a derrota, pode ser seleccionada.

As *Formações* fornecem aos jogadores informação relativa ao seu *Posicionamento Estratégico* dinâmico e informação relativa ao seu *Papel* individual (características relacionadas com o comportamento individual).

Uma *Formação* é definida por *nagentes Posicionamentos* cada qual definindo o *Posicionamento Estratégico* de cada *Agente* e respectivos *Papel* e *Importância* para a equipa.

$$Formação_i = (Posicionamento_{i,1}, Posicionamento_{i,2}, \ldots Posicionamento_{i,nAgentes})$$
$$\forall i = 1..nformações$$

Para cada *Táctica*, em cada *Situação* é utilizada uma das *Formações* definidas:

$$Sit_Form_i = (FormAtrib_{i,1}, FormAtrib_{i,2}, FormAtrib_{i,3}, \ldots FormAtrib_{i,nsituações})$$
$$\forall i = 1..ntácticas$$

As *Formações* utilizadas em cada *Situação* para cada *Táctica* devem estar definidas na *Estratégia da Equipa*:

$$FormAtrib_{i,s} \in \{1..nformações\} \quad \forall i = 1..ntácticas \quad \forall s = 1..nsituações$$

O *Posicionamento* de um dado *Agente* numa *Formação* é definido pela sua *Posição Estratégica de Referência*, pelo seu *Papel* e pela sua *Importância:*

$$Posicionamento_{i,p} = (Posiçãoreferência_{i,p}, PapelAtribuido_{i,p}, ImportânciaPosicionamento_{i,p})$$
$$\forall i = 1..nformações \quad \forall p = 1..nplayers_i$$

Para cada *Posicionamento* numa dada *Formação*, é definida uma *Posição Estratégica de Referência*:

---

[72] Neste trabalho não é abordada a definição dos planos flexíveis e sua utilização prática no domínio do futebol robótico simulado pois esse trabalho de investigação foi realizado em data posterior ao início da escrita desta dissertação.

$$Posiçãoreferência_{i,p} = (PosiçãoreferênciaX_{i,p}, PosiçãoreferênciaY_{i,p})$$
$$PosiçãoreferênciaX_{i,p} \in \{-field_length..field_length\}$$
$$PosiçãoreferênciaY_{i,p} \in \{-field_width..field_width\}$$

A *Posição Estratégica de Referência* é ajustada tendo em conta a informação correspondente à movimentação estratégica da *Formação* e à movimentação estratégica do *Papel* assumido pelo *Agente*.

Cada *Posicionamento* tem associada uma dada *Importância*. Por exemplo, no domínio do futebol robótico o *Posicionamento* do guarda-redes, defesa central e avançado centro são muito importantes, nunca devendo estar desocupados. A *Importância* de cada *Posicionamento* pode ser definida numa escala qualitativa:

$$ImportânciaPosicionamento_{i,p} \in \{VeryLow, Low, Medium, High, VeryHigh\}$$

Cada *Papel* define as características de um determinado *Agente*. Todos os *Papéis* utilizados em cada uma das *Formações* devem estar definidos na *Estratégia da Equipa*:

$$PapelAtribuido_{i,p} \in \{1..npapeis\} \quad \forall i = 1..nformações \quad \forall p = 1..nagentes_i$$

Os *Papéis* definidos na *Estratégia da Equipa* atribuem a cada *Agente* características a dois níveis distintos: *Estratégicas* e *Activas*. No domínio do futebol robótico simulado as *Características Activas* incluem dois tipos de características: *Posse de Bola* e *Recuperação de Bola*. Cada *Papel* inclui ainda um conjunto de *Regras de Situação Crítica* indicando as condições para o agente abandonar o *Comportamento Estratégico* e utilizar um *Comportamento Activo*.

$$Papel_i = (CaracteristicasActivas_i, CaracteristicasEstrategicas_i, RegrasSituaçãoCrítica_i)$$
$$\forall i = 1..npapeis$$

As características de *Posse de Bola* e *Recuperação de Bola*, contrariamente aos restantes conceitos apresentados na definição da *Estratégia da Equipa*, são dependentes do domínio e devem ser definidas de forma adequada a este.

$$CaracteristicasActivas_i = CaracteristicasPosseBola_i, Características\ RecuperaçãoBola_i,$$
$$RegrasPosseBola_i)$$
$$\forall i = 1..npapeis$$

Os *Papéis* podem ser utilizados para definir *Agentes* com diferentes comportamentos tais como um defesa_agressivo, defesa_posicional, extremo_driblador, medio_distribuidor, etc.

Uma descrição mais detalhada da definição de características activas e estratégicas, independentes do domínio, para os papéis definidos, pode ser encontrada em [Reis et al, 2001]

9.6.2 Definição de Situações

A aplicação da coordenação estratégica implica que os *Agentes* possuam a definição de um conjunto de *Situações* facilmente identificáveis e cuja identificação temporal não sofra variações consideráveis de *Agente* para *Agente*. Cada *Situação* é definida em termos de condições lógicas para os parâmetros respectivos no estado do mundo de alto-nível. No âmbito da aplicação da coordenação estratégica ao futebol robótico simulado, definiram-se vinte *Situações*. No entanto, a utilização da coordenação estratégica não implica a necessidade de definição de *Formações* para todas estas *Situações*. Cada *Situação* possui uma série de *Situações Substitutas* que a substituem no caso de não se encontrar uma *Formação* definida para ela. As *Situações* definidas e respectivas *Situações Substitutas* consideradas encontram-se representadas na tabela 20.

Situação	*Sit. Substituta 1*	*Sit. Substituta 2*	*Sit. Substituta 3*
Defesa			
Ataque	*Defesa*		
Passagem Defesa-Ataque	*Defesa*		
Passagem Ataque-Defesa	*Ataque*	*Defesa*	
Ataque Perigoso (Equipa)	*Ataque/Defesa*		
Livre (Equipa)	*Ataque/Defesa*		
Reposição de Bola (Equipa)	*Livre(Equipa)*	*Ataque/Defesa*	
Pontapé de Baliza (Equipa)	*Reposição de Bola (Equipa)*	*Livre(Equipa)*	*Ataque/Defesa*
Pontapé de Início (Equipa)	*Livre(Equipa)*	*Ataque/Defesa*	
Lançamento (Equipa)	*Livre(Equipa)*	*Ataque/Defesa*	
Canto (Equipa)	*Lançamento (Equipa)*	*Livre(Equipa)*	*Ataque/Defesa*
Penalty (Equipa)	*Livre (Equipa)*	*Ataque/Defesa*	

Tabela 20: Situações utilizadas e Situações substitutas

Embora o conceito seja generalizável a outros domínios, a forma de definição das *Situações* é específica de cada domínio. Por exemplo, no domínio do futebol robótico, uma *Situação* de ataque poderia ser definida (utilizando informação contida no estado do mundo situacional) como:

AtackSituation =

(PlayMode = Play_On E BallPossession(Our) E TimeBallPossession > TimeThres E

TheirMiddleField(BallPos) E Não(TheirDangerAttackPos(BallPos)))

Uma situação de ataque existe quando o modo de jogo indica que o jogo não está interrompido, que a posse de bola ser da nossa equipa (o jogador mais rápido a interceptar a bola é de equipa e o último chuto na bola ter sido executado por um jogador da equipa) durante um limiar de tempo pré-especificado, que a bola está localizada no meio campo do adversário mas não numa região perigosa.

A definição apresentada é muito simplificada relativamente às definições utilizadas em competições oficiais pela equipa FC Portugal para as várias *Situações* de jogo apresentadas. Em virtude de estas definições serem totalmente dependentes do domínio, não são descritas em detalhe neste trabalho. Uma análise mais detalhada das *Situações* utilizadas na equipa FC Portugal e sua definição, encontra-se em [Reis e Lau, 2001c].

9.6.3 SBSP – Posicionamento Estratégico Baseado em Situações

O *Posicionamento Estratégico Baseado em Situações* baseia-se na definição de *Formações* dinâmicas que estabelecem em cada instante uma *Posição Estratégica* para cada *Agente* que compõe a equipa. Em cada instante a *Formação* correspondente à *Situação* actual de jogo é activada, e tendo em conta parâmetros específicos do domínio a sua dinâmica é definida pelo algoritmo 12, que dada a táctica e situação actuais, determina a posição estratégica para um agente que ocupa um dado posicionamento na formação.

O algoritmo de *Posicionamento Estratégico Baseado em Situações* calcula, tendo em conta a *Táctica* e *Situação* actuais, a *Posição Estratégica (x, y)* no campo para um dado *Agente* que ocupa um dado *Posicionamento* na *Formação*. Começa por determinar a *Formação* e respectivos parâmetros utilizados na *Táctica* e *Situação* activa. Em seguida determina a *Posição de Referência (x, y)* para o *Posicionamento* que o *Agente* ocupa actualmente na *Formação* (tendo em conta as trocas dinâmicas de posicionamentos efectuadas e os parâmetros da *Formação* actual).

No caso de existir informação dinâmica específica para o *Papel Estratégico* atribuído ao *Agente*, esta informação é utilizada. Caso contrário, a informação dinâmica da *Formação* é utilizada por defeito. Esta informação dinâmica inclui a *Atracção pela Bola*, *Atracções e Repulsões por Regiões* específicas do campo ou objectos, quando a bola se encontra em regiões pré-especificadas, e *Limites de Movimentação* para cada jogador.

A *Posição Estratégica Base* do *Agente* é alterada considerando a *Atracção pela Bola* definida para o seu *Papel Estratégico* ou *Formação*. A *Posição da Bola* considerada para este efeito corresponde à posição prevista da próxima intercepção da bola por um dado *Agente*.

A *Posição Estratégica Base* calculada é ajustada posteriormente considerando as *Atracções e Repulsões por Regiões* ou objectos pré-especificados do campo e *Limites de Movimentação* para o *Papel Estratégico* ou *Formação*. Isto permite uma fácil adaptação do algoritmo de posicionamento estratégico a outros domínios.

Tendo em conta informação específica do domínio, a *Posição Estratégica* é ajustada para uma posição válida. No caso do futebol robótico simulado, as posições inválidas incluem, entre outras: posições de fora de jogo, posições no interior da área adversária no instante em que ocorrem os pontapés de baliza, posições no meio campo adversário antes do início

do jogo, posições a menos de nove metros de distância do local da marcação de livres. Informação específica do domínio pode também ser utilizada, numa fase final, para ajustar a *Posição Estratégica* para uma posição mais útil.

```
ALGORITMO SituationBasedStrategicPositioning(Tactic, Situation, Player)
RETORNA Position - Vector contendo a Posição Estratégica do Jogador Player na presente
                   Táctica e Formação
PARÂMETROS
    Tactic - Táctica activa da estratégia da equipa
    Situation - Situação activa
    Player - Número do jogador cujo posicionamento estratégico vai ser calculado
{
// Dada a Táctica e Situação actuais determina qual a Formação a utilizar e os parâmetros
relativos ao seu comprimento (eixo dos xx) e largura (eixo dos yy)
    Formation = SituationFormation(Tactic, Situation)
    FormHeight = SituationFormationHeight(Tactic, Situation)
    FormWidth = SituationFormationWidth(Tactic, Situation)
    Positioning = PlayerGetPositioning(Player)
// Determina a Posição de referência para o posicionamento que o agente ocupa actualmente
(é necessário ter em conta que os agentes podem trocar dinamicamente de posicionamentos)
    Position = HomeSBSPPosition(Formation, FormWidth, FormHeight, Positioning)
// Determina qual o papel atribuído ao jogador que ocupa o posicionamento em causa
e qual o papel estratégico correspondente a esse papel
    Role = FormationGetRole(Formation, Positioning)
    RoleStrategic = RoleGetStrategicRole(Role)
// Calcula posição da bola a utilizar no algoritmo de posicionamento estratégico. A
posição utilizada é a próxima posição prevista de intercepção da bola por um qualquer
jogador, ajustada para o interior do campo
    BallAdjPos = AdjustedBallPosition(BallPosition)
// Utilizando a posição básica e a informação relativa à dinâmica da formação e ao papel
estratégico do jogador, calcula a posição estratégica do jogador
    Position = BasicSBSPPosition(Position, Formation, RoleStrategic, Positioning,
               BallAdjPos)
// Utilizando a posição estratégica, e informação definida na formação e papel estratégico
do jogador relativa a atracções e repulsões de regiões do campo, calcula a nova posição
estratégica do jogador
    Position = RegionalAdjustSBSPPosition(Position, Formation, RoleStrategic, Positioning,
               BallAdjPos)
//Ajusta a posição SBSP para uma posição legal (i.e. admissível tendo em conta informação
específica do domínio). Por exemplo, a posição não deve estar em "fora de jogo", antes do
jogo começar a posição tem de estar no meio campo defensivo, na marcação de livres tem de
estar a 9m de distância, na marcação de pontapés de baliza tem de estar fora da área
    Position = LegalAdjustSBSPPosition(Position, BallAdjPos)
//Ajusta a posição SBSP para uma posição mais útil, tendo em conta informação específica
do domínio, por exemplo garantindo distâncias mínimas entre jogadores, ficando atrás da
linha da bola em determinadas situações, etc.
    Position = DomainAdjustSBSPPosition(Position, BallAdjPos)
    RETORNA Position
}
```

Algoritmo 12: Posicionamento Estratégico Baseado em Situações

O algoritmo 13 calcula a posição base SBSP do jogador, de acordo com a formação que se encontra em utilização e respectiva largura e altura.

```
ALGORITMO HomeSBSPPosition(Formation, Width, Height, Positioning)
RETORNA Position - Vector com a posição estratégica base de um posicionamento da formação
PARÂMETROS
    Formation - Formação activa da estratégia da equipa
    Width, Height - Largura (eixo dos yy) e comprimento (eixo dos xx) da formação
    Positioning - Posicionamento na formação cuja posição estratégica vai ser calculada
{
// Na informação relativa à formação actual retira a posição base do jogador que ocupa o
posicionamento Positioning na formação.
    Position = FormationGetHomePosition(Formation, Positioning)
//Ajusta a posição base do jogador tendo em conta a largura e comprimento da formação
(factores multiplicativos)
    Position.x = Position.x * Height
    Position.y = Position.y * Width
    RETORNA Position
}
```

Algoritmo 13: Cálculo da Posição Estratégica Base

O algoritmo de posicionamento estratégico básico (algoritmo 14), ajusta a posição base SBSP de acordo com a dinâmica da formação (ou do jogador individualmente) tendo em conta a posição da bola.

```
ALGORITMO BasicSBSPPosition(Position, Formation, RoleStrategic, Positioning, BallAdjPos)
    RETORNA Position - Vector com a posição estratégica dinâmica ajustada para a posição
            da bola
PARÂMETROS
    Position - Posição estratégica base do jogador
    Formation - Formação activa da estratégia da equipa
    RoleStrategic - Papel estratégico do jogador que ocupa Positioning
    Positioning - Posicionamento na formação cuja posição estratégica vai ser calculada
    BallAdjPos - Posição ajustada da bola relativa à qual Position vai ser calculada
{
// Se o Papel do jogador tiver definida a atracção pela posição da bola então utiliza essa
informação, senão utiliza a informação genérica da formação
    SE (RoleGetBallAttraction(RoleStrategic) != NULL) ENTÃO
       Attraction = RoleGetBallAttraction(RoleStrategic)
    SENÃO Attraction = FormationGetBallAttraction(Formation, Positioning)
// Ajusta a posição de acordo com a Posição ajustada da Bola de forma a deslocar a
formação coordenadamente para a zona onde a bola se encontra(rá)
    Position.x = Position.x + BallAdjPos.x * Attraction.x
    Position.y = Position.y + BallAdjPos.y * Attraction.y
    RETORNA Position
}
```

Algoritmo 14: Posicionamento Estratégico Dinâmico

O algoritmo 15 aplica atracções e repulsões por pontos e limita a posição estratégica à região admissível considerada no papel ou formação. Embora o algoritmo seja genérico, as regiões consideradas como pontos de atracção podem por exemplo ser as marcas de *penalty* (no caso de a bola estar próxima da área) ou pontos de chuto pré-definidos. Possíveis regiões consideradas como pontos de repulsão podem ser as linhas de limite do campo (em qualquer caso).

```
ALGORITMO RegionalAdjustSBSPPosition(Position, Formation, RoleStrategic, Positioning,
        BallAdjPos)
RETORNA Position - Vector com a posição estratégica dinâmica ajustada considerando
                atracções e repulsões regionais
PARÂMETROS
    Position - Posição estratégica base do jogador
    Formation - Formação activa da estratégia da equipa
    RoleStrategic - Papel estratégico do jogador que ocupa Positioning
    Positioning - Posicionamento na formação cuja posição estratégica vai ser calculada
    BallAdjPos - Posição ajustada da bola relativa à qual Position vai ser calculada
{
// Para cada região considerada como possível conducente a atracções/repulsões por pontos
    PARA Region = 1 ATÉ RegionMax FAÇA
// Se a posição ajustada da bola estiver na região considerada então
        Se InRegion(BallAdjPos, Region) ENTÃO
        {
//Se existir informação de atracção/repulsão no papel estratégico, é utilizada essa
informação, senão é utilizada a informação geral da formação. Valores negativos de
atracção correspondem a repulsões. É também determinada a região de atracção (que pode ser
um ponto, linha, círculo ou rectângulo.
            SE (RoleGetRegionAttraction(RoleStrategic) != NULL) ENTÃO
            {
                RegionAttraction = RoleGetRegionAttraction(Region, RoleStrategic)
                RegionAttractionLoc = RoleGetRegionAttractionLoc(Region, RoleStrategic)
            }
            SENÃO
            {
                RegionAttraction = FormationGetRegionAttraction(Region, Formation,
                        Positioning)
                RegionAttractionLoc = FormationGetRegionAttractionLoc(Region, Formation,
                        Positioning)
            }
//Ajusta a posição estratégica do jogador considerando o valor da atracção e a região para
onde a posição vai ser atraída/repelida
            Position = ApplyRegionAttraction(Position, RegionAttraction,
                        RegionAttractionLoc)
        }
//Garante que a posição estratégica não sai dos limites considerados para o papel
estratégico do jogador ou formação
    SE (RoleGetAdmissibleRegion(RoleStrategic) != NULL) ENTÃO
        RegionAdmissible = RoleGetAdmissibleRegion(RoleStrategic)
        SENÃO RegionAdmissible = FormationGetAdmissibleRegion(Formation, Positioning)
    Position = LimitRegionToRegionAdmissible(Position, RegionAdmissible)
    RETORNA Position
}
```

Algoritmo 15: Posicionamento Estratégico Baseado em Situações

Na figura 69, é apresentado um exemplo demonstrando posicionamentos calculados pelo algoritmo SBSP para diversas situações de jogo de ataque e defesa.

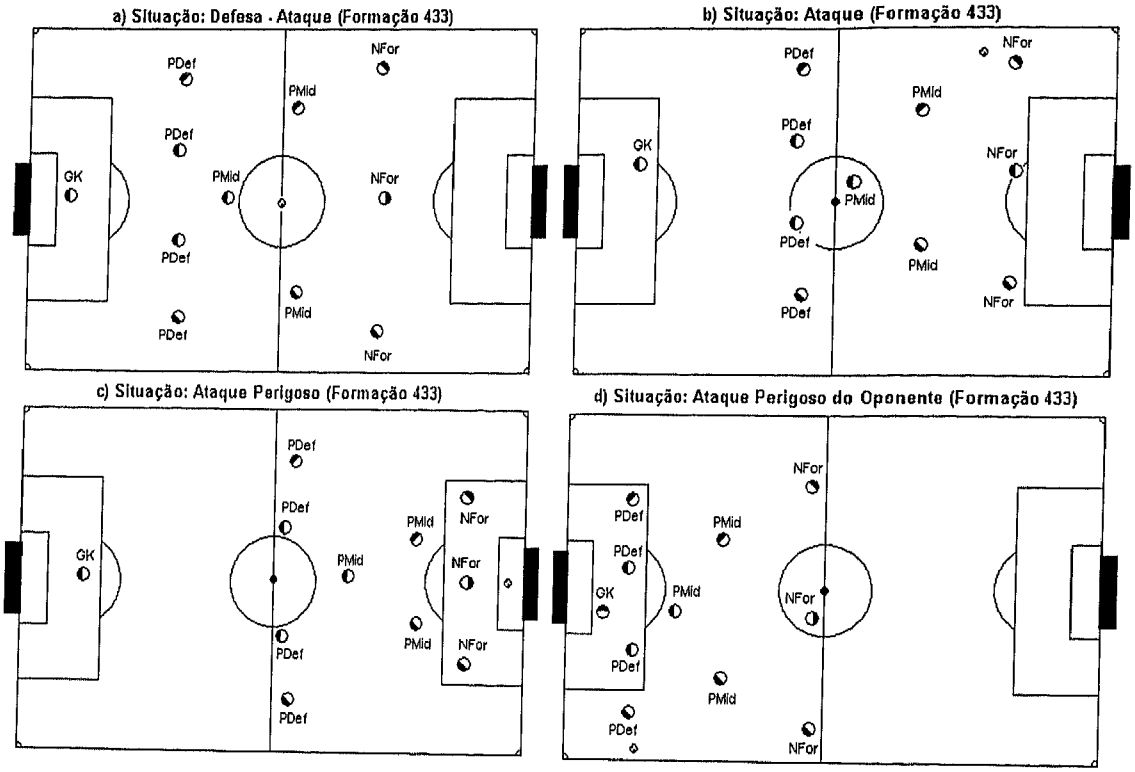

Figura 69: Exemplos de Aplicação do Algoritmo SBSP em Diversas Situações

Na figura 70 é visível a diferença do posicionamento utilizando o SBSP e o algoritmo utilizado pelos campeões de futebol robótico simulado em 1998 e 1999: SPAR [Stone et al., 2000a].

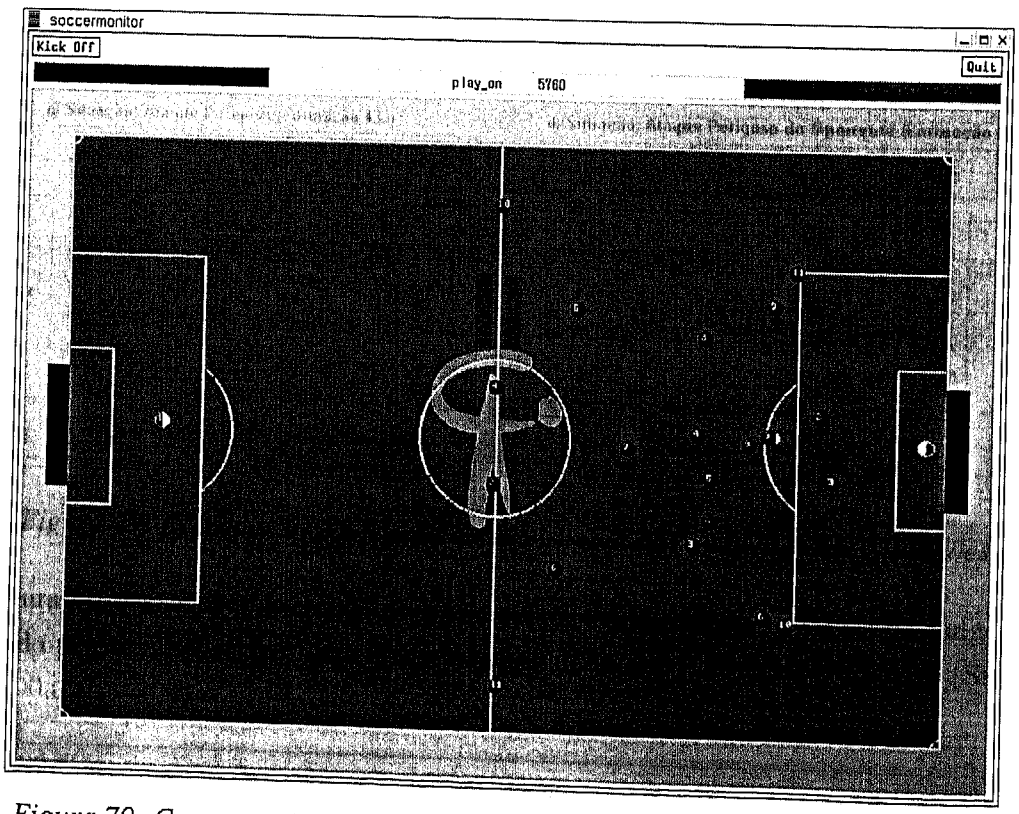

Figura 70: Comparação entre o SBSP e o algoritmo de posicionamento SPAR

O SPAR é um posicionamento activo baseado em atracções por adversários e repulsões por colegas de equipa. Na imagem são visíveis as vantagens de um posicionamento estratégico relativamente a um posicionamento activo. Enquanto os jogadores da equipa FC Portugal se encontram bem distribuídos no campo, numa formação 433, os jogadores da equipa adversária, utilizando o algoritmo SPAR não conseguem compor uma formação. Aliás, é também visível que o jogador da equipa FC Portugal que tem a posse de bola (avançado centro) possui não só espaço para avançar no terreno e rematar à baliza como também linhas de passe livres para os outros dois avançados da equipa.

Na secção 9.13.2, a influência do SBSP no comportamento global da equipa, nomeadamente nos seus resultados, será objecto de uma análise mais detalhada.

9.6.4 DPRE – Troca Dinâmica de Papéis e Posicionamentos

A coordenação estratégica baseia-se na definição de um conjunto de *Tácticas* compostas por *Formações* que atribuem a cada jogador, em cada instante, um *Posicionamento* (posição estratégica dinâmica) e um *Comportamento* distinto. Isto permite efectuar também dinamicamente a troca da alocação de cada posicionamento aos elementos que compõem a equipa, trocando também o respectivo comportamento.

A *Troca Dinâmica de Papéis e Posicionamentos (DPRE – Dynamic Positioning and Role Exchange)* utiliza o cálculo de uma função de utilidade para avaliar, em cada instante, a utilidade de efectuar a troca de posicionamentos entre agentes da equipa.

Como parte do seu estado do mundo, cada agente possui uma estimativa da posição de todos os outros agentes no campo:

$$PosiçõesAgentes = \{ PosAgente_1, PosAgent_2, ..., PosAgent_{nagentes} \}$$

$$PosAgente_p = (PosAgenteX_p, PosAgenteY_p)$$

$$\forall p = 1..nagentes$$

Em cada instante do jogo, cada agente possui um dado posicionamento e um papel (características de comportamento) na formação utilizada para a situação actual. Cada agente conhece não só o seu próprio posicionamento mas também a informação de quais os posicionamentos estimados para os restantes membros da equipa:

$$PosicionamentosAlocados = \{ PosicionAloc_1, PosicionAloc_2, ..., PosicionAloc_{nagentes} \}$$

$$PosicionAloc_p \in \{1..nagentes\}$$

Se a equipa for composta por agente homogéneos, cada agente pode desempenhar o papel de qualquer outro agente e ocupar qualquer posicionamento na formação. Tendo em conta que a equipa é composta por agentes cooperativos, a troca dinâmica de posicionamentos e papéis deverá ocorrer sempre que tal possa aumentar a utilidade global da equipa. Esta utilidade é calculada utilizando a seguinte fórmula:

$$UtilidadeDPRE(form, p_a, p_b) = UtilidadeChange(form, p_a, p_b) - UtilidadeKeep(form, p_a, p_b)$$
$$\forall p_a = 1..nagentes_{form}, \forall p_b = 1..nagentes_{form}, p_a < p_b$$

ou seja, a utilidade DPRE é definida como a diferença entre as utilidades de trocar e manter os posicionamentos. Estas utilidades são calculadas tendo em consideração as distâncias das posições actuais dos agentes às suas posições estratégicas, adequações dos agentes a cada papel e importância dos posicionamentos (algoritmo 16).

Se a utilidade para a troca de posicionamentos e papéis para um dado par (p_a,p_b) for positiva, então a troca dinâmica de posicionamentos e papéis tem lugar e cada agente assume o posicionamento e papel do outro agente.

$$\begin{cases} PositionAloc_{form,p_a} = PositionAloc_{form,p_b} \\ PositionAloc_{form,p_b} = PositionAloc_{form,p_a} \end{cases}$$

O algoritmo 16 demonstra a aplicação do DPRE ao domínio do futebol robótico simulado ou a outro domínio que inclua movimentação espacial de uma equipa de agentes homogéneos ou heterogéneos. Baseia-se em calcular a actualização de um vector contendo os posicionamentos de todos os jogadores da equipa, tendo em consideração possíveis trocas de posicionamentos entre os jogadores.

```
ALGORITMO DynamicPositioningExchange(WorldState, Situation, Positionings)
RETORNA Positionings(TeamSize) - Vector com os posicionamentos na formação de todos os
            jogadores da equipa
PARÂMETROS
    WorldState - Estado do Mundo disponível ao agente
    Positionings[TeamSize] - Posicionamentos atribuídos aos agentes da equipa
    Situation - Situação do jogo
{
// Para cada par de jogadores da equipa exceptuando o guarda-redes averiguar a utilidade
da sua troca de posicionamentos na formação actual
    PARA Jog1 = 2 ATÉ TeamSize-1 FAÇA
        PARA Jog2 = Jog+1 ATÉ TeamSize FAÇA {
// Se as posições de ambos os jogadores forem válidas
            Se PositionValid(Jog1) E PositionValid(Jog2) ENTÃO {
// Calcula as distâncias de cada um dos jogadores à sua posição estratégica e à do colega
                Dist11 = Distance(Position(Jog1),SBSPPosition(Jog1))
                Dist22 = Distance(Position(Jog2),SBSPPosition(Jog2))
                Dist12 = Distance(Position(Jog1),SBSPPosition(Jog2))
                Dist21 = Distance(Position(Jog2),SBSPPosition(Jog1))
// Calcula as adequações de cada um dos jogadores ao seu posicionamento actual e ao do
colega
                Adeq11 = PosAdequacy(Jog1, Positioning[Jog1])
                Adeq22 = PosAdequacy(Jog2, Positioning[Jog2])
                Adeq12 = PosAdequacy(Jog1, Positioning[Jog2])
                Adeq21 = PosAdequacy(Jog2, Positioning[Jog1])
// Calcula a utilidade de efectuar uma troca dinâmica de posições
                Util = ExchangePositions(DPREMode, Situation,
                     Dist11, Dist22, Dist12, Dist21, Adeq11, Adeq22, Adeq12, Adeq21,
                     PosImportance(Positioning[Jog1], PosImportance(Positioning([Jog2])
// Se a utilidade de troca for superior a um dado limiar dependente da situação de jogo,
então troca os posicionamentos
                SE Util > ThresUtil(Situation) ENTAO {
```

```
                    Aux = Positionings[Jog1]
                    Positionings[Jog1] = Positionings[Jog2]
                    Positionings[Jog2] = Aux
                }
            }
        }
    RETORNA Positionings
}
```

Algoritmo 16: Troca Dinâmica de Papéis e Posicionamentos

O cálculo da utilidade das trocas dinâmicas de posicionamentos é uma soma pesada de três factores relacionados respectivamente:

- Ganho de distância relativamente às posições estratégicas;
- Ganho de adequação dos agentes aos posicionamentos;
- Ganho de cobertura de posições importantes.

O algoritmo 17 descreve o método utilizado para realizar este cálculo de utilidade.

```
ALGORITMO ExchangePositions(DPREMode, Situation, Dist11, Dist22, Dist12, Dist21,
        Adeq11, Adeq22, Adeq12, Adeq21, Imp1, Imp2)
RETORNA
    Util - Utilidade da troca de posicionamentos de dois jogadores
PARÂMETROS
    DPREMode - Modo de troca de posicionamentos considerado
    Situation - Situação de jogo
    Dist11, Dist22, Dist12, Dist21 - Distâncias entre as posições actuais e estratégicas
    Adeq11, Adeq22, Adeq12, Adeq21 - Adequações dos agentes aos posicionamentos
    Imp1, Imp2 - Importâncias dos posicionamentos
{
// Cálculo do ganho em distância às posições estratégicas com a troca de posicionamentos
    DistKeep = Dist11 + Dist22
    DistChange = Dist12 + Dist21
    DistGain = DistKeep - DistChange
// Cálculo do ganho em adequação aos posicionamentos com a troca
    AdeqKeep = Adeq11 + Adeq22
    AdeqChange = Adeq12 + Adeq21
    AdeqGain = AdeqChange - AdeqKeep
// Cálculo do ganho em cobertura de posições importantes com a troca. O nível de cobertura
é calculado para o caso de manutenção de posições e troca de posições. Baseia-se na soma
da multiplicação da importância de cada posicionamento pelo respectivo nível de cobertura
    CoverKeep = Imp1*CoverLevel(Dist11,Adeq11) + Imp2*CoverLevel(Dist22,Adeq22)
    CoverChange = Imp1*CoverLevel(Dist21,Adeq21) + Imp2*CoverLevel(Dist12,Adeq12)
    CoverGain = CoverChange- CoverKeep
// Cálculo da utilidade tendo em conta factores WDist, WAdeq e WImp que estão definidos
para cada modo de troca dinâmica de posições e cada Situação de jogo
    Util =  WDist(DPREMode, Situation) * DistGain +
            WAdeq(DPREMode, Situation) * AdeqGain +
            WImp(DPREMode, Situation) * CoverGain
    RETORNA Util
}
```

Algoritmo 17: Calculo da Utilidade de uma Troca Dinâmica de Papéis e Posicionamentos

Embora as funções de utilidade sejam definidas para que convirjam espacialmente no sentido em que para cada par de agentes seja impossível um agente pensar que a troca de posições é útil e outro pensar o oposto, o facto de se tratar de um sistema multi-agente com agentes com percepções distintas do ambiente, torna o problema da troca de posicionamentos mais complexo. Dado os agentes possuírem apenas uma perspectiva local do ambiente, as suas utilidades para uma troca DPRE serão distintas e como tal um determinado agente pode considerar esta troca útil e o outro não. Para a resolução deste problema, é utilizada a comunicação para reforçar a troca dinâmica de posicionamentos. Acresce a este facto que quando um agente considera que a troca de posicionamentos tem uma utilidade positiva, irá tentar realizar essa troca deslocando-se para a posição estratégica do colega de equipa, o que tende a aumentar a utilidade estimada da troca para ambos os agentes.

De forma a testar o algoritmo de troca dinâmica de posicionamentos e papéis foram definidos os seguintes modos de DPRE:

- *DPRE0:* Trocas dinâmicas de posição não são efectuadas;
- *DPRE1:* Trocas dinâmicas de posição pouco frequentes baseadas unicamente em distâncias às posições estratégicas;
- *DPRE2:* Trocas dinâmicas de posição frequentes baseadas unicamente em distâncias às posições estratégicas;
- *DPRE3:* Trocas dinâmicas de posição frequentes baseadas em distâncias às posições estratégicas e adequações aos posicionamentos;
- *DPRE4:* Trocas dinâmicas de posição frequentes baseadas em distâncias às posições estratégicas, adequações aos posicionamentos e importâncias dos posicionamentos.

Uma primeira avaliação do algoritmo de troca dinâmica de posicionamentos desenvolvido foi realizada aferindo o número de descoordenações entre agentes no decurso de uma série de jogos utilizando os modos de *DPRE1* e *DPRE2* (tabela 21)[73].

[73] No RoboCup 2000 e no RoboCup 2001 a troca dinâmica de posicionamentos utilizada pelo FC Portugal foi o DPRE2. Nestas competições, a equipa FC Portugal não utilizou jogadores heterogéneos (introduzidos em 2001).

	Modo DPRE 1					Modo DPRE 2				
Adversário	Num DPRE	Desc1	Desc2	Desc1 /Num	Desc1 /Num	Num DPRE	Desc1	Desc2	Desc1 /Num	Desc2 /Num
CMUnited99	19.6	11.6	2.6	0.59	0.13	32.8	14.1	2.9	0.43	0.09
ATTCMUnited2000	26.3	15.5	3.3	0.59	0.13	36.8	18.1	2.3	0.49	0.06
YowAI2001	34.2	16.6	0.9	0.48	0.03	51.5	17.6	1.5	0.34	0.03
UvaTrilearn	39.2	20.6	2.8	0.52	0.07	62.2	19.6	2.3	0.31	0.04
Karlsruhe Brains.	27.8	17.2	1.6	0.62	0.06	43.8	17.3	1.2	0.39	0.03
Tsinghuaeolus	36.6	17.7	1.6	0.48	0.04	54.6	21.7	1.4	0.40	0.03
Média	30.6	16.5	2.1	0.54	0.08	47.0	18.1	1.9	0.39	0.05

Tabela 21: Descoordenações na Troca Dinâmica de Posições (10 jogos contra cada equipa)

Para efeitos de avaliação foram considerados unicamente dois tipos de descoordenações e sempre relativos aos próprios jogadores:

- *Desc1:* Descoordenações em que dois jogadores pensam que o seu posicionamento é o mesmo durante 2 a 5 ciclos
- *Desc2:* Descoordenações em que dois jogadores pensam que o seu posicionamento é o mesmo durante 6 ou mais ciclos

Analisando os resultados (tabela 21), facilmente se conclui que as descoordenações de tipo *Desc1* ocorrem com alguma frequência, pois é normal que durante 2 a 5 ciclos os jogadores não vejam o colega de equipa e devido às diferenças nos estados do mundo pensem que o posicionamento de ambos é o mesmo. Verifica-se também que as descoordenações de tipo *Desc2* ocorrem pouco frequentemente (média de 0.08 e 0.05 vezes por cada troca de posições). Usualmente estas descoordenações ocorrem quando ambos os jogadores se encontram a tentar interceptar a bola, e como estão em movimento rápido, olhando para bola, perdem deste modo, precisão na posição dos colegas de equipa. Finalmente, verifica-se que a percentagem de descoordenações na troca de posições é menor no modo DPRE2. Este resultado deve-se a que a função de troca de posições tem um declive superior à do modo DPRE1 e como tal é mais fácil que ambos os jogadores se apercebam no mesmo instante da utilidade de uma troca de posições.

9.7 Outros Mecanismos de Coordenação

9.7.1 Coordenação por Comunicação Avançada - ADVCOM

O mecanismo de coordenação por comunicação avançada ADVCOM baseia-se na criação de um estado do mundo comunicado, i.e., um estado do mundo construído, utilizando unicamente a informação proveniente da comunicação dos colegas de equipa, sem fazer uso de informação sensorial ou de predição do próprio jogador (figura 71). Da comparação entre a informação disponível neste estado do mundo comunicado e a informação disponível no seu próprio estado do mundo, cada jogador pode em cada instante aferir do interesse que cada item de informação do seu estado do mundo possui para a equipa.

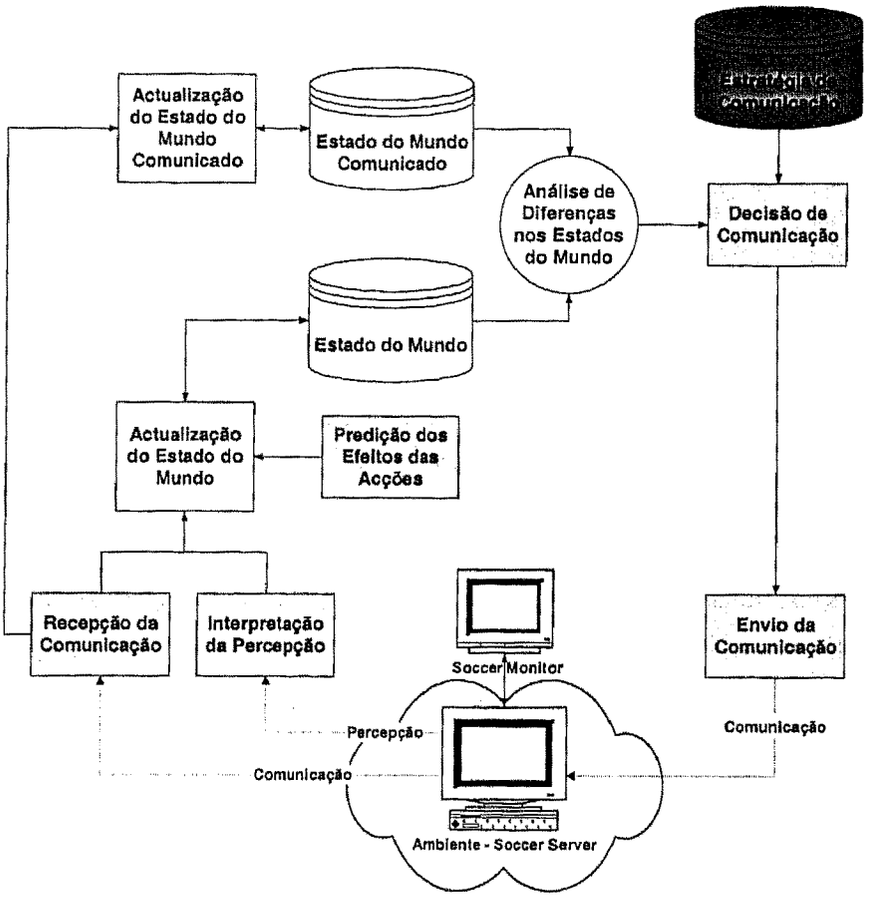

Figura 71: Criação de um Estado do Mundo Comunicado e sua Utilização na Decisão de Comunicação

O algoritmo 18 mostra a utilização da comunicação avançada no domínio do futebol robótico simulado. Tendo em conta a limitada largura de banda disponível, o algoritmo baseia-se em determinar a quantidade de informação a comunicar, e seleccionar para ser comunicada apenas a informação cuja variação entre o estado do mundo comunicado e o estado do mundo normal do agente é mais significativa.

```
ALGORITMO ADVCOM_Communication(WorlsState)
    RETORNA Message - Mensagem a comunicar ou Null
PARÂMETROS
    WorldState - Estado do Mundo disponível ao agente
{
// Cria a estrutura da mensagem a comunicar
    Message = CreateMessage
// Dependendo dos parâmetros de comunicação verifica quantas posições pode comunicar
    NPlayers = GetNumberPlayersCommunication()
// Calcula o interesse em comunicar a informação da bola
    BallInterest = CalculateBallCommunicationInterest(Situation, BallPos, BallConf,
        BallNewPos, BallKicked, BallFast, BallFree)
// Dependendo do interesse da bola decide comunicar ou não a bola e baseado nessa decisão
decide o número de posições de jogadores a comunicar e adiciona a informação da bola à
mensagem
    SE BallInterest > ThresBallInt ENTÃO {
        NPlayers = ReduceNPlayersBall()
        Message = AddInfoBall(Message)
    }
// Selecciona os jogadores cuja posição vai comunicar
    PARA Team = 1 ATÉ 2 FAÇA
        PARA Jog = 1 ATÉ TeamSize FAÇA {
            PlInterest = CalculatePlayerCommunicationInterest(Team, Jog, Situation,BallPos)
            SE NBest(PlInteres, NPlayers) ENTAO Message = AddInfoPlayer(Team, Jog, Message)
        }
// Calcula o interesse total da mensagem (bola e jogadores) e caso este esteja acima de um
determinado limiar retorna a mensagem
    Interest = CalculateMessageTotalInterest(Message)
    SE Interest > ThresCommunicate ENTAO RETORNA Message SENAO RETORNA Null
}
```

Algoritmo 18: ADVCOM – Comunicação Avançada

O cálculo do interesse em comunicar a posição dos jogadores é realizado tendo como base a diferença entre o estado do mundo do agente e o estado do mundo criado exclusivamente por comunicação (algoritmo 19). O algoritmo que determina o interesse da comunicação é independente do domínio e considera que interessa comunicar a posição de jogadores:

- Cuja confiança na posição seja superior à confiança comunicada recentemente ou cuja posição já não seja comunicada há algum tempo. Estas condições são consideradas da mesma forma pois a confiança na posição dos jogadores no estado do mundo comunicado é decrementada em cada ciclo e como tal, a confiança na posição de jogadores que não sejam comunicados há algum tempo, será reduzida;
- Cuja distância relativamente à última posição comunicada seja significativa. Se o jogador ainda se encontra na mesma posição que foi comunicada, é pouco interessante comunicar a sua posição novamente.
- Que estejam em posições de interesse na situação actual de jogo. Não interessa considerar jogadores que estejam muito longe da bola ou que estejam atrás da bola quando a equipa ataca. Interessa bastante comunicar a posição do guarda-redes e dos defesas adversários que se encontram na área, numa situação de

ataque perigoso da equipa.

```
ALGORITMO CalculatePlayerCommunicationInterest(Team, PlayerNumber, Situation, BallPos)
    RETORNA Interest - Interesse em comunicar a posição de um dado jogador
PARÂMETROS
    Team - Equipa do jogador
    PlayerNumber - Número do jogador
    Situation - Situação de jogo
    BallPos - Posição da Bola
{
//Se a confiança está abaixo do limiar mínimo então não interessa comunicar a posição
desse jogador
    PlConf = PlayerPositionConf(Team, PlayerNumber)
    PlPos = PlayerPosition(Team, PlayerNumber)
    SE (PlConf < ConfThreshols) ENTÃO RETORNA 0.0
// Distância à posição que foi comunicada pela última vez deste jogador (posição que se
encontra no estado do mundo comunicado)
    DistComm = Distance(PlPos, CommPosition(Team, PlayerNumber))
// Calcula a diferença entre a confiança da posição no estado do mundo e no estado do
mundo comunicado
    DifConf = PlConf - CommConf(Team, PlayerNumber)
// Calcula três factores: 1) Dependente da distância da posição actual do jogador à última
posição comunicada; 2) Dependente da diferença de confiança na posição do jogador do
estado do mundo e do estado do mundo "comunicado"; 3) Dependente do interesse que um
jogador naquela posição terá para o jogo que se encontra numa dada situação com a bola
numa dada posição
    FactDist = CalculateFactorDistance(DistComm)
    FactConf = CalculateFactorConfidence(DifConf)
    FactPosInt = CalculateFactorPosInterest(PlPos, Situation, BallPos)
// Baseado nos três factores calcula o interesse em comunicar a posição do jogador. O
algoritmo utilizado para isto é essencialmente a limitação dos três factores à escala
[1,10] seguida da multiplicação dos mesmos.
    Interest = CalculateInterest(FactDist, FactConf, FactPosInt)
    RETORNA Interest
}
```

Algoritmo 19: ADVCOM – Interesse em Comunicar a Posição de um Jogador

O cálculo do interesse em comunicar a posição da bola (algoritmo 20) é baseado em diversas heurísticas específicas do domínio. Interessa comunicar a posição da bola se a confiança na sua posição é elevada, se a bola mudou de posição subitamente, se foi chutada por um jogador ou se se desloca rapidamente no campo.

```
ALGORITMO CalculateBallCommunicationInterest(Situation, BallPos, BallConf, BallNewPos,
    BallKicked, BallFast, BallFree)
    RETORNA Interest - Interesse em comunicar a posição e velocidade da bola
PARÂMETROS
    Situation - Situação de jogo
    BallPos - Posição da Bola
    BallConf - Confiança na Posição da Bola
    BallNewPos - Posição da Bola mudou repentinamente
    BallKicked - Velocidade da bola mudou significativamente pelo que esta foi chutada
    BallFast - Bola está em deslocamento rápido no campo (velocidade elevada)
    BallFree - Bola está livre no campo, i.e., nenhum jogador a pode chutar
{
// Se a bola está livre no campo ou o jogador não tem uma elevada confiança na sua posição
então não tem interesse em comunicar a sua posição
    SE BallFree OU BallConf < ConfThreshols RETORNA 0
// Calcula o interesse que a bola tem na presente situação. O interesse será elevado se a
bola mudou de posição ou foi chutada ou desloca-se muito rápido
```

```
    BallInterest = CalculateBallInterest(Situation, BallNewPos, BallKicked, BallFast)
// Distância da bola à posição que foi comunicada pela última vez da bola
    DistBallComm = Distance(PlPos, BallCommPosition()
// Calcula a diferença entre a confiança da posição da bola no estado do mundo e no estado
do mundo comunicado
    DifBallConf = BallConf - BallCommConf()
// Calcula o interesse que comunicar a bola terá, tendo em conta o interesse da bola, a
confiança que o jogador tem na sua posição, a distância da bola à última posição
comunicada e a diferença de confiança entre o estado do mundo e o estado do mundo
comunicado
    Interest = CalculateBallCommInterest(BallInteres, BallConf, DistBallComm, DifBallConf)
    RETORNA Interest
}
```

Algoritmo 20: ADVCOM – Interesse em Comunicar a Posição da Bola

A aplicação do algoritmo de comunicação avançada permite uma diminuição significativa do erro no estado do mundo dos jogadores. Na tabela 22, é analisado o efeito da comunicação avançada no erro médio das posições dos jogadores, utilizando as seguintes estratégias de comunicação avançada:

- **ADVCOM0:** Sem comunicação;
- **ADVCOM1:** Velocidade e Posição da Bola + Velocidade e Posição de Todos os jogadores e respectivas confianças;
- **ADVCOM2:** Velocidade e Posição da Bola + Posição de quatro ou seis jogadores (dependendo se a bola é comunicada) e respectivas confianças;
- **ADVCOM3:** Velocidade e Posição da Bola + Posição de um ou três Jogadores (dependendo se a bola é comunicada) e respectivas confianças.

Na tabela 22 são visíveis os resultados obtidos em 5 jogos contra cada uma das equipas de teste. No cálculo do erro total foi considerado um peso de 5 para o erro na posição da bola relativamente ao erro na posição dos jogadores, e não foram consideradas as posições dos guarda-redes (pois estes estão por vezes muito afastados dos restantes jogadores e como tal a sua posição não é conhecida por estes). Nesta experiência os jogadores que se encontram em comportamento estratégico olham para uma direcção aleatória (entre -90 e 90 graus) e os restantes olham para a bola.

Adversário	*ADVCOM0*	*ADVCOM1*	*ADVCOM2*	*ADVCOM3*
CMUnited99	*5.4*	*1.5*	*1.9*	*3.8*
ATTCMU2000	*7.1*	*1.8*	*3.5*	*5.4*
YowAI2001	*8.3*	*1.7*	*2.8*	*5.6*
UvaTrilearn	*12.1*	*2.1*	*3.6*	*6.2*
Karlsruhe Brains.	*6.9*	*1.8*	*2.4*	*4.5*
Tsinghuaeolus	*9.2*	*1.7*	*3.2*	*4.9*
Média	*8.2*	*1.8*	*2.9*	*5.1*

Tabela 22: Erro Médio no Estado do Mundo com diferentes estratégias de comunicação

Na tabela é visível que a comunicação das posições e velocidades de todos os jogadores diminui significativamente os erros obtidos no estado do mundo. É também visível que os erros com a estratégia ADVCOM0 (sem comunicação) são muito mais elevados do que os restantes. Os maiores erros verificam-se contra a equipa UVATrilearn [de Boer et al., 2001]. Este facto deve-se à forma rápida do seu jogo (utilizando jogadores heterogéneos e passes longos). Caso, não só os jogadores se movimentam mais no campo, como a bola sofre maiores variações de posição, contribuindo para um maior erro médio final.

Na secção 9.13.4 são apresentados os resultados das experiências realizadas de forma a aferir da influência competitiva do ADVCOM.

9.7.2 Coordenação por Modelização Mútua

A utilização da coordenação estratégica, nomeadamente no que diz respeito ao posicionamento estratégico baseado em situações, permite a cada agente da equipa poder estimar a posição de cada colega de equipa se considerar que este se encontra em comportamento estratégico. Se não existir informação visual ou auditiva precisa e actual relativa à posição de um dado agente da equipa, é possível efectuar a previsão da posição deste agente considerando que o mesmo se desloca para a sua posição estratégica (e consequentemente prevendo a sequência de comandos e trajectória que esse agente irá seguir).

O algoritmo 21 demonstra a modelização estratégica utilizada para os elementos da própria equipa. Este conceito foi posteriormente estendido para a equipa adversária, utilizando uma predição prévia da formação dessa equipa, não estando no entanto este trabalho concluído à data de início da escrita desta tese.

O algoritmo de modelização mútua estratégica verifica, para cada jogador de uma equipa, se a confiança na posição desse jogador é inferior a um valor limiar de confiança (senão não utiliza a modelização mútua estratégica). Em caso afirmativo, verifica qual a posição actual do jogador e a sua posição estratégica e efectua a previsão do comando que o jogador irá emitir no sentido de se deslocar para a sua posição estratégica. A nova posição

para o jogador é depois armazenada no estado do mundo do jogador, substituindo a informação previamente disponível.

```
ALGORITMO StrategicMutualModelling(WorldState, Team)
    RETORNA WorldState - Estado do Mundo
PARÂMETROS
    WorldState - Estado do Mundo disponível ao agente
    Team - Equipa do jogador
{
    PARA Jog = 1 ATÉ TeamSize FAÇA {
//Se a confiança está abaixo do limiar mínimo então não interessa comunicar a posição
desse jogador
        PlConf = PlayerPositionConf(Team, Jog)
        SE PlConf < ThresMutualModelling ENTAO {
            PlPos = PlayerPosition(Team, Jog)
            ModelPos = PlayerStrategicPosition(Team, Jog)
            PlNewPos = PredictPlayerPosition(PlPos, ModelPos)
            StorePlayerPos(WorldState, Team, Jog, PlNewPos)
        }
    RETORNA WorldState
}
```

Algoritmo 21: Previsão da Posição dos Agentes da Equipa utilizando Modelização Mútua

Embora o algoritmo seja genérico, diversas heurísticas específicas do domínio foram adicionadas na equipa FC Portugal, no sentido de o tornar mais adequado ao futebol robótico simulado. Para testar a influência da modelização mútua na precisão do estado do mundo dos jogadores foram realizadas experiências com diferentes níveis de limiar de utilização da modelização mútua estratégica (correspondente á confiança na posição do jogador a partir da qual se utiliza a modelização mútua estratégica):

- **MM0:** Sem modelização mútua;
- **MM1:** Modelização mútua com *ThresMutualModelling = 0.90*;
- **MM2:** Modelização mútua com *ThresMutualModelling = 0.94*;
- **MM3:** Modelização mútua com *ThresMutualModelling = 0.97*;

Os resultados obtidos para o erro médio no posicionamento dos jogadores da própria equipa (sem considerar o guarda-redes, a bola e os adversários) em cinco jogos com as equipas de teste, utilizando o esquema de visão descrito anteriormente e sem utilizar comunicação podem ser analisados na tabela 23.

Adversário	MM0	MM1	MM2	MM3
CMUnited99	5.4	4.6	3.2	4.2
ATTCMU2000	5.8	4.9	3.8	5.1
YowAI2001	6.3	4.7	3.7	5.7
UvaTrilearn	7.2	5.8	3.4	5.4
Karlsruhe Brains.	5.9	4.8	4.3	6.1
Tsinghuaeolus	8.2	5.7	4.1	7.4
Média	6.5	5.1	3.8	5.7

Tabela 23: Erro Médio no Estado do Mundo com diferentes estratégias de modelização mútua

Analisando a tabela 23 verifica-se que o erro médio mais baixo é obtido utilizando um limiar de 0.94 (estratégia MM2). A utilização de uma estratégia de modelização mútua com um limiar reduzido (MM1) conduz a que a modelização só comece a ser utilizada quando a confiança na posição dos jogadores já é muito reduzida e produz um efeito de atraso no deslocamento dos jogadores para a sua posição estratégica. A utilização de um limiar elevado (MM3) conduz a que a modelização mútua seja utilizada logo após não ter sido visualizado um jogador num dado ciclo. Como tal, por vezes, mesmo quando os jogadores se encontram em comportamento activo e se afastam das suas posições estratégicas, a sua posição no estado do mundo dos restantes jogadores será sempre muito próxima da posição estratégica (com consequente incremento de erro relativamente à estratégia MM2). Este incremento do erro é aliás mais acentuado contra as equipas Karlsruhe Brainstormers e Tsinghuaeolus, que são equipas que driblam bastante com a bola e fazem com que os jogadores do FC Portugal abandonem frequentemente as suas posições estratégicas para tentarem tirar a bola aos adversários.

9.7.3 Percepção Inteligente

A percepção inteligente baseia-se numa utilização inteligente dos sensores do robô virtual para que a informação que provém destes, em cada instante, permita melhorar a precisão do estado do mundo do agente e incrementar a sua coordenação com os restantes agentes da equipa. Em cada instante os agentes decidem o melhor ângulo de posicionamento do seu pescoço flexível de acordo com a situação de jogo e com o conteúdo do seu estado do mundo de baixo-nível.

Os agentes do FC Portugal podem utilizar o pescoço para olhar de três formas:

- **Visão Activa.** Utilizar o pescoço rotativo de forma a olhar para o local de destino de uma dada acção (por exemplo um passe);
- **Visão da Bola.** De forma a reagir rapidamente a mudanças de velocidade da bola;
- **Visão Estratégica.** De forma a olhar estrategicamente no campo incrementando a

precisão do estado do mundo e maximizando a probabilidade de sucesso das suas acções cooperativas com os restantes agentes.

O algoritmo 22 apresenta o método de decisão entre estas três formas de utilização do pescoço.

```
ALGORITMO DecideLookingDir(WorldState, AgentAction)
    RETORNA LookingDir - Direcção para onde o jogador deve olhar
PARÂMETROS
    WorldState - Estado do Mundo do agente
{
// Se o agente está a executar um passe, um remate ou um passe em profundidade então
determina a direcção para onde vai enviar a bola e caso possa, olha para lá
    SE AgentAction = Shoot OU AgentAction = Pass OU AgentAction = Forward ENTAO {
        ActionDir = CalculateActionRelativeDir(AgentAction)
        SE PossibleToLook(ActionDir) ENTÃO RETORNA ActionDir
    }
// Se a bola for essencial, i.e., a confiança do jogador na sua posição for baixa, estiver
controlada (ou próximo disso) por um outro jogador, estiver muito próxima do jogador ou
deslocar-se muito rápido, então caso possa (i.e. rodando o pescoço consiga olhar nessa
direcção) o agente olha para a bola
    SE BallEssential() ENTAO
        BallDir = CalculateBallRelativeDir()
        SE PossibleToLook(BallDir) ENTÃO RETORNA BallDir
// Caso contraário, o agente olha de forma estratégica
    StrategicLookingDir = StrategicLookingMechanism(WorldState, AgentAction)
    RETORNA StrategicLookingDir
}
```

Algoritmo 22: Decisão da Direcção para onde olhar

A visão estratégica dos agentes é baseada em calcular a direcção, de entre todas as direcções para onde o agente pode olhar, que previsivelmente propiciará um maior incremento na precisão do estado do mundo do agente (algoritmo 23).

```
ALGORITMO StrategicLookingDir(WorldState, Situation, AgentAction)
    RETORNA StrategicLookingDir - Direcção estratégica para onde o jogador deve olhar
PARÂMETROS
    WorldState - Estado do Mundo do agente
    Situation - Situação actual
{
    BestInterest = 0
// Para todos os ângulos possíveis de rotação do pescoço (com uma dada discretização)
calcula a lista de jogadores que estarão visíveis de o jogador olhar nessa direcção (com
uma dada tolerância em termos de distância ao limite do cone de visualização)
    PARA Angle = MinAngle ATÉ MaxAngle FAÇA {
        ListPlayers = CalculateVisiblePlayersDir(Angle, Tolerance)
        TotalInterest = 0
// Para todos os jogadores da lista de jogadores soma o interesse que existe em visualiza-
los. Caso a bola esteja visível na direcção soma um factor de interesse em visualizar a
bola
        PARA Player EM ListPlayers FAÇA {
            Interest = CalculatePlayerVisualInterest(Player, Situation, BallPos)
            TotalInterest = TotalInterest + Interest
        }
        SE BallVisible(Angle) ENTAO
            TotalInterest = TotalInterest + CalculateBallVisualInterest(Situation, BallPos)
// Caso o interesse seja superior ao máximo anterior então actualiza o máximo e a direcção
respectiva
```

```
        SE TotalInterest > BestInterest ENTAO {
            BestInterest = TotalInterest
            StrategicLookingDir = Angle
        }
    }
    RETORNA StrategicLookDir
}
```

Algoritmo 23: Percepção Inteligente (SLM – Strategic Looking Mechanism)

O cálculo do interesse em visualizar um dado jogador ou visualizar a bola é realizado de forma semelhante ao cálculo do interesse em comunicar a posição de um dado jogador realizado no ADVCOM (algoritmo 24). Baseia-se em verificar o aumento de confiança na posição do jogador em causa através da sua visualização e o interesse que esse jogador tem na situação actual.

```
ALGORITMO CalculatePlayerVisualInterest(Player, Situation, BallPos)
    RETORNA Interest - Interesse em visualizar um dado jogador
PARÂMETROS
    Player -Jogador considerado
    Situation - Situação de jogo
    BallPos - Posição da Bola
{
// Calcula a diferença entre a confiança com que será previsivelmente visualizado o
jogador e a confiança da posição no estado do mundo actual
    DifConf = VisualizationConf(Player) - PlayerPositionConf(Player)
// Calcula o interesse da posição do jogador na situação actual
    FactPosInt = CalculateFactorPosInterest(Player, Situation, BallPos)
// Calcula o interesse global em visualizar o jogador (basicamente multiplica o ganho de
confiança pelo factor de interesse do jogador na situação actual)
    Interest = CalculateInterest(DifConf, FactPosInt)
    RETORNA Interest
}
```

Algoritmo 24: SLM – Interesse em Visualizar a Posição de um Jogador

De forma a aferir da influência dos esquemas de visão na precisão do estado do mundo dos agentes foram realizadas experiências com os seguintes esquemas de visão:

- **SLM0:** Visão Aleatória;
- **SLM1:** Visão da Bola sempre que possível, senão Visão Aleatória;
- **SLM2:** Visão Estratégica simples;
- **SLM3:** Visão da Bola quando esta é essencial, senão Visão Estratégica;

Os resultados obtidos para o erro médio no posicionamento de todos os jogadores (sem considerar os guarda-redes, e considerando um peso de 5 para a bola) em cinco jogos com as equipas de teste sem utilizar comunicação nem modelização mútua podem ser analisados na tabela 24.

Adversário	*SLM0*	*SLM1*	*SLM2*	*SLM3*
CMUnited99	*5.4*	*6.8*	*3.7*	*3.8*
ATTCMU2000	*7.1*	*8.3*	*4.2*	*4.9*
YowAI2001	*8.3*	*9.6*	*4.1*	*4.6*
UvaTrilearn	*12.1*	*13.4*	*5.3*	*5.2*
Karlsruhe Brains.	*6.9*	*12.2*	*4.5*	*7.4*
Tsinghuaeolus	*9.2*	*13.6*	*5.1*	*8.1*
Média	*8.2*	*10.7*	*4.5*	*5.7*

Tabela 24: Erro Médio no Estado do Mundo com diferentes estratégias de Visão Estratégica

A utilização de visão direccionada sempre para a bola provoca um decréscimo da precisão no estado do mundo. Este facto é sobretudo visível contra as equipas Karlsruhe Brainstormers e Tsinghuaeolus que ao manterem a bola controlada muito tempo sempre no mesmo local fazem com que os agentes do FC Portugal percam precisão relativamente ao que se passa no resto do campo.

A utilização de visão estratégica simples (sem preocupações com a bola) é a que permite obter melhores resultados a nível de erro médio no estado do mundo. De facto, como na região da bola existe usualmente uma grande concentração de jogadores, utilizando este tipo de visão, os jogadores têm tendência a olhar para essa região frequentemente, visualizando a bola e como tal não perdem precisão na informação relativa à bola. No entanto, o facto de por vezes não verem a bola ser chutada (dado que estão a olhar estrategicamente) e o consequente atraso na reacção para uma possível intercepção, faz com que o esquema de visão SLM2 tenha piores resultados competitivos do que o esquema SLM1. Na secção 9.13.4 são apresentados os resultados das experiências realizadas de forma a aferir da influência competitiva do SLM.

9.8 Decisão de Alto-Nível

Embora as metodologias de coordenação de equipas de agentes sejam extremamente importantes, para obter bons resultados numa competição como o RoboCup, é necessário também dispor de metodologias de decisão individuais competitivas. O futebol é um jogo muito complexo e o *SoccerServer* consegue modelizar essa complexidade de forma muito realista. Em cada situação, mesmo para um humano com bons conhecimentos de futebol e com uma perspectiva global e sem erros do jogo, é muito difícil decidir o que fazer numa determinada situação: Deverá o agente passar a bola? Para quem? Como? Para que posição especificamente? Deverá o agente driblar? Para onde? Deverá o agente segurar a bola e esperar que os colegas se desmarquem? Deverá tentar rematar à baliza? Para que local na baliza? E se não tiver a bola: Deverá o agente tentar interceptar a bola? De que

forma? Deverá fazer um *tackle*? Deverá marcar um adversário? Para um agente autónomo com perspectiva local e incompleta do mundo esta decisão é ainda mais complexa.

As metodologias de decisão de alto-nível individuais do FC Portugal dividem-se em quatro grupos principais:

- Decisão com Bola (posse de bola);
- Decisão sem Bola (recuperação de bola);
- Decisão com o Jogo Interrompido;
- Decisão do Guarda-Redes.

O algoritmo 25 descreve o método de decisão individual dos agentes. Este algoritmo determina qual o comportamento a seleccionar tendo entre os quatro comportamentos referidos tendo em conta a informação contida no estado do mundo do agente.

```
ALGORITMO IndividualDecision(ActionWorldState) RETORNA PlayerDecision
PARÂMETROS
    ActionWorldState - Estado do Mundo de Selecção de Acções
{
// Se o jogador não tiver confiança na sua própria posição ou na posição da bola vai
efectuar acções conducentes a melhorar a sua localização ou localizar a bola
    SE PlayerPosInvalid OU BallPosInvalid ENTAO RETORNA Decision_Localize
// Se o jogo estiver parado então efectua uma decisão de jogo parado
    SE GameStopped ENTAO RETORNA Decision_GameStopped
// Se o jogador estiver em situação crítica vai tomar uma decisão activa
    SE CriticalSituation ENTÃO {
// Se o jogador for o guarda-redes e o seu comportamento actual for de guarda-redes (pois
este agente pode-se comportar da mesma forma que os restantes jogadores)
        SE GoaliePlayer E GoalieBehavior ENTAO RETORNA Decision_Goalie
// Se o jogador estiver com a posse de bola e no próximo ciclo não for possível obter uma
posse mais segura de bola então efectua uma decisão com posse de bola senão efectua uma
decisão de recuperação de bola
        SENAO { SE BallPossession() E NOT(BetterPossessionNext()) ENTAO
            RETORNA Decision_BallPossession
            SENÃO RETORNA Decision_BallRecovery }
    }
// Se não estiver numa situação crítica então decide estrategicamente, i.e. efectua o
posicionamento estratégico
    SENÃO RETORNA Decision_StrategicPositioning
}
```

Algoritmo 25: Decisão Individual dos Agentes

O módulo de decisão do guarda-redes foi construído de forma separada pois os processos de decisão deste agente são muito distintos dos processos de decisão dos restantes agentes da equipa. Realça-se o facto de que o agente guarda-redes, pode em qualquer instante assumir o comportamento de um jogador comum, utilizando os processos de decisão

disponíveis para este tipo de agentes[74]. Nesta secção serão descritos, com um nível de detalhe reduzido, os processos de decisão individual dos agentes da equipa FC Portugal.

9.8.1 Decisão com Bola

O algoritmo de decisão com bola do FC Portugal é baseado na selecção entre seis tipos de acções possíveis em cada instante:

- **Remate à Baliza** *("Shoot").* Rematar a bola em direcção à baliza na tentativa de marcar um golo;
- **Passe Directo** *("Pass").* Passar a bola com velocidade na direcção aproximada de um colega de equipa. O colega de equipa deverá interceptar a bola ainda em movimento;
- **Passe em Profundidade** *("Forward").* Passar a bola para um ponto do campo onde um colega de equipa (ou o próprio jogador) a poderá recolher (já sem muita velocidade ou mesmo parada);
- **Drible** *("Dribble").* Avançar numa determinada direcção no campo mantendo a bola controlada (ou seja, sempre na área de chuto do jogador);
- **Aliviar a Bola** *("Clear").* Chutar a bola numa direcção em que evite adversários e a retire do local onde está (que usualmente é um local perigoso);
- **Segurar a Bola** *("Hold").* Manter a bola controlada sem que o jogador se movimente no campo, procurando evitar que os adversários mais próximos consigam chutar a bola.

A decisão de qual a acção a executar em cada instante é baseada no cálculo de um conjunto de factores de avaliação para cada acção possível. Os factores de avaliação são posteriormente utilizados no cálculo de uma avaliação global que não é mais do que uma soma pesada dos diversos factores de avaliação (algoritmo 26). Os pesos utilizados nesta soma pesada são diferentes para cada tipo de jogador, permitindo assim a criação de jogadores com comportamentos distintos.

```
ALGORITMO IndividualDecision(Role, ActionWorldState) RETORNA PlayerBallDecision
PARÂMETROS
    Role - Papel atribuído ao jogador na formação actual
    ActionWorldState - Estado do Mundo de Selecção de Acções
{
    Shoots = CalculateNBestShoots(ActionWorldState)
    Passes = CalculateNBestPasses(ActionWorldState)
    Forwards = CalculateNBestForwards(ActionWorldState)
```

[74] Esta potencialidade é aliás utilizada no FC Portugal, nomeadamente em tácticas que utilizam a formação designada por 443 (formação essa em que o guarda-redes abandona a baliza e se comporta como um defesa central).

```
    Dribbles = CalculateNBestDribbles(ActionWorldState)
    PlayerBallDecision = DecideBest(Shoots, Passes, Forwards, Dribbles)
}

ALGORITMO DecideBest(Role, Shoots, Passes, Forwards, Dribbles) RETORNA PlayerBallDecision
PARÂMETROS
    Role - Papel atribuído ao jogador na formação actual
    Shoots, Passes, Forwards, Dribbles - Melhores acções de cada tipo
{
// Determina o papel de posse de bola atribuído ao jogador
    RoleBall = GetBallPosssessionRole(Role)
// Pesa as decisões utilizando os pesos que o papel atribui a cada tipo de acção
    ShootValue = BestShootValue(Shoots) * RoleShootValue(RoleBall)
    PassValue = BestPassValue(Shoots) * RolePassValue(RoleBall)
    ForwardValue = BestForwardValue(Shoots) * RoleForwardValue(RoleBall)
    DribbleValue = BestDribbleValue(Shoots) * RoleDribbleValue(RoleBall)
// Decide a melhor acção comparando os valores pesados de cada acção
    PlayerBallDecision = GetBest(ShootValue, PassValue, ForwardValue, DribbleValue)
    RETORNA PlayerBallDecision
}
```

Algoritmo 26: Decisão Com Bola (Posse de Bola)

As acções *Hold* e *Clear* não foram utilizadas pelo FC Portugal até agora em competições oficiais (nem foram utilizadas nas experiências realizadas no âmbito desta dissertação) pelo que não serão descritas nesta secção.

9.8.1.1 Geração e Avaliação de Passes

A geração dos passes numa determinada circunstância é realizada considerando algumas das linhas de passe possíveis em cada instante. Uma discretização parameterizável destas linhas de passe é utilizada na geração. Dependendo da região do campo, passes para trás não são gerados. Na figura 72 é apresentado um esquema dos passes gerados numa determinada situação e que vão ser sujeitos a avaliação.

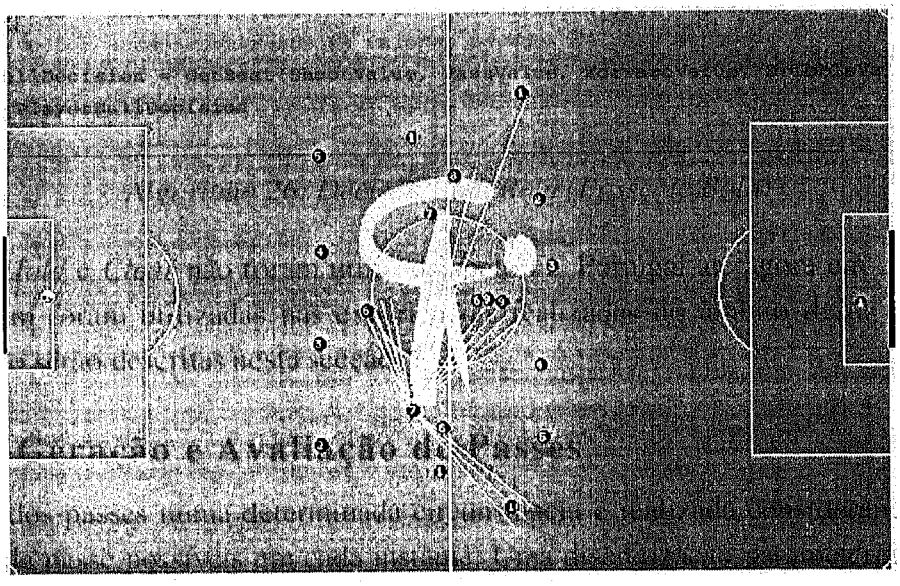

Figura 72: Geração de Passes

Analisando a figura é visível que na região considerada (meio campo defensivo) são gerados unicamente passes que não sejam muito para trás (i.e. com a direcção aproximada da própria baliza) e que tenham a direcção aproximada de jogadores da equipa.

A avaliação dos passes é realizada utilizando as seguintes medidas (consideradas numa escala qualitativa *{Very_Poor, Poor, Medium, Good, Very_Good}* convertível para a escala quantitativa *{0,1,2,3,4}*):

- **Valor Posicional (*vpos*).** Ganho posicional obtido se o passe for bem sucedido. A cada região do campo é atribuído um valor posicional, sendo o valor da região próxima da baliza adversária o mais elevado e o valor da região próxima da própria baliza o mais reduzido. O ganho posicional é obtido pela diferença entre o valor da região de destino do passe e o valor da região de origem.
- **Valor do Cone (*vcone*).** Cálculo da probabilidade de sucesso do passe em termos das possíveis intercepções dos oponentes que se encontram no cone que une a posição de origem à posição de destino do passe.
- **Valor da Confiança na Posição (*vcp*).** Confiança na posição do receptor. Se o jogador estiver a olhar para o receptor a confiança será elevada em passes para jogadores que se encontram nas costas do jogador, o valor da confiança será reduzido.
- **Valor da Confiança na Recepção (*vcr*).** Confiança que o jogador receptor consiga interceptar o passe. Passes efectuados directamente para uma posição muito próxima do receptor terão uma confiança mais elevada do que passes efectuados para uma posição distante do jogador.
- **Valor do Tipo de Recepção (*vtipo*).** Passes em que o jogador ao realizar a sua intercepção, receba a bola voltado para próximo da baliza adversária têm um valor de tipo superior a passes cuja recepção seja de costas para a baliza adversária, pois permitem avançar mais rapidamente no campo.
- **Valor de Erro (*verro*).** Valor relacionado com o que acontece no caso do passe chegar próximo do receptor mas ser mal sucedido, i.e., o receptor não conseguir interceptar a bola. No caso de a bola sair do campo, ficar numa região perigosa ou ficar em posse de um adversário, este valor será reduzido. No caso de outros jogadores da equipa poderem emendar um erro do receptor, interceptando a bola, então o valor de erro será elevado.
- **Valor de Distância (*vdist*).** Passes com uma distância média são preferidos a passes muito curtos ou muito longos. Este facto deve-se a que passes curtos podem ser mal sucedidos devido ao receptor não reagir imediatamente (por exemplo por não ver a bola partir) e passes longos podem ser mal sucedidos pois um pequeno erro no ângulo de saída da bola para o passe pode provocar um grande erro na posição de recepção.

- **Valor do Congestionamento Final (*vcongf*).** Passes para regiões do campo livres de adversários, i.e., com baixo congestionamento, são preferidos a passes para regiões congestionadas.
- **Valor da Congestionamento Inicial (*vcongi*).** Congestionamento inicial junto ao ponto de partida da bola na região por onde a bola irá passar de forma a ser enviada para o destino final. Convém evitar as zonas mais congestionadas à saída, para que o passe não seja imediatamente interceptado.
- **Valor do Guarda-Redes (*vgr*).** Possibilidade de intercepção do guarda-redes adversário tendo em conta a sua qualidade. Esta medida é útil sobretudo em centros para a área.
- **Valor de Chuto (*vchuto*).** Possibilidade de o passe ser uma assistência que permita ao receptor chutar a bola para a baliza. Se o receptor tiver uma linha de chuto livre para a baliza então o valor de chuto será muito elevado.

A avaliação final de cada passe é a soma pesada de todos os factores referidos, utilizando os pesos definidos no tipo de jogador (posse de bola):

$$VPass_i = vpos_i{*}fpos_p + vcone_i{*}fcone_p + vcp_i{*}fcp_p + vcr_i{*}fcr_p + vtipo_i{*}ftipo_p + verro_i{*}ferro_p + vdist_i{*}fdist_p + vcongf_i{*}fcongf_p + vcongi_i{*}fcongi_p + vgr_i{*}fgr_p + vchuto_i{*}fchuto_p$$

$$\forall i = 1..npasses \qquad p \in \{1..ntipos\}$$

Os melhores passes (os que têm a avaliação mais elevada) são analisados no sistema de avaliação de alto-nível e comparados com os melhores *forwards*, remates e dribles. No sistema implementado na prática no FC Portugal foram também incluídos limites (máximo e mínimo) para cada parâmetro de avaliação.

9.8.1.2 Geração e Avaliação de Passes em Profundidade (Forwards), Remates e Dribles

Os passes em profundidade incluem todos os passes que permitam colocar a bola parada em pontos de uma matriz discretizada com dimensão configurável (por defeito de 25 por 40 metros). Na figura 73 é apresentado um esquema explicativo dos possíveis passes em profundidade num dado instante.

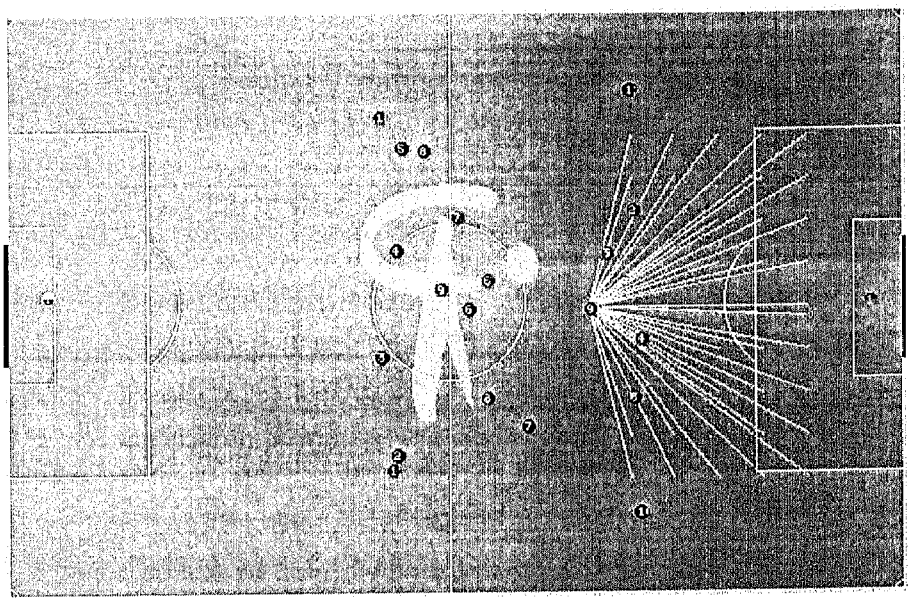

Figura 73: Geração de Passes em Profundidade (Forwards)

Para os remates, o procedimento é semelhante, sendo geradas como possíveis acções os remates para todos os pontos da baliza adversária, espaçados de 0.5 metros.

A avaliação dos *forwards,* remates e dribles é realizada de forma semelhante aos passes, utilizando no entanto parâmetros de avaliação distintos.

Detalhes sobre a avaliação destas acções do FC Portugal podem ser consultados em [Reis e Lau, 2001b] e [Reis e Lau behavior selection, 2002]

9.8.1.3 Fusão das Avaliações

A fusão das avaliações deverá ser realizada aplicando uma análise de alto-nível aos parâmetros de avaliação calculados para as n melhores acções de cada tipo. Na prática, o avaliador utilizado foi simplesmente uma comparação entre o valor final da avaliação da melhor acção de cada tipo pesado utilizando os factores definidos no tipo de jogador para cada acção de posse de bola. As restantes acções foram utilizadas unicamente para efeitos de correcção do código e afinação dos parâmetros de avaliação utilizando a ferramenta *Visual Debugger* cuja descrição resumida se encontra na secção 9.9.1.

9.8.1.4 Avaliação da Decisão com Bola

De modo a avaliar a qualidade da decisão com bola foram realizadas algumas experiências em 2000, analisando a qualidade desta decisão aferida por um especialista do domínio[75] [Reis e Lau, 2001b]. Os resultados obtidos na realização de 2 jogos (contra a equipa Mainz Rolling Brains) encontram-se na tabela 25.

---

[75] O especialista foi um treinador profissional de futebol da primeira liga portuguesa.

Decisão	Nº Ocorrências	Qualidade da Decisão			Qualidade dos Parâmetros		
		Boa	Média	Má	Boa	Média	Má
Pass	191	144 (75%)	34 (18%)	13 (7%)	128 (89%)	7 (5%)	9 (6%)
Forward	136	98 (72%)	23 (17%)	16 (11%)	43 (44%)	40 (41%)	15 (15%)
Dribble	130	62 (48%)	45 (34%)	23 (18%)	44 (71%)	15 (25%)	3 (4%)
Shoot	34	23 (67%)	11 (33%)	0 (0%)	21 (92%)	1 (4%)	1 (4%)
Total	*491*	327 (67%)	113 (23%)	52 (10%)	236 (72%)	63 (19%)	28 (9%)

Tabela 25: Resultados experimentais na avaliação da decisão com bola

A avaliação realizada incidiu sobre a qualidade de cada decisão tomada pelo agente (passe, passe em profundidade, drible ou remate) e pela qualidade dos parâmetros seleccionados na sua execução (velocidade, rapidez, posição). Embora o método de avaliação envolva alguma subjectividade, é visível que a decisão dos agentes é em geral acertada. Os resultados experimentais demonstram que em mais de dois terços dos casos, mesmo possuindo informação local e incompleta os agentes realizam decisões de boa qualidade e escolhem os parâmetros adequados na sua execução. Uma das conclusões interessantes, apontadas pelo especialista do domínio, era que os agentes driblavam demasiado em vez de realizarem passes rápidos ou passes em profundidade. Esta conclusão foi utilizada de forma a melhorar o estilo de jogo em torneios disputados após a realização deste estudo[76] [Reis e Lau, 2001b].

9.8.2 Decisão sem Bola

A decisão sem bola da equipa FC Portugal é realizada através da decisão de um dos seguintes comportamentos de alto-nível [Reis e Lau, 2001b]:

- **Intercepção.** Intercepção de uma bola em movimento no ponto do campo que permita a intercepção mais rápida possível ao jogador;
- **Intercepção Passiva.** Intercepção de uma bola em movimento esperando que esta atinja um determinado ponto onde o jogador aguarda a sua chegada;
- **Intercepção Arriscada.** Intercepção com baixa probabilidade de sucesso mas que dependendo dos erros na percepção, movimento e acção, pode eventualmente ser bem sucedida. Este comportamento é seleccionado em casos em que é importante interceptar a bola mas não existe qualquer intercepção segura possível;
- **Aproximação da Bola.** Embora não seja possível interceptar a bola, o jogador

[76] O estudo foi realizado em Maio de 2000.

aproxima-se desta. Este comportamento é utilizado para limitar o espaço disponível a adversários que controlam a bola;

- **Marcação de um Adversário.** O jogador marca um dado adversário, mantendo uma posição próxima deste;
- **Marcação de uma Linha de Passe.** O jogador marca a linha de passe de um adversário para outro adversário, mantendo uma posição em cima desta linha;
- **Marcação de Pontos.** O jogador marca um determinado ponto fixo pré-especificado, mantendo uma posição próxima deste ponto;
- **Desmarcação.** O jogador desmarca-se deslocando-se para um ponto livre de oponentes no campo.

A selecção do comportamento de recuperação de bola é efectuada através de um conjunto de regras que tendo em conta a informação do estado do mundo activo, permite atribuir um valor à execução de cada comportamento de recuperação de bola.

Tal como nos comportamentos de posse de bola, o tipo de jogador influencia significativamente o comportamento do jogador. Na definição do tipo de jogador são incluídos os pesos atribuídos à execução de cada comportamento de recuperação de bola. Tendo em conta os valores calculados para os vários comportamentos e os respectivos pesos definidos no tipo de jogador, são calculados os valores finais associados à execução de cada comportamento de recuperação de bola. O comportamento com o valor mais elevado é seleccionado para execução. Uma descrição mais detalhada dos algoritmos de recuperação de bola do FC Portugal pode ser encontrada em [Reis e Lau, 2001b] [Reis e Lau, 2001c].

9.8.3 Decisão do Guarda-Redes

O comportamento do guarda-redes é um dos mais afectados com a inexistência de uma terceira dimensão no futebol robótico simulado. Como não é possível passar a bola sobre o guarda-redes, diversas equipas utilizam estratégias de posicionamento do guarda-redes baseadas na sua colocação muito avançada e próxima dos limites da grande-área. Até ao ano de 2000, dado que a grande maioria das equipas utilizava essencialmente acções individuais dos seus jogadores como forma de chegar próximo da baliza adversária, esta estratégia era bastante eficaz. Desta forma, a estratégia do guarda-redes do FC Portugal 2000 incluía o seu posicionamento numa posição bastante avançada. No entanto, após o ano de 2000, a grande maioria das equipas optou por, tal como o FC Portugal em 2000, efectuar movimentos colectivos de ataque baseados na utilização dos flancos e centros para o interior da área. Deste facto, resultou que esta estratégia de defesa deixou de ser eficaz e o recuo do guarda-redes para uma posição mais típica do futebol real, foi realizada por quase todas as equipas, incluindo o FC Portugal 2001.

O guarda-redes, se a bola estiver próxima, utiliza sempre o modo de visão *narrow*. A utilização deste modo de visão permite-lhe obter informação visual em todos os ciclos o que lhe dá uma muito maior reactividade, essencial para o tipo de tarefa a executar por este agente.

A visão estratégica é unicamente utilizada pelo guarda-redes quando a bola não se encontra muito próxima e está claramente livre (não se encontra ao alcance de qualquer jogador). Nos outros casos, o guarda-redes prefere controlar a posição e velocidade da bola, estando constantemente a olhar para bola.

O guarda-redes do FC Portugal, embora seja idêntico aos restantes agentes da equipa[77], possui um conjunto de comportamentos bastante distinto dos restantes jogadores, incluindo a acção catch, a execução de pontapés de baliza, intercepções desesperadas e intercepções "às cegas". No anexo 9 é incluída uma descrição dos comportamentos do guarda-redes.

O desenvolvimento do guarda-redes do FC Portugal implicou ainda o desenvolvimento de um comportamento especial que permite que este agente se movimente mantendo sempre que possível a bola visualmente controlada.

Devido ao facto do guarda-redes ser um agente idêntico aos restantes, o seu comportamento pode variar ao longo do jogo e este agente pode, inclusivamente, optar por utilizar um comportamento idêntico aos dos restantes jogadores e jogar de forma idêntica aos restantes jogadores da equipa numa outra posição (avançado, defesa central, etc.). Esta flexibilidade táctica foi testada no nosso laboratório e tendo em conta os bons resultados obtidos (ver secção 9.12) foi explorada pela equipa FC Portugal (também com sucesso) em competições oficiais[78].

9.9 Ferramentas Associadas

No âmbito do projecto FC Portugal foram desenvolvidas cinco ferramentas de análise de jogos e *debug* da equipa:

- *Visual Debugger*;
- *Client Off-Line*;

---

[77] A única diferença do guarda-redes relativamente aos restantes jogadores consiste nos parâmetros da sua configuração inicial.

[78] Em Seattle no campeonato do mundo RoboCup 2001, nos quartos de final, a equipa FC Portugal encontrava-se a perder 0-1 contra os holandeses UvaTrilearn já perto do final do jogo. Após uma mudança táctica que implicou o guarda-redes jogar como defesa central, a equipa conseguiu inverter o resultado e vencer o jogo com o resultado final de 4-1.

- *WSMetrics* - Analisador do Erro do Estado do Mundo;
- Analisador de jogo e cálculo de estatísticas;
- Visualizador 3D.

Devido ao facto de a investigação realizada no desenvolvimento destas ferramentas ser um pouco periférica à investigação central realizada nesta tese[79], a sua descrição será realizada de modo muito resumido.

9.9.1 *Visual Debugger*

O *Visual Debugger* é a ferramenta principal de teste da equipa FC Portugal. Permite analisar o que os agentes da equipa vêem, sentem, ouvem, pensam e fazem ao longo de um jogo de futebol robótico simulado. Foi inicialmente construído através da integração do *SoccerMonitor* tradicional com a ferramenta *"layered extrospection"* desenvolvida pela Carneggie Mellon University [Stone et al., 1999] [Stone et al., 2000b]. A ferramenta foi posteriormente alterada eliminando as dependências de rede e adicionando uma série de novas capacidades.

O *Visual Debugger* baseia-se na gravação, por parte dos jogadores em análise, de *logfiles* especiais no decurso de um jogo, que depois podem ser analisados em conjunto com o *logfile* do jogo gravado pelo simulador. Ao ser visualizado, o filme do jogo é complementado com a informação gravada no respectivo *logfile* por cada jogador incluindo o conteúdo do seu estado do mundo, as suas percepções, decisões e acções. Esta informação pode ser visualizada num formato textual ou sob a forma de símbolos gráficos, incluindo: círculos, quadrados, rectângulos, linhas, cruzes. Nas figuras 74 e 75 são visíveis duas imagens do *Visual Debugger*.

[79] Aliás, a implementação do *Offline Client* e *WSMetrics* foi realizada essencialmente pelo Nuno Lau, a implementação do *Visualizador 3D* pelo Sérgio Louro e a implementação do *Visual Debugger* foi realizada também em grande parte pelo Nuno Lau.

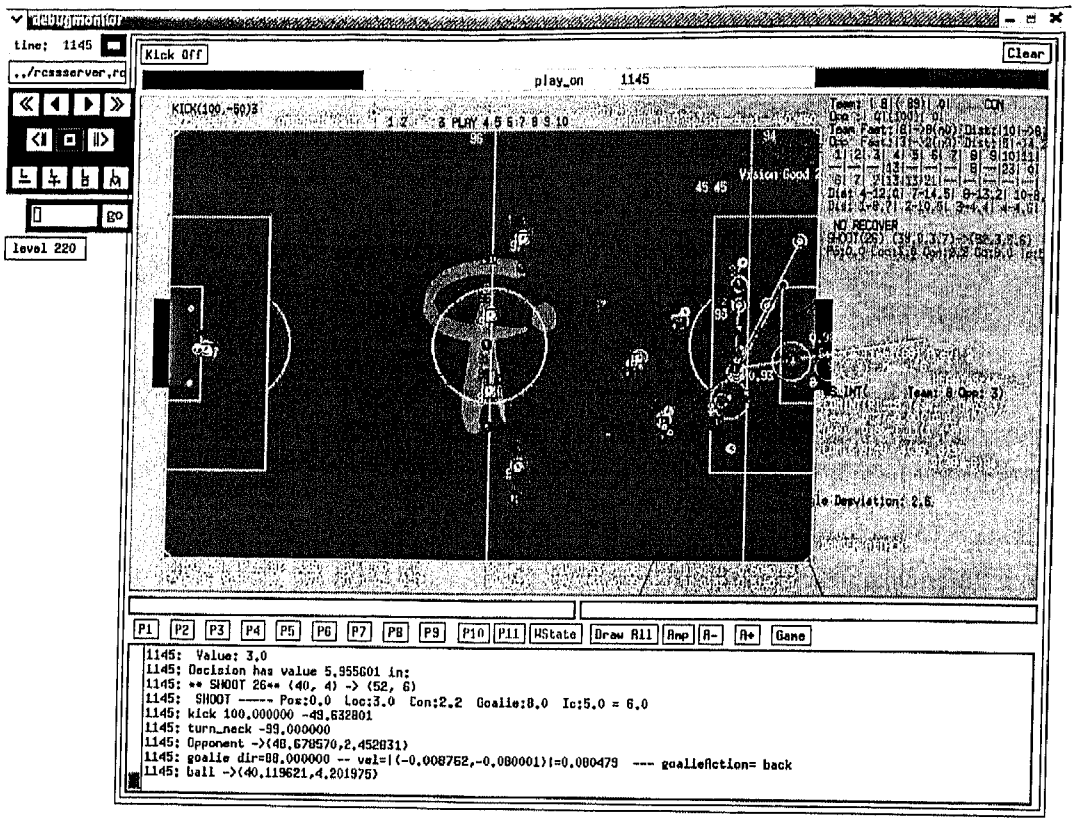

Figura 74: Interface do Visual Debugger – Decisão de Chuto à Baliza

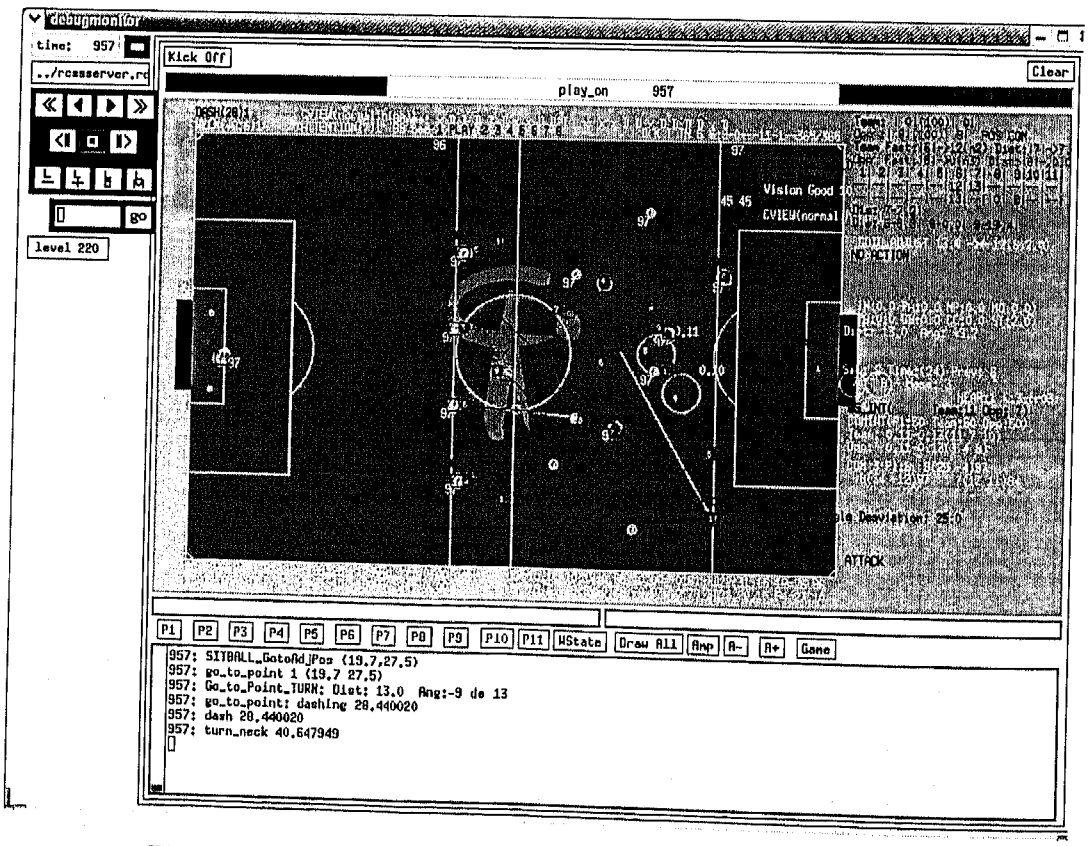

Figura 75: Interface do Visual Debugger – Decisão sem Bola

O *Visual Debugger* permite ainda visualizar ampliações de regiões do campo. Permite também efectuar comparações entre o estado do mundo dos jogadores e o estado do mundo real. Informação mais detalhada sobre esta ferramenta e uma versão para *download* da mesma pode ser encontrada em [Reis e Lau, 2001d].

9.9.2 Offline Client

Em diversas situações mesmo com a informação disponibilizada pelo *Visual Debugger*, não era possível compreender as razões de uma determinada decisão de um jogador num dado momento. Para efectuar uma análise de baixo-nível detalhada deste tipo de situações foi desenvolvida a ferramenta *Offline Client*. Esta ferramenta permite efectuar a repetição exacta do comportamento individual de um jogador num dado ciclo do jogo, podendo desta forma utilizar um *debugger* normal (tal como o *gdb*) para examinar o conteúdo das variáveis do jogador, introduzir pontos de quebra no código, etc.

De forma a poder utilizar o *Offline Client*, um *logfile* especial que regista todas as interacções entre o simulador e o cliente, e todos os sinais temporais é gerado e armazenado. Para repetir a execução do jogador, foi implementada uma rotina que lê o conteúdo deste *logfile* e o trata exactamente como se fosse a comunicação via *socket UDP* com o simulador. Sendo conhecida toda a informação recebida do simulador e os instantes temporais em que se deu essa recepção, é então possível repetir a execução do agente e utilizar um *debugger* normal para detectar e corrigir erros de baixo-nível. É também possível alterar o próprio código do agente e efectuar análises do tipo "*What if...*", analisando o que o agente faria numa determinada situação, utilizando diferentes algoritmos de decisão. Informação mais detalhada sobre esta ferramenta pode ser encontrada em [Reis e Lau, 2001d]

9.9.3 Analisador do Erro no Estado do Mundo

A disponibilidade de um estado do mundo actualizado e preciso é de extrema importância para o processo de tomada de decisão dos agentes. De forma a analisar a precisão do estado do mundo dos agentes da equipa foi desenvolvida uma ferramenta – *WSMetrics* com o objecto de calcular o erro no estado do mundo de cada agente ao longo de um jogo. O *WSMetrics* inclui rotinas que permitem a construção de dois ficheiros especiais contendo informação em cada ciclo do jogo relativa ao estado do mundo estimado pelo agente e do estado do mundo real utilizado pelo simulador. A ferramenta em si é depois capaz de ler os dois ficheiros referidos e calcular diversos tipos de erros entre o estado do mundo do agente e o estado do mundo real, incluindo erros na posição e velocidade da bola e na posição, velocidade e orientações de todos os jogadores.

Utilizando esta ferramenta é possível testar e comparar diversas abordagens de actualização do estado do mundo dos agentes. É também possível comparar o efeito das estratégias de visão e de comunicação na qualidade do estado do mundo dos agentes.

9.9.4 Analisador de Jogo e Cálculo de Estatísticas

O analisador de jogo calcula diversas estatísticas do jogo que podem depois ser utilizadas pelo treinador para definir a melhor táctica a utilizar. Entre estas estatísticas destacam-se a localização da bola por região do campo, número de remates às balizas e número de oportunidades de golo de cada equipa.

O analisador de jogo encontra-se actualmente integrado numa ferramenta mais extensa designada por *Team Designer*. O *Team Designer* permite também efectuar a edição gráfica de estratégias de jogo para a equipa FC Portugal (incluindo tácticas, formações, e tipos de jogadores). Permite ainda funcionar como treinador *off-line* ensaiando e treinando jogadas estudadas e outras situações de jogo.

9.9.5 Visualizador 3D

O visualizador 3D é uma ferramenta definida com o objectivo de permitir a visualização de jogos bidimensionais de futebol robótico simulado num ambiente tridimensional incluindo animação e som. A implementação foi realizada maioritariamente por Sérgio Louro [Louro et al., 2003] como parte da sua tese de Mestrado.

O sistema permite não só a visualização em tempo-real de jogos de futebol robótico simulado (através da conexão ao *soccerserver*) como também efectuar a visualização *off-line* de *logfiles*. O sistema é em si próprio um sistema multi-agente em que cada câmara é modelizada como um agente inteligente autónomo [Louro et al., 2003]. O sistema inclui ainda agentes para o realizador e para o público.

Os agentes "*camaramen*", de acordo com as suas percepções (posições e velocidades dos jogadores e da bola), decidem a melhor perspectiva de filmagem e *zoom* para a câmara que controlam. Estes agentes enviam informação de alto nível para o agente realizador que baseado nesta informação decide qual a imagem ou imagens a mostrar aos espectadores.

O sistema Virtual 3D inclui como principais potencialidades:

- Ambiente 3D incluindo um estádio e animações realistas;
- Controle de câmara inteligente através de um Sistema Multi-Agente;
- Repetições em câmara lenta dos melhores momentos do jogo;
- Geração de sons 3D baseado em agentes emocionais que representam os espectadores de cada equipa;
- Capacidade de conexão em tempo-real com o *soccerserver* e de leitura de *logfiles*

(versão 2 e 3);

- *Display* de estatísticas de jogo fornecidas por um agente analisador de jogo;

O sistema foi desenvolvido em linguagem C++ para a plataforma Windows XP. Permite a ligação em tempo-real ao *soccerserver*. Permite ainda a ligação em tempo-real ao agente analisador de jogo que fornece estatísticas de jogo que são disponibilizadas utilizando uma interface própria definida para o efeito. A arquitectura do sistema de visualização pode ser analisada na figura 76.

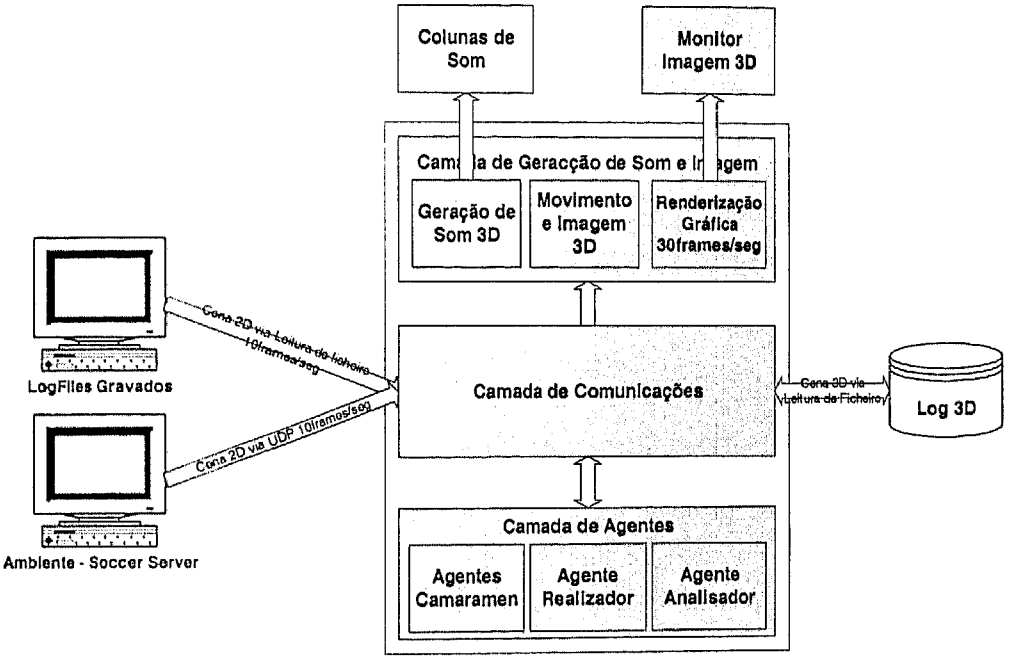

Figura 76: Arquitectura do Visualizador 3D

O sistema permite efectuar a leitura de ficheiros contendo *logfiles* dos jogos ou a ligação directa ao servidor no decurso de um jogo. Permite também efectuar a preparação, gravação e leitura de *logfiles* contendo já informação tridimensional do jogo. A camada de comunicações é responsável por efectuar a ligação entre os diversos módulos do sistema.

A camada de agentes contém três tipos de agentes: *camaramen*, realizador e analisador. Os agentes *camaramen* são responsáveis por decidir os parâmetros de filmagem das suas câmaras (ângulo de abertura, direcção e zoom) e enviar informação de alto-nível sobre a imagem visualizada ao agente realizador. O agente analisador é responsável por calcular diversas estatísticas de jogo e as enviar ao agente realizador. O agente realizador é responsável por decidir em cada instante qual a imagem e estatísticas de jogo a mostrar.

A camada de som e imagem contém as rotinas de geração de som e imagem tendo em conta a câmara utilizada na visualização de jogo e os seus parâmetros seleccionados pelo *camaramen*.

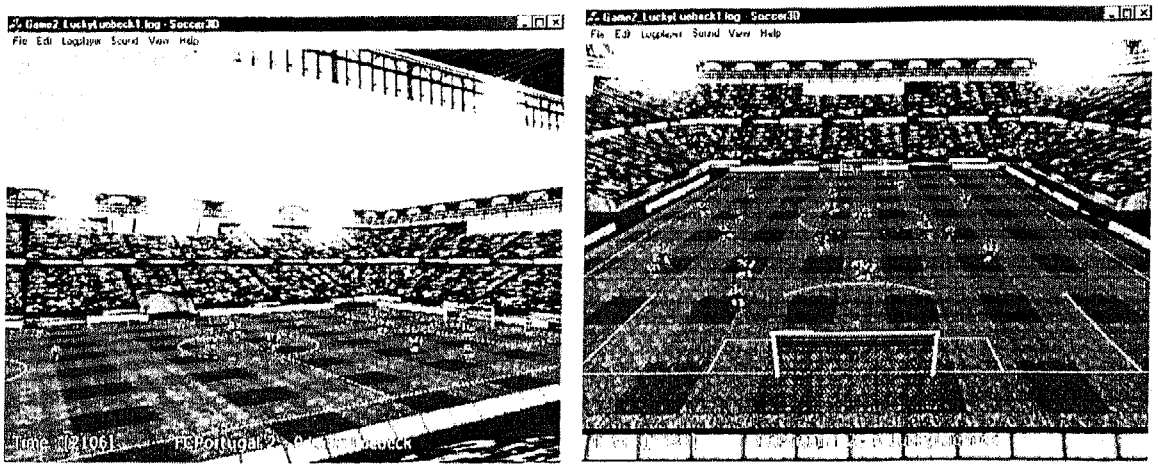

Figura 77: Interface do Visualizador 3D

Embora o visualizador 3D esteja ainda no início do seu desenvolvimento, o impacto do seu primeiro protótipo nos espectadores foi extremamente animador[80]. Os espectadores confirmaram que a visualização de jogos da liga de simulação utilizando o visualizador 3D, transmitia a sensação de observar jogos de futebol real na televisão. Este facto está relacionado não só com o ambiente 3D mas também com as animações realistas, efeitos de som e controlo inteligente de câmara.

9.10 A Linguagem *Coach Unilang*

9.10.1 Introdução

Embora a disponibilização de tácticas flexíveis seja essencial para dispor de uma equipa de futebol robótico robusta e capaz de jogar contra equipas com estratégias distintas, a capacidade de seleccionar a melhor táctica em cada instante é essencial. A funcionalidade de selecção da táctica pode ser inserida nos jogadores desde que a informação utilizada nesta selecção seja idêntica em todos os jogadores. No entanto, esta informação inclui estatísticas do jogo e informação relativa ao comportamento do oponente que é muito difícil de ser calculada da mesma forma por todos os jogadores.

A linguagem COACH UNILANG permite treinar uma equipa a vários níveis distintos. A linguagem introduz ainda um agente assistente de treinador que é capaz de analisar o jogo (directamente ou lendo um *logfile*) e recolher informação do oponente e informação estatística do jogo a vários níveis. Este agente pode ser usado para ajudar o treinador, fornecendo-lhe informação de alto-nível que possa ser utilizada para decidir a táctica a

[80] O primeiro protótipo deste sistema obteve o terceiro lugar na competição de sistemas de visualização e apresentação realizada no RoboCup 2002.

utilizar em cada instante de forma mais eficiente. O treinador assistente pode ser utilizado como um único agente (figura 78a). Pode também ser integrado com o treinador principal, funcionando ambos como um único agente (figura 78b). Esta arquitectura é a utilizada pela equipa FC Portugal actualmente. O treinador principal pode também ser incluído no interior dos agentes jogadores (figura 78c). Se o treinador principal apenas precisar de informação do oponente e estatísticas simples do jogo, provenientes do treinador adjunto, então ele pode ser incluído no interior dos agentes jogadores. Nesta arquitectura, os jogadores são os responsáveis pela mudança da táctica da equipa (utilizando todos o mesmo algoritmo). Nesta última arquitectura (figura 78d), o treinador não é um agente autónomo. Esta foi a arquitectura do treinador da equipa FC Portugal em 2000, em que os jogadores utilizaram estatísticas simples do jogo e informação do comportamento do adversário para alterarem a estratégia da equipa.

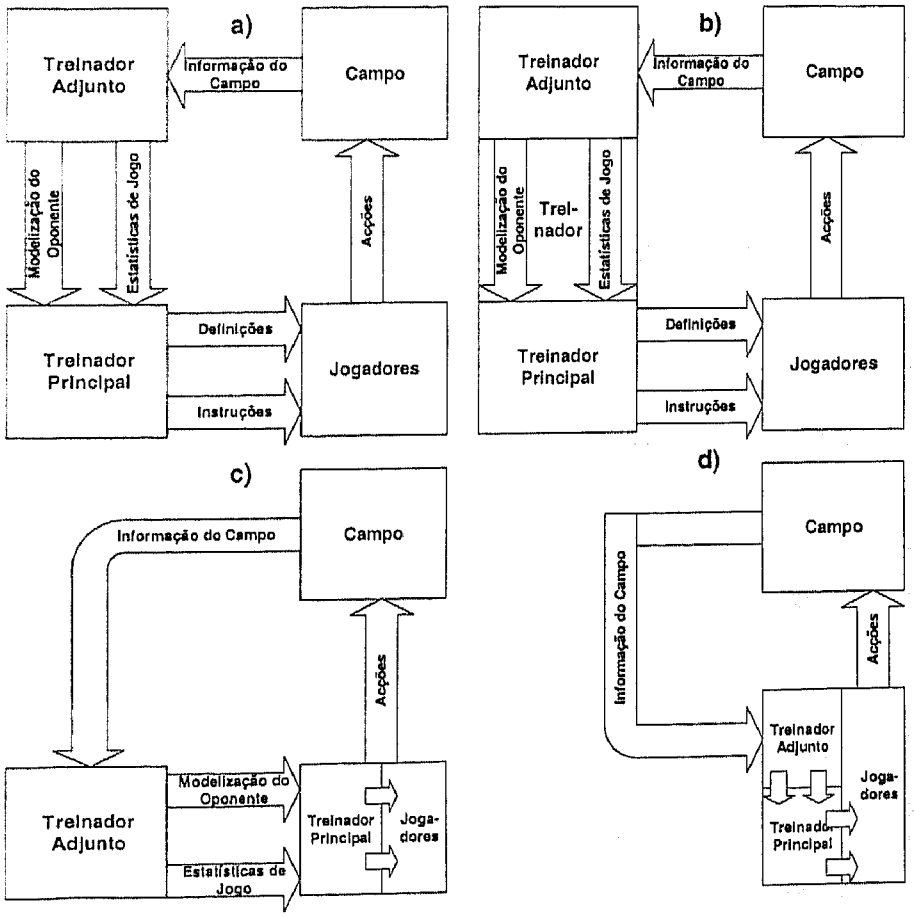

Figura 78: Arquitecturas de treinador possíveis utilizando a linguagem COACH UNILANG

A linguagem COACH UNILANG permite treinar uma equipa a diferentes níveis de abstracção:

- **Nível 1 – Instruções.** Este nível é utilizado para treinar equipas com jogadores inteligentes que estão habituados a jogar juntos. Permite treinar uma equipa a muito alto-nível, mas não é muito flexível porque utiliza conceitos genéricos

fixos de futebol;

- **Nível 2 – Estatísticas e Informação do Oponente.** Este nível é destinado unicamente a auxiliar um treinador a treinar a sua equipa. Esta é a linguagem utilizada pelo treinador adjunto do FC Portugal para comunicar com o treinador principal da equipa[81]. O treinador adjunto está preocupado com o cálculo das estatísticas de jogo como seja a posse de bola por região;
- **Nível 3 – Definições e Instruções.** Este nível de treino é extremamente flexível, uma vez que permite ao treinador definir os conceitos a utilizar na comunicação com os jogadores durante o jogo. Esta comunicação não se limitará a utilizar os conceitos pré-definidos na linguagem permitindo um tipo de treino de muito mais baixo-nível.

A linguagem COACH UNILANG é baseada em diversos conceitos derivados do futebol real. Estes incluem:

- **Regiões do Campo.** Correspondem a áreas do campo com diversas formas possíveis tais como rectângulos, círculos, semi-círculos, quadriláteros, etc;
- **Períodos de Tempo.** Períodos do jogo com uma dada duração e/ou instante de tempo inicial e final;
- **Tácticas.** Permitem a configuração a alto-nível do comportamento global da equipa definindo as características ofensivas e defensivas desta, tais como o ritmo de jogo, estilo de ataque e defesa, pressão colocada no oponente, formações utilizadas para diferentes situações, etc;
- **Formações.** Descrevem distribuições espaciais da equipa no campo;
- **Situações.** Análises de alto-nível do estado do jogo. Situações típicas de um jogo incluem o ataque, defesa, passagem da defesa para o ataque, lançamento, pontapé de baliza, canto, etc.;
- **Tipos de Jogadores.** Defininem o comportamento individual dos jogadores a diversos níveis. Permitem definir o comportamento dos jogadores com a posse de bola e sem a posse de bola.

9.10.2 Requisitos da Linguagem

Na definição da linguagem COACH UNILANG, foram tomados em consideração uma série de requisitos muito semelhantes aos considerados na definição da linguagem SCHEDULING UNILANG (ver secção 7.3). Estes requisitos procuram inserir na

[81] Devido ao facto de as regras da liga de simulação não permitirem a utilização de dois agentes como treinadores, na equipa FC Portugal, optamos por implementar os dois agentes como um único agente.

linguagem um compromisso entre generalidade e simplicidade. A linguagem deve ser suficientemente geral de forma a permitir representar as variantes mais comuns das estratégias empregues por treinadores reais de futebol, mas deve ter também uma sintaxe simples e capacidade para permitir treinar equipas a diferentes níveis de detalhe. Os requisitos principais na definição da linguagem COACH UNILANG foram:

- Independência de detalhes de implementação e estratégias de treinadores e jogadores;
- Simplicidade de expansão com novos conceitos e restrições;
- Linguagens e estratégias de equipas de futebol robótico existentes devem ser facilmente traduzidas para o formato COACH UNILANG;
- O formato deve ser simples de ser lido e interpretado por agentes mas também por humanos;
- O formato deve ser multi-resolução e permitir a representação de estratégias de treinadores a diversos níveis de abstracção;
- A semântica deve ser clara. Todos os conceitos e associações devem ser claros para pessoas com conhecimentos básicos de futebol e futebol robótico;
- Possibilidade de representar qualquer tipo de região do campo ou período de tempo e possibilidade de representar as tácticas, formações, situações ou tipos de jogadores mais comuns;
- Regiões, períodos de tempo, tácticas, formações, situações e tipos de jogadores mais comuns devem ser incluídos por defeito mas deverá ser simples definir outros;
- Deve ser o mais simples e compacta possível;
- Deve ser robusta permitindo validações simples de dados e ignorar erros básicos.

Com base nestes requisitos procedeu-se a uma análise detalhada das estratégias das melhores equipas de futebol robótico simulado de forma a procurar encontrar uma linguagem que as permita representar de forma genérica.

9.10.3 Especificação Formal

A linguagem COACH UNILANG possui quatro tipos de mensagens:

- **Definição.** Permite ao treinador definir ou alterar um determinado conceito (região, período de tempo, táctica, formação, situação ou tipo de jogador);
- **Estatística.** Permite ao treinador (ou treinador adjunto) informar os jogadores acerca de estatísticas de alto-nível do jogo (tais como número de remates, posse de bola, perdas e recuperações de bola, etc.)

- **Modelo do Oponente.** Este tipo de mensagens é utilizado pelo treinador (ou treinador adjunto) para informar os jogadores acerca do comportamento da equipa adversária ou dos seus jogadores individuais (a diversos níveis). Pode também ser utilizado para fornecer informação acerca do comportamento da sua própria equipa ou dos seus próprios jogadores[82];
- **Instruções.** Mensagens com instruções são utilizadas pelo treinador para alterar a táctica da equipa. Este tipo de mensagem permite alterar as diferentes partes que compõem a táctica da equipa (formações utilizadas, ritmo de jogo, estilo de jogo, tipos de jogadores em cada posição, etc.)

A linguagem *COACH UNILANG* foi definida utilizando BNF – BakusNaur form [Naur, 1960]. A definição dos tipos de mensagens utilizadas é a seguinte:

```
<MESSAGE>::=  (<TIME> <ID> <MESSAGE PART> {<MESSAGE PART>})
<MESSAGE PART>::=
          <DEFINITION_MESS> |
          <STATISTICS_MESS> |
          <OPP_MOD_MESS>|
          <INSTRUCTION_MESS>
<DEFINITION_MESS> ::= (definition <ID> <DEFINITION >)
<STATISTICS_MESS> ::= (statistics <ID> <STATISTICS>)
<OPP_MOD_MESS> ::= (opp_mod <ID> <OPP_MOD>)
<INSTRUCTION_MESS> ::= (instruction <ID> <INSTRUCTION>)
```

Foi também construída uma versão simplificada da linguagem *COACH UNILANG* denominada *COACH UNILANG LIGHT*. Esta versão é compatível na parte de baixo nível com a linguagem *Clang*, sendo deste modo útil para equipas que pretendam evoluir da linguagem de baixo nível *Clang* para uma linguagem que permita a representação de tácticas, formações e tipos de jogadores a alto-nível.

9.10.3.1 Definição de Conceitos

As mensagens de definição de conceitos permitem definir novos conceitos de forma a serem utilizados como parte da linguagem no decurso de um jogo. São utilizadas essencialmente antes do jogo começar como uma forma de melhorar a capacidade de comunicação entre o treinador e os jogadores durante o jogo (altura em que a largura de banda disponível será muito menor).

Os conceitos possíveis de serem definidos, incluem regiões, períodos de tempo, tácticas, formações, situações e tipos de jogadores (comportamento dos jogadores com e sem bola). Os conceitos definidos são identificados por um nome para que possam ser referenciados em futuras mensagens do treinador:

---

[82] Isto é especialmente útil no caso de os jogadores não conhecerem bem os seus colegas de equipa. Neste caso, o treinador, antes ou durante o jogo, poderá informar os seus jogadores das capacidades dos outros jogadores que compõem a equipa e das capacidades e fragilidades da equipa como um todo.

```
<DEFINITION>::=
          (region <REGION_NAME> <REGION>) |
          (period <PERIOD_NAME> < PERIOD>) |
          (tactic <TACTIC_NAME> <TACTIC>) |
          (formation <FORMATION_NAME> <FORMATION>) |
          (formation_types <FORMATION_TYPES_NAME> <FORMATION_TYPES>) |
          (situation <SITUATION_NAME> <SITUATION>) |
          (player_type <PLAYER_TYPE_NAME> <PLAYER_TYPE> ) |
          (active_type <ACTIVE_TYPE_NAME> <ACTIVE_TYPE>) |
          (recovery_type <RECOVERY_TYPE_NAME> < RECOVERY_TYPE>)
```

Os nomes definidos são simplesmente cadeias de caracteres (*strings*). Por uma questão de simplicidade, e de forma a manter o formato compacto, não devem exceder os 16 caracteres e não devem incluir espaços.

9.10.3.2 Regiões do Campo

Para treinar uma equipa de futebol é necessário ser capaz de comunicar em termos de regiões do campo. O conceito de região é um dos mais comuns em futebol e os profissionais de futebol (jogadores, treinadores e jornalistas), usualmente comunicam utilizando termos tais como: "a grande-área", "a faixa direita do ataque", "o meio campo ofensivo", "o miolo do terreno", etc. Adicionalmente, grande parte das tarefas individuais e cooperativas dos jogadores durante o jogo vai exigir um raciocínio proeminentemente espacial e as acções a efectuar vão depender em larga medida da região do campo onde o jogador se encontra em cada instante. A partilha de uma definição do campo em termos de regiões por todos os elementos da equipa, é essencial para uma boa coordenação da equipa.

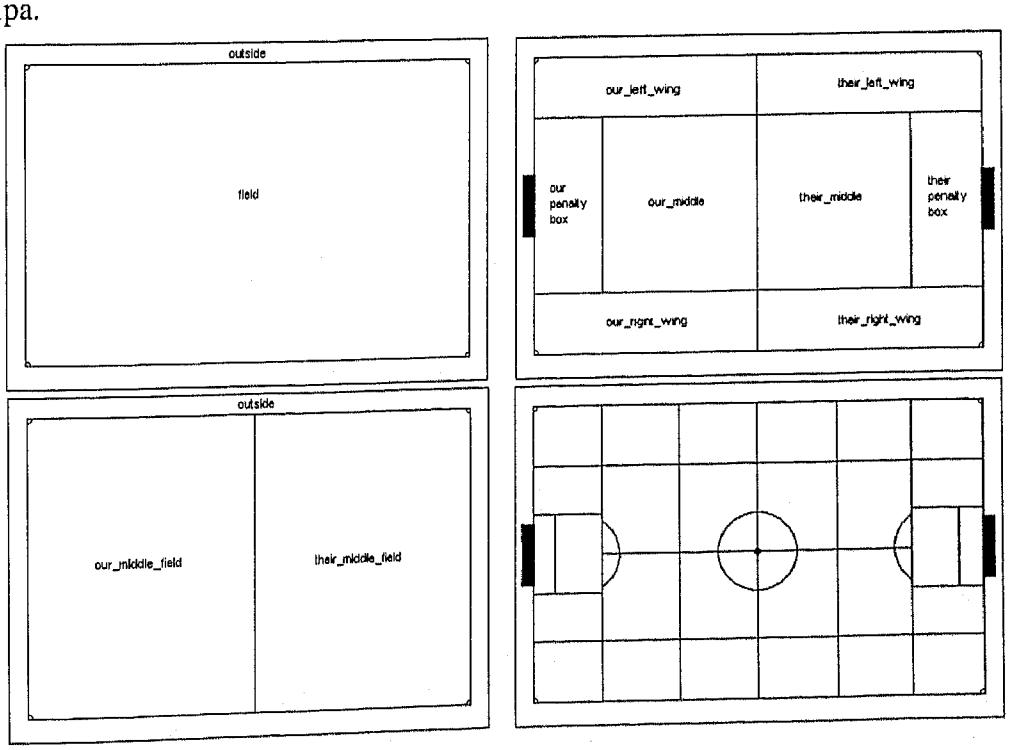

Figura 79: Estrutura Multi-Resolução composta pelas regiões predefinidas na linguagem

COACH UNILANG disponibiliza pré-definidas as regiões mais comuns de um campo de futebol. Disponibiliza também um mecanismo simples para definir novas regiões. As regiões pré-definidas incluídas na linguagem permitem uma descrição do campo em termos de uma estrutura multi-resolução (figura 79).

A definição de novas regiões pode ser efectuada definindo regiões simples (rectângulos, círculos, quadriláteros, etc.), agrupando regiões através da sua união ou intersectando regiões já definidas. As regiões pré-definidas procuram ser o mais simples possível, pelo que unicamente rectângulos foram utilizados na sua definição.

```
<REGION>::= <SIMPLE_REGION> | (<SIMPLE_REGION> {<SIMPLE_REGION>})
<SIMPLE_REGION>::= <PREDEFINED_REGION>|<REGION_NAME>|<REGION_DEFINITION>
<PREDEFINED_REGION>::=
          field | outside | our_middle_field | their_middle_field |
          our_leftwing | our_middle | our_rightwing | our_penalty_box |
          our_leftwing_back | our_leftwing_middle | our_leftwing_front|
          our_rightwing_back | our_rightwing_middle
          our_rightwing_front | our_middle_back | our_middle_front |
          our_middle_back_right | our_middle_back_left| ...
<REGION_DEFINITION> ::= <POINT> | NULL
          (rect <POINT> <POINT>) |
          (tri <POINT> <POINT> <POINT>) |
          (quad <POINT> <POINT> <POINT> <POINT>) |
          (circle <POINT> [real]) |
          (arc <POINT> [real] [real] [real] [real])
<POINT> ::= (point [real] [real])
```

9.10.3.3 Períodos de Tempo

O tempo é muito importante nas análises que um treinador tem de efectuar. Grande parte das análises utiliza conceitos como: "na segunda parte", "no prolongamento", "nos últimos 5 minutos", "desde que alterei a forma de atacar da equipa para x", "desde que efectuei a substituição y", etc.

COACH UNILANG providencia um mecanismo simples de definição de períodos de tempo. Existem diversos períodos de tempo que são conceitos pré-definidos na linguagem. Para além disso o treinador pode definir períodos de tempo para utilizar posteriormente nas suas mensagens ou pode ainda usar directamente intervalos de tempo nas suas mensagens.

```
<PERIOD>::= <PREDEFINED_PERIOD> | <PERIOD_NAME> | <PERIOD_DEFINITION>
<PREDEFINED_PERIOD>::= game | first_half | second_half | extra_time |
          last_1000 | last_500 | last_300 | last_100 | last_50 | last_20
<PERIOD_DEFINITION>::= (<TIME> <TIME>) | <LENGTH>
<TIME>::= [integer]|now
<LENGTH>::= [integer]
```

Os períodos de tempo pré-definidos incluem o jogo completo, a primeira parte, a segunda parte, o prolongamento, os últimos 1000 ciclos, etc. Para definir novos períodos de tempo deve ser definido o seu início e fim ou o seu início e duração. Um identificador especial "*now*" pode ser utilizado para o final do período. A sua utilização, por exemplo *(3000*

now), significa que o período refere-se ao intervalo de tempo desde o ciclo 3000 até ao instante actual. A definição de um período em termos de duração, por exemplo (100) significa que esse período corresponde aos últimos 100 ciclos de jogo.

9.10.3.4 Condições e Situações

Uma parcela considerável dos raciocínios efectuados por um jogador de futebol é baseada numa análise da situação em que se encontra a diversos níveis. Esta situação inclui um vasto conjunto de possíveis condições e análises espaciais. No entanto, de forma a manter a linguagem *COACH UNILANG* simples e compacta, optamos por incluir unicamente um reduzido conjunto de condições adicionado a construtores lógicos simples como forma de definir possíveis situações de jogo.

A definição das situações na linguagem *COACH UNILANG* é baseada no trabalho de Gal Kaminka, Patrick Riley e Timo Stefens na definição das condições a utilizar na linguagem *Clang* [Soccerserver, 2001]. As condições incluídas em *COACH UNILANG* são mais simples do que as da linguagem *Clang* pois pretende-se ser capaz de definir unicamente situações de alto-nível (tais como ataque, defesa, etc.) e não situações de baixo-nível que controlem totalmente o comportamento individual dos jogadores como é objectivo da linguagem *Clang*.

```
<SITUATION>::= <PREDEF_SITUATION|<SITUATION_NAME>|<SITUATION_DEFINITION>
<PREDEF_SITUATION>::= attack | defend | defense_to_attack | attack_to_defense|
          our_opportunity | their_opportunity | our_goalie_kick |
          their_goalie_kick
<SITUATION_DEFINITION>::= <CONDITION>
<CONDITION>::= (true) | (false) |
          (team_in_possession <TEAM><PERIOD>) |
          (player_position <TEAM> <UNUM_SET> <REGION>) |
          (ball_position <REGION>) |
          (ball_velocity <VALUE>) |
          (ball_possession <TEAM> <UNUM_SET>) |
          (play_mode <PLAY_MODE>) |
          (not <CONDITION>) |
          (and <CONDITION> <CONDITION>) |
          (or <CONDITION> <CONDITION>)
```

9.10.3.5 Tácticas

A táctica é provavelmente o conceito de futebol mais complexo. Na definição de uma táctica, diversos outros conceitos complexos são incluídos, como sejam formações, ritmo de jogo, tipos de jogadores, estilo de jogo, etc. Como foi analisado anteriormente, na nossa definição de estratégia, esta é composta por um conjunto de tácticas e em cada instante, num dado jogo, uma dada táctica está activa. Assim, uma táctica deve permitir definir totalmente o comportamento colectivo da equipa num dado instante. De acordo com o decurso do jogo e com o comportamento da equipa adversária, o treinador da equipa pode alterar a táctica em utilização globalmente ou parcialmente.

A definição de táctica na linguagem COACH UNILANG baseia-se em conceitos comuns do futebol tais como formações, tipos de jogadores, mentalidade da equipa, ritmo de jogo, pressão, estilo de jogo ofensivo e defensivo, risco assumido, etc. De forma a incluir estes parâmetros na táctica, a sua definição formal é a seguinte:

```
<TACTIC>::= <PREDEFINED_TACTIC> | <TACTIC_NAME> | <TACTIC_DEFINITION>
<PREDEFINED_TACTIC>::= normal | score_lots |counter_attack | keep_possession |
          attack_all | defend_all
<TACTIC_DEFINITION>::= <MENTALITY> <GAME_PACE> <PRESSURE> <FIELD_USE>
          <PLAYING_STYLE> <RISK_TAKEN> <OFFSIDE_TACTIC> <POS_EXCHANGE_USE>
          <FORMATIONS_USED>
<MENTALITY>::= very_offensive|offensive|normal|defensive|very_defensive
<GAME_PACE>::= sleep | ease_up | normal | speed_up | give_everything
<RISK_TAKEN>::= very_safe | safe | normal | risky | very_risky
<PRESSURE>::= no_pressure| standoff| normal| pressure| high_pressure
<FIELD_USE>::= wings | middle | balanced | right_wing | left_wing
<PLAYING_STYLE>::= (<ATTACKING_STYLE> <DEFENDING_STYLE>)
<ATTACKING_STYLE>::= short_passing | passing | normal | long_ball |
          verylong_ball
<DEFENDING_STYLE>::= zonal_defense | normal | individual_marking
<OFFSIDE_TACTIC>::= (<OFFSIDE_TRAP_USE> <OFFSIDE_TRAP_CAREFUL>)
<OFFSIDE_TRAP_USE>::= <VALUE>
<OFFSIDE_TRAP_CAREFUL>::= <VALUE>
<POS_EXCHANGE_USE> ::= <VALUE>
<FORMATIONS_USED>::= ({(<SITUATION> <FORMATION> <FORMATION_SHAPE>
          <FORMATION_TYPES)})
<FORMATION_SHAPE>::= <FORMATION_WIDTH> <FORMATION_HEIGHT>
<FORMATION_WIDTH>::= <VALUE>
<FORMATION HEIGHT>::= <VALUE>
```

Construir tácticas pré-definidas é uma tarefa extremamente complexa. Não existe nenhuma definição aceite por todos do que é uma táctica de futebol e que permita descrever a táctica de uma dada equipa a determinado momento num jogo. Como exemplo, incluímos diversas tácticas pré-definidas relativamente comuns em futebol na linguagem COACH UNILANG: táctica normal (*normal*), contra-ataque (*counter_attack*), posse de bola (*keep_possession*), ataque (*attack_all*), defesa (*defend_all*) e ataque total (*score_lots*).

A definição completa de uma táctica inclui as formações a utilizar para atacar e defender, os tipos de jogadores a utilizar em cada posição do campo e diversos parâmetros que permitem ajustar o comportamento individual da equipa de forma a seguir uma dada estratégia colectiva. Uma táctica é então composta por:

- **Mentalidade da equipa (muito ofensiva, ofensiva, normal, defensiva, muito defensiva).** Uma equipa com uma mentalidade ofensiva preocupa-se já com a forma de iniciar o ataque quando está a defender, i.e. sem a posse de bola. Este tipo de equipa usualmente defende com menos jogadores do que uma equipa com mentalidade defensiva. Por exemplo, alguns jogadores podem ficar já perto da linha de meio campo enquanto o adversário ataca. Opostamente, uma equipa com mentalidade defensiva, mesmo enquanto ataca mantém grandes preocupações

defensivas. Os seus jogadores procuram sempre permanecer em posições defensivas úteis que evitem contra-ataques rápidos do adversário.

- **Ritmo de jogo (muito rápido, rápido, normal, lento, muito lento).** O ritmo de jogo está relacionado com a velocidade imposta ao jogo pela equipa quer nos movimentos ofensivos quer nos movimentos defensivos. Equipas com um ritmo de jogo elevado (por exemplo equipas Inglesas típicas), utilizam uma alta velocidade para todos os movimentos efectuados. Equipas com um ritmo de jogo lento (por exemplo a selecção do Brasil ou equipas típicas brasileiras) usualmente movimentam a bola lentamente no campo. Usualmente os jogadores deste tipo de equipas recebem a bola, param a bola e só depois decidem o que fazer a seguir. Os jogadores destas equipas só se movimentam rapidamente no campo em situações críticas para a equipa preferindo em outras situações conservar a sua energia.
- **Risco (Muito Seguro, Seguro, Normal, Arriscado, Muito Arriscado).** Define o risco que os jogadores estão dispostos a assumir nas suas acções defensivas e ofensivas. Se uma táctica de alto risco for utilizada, então os jogadores tentarão intercepções arriscadas e passes arriscados. Quando uma equipa está a perder um jogo perto do final, é usual tentar tácticas com alto risco de forma a recuperar rapidamente a bola e chegar com perigo à área contrária também rapidamente. A utilização de um baixo grau de risco provoca que os jogadores utilizem movimentos mais seguros e não arrisquem a execução de acções cujo sucesso não é garantido.
- **Utilização do campo.** A utilização do campo refere-se à utilização das diferentes regiões do campo para o desenvolvimento de movimentos ofensivos. As equipas podem utilizar o campo de forma balanceada ou dar preferência à utilização de uma ou ambas as alas. Podem também utilizar prioritariamente o meio do campo como forma de ataque preferencial. Esta situação é típica contra adversários que cobrem bem as alas do terreno, por exemplo utilizando defesas laterais velozes[83].
- **Estilo de Jogo.** O estilo de jogo decompõe-se em duas partes: o estilo de jogo ofensivo (quando a equipa tem a posse de bola) e defensivo (quando a posse de bola é do adversário). O estilo de jogo define desta forma os estilos de ataque e de defesa utilizado pela equipa.
- **Estilo de Jogo Ofensivo (passes muito curtos, passes curtos, normal, bolas**

[83] Esta conclusão é válida não só no futebol real como também no futebol simulado desde a introdução dos jogadores heterogéneos. Diversas equipas como por exemplo os UvaTrilearn utilizaram os jogadores mais rápidos como extremos ou defesas laterais dificultando imenso a tarefa a equipas que utilizavam prioritariamente as alas apara atacar.

longas, bolas muito longas). O estilo de jogo ofensivo permite definir comportamentos colectivos de ataque distintos. Utilizando novamente como exemplo as equipas da Inglaterra e Brasil, é habitual as equipas Inglesas utilizarem um estilo de jogo com passes longos para as costas dos defesas enquanto as equipas Brasileiras preferem uma progressão mais lenta com fintas e passes curtos.

- **Estilo de Jogo Defensivo.** O estilo de jogo defensivo define o comportamento defensivo da equipa em termos do tipo de marcações utilizadas na zona defensiva;
- **Táctica de Fora de Jogo.** A táctica de fora de jogo está relacionada com as medidas tomadas pelo treinador para apanhar os adversários em fora de jogo e evitar que os jogadores da sua equipa sejam apanhados em fora de jogo. Usualmente a criação de uma armadilha de fora de jogo consiste em criar uma linha composta pelos jogadores mais atrasados da defesa. Os defesas devem manter a linha e avançar rapidamente de forma a procurar colocar os avançados numa posição de fora de jogo.
- **Troca de Posições.** A utilização da troca de posições está relacionada com a tendência que os jogadores terão para trocar de posição com colegas de equipa. Se um elevado grau de trocas de posição for utilizado, os jogadores trocarão de posição com colegas mesmo que o único objectivo de efectuar tal troca seja confundir os adversários. Se um baixo grau de utilização de trocas de posição for utilizado, então os jogadores manterão sempre a mesma posição na equipa.
- **Formações.** Uma das características mais importantes da táctica da equipa está relacionada com as formações a utilizar para as diversas situações de jogo. Por exemplo, uma equipa poderá utilizar uma formação 433 apertada para defender e uma formação larga 235 para atacar. Outras formações poderão ser utilizadas para cantos, pontapés de baliza e outras situações. A linguagem permite ao treinador definir as formações que serão utilizadas em cada situação e a sua forma (em termos de abertura horizontal e vertical).
- **Tipos de Jogadores.** De modo a definir a táctica, é necessário especificar não só as formações utilizadas em cada situação do jogo mas também o comportamento que deve ser assumido por cada jogador em cada posição dessa formação. A definição dos tipos de jogadores permite definir o comportamento individual de cada jogador na formação, atribuindo-lhe um dos comportamentos pré-definidos (*Player Types*). Assim, é possível, por exemplo, definir uma táctica que utiliza um 433 para atacar, cujos jogadores mais avançados da formação são respectivamente um avançado fintador, um avançado rematador e um avançado agressivo.

Embora um conjunto de identificadores seja especificado pela linguagem de forma a ser mais facilmente legível, quase todos os parâmetros permitem utilizar uma escala de valores discretos (*vhi, hi, mid, low, vlow*).

9.10.3.6 Formações

As formações descrevem a distribuição espacial dos jogadores em cada situação. COACH UNILANG inclui predefinidas as formações de futebol mais comuns (433, 442, 343, etc.). Inclui ainda um mecanismo de alto-nível de definição de formações baseado num quadro rectangular imaginário de 7*5 posições. A figura 80 demonstra a utilização deste quadro para definir as formações mais comuns de futebol. As posições por defeito podem também ser definidas sem o auxílio do quadro virtual mas sim, a baixo-nível, utilizando as regiões do campo definidas.

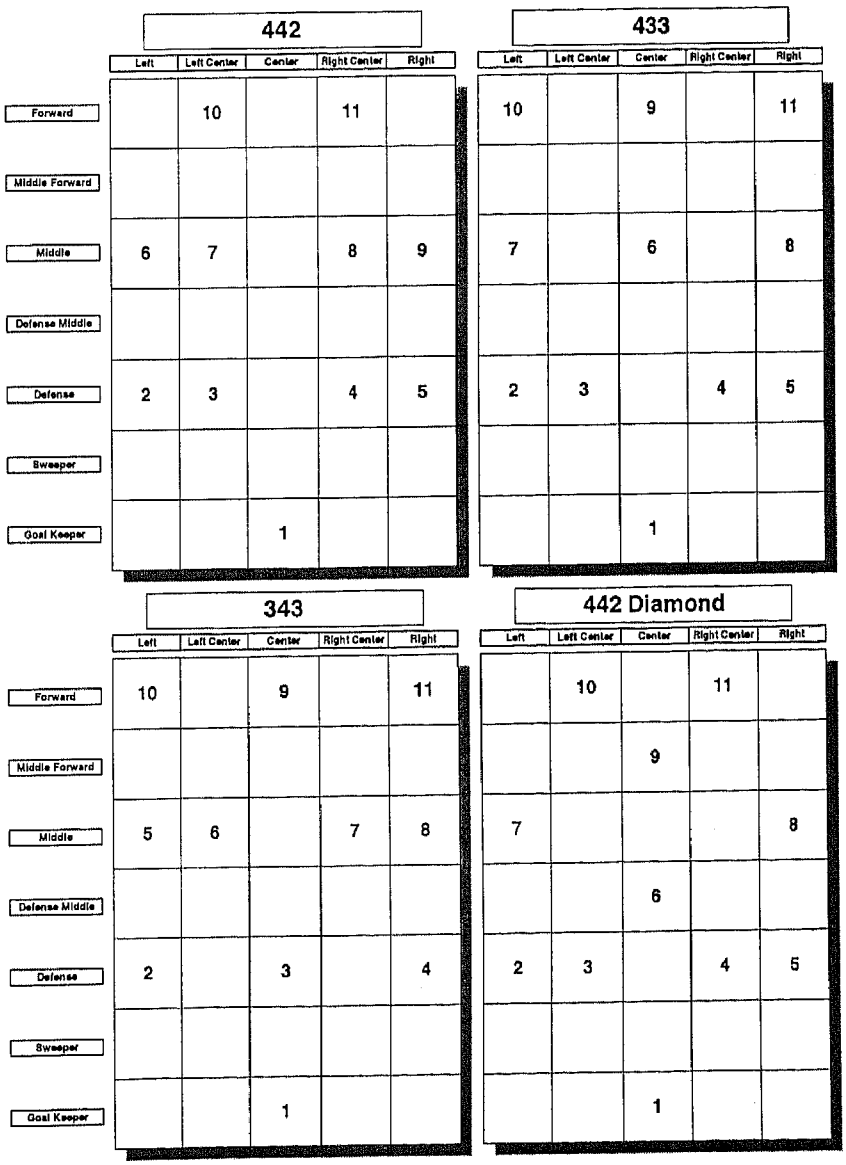

*Figura 80: Definição de Formações utilizando o quadro virtual 7*5*

A definição de uma dada formação, implica a definição, para cada jogador, do número do posicionamento a ocupar por defeito na formação[84] e qual a posição correspondente a esse posicionamento no campo. Esta posição pode ser definida com o auxílio do quadro virtual 7*5 ou utilizando regiões pré-definidas do campo. A definição formal de uma formação é a seguinte:

```
<FORMATION>::= <PREDEF_FORMATION>|<FORMATION_NAME>|<FORMATION_DEFINITION>
<PREDEF_FORMATION>::= 433 | 433att | 442 | 442att | 442diamond | 343 | 4123 |
          352 | 532 | 41212 | 4132 | 424 | 451 | 541 | 3223 | 3232
<FORMATION_DEFINITION>::={(<PLAYER><POSITIONING_NUMBER><PLAYER_POSITIONING>)}
<PLAYER_POSITIONING>::= (<VERTICAL_POSITIONING> <HORIZONTAL_POSITIONING>) |
          (<REGION>)
<VERTICAL_POSITIONING>::= gk | sweeper | defense | defense_middle | middle |
          middle_forward | forward
<HORIZONTAL_POSITIONING>::= left| left_center| center| right_center| right
```

A utilização do número do posicionamento na definição da formação permite descrever a forma de movimentação dos jogadores quando a equipa altera a formação activa. Por exemplo, se a equipa utilizar uma formação 451 para defender e uma formação 433 para atacar, o número do posicionamento permite determinar quais os dois jogadores que passam da linha de meio campo para a linha ofensiva quando a equipa passa para o ataque.

9.10.3.7 Tipos de Jogadores

A definição de um tipo de jogador corresponde a definir o comportamento desse jogador quando possui a bola ou quando não é o detentor da posse de bola. A definição dos tipos de jogadores começa pela definição das possíveis acções[85] de posse de bola e de recuperação de bola.

As acções de posse de bola (*action*) incluem as acções definidas anteriormente, ou seja: chutar à baliza (*shoot*), passar a bola directamente a um colega (*pass*) ou para a desmarcação (*forward*), driblar com a bola (*dribble*), correr com a bola (*run*), segurar a bola (*hold*), aliviar a bola (*clear*). É ainda incluída uma acção (*any*) que corresponde a qualquer das acções anteriores.

As acções de recuperação de bola correspondem às acções típicas, sem bola, dos jogadores de futebol, definidas previamente: intercepção (*interception*), corte (*tackle*), marcação de uma linha de passe (*mark_pass_line*), marcação de um jogador (*mark_player*), aproximação da bola (*approach_ball*) e movimento estratégico no campo respeitando a formação da equipa (*strategical_move*). É ainda incluída uma acção (*any*) que corresponde a qualquer das acções anteriores.

---

[84] Eventuais trocas de posicionamentos podem ser permitidas ou mesmo aconselhadas pela táctica.

[85] Estas acções são usualmente compostas por um conjunto de comandos básicos (dash, turn, kick, etc.)

Para além da definição das acções de posse e recuperação de bola, a linguagem define ainda os possíveis resultados destas acções. No caso das acções de posse de bola, o resultado pode ser o sucesso (*success*), a intercepção do adversário (*interception*), a perda de bola para um oponente no decurso da execução da acção (*stolen*), bola fora (*out*), a falha na execução da acção (*fail*), a intercepção por parte de um ou outro guarda-redes (*opp_goalie_catch* ou *our_goalie_catch*) ou qualquer destas acções (*any*). No caso das acções de recuperação de bola o resultado pode ainda ser a intercepção ou o corte por parte de um outro jogador (*stolen_other* e *interception_other*) enquanto o jogador considerado estava a efectuar a acção de recuperação de bola.

A execução de uma dada acção implica também a especificação dos parâmetros a serem utilizados para essa acção. Tendo em conta que um dos requisitos da linguagem é o ser genérica e permitir treinar diferentes equipas, e que existe um leque alargado de acções que podem ser executadas pelos agentes, consideram-se unicamente três parâmetros para as acções[86]: distância (*distance*), potência (*power*) e segurança (*safety*).

```
<ACTION>::= shoot | pass | forward | dribble | run | hold | clear | any
<RECOVERY>::= interception | tackle |mark_pass_line | mark_player |
          goto_ball | strategical_move | any
<ACTION_RESULT>::=  success | out | interception | stolen | fail |
          opp_goalie_catch | our_goalie_catch | any
<REC_RESULT>::=  <ACTION_RESULT> | stolen_other | interception_other
<ACTION_PARAMETERS>::= (<DISTANCE> <POWER> <SAFETY>)
<RECOVERY_PARAMETERS>::= (<DISTANCE> <POWER> <SAFETY>)
```

Para definir um tipo de jogador (*Player_Type*), o treinador deve definir o comportamento desse jogador com (*Active_Type*) e sem (*Recovery_Type*) bola. Alguns comportamentos simples de posse e recuperação de bola encontram-se pré-definidos na linguagem. A definição de novos tipos é bastante simples. No caso dos tipos activos (posse de bola), esta definição baseia-se em definir a tendência (valor numa escala qualitativa) para executar cada possível acção de posse de bola (chutar, passar, driblar, etc.) com início numa dada região do campo (*Region_From*) e destino numa dada região do campo (*Region_To*) e determinados parâmetros de execução (distância, potência e segurança). No caso dos tipos de recuperação de bola, estes são definidos de forma similar. Como principal diferença destaca-se que para as acções de recuperação de bola não é considerada a região de destino da acção.

Utilizando tipos de jogadores previamente definidos é possível compôr formações, atribuindo a cada posição da formação um dado tipo de jogador. Uma formação completa de tipos de jogador é composta pela definição dos 11 tipos de jogador que compõe uma dada equipa.

---

[86] Com perda de alguma generalidade e com um aumento significativo da complexidade poderiam ser utilizados os parâmetros utilizados pela equipa FC Portugal para a execução e avaliação de cada uma das acções de recuperação e posse de bola.

```
<FORM_TYPES)::= {(<PLAYER> <POSITIONING_NUMBER> <PLAYER_TYPE> )}
<PLAYER_TYPE>::= <ACTIVE_TYPE> <RECOVERY_TYPE>
<ACTIVE_TYPE>::= <PREDEF_ACTIVETYPE>|<ACTIVE_TYPE_NAME>|<ACTIVETYPE_DEF>
<PREDEF_ACTIVETYPE>::= basic | normal | shooter | dribbler | passer
<ACTIVETYPE_DEF>::= (((<ACTION> <REGION_FROM> <REGION_TO>
          <ACTION_PARAMETERS> <VALUE>)))
<RECOVERY_TYPE>::= <PREDEF_RECTYPE> | <RECOVERY_TYPE_NAME> | <RECTYPE_DEF>
<PREDEF_RECTYPE>::= normal | positional | marker | aggressive
<RECTYPE_DEF>::= (((<RECOVERY> <REGION> <RECOVERY_PARAMETERS> <VALUE>)))
```

9.10.4 A Linguagem do Treinador Adjunto

O treinador adjunto é responsável por recolher informação estatística do jogo e informação relativa ao comportamento colectivo da equipa adversária e fornecer esta informação, devidamente tratada e formatada, ao treinador principal ou aos jogadores da equipa.

9.10.4.1 Informação sobre o Oponente

A informação de modelização do oponente está relacionada com as características detectadas pelo treinador relativas à forma de jogar da equipa ou jogadores adversários [Reis e Lau, 2001c]. Pode também incluir características da própria equipa do treinador (caso este esteja a treinar uma equipa desconhecida).

Na linguagem COACH UNILANG esta informação é dividida em quatro níveis de abstracção [Reis e Lau, 2001c]:

- **Nível Social.** Informação sobre o comportamento colectivo da equipa;
- **Nível Individual.** Decisão individual dos jogadores e guarda-redes;
- **Capacidades de Baixo-nível.** Habilidade na execução das acções sem considerar a adequação da decisão de qual a acção a executar;
- **Características Físicas.** Para os jogadores heterogéneos: velocidade, arranque, inércia, tamanho, etc.

A definição da informação de modelização do oponente pode ser dos quatro tipos descritos e a sua representação formal é a seguinte [Reis e Lau, 2001c]:

```
<OPP_MOD> ::=
          <OPP_TEAM_LEVEL> |
          <OPP_INDIVIDUAL_LEVEL> |
          <OPP_LOW_LEVEL> |
          <OPP_PHYSICAL_LEVEL>
```

No nível social pode ser descrita a qualidade global da equipa adversária (ou da sua própria equipa) e a qualidade numa dada situação específica (defesa, ataque, cantos, etc.). Pode também ser indicada a táctica mais próxima da táctica utilizada pela equipa que pode incluir, para além dos parâmetros tácticos, as formações utilizadas pela equipa para cada situação. A informação é dada na escala qualitativa referida anteriormente *(very_low ... very_high)*

```
<OPP_TEAM_LEVEL>::=
          (opp_team_quality <TEAM> <PERIOD> <VALUE>) |
          (opp_team_situation_quality <TEAM> <SITUATION> <PERIOD> <VALUE>) |
          (opp_team_tactic <TEAM> <PERIOD> <TACTIC>)
          (opp_team_formation <TEAM> <PERIOD> <SITUATION> <FORMATION>)
```

A informação relativa aos jogadores individuais inclui a qualidade global dos jogadores, a sua capacidade para trocar de posicionamento com colegas, capacidade de pressionar oponentes, capacidade de reacção (a eventos no campo) e agressividade. Inclui também informação relativa aos mecanismos de decisão individual dos jogadores com e sem bola e aos mecanismos de decisão individual do guarda-redes: distância usual à baliza (numa escala qualitativa), qualidade de reposição de bola, qualidade de intercepção, qualidade de defesa (de chutos à baliza), qualidade de intercepção de cruzamentos, etc.

```
<OPP_INDIVIDUAL_LEVEL> ::=
          (opp_individual <OPP_INDIVIDUAL_INFO_TYPE> <TEAM> <PLAYER> <PERIOD>
             <VALUE>) |
          (opp_action_quality <ACTION> <TEAM> <PLAYER> <PERIOD> <VALUE>) |
          (opp_recovery_quality <RECOVERY> <TEAM> <PLAYER> <PERIOD> <VALUE>) |
          (opp_goalie <OPP_GOALIE_INFO_TYPE> <TEAM> <PERIOD> <VALUE>)
<OPP_INDIVIDUAL_INFO_TYPE>::= player_overall_quality |
          player_pos_changing_level | player_pressing_capability |
          player_reaction_capability | player_agressivity
<OPP_GOALIE_INFO_TYPE>::= goalie_distance_to_goal | goalie_reposition_quality|
          goalie_interception_quality | goalie_defense_quality |
          goalie_leaves_goal_toward_player | goalie_leaves_area |
          goalie_intercept_crosses
```

A informação relativa às capacidades de baixo-nível inclui a indicação (em termos qualitativos) da qualidade de movimentação, intercepção, chuto, drible e visão dos jogadores:

```
<OPP_LOW_LEVEL>::=
          (opp_low <OPP_LOW_INFO_TYPE> <TEAM> <PLAYER> <PERIOD> <VALUE>)
<OPP_LOW_INFO_TYPE>::= player_moving_quality | player_dribbling_quality |
          player_kick_quality | player_dribbling_quality |
          player_vision_quality
```

A informação física inclui a indicação do tipo de jogador heterogéneo ou tipo de visão utilizado por um dado jogador. Inclui também os parâmetros individuais relacionados com os jogadores heterogéneos:

```
<OPP_PHYSICAL_LEVEL>::=
          (opp_physical <OPP_PHYSICAL_INFO_TYPE> <TEAM> <PLAYER> <PERIOD>
             <VALUE>) |
          (player_het_type <TEAM> <PLAYER> <PERIOD> <HET_TYPE>) |
          (player_vision_type <TEAM> <PLAYER> <PERIOD> <VIEW_TYPE>)
<OPP_PHYSICAL_INFO_TYPE>::= player_physical_condition | player_speed |
          player_recovery | player_decay | player_inertia |
          player_acceleration | player_size | player_kickable_margin |
          player_kick_accuracy
```

Utilizando este formato é possível especificar a todos os níveis (desde o social até ao físico) o comportamento de uma dada equipa e dos seus jogadores.

9.10.4.2 Estatísticas de Jogo

A informação estatística do jogo é muito importante para treinar uma equipa de futebol (robótico). Esta informação contrariamente à informação de modelação do oponente não está directamente relacionada com o comportamento do oponente mas sim com o curso do jogo a diversos níveis. A informação estatística incluída em COACH UNILANG baseia-se na informação utilizada por treinadores reais e na informação disponibilizada por sistemas computorizados de análise de jogos de futebol [Reis e Lau, 2001c] como o *Match Analysis* [Match Analysis, 2001]. A definição da informação estatística em COACH UNILANG é [Reis e Lau, 2001c]:

```
<GAME_PLAYMODE> ::= playon | corner | offside | kickin | kickoff | free_kick |
          goalie_free_kick | goal_kick | any
<STATISTICS> ::= (clear) | (game_time <TIME>) |
          (game_result <PERIOD> [integer] [integer]) |
          (stopped_time <PERIOD> <TIME>) |
          (game_occurence <GAME_PLAYMODE> <REGION> <TEAM> <PERIOD> <COUNT>) |
          (action <ACTION> <REGION_FROM> <REGION_TO> <TEAM> <PLAYER> <PERIOD>
             <ACTION_RESULT> <COUNT>) |
          (recovery <RECOVERY> <REGION> <TEAM> <PLAYER> <PERIOD> <INT_RESULT>
             <COUNT>) |
          (ball_possession <REGION> <TEAM> <PLAYER> <PERIOD> <COUNT>) |
          (player_position <REGION> <TEAM> <PLAYER> <PERIOD> <COUNT>) |
          (attack <REGION_FROM> <REGION_TO> <TEAM> <PERIOD> <NPASSES>
             <ACTION_RESULT> <COUNT>) |
          (assist <REGION_FROM> <REGION_TO> <TEAM> <PLAYER> <PLAYER_TO>
             <PERIOD> <ACTION> <ACTION_RESULT> <COUNT>) |
          (ball_losses <ACTION> <REGION > <TEAM> <PLAYER> <PERIOD> <COUNT>) |
          (ball_recoveries <RECOVERY> <REGION > <TEAM> <PLAYER> <PERIOD>
             <COUNT>) |
          (action_to_player <ACTION> <REGION_FROM> <REGION_TO> <TEAM> <PLAYER>
             <PLAYER_TO> <PERIOD> <ACTION_RESULT> <COUNT>)
```

A informação estatística é muito importante para decidir a táctica a aplicar em cada momento. A maioria das equipas do RoboCup utiliza no entanto unicamente o tempo e o resultado do jogo na decisão de qual a táctica a utilizar. No entanto, estes dois itens não são suficientes para decidir, dados os objectivos da equipa para o jogo, qual a táctica a utilizar. COACH UNILANG inclui a possibilidade de transferir entre o treinador adjunto e o treinador ou de qualquer destes agentes para os jogadores, a seguinte informação estatística, calculada por região do campo, equipa, jogador e período de tempo:

- **Tempo de Jogo** (***"Game Time"***). Tempo total de jogo;
- **Resultado do Jogo** (***"Game Result"***). Resultado do jogo (golos marcados por cada equipa) num dado período;
- **Tempo de Jogo Parado** (***"Stopped Game Time"***). Tempo total em ciclos que o jogo esteve parado;
- **Ocorrências do Jogo** (***"Game Occurrences"***). Ocorrências de situações tais como cantos, lançamentos ou foras de jogo, sua contagem e tempo total. Por exemplo *(game_occurrence offside their_penalty_box our first_half 3)*, indica que 3

foras de jogo ocorreram na primeira parte do jogo no interior da grande-área adversária;

- **Número de vezes que cada acção de posse de bola foi executada e resultados (*"Action"*).** Indica, por região, jogador e período, o número de acções com bola de cada tipo executadas e o seu respectivo resultado. Por exemplo: *(action pass field field our any game success 20)* indica que a nossa equipa executou 20 passes com sucesso durante todo o jogo;
- **Número de vezes que cada acção de recuperação de bola foi executada e resultados (*"Recovery"*).** Indica, por região, jogador e período, o número de acções sem bola de cada tipo executadas e o seu respectivo resultado. Por exemplo: *(recovery interception their_middle_field our 10 last_300 fail 4)* indica que o jogador 10 da nossa equipa falhou 4 intercepções durante os últimos 300 ciclos de jogo;
- **Posse de Bola (*"Ball Possession*).** Estatísticas de posse bola (em termos de número de ciclos) por equipa e jogador, em cada região e período. A frase *(ball_possession their_left_wing our any game 250)* indica que a nossa equipa teve a posse de bola durante 250 ciclos na região *offensive_left_wing* durante o jogo;
- **Posições dos Jogadores (*"Players Positions"*).** Indica o número de ciclos que cada jogador esteve em cada região do campo em cada período.
- **Número de Ataques por Tipo e Resultado (*"Attack"*).** Fornece a informação relativa aos ataques de cada equipa, suas regiões inicias e finais e tipo (em termos do número de passes entre jogadores realizados) e o seu resultado final. Por exemplo: *(attack field field our game 3 opp_goalie_catch 4),* indica que a equipa realizou 4 ataques no jogo com apenas dois passes cada que resultaram em defesas do guarda-redes adversário;
- **Assistências (*"Assist"*).** Assistências (i.e. passes para remates à baliza) realizadas por cada equipa e jogador, região de partida, região de destino e período. É ainda permitido incluir a acção que resultou na assistência e o resultado final da acção de remate à baliza. Por exemplo: *(assist their_left_wing their_penalty_box any any game pass goal 3)* indica que foram realizadas três assistências para remates que resultaram em golo, utilizando passes a partir da ala esquerda do ataque da equipa para o centro da grande-área.
- **Perdas de Bola (*"Ball Losses"*).** Perdas do controlo da bola (para o adversário) por equipa, jogador, região e período de tempo. Pode também ser incluída a acção executada no momento da perda da bola. A frase *(ball_losses pass their_middle_field our any last_1000 10) indica* que 10 bolas foram perdidas no meio campo do adversário, ao realizar passes nos últimos 1000 ciclos.
- **Recuperações de Bola (*"Ball Recoveries"*).** Informação relativa à forma como

cada recuperação de bola foi executada pelas equipas. Permite incluir informação relativa à forma como cada recuperação de bola foi executada por cada equipa. A frase *(ball_recoveries tackle our_middle opponent any game 10)* indica que a bola foi recuperada 10 vezes pelo adversário no nosso meio campo, executando *tackles*, durante o jogo.

- **Circulação de bola ("*Action To Player*").** Permite indicar as acções executadas por cada jogador e para cada jogador, por região e período e os seus resultados. Por exemplo: *(action_to_player pass field field our 8 9 second_half successful 10)* indica que o jogador 8 fez 10 passes bem sucedidos para o jogador 9 durante a segunda parte do jogo.

O cálculo desta informação estatística não é no entanto trivial mesmo para um agente com visão global e sem erros como é o caso do treinador. Este cálculo implica transformar as acções de baixo nível executadas pelos jogadores (*dashs*, *turns*, *kicks* e *catches*) e a posição e velocidade da bola, em acções de alto-nível tais como intercepções ou passes.

9.10.5 Instruções do Treinador

O treinador pode enviar diversas instruções, nomeadamente tácticas, formações e comportamentos individuais de jogadores.

```
<INSTRUCTION>::=
            (change_tactic <TACTIC>)|
            (change_mentality <MENTALITY>)|
            (change_game_pace <GAME_PACE>)|
            (change_risk_taken <RISK_TAKEN>)|
            (change_field_use <FIELD_USE>)|
            (change_playing_style <PLAYING_STYLE>)|
            (change_pressure <PRESSURE>)|
            (change_offside_tactic<OFFSIDE_TACTIC>)|
            (change_pos_exchange_use <POS_EXCHANGE_USE>)|
            (change_formation <SITUATION_NAME> <FORMATION> <FORMATION_SHAPE>)|
            (change_formation_pos <FORMATION_NAME> <POSITIONING_NUMBER>
               <PLAYER_POSITIONING> <PLAYER_TYPE>)|
            (change_player_type <PLAYER_TYPE_NAME> <ACTIVETYPE_DEF>
               <RECTYPE_DEF>)
```

A coordenação por controlo parcialmente hierárquico baseou-se na criação de um treinador *online* capaz de enviar instruções aos jogadores da equipa.

9.10.6 A Linguagem *COACH UNILANG LIGHT*

De forma a permitir uma utilização mais alargada da linguagem COACH UNILANG foi definida uma versão básica da linguagem designada COACH UNILANG LIGHT. Esta versão inclui unicamente as características fundamentais da linguagem COACH UNILANG e é compatível nas condições de baixo-nível com a linguagem de baixo nível CLang. A sua definição completa é a seguinte:

```
<MESSAGE>::= <ADVICE_MSG> | <DEFINE_MSG> | <FREEFORM_MSG>
<ACTION_TOKEN>::= mp | md | mc | ms | mf | mr | mh | ma |
          ri | rt | rl | rp | rb | rm | rf | ra
<ACTION>::= (<ACTION_TOKEN> <REGION> <UNUM_SET>)
<ACTION_LIST>::= {<ACTION>}
<MODE>::= highdo | do | neutral | dont | highdont
<DIRECTIVE>::= (<MODE> <UNUM_SET> <ACTION_LIST>)
<DIRECTIVE_LIST>::= {<DIRECTIVE>}
<RULE>::= (<CONDITION> <DIRECTIVE_LIST>) | <RULE_NAME>
<PLAYER_ROLE> ::= {<RULE>}
<FORM_TOKEN>::= (<POS_NUM> <UNUM> <REGION> <PLAYER_ROLE>) | <FORM_TOKEN_NAME>
<FORMATION>::= {<FORM_TOKEN>}
<TACTIC_TOKEN>::= (<CONDITION> <FORMATION>) | <TACTIC_TOKEN_NAME>
<TACTIC>::= {<TACTIC_TOKEN>}
<TOKEN>::= <RULE> | <PLAYER_ROLE> | <FORMATION> | <TACTIC> | clear
<ADVICE_TOKEN>::= (<ID> <UNUM_SET> <TOKEN>)
<ADVICE_LIST>::= {<ADVICE_TOKEN>}
<ADVICE_MSG>::= (advice <ADVICE_LIST>)
<DEFINE_MSG>::= (define <DEFINE_TOKEN_LIST>)
<DEFINE_TOKEN>::= (definec <CONDITION_NAME> <CONDITION>)
          | (definer <REGION_NAME> <REGION>)
          | (defined <DIRECTIVE_NAME> <DIRECTIVE>)
          | (defineu <RULE_NAME> <RULE>)
          | (definep <PLAYER_ROLE_NAME> <PLAYER_ROLE>)
          | (definef <FORMATION_NAME> <FORMATION>)
          | (definet <TACTIC_NAME> <TACTIC>)
<DEFINE_TOKEN_LIST>::= {<DEFINE_TOKEN>}
<CONDITION>::= (true) | (false)
           | (ppos <TEAM> <UNUM_SET> <REGION>) | (bpos <REGION>)
           | (bowner <TEAM> <UNUM_SET>) | (pmode <PLAY_MODE>)
           | (and <CONDITION_LIST>) | (or <CONDITION_LIST>) | (not <CONDITION>)
           | <NAME>
<CONDITION_LIST>::= {<CONDITION>}
<REGION>::= <POINT> | (null) | (rec <POINT> <POINT>)    #rectangle
           | (arc <POINT> [real] [real] [real] [real]) | (reg <REGION_LIST>)
           | "STRING"
<POINT>::= (pt [real] [real]) | (add <POINT> <POINT>) | (mult <POINT> <POINT>)
           | (pt ball) | (pt <TEAM> <UNUM>)
<PLAY_MODE> ::= bko | time_over | play_on | ko_our | ko_opp | ki_out |
           ki_opp | fk_our | fk_opp| ck_our | ck_opp | gk_our | gk_opp |
           gc_our | gc_opp | ag_our | ag_opp
<TEAM>::= our | opp
<POS_NUM>::= [1..11]
<UNUM>::= [1..11]
<UNUM_LIST>::= {<UNUM>} | e
<NAME>::= "[string]"
<FREEFORM_MSG>::= (freeform "[string]")
```

Esta linguagem foi utilizada como base para a reformulação da linguagem CLang que foi utilizada na competição de treinadores realizada em Fukuoka no RoboCup 2002.

9.10.7 Conclusões

A linguagem COACH UNILANG foi a primeira tentativa para criar uma linguagem que permita o treino a alto-nível de uma equipa de futebol robótico. Claro que ainda existem diversos tópicos abertos e a sintaxe não permite representar tudo o que é necessário para

definir todas as tácticas de futebol. Os tipos de jogadores não são completos pois diferentes equipas podem ter acções específicas de posse de bola e recuperação de bola que não estão incluídas na linguagem. A linguagem inclui as acções mais comuns do futebol simulado/real com e sem bola e arquitecturas específicas de jogadores não devem ser incluídas numa linguagem standard com esta.

Pode ser argumentado que a definição de tácticas e formações é de muito alto-nível. No entanto a linguagem COACH UNILANG permite a definição de tácticas e formações a qualquer nível de detalhe desejado pelo treinador. O treinador pode utilizar unicamente instruções de alto-nível mas pode também definir o posicionamento e comportamento de cada jogador com elevado detalhe.

Outra crítica que pode ser realizada a esta linguagem é que é demasiada complexa. No entanto o futebol é um jogo extremamente complexo e o SoccerServer captura muito bem essa complexidade. Não existe qualquer interesse em definir uma linguagem que permite treinar unicamente equipas com mecanismos de decisão muito simples que não sejam capazes de jogar como equipas reais de futebol.

A linguagem está actualmente em extensão de forma a incluir planos flexíveis, marcadores de livres e cantos, instruções para o guarda-redes, estratégias de visão e estratégias de comunicação.

9.11 Coordenação por Controle Parcialmente Hierárquico

9.11.1 Definição de um Treinador on-line

A coordenação por controle parcialmente hierárquico implica a construção de um treinador para a equipa FC Portugal, capaz de comunicar com os jogadores utilizando a linguagem COACH UNILANG [Reis e Lau, 2003b]. A arquitectura definida para o agente treinador encontra-se representada na figura 81.

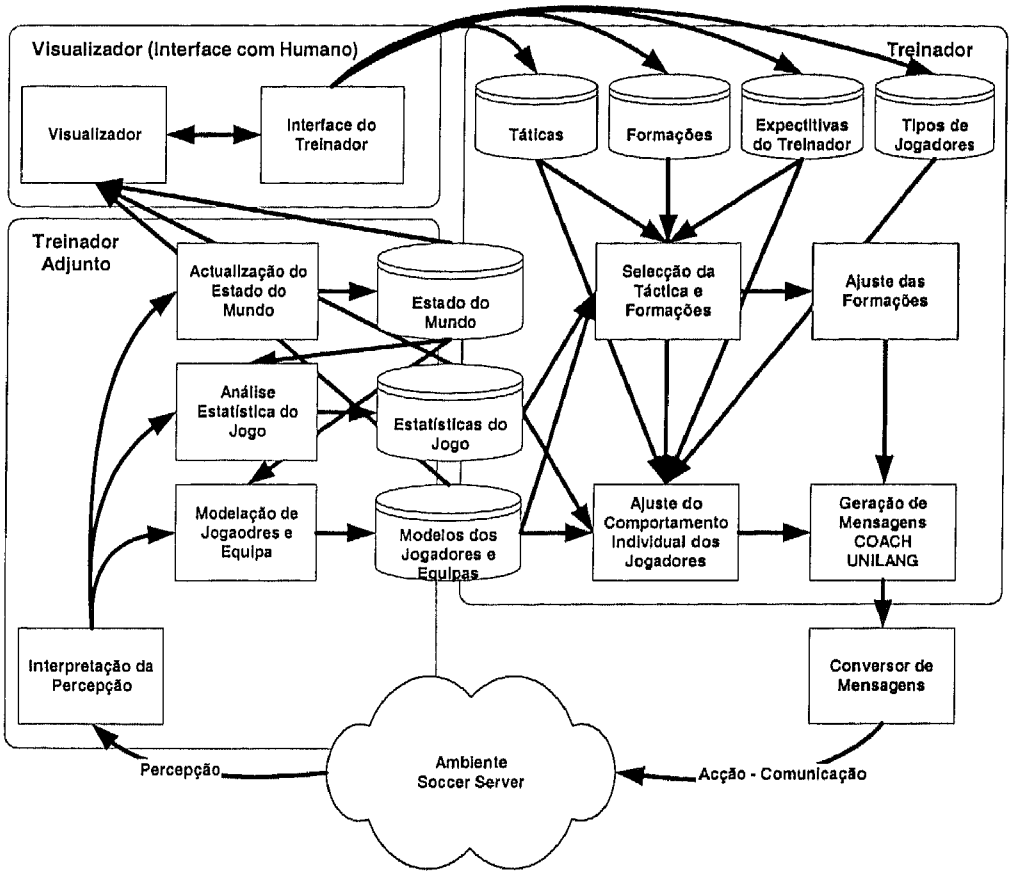

Figura 81: Arquitectura do Treinador da equipa FC Portugal

O Treinador possui uma interface gráfica que permite a um utilizador humano instanciar as suas estruturas de conhecimento e visualizar o seu raciocínio e informação de que dispõe no decurso de cada jogo. As estruturas de conhecimento do treinador são semelhantes às dos agentes jogadores: táctica, formações e papéis (tipos de jogadores). O treinador não possui informação sobre estratégias de comunicação ou visão, sendo estas deixadas ao critério dos jogadores. Possui no entanto informação relativa às expectativas para o jogo contendo o resultado final esperado (golos marcados e sofridos) para o jogo e informações diversas sobre o comportamento esperado do oponente.

A informação que o treinador utiliza para o seu processo de decisão é fornecida por um módulo autónomo designado por treinador assistente. Este módulo interpreta a informação de percepção do treinador e as comunicações efectuados pelos jogadores proveniente do simulador. Utilizando esta informação mantém actualizado um estado do mundo semelhante ao dos agentes jogadores e a informação relativa ao comportamento do adversário e estatísticas de jogo. Esta informação é "enviada" (partilhada pois ambos os agentes são implementados como um único agente) ao agente treinador que a utiliza no seu processo de decisão. A informação é também disponibilizada graficamente utilizando a interface com o utilizador.

Utilizando as suas estruturas de conhecimento e a informação que lhe é "enviada" pelo treinador assistente, o treinador principal decide qual a táctica a utilizar em cada instante de jogo e realiza ajustes às formações e tipos de jogadores utilizados pela táctica e aos parâmetros globais da mesma.

O treinador inclui ainda um gerador de mensagens utilizando a sintaxe da linguagem COACH UNILANG [Reis e Lau, 2002] e um conversor de mensagens que, caso seja activado, permite a simplificação destas mensagens e conversão para outras linguagens de treinador mais simples como a linguagem CLang.

9.11.2 Utilização das Instruções do Treinador nos Jogadores

Os jogadores da equipa FC Portugal podem usar as instruções do treinador de três modos distintos:

- **Instruções Interpretadas como Ordens:** As instruções provenientes do treinador são interpretadas como ordens absolutas e substituem as estruturas locais de conhecimento dos jogadores;
- **Instruções Interpretadas como Conselhos:** As instruções provenientes do treinador são fundidas com a base de conhecimento local dos agentes sendo-lhes atribuído um determinado grau de confiança;
- **Instruções Ignoradas:** As instruções provenientes do treinador são ignoradas .

A implementação prática é realizada associando ao treinador um determinado factor de confiança no início do jogo. Quando a equipa FC Portugal é treinada pelo seu próprio treinador, os jogadores são instanciados com uma confiança absoluta de 100% no treinador e como tal as instruções deste serão interpretadas como ordens. Por exemplo, se o treinador solicitar a alteração para uma outra táctica pré-definida, todos os jogadores seguirão de imediato as suas instruções e passarão a utilizar os parâmetros tácticos, formações e tipos de jogadores definidos nessa táctica [Reis e Lau, 2003b].

No caso do factor de confiança no treinador ser nulo, os jogadores ignoram qualquer ordem fornecida pelo treinador. No caso de um qualquer outro factor de confiança, as instruções do treinador relativas ao comportamento colectivo (tácticas e formações) são aceites desde que a confiança no treinador exceda um factor mínimo pré-definido. As instruções sobre o comportamento individual são interpretadas como conselhos e adicionados à base de conhecimento com uma confiança idêntica à confiança atribuída ao treinador.

9.12 Resultados em Competições RoboCup

Nesta secção apresentam-se os resultados obtidos pela equipa FC Portugal nos dois campeonatos da Europa e nos dois campeonatos do mundo em que participou utilizando as metodologias de coordenação descritas nesta dissertação. Destes resultados destaca-se que a FC Portugal venceu os dois campeonatos da Europa e o campeonato do mundo 2000 e classificou-se em terceiro lugar no campeonato do mundo em 2001 (embora obtendo um *goal-average* de 150-5 neste campeonato). Em todas as competições o FC Portugal obteve o melhor *goal-average*, marcando no total das 4 competições 387 golos, sofrendo unicamente 9[87].

9.12.1 Avaliação dos Resultados Obtidos

O resultado final de um jogo de futebol robótico, embora seja uma medida significativa, não pode ser utilizado como única medida do sucesso de uma determinada equipa ou metodologia específica. Embora o resultado (em termos de golos marcados e sofridos) seja muito importante, nomeadamente em competições oficiais, diversas outras medidas são também importantes, incluindo a posse de bola por região do campo, o número de remates à baliza e as oportunidades de golo. As medidas da qualidade do jogo da equipa utilizadas na avaliação realizada no âmbito desta dissertação foram:

- **Resultado – Golos Marcados (GM-GS):** Golos marcados e sofridos por cada uma das equipas em cada jogo;
- **Remates à Baliza (RE-RS):** Remates efectuados e sofridos em cada jogo. No âmbito desta avaliação, foi considerado um remate um chuto efectuado por um jogador da equipa atacante (ou em conjunto por um jogador da equipa atacante e outro jogador), na direcção aproximada da baliza (cuja trajectória intercepte um rectângulo que rodeia em um metro a linha de golo em todas as direcções) e cuja velocidade inicial permitisse atingir a baliza com um velocidade superior a 1 metro/ciclo ou que realmente atinja a baliza.
- **Oportunidades de Golo (OC-OS):** Oportunidades de golo construídas e sofridas em cada jogo. No âmbito desta avaliação foi considerada uma oportunidade de golo um remate ou uma bola que embora não tenha sido rematada em direcção à baliza chegou a ser controlada (estar na área de chuto) por um avançado numa zona perigosa. Por zona perigosa foi considerado o rectângulo com largura 30 m

---

[87] Em 2002 a FC Portugal não participou no campeonato europeu e classificou-se em quinto lugar no campeonato do mundo. Neste mesmo campeonato classificou-se em terceiro lugar na competições de visualizadores e venceu a competição de treinadores. O trabalho de investigação que conduziu à elaboração da equipa FC Portugal 2002 e da sua continuação FC Portugal 2003 não é descrito nesta dissertação.

e comprimento 18.5m (até ao limite da área) representado na figura 82. Foram também consideradas como oportunidades de golo bolas enviadas em direcção a um jogador que se encontrasse numa zona muito perigosa mas que tenham sido interceptadas com dificuldade por adversários. O conceito de região muito perigosa corresponde ao rectângulo com largura de 20m e comprimento 15m (ver figura 82). Uma intercepção difícil foi considerada aquela em que se o adversário que interceptou a bola, após ter começado o seu movimento de intercepção, tivesse falhado um comando não teria conseguido interceptar a bola.

- **Posse de Bola (Ataque, Meio-Campo ou Defesa):** Localização da bola (em percentagem) no decurso de cada jogo. Para estimar a posse de bola, o campo foi dividido em três rectângulos com igual área, considerando-se o tempo (em ciclos) que a bola permaneceu em cada um destes rectângulos.

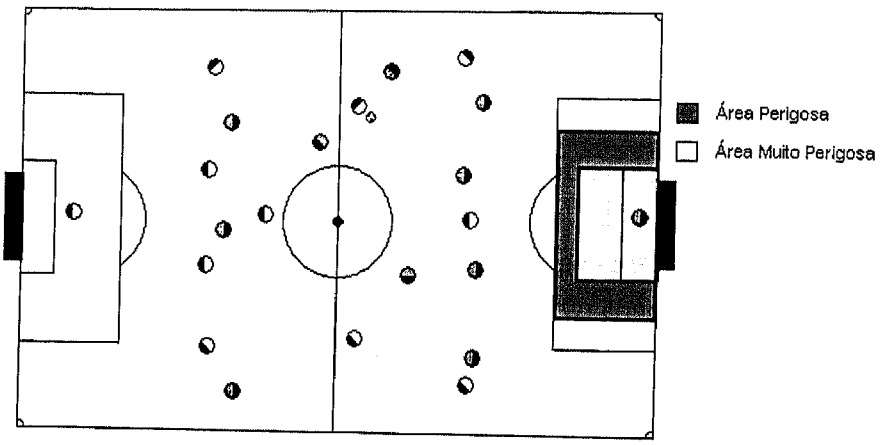

Figura 82: Regiões perigosa e muito perigosa consideradas como conducentes a oportunidades de golo

De forma a efectuar a avaliação, as medidas acima enumeradas foram incorporadas no analisador de jogo contido na ferramenta *Team Designer* e estimadas de forma automática.

9.12.2 Campeonato Europeu – Amesterdão 2000

O primeiro campeonato Europeu de RoboCup disputou-se em Amesterdão na Holanda de 29 de Maio a 2 de Junho de 2000. Esta foi a primeira competição oficial em que a equipa FC Portugal participou. Embora tenham sido já utilizados pela equipa versões preliminares do SBSP e DPRE, a equipa ainda não possuía a nível dos *low-level skills* o *Optimization Kick* e *Smart Drible* e não possuia as metodologias de coordenação por comunicação avançada, percepção inteligente e controlo parcialmente hierárquico. Os resultados obtidos pela equipa no campeonato Europeu de 2000 podem ser analisados na tabela 26.

Adversário	*Result.*	*Remat.*	*Oport.*	*Ataque*	*Meio*	*Defesa*
Primeira Fase – Grupo A						
Essex Wizards (Inglaterra)	*3-0*	*8-0*	*14-1*	*57%*	*32%*	*11%*
Lucky Lubeck (Alemanha)	*13-0*	*19-0*	*28-1*	*56%*	*37%*	*7%*
Cyberoos (Austrália)	*4-0*	*6-1*	*12-1*	*54%*	*43%*	*3%*
Pizza Tower (Itália)	*22-0*	*25-0*	*30-0*	*51%*	*49%*	*0%*
Polytech (Rússia)	*19-0*	*25-0*	*30-0*	*54%*	*32%*	*14%*
PSI (Rússia)	*6-0*	*21-0*	*36-0*	*78%*	*21%*	*1%*
Fase Final						
Wroclaw (Polónia)	*13-0*	*16-0*	*19-0*	*53%*	*40%*	*7%*
Essex Wizards (Inglaterra)	*5-0*	*8-2*	*11-8*	*33%*	*35%*	*32%*
Final						
Karlsruhe Brainstormers (Alemanha)	*2-0*	*8-4*	*14-5*	*37%*	*30%*	*33%*
Total	*87-0*	*136-7*	*194-16*	*53%*	*35%*	*12%*

Tabela 26: Resultados obtidos pela FC Portugal no campeonato Europeu – Amesterdão 2000

Na tabela 26 é visível que a equipa teve um grande domínio em todos os jogos com maior posse de bola no ataque, mais oportunidades de golo, mais remates e mais golos marcados do que qualquer adversário. O carácter um pouco aleatório dos jogos de futebol robótico simulado pode ser verificado comparando os dois jogos realizados entre a equipa FC Portugal e os Essex Wizards. No primeiro jogo a equipa FC Portugal teve um domínio muito superior com mais oportunidades de golo (14 contra 1 do adversário enquanto no segundo jogo as oportunidades foram 11 contra 8) e muito mais ataque (57% contra 11% de ataque no primeiro jogo contra unicamente 33% contra 32% de ataque no segundo jogo). No entanto o resultado final do primeiro jogo foi 3-0 contra 5-0 no segundo jogo. Na final, a capacidade colectiva da equipa FC Portugal superiorizou-se aos excelentes *low-level skills* (chuto, movimentação, drible e intercepção) dos Karlsruhe Brainstromers.

9.12.3 RoboCup 2000 - Melbourne

O campeonato do mundo de futebol robótico – RoboCup 2000 foi disputado em Melbourne de 26 de Agosto a 3 de Setembro de 2000. Destaca-se que a equipa FC Portugal foi a única equipa portuguesa a participar neste campeonato[88]. Neste campeonato a equipa FC Portugal utilizou já os mecanismos de visão estratégica, comunicação avançada, *Optimization Kick* e as versões descritas do mecanismo de posicionamento

[88] Embora diversas equipas portuguesas estivessem pré-qualificadas nas modalidades de robôs pequenos e médios.

estratégico - SBSP e troca dinâmica de posições – DPRE. Os resultados obtidos demonstram uma enorme superioridade em todas as estatísticas relativamente aos adversários.

Adversário	*Resul.*	*Remat.*	*Oport.*	*Ataque*	*Meio*	*Defesa*
Primeira Fase – Grupo D						
Oulu 2000 (Finlândia)	*33-0*	*34-0*	*37-0*	*51%*	*49%*	*0%*
Zeng 2000 (Japão)	*18-0*	*26-0*	*33-0*	*65%*	*33%*	*2%*
Robolog 2000 (Alemanha)	*20-0*	*26-0*	*30-0*	*61%*	*36%*	*3%*
Fase Final						
Essex Wizards (Inglaterra)	*7-0*	*15-0*	*24-0*	*70%*	*22%*	*8%*
Karlsruhe Brainstormers (Alemanha)	*3-0*	*6-2*	*15-3*	*48%*	*31%*	*21%*
YowAI2000 (Japão)	*6-0*	*19-0*	*33-0*	*76%*	*22%*	*2%*
ATT-CMUnited 2000 (EUA)	*6-0*	*12-1*	*18-2*	*53%*	*36%*	*11%*
Final						
Karlsruhe Brainstormers (Alemanha)	*1-0*	*6-0*	*11-2*	*46%*	*38%*	*16%*
Total	*94-0*	*144-3*	*201-7*	*59%*	*33%*	*8%*

Tabela 27: Resultados obtidos pela FC Portugal no campeonato Mundial – Melbourne 2000

Dos resultados obtidos (tabela 27) realça-se a vitória por 6-0 (12 remates contra 1 e 18 oportunidades contra 2) sobre os anteriores campeões do mundo (ATTCMUnited) que até esse jogo e nos últimos 3 campeonatos do mundo não tinham sofrido qualquer golo. Destaca-se também que embora o resultado da final tenha sido unicamente 1-0, a vitória foi obtida com grande facilidade (6-0 em remates, 11-2 em oportunidades e 46% contra 16% de ataque). Destaca-se ainda que o guarda-redes da equipa só teve de defender três remates nos oito jogos do torneio.

9.12.4 German Open 2001 – Paderborn

O primeiro German Open foi disputado em Paderborn entre 7 e 10 de Junho de 2001. Este campeonato, desde 2001 funcionou como campeonato europeu, reunindo não só as melhores equipas europeias das várias modalidades RoboCup como também algumas equipas fortes de países não europeus (como por exemplo o Irão).

Adversário	*Resul.*	*Remat.*	*Oport.*	*Ataque*	*Meio*	*Defesa*
Primeira Fase – Grupo A						
Sharif Arvand (Irão)	*2-0*	*3-1*	*6-5*	*24%*	*47%*	*29%*
Aras (Irão)	*3-1*	*3-1*	*11-2*	*36%*	*57%*	*8%*
Osna BallByters (Alemanha)	*26-0*	*32-0*	*41-0*	*70%*	*29%*	*1%*
UvaTrilearn (Holanda)	*5-2*	*6-6*	*10-7*	*29%*	*49%*	*22%*
Lucky Lubeck (Alemanha)	*3-1*	*8-4*	*11-4*	*35%*	*49%*	*16%*
Fase Final						
Dr. Web (Rússia)	*6-0*	*7-1*	*16-3*	*51%*	*41%*	*9%*
Robolog (Alemanha)	*9-0*	*13-2*	*13-2*	*28%*	*60%*	*12%*
Karlsruhe Brainstormers (Alemanha)	*1-0**	*4-1*	*10-1*	*21%*	*30%*	*49%*
Final					** após prolongamento*	
Karlsruhe Brainstormers (Alemanha)	*1-0*	*3-1*	*10-1*	*39%*	*32%*	*29%*
Total	*56-4*	*79-17*	*128-25*	*37%*	*44%*	*19%*

Tabela 28: Resultados obtidos pela FC Portugal no German Open 2001

A equipa FC Portugal utilizada neste torneio foi a equipa FC Portugal 2000 cuja configuração se mostrou desadequada às novas regras do RoboCup em 2001, nomeadamente à maior velocidade da bola resultante da maior potência de chuto e jogadores com mais energia[89]. Acresce a este facto que a equipa havia disponibilizado em Outubro de 2000 uma parte do código do FC Portugal, contendo uma versão simplificado do algoritmo de posicionamento estratégico e diversos artigos descrevendo as principais metodologias da equipa. Exactamente metade das equipas no German Open 2001 utilizou este código e consequente método de posicionamento e basearam a filosofia das suas equipas na do FC Portugal 2000.

Desta desadequação da equipa às novas regras resultou que, na primeira fase, diversos dos seus jogos foram bastante equilibrados com destaque para os jogos com as equipas Sharif Arvand e UvaTrilearn [de Boer et al., 2001].

Para os jogos da fase final a configuração da equipa foi ligeiramente alterada (aumentando a tendência dos jogadores para se deslocarem estrategicamente no campo e para tentarem intercepções menos seguras). Para a final, a configuração foi novamente alterada, aumentando a rapidez e segurança do jogo. O resultado destas alterações tácticas foi uma melhoria clara da qualidade do jogo que se reflecte nas estatísticas de jogo apresentadas na tabela 28. É visível que a equipa FC Portugal, embora jogando contra as melhores equipas da competição, atacou bastante mais na fase final do torneio e criou muito mais oportunidades de golo do que os seus adversários. Do primeiro jogo contra os

[89] Alterações introduzidas pelo comité técnico do RoboCup

Brainstormers para a final é também visível que a maior rapidez de jogo resultou numa passagem da defesa para o ataque mais rápida, fazendo com a equipa FC Portugal atacasse bastante mais neste jogo e o vencesse com mais facilidade.

9.12.5 RoboCup 2001 – Seattle

O campeonato do mundo de futebol robótico disputou-se em Seattle nos Estados Unidos de 2 a 10 de Agosto de 2001. Para este campeonato a principal alteração no comportamento da equipa FC Portugal foi a introdução da coordenação por controlo parcialmente hierárquico e diversas melhorias nos restantes mecanismos de coordenação. Os resultados obtidos pela equipa são apresentados na tabela 29.

Adversário	*Resul.*	*Remat.*	*Oport.*	*Ataque*	*Meio*	*Defesa*
Primeira Fase – Grupo E						
11Monkeys (Japão)	*29-0*	*+29-0*	*+29-0*	**Logfile indisponível*		
TUT-Grove (Japão)	*9-0*	*+9-0*	*+9-0*	**Logfile indisponível*		
RMIT Goannas (Austrália)	*32-0*	*+32-0*	*+32-0*	**Logfile indisponível*		
Robolog2001 (Alemanha)	*8-0*	*+8-0*	*+8-0*	**Logfile indisponível*		
Segunda Fase – Grupo C						
Helli Respina (Irão)	*5-0*	*6-1*	*15-1*	*54%*	*41%*	*5%*
UtUtd (Irão)	*16-0*	*29-0*	*42-0*	*79%*	*20%*	*1%*
FC Tripletta (Japão)	*4-0*	*12-1*	*22-1*	*67%*	*25%*	*8%*
AT Humboldt (Alemanha)	*13-0*	*22-0*	*32-0*	*83%*	*17%*	*0%*
ATTUnited2001 (EUA)	*22-0*	*24-0*	*34-0*	*74%*	*25%*	*1%*
Fase Final						
YowAI2001 (Japão)	*8-0*	*14-0*	*23-1*	*60%*	*28%*	*12%*
UvaTrilearn (Holanda)	*4-1*	*6-2*	*12-6*	*42%*	*40%*	*18%*
Tsinghuaeolus (China)	*0-3*	*3-3*	*9-3*	*37%*	*54%*	*10%*
Karlsruhe Brainstormers (Alem.)	*0-1*	*1-1*	*13-2*	*46%*	*32%*	*22%*
Total	*150-5*	*195*-8*	*281*-14*	*60%*	*32%*	*8%*

Tabela 29: Resultados obtidos pela FC Portugal no campeonato Mundial – Seattle 2001

Analisando os resultados apresentados na tabela 29 conclui-se facilmente que, tal como já tinha sido referido anteriormente, os resultados da competição nem sempre são justos e que o resultado final de um jogo não é a única medida a utilizar para retirar conclusões válidas deste tipo de experiências científicas. Na tabela 29 é visível que a equipa FC Portugal dominou claramente o torneio marcando 150 golos e sofrendo unicamente 5, tendo em todos os jogos um número igual ou superior de remates que os seus adversários, o dobro ou mais de oportunidades de golo e mais do dobro de ataque que os seus oponentes. No entanto, a equipa acabou por perder dois jogos em que teve claro domínio

de jogo e muito mais oportunidades de golo que os seus adversários, sendo eliminada do torneio e classificando-se unicamente em terceiro lugar. Verifica-se ainda que mesmo sem contabilizar os jogos disputados na primeira fase do torneio[90] em que o domínio da equipa FC Portugal foi total, este foi o campeonato em que a equipa mais oportunidades criou (mais de 281), mais rematou à baliza (mais de 195 remates), mais golos marcou (150), mais atacou (60%) e menos defendeu (8%).

Destaca-se que num dos jogos decisivos, os dois últimos golos da equipa Tsinghuaeolus contra o FC Portugal foram marcados no final do jogo quando o treinador da equipa tinha instruído a que a mesma jogasse sem defesas e sem guarda-redes numa tentativa desesperada de inverter o resultado. Esta táctica embora não tenha resultado neste jogo, foi de grande eficácia no jogo contra a equipa UvaTrilearn. Neste jogo e após 2/3 do jogo a equipa holandesa estava a vencer por 1-0, altura em que o treinador da FC Portugal instruiu o guarda-redes para ir também para o ataque. Com 11 jogadores no ataque a equipa FC Portugal dominou a partir daí claramente o jogo acabando por o vencer por 4-1.

9.13 Resultados em Experiências Controladas

Os resultados das competições oficiais RoboCup são bastante significativos na medida em que demonstram inequivocamente a qualidade global da equipa FC Portugal quando comparada com as equipas desenvolvidas pelas outras Universidades/Empresas participantes. Contudo, de forma a analisar os resultados individuais das diversas metodologias de coordenação desenvolvidas, torna-se necessário realizar experiências controladas que são impossíveis de realizar nas competições oficiais. As competições oficiais não são disputadas em ambientes controlados e cada equipa participante disputa no máximo um ou dois jogos contra o FC Portugal o que não permite realizar testes estatisticamente significativos. Desta forma foi realizado um extenso estudo no laboratório de forma a analisar separadamente a influência de cada metodologia desenvolvida na equipa FC Portugal.

9.13.1 Definição das Experiências

Nas experiências realizadas no âmbito desta tese[91], de forma a testar as metodologias de coordenação propostas, foram utilizadas as seguintes equipas[92]:

---

[90] Devido à indisponibilidade dos respectivos logfiles que não foram gravados correctamente pela organização do torneio

[91] As experiências foram realizadas no período compreendido entre Abril e Junho de 2002.

[92] Os binários utilizados correspondem aos binários disponíveis na Internet em Abril de 2002.

- TSI – Tsinghuaeolus – Tsinghua University - China
- KB – Karlsrhue Brainstormers 2001 – University of Karlsruhe - Alemanha
- UVA - UVA Trilearn 2001 – University of Amsterdam - Holanda
- Yow – YowAI 2001 – Electro Communications University -Japão
- FCM - FC Portugal Melbourne – Universidade do Porto/Aveiro - Portugal
- FA - FCP Agent – Universidade do Porto/Aveiro - Portugal
- ATT – ATTCMUnited 2000 – AT&T e Carneggie Mellon University, E.U.A.
- CMU – CMUnited 99 - Carneggie Mellon University, E.U.A.

A avaliação dos resultados foi efectuada tendo como base as medidas descritas na secção 9.12.1: golos marcados/sofridos, remates, oportunidades e posse de bola por região do campo. Os testes foram realizados utilizando o servidor *soccerserver 7.10* e as configurações por defeito de todas as equipas.

A tabela 30 sumaria os resultados obtidos pela equipa FC Portugal (utilizando a sua configuração por defeito) contra as diversas equipas teste utilizadas.

Equipas / Resultados	*TSI*	*KB*	*UVA*	*YOW*	*FCM*	*FA*	*ATT*	*CMU*
Golos (M-S)	8-4	16-0	35-8	92-0	78-2	165-0	222-0	245-0
Vít–Emp–Der	5-2-3	8-2-0	8-2-0	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	20-8	28-5	46-18	136-2	114-7	235-1	278-0	296-0
Oportunidades	76-25	95-10	123-52	194-8	148-19	304-3	393-1	404-0
Ataq–Meio-Def.	33-51-16	47-36-17	40-39-21	58-31-11	54-34-12	70-27-3	78-22-1	76-23-1

Tabela 30: Resultados das Experiências Controladas Simples.

Analisando a tabela 30 é visível que a equipa FC Portugal vence com facilidade todos os adversários, exceptuando a equipa Tsinghuaeolus, com a qual consegue 5 vitórias contra 3 derrotas, nas 10 experiências efectuadas. Unicamente três equipas conseguem marcar golos à equipa FC Portugal: Tsinghuaeolus, UVATrilearn e FC Portugal Melbourne. Um resultado interessante é que a equipa Brainstormers, não conseguiu marcar qualquer golo à FC Portugal nas experiências realizadas (tendo perdido 8 jogos e empatado a 0-0 dois jogos). Este resultado mostra que os resultados da competição, tal como foi referido anteriormente, nem sempre são significativos (no RoboCup 2001, a equipa Brainstormers, eliminou o FC Portugal, vencendo por 1-0).

9.13.2 Coordenação Estratégica

De forma a averiguar a adequação da coordenação estratégica foram realizadas diversas experiências variando a forma de utilização de tácticas, formações, situações, tipos de jogadores e troca dinâmica de posicionamentos.

9.13.2.1 Definição de Tácticas, Formações e Papéis

Na primeira experiência foram utilizados três estratégias distintas:

- ***STSimple*** – Estratégia simples com uma única táctica, uma única formação (433) e um único tipo de jogador (com pesos iguais para as diversas acções com bola ou sem bola)
- ***STTunned*** – Estratégia simples com uma única táctica e a formação (433) utilizada por defeito pela equipa FC Portugal com uma única situação. Esta formação foi treinada extensivamente para jogar contra a maior parte das equipas e utiliza diversos tipos de jogadores distintos.
- ***STChange*** – Estratégia mais elaborada com três tácticas (uma táctica defensiva, uma normal e uma ofensiva), diversos tipos de jogadores utilizados nas tácticas e três formações distintas. Nesta estratégia foi utilizada uma metodologia muito simples para alternar a táctica activa: Uma táctica normal é utilizada no início do jogo. Caso a equipa esteja a vencer o jogo por menos de três golos de diferença ou a perder por mais de dois, a táctica defensiva é utilizada. Caso a equipa esteja a perder por menos de três golos de diferença ou a vencer por mais de dois, a táctica ofensiva é utilizada.

Os resultados obtidos nesta experiência encontram-se representados na tabela 31.

Equipas / Resultados	TSI	KB	UVA	YOW	FCM	FA	ATT	CMU
Estratégia *STSimple*								
Golos (M-S)	4-6	6-2	18-12	76-0	68-2	113-0	196-0	204-0
Vít–Emp–Der	2-4-4	4-5-1	5-3-2	10-0-0	9-1-0	10-0-0	10-0-0	10-0-0
Remates	12-10	14-8	28-18	104-5	96-12	152-3	232-0	246-0
Ataq–Meio-Def.	25-48-27	34-40-26	30-41-29	50-36-14	42-37-21	59-33-8	68-27-5	72-25-3
Estratégia *STTunned*								
Golos (M-S)	9-6	12-1	30-7	91-0	81-1	145-0	209-0	236-0
Vít–Emp–Der	4-3-3	7-3-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	23-12	22-6	42-18	132-3	108-10	196-1	264-0	256-0
Ataq–Meio-Def.	31-51-18	48-33-19	38-37-25	60-30-10	53-34-13	68-29-3	74-25-1	77-22-1
Estratégia *STChange*								
Golos (M-S)	8-4	15-0	34-10	89-0	75-3	159-0	220-0	244-0
Vít–Emp–Der	5-2-3	8-2-0	8-2-0	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	18-7	32-3	47-22	138-1	102-6	221-1	268-0	281-0
Ataq–Meio-Def.	28-58-14	45-40-15	35-48-17	58-31-11	50-36-14	69-28-3	75-23-2	74-25-1

Tabela 31: Resultados das Experiências com Variação de Tácticas

Na tabela 31 é visível que a estratégia *STTunned* permite obter melhores resultados contra todas as equipas do que a estratégia *STSimple*. Este facto deve-se essencialmente a que embora ambas as estratégias utilizem a mesma formação, os tipos de jogadores utilizados pela estratégia *STTunned* são mais adequados à realização da tarefa cooperativa: avançados que rematam mais à baliza, extremos que correm à linha e cruzam a bola, médios mais agressivos, etc.

Uma conclusão interessante e um pouco surpreendente da análise dos resultados obtidos, consiste em que a estratégia *STTunned* permite obter melhores resultados contra algumas equipas do que a estratégia *STChanged.* Nos jogos contra as melhores equipas, nomeadamente no que diz respeito ao número de vitórias, empates e derrotas, a estratégia *STChange* permite claramente obter melhores resultados. Como nesta estratégia a equipa procura defender no caso de estar a ganhar o jogo e atacar no caso de estar a perder, verifica-se que contra a equipa Tsinghuaeolus esta estratégia diminui o número de golos marcados, remates e os tempos de ataque de ambas as equipas.

Uma conclusão importante a retirar é que as alterações tácticas, pelo menos se forem executadas baseadas unicamente no resultado, não produzem um efeito muito visível no comportamento da equipa, sobretudo contra equipas claramente mais fracas. São no entanto muito úteis contra equipas de um nível semelhante, permitindo melhorar o resultado global da equipa.

9.13.2.2 Coordenação por Posicionamento Estratégico

De forma a testar a influência do posicionamento estratégico no comportamento global da equipa foi utilizada uma equipa com uma única táctica (*STTunned*) e cinco estratégias de posicionamento distintas:

- *PACTS* **(Posicionamento Activo Simples)** – Inexistência do conceito de posicionamento estratégico e da distinção entre situação crítica (activa) e situação estratégica. Neste tipo de estratégia, os agentes assumem sempre um comportamento activo sem bola: intercepção, intercepção passiva, aproximação da bola, marcação de linha de passe ou marcação de adversário.
- *PACTF* **(Posicionamento Activo com Formação Estática)** – Semelhante ao anterior mas todos os jogadores possuem uma posição por defeito estática que permite definir uma formação. No caso de nenhuma acção sem bola ter um valor suficientemente elevado os jogadores regressam para esta posição por defeito.
- *SPAR* **(Posicionamento por Atracções e Repulsões)** – Posicionamento baseado no algoritmo de Stone et al. [Stone et al., 2000a]. Jogadores são atraídos pela bola, repelidos pelos colegas de equipa e atraídos pelos adversários se estes tiverem a posse de bola e repelidos caso contrário.
- *SP* **(Posicionamento Estratégico Simples)** – Utilizando unicamente uma situação.
- *SBSP* **(Posicionamento Estratégico baseado em Situações Elaborado)** - Utilizando situações para ataque, defesa, reposições de bola pelos guarda-redes, pontapés de baliza, livres, ocasiões de golo, cantos e lançamentos.

A tabela 32 sumaria os resultados obtidos nas experiências realizadas.

Equipas / Resultados	TSI	KB	UVA	YOW	FCM	FA	ATT	CMU
PACTS (Posicionamento Activo Simples)								
Golos (M-S)	0-65	0-22	1-38	6-19	0-34	2-2	14-6	21-0
Vít–Emp–Der	0-0-10	0-2-8	0-0-10	1-3-6	0-0-10	2-6-2	8-1-1	10-0-0
Remates	1-86	2-36	2-72	13-37	2-61	5-16	26-14	37-16
Ataq–Meio-Def.	8-58-34	9-61-30	14-44-42	16-54-30	8-47-45	14-58-28	38-46-16	42-46-12
PACTF (Posicionamento Activo com Formação Estática)								
Golos (M-S)	0-48	1-18	2-46	10-16	4-24	4-3	24-4	32-0
Vít–Emp–Der	0-0-10	0-3-7	0-0-10	3-4-3	0-0-10	3-5-2	10-0-0	10-0-0
Remates	2-75	3-32	5-89	21-39	6-48	6-20	39-12	54-8
Ataq–Meio-Def.	12-56-32	10-58-32	18-47-35	22-50-28	14-55-31	24-50-26	44-42-14	50-43-7
SPAR (Posicionamento por Atracções e Repulsões)								
Golos (M-S)	0-36	2-12	1-42	8-14	5-25	6-2	33-3	45-0
Vít–Emp–Der	0-0-10	1-4-5	0-0-10	2-4-4	0-0-10	4-5-1	10-0-0	10-0-0
Remates	2-55	8-26	12-76	16-42	6-46	8-12	52-10	71-5
Ataq–Meio-Def.	10-54-36	13-52-35	19-41-40	25-45-30	16-41-43	29-45-26	50-37-13	58-34-8
SP (Posicionamento Estratégico Simples)								
Golos (M-S)	9-6	12-1	30-7	91-0	81-1	145-0	209-0	236-0
Vít–Emp–Der	4-3-3	7-3-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	23-12	22-6	42-18	132-3	108-10	196-1	264-0	256-0
Ataq–Meio-Def.	31-51-18	48-33-19	38-37-25	60-30-10	53-34-13	68-29-3	74-25-1	77-22-1
SBSP(Posicionamento Estratégico baseado em Situações)								
Golos (M-S)	10-3	19-0	32-6	98-0	76-1	158-0	204-0	246-0
Vít–Emp–Der	6-2-2	7-3-0	9-1-0	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	16-4	35-2	45-16	146-0	102-3	209-0	253-0	273-0
Ataq–Meio-Def.	34-50-16	40-41-19	40-35-25	63-25-12	55-32-13	71-27-1	72-26-2	75-24-1

Tabela 32: Resultados das Experiências com Variação de Esquema de Posicionamento

Analisando a tabela 32 é visível que os resultados obtidos através da utilização do SBSP são claramente melhores do que os resultados dos restantes esquemas de posicionamento. Verifica-se também que a utilização de diferentes esquemas de posicionamento por situações melhora o comportamento da equipa FC Portugal, sobretudo contra as equipas mais fortes. Verifica-se também que a utilização do SBSP com situações permite à equipa sofrer menos golos do que não utilizando situações. Analisando os jogos disputados verifica-se que este facto está relacionado com o incremento de qualidade da defesa da equipa obtido através da utilização de diferentes esquemas de posicionamento para situações defensivas.

9.13.2.3 Coordenação por Troca Dinâmica de Papéis e Posicionamentos

De forma a aferir da influência da troca dinâmica de papéis e posicionamentos no desempenho global da equipa foram realizadas experiências utilizando diferentes esquemas de DPRE:

- ***DPRE0:*** Trocas dinâmicas de posicionamento não são efectuadas:
- ***DPRE1:*** Trocas dinâmicas de posição pouco frequentes baseadas unicamente em distâncias às posições estratégicas;
- ***DPRE2:*** Trocas dinâmicas de posição frequentes baseadas unicamente em distâncias às posições estratégicas.

Não foram efectuadas experiências utilizando a adequação dos agentes aos papéis pois o FC Portugal não dispunha de agentes heterogéneos na data de realização destas experiências. Na realização das experiências foi utilizada a estratégia *STTunned* referida anteriormente. Dado que as diferenças de resultado obtidas com a utilização dos esquemas DPRE1 e DPRE2 não foram significativas, na tabela 33, são apresentados unicamente os resultados comparando a utilização do esquema de troca de posições DPRE2 com o esquema DPRE0.

Equipas / *Resultados*	*TSI*	*KB*	*UVA*	*YOW*	*FCM*	*FA*	*ATT*	*CMU*
***DPRE 0* (Sem Trocas de Posição)**								
Golos (M-S)	8-10	10-3	24-12	76-0	65-6	126-0	187-0	225-0
Vít–Emp–Der	2-4-4	6-3-1	6-2-2	10-0-0	9-0-1	10-0-0	10-0-0	10-0-0
Remates	18-14	16-7	34-21	108-2	89-15	170-1	232-0	243-0
Ataq–Meio-Def.	27-50-23	40-36-22	34-39-27	55-34-11	47-33-20	62-33-5	69-28-3	78-21-1
***DPRE 2* (Estratégia *STTunned*)**								
Golos (M-S)	9-6	12-1	30-7	91-0	81-1	145-0	209-0	236-0
Vít–Emp–Der	4-3-3	7-3-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	23-12	22-6	42-18	132-3	108-10	196-1	264-0	256-0
Ataq–Meio-Def.	31-51-18	48-33-19	38-37-25	60-30-10	53-34-13	68-29-3	74-25-1	77-22-1

Tabela 33: Resultados das Experiências com Variação do Esquema de Troca Dinâmica de Posicionamentos

Analisando os resultados é visível que a troca dinâmica de posições tem grande influência no desempenho da equipa sobretudo contra as melhores equipas. Embora o número de remates dos adversários não aumente significativamente não utilizando o DPRE, verifica-se que o número de golos dos adversários sofre um acréscimo significativo. Analisando alguns dos jogos em pormenor verifica-se que o acréscimo de golos se deve a situações

defensivas em que a não execução de uma dada troca de posicionamento, deixa um "buraco" na defesa que permite aos adversários marcar um golo de forma fácil.

A utilização do DPRE permite também que uma equipa com um número mais reduzido de jogadores obtenha um bom desempenho. Foram realizadas algumas experiências simples com os seguintes cenários:

- ***NODPRE-5PL*** - Sem DPRE utilizando uma equipa composta por 5 jogadores (guarda –redes e quatro jogadores de campo);
- ***DPRE-5PL*** – Com DPRE2 utilizando uma equipa composta por 5 jogadores.

Utilizando como equipa de teste a equipa CMUnited verificou-se que a equipa FC Portugal com cinco jogadores sem DPRE venceu 3 jogos em cinco, empatando os restantes dois contra a equipa CMUnited, marcando um total de 14 golos contra 4 do adversário. No entanto, a mesma equipa utilizando DPRE, consegue vencer todos os jogos marcando um total de 36 golos contra 2 do adversário também em 5 jogos.

9.13.3 Coordenação por Controlo Parcialmente Hierárquico

Infelizmente é muito complicado medir quantitativamente no laboratório a influência da coordenação por controlo parcialmente hierárquico. A melhor medida do sucesso desta metodologia foi o seu resultado em competições oficiais, nomeadamente no jogo entre a FC Portugal e a equipa UVATrilearn disputado no campeonato do mundo em 2001. Neste jogo, como foi já descrito, a influência do treinador foi determinante para permitir já perto do final do jogo alterar um resultado desfavorável de 0-1 para um resultado favorável de 4-1.

As experiências controladas da Coordenação por Controlo Parcialmente Hierárquico realizadas utilizaram os seguintes cenários:

- ***PH0*** – Táctica por defeito *(STTunned)* é utilizada;
- ***PH1*** – Treinador altera a táctica (seleccionando outra táctica aleatoriamente) quando o resultado for inferior ao esperado;
- ***PH2*** – Treinador selecciona a táctica baseado nas estatísticas do jogo e comportamento do adversário.

Os resultados obtidos podem ser visualizados na tabela 34.

Equipas / Resultados	TSI	KB	UVA	YOW	FCM	FA	ATT	CMU
PH0 – Táctica Única (*STTunned*)								
Golos (M-S)	9-6	12-1	30-7	91-0	81-1	145-0	209-0	236-0
Vít–Emp–Der	4-3-3	7-3-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	23-12	22-6	42-18	132-3	108-10	196-1	264-0	256-0
Ataq–Meio-Def.	31-51-18	48-33-19	38-37-25	60-30-10	53-34-13	68-29-3	74-25-1	77-22-1
PH1 – Alteração de Táctica em Função do Resultado								
Golos (M-S)	7-8	13-2	34-5	82-0	73-3	147-0	184-0	203-0
Vít–Emp–Der	3-4-3	6-4-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	18-13	19-7	48-17	118-3	96-16	192-6	232-0	246-0
Ataq–Meio-Def.	28-50-22	49-30-21	40-38-22	56-33-11	45-36-19	64-32-4	65-29-6	71-28-1
PH2 – Táctica em Função do Resultado e Estatísticas de Jogo								
Golos (M-S)	8-3	11-1	36-5	99-0	89-2	139-0	203-0	228-0
Vít–Emp–Der	5-3-2	8-2-0	7-2-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	17-8	21-4	52-14	147-3	138-12	185-3	238-0	262-0
Ataq–Meio-Def.	28-56-16	47-36-17	36-36-28	54-38-8	55-33-12	66-31-3	74-25-1	77-22-1

Tabela 34: Resultados das Experiências com Coordenação Parcialmente Hierárquica

Analisando os resultados conclui-se que a influência da coordenação parcialmente hierárquica nos resultados competitivos da equipa é bastante reduzida. Aliás esta coordenação parece ter um efeito detrimental na equipa quando esta joga com as equipas mais fracas. A explicação parece estar relacionada com o facto de um táctica bem afinada ser melhor para jogar com equipa fracas do que tácticas mais elaboradas mas com pormenores menos afinados.

A estratégia PH2 funciona bastante bem contra as melhores equipas, permitindo aumentar o número de vitórias mas diminuindo o número de golos marcados. Este facto prende-se com o facto de o treinador, verificando através das estatísticas de jogo que estes são jogos equilibrados, após a equipa marcar o primeiro golo, ordenar a utilização de tácticas mais defensivas. Desta forma o jogo processa-se mais a meio campo e com menos remates a ambas as balizas e consequentemente com menos golos marcados.

9.13.4 Coordenação por Comunicação, Percepção Inteligente e Modelização Mútua

De forma a aferir do interesse das metodologias de comunicação avançada, percepção inteligente e modelização mútua foram realizadas diversas experiências analisando os resultados competitivos obtidos através da sua utilização. Em todas as experiências foi utilizada a estratégia *STTunned*:

- ***ADVCOM+MM+SLM:*** Todas as metodologias (ADVCOM, SLM e modelização mútua) são utilizadas;
- ***NOADVCOM*** – Sem a utilização de comunicação;
- ***NOMM*** – Sem a utilização de modelização mútua;
- ***NOSLM*** – Sem utilização de visão estratégica;
- ***NO*** - Sem Comunicação, SLM e Modelização Mútua (visão aleatória é utilizada).

Na tabela 35 são sumariados os resultados obtidos em cada estratégia:

Equipas / *Resultados*	*TSI*	*KB*	*UVA*	*YOW*	*FCM*	*FA*	*ATT*	*CMU*
ADVCOM + MM + SLM (STTunned)								
Golos (M-S)	9-6	12-1	30-7	91-0	81-1	145-0	209-0	236-0
Vít–Emp–Der	4-3-3	7-3-0	8-1-1	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	23-12	22-6	42-18	132-3	108-10	196-1	264-0	256-0
Ataq–Meio-Def.	31-51-18	48-33-19	38-37-25	60-30-10	53-34-13	68-29-3	74-25-1	77-22-1
NOADVCOM – Sem *ADVCOM*								
Golos (M-S)	6-12	8-3	14-15	59-2	44-8	98-0	167-0	174-0
Vít–Emp–Der	2-4-4	5-4-1	3-3-4	10-0-0	9-0-1	10-0-0	10-0-0	10-0-0
Remates	12-18	16-8	26-28	83-12	62-22	125-12	194-1	220-0
Ataq–Meio-Def.	24-50-26	40-37-23	26-44-30	48-36-16	38-40-22	50-40-10	63-31-6	65-31-4
NOMM - Sem *MM*								
Golos (M-S)	7-5	8-2	23-10	83-0	68-3	138-0	198-0	239-0
Vít–Emp–Der	4-4-2	5-4-1	6-2-2	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	19-11	17-5	36-24	111-2	82-16	184-2	225-0	251-0
Ataq–Meio-Def.	32-42-26	46-35-19	32-43-25	56-32-12	46-34-20	71-25-4	70-29-1	76-21-1
NOSLM - Sem *SLM*								
Golos (M-S)	8-9	10-4	18-12	72-0	72-1	124-0	186-0	211-0
Vít–Emp–Der	3-3-4	8-2-0	6-2-2	10-0-0	10-0-0	10-0-0	10-0-0	10-0-0
Remates	16-14	17-8	34-22	100-6	99-18	160-3	232-2	234-0
Ataq–Meio-Def.	28-48-24	42-41-17	30-40-30	54-36-10	48-36-16	58-36-6	66-31-3	72-27-1
NO - Sem *ADVCOM, MM* e *SLM*								
Golos (M-S)	1-45	3-5	8-22	18-6	18-14	64-1	127-0	152-0
Vít–Emp–Der	0-0-10	2-6-2	1-4-5	8-2-0	6-2-2	10-0-0	10-0-0	10-0-0
Remates	3-58	7-13	15-43	42-13	32-26	95-28	171-6	180-0
Ataq–Meio-Def.	14-32-54	11-49-40	17-48-35	24-40-36	24-45-31	21-61-18	42-40-18	52-40-8

Tabela 35: Resultados das Experiências com Coordenação por Comunicação, Modelização Mútua e Percepção Inteligente

Analisando os resultados conclui-se que individualmente a metodologia com maior influência no desempenho da equipa é o ADVCOM. Sobretudo contra as equipas Tsinghuaeolus e UvaTrilearn, a simples desactivação da comunicação do FC Portugal faz com a equipa seja derrotada em mais jogos do que os que vence contra estas equipas.

Contra equipas fracas, o efeito detrimental no desempenho não é muito acentuado para nenhuma das metodologias, sendo no entanto mais acentuado também para o ADVCOM.

A influência negativa da desactivação do SLM é mais acentuada em quase todos os casos do que a influência negativa da desactivação da modelização mútua. Exceptuam-se os testes contra a equipa FCP Melbourne em que a desectivação do SLM teve pouca influência competitiva.

A desactivação das três metodologias em simultâneo revela-se desastrosa para o desempenho da equipa. A equipa perde todos os jogos contra os Tsinghuaeolus, sofrendo um total de 45 golos, marcando unicamente um, é derrotada na maioria dos jogos contra os UvaTrilearn e tem inclusivamente muitas dificuldades em vencer o FC Portugal Melbourne. De facto estas três metodologias são complementares permitindo ao agente balancear a informação recebida através da visão, comunicação e predição, na criação de um estado do mundo preciso. A desactivação de uma ou duas metodologias não é ainda crítica pois as metodologias activas permitem ainda isoladamente manter o estado do mundo relativamente actualizado. No entanto, desactivando as três metodologias em simultâneo, o comportamento da equipa altera-se completamente pois o estado do mundo impreciso resultante não permite efectuar correctamente os raciocínios de posicionamento, troca de posições e decisão individual necessários para o seu funcionamento.

9.13.5 Análise de Resultados

Embora o número de experiências realizadas tenha sido bastante elevado, a complexidade do domínio e a dificuldade em encontrar medidas quantitativas que permitam avaliar as metodologias propostas, conduzem à conclusão de que muito mais experiências seriam necessárias para realmente demonstrar a influência das diversas metodologias de coordenação no comportamento global da equipa. Das experiências realizadas conclui-se que as metodologias com maior influência no desempenho global da equipa são: SBSP – Posicionamento Estratégico Baseado em Situações, DPRE – Troca Dinâmica de Posições e Papéis e o ADVCOM – Comunicação avançada. Conclui-se também que metodologias que implicam mudanças tácticas e a utilização do treinador têm sobretudo influência contra as melhores equipas de teste.

9.14 Conclusões

A aplicação de novas metodologias de coordenação, nomeadamente da coordenação estratégica ao domínio do RoboCup, levou a equipa FC Portugal a vencer um campeonato do mundo e dois campeonatos europeus de futebol robótico simulado. A flexibilidade que esta coordenação permite foi a "arma" principal do FC Portugal nestes campeonatos onde teve de medir forças com equipas compostas por jogadores com muito melhores capacidades individuais.

Através de resultados experimentais obtidos num conjunto alargado de experiências foi comprovado que a capacidade da equipa FC Portugal está directamente relacionada com as suas metodologias de coordenação. Os resultados apresentados demonstram claramente que a utilização de metodologias de coordenação como o SBSP ou o DPRE permitem aumentar a capacidade da equipa para realizar a tarefa cooperativa em causa: jogar um jogo de futebol.

Embora as metodologias de coordenação apresentadas tenham sido testadas através de um conjunto alargado de experiências, muito mais experiências seria necessário realizar de forma a averiguar as interacções entre cada tipo de metodologia de coordenação. Por outro lado, a influência de metodologias como a coordenação parcialmente hierárquica ou a utilização de estratégias complexas compostos por múltiplas tácticas com condições de activação baseadas em estatísticas de jogo é muito difícil de analisar. Por vezes, uma única táctica com uma única formação bem definida para um dado jogo, funciona muito bem, sendo muito difícil um treinador conseguir afinar essa táctica e melhorar o desempenho global da equipa.

Capítulo 10

10. Conclusões e Perspectivas de Desenvolvimento

10.1 Síntese do Trabalho Desenvolvido

O objectivo geral deste trabalho foi o estudo e desenvolvimento de metodologias de coordenação em sistemas multi-agente e sua aplicação em equipas de agentes que realizam tarefas complexas em ambientes dinâmicos, inacessíveis e não determinísticos. Neste contexto, foram:

- Estudados os conceitos de Agente Autónomo e Sistema Multi-Agente e as diversas metodologias propostas pelos investigadores da área para realizar a coordenação de agentes;
- Analisados domínios de aplicação adequados à aplicação de metodologias de coordenação de agentes, nomeadamente: problemas de escalonamento na gestão universitária e à liga de simulação do RoboCup;
- Estudadas as abordagens de outros investigadores da área aos domínios estudados com ênfase nas metodologias de coordenação aplicadas a estes domínios;
- Definida uma linguagem – *SCHEDULING UNILANG* – que permite a coordenação de agentes por conhecimento *a priori* no domínio da resolução de problemas de escalonamento na gestão universitária;
- Projectado e implementado um sistema de resolução de problemas de gestão universitária: *UNIPS – University Planning and Scheduling System.*
- Proposta a coordenação estratégica como metodologia de coordenação para a execução de tarefas cooperativas através de equipas de agentes com mobilidade espacial;
- Propostas diversas metodologias de coordenação para equipas de agentes com mobilidade espacial, tais como o SBSP – Posicionamento Estratégico Baseado

em Situações e o DPRE – Troca Dinâmica de Posicionamentos e Papéis;

- Propostas diversas metodologias de coordenação, incluindo a Comunicação Avançada, Percepção Inteligente, Modelização Mútua e Controlo Parcialmente Hierárquico;
- Definida uma linguagem – *COACH UNILANG* que permite a um agente treinador utilizar a Coordenação por Controlo Parcialmente Hierárquico;
- Projectado e Implementado um Sistema Multi-Agente totalmente funcional – FC Portugal – que inclui a implementação de todas as metodologias de coordenação propostas;
- Validação das metodologias desenvolvidas em competições oficiais de futebol robótico e experiências controladas (em comparação directa com as metodologias de outros investigadores), comprovando o aumento do desempenho fornecido pela utilização de cada metodologia de coordenação.

10.2 Principais Resultados e Conclusões

A análise do domínio da Gestão Universitária revelou a complexidade da procura de soluções para o problema da geração de horários e os problemas que lhe estão associados (distribuição de serviço docente, calendarização de exames e alocação de salas). Foi concluído que as metodologias de resolução e os sistemas comerciais de geração de horários não permitem resolver problemas de escalonamento da gestão universitária de forma integrada, efectuando a ligação entre as diversas fases de resolução do mesmo. Com base nesta conclusão, foi definida uma linguagem – *SCHEDULING UNILANG* – que permite representar os problemas de escalonamento universitário de forma integrada e permite a sua resolução através de um conjunto de agentes autónomos. O domínio da gestão universitária foi utilizado nesta dissertação de forma bastante simplificada uma vez que as metodologias de coordenação desenvolvidas implicam na sua maioria a movimentação espacial de agentes.

Partindo do estudo efectuado, foram desenvolvidas formalizações para os principais problemas de escalonamento universitário e foi definida uma linguagem – *SCHEDULING UNILANG* – que permite efectuar a coordenação por conhecimento *a priori*, representando qualquer problema de escalonamento na gestão universitária. Através da utilização desta linguagem é possível coordenar um conjunto de agentes escalonadores e agentes que representam as diversas entidades presentes no processo (docente, turmas e salas) de forma a resolver os problemas de escalonamento referidos.

Do estudo realizado relativamente a possíveis domínios de aplicação, foi concluído que o RoboCup e em particular a liga de simulação constituíam o domínio ideal para a aplicação das metodologias de coordenação desenvolvidas. A liga de simulação inclui

complexidades encontradas nos sistemas robóticos (tais como erros na percepção e acção) mas ao mesmo tempo utiliza um cenário muito realista do ponto de vista da tarefa cooperativa a executar pelas equipas de agentes. As características da simulação, tornam este cenário particularmente adequado às metodologias de coordenação desenvolvidas:

- O facto de ser uma simulação em tempo-real em que os agentes dispõem de diversas fontes de informação sensorial, sensores configuráveis e comunicação pouco fiável e com baixa largura de banda, torna o domínio adequado à aplicação da comunicação avançada, percepção inteligente e modelização mútua;
- O ambiente multi-objectivo, parcialmente cooperativo e parcialmente adverso com um vasto conjunto de acções de baixo-nível disponíveis e em que os agentes são heterogéneos, têm a necessidade de criar acções colectivas complexas e enfrentar um oponente com uma estratégia desconhecida à partida, torna este ambiente particularmente atractivo para a coordenação estratégica.
- A complexidade da tarefa a desempenhar pela equipa de agentes autónomos associada à disponibilidade de um agente privilegiado, hierarquicamente superior aos restantes e com uma visão global do mundo, torna este domínio particularmente adequado à coordenação por controlo parcialmente hierárquico.

Do estudo relativo às metodologias de coordenação, negociação e cooperação entre agentes resultou a definição de algumas novas metodologias de alto-nível de coordenação de agentes sendo a principal proposta a coordenação estratégica (incluindo análise situacional, posicionamento estratégico baseado em situações, utilizando formações dinâmicas, definição e trocas de papéis e posicionamentos de agentes). Outras metodologias de coordenação propostas incluíram a comunicação avançada, modelização mútua, percepção inteligente e controlo parcialmente hierárquico.

As estratégias de coordenação propostas foram aplicadas a domínios distribuídos e descentralizados em que múltiplos agentes executam uma tarefa cooperativa complexa. As metodologias de coordenação são, no entanto, generalizáveis de forma a serem aplicadas a outros domínios socialmente mais úteis.

Através de resultados experimentais no domínio do futebol robótico, obtidos em competições oficiais e experiências controladas, foi comprovado que a coordenação estratégica proposta no âmbito desta tese, permite realizar uma coordenação de alto-nível muito eficaz na realização de tarefas complexas, incrementando significativamente o desempenho de uma equipa de agentes. Foi também comprovada a eficácia das outras metodologias de coordenação propostas.

O principal resultado a referir é provavelmente que as metodologias de coordenação desenvolvidas no âmbito desta tese conduziram a excelentes resultados na competição RoboCup. A equipa FC Portugal, desenvolvida no âmbito desta tese, venceu um campeonato do mundo e dois campeonatos da Europa da liga de simulação de futebol

robótico, marcando um total de 387 golos e sofrendo unicamente 9 nas competições realizadas em 2000 e 2001. Nestas competições, a equipa confrontou por 39 vezes, as suas metodologias de coordenação com as metodologias desenvolvidas por algumas das melhores universidades mundiais de 12 países, incluindo a Alemanha, Japão, E.U.A., Inglaterra, Holanda, China, Rússia e Austrália.

10.3 Originalidades

A principal proposta desta dissertação, a coordenação estratégica, constitui a sua principal originalidade. De facto, a utilização da coordenação estratégica, associada à coordenação espacial por posicionamento estratégico e à coordenação por definição e troca de papéis, permite obter uma enorme flexibilidade na execução de tarefas cooperativas complexas por equipas de agentes, em ambientes dinâmicos e inacessíveis.

Os resultados obtidos em experiências controladas, demonstram claramente que a coordenação estratégica é a grande responsável pela capacidade da equipa FC Portugal e consequentemente demonstram também que esta metodologia é a principal responsável pelos resultados obtidos pela FC Portugal em competições oficiais de futebol robótico. Os campeões do mundo em anteriores competições (CMUnited - 1999) e seguintes (Tsinghuaeolus - 2001) utilizaram muito boas capacidades individuais de baixo-nível (drible, intercepção) e uma estratégia única bem afinada para toda a competição, sendo estas as suas principais forças. Pelo contrário, a equipa FC Portugal utilizou as metodologias de coordenação descritas nesta tese sendo capaz de alterar a sua estratégia de jogo de forma drástica, de acordo com o curso de cada jogo com evidentes resultados competitivos.

Outras originalidades da tese incluem:

- Os mecanismos associados à coordenação estratégica, nomeadamente o SBSP – Posicionamento Estratégico Baseado em Situações e DPRE – Troca Dinâmica de Posicionamentos e Papéis. O SBSP após a sua publicação e divulgação de parte do código tornou-se no mecanismo de posicionamento mais utilizado pelas equipas de futebol robótico simulado. No campeonato do mundo de 2001, 6 dos 8 finalistas utilizaram este mecanismo de posicionamento;
- As linguagens definidas para representação de problemas de escalonamento na gestão universitária – SCHEDULING UNILANG – e representação de estratégias no domínio do futebol robótico – COACH UNILANG. A originalidade inclui não só a sintaxe e semântica das linguagens mas também a sua utilização como forma a permitir a um conjunto de agentes se coordenarem por conhecimento *a priori* e executarem uma tarefa cooperativa complexa;
- Os mecanismos de coordenação por comunicação avançada, modelização mútua

e percepção inteligente. A sua utilização individual e sobretudo a sua utilização combinada, permitem aumentar consideravelmente o desempenho de uma equipa de agentes na execução de uma tarefa cooperativa complexa.

10.4 Limitações

As principais limitações do trabalho prendem-se com a utilização de um único domínio para validar as metodologias de coordenação propostas. De facto, as principais propostas deste trabalho (coordenação estratégica, SBSP, DPRE, SLM, ADVCOM, modelização mútua e coordenação por controlo parcialmente hierárquico) foram aplicadas exclusivamente no domínio do futebol robótico simulado. Este facto conduz facilmente à crítica que as metodologias de coordenação não são genéricas e sim dependentes do domínio. No entanto, tal conclusão não é, em nossa opinião, verdadeira.

A definição de metodologias de coordenação eficazes implica sempre um dado tipo de domínio e um determinado tipo de tarefa cooperativa. Evidentemente que a coordenação estratégica, nomeadamente no que diz respeito ao posicionamento estratégico baseado em situações e troca dinâmica de posicionamentos, só é aplicável em domínios em que equipas de agentes com mobilidade espacial realizam tarefas cooperativas complexas, que implicam a necessidade de formações dinâmicas e comportamentos distintos, dependendo da situação. No entanto, domínios com estas características incluem não só o futebol robótico simulado, como também, entre outros: as outras ligas do futebol robótico, o RoboCup Rescue, cenários de combate a incêndios florestais e limpeza de minas e todos os cenários de guerra ou batalha.

10.5 Perspectivas de Desenvolvimento

Dada a natureza do trabalho realizado no âmbito desta dissertação, as perspectivas de desenvolvimento que se apresentam são diversas. Relativamente ao sistema UNIPS e metodologias de coordenação desenvolvidas no âmbito da gestão universitária, as perspectivas de desenvolvimento incluem:

- Extensão do sistema UNIPS com capacidades para efectuar a marcação automática de reuniões, novos algoritmos de geração de horários e resolução dos problemas associados;
- Realização de um conjunto mais alargado de experiências utilizando o Sistema UNIPS em modo multi-agente e comparação dos resultados obtidos com os resultados obtidos através da resolução manual e assistida e utilizando o modo centralizado;
- Criação de um repositório de problemas de escalonamento universitário, tendo

como base a linguagem *SCHEDULING UNILANG*;

- Definição de uma competição internacional de geração de horários (e resolução de problemas associados) tendo como base problemas definidos em *SCHEDULING UNILANG*;
- Desenvolvimento do sistema UNIPS de forma a ser aplicado numa universidade portuguesa.

No âmbito do desenvolvimento de metodologias de coordenação no domínio do futebol robótico simulado, apresentam-se como principais perspectivas de desenvolvimento:

- Extensão da formalização da estratégia de equipa com a inclusão de planos flexíveis e respectivas condições de activação e término. A representação dos planos flexíveis deverá ser realizada definindo os diversos passos do plano, agentes participantes, suas acções preferenciais e formação utilizada em cada passo.
- Extensão das metodologias de coordenação espacial definidas para domínios tri-dimensionais (como por exemplo o combate aéreo);
- Realização de experiências com os protocolos de definição e troca dinâmica de papéis e posicionamentos para agentes heterogéneos com inclusão de importâncias de posicionamentos definidos;
- Discussão dos conceitos, sintaxe e semântica da linguagem COACH UNILANG com especialistas do domínio do futebol (treinadores profissionais, jornalistas e jogadores). Esta discussão deverá permitir refinar a linguagem, utilizando conceitos utilizados pela comunidade de futebol real;
- Extensão das metodologias de modelização mútua através da criação de modelos da equipa adversária através da observação do seu comportamento ao longo do tempo. Utilizando o agente treinador para detectar a dinâmica das formações da equipa adversária utilizando metodologias como: *ILP – Inductive Logic Programming*, Árvores de Decisão, Regras de Associação e/ou Redes Neuronais.
- Extensão dos conceitos do *Optimization Kick* para outros comportamentos de baixo nível como o drible ou a intercepção. Utilização de outras metodologias de resolução de problemas de optimização como o arrefecimento simulado e os algoritmos genéticos na resolução destes problemas, e comparação dos resultados com os obtidos através do *Optimization Kick*.
- Aplicação de técnicas de aprendizagem no contexto do futebol robótico simulado (desde as acções de baixo-nível: drible, chuto e intercepção, até à camada estratégica: definição e adaptação da táctica em cada jogo);
- Estudo de generalização das metodologias de coordenação desenvolvidas através da sua aplicação a outros domínios, nomeadamente as outras ligas do RoboCup

(robôs pequenos, robôs médios, robôs com pernas e RoboCup Rescue);

- Construção de um agente capaz de receber a análise de alto nível de um jogo de futebol simulado (fornecida pelo treinador adjunto) e gerar um comentário inteligente em tempo-real, utilizando linguagem natural e síntese de voz. Interligação deste agente com o visualizador tridimensional desenvolvido, construindo um sistema integrado de visualização, análise e comentário.

Refira-se que se encontram actualmente em curso projectos de investigação, resultantes desta dissertação e nos quais o autor se encontra envolvido no LIACC, conducentes à realização de grande parte destas perspectivas de desenvolvimento. Destes projectos destaca-se o projecto *Portus – A Common Framework for Cooperative Mobile Robotics*, *FC Portugal – New Coordination Methodologies applied to the Simulation League* e *LEMAS – Learning in Multi-Agent Systems in the Sony Legged League*. A realização do campeonato do mundo de futebol robótico – RoboCup 2004 – em Lisboa (em Junho de 2004) permitirá efectuar a avaliação destes projectos utilizando como domínio de teste as diversas ligas do futebol robótico.

Referências Bibliográficas

[Abowd et al., 1996] Abowd, G., Engeslma, J., Guadagno L. e Okon, O. *Architectural Analysis of Object Request Brokers,* Object Magazine, pp. 44-51, 98, Março de, 1996

[Abramson, 1991] Abramson, D., *Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms,* Management Science, Vol. 37(1), pp. 98-113, 1991

[Abramson, 1993] Abramson, D. and Dang, H., *School Timetables: A Case Study Using Simulated Annealing,* Applied SA, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, pp.104-124, 1993

[ADE, 2001] Adesoft, "ADE", França, 2001, [Em linha]. Disponível em http://www.timetabling.com/ [Consultado em 30/10/2001]

[Agorics, 1996] Agorics, 1996 [Em linha]. Disponível em http://www.agorics.com/ [Consultado em 30/11/2001]

[AHD, 2000] Pickett, Joseph P. et al. editores, *The American Heritage® Dictionary of the English Language,* 4th edition, Boston, Houghton Mifflin, 2000

[Allen et al. 1990] Allen, J. F., Hendler, J. e Tate, A. editores, *Readings in Planning,* Morgan Kaufmann, San Mateo, CA, 1990

[Aloimonos et al., 1998] Aloimonos, Y., Weiss, I. e Bandopadhay, A. *Active Vision,* International Journal of Computer Vision, Vol. 1(3) pp. 333-356, 1988

[Asada e Kitano, 1999] Asada, M. e Kitano, H. editores. *RoboCup-98: Robot Soccer World Cup II,* Springer, Lecture Notes in Artificial Intelligence, Vol. 1604, 1999

[Austin, 1962] Austin, J. L. *How to do Things with Words,* Clarendon, Oxford, U.K., 1962

[Axelrod, 1984] Axelrod, R., *The Evolution of Cooperation,* Basic Books, New York, 1984

[Azevedo e Barahona 1994] Azevedo, Francisco e Barahona, Pedro, *Timetabling in Constraint Logic Programming,* Proceedings of World Congress on Expert Systems, Mexico, 1994

[Bajcsy, 1988] Bajcsy, R. *Active Perception,* Proceedings of the IEEE, Vol. 76(8) pp. 996-1005, Agosto de 1988

[Balch e Arkin, 1994] Balch, T. e Arkin, R. C. *Communication in Reactive Multi-Agent Robotic Systems,* Autonomous Robots, Vol. 1 (1), pp. 27-53, 1994

[Barber, 1997] Barber, R. and Forster, G., *The Syllabus Plus Exam Scheduler: Design Overview and Initial Results,* Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling PATAT97, Toronto, pp.72-80, 1997

[Barr e Feigenbaum, 1981] Barr, A. e Feigenbaum, A. *The Handbook of Artificial Intelligence,* Vol. 1, Los altos, Morgan Kaufmann, 1981

[Beer et al., 1998] Beer, M., Inverno, M., Luck, M., Jennings, N., Preist, C., Schoeder, M., *Negotiation in Multi-Agent Systems,* Workshop of the Special Interest Group on Multi-Agent Systems (UKMAS'98), U.K., 1998

[Benson et al., 1987] Benson, J., Kennedy, G., McMillan, J. *Managing Negotiations,* Hutchinson Business Books Limited, 1987

[Bercovitch, 1984] Bercovitch, J. *Problems and Approaches in the Study of Bargaining and Negotiation. Political Science,* Vol. 36 (2), pp. 125-145, 1984

[Birk et al., 2002] Birk, Andreas; Coradeschi, S. e Takadoro, S., *RoboCup 2001: Robot Soccer World Cup V.* Springer LNAI 2377, Berlin, 2002

[Bloomfield e Mcsarry, 1979] Bloomfield, S. D. e McSharry, M. M., *Preferential Course Scheduling,* Interfaces, vol. 9, nº 4, pp. 24-31, The Institute of Management Sciences, Agosto de 1979

[Bond e Gasser, 1988a] Bond, Alan H. e Gasser, Les, *Readings in Distributed Artificial Intelligence,* Morgan Kaufmann Publishers Inc., San Mateo, CA, 1988.

[Bond e Gasser, 1988b] Bond, Alan H. e Gasser, Les, *An Analysis of Problems and Research in DAI,* Em [Bond e Gasser, 1988a], pp. 3-35, Morgan Kaufmann Publishers Inc., San Mateo, CA, 1988.

[Booch, 1994,] Booch, G., *Object-Oriented Analysis and Design,* 2nd, Edition, Addison-Wesley, Reading, MA, 1994

[Bradshaw, 1997] Bradshaw, J. editor, *Software Agents,* MIT Press, Cambridge, MA, 1997

[Brandt et al., 2000] Brandt, Felix; Brauser, Wilfried e Weiβ, Gerhard, *Task Assignment in Multiagent Systems based on Vickrey-Type Auctioning and Levelled Commitment Contracting,* Fourth International Workshop on Cooperative Information Agents, Boston, USA, 2000.

[Bratman, 1990] Bratman, Michael E. *What is Intention?* Intentions in Communication, Editado por Phillip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, pp. 15-31. MIT Press, Cambridge, MA, 1990.

[Bratman, 1992] Bratman, Michael E., *Planning and the Stability of Intentions,* Minds and Machines, Vol. 2, pp. 1-16, 1992

[Bronson, 1997] Bronson, Gary J., *Program Development and Design Using C++,* PWS Publishing Company; 1997.

[Brooks, 1986] Brooks, Rodney. *A Robust Layered Control System for a Mobile Robot,* IEEE Journal of Robotics and Automation, pp. 14-23, 1986

[Brooks, 1991] Brooks, Rodney. *Intelligence Without Reason,* Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), pp. 569-595, Morgan Kaufmann, Sydney, 1991

[Bruce a tal., 2001] Bruce, J.; Balch, T.; Veloso, M., *CMVision,* [Em Linha] Disponível em http://www.coral.cs.cmu.edu/vision, [Consultado em 12/06/2002]

[Bruce et al., 2000] Bruce, J.; Balch, T.; Veloso, M., *Fast and Inexpensive Color Image Segmentation for Interactive Robots,* IROS-2000, Japan, October 2000.

[Burke et al. 1998] Burke, E.; Kingston, J.; Pepper, P. *A Standard Data Format for Timetabling Instances,* Practice and Theory of Automated Timetabling II, LNCS, Vol. 1408, pp. 213-222, 1998

[Cai et al., 2002] Cai, Yunpeng; Chen, J.; Yao, J.; Li, S. *Global Planning from Local Eyeshot: An Implementation of Observation-Based Plan Coordination in RoboCup Simulation Games,* Em [Birk et al., 2002], pp. 12-21, 2002

[Cardoso e Oliveira, 2000] Cardoso, Henrique L. e Oliveira, Eugénio, *A Platform for Electronic Commerce with Adaptive Agents,* AA'2000, in Agent-Mediated Electronic Commerce III, Frank Dignum and Ulises Cortés (eds.), LNAI 2003, pp. 96-107, Springer-Verlag, 2000

[Cardoso, 1999] Cardoso, Henrique L., *Sistema Multi-Agente para Comércio Electrónico*, Tese de Mestrado em Inteligência Artificial e Computação (FEUP, FCUP e FEP), 1999

[Carpenter et al., 2003] Carpenter, P., Riley, P., Veloso, M. e Kaminka, G. *Integration of Advice in an Action-Selection Architecture*, RoboCup-2002: Robot Soccer World Cup VI, edited by Gal Kaminka, Pedro Lima e Raul Rojas, Springer, LNAI, 2003

[Carter e Laporte 1996] Carter, M.; Laporte G., *Recent Developments in Practical Examination Timetabling*, Practice and Theory of Automated Timetabling II, LNCS, Springer Verlag, Vol. 1408, pp. 3-21, 1998

[Castelfranchi, 1995] Castelfranchi, Cristiano, *Commitments: From Individual Intentions to Groups and Organizations,* Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pp. 41-48, Menlo Park, California, Junho de 1995

[Celcat, 2001] CELCAT, *"CELCAT"*, Inglaterra, 2001, [Em linha]. Disponível em http://www.celcat.com/ [Consultado em 30/10/2001]

[Chandrasekaran, 1981] Chandrasekaran, B., *Natural and Social System Metaphors for Distributed Problem Solving: Introduction to the Issue*, IEEE Transactions on S.M.C., 1981

[Chavarria, 2002] Chavarria, Manuel, *Um Estudo dos Métodos e Programas para Construção de Horários Escolares em Portugal,* Monografia em Informática de Gestão, Universidade Fernando Pessoa, Portugal, Fevereiro de 2002

[Chavez e Maes, 1996] Chavez, A., e Maes, P., *Kasbah: An Agent Marketplace for Buying and Selling Goods,* Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96), pp. 75-90, 1996

[Ciancarini, 1996] Ciancarini, P. *Coordination Models and Languages as Software Integrators*, ACM Computing Surveys, Vol. 28, No. 2, pp. 300-302, Junho de 1996

[Coelho, 1994] Coelho, Hélder, *Inteligência Artificial em 25 Lições,* Fundação Calouste Gulbenkian, 1994

[Cohen e Levesque, 1990] Cohen, P. R. e Levesque, H. J., *Intention is Choice with Commitment*, Artificial Intelligence, Vol. 42, pp.213-261, 1990

[Cohen e Levesque, 1991] Cohen, P. R. e Levesque, H. J., *Teamwork*, Nous, Vol. 25(4), pp. 487-512, 1991

[Cohen et al., 1990] Cohen, P. R.; Levesque, H. J.; Nunes, J. H.; Oviatt, S. L. *Task-Oriented Dialogue as a Consequence of Joint Activity,* Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Japanese Society for Artificial Intelligence, pp. 203-208, Nagoya, Japan, Novembro de 1990

[Collingwood et al. 1996] Collingwood, E.; Ross, P.; Corne, D., *A Guide to GATT*, University of Edinburgh, 1996

[Colmerauer 1975] Colmerauer, A., *Les Grammaires de Metamorphose,* Technical Report, Marseille: Groupe d'Intelligence Artificielle, Université de Marseille-Luminy, 1975

[Colorni et al. 1992] Colorni, A.; Dorigo, M.; Maniezzo, V., *A Genetic Algorithm to Solve the Timetable Problem*, Technical Report. 90-060 revised, Politecnico di Milano, Italy, Computacional Optimization and Applications, 1992

[Cooper e Kingston 1993] Cooper, T. e Kingston, J., *The Solution of Real Instances of the Timetabling Problems*, The Computer Journal, Vol. 36, pp. 645-653, 1993

[Costa et al., 2000] Costa, P.; Marques, P.; Moreira, A.P.; Sousa, A.; Costa, P., *Tracking and Identifying in Real Time the Robots of a F-180 Team,* in [Veloso et al., 2000], pp. 286-291, 2000

[Cumming e Paechter 1995] Cumming, A. e Paechter, B., *Seminar: Standard Timetabling Data Format*, Em Edmund Burke, Dave Corne, Ben Paetcher and Peter Ross, First International Conference on The Practice and Theory of Automated Timetabling – PATAT'95, Napier, University, Edinburgh, UK,1995

[Cunha, 1999] Cunha, Pedro, *Estratégias e Tácticas em Negociação: Para um Modelo de Eficácia Negocial.* Tese de Doutoramento. Santiago de Compostela, Espanha, 1999

[DAML, 2001] The *DARPA Agent Markup Language*, 2001, [Em linha]. Disponível em http://www.daml.org, [Consultado em 20/5/2002]

[de Boer et al., 2001] de Boer, R.; Kok, J.; Groen, F., *UvA Trilearn 2001 Team Description,* em [Birk et al., 2002], pp.551-554, 2002

[de Kleer, 1986] de Kleer, Johan, *Problem Solving with the ATMS*, Artificial Intelligence, Vol. 28, Nº2, 1986

[Decker e Lesser, 1995] Decker, K. e Lesser, V., *Designing a Family of Coordination Algorithms,* Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pp. 73-80, San Francisco, CA, 1995

[Decker, 1995] Decker, Keith, *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD Thesis, University of Massachusetts, 1995

[Decker, 1996] Decker, Keith, *TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms,* em [O'Hare e Jennings, 1996], Chapter 16, pp. 429-447, 1996

[Deep Blue, 1997] Deep Blue, [Em Linha] Disponível em: http://www.research.ibm.com/ deepblue/ [Consultado em 15/02/2001]

[Deitel e Deitel, 1999] Deitel, H.M e Deitel, P.J., *Java How To Program.* Prentice-Hall. New Jersey; 1999.

[Demazeau e Muller, 1990] Demazeau, Y. e Muller, J. P. *Decentralized Artificial Intelligence*, Elsevier Science Publishers, 1990

[Deutsch, 1973] Deutsch, M. *The Resolution of Conflict, Constructive and Destructive Processes.* New Haven, CT: Yale University Press, 1973

[Deutsch, 1990] Deutsch, M. *A Framework for Teaching Conflict Resolution in the Schools,* Research on Negotiation in Organizations, Editado por B. H. Sheppard, M. H. Bazerman e R. J. Lewicki (editores.), pp. 189-203, Greenwich, CT, JAI Press, 1990

[deWerra 1985] deWerra, D., *An Introduction to Timetabling*, European Journal of Operational Research, 19, pp. 151-162, 1985

[Dias, 1999] Dias, Ricardo, *Sistemas de Distribuição de Serviço Docente em Universidades,* Monografia em Informática de Gestão, Universidade Fernando Pessoa, Portugal, Julho de 1999

[Dorer, 2000] Dorer, Klaus, *The Magma Freiburg Soccer Team,* em [Veloso et al., 2000], pp.600-603, 2000

[Douglas, 1962] Douglas, A., *Industrial Peacemaking*. Col. University Press, New York, 1962

[Durão, 1998] Durão, D.; Conceição, P.; Heitor, M.; Santos, F.,. *Novas Ideias para a Universidade,* Instituto Superior Técnico, IST Press, 1998

[Durfee e Lesser, 1987] Durfee, E. H. e Lesser, V., *Using Partial Global Plans to Coordinate Distributed Problem Solvers,* Proceedings of the 10th International Joint Conference on Artificial Intelligence, IJCAI'87, Milão, Itália, pp. 875-883, 1987

[Durfee et al., 1989] Durfee, E. H.; Lesser, V.; Corkill, D., *Cooperative Distributed Problem Solving,* Handbook of Artificial Intelligence, Edited by E. Feigenbaum, A. Barr, e P. Cohen, Vol. 4, pp. 83-147, Addison-Wesley, Reading MA, 1989

[Durfee, 1988] Durfee, E. H., *Coordination of Distributed Problem Solvers,* Kluwer Academic, Boston, MA, 1988

[Durfee, 1996] Durfee, E. H., *Planning in Distributed Artificial Intelligence*, em [O'Hare e Jennings, 1996], pp. 231-245, 1996

[Durfee, 1999] Durfee, E. H., *Distributed Problem Solving and Planning*, em [Weiss, 1999], pp. 121-164, 1999

[Even et al., 1976] Even, S.; Itai A.; Shamir, A., *The Complexity of Timetabling and MultiCommodity Flow Problems"*, SIAM Journal of Computation 5, pp. 691-703, 1976.

[Fahrion e Dollanski, 1992] Fahrion, R. e Dollanski G., *Construction of University Faculty Timetables Using Logic Programming Techniques,* Discrete Applied Mathematics, Vol. 35, pp. 221-236, 1992

[Faratin et al., 1998] Faratin, P.; Sierra, C.; Jennings, N., *Negotiation Decision Functions for Autonomous Agents*, International Journal of Robotics and Autonomous Systems, 24 (3-4), pp. 159-182, 1998

[Faratin et al., 1999] Faratin, P.; Sierra, C.; Jennings, N.; Buckle, P., *Designing Responsive and Deliberative Automated Negotiators.* Proceedings AAAI Workshop on Negotiation : Settling Conflicts and Identifying Opportunities, Orlando, FL, pp. 12-18, 1999

[Faria, 2000] Faria, Pedro, *Um Sistema de Construção Automática de Horários em Universidades, utilizando o Algoritmo do Arrefecimento Simulado,* Tese de Mestrado em Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, Novembro de 2000

[Feldman e Mehra., 1993] Feldman, R e Mehra, R. *Auctions Theory and Applications,* IMF Staff Papers, pp. 485-511, 1993

[Ferber e Drogoul, 1992] Ferber, Jacques e Drogoul, Alexis, *Using Reactive Multi-Agent Systems in Simulation and Problem Solving,* Distributed Artificial Intelligence: Theory and practice, Gasser, L. e Avouris, N., editors, Kluwer, Academic Publishers, 1992

[Ferber, 1999] Ferber, Jacques, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, Longman Inc., New York, 1999

[Férnandez-Ríos, 1986] Fernández-Ríos, M. *Génesis y Dinámica del Conflicto Laboral: Aportaciones a un Modelo Multidimensional y Sistémico* (I). Revista de Psicología del Trabajo y de las Organizaciones, 2 (3), pp. 10-26, 1986

[FIFA, 2001] FIFA, *Laws of the Game* [Em Linha] Disponível em http://www.fifa.com, 2000 [Consultado em 12/02/2001]

[Finin et al., 1993] Finin, T. et al., *Specification of the KQML Agent Communication Language,* DARPA Knowledge Sharing Initiative External Interfaces Working Group, 1993

[FIPA, 1999] *FIPA Agent Communication Language Overview*, Foundation for Intelligent Physical Agents, 1999 [Em Linha] Disponível em http://www.fipa.org [Consultado em 10/12/2001]

[Fisher et al., 1993] Fisher, R.; Ury, W.; Patton, B., *Como Conduzir uma Negociação? Como Negociar um Acordo sem Desistir*. Porto, Edições Asa, 1993

[Foner, 1993] Foner, L. *What is an Agent, Anyway? A Sociological Case Study,* Agents Memo, Media Lab, MIT, pp. 93-101, 1993

[Fonseca, 2000] Fonseca, José M., *Protocolos de Negociação com Coligações em Sistemas Multi-Agente: Uma Aplicação à Gestão Distribuída de Recursos*, Tese de Doutoramento, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Portugal, 2000

[Frank, 2001] Frank, Ian et al., *And The Fans are Going Wild! SIG plus MIKE,* em [Stone et al., 2001a], pp. 139-148 , Berlin, 2001

[Franklin e Graesser, 1996] Franklin, S. e Graesser, A. *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents,* Proceedings of the Third International Workshop on Agent Theories, Springer-Verlag, 1996.

[Friedberg, 1995] Friedberg, E., *O Poder e a Regra – Dinâmicas da Acção Organizada*, Lisboa: Instituto Piaget, 1995

[Garey e Johnson, 1979] Garey, M. R. e Johnson, D. S., *Computers and Intractability – A Guide to NP-Completeness,* W. H. Freeman and Company, San Francisco, USA, 1979.

[Gasser et al., 1987] Gasser, Les; Braganza, Carl; Herman, Nava, *MACE: A Flexible Testbed for Distributed AI Research,* em Huhns, M. editor, Distributed Artificial Intelligence, Pitman Publishers, 119-152, 1987

[Gasser, 1992] Gasser, Les, *DAI Approaches to Coordination,* Distributed Artificial Intelligence: Theory and Praxis, Avouris, N. e Gasser, L. editores, Kluwer Academic, Publishers, Boston, pp. 31-51, 1992

[Genesereth e Fikes, 1992] Genesereth, M. R. e Fikes, R. E., *Knowledge Interchange Format, Version 3.0 Reference Manual,* Technical Report, Computer Science Department, Stanford University, 1992

[Genesereth e Ketchpel, 1994] Genesereth, M. R. e Ketchpel S. P., *Software Agents*, Communications of the ACM, 37(7), pp. 48-53, Julho de 1994

[Genesereth et al., 1986] Genesereth, M.; Ginsberg, M.; Rosenschein, J., *Cooperation without Communication,* Proceedings of the 5th National Conference on Artificial Intelligence, AAAI'86, Philadelphia, PA, pp. 51-57, 1986

[GestHor, 2001] Cronológica, *"GestHor"*, Portugal, 2001, [Em Linha] Disponível em http://www.cronologica.pt/ [Consultado em 30/10/2001]

[Glover e Laguna 1993] Glover, F. e Laguna, M., *Tabu Search,* em Reeves, C. R. (Ed.), Modern Heuristic Techniques for Combinatorial Problems. Scientific Publications, Oxford, 1993

[Gorry e Morton1971] Gorry, G. M. e Morton, Scott, *A Framework for Management Information Systems*, Sloan Management Review, 1971

[GPUntis, 2001] Cardita – Análise Informática, Lda, *"GP-Untis"*, Portugal, 2001, [Em Linha]. Disponível em http://www.cardita.com/ [Consultado em 30/10/2001]

[Greenhalgh, 1987] Greenhalgh, L., *Interpersonal Conflicts in Organizations,* International Review of Industrial and Organizational Psychology, pp. 229-271, 1987

[Grosof, 1995] Grosof, B., *Conflict Handling in Advice Taking and Instruction*, Technical Report 20123, IBM Research, 1995

[Grosz e Kraus, 1996] Grosz, Barbara e Kraus, Sarit, *Collaborative Plans for Complex Group Actions*, Artificial Intelligence, Vol. 86(2), , pp. 269-358, 1996

[Grosz e Sidner, 1990] Grosz, Barbara e Sidner, Candace, *Plans of Discourse*, em Cohen, Phil; Morgan J. e Pollack, Martha, editores, Intentions in Communications, MIT Press, Cambridge, MA, pp. 417-444, 1990

[Grosz, 1996] Grosz, Barbara, *Collaborating Systems*, AI Magazine, Vol. 17(2), pp. 67-85, 1996

[Gruber, 1993] Gruber, T. R., *Toward Principles for Design of Ontologies Used for Knowledge Sharing*, Formal Ontology in Conceptual Analysis and Knowledge Representation, Guarino, Nicola e Poli, Roberto, editors, Kluwer Academic Publishers, 1993.

[Grüninger e Fox, 1995] Fox, M. S. e Grüninger, M., *Methodology for the Design and Evaluation of Ontologies*, Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAR-95, Montreal.

[Gudes et al. 1990] Gudes, Ehud; Tsvi, Kuflik; Amnon, Meisels, *On Resource Allocation By an Expert System*, Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, pp. 84-105, Israel, 1990

[Guerét et al. 1995] Guerét, Christelle; Jussien, Narendra; Boizumault, Patrice; Prins, Christian, *Building University Timetables using Constraint Logic Programming*, em Edmund Burke, E.; Corne, D.; Paetcher, B. e Ross, P. editores, First International Conference on the Practice and Theory of Automated Timetabling, PATAT'95, pp. 393-408, Edinburgh, 1995

[Hannenbauer et al., 2001] Hannenbauer, Markus, Wendler, Jan e Pagello, Enrico, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems – A Short Guide to the Contributions*, Balancing Reactivity and Social Deliberation in Multi-Agent Systems: From RoboCup to Real-World Applications, Springer Verlag LNAI 2103, pp. 175-197, Berlin, 2001

[Hayes-Roth et al., 1995] Hayes-Roth, B.; Brownston, L.; Gen, R. V., *Multi-Agent Collaboration in Directed Improvisation*, Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, 1995

[Hayes-Roth, 1995] Hayes-Roth, Barbara, *An Architecture for Adaptive Intelligent Systems*, Artificial Intelligence: Special Issue on Agents and Interactivity, Vol. 72, pp. 329-365, 1995

[Hendler, 2001] Hendler, J., *Agents and the Semantic Web*, IEEE Intelligent Systems, Vol. 16(2), pp. 30-37, 2001

[Hertz, 1992] Hertz, A., *Finding a Feasible Course Schedule using Tabu Search*, Dis. Applied Maths, Vol.35, pp.225-270, 1992

[Hewitt, 1986] Hewitt, C. E., *Offices are Open Systems*, ACM Transactions on Office Information Systems, Vol. 4 (3), pp. 271-287, 1986

[Hindle, 1998] Hindle, T., *Como Realizar Negociações*. Livraria Civilização Editora, Porto, 1998

[Holland 1975] Holland, John H., *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975

[Huhns e Singh, 1997a] Huhns, M. N. e Singh, M. P., editores, *Readings in Agents*, Morgan Kaufmann Publishers, 1997

[Huhns e Singh, 1997b] Huhns, M. N., e Singh, M. P., *Ontologies for Agents,* IEEE Internet Computing, pp. 81-83, November-Dezembro de 1997

[Huhns e Stephens, 1999] Huhns, Michael e Stephens, Larry M., *Multiagent Systems and Societies of Agents*, em Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, editado por Gerhard Weiss, The MIT Press, Cap 2, pp. 121-164, 1999

[IBM, 1997] *IBM's Intelligent Agent Strategy White Paper,* Intelligent Business Machines Inc., 1997

[INSPIRE, 2001] *Inspire Negotiation System,* [Em Linha] Disponível em http://www.iiasa.ac.at/Research/DAS/interneg/ [Consultado em 15/02/2001]

[Ishida et al, 1990] Ishida, T.; Yokoo, M.; Gasser, Les, *An Organisational Approach to Adaptive Production Systems,* Proceedings of the 8th National Conference on Artificial Intelligence, Boston, pp. 52-58, 1990

[Jaffar e Lassez, 1987] Jaffar, Joxan e Lazzez, Jean-Louis, *Constraint Logic Programming*, Proceedings 14th ACM Symposium on Principles of Programming Languages, pp. 111-119, Munich. ACM, 1987

[Jaffar e Lassez, 1994] Jaffar, Joxan e Lazzez, Jean-Louis, *Constraint Logic Programming – A Survey*, Journal of Logic Programming, 1994

[JatLite, 2001] *JATLite – Overview*, Stanford University, EUA, [Em Linha] Disponível em http://java.santford.edu/ [Consultado em 15/02/2001]

[Jennings et al., 1998] Jennings, Nick; Sycara, Katia; Wooldridge, Michael, *A Roadmap of Agent Research and Development,* Autonomous Agents and Multi-Agent Systems, Vol. 1, pp. 7-38, Kluwer, Academic Publisher, 1998

[Jennings et al., 2000] Jennings, N.; Parsons, S.; Sierra, C.; Faratin, P., *Automated Negotiation.* Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems (PAAM-2000), Manchester, UK, pp. 23-30, 2000

[Jennings, 1993] Jennings, Nick, *Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems,* The Knowledge Engineering Review, Vol. 8(3), pp. 223-250, 1993

[Jennings, 1995] Jennings, Nick, *Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions.* Artificial Intelligence, Vol. 75, 1995

[Jennings, 1996] Jennings, Nick, *Coordination Techniques for Distributed Artificial Intelligence,* em [O'Hare e Jennings, 1996], Chapter 6, pp. 187 -210, 1996

[Jennings, 2000] Jennings, Nick R., *On Agent-Base Software Engineering,* Artificial Intelligence, Vol. 117, pp.277-296, 2000

[Jesuino, 1998] Jesuino, J., *A Negociação – Estratégias e Tácticas.* Porto Editora, 3ª Edição, Lisboa, 1998

[Jini, 2001] *Jini Technology – Overview,* SUN Microsystems, [Em Linha] Disponível em http://www.sun.com/jini/ [Consultado em 15/02/2001]

[Kaminka, 2000] Kaminka, G. A. *Execution Monitoring in Multi-Agent Environments* PhD Thesis, University of Southern California, Computer Science Department, 2000

[Kang e White, 1992] Kang, L. e White, G., *A Logic Approach to the Resolution of Constraints in Timetabling,* European Journal of Operational Research, Vol.61, pp.306-317, 1992

[Kennedy, 1961] Kennedy, John F., *Urgent National Needs.* Congressional Record - House (25 de Maio de 1961), 1961

[Kingston, 1999] Kingston, Jeffrey H., *Modelling Timetabling Problems with STTL*, Basser Department of Computer Science, The University of Sydney 2006, Australia, 1999

[Kirkpatrick et al. 1983] Kirkpatrick, S.; Gelatt, C.; Vecchi, M., *Optimization by Simulated Annealing*, Science, Vol. 220 (4598), pp. 671-680, 1983

[Kitano et al., 1995] Kitano, H.; Asada, M. ; Kuniyoshi, Y.; Noda, I.; Osawa E., *Robocup: The Robot World Cup Initiative.* Proceedings of IJCAI'95 Workshop on Entertainment and AI/Alife, pp. 19-24, 1995

[Kitano et al., 1997] Kitano, H.; Tambe, M.; Stone, P.; Veloso, M.; Noda, I.; Osawa, E.; Asada, M., *The RoboCup Synthetic Agents' Challenge.* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997

[Kitano et al., 1999] Kitano, H. et al., *RoboCup-Rescue: Search and Rescue for Large Scale Disasters as a Domain for Multi-Agent Research,* Proceedings of IEEE Conference on Man, Systems, and Cybernetics (SMC-99), 1999

[Kitano, 1997] Kitano, Hiroaki, *RoboCup: The Robot World Cup Initiative,* Proceedings of the 1st International Conference on Autonomous Agent (Agents'97), Marina del Ray, The ACM Press, 1997.

[Kitano, 1998] Kitano, Hiroaki, editor, *RoboCup-97: Robot Soccer World Cup I.* Springer Verlag, Berlin, 1998

[Knapik e Johnson, 1998] Knapik, M. e Johnson, J., *Developing Intelligent Agents for Distributed Systems.* Computing McGraw-Hill, NY. McGraw-Hill, 1998

[Kowalski, 1979] Kowalski, R., *Logic for Problem Solving*, New York, North-Holland, 1979

[Labrou e Finin 1994] Labrou, Yannis e Finin, Tim, *A Semantics Approach for KQML – A General Purpose Communication Language for Software Agents*, em Proceedings of International Conference on Information and Knowledge Management, 1994

[Labrou e Finin 1997] Labrou, Yannis e Finin, Tim, *A Proposal for a New KQML Specification*, Computer Science and Electrical Engineering Department (CSEE), University of Maryland Baltimore County, 1997

[Lau e Reis, 2000] Lau, Nuno e Reis, Luis Paulo, *FC Portugal Most Interesting Research: Overview*, [Em Linha] Disponível em http://www.ieeta.pt/robocup/documents/ FCPortugalInteresting.ps.zip, Novembro de 2000

[Lau e Reis, 2001] Lau, Nuno e Reis, Luis Paulo, *FC Portugal 2001: Low Level Skills and World State Update*, LIACC (NIAD&R) – Research Report, Outubro de 2001

[Lau e Reis, 2002] Lau, Nuno e Reis, Luis Paulo, *FC Portugal 2001 Team Description: Configurable Strategy and Flexible Teamwork*, em [Birk et al., 2002], pp. 515-518, 2002

[Laudon e Laudon 1997] Laudon, Kenneth C. e Laudon, Jane Price, *Fundamentals of Management Science*, 6th edition, Burr Ridge, 1994

[Lawrie, 1969] Lawrie, N., *An Integer Programming Model of a School Timetabling Problem,* The Computer Journal, Vol. 12, pp. 307-316, 1969

[Lemon et al., 1996] Lemon, B.; Pynadath, D.; Taylor, G.; Wray, B., *Cognitive Architectures,* [Em Linha], Disponível em Http://krusty.eecs.umich.edu/cogarch4/toc.html, 1996. [Consultado em 15/01/2002]

[Lenting e Braspenning, 1994] Lenting, Jacques e Braspenning, Peter, *An All-Pay Auction Approach to Reallocation,* Proceedings of the 11th European Conference on Artificial Intelligence, ECAI'94, John Wiley & Sons, pp. 259-263, 1994

[Lesser e Corkill, 1981] Lesser, V. e Corkill, D., *Functionally Accurate, Cooperative Distributed Systems,* IEEE Transactions on Systems, Man and Cybernetics, Vol. 11(1), pp. 81-96, 1981

[Lesser e Corkill, 1987] Lesser, V. e Corkill, D., *Distributed Problem Solving*, Encyclopedia of Artificial Intelligence, Editada por Shapiro, S., pp. 245-251, Wiley, New York, 1987

[Lesser, 1998] Lesser, Victor, *Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture*, Autonomous Agents and Multi Agent-Systems, Vol. 1, pp. 89-111, Kluwer, Academic Publisher, 1998

[Lesser, 1999] Lesser, Victor. *Cooperative Multi-Agent Systems: A Personal View of the State of the Art,* IEEE Transactions on Knowledge and Data Engineering, Vol. 11, Nº 1, Janeiro/Fevereiro de 1999

[Levesque et al, 1990] Levesque, Hector J.; Cohen, Philip R.; Nunes, Jose, *On Acting Together,* Proceedings of the 8th National Conference on Artificial Intelligence, Morgan Kaufmann Publishers, Inc., pp. 94-99, Boston, Agosto de 1990

[Lewicky et al., 1992] Lewicky, R.; Weiss, S.; Lewin, D., *Models of Conflict, Negotiation and Third Party Intervention: A Review and Synthesis,* Journal of Organizational Behavior, 13, 209-252, 1992

[Lomuscio et al., 2000] Lomuscio, A.; Wooldridge, M.; Jennings, N., *A Classification Scheme for Negotiation in Electronic Commerce,* Agent-Mediated Electronic Commerce: A European AgentLink Perspective, Springer-Verlag, pp. 19-33, 2000

[Lopes et al., 1999] Lopes, Luís Seabra; Lau, Nuno; Reis, Luís Paulo, *D. Dinis: um Robô com Sentido de Orientação*, Electrónica e Telecomunicações, 2 (6), Universidade de Aveiro, pp. 811-813, 1999

[Lopes et al., 2000] Lopes, Luís Seabra; Lau, Nuno; Reis, Luís Paulo, *Intelligent Control and Decision-Making demonstrated on a Simple Compass-Guided Robot,* SMC2000 - Proc. of the 2000 IEEE International Conference on Systems, Man and Cybernetics, Nashville, Tennessee

[Louro et al., 2003] Louro, Sérgio; Reis, Luis Paulo; Oliveira, Eugénio, *Virtual 3D: A Multi-Agent System for Visualizing RoboCup Simulation League Games with Intelligent Camera Control,* em Gal Kaminka, Pedro Lima e Raul Rojas editores, RoboCup-2002: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Berlin, 2003

[Macedo, 2001] Macedo, Ana Paula Rocha, *Metodologias de Negociação em Sistemas Multi-Agentes para Empresas Virtuais.* Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Portugal, 2001

[Mackworth, 1993] MackWorth, Alan, *On Seeing Robots,* Computer Vision: System, Theory, and Applications, pp. 1-13, World Scientific Press, Singapore, 1993

[Maes, 1996] Maes, Pattie, *Intelligent Software: Programs That Can Act Independently Will Ease the Burdens that Computers Put on People,* IEEE Expert Systems, Vol. 11, No. 6, pp. 62-63,. Dezembro de 1996

[Malheiro e Oliveira, 2000] Malheiro, Benedita e Oliveira, Eugénio, *Solving Conflicting Believes with a Distributed Belief Revision Approach,* Advances in Artificial Intelligence, International Joint Conference IBERAMIA'2000/SBIA'2000, LNAI 1952, pp. 146-155, Springer, Novembro de 2000

[Malheiro, 1999] Malheiro, Benedita, *Metodologias de Revisão de Crenças em Sistemas Multi-Agente,* Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Portugal, 1999

[Malone e Crowston, 1991] Malone, Thomas e Crowston, Kevin, *Towards an Interdisciplinary Theory of Coordination,* Technical Report, 120, Cambridge, MA, MIT, Center for Coordination Science, 1991

[Malone e Crowston, 1994] Malone, Thomas e Crowston, Kevin, *The Interdisciplinary Study of Coordination,* ACM Computing Surveys, Vol. 26 (1), pp. 87-119, Março de 1994

[Malone, 1987] Malone, T. W., *Moddeling Coordination in Organizations and Markets*, Management Science, 33, Nº10, pp. 1317-1332, 1987

[Malucelli, 2002] Malucelli, Andreia, *Segundo Relatório de Progresso*, Faculdade de Engenharia da Universidade do Porto, Porto, Maio de 2002

[Mark e Snyder, 1957] Mack, R. W. e Snyder, R. C., *The Analysis of Social Conflict: Toward an Overview and Synthesis*, Journal of Conflict Resolution, Vol. 1(4), pp. 212-248, 1957

[Marsella et al., 2001] Marsella, S.; Tambe, M.; Adibi, J.; Yaser, A.; Kaminka, G.; Muslea, I. *Experiences Acquired in the Design of RoboCup Teams: A Comparison of Two Fielded Teams,* Autonomous Agents and Multi-Agent Systems, Vol. 4, pp. 115-129, Kluwer Acad. Pub. 2001

[MatchAnalysis, 2002] Match Analysys System HomePage [Em Linha] Disponível em: http://www.matchanalysis.com/ [Consultado em 15/02/2002]

[Matos e Sierra, 1998] Matos, Noyda e Sierra, Carlos, *Evolutionary Computing and Negotiating Agents.* Workshop on Agent Mediated Electronic Trading, 2nd International Conference on Autonomous Agents, Minneapolis/St. Paul, USA, 1998

[McCarthy, 1958] McCarthy, *Programs with Common Sense*, Proceedings of the Symposium on Mechanization of Thought Processes, pp. 77-81, 1958

[Merke, 2002] *FrameView HomePage* [Em Linha] Disponível em http://ls1-www.cs.uni-dortmund.de/~merke/software/frameview/index.html [Consultado em 15/02/2002]

[Milgrom, 1989] Milgrom, P., *Auctions and Bidding: A Primer,* Journal of Economic Perspectives, Vol. 3, pp. 3-22, 1989

[Mimosa, 2001] Mimosa Software, "MIMOSA", Finlândia, 2001, [Em Linha], Disponível em http://www.mimosasoftware.com/ [Consultado em 30/10/2001]

[Mintzberg, 1979] Mintzberg, H., *The Structuring of Organization,* New Jersey, Prentice-Hall Inc., 1979

[Mitchell, 1997] Mitchell, Tom, *Machine Learning*, McGraw Hill, 1997

[Moore, 1990] Moore, R. C., *A Formal Theory of Knowledge and Action*, em [Allen et al., 1990], pp. 480-519, 1990

[Moreira et al., 2002] Moreira, António Paulo; Costa, Paulo; Sousa, Armando; Reis, Luís Paulo, *5dpo 2001 Team Description – Middle Size League*, Proceedings of the Fifth International Workshop on RoboCup, Fukuoka, Japão, pp. 24-25, Junho de 2002

[Moulin e Chaib-Draa, 1996] Moulin, Bernard e Chaib-Draa, Brahim, *An Overview of Distributed Artificial Intelligence,* em [O'Hare e Jennings, 1996], Chapter 1, pp. 3 -56, 1996

[Mouta, 1996], Mouta, Fernando A., *Ambiente de Desenvolvimento de Sistemas Multiagente para Controlo e Supervisão de Processos em Aplicações Distribuídas,* Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Portugal, 1996

[Mowbray, 1995] Mowbray, T., *Essentials of Object-Oriented Architecture*, Object Magazine, pp. 28-32, Setembro de 1995

[Muller, 1996] Muller, H., *Negotiation Principles,* em [O'Hare e Jennings, 1996], Chapter 7, pp. 211-230, 1996

[Nash, 1950] Nash, John, *Non-Cooperative Games,* PhD Thesis, Mathematics Department, Princeton University, 1950

[Naur 60] NAUR, Peter editor, *Revised Report on the Algorithmic Language ALGOL 60,* Communications of the ACM, Vol. 3 No.5, pp. 299-314, May 1960.

[Neves e Oliveira, 1997] Neves, Maria C. e Oliveira, Eugénio, *A Control Architecture for an Autonomous Mobile Robot,* Proceedings of First International Conference on Autonomous Agents (AA'97) in Marina d'el Rey -California- USA - February 1997

[Neves, 1999] Neves, Maria C., *Arquitectura Híbrida, Distribuída e Adaptável para o Controlo de um Robô Móvel: Metodologia e Implementação*, Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Portugal, 1999

[Newall 1999] Newall, J. P., *Hybrid Methods for Automated Timetabling*, PhD Thesis, University of Nottingham, 1999

[Nilsson, 1998] Nilsson, Nils, *Artificial Intelligence: A New Synthesis,* Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998

[Noda e Stone, 2002] Noda, Itsuki e Stone, Peter, *The RoboCup Soccer Server and CMUnited Clients: Implemented Infrastructure for MAS Research,* Autonomous Agents and Multi-Agent Systems, 2002

[Noda et al., 1998] Noda, Itsuki; Matsubara, Hitoshi; Hiraki, Kazuo; Frank, Ian, *Soccer Server: A Tool for Research on Multiagent Systems.* Applied Artificial Intelligence, Vol. 12, pp.233-250, 1998

[Noda, 1995] Noda, Itsuki, *Soccer Server: A Simulator of Robocup,* Proceedings of AI symposium '95 Japanese Society for Artificial Intelligence, pp. 29-34, 1995

[Norman et al., 1998] Norman, Timothy J.; Sierra, Carlos; Jennings, Nick R., *Rights and Commitment in Multi-Agent Agreements,* 3rd International Conference on Multi-Agent Systems, Paris, France, 1995

[Noy e McGuinness, 2001] Noy, N. F. e McGuinness, D. L., *Ontology Development: A Guide to Creating Your First Ontology,* Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, Março de 2001

[NSF-IRIS, 1989] *Coordination Theory and Collaboration Technology Workshop Summary,* Washington DC, National Science Foundation, Junho de 1991

[Nunes e Oliveira, 2002] Nunes, Luís e Oliveira, Eugénio, *Cooperative Learning using Advice Exchange,* Adaptive Agents and Multi-Agent Systems, Proceedings of the Second Symposium on Adaptive Agents and Multi-Agent Systems, Imperial College, London, April 2002, (Adaptive Agents and Multi-Agent Systems, Springer, Vol .2636, pp. 33-48, Springer 2003)

[Nwana et al., 1996] Nwana, H. S.; Lee, L.; Jennings, N.R., *Coordination in Software Agent Systems,* BT Technology Journal, Vol. 14 (4), pp. 79-88, 1996

[Nwana, 1996] Nwana, Hyacinth, *Software Agents: An Overview*, Knowledge Engineering Review, Vol. 11, N. 3, pp. 1-40, Setembro de 1996

[O'Hare e Jennings, 1996] Editado por O'Hare, G. e Jennings, N. *Foundations of Distributed Artificial Intelligence,* John, Wiley & Sons, 1996

[Oliveira e Rocha, 2000] Oliveira, Eugénio e Rocha, Ana Paula, *Agents Advanced Features for Negotiation in Electronic Commerce and Virtual Organisations Formation Process,* European

Perspectives on Agent Mediated Electronic Commerce, LNAI Vol. 1991, Springer Verlag, Junho de 2000

[Oliveira et al., 1992] Oliveira, Eugénio; Camacho, Rui; Qiegang, L. *ARCHON: Architecture for Cooperative Multi-Agent Systems* (Chapters 2, 6, 7), Editado por Wittig, Thies, Ellis Horwood, 1992

[Oliveira et al., 1997] Oliveira, Eugénio; Fonseca, José M.; Steiger-Garção, Adolfo, *MACIV: A DAI Based Resource Management System,* Applied Artificial Intelligence Journal, Vol. 11(6), pp. 525-550, Taylor & Francis, Setembro de 1997

[Oliveira et al., 1999] Oliveira, Eugénio; Fonseca, José M.; Steiger-Garção, Adolfo, *Multi-Criteria Negotiation on Multi-Agent Systems.* First International Workshop of Central and Eastern Europe on Multi-Agent Systems. St. Petersburg, Russia, 1999

[Oliveira, 1984] Oliveira, Eugénio, *Engenharia do Conhecimento: Sistemas Periciais e Prolog*, Tese de Doutoramento, Universidade Nova de Lisboa, Portugal, 1984

[Oliveira, 2001] Oliveira, Eugénio, *Agents' Advanced Features for Negotiation and Coordination,* Proceedings of 9th ECCAI Advanced Course, ACAI'01, Editado por. M. Luck, V. Marik, O. Stepankova, R. Trappl, LNAI 2086, Springer Verlag, Junho de 2001

[Parsons et al., 1998] Parsons, Simon; Sierra, Carlos; Jennings, Nick, *Agents that Reason and Negotiate by Arguing,* Journal of Logic and Computation Vol. 8(3), pp. 261-292, 1998

[Parsons et al., 2001] Parsons, S.; Jennings, N.; Lomuscio, A.; Sierra, C.; Wooldridge, M., *Automated Negotiation: Prospects, Methods and Challenges.* International Journal of Group Decision and Negotiation, Vol. 10 (2), pp. 199-215, 2001

[PATAT, 2000] PATAT2000 - Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, Konstanz, Germany, Agosto, 16-18, 2000

[Pereira, 2000] Pereira, António, *Formação de Empresas Virtuais com Agentes Inteligentes*, Tese de Mestrado em Inteligência Artificial e Computação (FEUP, FCUP e FEP), Portugal, Dezembro de 2000

[Prasad e Lesser, 1999] Prasad, M. e Lesser, V., *Learning Situation-Specific Coordination in Cooperative Multi-Agent Systems,* Autonomous Agents and Multi-Agent Systems, Vol. 2, pp. 173-207, Kluwer Academic Publishers, 1999

[Prokopenko e Butler, 1999] Prokopenko, M. e Butler, M., *Tactical Reasoning in Synthetic Multi-Agent Systems: a Case Study.* Proceedings of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action and Change, pp. 57-64, Estocolomo, 1999

[Pruitt e Gahagan, 1974] Pruitt, D. G., e Gahagan, J. P., *Campus Crisis: The Search for Power,* em Tedeshi, J. T., editor, Perspectives on Social Power, pp. 349-392, Chicago, 1974

[Pruitt e Rubin ,1986] Pruitt, D. G. e Rubin, J. Z., *Social Conflict, Escalation, Stalemate, and Settlement,* New York. Random House, 1986

[Pruitt, 1981] Pruitt, D. G., *Negotiation Behavior,* Academic Press. New York, 1981

[Rao e Georgeff, 1991] Rao, Anand S. e Georgeff, Michael P., *Modeling Rational Agents within a BDI-Architecture,* Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, USA, 1991.

[Rao e Georgeff, 1995] Rao, Anand S. e Georgeff, Michael P., *BDI-Agents: From Theory to Practice,* Proceedings of the First International Conference on MultiAgent Systems (ICMAS-95), San Francisco, California, EUA, pp. 312-319, Junho de 1995

[Rao et al., 1993] Rao, A. S.; Lucas, A.; Morley, D.; Selvestrel, M.; Murray, G., *Agent-Oriented Architectures for Air-Combat Simulation,* Australian Artificial Intelligence Institute Technical Report, 1993

[Reis e Lau, 2001] Reis, Luís Paulo e Lau, Nuno, *FC Portugal Approach Overview,* [Em Linha] Disponível em Http://www.ieeta.pt/robocup/overview.htm 2001

[Reis e Lau, 2001a] Reis, Luís Paulo e Lau, Nuno, *FC Portugal Team Description: RoboCup 2000 Simulation League Champion*, em Peter Stone, Tucker Balch and Gerhard Kraetzschmar, editors, RoboCup-2000: Robot Soccer World Cup IV, Springer Verlag Lecture Notes in Artificial Intelligence, Vol. 2019, pp.29-40, Berlin, 2001

[Reis e Lau, 2001b] Reis, Luís Paulo e Lau, Nuno, *How to Give Intelligence to Soccer Playing Agents,* [Em Linha] Disponível em Http://www.fe.up.pt/~lpreis/ Individual.ps.zip, Janeiro de 2001

[Reis e Lau, 2001c] Reis, Luís Paulo e Lau, Nuno, *Coach Unilang*, [Em Linha] Disponível em Http://www.fe.up.pt/~lpreis/ Coach_Unilang.ps.zip, Janeiro de 2001

[Reis e Lau, 2001d] Reis, Luís Paulo e Lau, Nuno, *FCPortugal Home Page* [Em Linha] Disponível em Http://www.ieeta.pt/robocup, Janeiro de 2001

[Reis e Lau, 2002] Reis, Luís Paulo e Lau, Nuno, *COACH UNILANG – A Standard Language for Coaching a (Robo) Soccer Team*, em Andreas Birk, Silvia Coradeschi and Satoshi Tadokoro, editors, RoboCup-2001: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Vol. 2377, pp.183-192, Berlin, 2002

[Reis e Lau, 2003a] Reis, Luís Paulo e Lau, Nuno, *FC Portugal 2002 Team Description: Flexible Coordination Techniques* em Gal Kaminka, Pedro Lima e Raul Rojas editors, RoboCup-2002: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Berlin, 2003

[Reis e Lau, 2003b] Reis, Luís Paulo e Lau, Nuno, *FC Portugal 2002 Coach: High-Level Coaching of RoboSoccer Games* em Gal Kaminka, Pedro Lima e Raul Rojas editors, RoboCup-2002: Robot Soccer World Cup V, Springer Verlag Lecture Notes in Artificial Intelligence, Berlin, 2003

[Reis e Oliveira, 1999a] Reis, Luís Paulo e Oliveira, Eugénio, *A Constraint Logic Programming Approach to Examination Scheduling*, AICS'99 - 10th Irish Conference on Artificial Intelligence & Cognitive Science, Cork, Ireland, 1-3 Setembro, 1999

[Reis e Oliveira, 1999b] Reis, Luís Paulo e Oliveira, Eugénio, *Constraint Logic Programming using Set Variables for Solving Timetabling Problems*, INAP'99 - 12th International Conference on Applications of Prolog, Tokyo, Japão, 3-5 Setembro, 1999

[Reis e Oliveira, 2000a] Reis, Luís Paulo e Oliveira, Eugénio, *Examination Timetabling using Set Variables*, PATAT2000 - Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, pp.181-183, Konstanz, Germany, Agosto 16-18, 2000

[Reis e Oliveira, 2000b] Reis, Luís Paulo e Oliveira, Eugénio, *A Language for Specifying Complete Timetabling Problems,* PATAT2000 – Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling, pp.456-471, Konstanz, Germany, Agosto 16-18, 2000

[Reis e Oliveira., 2001] Reis, Luís Paulo e Oliveira, Eugénio, *A Language for Specifying Complete Timetabling Problems,* em Edmund Burke and Wilhelm Erben editors, Practice and Theory of Automated Timetabling III, Third International Conference, PATAT 2000, Konstanz, Germany, August 2000, Selected Papers, Springer Verlag LNCS, Vol. 2079, pp. 322-341, Berlin, 2001

[Reis e Rocha, 1993] Reis, Luís Paulo e Rocha, Manuela, *Um Sistema Baseado em Conhecimento para a Geração de Horários no Ensino Básico,* Faculdade de Engenharia da Universidade do Porto, Portugal, 1993

[Reis et al., 1999] Reis, Luís Paulo; Teixeira, Paulo; Oliveira, Eugénio, *Examination Timetabling Using Constraint Logic Programming*, ECP'99 - 5th European Conference on Planning, Durham, Inglaterra, 8-10 Setembro de 1999

[Reis et al., 2000] Reis, Luís Paulo; Lau; Nuno; Lopes, Luis Seabra, *FC Portugal Team Description Paper: RoboCup 2000 Qualifying*, [Em Linha] Disponível em Http://www.ieeta.pt/robocup/ documents/ FCPortugalTeamDescription.ps.zip, Maio de 2000

[Reis et al., 2001] Reis, Luís Paulo; Lau; Nuno; Oliveira, Eugénio C., *Situation Based Strategic Positioning for Coordinating a Team of Homogeneous Agents* em Markus Hannenbauer, Jan Wendler and Enrico Pagello editores, Balancing Reactivity and Social Deliberation in Multi-Agent System – From RoboCup to Real-World Applications, Springer's Lecture Notes in Artificial Intelligence, Vol. 2103, pp. 175-197, Berlin, 2001

[Reis, 1995] Reis, Luís Paulo, *Sistema de Detecção e Seguimento de Contornos*, Dissertação de Mestrado, Faculdade de Engenharia da Universidade do Porto, Setembro de 1995

[Reis, 1997] Reis, Luís Paulo, *Programação em Lógica com Restrições – Sistemas e Técnicas,* LIACC (NIAD&R) – Research Report, Março de 1997

[Reis, 1999] Reis, Luís Paulo, *Metodologias de Geração de Horários,* LIACC (NIAD&R) – Research Report, Dezembro de 1999

[Reis, 2000] Reis, Luís Paulo, *How to Give Intelligence to Football Playing Agents,* LIACC (NIAD&R) – Research Report, Abril de 2000

[Reis, 2001a] *RoboCup Tutorial – The RoboCup International Initiative: AI Meets Robotics to Create RoboSoccer Teams,* editado por Reis, Luís Paulo, integrado no EPIA2001 – 10th Portuguese Conference on Artificial Intelligence, Seminário de Vilar, Porto, 20 de Dezembro de 2001

[Reis, 2001b] Reis, Luís Paulo, *How to Win in Robotic Football? Strategy, Cooperation and Communication to build an Artificial Football Team,* WIAA2001, Instituto Superior Técnico - Lisboa, Março de 2001

[Reis, 2001c] Reis, Luís Paulo, *Flexible Teamwork and Configurable Strategy for Robotic Soccer Competitions,* SWERC 2001, SouthWestern European Regional ACM Programming Contest, Auditório da FEUP, 17 de Novembro de 2001

[Reis, 2001d] Reis, Luís Paulo, *Problemas de Escalonamento na Gestão Universitária,* LIACC (NIAD&R) – Research Report, Novembro de 2001

[Reis, 2002a] Reis, Luís Paulo, *Ciber-FEUP – Ensino de Robótica e Inteligência Artificial através da Participação em Competições Robóticas,* Electrónica e Telecomunicações, Universidade de Aveiro, Portugal, 2002

[Reis, 2002b] Reis, Luís Paulo, *Agentes Autónomos,* LIACC (NIAD&R) – Research Report, Março de 2002

[Reis, 2002c] Reis, Luís Paulo, *Sistemas Multi-Agente,* LIACC (NIAD&R) – Research Report, Abril de 2002

[Reis, 2002d] Reis, Luís Paulo, *Coordenação de Agentes Competitivos: Negociação e Resolução de Conflitos,* LIACC (NIAD&R) – Research Report, Maio de 2002

[Reis, 2002e] Reis, Luís Paulo, *Coordenação de Agentes Cooperativos e Trabalho de Equipa,* LIACC (NIAD&R) – Research Report, Maio de 2002

[Reis, 2002f] Reis, Luís Paulo, *Coordenação Estratégica – Aplicação no Futebol Robótico Simulado,* LIACC (NIAD&R) – Research Report, Junho de 2002

[Reis, 2002g] Reis, Luís Paulo, *Estratégias e Tácticas no Futebol Robótico,* Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, 6ª Semana das Engenharias, Dia da Engenharia Electrotécnica, Auditório Eng. Alcínio Miguel, 15 de Abril de 2002

[Rescue, 2001] RoboCup Rescue Oficial Page, [Em Linha] Disponível em Http://www.r.cs.kobe-u.ac.jp/robocup-rescue/ [Consultado em 18/02/2002]

[Riedmiller et al., 2000] Riedmiller. Martin, et al., *Karlsruhe Brainstormers 2000 - A Reinforcement Learning Approach to Robotic Soccer.* Proceedings of the Fourth International Workshop on RoboCup. Melbourne, Agosto de 2000

[Robbins, 1987] Robbins, S. P., *Comportamiento Organizacional,* Prentice-Hall, 1987

[RoboCup, 2001] *The Goals of RoboCup.* by RoboCup Federation, [Em Linha] Disponível em Http://www.robocup.org/overview/22.html, 2000, [Consultado em 18/02/2002]

[Rocha e Oliveira, 2001a] Rocha, Ana Paula e Oliveira, Eugénio, *Electronic Institutions as a Framework for Agents' Negotiation and Mutual Commitment,* Progress in Artificial Intelligence (Proceedings of 10th EPIA), Eds. P. Brazdil, A. Jorge, LNAI 2258, pp. 232-245, Springer, Dezembro de 2001

[Rocha e Oliveira, 2001b] Rocha, Ana Paula e Oliveira, Eugénio, *Adaptive Multi-Issue Negotiation Protocol for Electronic Commerce,* Proceedings of the Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000), Manchester, UK, 10-12 Abril de 2000

[Rosenschein e Zlotkin, 1994] Rosenschein, S. e Zlotkin, G., *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers,* MIT Press, Cambridge, MA, 1994

[Russel e Norvig, 1995] Russel, Stuart e Norvig, Peter, *Artificial Intelligence – A Modern Approach,* Prentice Hall Series in Artificial Intelligence, 1995

[Sandholm, 1999] Sandholm, Tuomas, W. *Distributed Rational Decision Making,* em [Weiss, 1999], Cap. 5, pp. 201-258, 1999

[Schaerf e Schaerf 1995] Schaerf, A. e Schaerf, M., *Local Search Techniques for High School Timetabling* em Edmund Burke, Dave Corne, Ben Paetcher and Peter Ross, First International Conference on the Practice and Theory of Automated Timetabling, pp. 440-446, Napier University, Edinburgh, 1995

[Schaerf, 1995] Schaerf, Andrea, *A Survey of Automated Timetabling,* Centrum Voor Wiskunde en Informatica, Computer Science/Department of Software Technology, 1995

[Searle, 1969] Searle, J. R., *Speech Acts: An Essay in the Philosophy of Language,* Cambridge University Press, 1969

[Sen e Weiss, 1999] Sen, Sandip e Weiss, Gerhard, *Learning in Multiagent Systems,* em [Weiss, 1999], pp. 259-298, 1999

[Serrano, 1996] Serrano, G., *Elogio de la Negociación,* Universidade de Santiago de Compostela, Santiago de Compostela, 1996

[Shoham e Tennenholtz, 1992] Y. Shoham e M. Tennenholtz, *On the Synthesis of Useful Social Laws for Artificial Agent Societies,* Proceedings of the 10th National Conference on Artificial Intelligence, San José, EUA pp. 276-281, 1992

[Shoham, 1993] Shoham, Y., *Agent-Oriented Programming.* Artificial Intelligence, Vol. 60, pp. 51-92, 1993

[Sichman e Demazeau, 1995] Sichman, J. S. e Demazeau, Y., *Exploiting Social Reasoning to Deal with Agency Level Inconsistency,* Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, pp. 352-359, 1995

[Sichman, 1994] Sichman, J. S., *A Social Reasoning Mechanism Based on Dependence Networks,* Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94), Amsterdam, pp. 188-192, 1994

[Silva, 2000a] Silva, Pedro Nuno, *Sistema Multi-Agente para Geração de Horários no Ensino Secundário,* Tese de Mestrado em Inteligência Artificial e Computação (FEUP, FCUP e FEP), Portugal, Dezembro de 2000

[Silva, 2000b] Silva, Rui Camacho, *Inducing Models of Human Control Skills using Machine Learning Algorithms,* Tese de Doutoramento, Faculdade de Engenharia da Universidade do Porto, Portugal, 2000

[Silva, 2002] Silva, Paula, *Gestão Pedagógica Universitária em Portugal: Gestão Universitária e Alocação de Recursos,* Monografia em Informática de Gestão, Universidade Fernando Pessoa, Portugal, Fevereiro de 2002

[Singh, 1990] Singh, Munindar, *Group Intentions,* Proceedings of the 10th International Workshop on Distributed Artificial Intelligence (DAI-10). MCC Technical Report ACT-AI-355-90, 1990

[Singh, 1992] Singh, B., *Interconnected Roles (IR): A Coordination Model,* Technical Report CT084-92), Austin, TX: Microelectronics and Computer Technology Corp. (MCC), 1992

[Singh, 1997] Singh, Munindar, *Considerations on Agent Communication,* FIPA Workshop, 1997

[Sklar et al., 2002] Sklar, Elizabeth; Eguchi, Amy; Johnson, Jeffrey, *RoboCupJunior: Learning with Educational Robotics,* RoboCup-2002: Robot Soccer World Cup VI, Springer Verlag, Berlin, 2003

[Smith e Davis, 1980] Smith, R. G. e Davis, R. *Frameworks for Cooperation in Distributed Problem Solving,* IEEE Transactions on System, Man and Cybernetics, Vol.11(1), 1980

[Smith et al., 1994] Smith, David C.; Cypher, Allen; Spohrer, Jim, *KidSim: Programming Agents Without a Programming Language,* Communications of the ACM, Vol. 37(7), pp. 54-67, Julho de 1994

[Smith, 1977] Smith, R. G., *The Contract Net: A Formalism for the Control of Distributed Problem Solving,* Proceedings of the 5th International Joint Conference on Artificial Intelligence, IJCAI'77, Cambridge, MA, 1977

[Smith, 1980a] Smith, R. G., The *Contract Net Protocol,* IEEE Transactions on Computers, Vol. 29(12), 1980

[Smith, 1980b] Smith, R. G., *A Framework for Distributed Problem Solving,* UMI Research Press, 1980

[Soccerserver, 2001] Chen, M.; Foroughi, E.; Heintz, S.; Kapetanakis, S.; Kostiadis, K.; Kummeneje, J.; Noda, I.; Obst, O.; Riley, P.; Steffens, T.; Wang, Y.,; Yin, X., *RoboCup Soccer Server manual for Soccer Server version 7.07 or Latest*, [Em Linha], Disponível em Http://sourceforge.net/projects/sserver [Consultado em 10/07/2002]

[Stamatopoulos et al. 1998] Stamatopoulos, P.; Viglas, E.; Karaboyas, S., *Nearly Optimum Timetable Construction Through CLP and Intelligent Search*, International Journal on Artificial Intelligence Tools, Vol. 7, No. 4, pp. 415-442, 1998

[Stone e Veloso, 1996] Stone, Peter e Veloso, Manuela, *Multi-Agent Systems: A Survey from a Machine Learning Perspective*, IEEE Transactions on Knowledge and Data Engineering (TKDE), 1996

[Stone e Veloso, 1999] Stone, Peter e Veloso, Manuela, *Task Decomposition, Dynamic Role Assignment and Low-Bandwidth Communication for Real-Time Strategic Teamwork.* Artificial Intelligence, Vol. 110 (2), pp. 241-273, Junho de 1999.

[Stone et al., 1999] Stone, Peter; Riley, Patrick; Veloso, Manuela, *CMUnited-99 Source Code*, 1999. [Em Linha] Disponível em Http://www.cs.cmu.edu/~pstone/ RoboCup/ CMUnited99-sim.html [Consultado em 15/02/2000]

[Stone et al., 2000a] Stone, Peter; Riley, Patrick; Veloso, Manuela, *The CMUnited-99 Champion Simulator Team*, em [Veloso et al., 2000]

[Stone et al., 2000b] Stone, Peter; Riley, Patrick; Veloso, Manuela, *Layered Disclosure: Why is the Agent Doing What It's Doing?*, Agents 2000, Fourth International Conference on Autonomous Agents, Barcelona, Junho de 2000

[Stone et al., 2001a] Stone, Peter; Balch, Tucker; Kraetzschmar, Gerhard, editores, *RoboCup-2000: Robot Soccer World Cup IV*, Springer Verlag, Berlin, 2001

[Stone et al., 2001b] Stone, Peter et al. editores, *RoboCup-2000 The Forth Robotic Soccer World Championships*, AI magazine, Vol. 22, No. 1, pp. 11-38, 2001.

[Stone et al., 2001c] Stone, Peter et al. editors, *Overview of RoboCup-2000* in [Stone et al., 2001a], pp.1-28, 2001

[Stone, 1998] Stone, Peter, Layered *Learning in Multi-Agent System*, PhD thesis, School of Computer Science, Carnegie Mellon University, Dezembro de 1998.

[Stone, 2002] Stone, Peter, *Multiagent Competitions and Research: Lessons from RoboCup and TAC*, Sixth RoboCup International Symposium, Fukuoka, Japão, 2002

[TAC, 2001] *TAC – Trading Agent Competition*, 2001 [Em Linha]. Disponível em http://tac.eecs.umich.edu/ [Consultado em 20/12/2001]

[Tambe et al.,1999] Tambe, M., Adibi, J., Alonaizon, Y., Erdem, A., Kaminka, G., Marsella, S. and Muslea, I., *Building Agent Teams Using an Explicit Teamwork Model and Learning.*, Artificial Intelligence, Vol. 110, pp. 215-240, 1999

[Tambe, 1995] Tambe, Milind, *Recursive Agent and Agent Group Tracking in a Real-Time, Dynamic Environment*, Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, AAAI Press, pp. 368-375, Junho de 1995

[Tambe, 1997] Tambe, Milind, *Towards Flexible Teamwork*, Journal of Artificial Intelligence Research, Vol. 7, pp. 81-124, 1997

[Teixeira, 2002] Teixeira, Cláudia, *Negociação Convencional e Electrónica,* Monografia em Informática de Gestão, Universidade Fernando Pessoa, Portugal, Fevereiro de 2002

[Turban e Meredith 1994] Turban, Efrain; Meredith, J., *Fundamentals of Management Science*, 6th edition, Burr Ridge, 1994

[Uschold e Gruninger, 1996] Uschold, M. e Gruninger, M., *Ontologies: Principles, Methods and Applications,* Knowledge Engineering Review, Vol. 11 (2), pp. 93-136, 1996

[VanLehn, 1991] VanLehn, K. editor, *Architectures for Intelligence*, Lawrence Erlbaum Associates, New Jersey, 1991

[Veloso et al., 2000] Veloso, Manuela; Pagello, Enriço; Kitano, Hiroaki, editores. *RoboCup-99: Robot Soccer World Cup III,* Springer, Lecture Notes in Artificial Intelligence, 2000

[Verbraeck, 1990] Verbraeck, A., *A Decision Support System for Timetable Construction,* International Conference on Expert Planning Systems, Brighton, England, pp. 207-211, 1990

[Vickrey, 1961] Vickrey, W. *Counterspeculation, Auctions and Competitive Sealed Tenders*, Journal of Finance, Vol. 16, pp. 8-37, 1961

[Vulkans e Jennings, 1998] Vulkans, Nir e Jennings, Nicholas R., *Efficient Mechanisms for the Supply of Services in Multi-Agent Environments,* First International Conference on Information and Computation Economics. Charleston, South Carolina, USA, 1998

[Waltz, 1972] Waltz, D., *Generating Semantic Descriptions from Drawings of Scenes with Shadows*". Technical Report AI271, MIT, Massachusetts, 1972

[Weare, 1995] Weare, R., *Automated Examination Timetabling,* PhD Thesis, Department of Computer Science, University Nottingham, U.K., June 1995

[Weiss, 1996] Weiss, Gerhard, *Adaptation and Learning in Multi-Agent Systems: Some Re-marks and a Bibliography.* Em Weiss, G., e Sen, S. (Editores.) (1996). Adaptation and Learning in Multi-Agent Systems. Lecture Notes in Artificial Intelligence, Vol. 1042, pp. 1-21, Springer-Verlag, 1996

[Weiss, 1999] Weiss, Gerhard, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,* MIT Press, 1999

[Welsh e Powell, 1967] Welsh, D. and Powell M., *An Upper Bound for the Chromatic Number of a Graph and its Applications to Timetabling Problems,* Computer Journal, Vol. 10, pp. 85-86, 1967

[Wendler, 2001] Wendler, J. et al, *Fault-Tolerant Self-Localization by CBR*, in [Stone et al., 2001], pp. 259-268, 2001

[Werner, 1989] Werner, E., *Cooperating Agents: A Unified Theory of Communication and Social Structure,* Distributed Artificial Intelligence, Edited by Les Gasser e M. Huhns, Vol. 2, pp. 3-36, Morgan Kaufmann, Los Altos, CA/Pilman, Londres, 1989

[Wertheim, 2000] Wertheim, E., *Negotiations and Resolving Conflicts: An Overview.* [Em linha] Disponível em http://www.cba.neu.edu/~ewertheim [Consultado em 05/11/2001].

[Williamson et al, 1996] Williamson, M.; Sycara, K.; Decker, K., *Executing Decision-Theoretic Plans in Multi-Agents Environments,* Proceedings of the AAAI Fall Symposium on Plan Execution: Problems and Issues, 1996

[Winston, 1991] Winston, W. L., *Operations Research: Applications and Algorithms,* 2ª edição, PWS-Kent, Boston, Massachusetts, 1991

[Wittig et al., 1994] Wittig, T.; Jennings, N. R. e Mamdani, E. H., *ARCHON - A Framework for Intelligent Cooperation,* IEE-BCS Journal of Intelligent Systems Engineering - Special Issue on Real-time Intelligent Systems in ESPRIT, Vol. 3 (3), pp. 168-179, 1994

[Wooldridge e Jennings, 1994] Wooldridge, Michael J. e Jennings, Nick R., *Agent Theories, Architectures, and Languages: A Survey.* Workshop on Agent Theories, Architectures and Languages, 11th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, 1994

[Wooldridge e Jennings, 1995] Wooldridge, Michael J. e Jennings, Nick R., *Intelligent Agents: Theory and Practice.* The Knowledge Engineering Review, Vol.10 (2), pp.115-152, 1995.

[Wooldridge, 1994] Wooldridge, Michael, *Coherent Social Action,* Proceedings of the 11th European Conference in Artificial Intelligent – ECAI94, Amsterdam, pp. 279-283, 1994

[Wooldridge, 1999] Wooldridge, Michael, *Intelligent Agents.* em Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, editado por Gerhard Weiβ, MIT Press, pp.27-77, 1999

[Wooldridge, 2002], Wooldridge, Michael, *An Introduction to Multi-Agent Systems,* John Wiley & Sons, Ltd, 2002

[Wurman et al., 1998] Wurman, Peter R.; Wellman, Michael P.; Walsh, William E., *The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents.* International Conference on Autonomous Agents, Minneapolis, St. Paul, USA, 1998.

[XML, 2001] *The eXtensible Markup Language*, 2001, [Em Linha]. Disponível em http://www.xml.org, [Consultado em 20/5/2002]

[Yamamoto e Fujita, 2000] Yamamoto, T. e Fujita M.,. *A Quadruped Robot Platform with Basic Software for Robocup-99 Legged Robot League*, Proceedings of IROS'00, 2000.

[Yoshikawa et al. 1994] Yoshikawa, M.; Kaneko, K.; Nomura, Y.; Watanabe, M., *A Constraint-Based Approach to High-School Timetabling Problems: A Case Study*, Proceedings of the 12th National Conference on Artificial Intelligence AAAI'94, pp. 111-116, 1994

[Zlotkin e Rosenchein, 1994] Rosenschein, J. S. e Zlotkin, G., *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers* Boston: M1T Press, 1994

Anexo 1: Tipos de Leilões

Existem diversos tipos de leilões aplicáveis em situações distintas e com vantagens e desvantagens relativas. Entre os mais utilizados incluem-se o Leilão Inglês, Leilão Holandês (ou de preço descendente), Leilão de Propostas Seladas – 1° Preço (ou discriminatório), Leilão de Vickrey – 2° Preço, Leilão de Pagamento Generalizado, Leilão Duplo.

Leilão Inglês

O leilão Inglês [Milgrom, 1989] é também conhecido como leilão de preço crescente, onde as propostas são *"one sided"* e *abertas*, ou seja não são propostas seladas. Neste tipo de leilão, o leiloeiro (vendedor) começa pelo preço mínimo aceitável (base de licitação) para aquele bem, e solicita aos clientes ofertas mais altas. As ofertas vão se sucedendo até que nenhum possível comprador aumente o último valor oferecido. O leilão termina com a venda do bem ao cliente que tenha oferecido a última proposta (correspondente à quantia mais elevada).

Muitas vezes o leiloeiro dá início ao leilão sem revelar qual a quantia reservada para o preço mínimo aceitável, para desta forma evitar que se formem anéis, isto é, conjuntos de clientes que se juntam e concordam em não ultrapassar as ofertas uns dos outros, o que fará com que a oferta vencedora seja mais baixa.

Neste tipo de leilão, o leiloeiro tem um papel muito importante, uma vez que, através da sua voz ou tom de voz e da sua personalidade, pode manipular os clientes. Assim sendo pode aumentar o ritmo ou até mesmo ignorar as ofertas de determinados clientes, caso ele ache que este cliente faz parte de um anel.

No cálculo das suas propostas é usual que os compradores entrem em consideração não só com a sua própria avaliação do bem leiloado mas também com o histórico do leilão e com o seu conhecimento à-priori sobre os outros participantes no leilão. No entanto, a estratégia dominante neste leilão consiste em oferecer em cada instante o valor da melhor oferta até esse momento acrescida de uma pequena quantia até que o valor da avaliação pessoal seja atingido. Quando tal se verifica o agente não emite mais nenhuma proposta até ao final do leilão.

Leilão Holandês ou Descendente

O leilão Holandês teve a sua origem com os leilões de flores que, ainda hoje, são realizados na Holanda, daí a sua denominação de leilão Holandês. É também, bastante utilizado nas lotas de peixe. Este leilão é também conhecido como leilão de preço decrescente e, tal como o

leilão Inglês, também aqui as propostas são *abertas* e *"one sided"*. Aqui, a base de licitação começa por um preço muito elevado e vai baixando progressivamente até que um cliente reclame o bem como sendo seu. Quando se trata de leiloar mais do que um produto é possível acontecer que vários clientes reclamem o produto à medida que o preço vai descendo e, quando tal acontece, o primeiro vencedor paga o preço que reclamou enquanto os outros vão pagar um valor inferior. O leilão termina quando não houver mais bens para vender.

No tipo de leilão Inglês pode acontecer que o vencedor pague pelo bem muito menos do que a sua avaliação privada e, assim sendo o vendedor não recebe o preço máximo pelo seu produto. No leilão Holandês, o cliente com mais interesse no produto em leilão, não pode deixar para muito tarde para reclamar a sua oferta, pois pode perder o produto para outro cliente. Desta forma, significa que o cliente deve pagar pelo bem o equivalente ou acima do valor que este considerou para a sua avaliação privada.

Este leilão não permite grandes reflexões por parte dos compradores, uma vez que estes têm de reagir sob grande pressão. Daí resulta a sua utilização essencialmente na venda de produtos de baixo valor e fácil avaliação.

Leilão de Propostas Seladas – 1° Preço (ou Discriminatório)

O leilão de Propostas Seladas tem, logo à partida, uma característica que o distingue dos restantes que é o facto de as propostas serem apresentadas em envelopes fechados, o que faz com que as propostas dos compradores não sejam conhecidas entre eles. O comprador vencedor irá pagar pelo bem em leilão exactamente o mesmo que ofertou na proposta selada.

De uma forma geral, este formato de proposta selada tem duas partes distintas, que são, em primeiro lugar um período em que os participantes submetem as suas propostas, e em segundo lugar uma fase de resolução na qual as suas propostas são abertas e o vencedor é conhecido.

A denominação de 1° preço ou discriminatório advém do facto de num mesmo período estar a ser leiloado um bem, ou vários bens, respectivamente. No que diz respeito ao nome 1° preço este vem do facto que cada cliente faz a sua proposta secreta e ganha o cliente que mais ofereceu pelo bem, preço este que irá pagar por ele, mas no caso de vários bens em leilão, nem todos os clientes vencedores pagam a mesma quantia, daí que se apelide de discriminatório, isto porque as propostas seladas são extraídas, pelo leiloeiro, da mais alta para a mais baixa e atribuídas da mesma forma para os itens em leilão, até que o stock de produtos termine.

Neste tipo de leilão, no cálculo das suas propostas, é usual que os compradores entrem em linha de conta não só com a sua própria avaliação mas também com o seu conhecimento *a priori* sobre os restantes agentes participantes no leilão. Como tal, é normal que os

compradores façam uma oferta estratégica ligeiramente superior ao valor estimado como sendo o máximo dos restantes participantes.

Este tipo de leilão tem como vantagem óbvia a sua simplicidade e eficiência. Como desvantagens apresenta o facto de não existir interacção entre os licitadores e o facto do leiloeiro não ter qualquer garantia de obter o melhor valor para o produto leiloado.

Leilão de Vickrey – 2° Preço

À semelhança do tipo de leilão anterior, também aqui as propostas são seladas e cada cliente não tem conhecimento das ofertas dos restantes [Vickrey, 1961]. No entanto, neste tipo de leilão o item é vendido para o ofertante com a maior proposta, mas este apenas paga o preço da segunda proposta mais alta. Neste tipo de leilão existe sempre a questão de "porque não utilizar o leilão de 1° preço?", uma vez que, *a priori*, este realiza muito mais dinheiro. Na verdade não é assim porque no leilão de Vickrey ninguém fica com a impressão de ter pago demasiado pelo bem a ser leiloado pois é o segundo preço que vigora na hora do pagamento.

No leilão de Vickrey, a estratégia dominante consiste em oferecer a verdadeira avaliação do produto. Isto deve-se ao facto de que, se o agente oferecer um valor superior à sua avaliação pessoal (com o objectivo de vencer os restantes agentes) corre o risco de adquirir o produto com prejuízo pois a segunda melhor oferta pode também ser superior à sua avaliação. No caso de oferecer um valor inferior à sua avaliação apenas diminui as suas hipóteses de adquirir o produto sem que isso tenha qualquer influência no preço a pagar (definido pela segunda melhor oferta).

Leilão de Pagamento Generalizado

O leilão de pagamento generalizado tem um funcionamento muito semelhante ao leilão inglês, vencendo o licitador que realizar a oferta mais elevada. No entanto, no final do leilão, todos os licitadores entregam ao leiloeiro o valor da sua última oferta [Lenting e Braspenning, 1994] [Fonseca, 2000]. Embora este leilão pareça um pouco absurdo e muito penalizador para os compradores, destina-se essencialmente à venda de artigos de elevado valor por um preço muito reduzido. No final do leilão é usual que o vencedor pague um preço muito baixo mas que o leiloeiro fruto das diversas ofertas realizadas pelos participantes, receba uma quantia apreciável. Fonseca [Fonseca, 2000] denota que este tipo de leilão se assemelha à lotaria onde todos pagam uma pequena quantia na esperança de receber uma quantia elevada embora exista uma baixa probabilidade que tal aconteça.

Leilão Duplo

O leilão duplo [Feldman e Mehra, 1993], como o próprio nome indica, é do tipo *"double sided"*, isto é, permite que, ao mesmo tempo, sejam feitas propostas quer por parte de quem vende quer por parte de quem compra, da mais alta para a mais baixa, por parte dos vendedores, e da mais baixa para a mais alta, por parte dos compradores. As propostas são aceites por parte dos vendedores a qualquer instante do processo. Pode parecer um pouco confuso pensar no funcionamento deste tipo de leilão, uma vez que quer os possíveis vendedores quer os possíveis compradores se encontram a realizar propostas ao mesmo tempo. No entanto, se realizado por meios electrónicos, as ofertas de compra e venda são ordenadas por ordem temporal de chegada não se sobrepondo.

Desta forma, o leilão duplo é utilizado, principalmente, para compra e venda de produtos financeiros sendo desta forma, bastante utilizado em bolsa de valores.

Anexo 2: Vulnerabilidade dos Leilões

Os tipos de leilões apresentados no anexo 1 podem ser vulneráveis a coligações entre os compradores, compradores mentirosos, revelação indesejada de informação privada, leiloeiros mentirosos e anúncios fictícios.

Coligações entre Compradores

As coligações entre os clientes são uma das desvantagens que está presente em todos os tipos de leilões [Sandholm, 1999]. Os compradores podem unir-se, formando um anel, e baixar o valor das ofertas para a compra de um determinado item, ou seja, desta forma os compradores acabam por adquirir o bem por um preço mais baixo do que ele realmente vale.

Os leilões *Inglês* e de *Vickrey* proporcionam um melhor ambiente para a formação de coligações devido às suas características. O leilão *Inglês* por ser em público e com os clientes todos presentes, e o leilão de *Vickrey* por, apesar de ser em propostas seladas, qualquer que seja o valor mais alto, o preço a pagar pelo bem é o da segunda melhor oferta. Por estas razões, do ponto de vista de afastar a possível formação de coligações, os leilões de *Propostas Seladas – 1º Preço* e o *Holandês* são preferíveis.

Um exemplo ilustrativo deste problema, será: existe um comprador C1 que tem um valor privado sobre o bem em leilão de 20; os compradores C2 e C3 têm um valor privado de 18 e os restantes compradores apenas o lhe atribuem o valor 8. Neste caso, C1, C2 e C3 podem-se unir decidindo que C2 e C3 oferecerão no máximo 9 pelo bem, deixando a compra para C1. Se se tratar de um leilão *Inglês* pode-se observar o seu incentivo à formação das coligações, uma vez que se C2 ou C3 ultrapassar o valor definido de 9, C1 observará e subirá a sua oferta até próximo do seu valor privado de 20. Desta forma, o batoteiro não ganha nada com o facto de ter traído a coligação em que estava inserido. No caso do leilão de *Vickrey*, se C1 oferecer 20 e C2 e C3 oferecerem 9, C1 fica com o item e apenas paga 9. O que acontece é que o facto de C1 oferecer 20 desmoraliza os restantes compradores a abandonarem a coligação e oferecerem entre 9 e a sua avaliação privada, 18, uma vez que nenhuma destas ofertas ganharia o leilão.

Por outro lado, pode-se ver como não existe um ambiente tão propício para a formação de coligações nos leilões de *Propostas Seladas* e *Holandês*. No caso do leilão *de Propostas Seladas – 1º Preço*, se o comprador C1 oferecer abaixo do valor 18 os restantes compradores ficam incentivados a oferecer um valor mais alto para poderem ganhar o leilão, e desta forma saírem da coligação. O mesmo se passa com o leilão *Holandês*.

Uma diferença acompanha estes leilões, isto é, para que uma coligação se forme num leilão de *Vickrey*, de *Propostas Seladas – 1º Preço e Holandês*, os compradores têm de se identificar uns aos outros antes de submeterem as suas propostas, porque caso isto não aconteça um comprador que não seja membro da coligação pode ganhar o leilão. Nestes três leilões é também necessário que todos os agentes respeitem a coligação de forma a esta ter sucesso. No caso do leilão *Inglês* isto não acontece porque todos os possíveis compradores estão presentes no decorrer do leilão e caso algum desrespeite a coligação não ganhará nada com isso.

Embora a existência de coligações seja um perigo para o leiloeiro, este pode-se defender introduzindo especulação contra-coligação. Uma forma de o fazer será através da introdução de falsos licitadores para que estes causem uma competição mais dinâmica. No entanto, este comportamento tem também alguns riscos como seja o caso em que os próprios especuladores vencem o leilão e ficam com o produto leiloado.

Compradores Mentirosos em Leilões de Valor Não Privado

Na maioria dos leilões não se leva em conta apenas o valor privado de cada comprador, isto porque a avaliação por parte do comprador de um bem em leilão sofre influências vindas das avaliações dos restantes compradores [Sandholm, 1999].

Os leilões de valor comum sofrem sempre influências oriundas de outras experiências vividas pelos compradores. Se um comprador oferecer um valor correspondente à sua avaliação privada e ganhar o leilão, vai ficar sempre com a impressão que a sua avaliação privada era muito alta e por isso os outros compradores ofereceram menos, daí que ganhar um leilão corresponde a uma perda monetária. Sendo conhecedores deste facto os compradores oferecem menos que o correspondente à sua avaliação.

Revelação Indesejada de Informação Privada

Existe determinada informação que, depois de revelada pode ser utilizada de forma pouco correcta, como sendo o conhecimento por parte dos restantes compradores da avaliação privada de um outro comprador [Sandholm, 1999]. Quando se trata do leilão de propostas seladas, e sabendo que este tem como principal estratégia, em leilões de valor privado, a integridade das propostas por parte dos compradores, leva a que o comprador tenha que revelar a sua verdadeira avaliação do produto em causa, o que poderá constituir um "*handicap*" quando este quiser realizar uma negociação como titular do bem. Posto isto, os outros tipos de leilão poderão ser preferíveis para realizar negociações entre humanos.

Leiloeiro Mentiroso

Caso o leiloeiro não seja íntegro, pode mentir em relação às propostas recebidas simulando propostas de maior valor de forma a estimular a concorrência e induzir os participantes a oferecerem melhores propostas. A falta de sinceridade por parte do leiloeiro é um dos problemas que facilmente surge no leilão de *Vickrey* [Sandholm, 1999]. Caso o leiloeiro tenha interesse nisso, pode elevar a segunda melhor oferta para um valor próximo da mais alta, caso o comprador que fez a oferta não possa verificar isso. Esta possibilidade faz com que o leilão de *Vickrey* seja pouco usado na negociação convencional. No entanto, sempre que um participante suspeitar ser vítima de especulação por parte do leiloeiro, pode pedir a autenticação das ofertas que deram origem à melhor proposta, sendo os infractores penalizados pela tentativa de fraude. Esta possibilidade pode ser garantida, na negociação electrónica através da utilização de mensagens autenticadas através de assinaturas digitais [Fonseca, 2000].

No caso dos restantes leilões, o leiloeiro mentiroso não tem sucesso, uma vez que a oferta mais alta ganha o leilão e o comprador paga esse mesmo valor pelo bem. *Nos leilões de Propostas Seladas - 1º Preço* e *Vickrey*, o leiloeiro pode submeter a sua própria oferta, por forma a garantir que o item não será vendido por um valor inferior.

Anúncios Fictícios

Se o valor atingido no leilão não for satisfatório e caso não existam mecanismos que o impeçam, o titular do produto em leilão poderá cancelar a venda no final do leilão. Desta forma, e uma vez que este tipo de comportamento poderá levar ao descrédito do leilão, é necessário definir desde o início, as condições de venda, especificando o tipo de compromisso assumido pelo titular do produto e pelos compradores.

Anexo 3: *Workshop* e Prémios Científicos do RoboCup

A iniciativa RoboCup pretende desenvolver a investigação científica nas áreas da Inteligência Artificial e Robótica Distribuída. Desta forma, em conjunto com as competições oficiais e demonstrações, é organizado anualmente um *Workshop* com publicações científicas directa e indirectamente relacionadas com o futebol robótico. Este evento científico permite demonstrar os avanços científicos realizados ao longo do ano sem preocupações competitivas, sendo desta forma o complemento ideal do evento anual RoboCup. As áreas de interesse do *Workshop* incluem os Sistemas Multi-Agente e Multi-Robôs, coordenação e cooperação, controlo de sensores e actuadores, fusão sensorial, visão e processamento de imagem, localização e navegação, planeamento, raciocínio e modelação de oponentes, autonomia ajustável, aprendizagem e alocação de recursos (agentes heterogéneos).

O *Workshop* está ainda focalizado em áreas como sejam a simulação e visualização, robótica e computação para entretenimento, síntese de voz e geração de linguagem natural. O *Rescue* é outro domínio forte do *Workshop*, que inclui áreas como sejam a construção de robôs para resgate e salvamento ou a simulação de catástrofes. A educação é uma área de particular destaque pois a ênfase é colocada na utilização do RoboCup (nomeadamente do *RoboCup Júnior*) na educação a diversos níveis. Outros assuntos em destaque são a construção de robôs incluindo sistemas de visão e locomoção omnidireccionais, novos sensores e actuadores, baterias para robôs, mecanismos de locomoção com pernas e desenvolvimento de novos materiais (incluindo materiais "inteligentes").

Um factor relevante é que uma parcela significativa dos trabalhos apresentados no *Workshop* não é directamente aplicada na competição RoboCup ou são também aplicados em outros domínios.

Como forma de premiar os avanços científicos na área, a Federação RoboCup atribui dois prémios científicos no *Workshop*: O prémio científico e o prémio de engenharia. Estes prémios são atribuídos a trabalhos de uma qualidade excelente que constituam avanços científicos e tecnológicos relevantes no âmbito do RoboCup e áreas relacionadas. De referir que em 2000, Portugal conquistou o prémio de engenharia através do trabalho realizado por Carlos Marques e Pedro Lima no Instituto Superior Técnico da Universidade Técnica de Lisboa sobre Localização de Robôs Futebolistas utilizando visão omnidireccional [Marques, 2000], intitulado: *"A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor"*.

Anexo 4: O Árbitro Artificial

O árbitro artificial é um agente que consegue arbitrar um jogo de futebol robótico simulado de forma quase totalmente autónoma sem necessidade de auxílio de um árbitro humano. As faltas que são punidas com livres a favor de uma equipa ou bolas ao solo, e que devem ser assinaladas pelo árbitro, incluem:

1) Rodear a bola com demasiados jogadores;
2) Bloquear intencionalmente a baliza com demasiados jogadores;
3) Bloquear repetidamente e intencionalmente o movimento aos jogadores adversários;
4) Inundar o servidor de mensagens (cada jogador não deve enviar mais do que um comando de cada tipo ao simulador em cada ciclo);
5) A equipa ou alguns dos seus jogadores apresentarem um comportamento inapropriado (por exemplo, fazendo de propósito para perder o jogo ou lançando repetitivamente a bola para fora do campo);
6) Não repor a bola em jogo num tempo razoável aquando de um livre a favor da equipa;
7) Abusar do comando *catch* do guarda-redes seguido do comando *move* (o guarda-redes deve, após agarrar a bola efectuar um único *move* e repor a bola em jogo);
8) Atrasar a bola ao guarda-redes (o guarda-redes não pode agarrar a bola se o último jogador a chutá-lo tiver sido um colega de equipa)[93];
9) Sair com a bola controlada na marcação de um livre em vez de a passar a um colega[94].

Embora o árbitro artificial possua capacidades para garantir o cumprimento das regras 5 a 9, por vezes existem casos repetitivos em que o árbitro humano tem maior capacidade de análise. É o caso, por exemplo, de uma equipa que repetitivamente não recoloca a bola em jogo quando tem livres a seu favor. Neste caso, o árbitro artificial esperaria um determinado tempo pré-especificado antes de lançar a bola ao solo. No entanto um árbitro humano pode analisar melhor a situação verificar se algum jogador da equipa parece ter intenções de repor a bola em jogo e, caso contrário, lançar imediatamente a bola ao solo de forma a garantir uma maior fluidez do jogo.

---

[93] Regra introduzida pelo comité técnico em 2002

[94] Regra introduzida pelo comité técnico em 2002

Para além das regras mencionadas, o árbitro artificial é responsável por garantir o cumprimento das seguintes regras:

1) **Bola ao Centro.** Antes do início do jogo e após cada golo de uma das equipas, (modo *Kick_Off*) o árbitro garante que os jogadores de cada equipa se encontram na sua metade do campo antes de permitir o reinício do jogo. Para permitir isto, após a marcação de um golo, o árbitro suspende o jogo por 5 segundos, permitindo aos jogadores neste intervalo utilizar o comando *move* para se deslocarem directamente para um dado ponto na sua metade do campo. No caso de um dado jogador permanecer na metade oposta do campo, o árbitro artificial, antes do reinício do jogo, recoloca-o numa posição aleatória da sua metade do campo;

2) **Golo.** Após a marcação de cada golo, o árbitro envia uma mensagem a todos os jogadores, actualiza o resultado, recoloca a bola no centro do terreno e entra no modo de *kick_off* aguardando que os jogadores se reposicionem na sua metade do campo;

3) **Bola Fora do Campo.** Quando a bola sai do campo (cruza totalmente uma das quatro linhas que o limitam), o árbitro recoloca a bola num ponto apropriado (ponto onde cruzou a linha lateral, marca de canto ou marca de pontapé de baliza), altera o modo de jogo para *kick_in, corner_kick* ou *goal_kick* dependendo do ponto onde a bola cruzou a linha e de qual o último jogador que a chutou. Informa também todos os jogadores da ocorrência através de uma mensagem apropriada;

4) **Fora de Jogo.** Outra incumbência do árbitro artificial consiste em assinalar a ocorrência de foras de jogo (*offsides*) marcando um livre contra a equipa infractora. No futebol robótico simulado um fora de jogo é marcado quando, no momento de um passe, o jogador receptor se encontrar para além da linha de meio campo e do último defesa da equipa adversária (sem considerar o guarda-redes)[95];

5) **Afastamento de adversários do ponto de marcação de um livre.** Quando o modo de jogo muda para *kick_off, kick_in* ou *corner_kick*, o árbitro remove todos os jogadores que estejam localizados num círculo com um dado raio (por defeito 9.15m) em torno da bola, colocando-os no perímetro deste círculo. No caso de um *offside*, o árbitro coloca todos os jogadores que se encontravam fora de jogo para além do último defesa da equipa adversária. No caso de um pontapé de baliza (modo de jogo *goal_kick*), todos os jogadores adversários são deslocados para fora da grande-área para que o pontapé de

[95] Esta regra é ligeiramente diferente do fora de jogo no futebol real em que a infracção é assinalada no caso de existirem menos de dois jogadores adversários entre o jogador que recebe a bola e a linha final do campo independentemente de um deles ser o guarda-redes.

baliza possa ser realizado. O árbitro garante ainda que estes jogadores não são autorizados a entrar na área antes da bola ter saído dos limites desta (e o modo de jogo ter mudado para *play_on*).

6) **Controlo do modo de jogo.** Quando o modo de jogo é *kick_off, kick_in, corner_kick* ou *goal_kick*, o árbitro trata de verificar quando o livre correspondente foi executado (a bola foi chutada, mudando o modo de jogo para *play_on*).

7) **Controlo do tempo e do final do jogo.** O árbitro suspende o jogo quando a primeira ou a segunda parte terminam. Cada parte do jogo tem uma duração por defeito de 5 minutos (3000 ciclos). Se o jogo estiver empatado no final do tempo regulamentar (10 minutos – 6000 ciclos), o árbitro encarrega-se de estender o jogo para que o mesmo jogo prossiga, até que uma das equipas marque o golo decisivo (prolongamento em moldes golo dourado ou morte súbita).

As mensagens que o árbitro artificial pode enviar aos jogadores e a sua sintaxe encontram-se descritas na tabela 36:

Mensagem	***tc***	***Modo de Jogo Seguinte***	***Comentário***
(**referee** *before_kick_off*)	*0*	*kick_off_Equipa*	No início de uma parte do jogo (jogo normal)
(**referee** *play_on*)		*play_on*	Jogo normal
(**referee** *half_time*)		*before_kick_off*	Fim da primeira parte
(**referee** *time_extended*)		*before_kick_off*	Fim do jogo (empatado)
(**referee** *time_up*)		*time_over*	Final do jogo (após o final da segunda parte existe um vencedor)
(**referee** *time_up_without_a_team*)		*time_over*	Final do jogo (sem oponente até ao final da segunda parte)
(**referee** *kick_off_Equipa*)	*X*	*play_on*	Início do jogo (depois de pressionado o botão *Kick_Off* no monitor). Modo de jogo muda após bola ser chutada
(**referee** *kick_in_Equipa*)	*X*	*play_on*	Modo de jogo muda após bola ser chutada
(**referee** *free_kick_Equipa*)	*X*	*play_on*	Modo de jogo muda após bola ser chutada
(**referee** *corner_kick_Equipa*)	*X*	*play_on*	Modo de jogo muda após bola ser chutada
(**referee** *goal_kick_Equipa*)	*X*	*play_on*	Modo de jogo muda após bola sair da grande-área da equipa
(**referee** *drop_ball*)	*0*	*play_on*	Ocorre se a bola não é colocada em jogo depois de *drop_ball_time* ciclos ou assinalado pelo árbitro humano.
(**referee** *goal_Equipa_Num*)	*50*	*kick_off_EqOp*	Golo número *Num* da equipa. Após 50 ciclos será dado o reinício do jogo com um *kick_off* para a equipa adversária
(**referee** *offside_Equipa*)	*30*	*free_kick_EqOp*	Fora de jogo. Livre a favor da equipa

			adversária após 30 ciclos.
(**referee** *foul_Equipa)*	*0*	*free_kick_EqOp*	Falta cometida por uma equipa
(**referee** *goalie_catch_ball_Equipa)*	*0*	*goalie_free_kick_ Equipa*	Guarda-redes da Equipa agarrou a bola.

Tabela 36: Mensagens do Árbitro Artificial e seu Significado

Todos os anos, as capacidades do árbitro artificial são estendidas de forma a torná-lo cada vez mais autónomo e evitar a necessidade de existir intervenção humana durante o jogo.

Anexo 5: Algoritmos de Actualização do Estado do Mundo com e sem Bola

```
//Refresca informação relativa a parâmetros individuais do jogador relacionados com a posse de bola
ALGORIMO RefreshBallPossessionWS(ActionWorldState, WorldState) RETORNA ActionWorldState
{
// Refresca a informação relativa a todos os remates à baliza contrária possíveis e respectivos parâmetros de avaliação
    ActionWorldState = RefreshShootInfo(WorldState, ActionWorldState)
// Refresca a informação relativa a todos os passes de diversos tipos possíveis para todos os colegas de equipa e respectivos parâmetros de avaliação
    ActionWorldState = RefreshPassInfo(WorldState, ActionWorldState)
// Refresca a informação relativa a todos os passes em profundidade (passes que não são directos para um jogador mas sim para um ponto onde o jogador irá receber a bola quase parada) e respectivos parâmetros de avaliação
    ActionWorldState = RefreshForwardInfo(WorldState, ActionWorldState)
// Refresca a informação relativa aos dribles possíveis e seus parâmetros de avaliação
    ActionWorldState = RefreshDribbledInfo(WorldState, ActionWorldState)
// Refresca a informação relativa às possíveis acções para aliviar a bola e seus parâmetros de avaliação
    ActionWorldState = RefreshClearInfo(WorldState, ActionWorldState)
// Refresca a informação relativa às formas de segurar a bola e suas avaliações
    ActionWorldState = RefreshHoldInfo(WorldState, ActionWorldState)
    RETORNA ActionWorldState
}
//Refresca informação relativa a parâmetros individuais do jogador sem bola
ALGORIMO RefreshBallRecoveryWS(ActionWorldState, WorldState) RETORNA ActionWorldState
{
//Refresca informação de alto-nível do próprio jogador
    ActionWorldState = RefreshWSPlayer(WorldState, ActionWorldState)
//Actualiza a informação relativa à posse e trajectória da bola e actualiza um conjunto de variáveis relacionado indicando se a bola está livre, se foi chutada, se mudou subitamente de posição, se foi centrada para a área, se foi chutada à baliza, se vai ser interceptada pelo guarda-redes, etc.
    ActionWorldState = RefreshWSHighLevelBall(WorldState, ActionWorldState)
//Refresca informação de alto-nível sobre Distâncias e Intercepções de todos os jogadores e determina os jogadores mais próximos e mais rápidos a chegar à bola e respectivos pontos de intercepção da mesma
    ActionWorldState = RefreshWSInterceptions(WorldState, ActionWorldState)
//Funde informação comunicada de alto-nível sobre recuperação de bola com o estado do mundo. Esta informação inclui, por exemplo, dados sobre as intercepções dos jogadores mais rápidos de cada equipa
    ActionWorldState = RefreshWSMergeHighLevelComm(WorldState, ActionWorldState)
//Actualiza informação relacionada com a disposição regional de jogadores, incluindo por exemplo as linhas de fora de jogo e respectivas confianças
    ActionWorldState = RefreshWSRegionalInfo(WorldState, ActionWorldState)
    RETORNA ActionWorldState
}
```

Algoritmo 27: Actualização do Estado do Mundo de Acção

Anexo 6: Algoritmo de Movimentação Evitando Adversários

```
ALGORITMO MoveToPoint(MyPos, MyDir, PosTarget, DirFinal, Buffer, BufferX, BufferY)
    RETORNA PointReached - Ponto final atingido ou não
PARÂMETROS
    MyPos - Posição actual (x,y) do jogador
    MyDir - Direcção actual do corpo do jogador
    PosTarget - Posição objectivo (x,y) a atingir
    DirFinal - Direcção final desejada para o agente
    Buffer, BufferX, BufferY - Tolerâncias globais e segundo os eixos dos xx e yy na posição
final
{
// Calcula a distância e ângulo para o ponto de destino
    DistTarget = Distance(MyPos, PosTarget)
    AngleTarget = Direction(PosTarget, Mypos) - MyDir
// Se o ponto de destino foi atingido então roda para a direcção final desejada e finaliza
    SE DistTarget< Buffer OU DistTarget.x < BufferX E DistTarget.y < BufferY ENTAO {
       turn(DirFinal)
       RETORNA TRUE
       }
// Se vai colidir com um jogador modifica destino momentaneamente de forma a contorná-lo
    SE PlayerColisionPredicted() ENTAO {
       PosPlayer = PlayerColisionPredictedPosition()
       DodgePlayer(MyPos, MyDir, PosTarget, PosPlayer)
       RETORNA FALSE
       }
// Dependendo da distância ao destino e da situação de jogo e posição no campo calcula o erro
máximo no ângulo de orientação do jogador para o ponto de destino. O erro admissível será
tanto maior quanto maior for a proximidade do jogador ao ponto de destino
    AngMaxError = CalculateMaxError(DistTarget)
// Se o ângulo para o ponto de destino é maior do que o erro máximo admissível então roda para
o ponto de destino senão acelera para o ponto tendo cuidado com a energia
    SE abs(AngleTarget) > AngMaxError ENTAO turn(AngleTarget)
       SENAO {
              DashPower = CalculateDashPower(DistTarget)
              dash(DashPower)
          }
   RETORNA FALSE
}
```

Algoritmo 28: Movimentação Evitando Colisões com Adversários

Anexo 7: Algoritmo de Intercepção

```
ALGORITMO InterceptBall(BallPos, BallVel, MyPos, MyVel, MyDir)
    RETORNA IntercState - Estado da Intercepção (finalizada, em execução ou impossível
PARÂMETROS
    BallPos, BallVel - Posição e velocidade (x,y) da bola
    MyPos, MyVel, MyDir - Posição (x,y), Velocidade (x,y) e direcção do corpo do jogador
{
// Se a posição do jogador ou da bola não forem válidas então não é possível planear uma
intercepção
    SE PlayerPosInvalid OU BallPosInvalid ENTAO RETORNA FALSE
// Se a bola for controlável com um comando kick então está interceptada
    SE BallControlPossible() ENTAO RETORNA TRUE
// Se a bola estiver parada então o jogador movimenta-se para o ponto onde esta se encontra
    SE BallStopped() ENTAO {
        MoveToPoint(MyPos, MyDir, BallPos, CalculateBestDir(MyDir),
                    CalculateBuffer(MyPos, BallPos))
        RETORNA EXECUTION
    }
// Calcula se é possível interceptar a bola utilizando um ou dois turns e uma sequência de
dashs. Se não for possível interceptar a bola num número baixo de ciclos retorna impossível
    IntercPossible = InterceptionPossible(MyPos, MyVel, MyDir, BallPos, BallVel)
    SE NOT(IntercPossible) ENTAO RETORNA FALSE
// Tendo em conta a posição de intercepção a considerar (previamente calculada), a distância
ao ponto de intercepção e o ângulo entre a direcção actual do jogador e a direcção do ponto de
intercepção
    InterceptPos = GetInterceptPoint(MyPos, MyVel, MyDir, BallPos, BallVel)
    DistTarget = Distance(MyPos, InterceptPos)
    AngleTarget = Direction(InterceptPos, Mypos) - MyDir
// Se é possível uma intercepção utilizando unicamente comandos dash então executa um dash
    SE DashSequenceIntercPossible(InterceptPos) ENTAO {
        DashPower = CalculateDashPower(MyPos, InterceptPos)
        dash(DashPower)
        RETORNA EXECUTION
    }
// Se é possível uma intercepção utilizando um turn e depois uma sequência de dashs então
executa o turn para a direcção de intercepção
    SE TurnDashSequenceIntercPossible(InterceptPos) ENTAO {
        turn(AngleTarget)
        RETORNA EXECUTION
    }
//Devido à inércia dos jogadores é usual não ser possível um jogador direccionar-se para a
posição de intercepção com um único turn. Neste caso, o jogador vai primeiro executar um turn
para uma posição intermédia e só depois um turn para a posição de intercepção (ou para uma
nova posição intermédia). A posição intermédia é calculada dependendo de vários factores como
por exemplo a possibilidade de manter contacto visual com a bola
    SE SeveralTurnsDashSequenceIntercPossible(InterceptPos) ENTAO {
        turn(CalculateBestTurnMiddleDir())
        RETORNA EXECUTION }
    RETORNA FALSE
}
```

Algoritmo 29: Intercepção de Bola em Movimento

Anexo 8: Algoritmos de Cálculo dos Parâmetros do Drible

```
CalculateDribbleSmartAngle(ActionWorldState)
    RETORNA FinalAng - Ângulo da bola relativamente ao corpo do agente
PARÂMETROS
    ActionWorldState - Estado do mundo de acção
{
// Se a posição do jogador ou da bola não forem válidos então não é possível planear um drible
    SE PlayerPosInvalid OU BallPosInvalid ENTAO RETORNA FALSE
// Para todos os adversários vai verificar as suas distâncias e direcções de intercepção,
calcular os melhores ângulos de drible para os evitar e acumulá-los num vector, seleccionando
no final qual o melhor ângulo para evitar intercepções de todos os adversários
    PARA Opp = 1 ATÉ TeamSize FAÇA
       SE OpponentPosValid(Opp) ENTAO {
// Caso a posição do oponente Opp seja conhecida, calcula a sua distância e ângulo relativos,
limitando a distância a um mínimo (MinDist = 2)
                DistOpp = CalculateClosestOpponentDistance(Opp)
                AngleOpp = CalculateOpponentAngle(Opp)
                SE (DistOpp < MinDist) ENTÃO DistOpp = MinDist
// Caso os adversários venham detrás do jogador então procura colocar a bola no ponto oposto à
sua direcção relativa
                SE (AngOpp>45 E AngOpp<=180) ENTÃO DribAng = -180 + AngOpp
                SE (AngOpp>=-180 E AngOpp<-45) ENTÃO DribAng = 180 + AngOpp
// Caso os oponentes estejam de um lado ou outro procura colocar a bola nas direcções em que o
drible é mais rápido (135 ou -135 graus)
                SE (DribAng > 135) ENTÃO DribAng = 135
                SE (DribAng < -135) ENTÃO DribAng = -135
// Calcula o peso do oponente (probabilidade de intercepção da bola futura deste oponente)
tendo em conta a sua distância e ângulo. O peso é igual a 2.0/DistOpp com alguns ajustes para
se o jogador for o guarda-redes adversário ou se estiver em frente ao jogador (caso em que
será incrementado)
                DribWeight = CalculateDribbleAngleWeight(Opp, DistOpp, AngOpp)
// Acumula o peso calculado na posição respectiva de um vector de preferência de ângulos de
drible
                AngArray[DribAng] = AngArray[DribAng] + Weight;
       }
// Calcula o melhor ângulo de drible tendo em conta os pesos dados a cada um dos ângulos de
drible. Se unicamente um ângulo de drible tiver um peso elevado esse é seleccionado. Caso
contrário, o vector é filtrado e é efectuada uma média pesada entre os melhores ângulos de
drible.
    FinalAng = CalculateBestAngle(AngArray)
    RETORNA FinalAng
}
```

Algoritmo 30: Cálculo do Ângulo de Drible Inteligente

```
// Altera a direcção desejada de drible de forma a contornar os adversários mais próximos
ALGORITMO CalculateDribbleSmartDir(DesiredDir, ActionWorldState) RETORNA FinalDir
PARÂMETROS
    DesiredDir - Direcção desejada de drible no campo
    ActionWorldState - Estado do mundo de acção
{
// Se a posição do jogador ou da bola não forem válidas então não é possível planear um drible
    SE PlayerPosInvalid OU BallPosInvalid ENTAO RETORNA DesiredDir
```

```
// Para todos os adversários vai verificar as suas distâncias e direcções de intercepção,
calcular os melhores ângulos de drible para os evitar e acumulá-los num vector, seleccionando
no final qual o melhor ângulo para evitar intercepções de todos os adversários
    PARA Opp = 1 ATÉ TeamSize FAÇA
       SE OpponentPosValid(Opp) ENTAO
              DistOpp = CalculateClosestOpponentDistance(Opp)
              SE (DistOpp < MinDist) ENTÃO DistOpp = MinDist
// Calcula a diferença entre a direcção absoluta desejada de drible e direcção absoluta do
oponente
              DirOpp = CalculateClosestOpponentAbsoluteDir(Opp)
              DiffDir = Normalize(DesiredDir - DirOpp)
// Tendo em conta a direcção e distância do oponente calcula a alteração da direcção de drible
melhor para contornar o adversário (esta direcção é discretizada com uma precisão de 10 graus
e limitada ao intervalo [-90,90] sendo tanto maior quanto mais próximos e em frente estiverem
os adversários).
              DirChange = CalculateDribbleDirChange(Opp, DistOpp, DiffDir)
              Weight = CalculateDribbleDirChangeWeight(Opp, DistOpp, DiffDir)
              DirArray[DirChange] = DirArray[DirChange] + Weight;
       }
//Utilizando o vector contendo os pesos atribuídos a todas as variações de direcção
aconselháveis para contornar os adversários é seleccionada a melhor variação de direcção
    FinalDir = CalculateBestDirChange(DirArray)
    RETORNA FinalDir
}
```

Algoritmo 31: Cálculo da Direcção de Drible Inteligente

```
// Para todos os adversários vai verificar as suas distâncias e direcções de intercepção,
calcular os melhores ângulos de drible para os evitar e acumulá-los num vector, seleccionando
no final qual o melhor ângulo para evitar intercepções de todos os adversários
    PARA Opp = 1 ATÉ TeamSize FAÇA
       SE OpponentPosValid(Opp) ENTAO
              DistOpp = CalculateClosestOpponentDistance(Opp)
              SE (DistOpp < MinDist) ENTÃO DistOpp = MinDist
// Calcula a diferença entre a direcção absoluta desejada de drible e direcção absoluta do
oponente
              DirOpp = CalculateClosestOpponentAbsoluteDir(Opp)
              DiffDir = Normalize(DesiredDir - DirOpp)
// Tendo em conta a direcção e distância do oponente calcula a alteração da direcção de drible
melhor para contornar o adversário (esta direcção é discretizada com uma precisão de 10 graus
e limitada ao intervalo [-90,90] sendo tanto maior quanto mais próximos e em frente estiverem
os adversários).
              DirChange = CalculateDribbleDirChange(Opp, DistOpp, DiffDir)
              Weight = CalculateDribbleDirChangeWeight(Opp, DistOpp, DiffDir)
              DirArray[DirChange] = DirArray[DirChange] + Weight;
       }
//Utilizando o vector contendo os pesos atribuídos a todas as variações de direcção
aconselháveis para contornar os adversários é seleccionada a melhor variação de direcção
    FinalDir = CalculateBestDirChange(DirArray)
    RETORNA FinalDir
}
```

Algoritmo 31: Cálculo da Direcção de Drible Inteligente

Anexo 9: Comportamento do Guarda-Redes

Os comportamentos do guarda-redes do FC Portugal são:

- **Movimento Estratégico (*Strategic Move*).** O guarda-redes desloca-se para a sua posição de ajuste que corresponde à sua posição estratégica. A posição estratégica do guarda-redes consiste basicamente numa linha definida utilizando uma série de parâmetros. Esta linha pode ser um rectângulo ou outro tipo de poliedro com parâmetros definidos na configuração da estratégia do guarda-redes para cada jogo. A posição estratégica do guarda-redes é calculada como sendo o ponto sobre a sua linha de movimento estratégico que minimiza a distância máxima que o guarda-redes terá de percorrer para interceptar a bola quando esta é chutada por um oponente partindo de um dado ponto (designado ponto de marcação) para um dos cantos da baliza. O ponto de marcação é usualmente a posição da bola. No entanto, quando a bola se encontra livre e se desloca rapidamente (por exemplo num centro para a área), o guarda-redes prevê qual o próximo ponto de intercepção da bola por parte de um oponente e utiliza esse ponto como ponto de marcação.

- **Intercepção Passiva (*Passive Interception*).** Este comportamento é utilizado para realizar intercepções seguras. O ponto mais próximo entre a trajectória da bola e a posição do guarda-redes é seleccionado como ponto de intercepção. Este comportamento é utilizado se o guarda-redes for claramente mais rápido a chegar a este ponto de intercepção do que qualquer adversário ou colega de equipa a alcançar a bola. Caso contrário uma intercepção activa é seleccionada.

- **Intercepção Activa (*Active Interception*).** Em situações mais disputadas, o guarda-redes deve correr para a bola de forma a poder agarrá-la antes que qualquer colega ou adversário a possa alcançar. Neste tipo de intercepção, a forma de intercepção mais rápida é seleccionada. Uma vez que o seu sucesso é crítico, o comportamento de intercepção activa do guarda-redes permite a realização de diversas acções *dash* com valor negativo consecutivas. A realização de intercepções utilizando estas acções embora consuma uma quantidade de energia dupla, permite por vezes alcançar a bola em situações em que uma intercepção normal (andando para a frente) não o permite. Isto pode ser a diferença entre sofrer ou não um golo.

- **Intercepção Desesperada (*Goal Interception*).** Em casos em que nenhuma outra alternativa resta ao guarda-redes, este tenta correr de forma desesperada para o ponto de intercepção da trajectória da bola com a linha de baliza na tentativa de interceptar a bola nesse ponto. Este comportamento embora com baixa probabilidade de sucesso é a única alternativa que resta ao guarda-redes quando a bola se encaminha para a

sua baliza sem que este a possa interceptar de forma passiva ou activa.

- **Intercepção às Cegas (*Blind Interception*).** Este comportamento é utilizado no caso de o guarda-redes perder de vista a bola quando no instante anterior esta se encontrava na posse de um oponente. Devido ao guarda-redes utilizar um cone de visão reduzido (de forma a receber informação visual todos os ciclos) existe a hipótese de que o adversário ao rematar a bola esta saia do cone de visão do guarda-redes. Neste caso, o guarda-redes tenta realizar um *dash* na direcção em que com maior probabilidade o avançado ter chutado a bola.

- **Agarrar a Bola (*Catch*).** A decisão de agarrar ou não a bola parece extremamente simples: se a bola está mais próxima do que *catch_length*, agarrá-la. No entanto, após uma análise mais cuidada do problema, verifica-se que, devido ao ruído introduzido nos sensores e actuadores pelo servidor, o guarda-redes pode facilmente pensar que a bola está ao seu alcance sem que, na realidade, o esteja. O guarda-redes do FC Portugal, agarra a bola, unicamente se esta estiver com segurança ao seu alcance. Caso contrário, se a bola não estiver ao alcance de qualquer oponente e for com maior segurança alcançável no instante de tempo seguinte (estiver mais próxima), o guarda-redes prefere esperar e agarrar a bola com maior segurança.

- **Chutar a Bola (*Kick*).** Em casos em que o guarda-redes não possa efectuar um *catch* da bola (devido a ter tentado efectuar um pouco tempo antes) mas a bola esteja no interior da sua *kickable_area*, o guarda-redes efectuará um chuto de forma a tentar parar a bola para que a possa agarrar momentos depois, ou para que a afaste da zona de perigo.

BIBLIOTECA F.E. PORTO