UNIVERSIDADE DO PORTO
FACULDADE DE ENGENHARIA

Um Sistema de Construção Automática de Horários em Universidades, utilizando o Algoritmo de *Arrefecimento Simulado*

Pedro Miguel Teixeira Faria

Porto, Novembro de 2000

UNIVERSIDADE DO PORTO

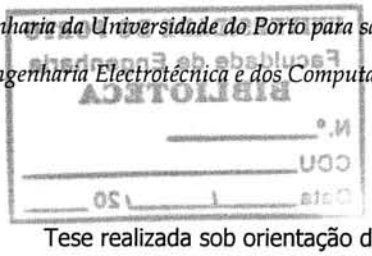
FACULDADE DE ENGENHARIA

Um Sistema de Construção Automática de Horários em Universidades, utilizando o Algoritmo de *Arrefecimento Simulado*

Pedro Miguel Teixeira Faria

Universidade do Porto
Faculdade de Engenharia
FEUP

Tese submetida à Faculdade de Engenharia da Universidade do Porto para satisfação parcial dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrotécnica e dos Computadores (Ramo de Informática Industrial).



Tese realizada sob orientação 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
e co-orientação do Mestre Luís Paulo Reis
Mestre Assistente do Departamento de Ciência e Tecnologia
da Universidade Fernando Pessoa

Porto, Novembro de 2000

À Bela

Aos meus Pais

À Mónica e ao Filipe

Agradecimentos

Gostaria de agradecer ao meu orientador, Professor Eugénio da Costa Oliveira, pelo interesse demonstrado na realização deste projecto, pelas ideias e sugestões transmitidas e pelas excelentes condições de trabalho que proporcionou, nomeadamente a disponibilidade do Laboratório do Núcleo de Inteligência Artificial Distribuída e Robótica, o que contribuiu grandemente para o êxito alcançado pela realização deste trabalho.

Ao Luís Paulo, pela dupla função de co-orientador e de amigo. Sem ele, seria impossível desenvolver o projecto aqui apresentado nesta dissertação. Forneceu-me o vasto conjunto de artigos necessários para o arranque do trabalho e mostrou-se sempre disponível e paciente para discutir qualquer dúvida que surgisse. Pelo incentivo que me transmitiu, pelo empenho e dedicação demonstrados ao longo deste trabalho e sobretudo pela amizade, quero agradecer-lhe.

A todos os elementos do NIAD&R (Núcleo de Inteligência Artificial Distribuída e Robótica) agradeço a amabilidade com que me receberam, nomeadamente a Ana Paula, o António Pereira, a Benedita, a Conceição, o Henrique e o Pedro Silva.

Em especial, quero agradecer à Bela o apoio, incentivo e paciência, principalmente na fase mais complicada de conclusão da dissertação, do tempo que era nosso e que este trabalho nos tirou. Quero também agradecer-lhe a leitura efectuada a este trabalho e as ideias sugeridas.

À minha família, nomeadamente aos meus pais, à minha irmã e ao meu irmão, pelo apoio e incentivo que me transmitiram durante o período de realização do mestrado e pelo tempo em que não pude estar com eles, não há palavras para agradecer.

À Mónica, pela leitura do trabalho e sugestões transmitidas, quero agradecer.

Ao Pedro Moreira, agradeço o apoio na leccionação das aulas, no período final de conclusão da dissertação, altura em que o tempo de preparação das mesmas foi mais curto.

À Isabel Ribeiro, tenho de agradecer a tradução do resumo desta tese para Francês.

Resumo

Nesta dissertação é realizado um estudo sobre o *Problema da Construção de Horários em Universidades*. O referido problema consiste em fixar, no tempo (dia e hora) e no espaço (sala), uma sequência de encontros entre docentes e alunos, num determinado período de tempo (tipicamente uma semana), satisfazendo um conjunto de restrições de diversos tipos.

É efectuada uma apresentação dos diversos métodos de resolução automática do problema, encontrados na literatura, sendo dado particular destaque a algumas das técnicas inteligentes de optimização conhecidas: *Arrefecimento Simulado*, *Pesquisa Tabu* e *Algoritmos Genéticos*.

A seguir é apresentada a arquitectura base de um sistema de construção de horários. O referido sistema utiliza uma linguagem *standard* de representação de problemas de construção de horários (*UNILANG*), e uma base de dados definida de forma a possibilitar a estruturação e armazenamento da maior variedade possível de problemas deste género. O sistema automático de geração de horários foi implementado utilizando uma linguagem de programação gráfica (*DELPHI*), que permitiu o desenvolvimento de uma interface com o objectivo de facilitar a interacção do utilizador com o sistema. Ao nível algorítmico, com vista à resolução de problemas de construção de horários foi implementada uma heurística baseada no algoritmo de *Arrefecimento Simulado*.

Os resultados obtidos parecem demonstrar as potencialidades da utilização de um sistema automático de geração de horários. O sistema é capaz de resolver de forma totalmente automática problemas complexos, através do algoritmo implementado. A sua interface amigável permite ainda uma fácil interacção com o utilizador do sistema, reduzindo assim o elevado número de pessoas envolvido no processo, bem como a enorme quantidade de tempo despendido na obtenção de horários.

Abstract

This thesis contains a study of the *general Timetabling Problem in Universities*. The timetabling problem consist in fixing in time (day and hour) and space (room), a sequence of meeting between teachers and students, in a pre-determined period of time (typically a week), satisfying a set of constraints of several types.

The methods used for automated timetabling that may be found in the specialized literature are reviewed with particular emphasis being given to some intelligent optimisation techniques like: *Simulated Annealing*, *Tabu Search* and *Genetic Algorithms*.

The architecture for a complete timetabling system is presented along with a standard language to represent timetabling problems (UNILANG) and a database defined in order to structure and store the information of a large variety of timetabling problems. The timetabling system was implemented using a graphical programming language (DELPHI), which enabled the development of a friendly user interface. At the algorithmic level, the system uses a heuristic based on Simulated Annealing to solve university-timetabling problems.

The result achieved by the system in medium and large timetabling problems show its capabilities. The system is able to solve very complex timetabling problems in a totally automatic fashion through the implemented algorithm. Its friendly interface enables a very easy interaction with the user, reducing the number of people involved in the timetabling process and the time needed to achieve a good timetable for the institution.

Résumé

Cette dissertation décrit une étude qui a été reálisée sur le sujet du Problème de la Construction des Horaires au niveau de l'Enseignement Supérieur. Le problème mentionné ci-dessus, consiste à fixer, dans le temps (jour et heure) et dans l'espace (salle de classe), une séquence de rencontres entre les enseignants et les élèves, pendant un intervalle de temps déterminé (typiquement une semaine) et en satisfaisant un ensemble de restrictions de plusieurs genres.

Une revue de la littérature concernant les différentes méthodes de résolution automatique du problème est décrite, en mettant en évidence les algorithms d'optimization de functions les plus connues: le *Refroidissement Simulé*, la *Recherche Tabou* et les *Algorithmes Génétiques*.

Ensuite, l'architecture de base d'un système de construction des horaires est décrite. Le système utilise un langage standard pour la représentation des problèmes de construction des horaires (*UNILANG*) et une banque de données définie de manière à permettre la structuration et le stockage de la plus grande variété possible de problèmes du genre. Le système automatique de génération des horaires a été construit en ayant recours à un langage de programmation graphique (*DELPHI*), qui a permis l'élaboration d'une interface qui puisse faciliter l'interaction de l'utilisateur avec le système. Du point de vue algorithmique et en ayant pour but la résolution des problèmes de construction des horaires, une méta-heuristique dénommée "Refroidissement Simulé" a été implémentée.

Les résultats obtenus paraissent démontrer les potentialités de l'utilisation d'un système automatique pour la génération des horaires. Le système est capable de résoudre de mode complètement automatique des problèmes complexes, en ayant recours à l'algorithme mentionné ci-dessus. Son interface amiable permet une interaction facile avec l'utilisateur du système, ce qui permet ainsi de réduire, non seulement le grand nombre de interveniants nécessaires pour accomplir la construction des horaires, mais aussi la longue durée de ce travail.

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1
Introdução

1.1 MOTIVAÇÃO

Um estudante, durante a sua vida académica, pouco ou nada reflecte sobre os horários de aulas que todos os anos lhe são atribuídos, pela instituição de ensino, na qual se encontra inserido. No entanto, é por estes que rege o seu tempo, ao longo do ano lectivo. Não sabe que todos os anos, durante um longo período de tempo, um enorme esforço humano é despendido por parte de pessoas que geralmente exercem funções na instituição, para que alunos e docentes tenham os seus horários a tempo de iniciar-se um novo período lectivo. À primeira vista poderia julgar tratar-se de um problema de resolução simples, devido ao número reduzido de entidades envolvidas: alunos, docentes e salas.

Após uma análise mais pormenorizada do problema, facilmente se compreende a enorme quantidade de recursos físicos e humanos envolvidos, cuja informação associada é necessário estruturar correctamente, por forma a obterem-se bons horários. Este problema real de grande aplicabilidade e de resolução complexa, parece bastante interessante estudar, em teoria e na prática, notando-se cada vez mais essa necessidade, devido ao contínuo crescimento das instituições de ensino superior.

Um estudo extremamente interessante de realizar, consiste na análise detalhada do problema da construção de horários e no estudo de técnicas possíveis de aplicar na resolução do referido problema. De seguida, definir e implementar um projecto que passe pela implementação prática de um sistema automático de construção de horários, o mais abrangente possível, utilizando uma das técnicas estudadas.

1.2 OBJECTIVOS

Com o presente trabalho pretende-se inicialmente estudar o problema da construção de horários, nomeadamente na vertente da construção automática, utilizando diversos métodos existentes aplicáveis à construção de horários. Com base nesse estudo, pretende-se projectar um sistema de construção automática de horários, possível de implementar e que possa facilitar o trabalho do

utilizador, nas diversas etapas até à obtenção dos horários finais. Será importante efectuar uma análise comparativa dos diversos métodos, que podem ser utilizados no processo de geração automática dos horários. Sendo assim, os objectivos gerais deste projecto podem ser enunciados da seguinte forma:

- *Estudo do problema da construção de horários;*
- *Estudo de diversos métodos de construção automática de horários;*
- *Definição da arquitectura base de um sistema de dados, aplicada ao problema de construção de horários numa universidade;*
- *Estudo e selecção de uma plataforma que permita o desenvolvimento de raiz de uma aplicação genérica de construção automática de horários;*
- *Projecto de um sistema automático de construção de horários;*
- *Implementação de um sistema de geração automática de horários em instituições de ensino superior, utilizando uma técnica anteriormente estudada;*
- *Aplicação do sistema na resolução de problemas de geração de horários.*

1.3 CONTRIBUIÇÕES

Podemos considerar como principais contribuições proporcionadas pelo trabalho desenvolvido, as seguintes:

- *Realização de uma pesquisa bibliográfica sobre o tema de construção de horários;*
- *Construção de um projecto com uma arquitectura flexível, permitindo a integração no futuro, de inovações no trabalho desenvolvido;*
- *Projecto e implementação de raiz de um sistema de construção automática de horários, utilizando uma meta-heurística;*
- *Realização na Faculdade de Engenharia da Universidade do Porto de um projecto específico de construção automática de horários, embora desenvolvido a um nível de protótipo.*

1.4 ORGANIZAÇÃO GLOBAL DA DISSERTAÇÃO

A dissertação aqui apresentada encontra-se estruturada em seis capítulos. O primeiro, no qual se integra este sub-capítulo, faz uma breve introdução ao trabalho desenvolvido. No segundo capítulo é efectuada uma aproximação ao problema da construção de horários, sendo apresentadas diversas

considerações sobre o problema em causa. No terceiro capítulo são efectuadas diversas abordagens heurísticas ao problema, através do estudo de diversos métodos usados no processo de construção de horários. No quarto capítulo é apresentada a arquitectura base do sistema de dados, a utilizar num projecto de implementação de um sistema automático de construção de horários. No quinto capítulo é apresentado o sistema desenvolvido e efectuada uma análise aos resultados obtidos na resolução de diversos problemas de construção de horários. Por último, são apresentadas as conclusões e considerações finais do trabalho.

Capítulo 2
Construção de Horários

2.1 INTRODUÇÃO

Basicamente, um problema de *Construção 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 encontro é uma combinação de recursos (ex: salas, equipamentos e pessoas), podendo alguns deles ser especificados pelo problema, e outros ser definidos como parte da solução.

[Carter et al. 1994], relativamente ao problema de *Calendarização de Exames*, define bem a essência do problema quando diz que "*o desafio básico é escalonar os exames num determinado período de tempo para evitar conflitos e satisfazer um conjunto de restrições laterais*" (no entanto, esta afirmação aplica-se tanto a exames como a aulas). Os conflitos a que se refere ocorrem quando determinado horário exige a presença de algum recurso em dois locais ao mesmo tempo. As restrições laterais variam de instituição para instituição, bem como entre os problemas de exames e aulas, sendo a lista praticamente interminável. "*Desafio*" é uma palavra muito apropriada. Como afirmam *[Bloomfield e McSharry 1979]*:

> *"Dependendo do tamanho do departamento e da diversidade de ofertas de disciplinas, o tempo necessário para escalonar as turmas pode variar de uma simples tarde até um intensivo mês de trabalho árduo".*

Na sua essência, um problema de *Construção de Horários* é um problema de escalonamento. Escalonamento é definido por *[Wren 96]* da seguinte forma:

> *"Escalonamento pode ser visto como o arranjo de objectos num padrão no tempo ou espaço, de determinada forma em que alguns objectivos são atingidos, ou quase, e as restrições pelas quais os objectos serão dispostos, são satisfeitas ou quase."*

O processo de *Construção de Horários* é uma tarefa difícil e demorada. A enormidade de restrições e questões associadas faz com que as decisões a tomar não sejam muitas vezes aparentes. A

criatividade e o poder de computação são dois dos ingredientes para o sucesso na criação de horários.

A *Construção de Horários* para escolas e universidades é claramente um exemplo de um problema de escalonamento difícil. Para alguns problemas de escalonamento, como o Problema do Caixeiro-Viajante, existe sempre uma solução. O desafio num problema desse tipo é encontrar a melhor solução. A maior complicação num problema de *Construção de Horários* é não ser claro, à partida, que uma solução existe.

Um dos factores que torna o processo de *Construção de Horários* difícil de realizar é o facto de envolver muita gente. *[Romero 1982]* identificou 3 principais interessados neste processo, cada qual com os seus conjuntos de necessidades e interesses:

- *A administração define os standards mínimos a que os horários devem respeitar. Por exemplo, algumas universidades especificam que nenhum aluno pode ter dois exames em períodos consecutivos.*
- *As preocupações dos departamentos têm mais a ver com serem influentes nos horários das disciplinas. Pretendem que "o escalonamento seja consonante com o desenvolvimento da matéria leccionada", bem como apresentarem mais exigências específicas para determinadas salas ou laboratórios. Num contexto de exames, é provavelmente desejado que os exames maiores sejam escalonados mais cedo, no sentido de permitir mais tempo para marcações.*
- *O terceiro grupo de interessados é o dos alunos, cuja observação do horário será restringida à parte que o afecta. Dado o número de alunos envolvidos é difícil obter critérios específicos como, qual é o melhor horário para os alunos. Muitos alunos preferem não ter aulas até tarde à sexta-feira, ou ter um intervalo entre exames consecutivos.*

Para além dos grupos de interessados acima referidos, existem também outros elementos, altamente interessados no resultado final obtido do processo de construção dos horários, que são os docentes.

Existem diversas variações fundamentais no problema de *Construção de Horários*. Horários de exames e horários de aulas têm restrições e considerações completamente diferentes. Além disso, pode ser exigida a *Construção de Horários* antes ou depois da inscrição dos alunos nas turmas.

A complexidade do problema de *Construção de Horários* serviu de estímulo para o aparecimento de diversas propostas de resolução, apresentadas por muitos autores. E para se ter uma melhor noção da dificuldade do problema em causa, considere-se o seguinte exemplo *[Frangouli et al. 1995]*:

Suponhamos um semestre, onde tem de ser definido um horário para trinta disciplinas, com quatro aulas cada (A_i), por semana. Consideremos uma semana com cinco dias de trabalho, com dez períodos de tempo de aulas todos os dias, e em que só existem três salas de aulas. Neste caso, o espaço de pesquisa para a construção de um horário semanal é tão grande como $(5\times10\times3)^{4*30} = 1.35\times10^{261}$. Na Figura 1 encontra-se uma representação esquemática do problema referido.

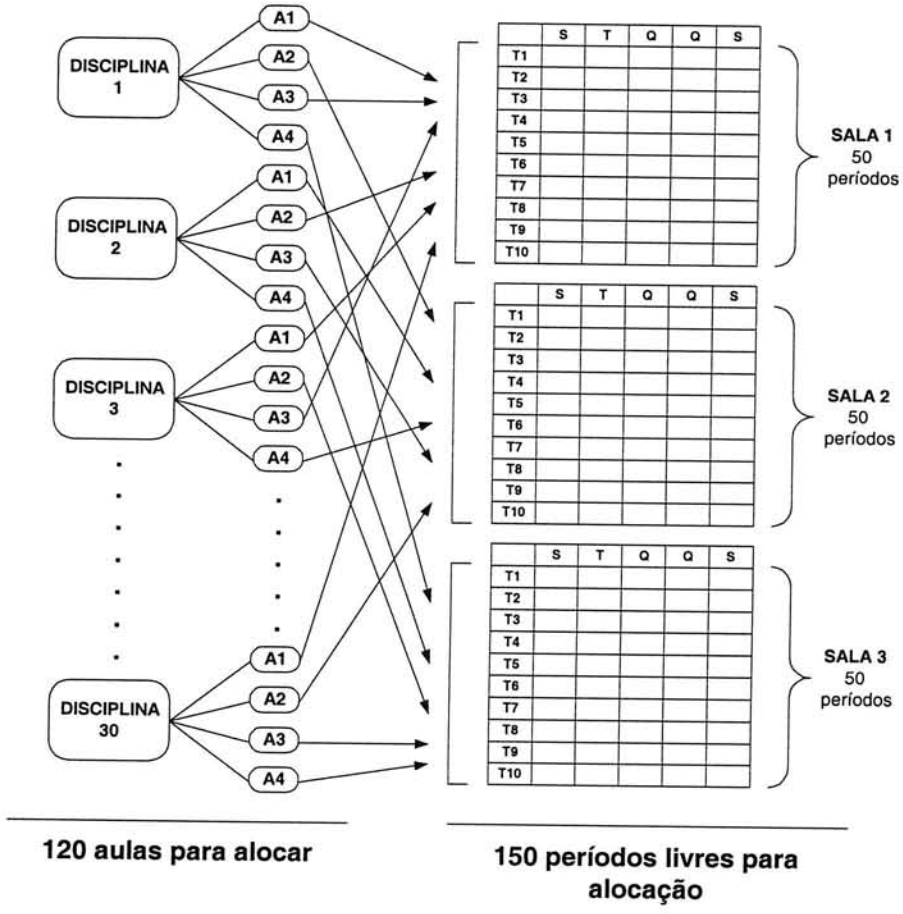


Figura 1: Exemplo de um problema geral de escalonamento de aulas

Como facilmente se pode constatar, o problema é encontrar um ponto neste enorme espaço de pesquisa, que satisfaça todas as restrições, o que de facto torna necessário a existência de sistemas específicos de construção de horários.

No sentido de assegurar um método apropriado para resolver determinado problema é necessário termos primeiro uma ideia de quão difícil é o problema. Muitos problemas combinatórios de pesquisa fazem parte da categoria de problemas NP-Completos[1].

2.2 *FORMULAÇÃO MATEMÁTICA DO PROBLEMA*

Existem várias formulações matemáticas, de diversos autores, do problema de definição de um horário. Numa possível formulação *[Werra 1985]*, a geração de horários é vista como um problema polinomial simplificado:

Partindo do princípio que existem: m turmas ($\mathbf{t}_1, \ldots, \mathbf{t}_m$), n docentes ($\mathbf{d}_1, \ldots, \mathbf{d}_n$) e p períodos ($1, \ldots, p$), existe também uma matriz $R_{m \times n}$ a que se chama *Matriz de Requisitos*, onde $\mathbf{r}_{ij}$ é o número de aulas leccionadas por um docente $\mathbf{d}_i$ à turma $\mathbf{t}_j$.

O problema consiste em atribuir aulas a períodos (tempos) tal que nenhum docente ou turma esteja envolvido em mais do que uma aula ao mesmo tempo. A formulação matemática do problema é apresentada da seguinte forma:

- *Existe uma aula se ocorrer um encontro, num determinado período (k). entre um docente (i) e uma turma (j).*

  $x_{ijk} = 0$ *ou* $1 \quad (i = 1...n; j = 1...m; k = 1...p)$

  onde $x_{ijk} = 1$ *se o docente* d_i *e a turma* t_j *se encontrarem no período* k. *Caso contrário,* $x_{ijk} = 0$.

- *A restrição seguinte garante que cada docente lecciona o número certo de aulas por turma.*

$$\sum_{k=1}^{p} x_{ijk} = r_{ij} \qquad (i = 1,\ldots,n;\ j = 1,\ldots,m)$$

- *Com a restrição seguinte fica garantido que para cada período, cada docente só pode leccionar a uma turma.*

$$\sum_{j=1}^{m} x_{ijk} \leq 1 \qquad (i = 1,\ldots,n;\ k = 1,\ldots,p)$$

- *A próxima restrição garante que para cada período, uma turma só pode ter aulas com um docente.*

[1] Não se conhece nenhum método de resolução de um problema NP-Completo em tempo polinomial. O tempo necessário para a resolução do problema cresce exponencialmente com a dimensão do mesmo.

$$\sum_{i=1}^{n} x_{ijk} \leq 1 \qquad (j = 1,\ldots,m;\ k = 1,\ldots,p)$$

Outra formulação matemática é apresentada por *[Stamatopoulos et al. 1998]* e é assumido que existem N disciplinas, M docentes e L salas. Considere-se, 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 = {09:00-10:00, 10:00-11:00, ..., 19:00-20:00}***. Para cada disciplina i ($1 \leq i \leq N$) existe um dado número de parâmetros:

v_i: *número aproximado de alunos que assistem 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***)

Temos ainda, para o docente i ($1 \leq i \leq M$) 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 dados da 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:

1ª: *Duas sessões não podem ter lugar ao mesmo tempo e na mesma sala:*

$$x_{ijk} \neq x_{i'j'k'} \quad \vee \quad y_{ijk} \neq y_{i'j'k'} \quad \vee \quad z_{ijk} \neq z_{i'j'k'}$$

$\forall i, j, k, i', j', k'$ ***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*** $k \neq k'$.

2ª: *As sessões de uma aula têm de ter lugar em períodos de leccionação consecutivos, no mesmo dia e na mesma sala:*

$$x_{ijk} = x_{ij(k-1)}$$

$$y_{ijk} = \textbf{próximo}(y_{ij(k-1)})$$

$$z_{ijk} = z_{ij(k-1)}$$

$\forall i, j, k$ ***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(10:00-11:00) = 11:00-12:00).***

3ª: *As aulas de uma disciplina têm de ter lugar em dias diferentes:*

$$x_{ijk} \neq x_{ij'k'}$$

$\forall i, j, j', k, k'$ ***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*** $1 \leq k' \leq p_{ij'}$.

4ª: *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*** $1 \leq k \leq p_{ij}$.

5ª: *Nenhum docente pode leccionar duas aulas ao mesmo tempo:*

$$x_{ijk} \neq x_{i'j'k'} \quad \vee \quad y_{ijk} \neq y_{i'j'k'}$$

$\forall i, j, k, i', j', k'$ ***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*** $\{n_{ij''} \mid 1 \leq j'' \leq t_{ij}\} \cap \{n_{i'j'''} \mid 1 \leq j''' \leq t_{i'j}\} \neq \varnothing$.

6ª: Os requisitos de indisponibilidade dos docentes têm de ser respeitados:

$$dut_{nijj'} \neq x_{ij''k} \quad \vee \quad put_{nijj'} \neq y_{ij''k}$$

$\forall i, j, j', j'', k$ *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* $1 \leq k \leq p_{ij''}$.

7ª: Os requisitos de indisponibilidade das salas têm de ser respeitados:

$$duc_{zijkj'} \neq x_{ijk} \quad \vee \quad puc_{zijkj'} \neq y_{ijk}$$

$\forall i, j, k, j'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$ *e* $1 \leq j' \leq uc_{zijk}$.

8ª: Os requisitos de indisponibilidade das disciplinas têm de ser respeitados:

$$dus_{ij} \neq x_{ij'k} \quad \vee \quad pus_{ij} \neq y_{ij'k}$$

$\forall i, j, j', k$ *com* $1 \leq i \leq N$, $1 \leq j \leq us_i$, $1 \leq j' \leq l_i$ *e* $1 \leq k \leq p_{ij'}$.

9ª: Os requisitos de não conflitualidade entre as disciplinas têm de ser respeitados:

$$x_{ijk} \neq x_{sij'j''k'} \quad \vee \quad y_{ijk} \neq y_{sij'j''k'}$$

$\forall i, j, k, j', j'', k'$ *com* $1 \leq i \leq N$, $1 \leq j \leq l_i$, $1 \leq k \leq p_{ij}$, $1 \leq j' \leq d_i$, $1 \leq j'' \leq l_{sij'}$ *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.

Da formulação matemática acima apresentada conclui-se que, o objectivo principal é encontrar uma solução possível. Dessa forma, tal acontece quando estamos perante um problema de pesquisa e não de optimização de uma solução.

2.3 PROBLEMAS ASSOCIADOS À GERAÇÃO DE HORÁRIOS

2.3.1 INTRODUÇÃO

De um modo geral, existem três problemas que podem ser analisados e resolvidos de forma independente uns dos outros mas que, no entanto, são indissociáveis numa instituição de ensino. São eles: a *Distribuição de Serviço Docente*, a *Construção de Horários* e a *Calendarização de Exames*. Todos, são problemas que surgem a uma instituição de ensino em determinadas alturas do ano. O primeiro e o segundo surgem antes da entrada em funcionamento do ano lectivo sendo, no entanto, necessário em algumas instituições de ensino, construir horários a meio do referido ano. O problema da *Calendarização de Exames* surge, geralmente, a meio e no final de um ano lectivo.

Todos estes problemas são de resolução complexa, e para aquelas instituições que adoptam um processo manual para os solucionar, muito tempo é necessário para a sua conclusão. No entanto, comercialmente, vão surgindo sistemas automáticos para a sua resolução. Sistemas estes, que geralmente apresentam algoritmos complexos, baseados em técnicas de Inteligência Artificial e Investigação Operacional.

Apesar dos três problemas referidos serem indissociáveis, e poderem ser tratados de uma forma centralizada, tal poderá não acontecer (de acordo, por exemplo, com o tamanho da instituição), podendo adoptar-se um processo de resolução dos problemas, ao nível de cada departamento da instituição.

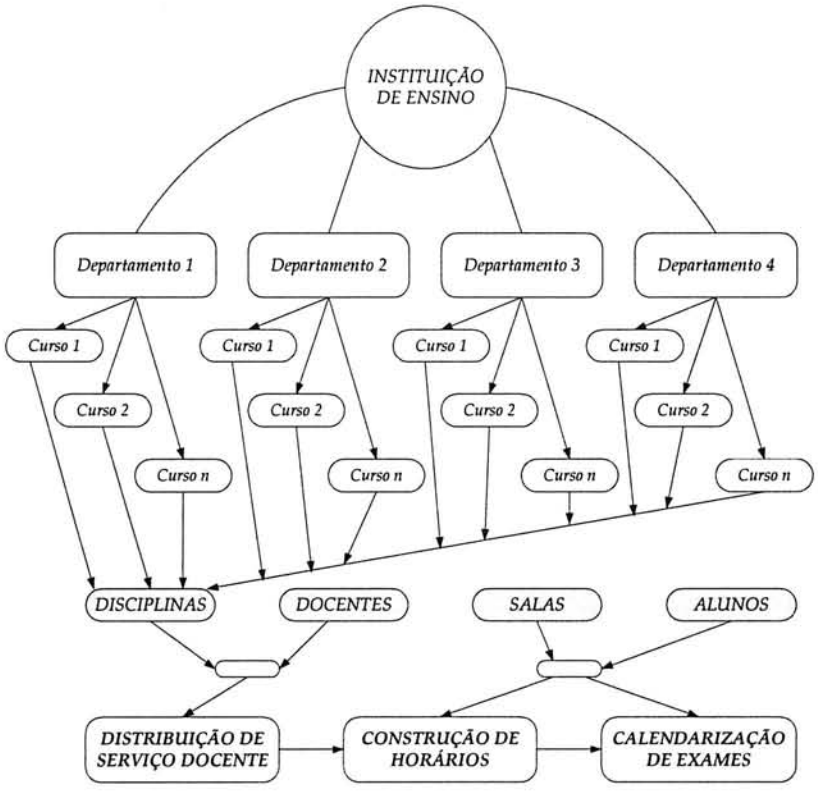


Figura 2: A *Construção de Horários*, a *Distribuição de Serviço Docente* e a *Calendarização de Exames*, numa Instituição de Ensino Superior

Futuramente, procurar-se-á o desenvolvimento de sistemas que reunam os três problemas atrás referidos e os integrem como componentes de um único sistema, que utilizará e partilhará a informação comum, por forma a facilitar o processo de resolução do problema e, consequentemente, apoiar a gestão da instituição de ensino.

2.3.2 CALENDARIZAÇÃO DE EXAMES

Associado ao problema de *Construção de Horários* encontra-se o problema de *Calendarização de Exames*, que consiste na atribuição de blocos de tempo a um determinado número de exames, usualmente, um por disciplina, dentro de um espaço de tempo limitado e satisfazendo um conjunto de restrições.

Este problema contém diversas semelhanças com o problema da *Construção de Horários*, salientando-se no entanto as seguintes características:

- *Só existe um exame por cada disciplina (e normalmente diversas aulas de diferentes tipos);*
- *O período de avaliação é tipicamente maior do que uma semana, e o número de períodos pode variar;*
- *Na maioria dos problemas de Calendarização de Exames pode existir mais do que um exame na mesma sala;*
- *Um exame pode ser dividido por várias salas;*
- *O tipo de sala e a disposição dos lugares são importantes para o processo de Calendarização dos Exames;*
- *As sobreposições de exame não são normalmente permitidas. Pode-se admitir que um aluno seja forçado a faltar a uma aula devido a uma sobreposição, mas nunca que falte a um exame;*
- *Os exames, por aluno, devem ser dispersados pelo calendário, ao contrário dos horários das aulas em que é aconselhável agrupar as disciplinas em blocos.*

Segundo *[Schaerf 1995]*, um calendário de exames consiste:

> *"No escalonamento dos exames de um conjunto de disciplinas de uma escola, evitando a sobreposição de exames das disciplinas com alunos em comum, e dispersando os exames dos alunos, o mais possível."*

Sendo assim, o problema pode ser dividido em duas fases: A primeira consiste na elaboração de um horário que respeite determinadas restrições rígidas, que conduzirá a um horário aplicável. Na segunda fase introduzir-se-ão factores de optimização do horário, que permitirão melhorar a qualidade do calendário final.

O objectivo final do processo de *Calendarização de Exames* é o de garantir que todos os alunos possam realizar os exames a que se propõem, sem que o calendário afecte significativamente as suas hipóteses de obter aprovação nas disciplinas, ou pelo menos, que possa ter um efeito semelhante em todos os alunos.

2.3.3 DISTRIBUIÇÃO DE SERVIÇO DOCENTE

Previamente ao complexo processo de *Construção dos Horários* de uma instituição de ensino, surge o, também complexo, processo de *Distribuição do Serviço Docente*.

A *Distribuição de Serviço Docente* consiste em atribuir disciplinas de um determinado tipo (teórica, prática, teórico-prática, laboratorial) aos docentes qualificados para as leccionar, tendo em atenção diversos critérios como, por exemplo, o interesse (preferências) demonstrado por estes em as leccionar, ou a apreciação realizada pelos responsáveis dos cursos e das respectivas disciplinas em causa.

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, faz com que o processo de distribuição do serviço docente seja, por vezes, executado rapidamente, ficando longe de ser optimizado, sendo os principais lesados os docentes e alunos, bem como os responsáveis pelo processo de *Construção dos Horários* lectivos.

O processo de *Distribuição de Serviço Docente* consiste em três fases fundamentais. A primeira fase inicia-se muito 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 (que normalmente varia entre duas e dez). Em determinadas Instituições de Ensino Superior aproveita-se esta fase para averiguar também as preferências de horário do docente e o número de horas que pretende leccionar (no caso de docentes com contrato a tempo parcial, em que este valor não se encontra especificado).

A segunda fase do processo é a mais trabalhosa e consiste na *Distribuição de Serviço Docente* propriamente dita. Para cada disciplina de um determinado tipo (teórica, prática, teórico-prática, laboratorial) existe uma lista de docentes interessados em leccioná-la. Os responsáveis pela distribuição escolhem o docente que julguem ser o mais adequado para reger a disciplina respeitando, dentro do possível, e se for prática corrente na instituição, o número de horas que o docente pretende leccionar no referido período. Além do regente (responsável) da disciplina, são definidos todos os docentes necessários para garantir todas as aulas (dos diversos tipos) a todas as turmas dessa disciplina.

De seguida, surge uma terceira fase que consiste em "ajustar" a distribuição usando o "*feedback*" dos docentes (reclamações e sugestões). Este processo termina quando é atingida uma solução

aceitável (não óptima). 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, 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, utilizando-se o computador apenas como ferramenta de edição de texto (sobretudo para a construção dos relatórios finais). No entanto, cada vez mais instituições começam a adoptar a utilização de *software* para a construção de horários.

2.4 HORÁRIOS ESCOLARES / UNIVERSITÁRIOS

Um vasto número de sistemas para a resolução do problema de *Construção de Horários* tem sido proposto ao longo do tempo, variando de acordo com diversos factores, como sejam:

- *tipo e dimensão das instituições envolvidas;*
- *tipo e variedade de restrições existentes;*
- *necessidades gerais e específicas da instituição;*
- *tipo de utilizador (responsável pela construção dos horários na instituição).*

Apesar dos factores acima referidos bem como outros de carácter mais específico, foi necessário diferenciar, ao longo do tempo, o problema da *Construção de Horários* em dois tipos principais:

- *Horários escolares;*
- *Horários universitários.*

No caso do problema da *Construção de Horários Escolares*, este consiste em definir horários semanais para todas as turmas de uma escola, evitando que um docente ou uma turma tenham mais do que um evento ao mesmo tempo. Dois conceitos importantes encontram-se associados à resolução deste problema: o conceito de "*junção*" e de "*desdobramento*". O primeiro ocorre quando se pretende que duas ou mais turmas assistam a um evento em conjunto, por forma a optimizar recursos. Por exemplo, no caso de se pretender leccionar a disciplina teórica de Matemática, numa determinada sala, com capacidade suficiente para juntar os alunos de duas turmas de dois cursos diferentes. Neste caso é optimizado o espaço físico e o recurso humano

utilizado. O conceito de desdobramento encontra-se associado à separação do tempo semanal total de leccionação de um evento em dois ou mais eventos de menor duração, mas em que a sua soma perfaz o tempo total definido. Por exemplo, no caso de haver uma disciplina de Português, com a duração total semanal de 4 horas, essa duração pode ser repartida por três tempos (um de duas horas e dois de uma hora). Os desdobramentos poderão ser efectuados não só por ser uma restrição imposta por um docente ou qualquer outra entidade, mas também pela necessidade de conjugar os eventos com as junções anteriormente definidas.

No caso do problema da *Construção dos Horários Universitários*, este consiste na definição de horários para todas as aulas das diversas disciplinas, dos cursos da universidade, minimizando a sobreposição de aulas com alunos em comum. Este modelo corresponde ao sistema de unidades de crédito em que os alunos podem construir o próprio plano curricular, ao longo do seu curso. Ou seja, as restrições associadas com um conjunto de alunos (turma) são, neste caso, maiores porque passa-se a ter em consideração um aluno em particular, que em determinados anos do seu curso, tem um vasto conjunto de opções disciplinares que terão de ser contempladas aquando do processo de *Construção dos Horários*.

Geralmente, algumas considerações que é necessário ter em relação à construção de horários escolares (geralmente, escolas secundárias) e a construção de horários universitários prendem-se com as cargas horárias dos docentes e dos alunos. No caso de horários escolares, os docentes têm uma carga horária superior, o que obriga a que a construção dos seus horários tenha um peso maior no algoritmo de resolução do problema, do que nos horários universitários.

2.5 CONSTRUÇÃO DE HORÁRIOS SEMI-AUTOMÁTICOS / AUTOMÁTICOS

Muitos autores acreditam que o problema da *Construção de Horários* não pode ser resolvido completamente, de uma forma automática *[Frangouli et al. 1995]*, *[Falcão 1990]*, *[Liatsos 1995]*. São duas as justificações básicas apontadas: por um lado, há razões que fazem um horário melhor que outro, que não podem ser facilmente expressas num sistema automático. Por outro lado, como o espaço de pesquisa é geralmente enorme, a intervenção humana pode desviar a pesquisa para uma determinada direcção, que o sistema sozinho podia ser incapaz de encontrar. Por estas razões, a maioria dos sistemas permite o ajuste manual dos resultados finais, existindo ainda outros que exigem mesmo, uma maior intervenção humana por parte do utilizador.

Muitos autores, defensores da utilização de sistemas de geração de horários semi-automáticos, chamam a atenção para a componente humana, como é o caso de *[Werra 1985]*: "*Parece-me que o uso de computadores não é largamente aceite pelos Docentes, que não gostam de ser escalonados*

por uma máquina". Este tipo de expressões reforça a ideia da implementação de sistemas interactivos de geração de horários, onde os interesses específicos terão maiores possibilidades de serem considerados.

Por outro lado, surgem os defensores da implementação de sistemas automáticos de construção de horários como *[Schaerf 1995]*, que justifica o interesse na utilização de sistemas automáticos, da seguinte forma:

> *"A solução manual do problema de construção de horários, normalmente, requer diversos dias de trabalho. Mais ainda, a solução obtida poderá ser insatisfatória em relação a algo. Por exemplo, a um aluno pode não ser permitido frequentar determinadas disciplinas que pretende, porque estas estão escalonadas ao mesmo tempo".*

Determinadas instituições têm interesse em sistemas semi-automáticos, enquanto outras preferem aplicações que produzam automaticamente parte da solução, podendo editar/alterar depois a solução proposta. Tanto um como outro tipo de aplicação apresenta vantagens e desvantagens. No caso dos sistemas semi-automáticos, as vantagens básicas são:

- *Possibilidade de visualização constante da evolução do horário, e como tal, possibilidade de efectuar alterações imediatas no mesmo;*
- *Maior interactividade com o utilizador;*
- *Facilidade de edição/alteração de restrições específicas de determinado recurso (ex: normalmente é mais fácil considerar os interesses específicos de um docente ao nível das preferências temporais).*

Como desvantagens, os sistemas semi-automáticos de construção de horários apresentam:

- *Semanas de trabalho manual e tedioso;*
- *Maior probabilidade de ocorrência de erros humanos;*
- *A pior qualidade da solução obtida, medida por critérios objectivos.*

Os sistemas automáticos apresentam como vantagens básicas:

- *Uma maior rapidez no processo de construção dos horários;*
- *Menos trabalho para a obtenção de resultados, depois de bem definidas as características do problema.*

As desvantagens básicas dos sistemas automáticos são:

- *A necessidade de uma definição o mais exaustiva possível do problema, e todas as componentes associadas;*
- *A impossibilidade de alteração de parâmetros durante o processo de geração dos horários.*

De uma forma geral, os sistemas automáticos de construção de horários (com ou sem possibilidade de edição e alteração interactiva de soluções, posteriormente à obtenção das mesmas) surgem como aqueles em que a comunidade científica vem dando maior atenção no seu desenvolvimento, aplicando diversas técnicas, algumas das quais já referenciadas.

2.6 RESTRIÇÕES

2.6.1 INTRODUÇÃO

As restrições são um elemento fundamental em todo o processo de *Construção de Horários*, pois são elas que determinam a flexibilidade de alocação dos recursos e eventos da instituição. Em qualquer sistema de construção de horários, um dos problemas principais que se coloca, é a gestão do elevado número de restrições que é necessário respeitar, de modo a obter bons horários. É evidente, que quanto maior o volume de dados a tratar, maior será o número de restrições a ter em atenção, e como tal, algumas delas poderão não ser respeitadas.

As restrições consideradas num problema de *Construção de Horários* são de diversos tipos, devendo ser atribuído um grau de preferência a cada uma delas, o que permitirá estabelecer prioridades na resolução do problema. Geralmente, os tipos de restrições dividem-se em dois: as restrições flexíveis e as restrições rígidas. Enquanto as primeiras permitem alguma flexibilidade na sua utilização, tal já não acontece com as segundas que impõem o respeito total.

2.6.2 TIPOS

As restrições de horários são muitas e variadas. Apresentam-se a seguir algumas dos tipos mais comuns:

- ***Atribuições:*** *Um recurso poderá ser atribuído a outro de tipo diferente, ou a um encontro. Por exemplo, um docente prefere leccionar numa determinada sala específica (por qualquer razão), ou exame deverá decorrer numa sala ou edifício em particular;*
- ***Atribuições de Tempo:*** *Um encontro ou um recurso pode ser atribuído a um tempo. Esta restrição pode ser usada para especificar dias em que um docente não se encontra disponível, ou pré-atribuir um tempo a um encontro em particular;*

- ***Restrições de Tempo entre Encontros:*** *Exemplos comuns desta classe de restrições são que uma aula em particular deverá ter lugar antes de outra, ou um conjunto de aulas deverá ser escalonado simultaneamente;*
- ***Grau de Dispersão de Encontros:*** *Os encontros devem ser dispersos ao longo do tempo. Por exemplo, nenhum aluno deverá ter mais do que três ou quatro aulas num dia;*
- ***Coerência de Encontros:*** *Estas restrições são definidas para produzir horários mais organizados e convenientes, e muitas vezes vão em sentido contrário às restrições de "Dispersão de Encontros". Por exemplo, um docente prefere ter todas as suas aulas em três dias, obtendo dois dias de aulas livres;*
- ***Capacidade das Salas:*** *O número de alunos numa sala não deverá exceder a sua capacidade;*
- ***Continuidade:*** *Quaisquer restrições, cujo objectivo principal é garantir que determinados acontecimentos com os horários dos alunos são constantes ou previsíveis. Por exemplo, aulas da mesma disciplina devem ser escalonadas na mesma sala, ou à mesma hora de determinados dias.*

2.6.2.1 Restrições Rígidas

Apresentam-se de seguida algumas das restrições rígidas a respeitar:

- *Nenhum docente pode leccionar dois ou mais eventos simultaneamente;*
- *Um evento só pode ser seleccionado e atribuído a um período de tempo, se os alunos de determinada turma não estão atribuídos a outro evento, no mesmo período de tempo;*
- *O número de salas deve ser suficiente para receber todas os eventos definidos;*
- *A capacidade da sala tem de ser respeitada, e para tal é necessário ter atenção aquando da definição das turmas;*
- *O Serviço Docente atribuído deve ser suficiente para leccionar todas as aulas;*
- *Restrições relativas à carga de trabalho – a carga de trabalho definida não pode ser superior a um dado limite;*
- *Restrições de ordenamento – um evento pode ser escalonado exactamente, pelo menos ou no máximo número de tempos (ou dias) antes de outro evento. Outro tipo habitual de restrição estabelece que existem exactamente, pelo menos ou no máximo, um dado número de tempos (ou dias) de intervalo entre dois eventos (sem qualquer preocupação com o evento que surge primeiro). Estas designam-se restrições de ordenação.*

2.6.2.2 Restrições Flexíveis

Como restrições flexíveis podemos considerar aquelas cujo domínio é flexível ou seja, que poderão não ser respeitadas, total ou parcialmente, se tal for absolutamente necessário para a construção de um horário. Apresentam-se de seguida algumas das restrições flexíveis a considerar:

- *Preferências gerais dos docentes, relativamente às disciplinas a leccionar em cada período;*
- *Preferências e indisponibilidades de tempos lectivos do docente ou grupo de docentes;*
- *Preferências e indisponibilidades de tempos lectivos do aluno ou grupo de alunos (turma);*
- *Preferências e indisponibilidades de tempos lectivos por sala ou grupo de salas;*
- *Uma sala pode suportar um número limitado de eventos (geralmente 1) num determinado tempo;*
- *Partes de eventos não devem ter lugar no mesmo dia;*
- *Restrições de dispersão – a dispersão de eventos é importante e esta restrição encontra-se relacionada com a anterior, na medida em que como partes de eventos não devem ter lugar no mesmo dia, também seria importante dispersar essas partes o mais possível pelo horário.*

Após o registo de todos os dados no sistema, dever-se-á realizar um teste prévio, de modo a analisar a validade destas e outras restrições para seguidamente poder passar-se para outra fase do processo de *Construção de Horários*.

2.7 QUALIDADE DE UM HORÁRIO

Geralmente, é difícil definir um conceito de horário óptimo *[Bloomfield e McSharry 1979]*, e como tal é necessário conhecer previamente os critérios, que variam de problema para problema, e que vão servir para determinar a qualidade do horário. De uma forma geral, o que é usado como critério é o número de restrições que não são respeitadas, sendo-lhes atribuídos pesos. *[White e Chan 1979]* afirmaram que: "*Os critérios que vão determinar um bom horário, são fáceis de definir mas difíceis de alcançar*". Alguns autores atribuem custos aos critérios definidos, e depois de encontrarem uma solução para o problema, procuram sucessivamente optimizar a solução obtida, ou seja encontrar a mais "barata" *[Frangouli et al. 1995]*.

Deste modo, a definição de critérios de qualidade de um horário terá de ser realizada, de forma a agradar às entidades directamente interessadas no resultado do processo de *Construção de Horários* (Reitoria, Docentes e Alunos). A reitoria espera que não hajam quaisquer sobressaltos no funcionamento da instituição. Os docentes esperam que as suas preferências sejam o mais possível

respeitadas e os alunos, por sua vez, esperam poder beneficiar com os horários atribuídos, pelos quais vão reger o seu tempo.

Um horário correcto é aquele que satisfaz todas as restrições impostas pelos seus dados. Apesar de satisfeita esta condição, tal não é suficiente para um horário ficar completo. Para se ter uma ideia geral, um horário correcto pode ser um que gere os dados por forma a compactar todas as aulas em três dias, numa base de cinco dias por semana. O grau de qualidade assenta no maior ou menor respeito pelos diversos critérios definidos.

O primeiro desses critérios encontra-se relacionado com o balanceamento das aulas pelos semestres considerados. Por exemplo, se um semestre contém um total de trinta períodos de leccionação semanais, então é necessário tomar algum cuidado para que o número de períodos de leccionação semanais, para esse semestre em particular, se encontre o mais possível próximo da média. Mais, as aulas de todas as disciplinas num dia deverão ser o mais concentradas possível, por forma a não existir nenhum tempo livre entre as diversas disciplinas. Deverão existir critérios para os docentes, que não deverão leccionar mais do que um determinado limite de horas diárias de aulas.

Outro critério é originário da duração de uma disciplina. Uma disciplina com um total de cinco períodos de leccionação é melhor ser dividida em duas ou três aulas. Se for esse o caso, então essas aulas deverão ter a maior distância (dias a separar) possível entre elas no horário.

O quarto critério tem a ver com a minimização de movimento de alunos entre aulas de diferentes disciplinas. Isto é, num semestre em particular deverão passar o maior tempo possível na mesma sala, num determinado dia. A capacidade das salas cria outro critério, que tem a ver com a sua utilização. Uma sala deverá albergar disciplinas cuja frequência seja a mais próxima da sua capacidade. Desta forma, é obtida a maximização de utilização.

Outros dois critérios finais estão directamente ligados com as preferências daqueles que são afectados pelo horário. Essas preferências, normalmente são originárias dos docentes e do corpo estudantil. Por exemplo, apesar de um docente estar disponível para leccionar durante um período específico, pode no entanto preferir, por qualquer razão, não leccionar. Por outro lado, disciplinas de tipos diferentes, que em circunstâncias normais são escalonadas simultaneamente, o corpo estudantil prefere que tal não aconteça, talvez porque existe um elevado número de alunos que prefere frequentar essas mesmas disciplinas.

Até agora, é possível perceber que todos os critérios referidos são preferências e não restrições. Isto significa que elas podem ser relaxadas ou mesmo ignoradas, por forma a ser construído um horário de maior qualidade. No entanto, alguns dos *standards* de qualidade encontram-se obviamente em

oposição. Por exemplo, uma melhor utilização pode significar que os alunos deverão ser movimentados entre diferentes salas. Provavelmente, é impossível construir um horário de forma a que todos os critérios sejam respeitados. Isto leva-nos de volta à natureza subjectiva de qualidade de um horário.

De uma forma geral, num problema de *Construção de Horários*, seja qual for o seu tipo, é necessário definir funções de avaliação (*benchmarking*) da solução apresentada, para que diferentes soluções possam ser comparadas e analisadas por forma a decidir qual a adoptar. Normalmente, essa decisão é baseada no peso obtido pela função de avaliação, que por sua vez costuma ser construída com base nas restrições consideradas no problema.

2.8 SOFTWARE COMERCIAL PARA A CONSTRUÇÃO DE HORÁRIOS

2.8.1 INTRODUÇÃO

A partir da análise do mercado de aplicações de construção de horários, constata-se a existência de diverso *software* deste tipo. A maioria das aplicações existentes apresenta um processo automático de construção de horários. A apresentação gráfica das aplicações é diversa, dependo por vezes da metodologia adoptada na construção dos horários.

Das aplicações analisadas destacam-se sete, sendo as primeiras três estrangeiras, das mais utilizadas a nível internacional. De seguida são apresentados três pacotes de *software*, dois dos quais portugueses e um terceiro estrangeiro, muito utilizados por escolas nacionais. Finalmente, é apresentado um pacote integrado de escalonamento.

2.8.2 ADE

Nesta aplicação nota-se uma grande evolução no *software* de construção de horários, face a outros analisados. O *ADE* é um sistema automático extremamente poderoso, aliado a uma simplicidade de utilização e uma interface amigável, o que o torna num dos sistemas mais poderosos existentes no mercado. Utilizando o sistema *ADE*, existem 4 passos para gerar os horários finais:

- *Definição dos critérios de optimização;*
- *Validação dos dados introduzidos;*
- *Pesquisa da primeira solução (durante esta pesquisa que pode demorar uns minutos, o utilizador tem permissão para interromper o processo e modificar todos os dados ou atribuições já efectuadas);*
- *e, finalmente, a melhoria da solução (procurando optimizar os critérios definidos).*

Encontram-se associadas diversas características ao *ADE*, entre as quais:

- *Solução Aberta – Cliente – Servidor;*
- *Alocação automática e optimização;*
- *Interface interactiva, prática e funcional, que permite gerir os horários dos alunos, docentes, salas e recursos;*

Na Figura 3 pode ser visualizado um formulário típico da aplicação.

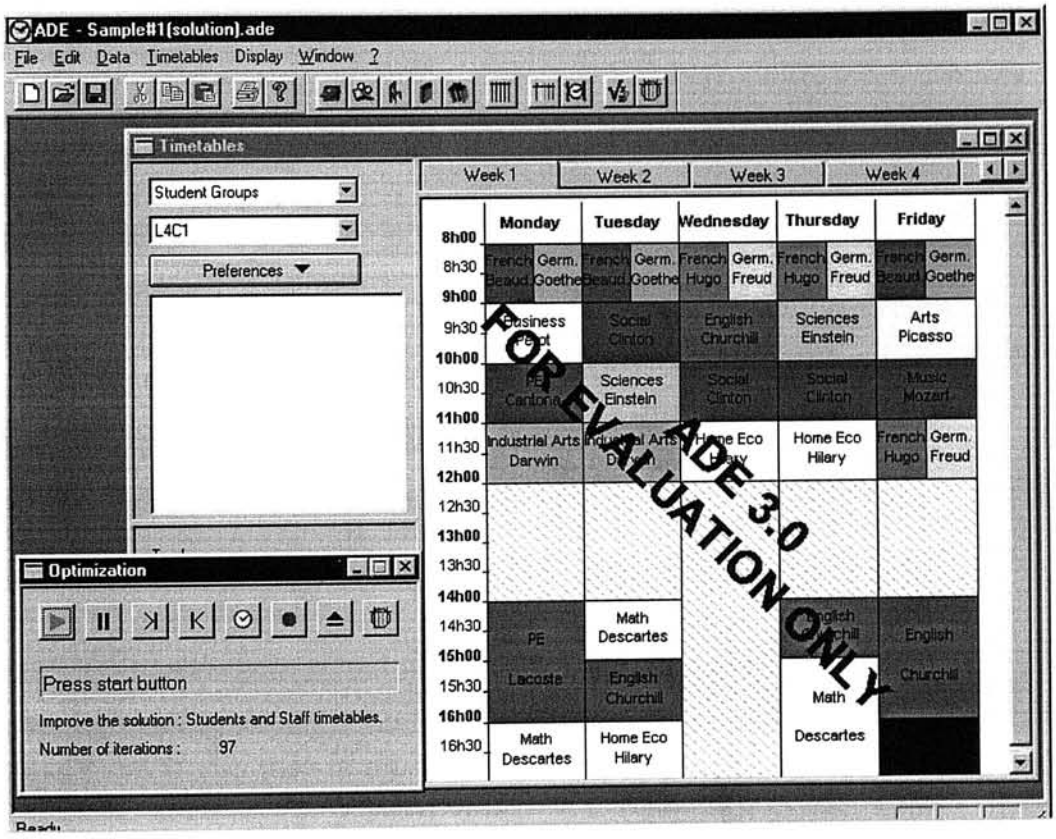


Figura 3: Formulário típico da aplicação *ADE*

De uma forma geral, o pacote de *software ADE* apresenta-se como uma excelente solução para o problema de *Construção de Horários*.

2.8.3 CELCAT

Uma análise geral desta aplicação permitiu compreender que se trata de um sistema semi-automático para a construção de horários, que procura garantir a inexistência de alocações duplas nos horários (dos docentes, turmas e salas), evitar a alocação de salas inadequadas (em dimensão

ou equipamentos disponíveis) e providenciar uma base sólida para tratar todo o processo de construção e distribuição de horários. Utiliza uma interface *Windows* típica incluindo menus, janelas sobreponíveis, barra de tarefa, barra de estado e auxílio on-line. O módulo de entrada de dados do sistema permite efectuar a importação de dados ou a introdução destes directamente no sistema. Existe possibilidade de extrair os dados, representados no formato *CELCAT*, para diversas aplicações, nomeadamente o *MS-Access* e o *MS-Word*. É possível visualizar na Figura 4 um formulário típico da aplicação.

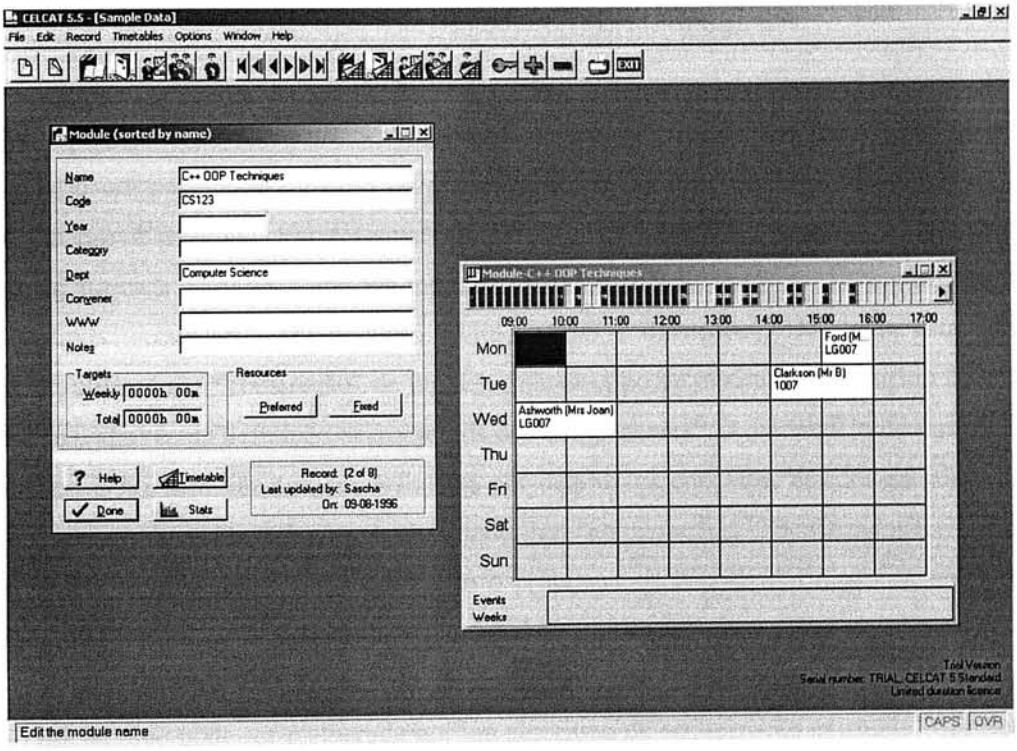


Figura 4: Formulário típico da aplicação *CELCAT*

Possuí várias características, destacando-se as seguintes:

- *Permite a construção de horários de forma centralizada, mas também a nível departamental, atribuindo níveis de acesso aos utilizadores que pretendam trabalhar em rede, acelerando, desta forma, o processo de construção dos horários;*
- *O CELCAT possui um vasto número de funções de "aconselhamento", por forma a ser mais fácil localizar recursos para os eventos a escalonar;*

- *Existe flexibilidade na definição e estruturação dos recursos existentes. Por exemplo: é possível definir uma sala para aulas teóricas com uma capacidade de 25 lugares, mas também redefini-la como um laboratório informático com uma capacidade de 10 lugares;*
- *Existem estatísticas que permitem controlar todos os recursos do horário. É possível mesmo calcular quantas horas um docente lecciona numa determinada sala ou quanto tempo está com determinado grupo de alunos.*

Esta aplicação apresenta-se também como uma boa hipótese de utilização no processo de *Construção de Horários* escolares ou universitários de uma instituição.

2.8.4 MIMOSA

O *Mimosa* é um pacote de *software* de planeamento de horários e actividades lectivas, para o sistema operativo *Windows*. Foi desenvolvido para ser, na medida do possível, o mais flexível e versátil, de forma a fornecer apoio em todas as áreas de planeamento escolar em instituições de diversos tipos (secundário, universitário, entre outros) e dimensões. De facto, esta aplicação permite ao utilizador a "construção" de um determinado tipo de instituição de ensino (que poderá nem existir), através da definição de conjuntos de "componentes". Ao contrário de outros pacotes de *software*, o *Mimosa* não contém definidos, por defeito, os conceitos básicos da construção de horários (turmas, docentes, salas, etc.), chamando-lhes "componentes". O utilizador tem a tarefa de nomear esses "componentes" de acordo com o problema. A fase de definição de um problema demora algum tempo, sendo necessária alguma habituação ao *software*. O tempo necessário para a construção dos horários está dependente de como se define o problema. A aplicação gera uma solução inicial, que pode ser melhorada por um algoritmo de optimização existente. Apresenta-se, na Figura 5, um formulário típico da aplicação.

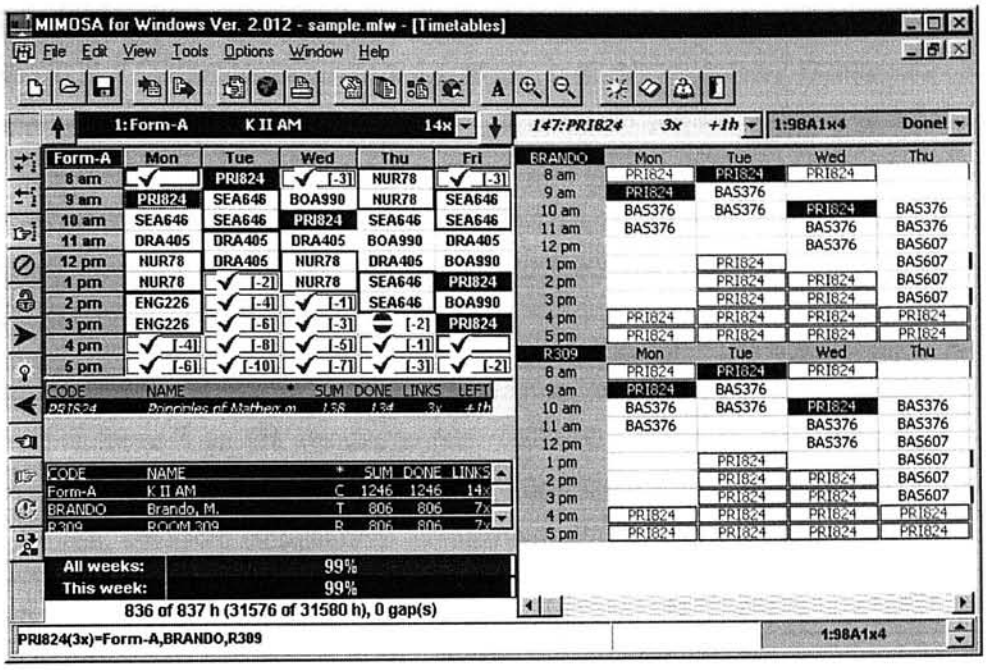


Figura 5: Formulário típico da aplicação *MIMOSA*

Diversas características encontram-se associadas à aplicação, destacando-se as seguintes:

- *Contém diversos menus e icons de acesso às funcionalidades da aplicação;*
- *A estrutura das disciplinas pode ser alterada em qualquer fase do processo de construção dos horários;*
- *Permite a transferência de dados entre diversas aplicações;*
- *Ficheiros em rede, de diversos utilizadores, podem ser combinados e distribuídos;*
- *Como curiosidade, de referir que, apesar de ainda limitadas, existem potencialidades ao nível da tecnologia WAP, que permitem a visualização de horários em telemóveis.*

O *Mimosa* é uma aplicação bastante completa e abrangente, necessitando apenas de ser utilizada por alguém com bastantes conhecimentos da problemática da construção de horários e treino suficiente, para tirar proveito de todas as capacidades da aplicação.

2.8.5 GESTHOR

O *Gesthor* é um produto recente, fruto do desenvolvimento de uma empresa portuguesa (*Cronológica*) e que cada vez mais se vai implantando no mercado nacional, em escolas básicas, secundárias, profissionais e superiores.

O *GestHor* pode ser visto, em parte, como um *Sistema de Apoio à Decisão*, ou seja, tem por objectivo auxiliar o utilizador na tarefa de construção de horários escolares. O utilizador continua a ter o poder total para decidir quem vai leccionar determinado evento, em que períodos de tempo e em que sala, sendo o papel principal do *Gesthor* validar essas decisões, assegurando que elas não introduzem problemas, como por exemplo, sobreposição de aulas. No entanto, existe uma versão do *software*, que permite a construção automática de horários, após o processo de definição de dados e restrições do problema.

Para além de assegurar que os horários não contêm erros, este *software* vai aconselhando o utilizador por forma a ter em atenção as preferências dos vários intervenientes (especialmente docentes e alunos). Não se limita, somente, a simplificar o processo de construção de horários, mas fornece também um apoio indispensável na introdução dos dados relativos ao estabelecimento de ensino.

No final do processo de construção dos horários, o *Gesthor* permite ao utilizador avaliar a qualidade do horário final produzido, com base nos mais variados critérios (satisfação das preferências dos docentes e das turmas, número de "furos", períodos de tempo livre entre aulas, entre outros).

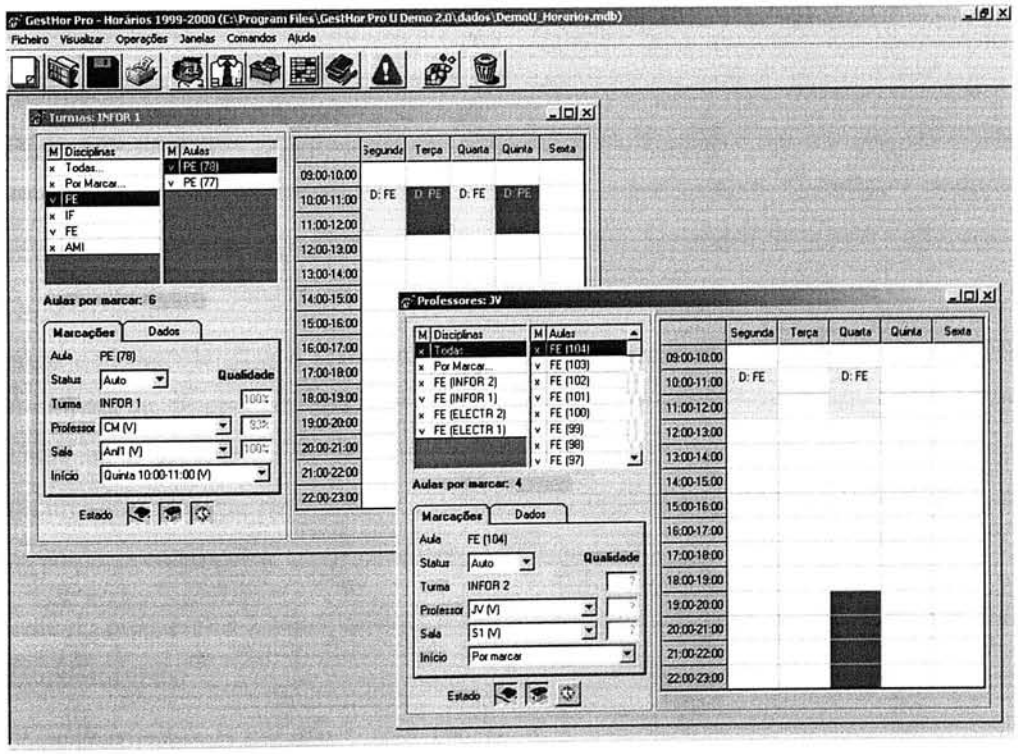


Figura 6: Formulário típico da aplicação *GESTHOR*

Este pacote de *software* tem algumas características inovadoras como:

- *A utilização de um assistente para a definição de novos problemas, que segue uma lógica sequencial de eventos;*
- *Um estilo agradável e funcional das janelas;*
- *Uma boa adaptabilidade para diversos estabelecimentos de ensino;*
- *A possibilidade de adopção de um método semi-automático ou automático de construção dos horários;*
- *Facilidade de aplicação dos conceitos de junção e desdobramento.*

No entanto, uma desvantagem assinalável deste *software* é o elevado tempo que necessita, para análise e aceitação de determinadas operações de escalonamento (dependendo da dimensão do problema em causa), executadas pelo utilizador (por exemplo, a alocação de aulas num horário).

O surgimento desta aplicação, desenvolvida por uma empresa nacional, demonstra claramente a percepção de algumas pessoas para a necessidade de desenvolvimento de *software* específico para a construção de horários. Existem diversas vantagens a retirar desta atitude, e a principal é não haver necessidade de as instituições de ensino adquirirem *software* estrangeiro, por vezes inadequado à realidade nacional.

2.8.6 GPUNTIS

Trata-se de uma aplicação desenvolvida por uma empresa de análise informática, encontrando-se bastante divulgado entre as Escolas Básicas e Secundárias portuguesas. O *software* analisado tem uma apresentação em ambiente *Windows*, mas cujo correcto funcionamento com a aplicação exige alguma experiência e habituação ao ambiente apresentado.

A empresa que comercializa o pacote de *software* tem como grande vantagem a inclusão de um curso de formação, fornecendo também os manuais de apoio necessários. Neste curso é explicado resumidamente o sistema de informação associado ao processo de construção de horários, e é proposta uma metodologia para resolver este problema. Apesar de ser uma aplicação desenvolvida por uma empresa estrangeira, existe uma versão em português.

A vertente do programa é a construção de horários escolares dividindo-se a sua estrutura em cinco pontos fundamentais:

- *Recolha e tratamento de dados;*
- *Definição de critérios de ponderação e optimização;*

- *Análise do diagnóstico da aplicação;*
- *Elaboração automática de horários;*
- *Impressão dos horários.*

De seguida apresenta-se, na Figura 7 ,um formulário típico do programa analisado:

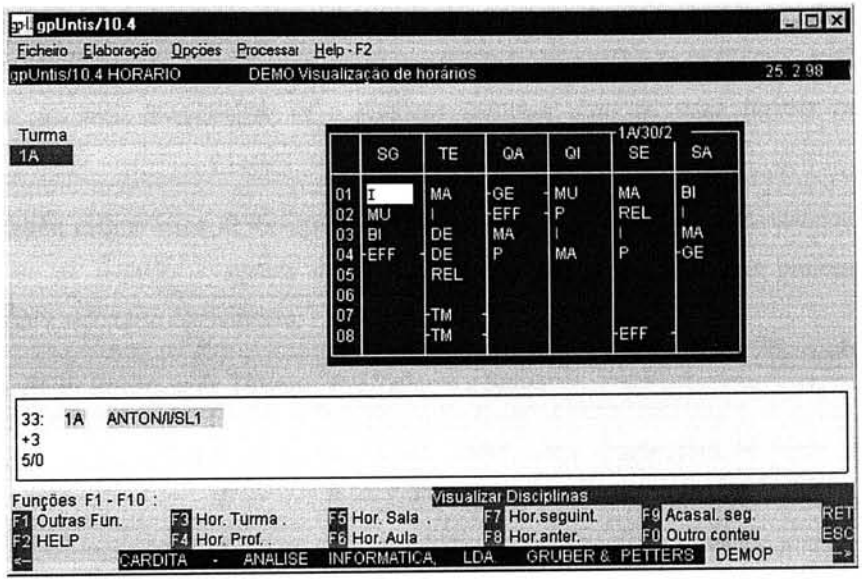


Figura 7: Formulário típico da aplicação GP-UNTIS

Trata-se de um *software* já com algum tempo de desenvolvimento mas que, no entanto, o ambiente utilizado não se encontra totalmente optimizado para *Windows*, fazendo ainda uso de propriedades do *MS-DOS*. Como tal, torna-se pouco atractivo para alguém que pretenda utilizá-lo face a outras aplicações que vão surgindo no mercado.

2.8.7 DCS-HORÁRIOS

O *DCS-HORÁRIOS* é mais uma aplicação desenvolvida por uma empresa nacional direccionada, essencialmente, para as escolas básicas e secundárias. Foi construída para funcionar no ambiente *Windows*, sendo simples e intuitiva de utilizar. A aplicação tem como funcionalidades principais:

- ***Introdução dos dados:*** *A entrada de dados encontra-se facilitada com a existência de rotinas para criar salas automaticamente; copiar planos curriculares de umas turmas para outras; colocar aulas sem sala; definir salas das turmas; definir carga horária de professores e turmas;*
- ***Elaboração automática de horários:*** *Satisfação das regras de elaboração de horários designadamente, "furos" ("slots" vazios nos horários) dos professores, máximo de horas seguidas (dos*

professores e turmas), máximo de tempos por dia (dos professores e turmas), línguas estrangeiras seguidas, educação física depois do almoço, disciplinas opcionais no inicio/fim de um bloco de aulas; distribuição da carga horária dos professores e turmas; inicio e duração do almoço das turmas e professores; reserva de tempos lectivos (para turmas, professores e salas), impedindo o gerador de colocar aulas nesses tempos; dia livre do professor (automático, definido ou nenhum), entre outras;

- ***Edição manual de horários:*** *Possibilidade de construção de horários de raiz ou editar/completar os horários resultantes da geração automática; possibilidade de construção de horários a partir da turma, professor ou sala; possibilidade de o professor leccionar duas ou mais turmas em simultâneo; possibilidade de desdobrar a turma até seis níveis em simultâneo;*
- ***Elaboração automática de calendários para reuniões de avaliação:*** *Possibilidade de imprimir calendário de reuniões e agenda de reuniões para cada professor; definir professores volantes, presidente e secretário das reuniões;*
- ***Impressão de horários de Turmas, Professores e Salas;***
- ***Facilidade de utilização:*** *Existência de um "tutor" para acompanhar ao longo do processo de construção dos horários; várias interfaces gráficas para tornar a introdução de dados mais rápida e intuitiva (Editor de horários, Editor de Salas e Mapa de distribuição de serviço).*

Na Figura 8 encontra-se representado o painel principal da aplicação.

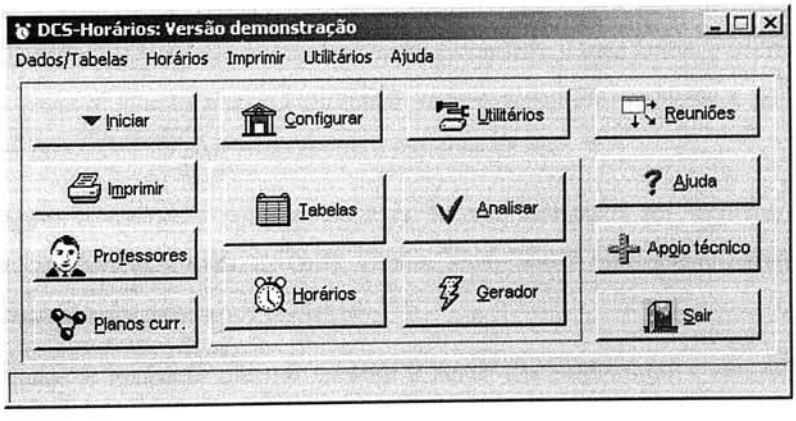


Figura 8: Formulário típico da aplicação DCS-HORÁRIOS

No caso de se tratar de uma instituição de ensino básico ou secundário, esta aplicação cobre as necessidades essenciais desse tipo de instituições de ensino.

2.8.8 SYLLABUS PLUS

O *Syllabus Plus* é um sistema completo de escalonamento, ou seja, é uma ferramenta integrada para fazer o melhor uso de todos os recursos das instituições de ensino. Uma larga variedade de serviços é oferecida pelo *Syllabus Plus*, e que inclui suporte técnico, formação e consultoria, bem como diversos seminários de informação.

Este sistema possui diversas funcionalidades, nomeadamente:

- ***Construtor de Horários:*** *escalonamento de turmas e restantes recursos da instituição;*
- ***Distribuição de Serviço Docente;***
- ***Calendarização de Exames;***
- ***Gestão de Bases de Dados:*** *suporte da utilização distribuída de outros módulos cliente;*
- ***Utilização de Tecnologias WEB:*** *permite o acesso por Internet/Intranet aos relatórios produzidos;*
- ***Troca de Dados entre Sistemas:*** *permite a transferência de dados entre diversos sistemas.*

Ao nível da construção de horários, existe um conjunto de características de vários tipos, associadas à aplicação, nomeadamente:

- ***Controle:*** *Esta aplicação traz consigo informação das principais áreas que são afectadas pela construção de horários, por forma a facilitar o escalonamento. Existe um balanceamento entre aquilo que os utilizadores podem fazer e o escalonamento automático. Desta forma, o utilizador consegue ter o controle sobre a solução final. O utilizador pode modelar um problema a partir de estruturas tradicionais fornecidas, ou pode então definir o problema de base.*
- ***Optimização:*** *A aplicação examina todos os factores associados aos recursos que é necessário considerar pelo horário – salas, docentes, alunos, disciplinas e equipamento, tentando optimizar o máximo possível todos os recursos existentes.*
- ***Configuração:*** *A aplicação constrói horários a partir das preferências e restrições definidas pelo utilizador, por forma a respeitar os interesses da instituição. Ou seja, o utilizador tem plenos poderes de configuração de detalhes como preferências de tempos, restrições de salas, serviço docente atribuído, entre outros.*
- ***Escalonamento Automático:*** *Existe a possibilidade de escalonar ou re-escalonar todo ou parte do horário da instituição. A resolução de conflitos é automática.*

O *Syllabus Plus* é uma aplicação recente, não tendo sido possível analisar uma demonstração do *software*, por este não se encontrar disponível. Mas, a descrição do seu conteúdo apresenta um

conceito de integração de componentes muito importante, no sentido de facilitar o processo de gestão de recursos numa instituição de ensino.

2.8.9 BREVE CONCLUSÃO DA ANÁLISE DO SOFTWARE PARA CONSTRUÇÃO DE HORÁRIOS

De um modo geral, nota-se que os sistemas de construção de horários se apresentam cada vez mais completos, salientando-se as suas potencialidades gráficas, que em muito facilitam o trabalho dos utilizadores, pela sua interactividade, intuição e simplicidade de utilização.

Existe um mercado em contínuo crescimento para o *software* de construção de horários, devido ao número de instituições de ensino que existem, bem como aquelas que vão surgindo, algumas das quais com um elevado número de cursos e como tal um elevado número de recursos que é necessário escalonar.

Com o evoluir das aplicações, cada vez mais o mercado exigirá aplicações integradas, que incluam para além do sistema de *Construção de Horários* também os sistemas de *Distribuição de Serviço Docente* e *Calendarização de Exames*. Exige-se também, sistemas que contenham integrados processos de construção semi-automática e automática de horários (por exemplo, a pessoa responsável pela construção dos horários de uma dada instituição, pode pretender um sistema que lhe permita construir de forma semi-automática os horários dos docentes e dos alunos, e após concluída essa fase, o sistema encarregar-se-ia de atribuir as salas a todas as aulas). Finalmente, os horários poderiam também ser automaticamente publicados na Internet.

Aplicações deste género começam a surgir, e a vulgarizarem-se no mercado, como é o caso do *Syllabus Plus*.

2.9 CONCLUSÃO

Neste capítulo foi efectuada uma abordagem ao *Problema da Construção de Horários em Universidades*, sob diversas perspectivas e considerações necessárias. Foram apresentados os problemas associados: *Calendarização de Exames* e *Distribuição de Serviço Docente*. Realizou-se um comparativo entre o problema dos horários escolares em oposição aos horários universitários. Apresentaram-se ainda vantagens e desvantagens dos horários semi-automáticos face aos automáticos. Foram explicados os tipos de restrições existentes e formas de avaliação da qualidade de um horário. Finalmente, apresentaram-se alguns pacotes de *software* comercial, nacionais e estrangeiros, utilizados por escolas e universidades.

Após análise do *estado da arte* do *Problema da Construção de Horários*, segue-se um capítulo de abordagem a diversas técnicas existentes para a resolução automática do problema. Estas técnicas foram evoluindo ao longo do tempo, acompanhando o crescimento do poder computacional das máquinas usadas. Em algumas das técnicas abordadas, nomeadamente nas *meta-heurísticas*, para além das características associadas, são apresentados os respectivos algoritmos base de resolução do problema e suas aplicações práticas à geração automática de horários.

Capítulo 3

Técnicas para a Construção de Horários

3.1 INTRODUÇÃO

Ao longo do tempo, muitas e variadas técnicas de construção de horários 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 aquelas que apresentavam o maior número de restrições. As diferentes técnicas diferiam apenas nos critérios usados para definir os tipos de restrições. Mais tarde, começou-se a aplicar técnicas automáticas de resolução deste problema, como sejam a *Programação Inteira* e a *Coloração de Grafos*. Esta última tornou-se, de forma progressiva, naquela sobre a qual os autores mais se debruçaram no desenvolvimento de algoritmos.

Os diferentes métodos para resolver, com sucesso, problemas reais de construção de horários, em quase todos os casos são usados somente numa escola ou instituição em particular, e raramente são devidamente comparados entre si. 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.

3.2 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, destacando-se *[Carter 1986]*, *[Johnson et al. 1991]*, *[Leighton 1979]*, *[Mehta 1981]*, *[Welsh e Powell 1967]*.

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. O problema reduz-se a ir encontrando colorações 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. Nesta altura não é preocupação saber as razões dos conflitos, ou seja, só interessa o facto de que certos eventos não podem ser escalonados juntos (por quaisquer razões). Isto pode ser representado como um grafo, que consiste num conjunto **V** de vértices $\mathbf{V} = \{v_1, ..., v_n\}$, 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 $\mathbf{E} = \{i, j \in N \bullet conflito(ev_i, ev_j) \bullet (v_i, v_j)\}$ 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, por forma a 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 de forma a 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. Isto é conhecido como o *Problema da Coloração de Grafos*, e chegou a ser bastante discutido na literatura do género.

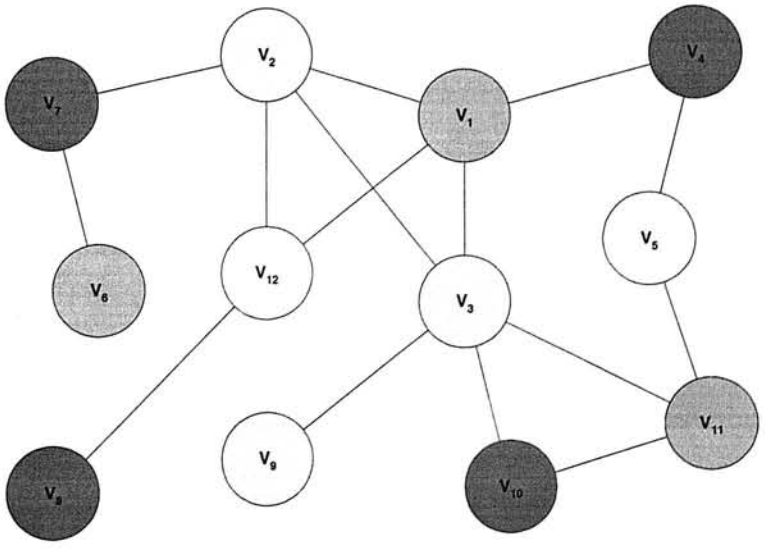


Figura 9: Modelo simples de um grafo representado com algumas restrições

Na Figura 9 apresenta-se um exemplo de um problema simples de construção de horários, com algumas restrições:

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.

3.3 PROGRAMAÇÃO INTEIRA

Genericamente, um problema de programação inteira não é mais do que um problema de programação linear, em que algumas ou todas as variáveis a optimizar, têm de ser positivas. *[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.*

A partir de uma notação matemática podemos representar formalmente o problema de uma maneira alternativa. *[Tripathy 1984]* formulou o problema de construção de horários aplicando uma técnica de programação linear inteira binária. Pode ser usada uma formulação similar para descrever o problema geral de construção de horários e calendarização de exames, constituído por E eventos e P períodos, definindo primeiro:

$t_{ip} = 1$	*1 se o evento i é escalonado no período p* *0 em caso contrário*
c_{ij}	*número de alunos a assistir a ambos os eventos i e j*

Um horário válido é então sujeito à restrição que todos os eventos devem ser escalonados uma e única vez no horário (***Equação 1.1***). Segundo, se dois eventos são escalonados no mesmo período então não devem estar em conflito (***Equação 1.2***).

$$\sum_{p=1}^{P} t_{ip} = 1, \qquad \forall i \in (1,\ldots,E) \qquad \textit{(Equação 1.1)}$$

$$\sum_{i=1}^{E-1}\sum_{j=i+1}^{E}\sum_{p=1}^{P} t_{ip}t_{jp}c_{ij} = 0 \qquad \textit{(Equação 1.2)}$$

Enquanto este modelo descreve somente as restrições básicas de um horário, pode facilmente ser estendido para descrever formalmente uma maior variedade de outras restrições existentes num problema de construção de horários.

3.4 HEURÍSTICAS SIMPLES

A ideia de uma heurística é a de usar a estrutura do problema para obter soluções satisfatórias de uma forma eficiente. É uma forma de abordagem dos problemas combinatórios de resolução complicada. Na Figura 10 encontra-se representado um gráfico que traduz essa necessidade de conciliação de 2 factores: a eficiência (rapidez de execução) do algoritmo de resolução e a sua eficácia (qualidade das soluções obtidas), devendo definir-se pontos (***a*** ou ***b***, no gráfico) de compromisso entre estes dois factores.

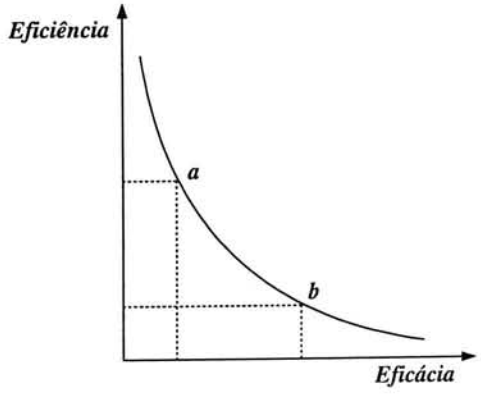


Figura 10: A relação entre Eficiência e Eficácia

Figura 11: Estrutura típica de uma abordagem heurística

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, as heurísticas começam a efectuar algumas trocas, por forma a acomodarem outras aulas. Um exemplo típico deste método é o sistema *Schola [Uhlemann et al. 1969]*, apresentado por *[Schaerf 1995]*. 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 referido 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 saída. A estratégia **C** permite, de uma forma limitada, *backtracking*, por forma a recuperar dos "erros" da estratégia **A**.

Normalmente, as heurísticas simples designam-se por *heurísticas de melhoramentos*, pois as suas estratégias de pesquisa baseiam-se em procurar a melhorar solução na vizinhança de soluções possíveis, até encontrar uma solução melhor do que a actual. Ou seja, estas heurísticas constituem a base de um processo iterativo, cuja regra de paragem é, em geral, simples (número de iterações, por exemplo).

3.5 AS META-HEURÍSTICAS

3.5.1 INTRODUÇÃO

Mais recentemente, algumas aproximações a técnicas de pesquisa baseadas na Inteligência Artificial e Investigação Operacional, designadas por *Meta-Heurísticas [Osman 1995], [Howe 1999]* foram aparecendo. As meta-heurísticas, são técnicas inteligentes de optimização *[Karaboga e Pham 1998]* que constituem uma estrutura geral para construir heurísticas para resolver problemas tipicamente NP-Completos. Podem também aplicadas a problemas combinatórios em que uma solução pode ser calculada em tempo polinomial mas tal não é prático *[Osman 1995], [Howe 1999]*. 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. A ideia geral do meta é de nível (superior). 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 melhoramentos), constituem métodos de pesquisa genérica aplicáveis a grandes conjuntos de problemas de optimização.

3.5.2 ARREFECIMENTO SIMULADO

3.5.2.1 Introdução

O *Arrefecimento Simulado* impôs-se naturalmente, no início dos anos 80, pela sua simplicidade de implementação e ampla aplicabilidade. A utilização de um algoritmo de *Arrefecimento Simulado* foi proposto por *[Kirkpatrick et al. 1983]*. No "coração" do algoritmo, encontra-se uma 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, conhecido como metalurgia. 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.

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. Com este critério de aceitação, é possível sair de óptimos locais. Conforme a temperatura vai diminuindo, esta probabilidade de aceitação vai diminuindo de forma a que , no final, só soluções melhores são aceites.

3.5.2.2 Princípio de Funcionamento

O desenho de um bom algoritmo de arrefecimento não é trivial, e geralmente compreende cinco partes *[Melicio et al. 1998]*:

- ***Espaço de configuração:*** *O conjunto de configurações permitidas pelo sistema deve facilitar a representação directa de cada estado e a fácil geração de perturbações;*
- ***Conjunto de movimentos:*** *O conjunto de possíveis movimentos deverá ser suficientemente rico que todas as soluções razoáveis possam ser encontradas a partir da aplicação de uma sequência de movimentos desse conjunto. Adicionalmente, esses movimentos deverão ser relativamente "baratos" de calcular, uma vez que serão realizados diversos movimentos;*
- ***Função de custo:*** *A função deverá ser calculável de forma incremental para que o tempo de avaliação de cada movimento seja mínimo;*
- ***Escalonamento de arrefecimento:*** *A forma como o parâmetro de controle, designado como temperatura* **t** *baixa, durante o arrefecimento, é crucial. Iniciando muito frio, parando muito quente, ou arrefecendo muito rapidamente, o algoritmo só produzirá soluções sub-óptimas;*
- ***Estruturas de dados:*** *A habilidade de propor e avaliar movimentos de forma eficiente, é muito dependente de uma boa representação dos objectos básicos no problema.*

O esquema de arrefecimento é provavelmente a parte mais importante do algoritmo, e o seu estudo cuidado é essencial para o sucesso desta técnica. Na Tabela 1 é apresentado um comparativo entre a *Simulação Termodinâmica* e a *Optimização Combinatória*.

Simulação Termodinâmica	***Optimização Combinatória***
Estados do Sistema	Soluções Admissíveis
Energia	Função Objectivo (Custo)
Mudança de Estado	Solução Vizinha
Temperatura	Parâmetro de Controlo
Estado 'congelado'	Solução (Heurística)

Tabela 1: Comparativo entre a *Simulação Termodinâmica* e a *Optimização Combinatória*

A seguir apresenta-se um exemplo básico do algoritmo de *Arrefecimento Simulado [Osman e Kelly 1996].*

1) Geração da solução inicial S (aleatória ou usando uma heurística)
(i) Define uma temperatura inicial T, e outros parâmetros de arrefecimento

2) Escolhe, aleatoriamente, $S' \in N(S)$ e computa $\Delta = C(S') - C(S)$

3) Se:
(i) S' é melhor do que S ($\Delta < 0$), ou
(ii) S' é pior do que S, mas "aceitável" pelo processo aleatório, à temperatura T actual $e^{(-\Delta/T)} > \theta$, (onde $0 < \theta < 1$ é um número aleatório).

***Então** substitui S por S'*
***Senão** retém o valor actual da solução S.*

4) Actualiza a temperatura T, dependendo de um conjunto de regras, incluindo:
(i) O tipo de arrefecimento usado
(ii) quer tenha sido obtido um melhoramento no passo 3) anterior
(iii) quer a vizinhança N(S) tenha sido completamente pesquisada.

c) Se o critério de paragem for verdade, pára, senão continua no passo 2).

Algoritmo 1: Algoritmo básico de *Arrefecimento Simulado*

Durante o processo de implementação existe um conjunto de decisões tácticas que é necessário tomar. São elas, decisões genéricas e decisões específicas.

Decisões Genéricas (esquema de arrefecimento)	***Decisões Específicas (do problema)***
T_k (função temperatura)	X_0 (solução inicial)
T_0 (temperatura inicial)	Estrutura de vizinhança
L_k (nº de iterações)	Cálculo de ΔC
Critério de paragem	

Tabela 2: Decisões Genéricas e Específicas a considerar num algoritmo de *Arrefecimento Simulado*

No caso das decisões genéricas, o *Arrefecimento Geométrico* é uma das estratégias mais utilizadas, e que consiste numa função geométrica $T_{k+1} = \alpha T_k$, com $0 < \alpha < 1$ (tipicamente, $0.8 \leq \alpha \leq 0.99$), que corresponde a um arrefecimento lento. Em certas variantes, o número de iterações por temperatura cresce de forma geométrica, traduzindo a importância de efectuar um maior número de iterações em temperaturas mais baixas, assegurando que se expressa totalmente um mínimo local. Deverão ser definidas regras de paragem, como por exemplo: C^* não melhora $y_1\%$, após N_1 temperaturas consecutivas ($1 \leq y_1 \leq 5$).

Quanto às decisões específicas, na escolha da solução inicial é necessário definir se é dada preferência a uma boa solução, em detrimento de uma solução aleatória. É importante definir uma boa estrutura de vizinhança, o que impõe uma topologia suave, mas não existem regras gerais.

3.5.2.3 Aplicação

[Schaerf 1995] apresenta um exemplo de aplicação de *Arrefecimento Simulado* ao *Problema de Construção de Horários*, desenvolvido por *[Abramson 1991]*. Considera, como extensão, a possibilidade que duas turmas diferentes tenham alunos em comum. Desta forma, a estrutura do seu trabalho passa a assentar na categoria da construção de horários por disciplina.

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]* escolheu uma taxa de arrefecimento de 0.9, no entanto, também experimentou com vários outros valores. O número de iterações efectuadas é da ordem dos 3 milhões.

3.5.2.4 Conclusão

Este algoritmo apresenta como principais vantagens a sua simplicidade e robustez. Exige longos tempos de computação, comparativamente a outros métodos. Existe um processo de convergência e entropia (em termodinâmica, é a função que define o estado de desordem de um sistema, ou seja, o valor que permite avaliar esse estado de desordem e que vai aumentando à medida que este evolui para um estado de equilíbrio). É possível melhorar o funcionamento do algoritmo, pela alteração de parâmetros, o que pode ser efectuado pelo utilizador.

3.5.3 PESQUISA TABU

3.5.3.1 Introdução

A *Pesquisa Tabu [Glover e Laguna 1993], [Glover e Laguna 1997]* é uma técnica de pesquisa local, e funciona mantendo uma lista das últimas ***n*** soluções testadas ou alterações efectuadas (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 opção e por aí adiante.

Quanto maior for a *Lista Tabu*, melhor se evita que o algoritmo entre em ciclo, mas tal, envolve maior tempo de CPU 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. Um movimento de vizinhança é considerado como o movimento de um único evento para um novo período. Como o tamanho da vizinhança, utilizando este critério, vai ser enorme para problemas práticos, somente uma fracção da vizinhança é calculada, proporcional ao número de eventos. Em segundo lugar, 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 é guardado. Em vez disso, a *Lista Tabu* é armazenada em termos de pares (evento, período), representando a posição do evento anterior a um 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 de não visita.

3.5.3.2 Princípio de Funcionamento

Durante a fase de implementação do algoritmo, existem diversas decisões a tomar e operações a efectuar, nomeadamente:

- *A especificação duma estrutura de vizinhança $V(x_k)$ ou $V'(x_k)$;*
- *A definição do tamanho da Lista Tabu e as soluções recentes;*
- *É essencial a definição do critério de aspiração (onde é definida uma solução suficientemente boa, permitindo o movimento, mesmo sendo tabu).*
- *Definir a regra de paragem.*

Existem outras considerações a ter em atenção, como sejam:

- *Registar atributos de movimentos e não últimas soluções (pode evitar-se um esforço excessivo, mas não impede os ciclos);*
- *Interditar a solução com um dado atributo pode ser restritivo (impedir o acesso a soluções que não foram visitadas).*

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 de aprendizagem, procurando imitar o comportamento humano. Apesar da falta de fundamentação teórica, 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.

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, pode guardar-se numa *Lista Tabu* as soluções já visitadas, embora 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 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[2]. 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 deverá fazer sempre parte de qualquer implementação de algoritmos de *Pesquisa Tabu*, é que a solução gerada seja melhor, que a melhor solução encontrada até ao momento.

Apresenta-se, de seguida, um algoritmo simples de *Pesquisa Tabu*.

```
INÍCIO
Geração do estado inicial s
T = ∅                    (lista tabu de movimentos proibidos)
s* = s                   (melhor solução)
p = q = 0                (contadores de iteração)
enquanto (p ≠ p_max) e (q ≠ q_max)
        escolha do melhor vizinho s' ∈ V(s), tal que é_tabu(s', T) = Falso
            (é_tabu(x) devolve Verdade sempre que x for considerado "tabu",
             considerando a lista tabu de movimento T)
        escolha do melhor vizinho s'' ∈ V(s), tal que é_tabu(s', T) = Verdade
            (satisfação do critério de aspiração)
        se f(s'') < f(s') então s' = s''
        se f(s') < f(s*) então s* = s' e q = 0
        Insere movimento(s', s) em T
        s = s'
        p = p + 1
        q = q + 1
Devolve s*
```

Algoritmo 2: Exemplo simples de um algoritmo de *Pesquisa Tabu*

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.

Tudo o que foi descrito relativamente às estruturas de vizinhança, na *Pesquisa por Arrefecimento Simulado*, mantém-se válido. Acresce aqui, a necessidade frequente de considerar e definir uma

[2] 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.

sub-vizinhança de x_n em vez da vizinhança completa. É frequente construir a sub-vizinhança por amostragem aleatória de $V(x_n)$. É importante escolher cuidadosamente os atributos aos quais se vai aplicar o estado *tabu*. Para além da desejável relação unívoca entre a proibição de um atributo e a proibição de uma solução, é necessário atender a questões como:

- *O espaço de memória necessário para armazenar os atributos tabu;*
- *A eficiência nas operações de inserção e retirada de atributos da lista tabu;*
- *A verificação rápida da existência de um atributo pertencente à lista tabu.*

Quanto ao tamanho da *Lista Tabu* (número máximo de valores diferentes dos atributos, a que simultaneamente se vai aplicar o estado *tabu*), não existe qualquer tipo de regra, nem mesmo empírica, apropriada a todos os problemas de optimização combinatória, para a definição do seu tamanho. A única forma de determinar o melhor tamanho para a *Lista Tabu*, ou seja, aquele que conduz a melhores resultados durante a pesquisa, é através da experimentação. Numa lista demasiado pequena existirão ciclos frequentes, e poderá ocorrer uma eventual paragem prematura do algoritmo. No caso da lista ser demasiado grande, há uma degradação da solução final, devido a demasiadas soluções não serem atingíveis.

3.5.3.3 Aplicação

[Costa 1994] e *[Schaerf e Schaerf 1995]* aplicaram a *Pesquisa Tabu* a um problema standard de Construção de Horários, numa escola. A representação escolhida por *[Schaerf e Schaerf 1995]* foi a mesma que *[Colorni et al. 1992]*, na qual a solução é representada como uma matriz $M_{m \times p}$, na qual, cada entrada m_{ik} contém o nome da turma que o docente irá encontrar no período k. Um movimento consiste na troca de duas aulas de um dado docente ou a mudança de uma aula para um período diferente. Por outro lado, *[Costa 1994]* empregou um tipo diferente de movimento, em que só era permitida a re-atribuição de uma única aula a um período diferente. No entanto, na sua representação, um único docente pode leccionar mais de uma aula ao mesmo tempo e como tal, pode ser efectuada uma troca de atribuições em movimentos consecutivos, deixando ambas as atribuições no mesmo período, no passo intermédio.

3.5.3.4 Conclusão

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.

3.5.4 ALGORITMOS GENÉTICOS

3.5.4.1 Introdução

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 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 e seleccionando a seguir as que apresentam melhores condições de sobrevivência à próxima geração.

3.5.4.2 Princípio de Funcionamento

Os *Algoritmos Genéticos* lidam com populações em vez de soluções individuais *[Grefory 1991]*. As as soluções interactuam, misturam-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 populações do mesmo tamanho N, 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).

A obtenção da geração (***n+1***) a partir da geração ***n*** ($X^{(n)}$) processa-se da seguinte forma:

- *Os melhores indivíduos (soluções) de* $X^{(n)}$ *são seleccionados;*
- *São efectuadas operações de cruzamento sobre os 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.*

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".

Quanto ao *crossover*, este é realizado pelos 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. Um exemplo do seu funcionamento é o seguinte: suponha-se que as soluções estão codificadas em *strings* binárias de 8 *bits* e que o par de indivíduos seleccionado é o seguinte:

Indivíduo 1 ⇒ **0 1 0 1 1 0 0 1**
Indivíduo 2 ⇒ **1 1 0 0 0 1 1 1**

São escolhidas, aleatoriamente, duas posições do cromossoma (por exemplo, a 2ª e a 5ª). Os conteúdos dos genes entre essas posições são trocados entre cromossomas, originando duas soluções "filhas". Para o exemplo:

Pais

Pai 1	⇒	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
Pai 2	⇒	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**

Filhos

Filho 1	⇒	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
Filho 2	⇒	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**

De seguida, cada um dos filhos é submetido a uma operação de mutação, com determinada probabilidade. O operador mutação mais simples consiste em escolher, aleatoriamente, um gene do cromossoma e trocar o caracter que lá se encontra por outro caracter do alfabeto. Para o exemplo considerado, e operando uma mutação na posição 6, obtemos o seguinte:

***String* original**

0 1 0 1 1 0 0 1

***String* mutada**

0 1 0 1 1 1 0 1

O passo final na geração da nova população é a substituição dos "maus" indivíduos, da população corrente, pelas novos (eventualmente mutados) elementos (descendentes da geração anterior).

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. Apresenta-se a seguir um exemplo simples de um algoritmo genético *[Oliveira J. 1999]*.

a) Inicialização:

(i) selecciona uma população inicial $X^{(1)} = \{x_1^{(1)}, \ldots, x_n^{(1)}\} \subseteq X$;

(ii) inicializa o melhor valor de f, f^* *e correspondente solução* x^*:

$f^* \leftarrow max\{f(x_i^{(1)}), i = 1, \ldots, N\}$

$x^* \leftarrow arg\ max\{f(x_i^{(1)}), i = 1, \ldots, N\}$

b) Passo n = 1, 2, ..., N ($X^{(n)}$ ***representa a população de soluções corrente):***

(i) Selecção dos "bons" indivíduos de $X^{(n)}$***:***

Sejam $\{y_j, j = 1, \ldots, 2M\}$ *2M indivíduos retirados aleatoriamente, com reposição, de* $X^{(n)}$*, com uma probabilidade de* $x_i^{(n)}$ *ser escolhido, que é uma função crescente de* $f(x_i(n))$.

(ii) Crossover:

Para $k = 1, \ldots, M$*, o operador de crossover é aplicado aos pares* (y_{2k}, y_{2k+1}) *com probabilidade . Geram-se assim, M pares de filhos* (z_{2k}, z_{2k+1}) *(que são idênticos aos seus pais, com probabilidade* $(1-\chi)$*).*

(iii) Mutação:

Para $k = 1, \ldots, 2M$*, o operador mutação é aplicado a* z_j*, com probabilidade* μ*. Isto origina 2M filhos* w_j*, possivelmente mutantes, com* $j = 1, \ldots, 2M$ *(e que são idênticos a* z_j*, com probabilidade* $(1-\mu)$*).*

(iv) Substituição dos "maus" indivíduos:

Retiram-se 2M indivíduos aleatoriamente, sem reposição, de $X^{(n)}$*, em que a probabilidade de* $x_i^{(n)}$ *ser escolhido é uma função decrescente de* $f(x_i^{(n)})$*.*

$X^{(n+1)}$ *é obtido por substituição destes 2M indivíduos "maus" pelos descendentes* $\{w_j, j = 1, \ldots, 2M\}$.

Para todo $j = 1, \ldots; N$*, se* $f(x_j^{(n+1)}) > f^*$*, então:*

$F^* \leftarrow f(x_j^{(n+1)})$

$x^* \leftarrow x_j^{(n+1)}$

c) Fim:

Se $(n+1) \geq$ *número pré-determinado de iterações, então termina.*

Algoritmo 3: Exemplo simples de um *Algoritmo Genético*

Segue-se um exemplo de aplicação do algoritmo, em que se pretende maximizar $f(x) = x^2$, sobre o conjunto dos inteiros {0, 1, ..., 31}. Codificação das soluções: *strings* de bits de comprimento 5. População inicial (geração aleatória):

j	Indivíduos	Solução	f(x)	j	Indivíduos	Solução	f(x)
1	01101	13	169	3	01000	8	64
2	11000	24	576	4	10011	19	361

Tabela 3: Exemplo simples de aplicação de um *Algoritmo Genético*

- ***Geração da nova população:*** *em cada iteração, todos os indivíduos são substituídos (2M = N = 4);*
- ***Probabilidade de crossover*** (χ)*: 1;*

- ***Operador de crossover:*** *string partida num ponto e todos os caracteres situados após esse ponto são trocados;*

- ***Probabilidade de mutação*** (μ)*:* 10^{-3};

Efectuando a primeira iteração (note-se que $x_2^{(1)}$ é usado duas vezes):

j	**Indivíduos** $\{y_j, j = 1, ..., 2M\}$	**Ponto de** *crossover*	**Filhos** $z_j = w_j$	**Solução**	$f(z_i)$
1	0110|1	4	01100	12	144
2	1100|0	4	11001	25	625
3	11|000	2	11011	27	729
4	10|011	2	10000	16	256

Tabela 4: Resultados da primeira iteração do exemplo de aplicação de um *Algoritmo Genético*

Analisando a Tabela 4 apercebemo-nos que, após a primeira iteração, já temos uma solução muito boa (27), não havendo necessidade de continuar as iterações.

3.5.4.3 Aplicação

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 *[Schaerf 1995]*, referente ao trabalho desenvolvido por *[Colorni et al. 1992]*, no qual, horários inviáveis são considerados no espaço de pesquisa deste tipo de algoritmos, em que a função objectivo engloba o número de inviabilidades. Por forma a desviar a pesquisa no sentido da obtenção de horários viáveis, é atribuído um grande peso às inviabilidades consideradas na referida função objectivo. Uma solução é representada através de uma matriz $M_{m\times p}$, 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 de 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 que move a solução para o seu óptimo local.

3.5.4.4 Conclusão

Após análise do algoritmo, é possível apercebermo-nos da importância dos parâmetros do mesmo, como sejam:

- *O tamanho da população;*
- *A taxa de substituição de soluções (dos pais pelos filhos);*
- *A probabilidade de crossover;*
- *E a probabilidade de mutação.*

Existem também escolhas estruturais que é necessário efectuar, nomeadamente:

- *Codificação das soluções;*
- *Operador de crossover;*
- *Operador de mutação.*

Existem alguns cuidados importantes que é necessário, 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, leva a que cada aproximação, utilizando algoritmos genéticos tenda a ser radicalmente diferente de outra.

3.5.5 PROGRAMAÇÃO EM LÓGICA

A *Programação em Lógica* ganhou notoriedade dentro da comunidade científica desde meados dos anos setenta, altura em que surgiu o *PROLOG* (***Prog**ramation et **Log**ique*), originário das ideias de *[Colmerauer 1975]*, em Marselha e *[Kowalski 1979]*, em Edimburgo. Esta linguagem depressa se tornou naquela que melhor representava uma nova classe de linguagens. 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.

Segundo alguns autores, *[Kowalski 1979]* e *[Gomes 1997]*, um programa pode ser subdividido em dois componentes, que podemos designar por competência e desempenho, ou lógica e controlo. 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. Um 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 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 *backtacking* 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 que 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 a simples mas ineficiente regra de computação de busca, usando inicialmente, a técnica de pesquisa em profundidade. Este mecanismo de controlo é basicamente um método de geração e teste, cuja performance decresce largamente em problemas de grande dimensão.

3.5.6 PROGRAMAÇÃO EM LÓGICA COM RESTRIÇÕES

A *Programação em Lógica com Restrições (PLR)* (*Constraint Logic Programming – CLP*), surgiu numa tentativa de ultrapassar as limitações de linguagens de programação lógica e particularmente do Prolog. Neste sentido, é possível ver a *PLR* como uma extensão de um sistema de programação lógica, no qual se introduzem estruturas de dados mais ricas e complexas, as quais permitem que objectos, como expressões aritméticas, sejam codificados e manipulados directamente. A 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]*, *[Gomes 1997]*.

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 1996b]*.

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.

Como forma de ilustrar as vantagens da aplicação de restrições são, normalmente, utilizados criptogramas de transformação de palavras em números, como ***SEND + MORE = MONEY*** ou ***DONALD + GERALD = ROBERT***. Apresenta-se, de seguida, um esquema representativo deste último criptograma (um problema simples de Satisfação de Restrições).

Os criptogramas de transformação de palavras em números, como o que se segue, podem ser expressos como PSR. A cada letra da soma seguinte corresponde um número diferente, entre 0 e 9. O objectivo é encontrar o seu valor.

$$\begin{array}{cccccccc} & C_5 & C_4 & C_3 & C_2 & C_1 & & \\ & D & O & N & A & L & D & \\ + & G & E & R & A & L & D & \\ \hline = & \boldsymbol{R} & \boldsymbol{O} & \boldsymbol{B} & \boldsymbol{E} & \boldsymbol{R} & \boldsymbol{T} & \end{array}$$

As variáveis do problema são as letras: **D, O, N, A, L, G, E, R, B, T;** *e o seu domínio é o conjunto de números inteiros* **{0...9}**, *tal que* **D, G, R** *não podem ser* **0**. *A resolução do problema está sujeita às seguintes restrições:*

$D \neq O \neq N \neq A \neq L \neq G \neq E \neq R \neq B \neq T$

$D \neq 0, G \neq 0, R \neq 0$

$D + D = T + C1 * 10$

$L + L + C1 = R + C2 * 10$

$A + A + C2 = E + C3 * 10$

$N + R + C3 = B + C4 * 10$

$O + E + C4 = O + C5 * 10$

$D + G + C5 = R$

A resolução deste problema mostra que a solução é a seguinte:

$D = 5 / O = 2 / N = 6 / A = 4 / L = 8 / G = 1 / E = 9 / R = 7 / B = 3 / T = 0$

Exemplo 1 - Exemplo de um problema simples de satisfação de restrições

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 1987a]* 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 1994b]*.

[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 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. Ou seja, problemas que exigem demasiado às técnicas de programação convencionais.

3.5.7 SISTEMAS PERICIAIS

Os problemas de escalonamento são de resolução complexa. Muitas organizações resolvem os seus problemas específicos, utilizando peritos humanos, que obtiveram muita 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.

Mesmo quando é tentada uma formulação analítica do problema (como, por exemplo, o mapeamento da calendarização de exames para um problema de *Coloração de Grafos*), muitas restrições intricadas não são formalizadas e somente restrições mais simples são usadas. Nos casos

em que uma solução não é encontrada, os especialistas utilizam muitas vezes o juízo humano, na relaxação de algumas restrições, situação que não é possível numa formalização analítica.

> *"Capacidades dedutivas, alteração interactiva dos conhecimentos, explicabilidade, linguagem amigável de interacção, são algumas das características fundamentais dos Sistemas Periciais" [Oliveira 1984].*

[Gudes et al. 1990] afirmam que a principal ênfase a dar num sistema pericial é à generalidade. Nos seus trabalhos é permitida uma forma muito geral de restrições, e as suas estratégias de controle 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.

3.5.8 SISTEMAS INTERACTIVOS

Nos *Sistemas Interactivos* de *Construção de Horários* existe interactividade entre o sistema e um utilizador, que terá de tomar diversas decisões por forma a construir os horários pretendidos. 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 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 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 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:

- ***O 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**: Um modelo é uma representação abstracta que ilustra os componentes ou relações de um determinado fenómeno. Diversos modelos são utilizados num Sistema de Apoio à Decisão;*
- ***O Sistema de Gestão da Interface Gráfica com o Utilizador**: A interface com o utilizador de um Sistema de Apoio à Decisão proporciona a facilidade de interacção entre o seu utilizador, a sua base de dados e a sua base de modelos.*

Pode-se observar na Figura 12, uma representação do relacionamento entre os 3 componentes de um *Sistema de Apoio à Decisão*.

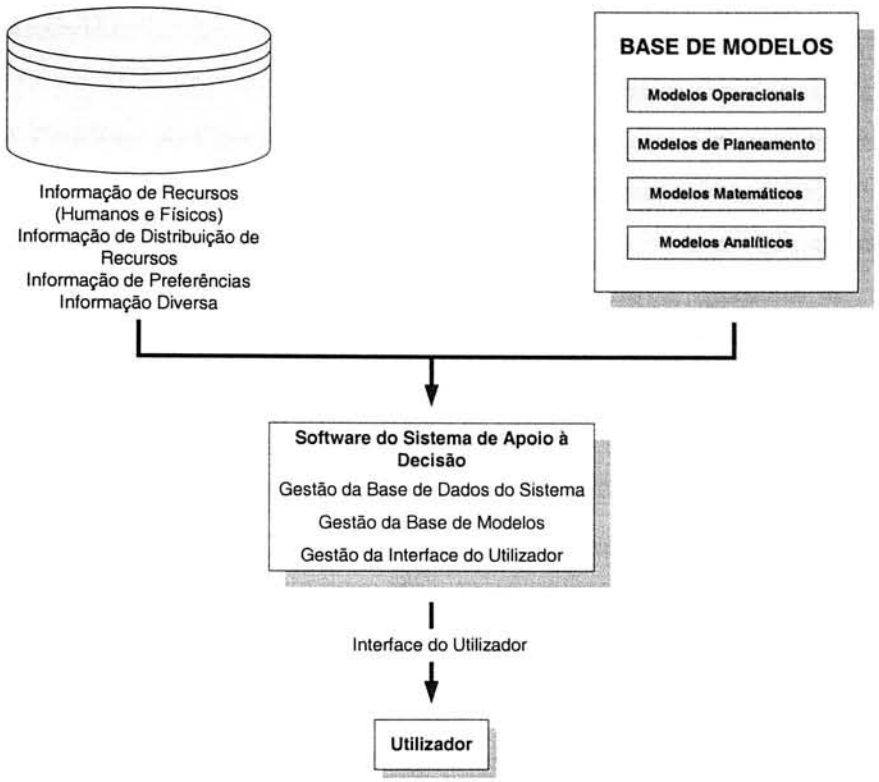


Figura 12: Esquema Geral de um Sistema de Apoio à Decisão

Os *Sistemas de Apoio à Decisão* proporcionam apoio a decisões interdependentes ou sequenciais, acompanhando, se necessário, todo o processo de tomada de decisão.

> *"Os Sistemas de Apoio à Decisão são capazes de testar diferentes estratégias, rápida e objectivamente (análise "what-if"), introduzindo melhorias significativas nos resultados da organização e no controle da gestão" [Dias 1999].*

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* (necessários num processo de construção de horários), 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.

3.6 CONCLUSÃO

Neste capítulo foi efectuada uma abordagem a diversas técnicas que podem ser utilizadas na resolução do *Problema da Construção de Horários em Universidades*. Foram descritas algumas técnicas que evoluíram ao longo do tempo, como sejam: a *Coloração de Grafos*, a *Programação Inteira* e as meta-heurísticas *Algoritmos Genéticos*, *Arrefecimento Simulado* e *Pesquisa Tabu*. Para estas três últimas técnicas foram ainda apresentados os respectivos algoritmos de resolução do problema e exemplos de aplicação das técnicas. De seguida foi efectuada uma introdução à *Programação em Lógica* e à *Programação em Lógica com Restrições*, esta última com um número crescente de investigadores que a aplicam na resolução dos seus problemas de escalonamento. Finalmente, foi efectuada uma abordagem aos *Sistemas Periciais* e aos *Sistemas Interactivos*.

Após análise de algumas das diversas técnicas possíveis de aplicar na resolução do *Problema de Construção de Horários*, segue-se um capítulo onde é descrita uma possível arquitectura de um sistema automático de construção de horários. É apresentada uma linguagem inovadora, que permite a representação de problemas, sendo de seguida, definida uma base de dados destinada ao armazenamento da informação referente ao problema. Depois é referida a adaptação e utilização de um sistema de "*parsing*", dos dados representados na linguagem, para a referida base de dados.

Capítulo 4

Arquitectura Base do Sistema de Dados

4.1 INTRODUÇÃO

Após análise do estado da arte relativo à problemática da construção de horários em universidades, bem como diversas técnicas com aplicação no referido problema, definiu-se um projecto, composto por diversas fases representativas do processo de desenvolvimento de uma aplicação para construção de horários. Na aplicação desenvolvida é importante a utilização de uma técnica de construção automática de horários, para a obtenção de resultados, passíveis de análise.

As diversas fases que compõem o projecto desenvolvido são as seguintes:

1ª. *Utilização de uma linguagem inovadora (UNILANG) [Reis e Oliveira 2000a], que permite a especificação completa de problemas de construção de horários (nomeadamente, a construção de horários, a calendarização de exames e a distribuição de serviço docente);*

2ª. *Definição de uma base de dados completa, que permita a estruturação de todos os dados relativos ao problema, para uma maior facilidade de manutenção dos mesmos;*

3ª. *Adaptação e utilização de um sistema de "parsing" (leitura e conversão para a estrutura de dados definida) dos dados representados na linguagem acima referida. Para uma maior facilidade e rapidez de manutenção dos dados, estes serão colocados em memória, utilizando determinadas estruturas pré-definidas;*

4ª. *Construção de uma interface gráfica, prática e funcional, que permita a devida apresentação dos dados e cuja interacção com o utilizador seja fácil e intuitiva;*

5ª. *Desenvolvimento de funcionalidade que permita a construção automática de horários, utilizando uma meta-heurística aplicada ao problema;*

6ª. *Efectuar um caso de estudo, em diversos problemas de geração de horários, utilizando as potencialidades previamente desenvolvidas;*

7ª. *Análise dos resultados obtidos.*

A arquitectura proposta do projecto para a construção automática de horários, permite obter uma noção mais específica das etapas pelas quais passa a representação e resolução do problema.

Neste capítulo é descrita a arquitectura base do sistema de dados, referente às três primeiras fases do projecto, atrás referido. No capítulo seguinte, são apresentadas as restantes fases, pelas quais passa o desenvolvimento do projecto definido, com o objectivo de obtenção de horários de forma automática.

4.2 STANDARDS E LINGUAGENS DE REPRESENTAÇÃO DE PROBLEMAS DE GERAÇÃO DE HORÁRIOS

4.2.1 INTRODUÇÃO

O crescimento abismal que se vem notando ao longo dos tempos na área da construção de horários torna-se em si próprio um problema. Devido à variedade de formatos de dados, actualmente em uso, e a existência de diversos problemas de geração de horários que tornam a comparação de resultados de pesquisa e a troca de ideias de investigação bem como dados relativos a problemas reais extremamente difícil. 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 uma linguagem universal aceite para descrever estes problemas.

4.2.2 *STANDARDS* DE DADOS PARA A CONSTRUÇÃO DE HORÁRIOS

Actualmente, não existe nenhuma linguagem universalmente aceite para a descrição de problemas de construção de horários. Foram realizadas diversas tentativas no sentido de tentar encontrar um formato de informação *standard*, tendo sido desenvolvidos alguns que, no entanto, são habitualmente incompletos em algum aspecto.

4.2.2.1 A Linguagem TTL de Cooper e Kingston

[Cooper e Kingston 1993a], *[Cooper e Kingston 1993b]*, *[Cooper e Kingston 1993c]* 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 pesquisa e alguns encontros. Um grupo de tempo define os nomes dos tempos disponíveis para encontros, 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 pesquisa podem conter subgrupos, que são subconjuntos do conjunto de pesquisa que define funções que podem ser executadas. Um recurso pode encontrar-se em qualquer número de subgrupos. Nesta linguagem de especificação, reuniões 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.

4.2.2.2 A Linguagem Proposta por 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 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 e 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). 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. Algumas omissões importantes desse trabalho são 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.

4.2.2.3 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 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 da secção) 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.

4.2.2.4 A Linguagem Proposta por Burke, Kingston e Pepper

Um artigo mais recente escrito por *[Burke et al. 1998]*, propõe um tipo de *standard* diferente para instâncias de horários. Eles incluem uma simples, mas 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*. Adicionalmente, prevêem o desenvolvimento de uma biblioteca de funções *standard* que permitirá uma maior facilidade de tratamento das restrições.

4.2.2.5 A Linguagem STTL de Kingston

[Kingston 1999a], *[Kingston 1999b]* 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, como os meios naturais de definição de entidades como sejam tempos, recursos e encontros, enquanto funções permitem que requisitos arbitrários sejam claramente expressos. Um problema é considerado uma questão geral, à qual é necessário responder como seja o *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 controle que permitem a definição de expressões, mais ou menos complexas, dos parâmetros do problema. Deste modo, é necessária alguma aprendizagem da estrutura léxica, estrutural e funcional da linguagem, por forma a ser possível a definição de um problema.

4.3 A LINGUAGEM UNILANG

Neste capítulo será apresentada uma linguagem (*UNILANG*) para representação de problemas de construção de horários, que pode facilmente ser lida e entendida por informáticos e administradores de escolas. A linguagem *UNILANG*, proposta por *[Reis e Oliveira 2000a]*, é baseada na definição de oito sub-problemas do problema completo de construção de horários. Esta linguagem foi utilizada extensivamente neste projecto. Como tal, será apresentada a sua descrição completa, baseada essencialmente no artigo de *[Reis e Oliveira 2000a]*.

4.3.1 SUB-PROBLEMAS DO PROBLEMA 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 por 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 construção de horários.

É possível visualizar na Figura 13 uma representação genérica de um problema de construção de horários.

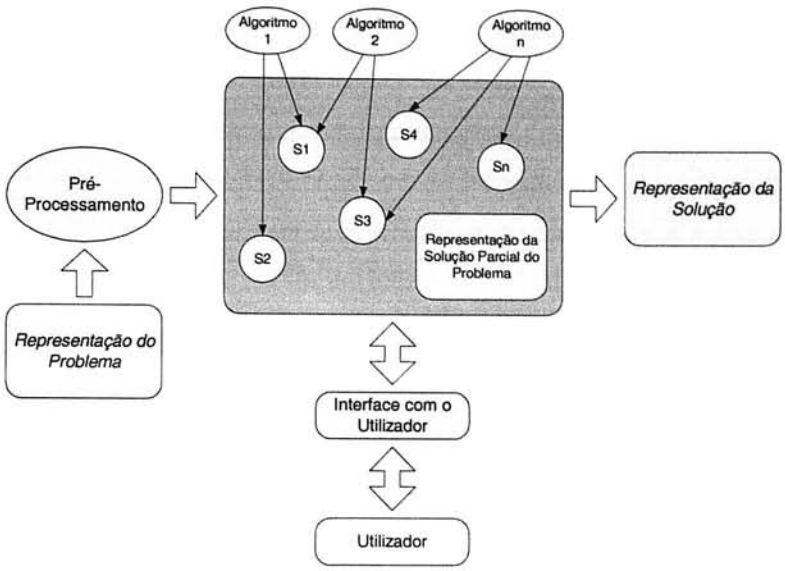


Figura 13: Representação genérica de um problema de construção de horários ***[Reis e Oliveira 2000a]***

Estendendo a definição de *[Carter e Laporte 1996]*, dos subproblemas de horários, *[Reis e Oliveira 2000a]* propuseram a definição das oito classes seguintes de subproblemas de horários inter-relacionados:

- ***HTD – Horário Turma-Docente:*** *Isto é um problema comum na maior parte das escolas e instituições portuguesas de ensino superior e consiste em escalonar um tempo (ou conjunto de tempos) a cada aula de cada módulo na escola (ou universidade). A unidade de escalonamento é um grupo de alunos (ou turma) que tem um programa comum. É considerado que a atribuição de docentes e turmas aos eventos já foi efectuada previamente. Normalmente, as salas são usadas como restrições neste problema ou a alocação de salas é executada simultaneamente com o horário de turma-docente;*
- ***HD – Horário Disciplina:*** *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 do escalonamento do Horário-Disciplina. Encontram-se normalmente, atribuídos aos eventos, mas em algumas universidades isto pode também ser falso;*
- ***HE – Horário Exame:*** *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 Horário-Exame;

- ***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) existentes. Um problema similar ocorre no construção do Horário-Exame se considerarmos a separação de exames (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. Uma vez os alunos terem escolhido os seus 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 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 subproblemas 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 subproblemas irão ser resolvidos e a ordem pela qual isso será efectuado.

4.3.2 REQUISITOS DA LINGUAGEM *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 deveriam ser traduzidos facilmente para este formato;*
- *O formato deveria ser facilmente lido pela máquina (e qualquer sistema de horários) bem como pelo utilizador humano;*
- *A informação não directamente relacionada com o problema de escalonamento deveria não 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 deste 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 ambas as restrições simples e complexas;*
- *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 o mais reduzida possível;*
- *O formato deverá permitir a representação de problemas, bem, como soluções completas e incompletas;*
- *Deverá ser robusta permitindo a validação de dados simples e a eliminação de erros comuns.*

4.3.3 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 um conjunto de períodos de tempo localizados em determinado dia (nessa semana) e determinado 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 carga de trabalho diária. Os docentes podem ter um número mínimo e máximo de restrições de atribuição de eventos e restrições de capacidade (perceber 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). Assim, são considerados componentes: Períodos, Períodos de Tempo, Recursos (Salas, Docentes e Alunos) e Eventos.

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 período de tempo (ou conjunto de períodos de tempo). Não são considerados outros recursos como equipamento porque estes não são particularmente importantes em problemas de construção de horários.

Este modelo permite a especificação de dados e restrições num formato comum *[Reis e Oliveira 2000a]*. Como, por vezes, a distinção entre dados e restrições não é completamente clara, este parece ser um procedimento correcto. 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, para cada um dos eventos considerados, um conjunto de *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.

Por forma a que o formato fosse o mais robusto possível, seguindo a ideia de *[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, apresentando-se na Tabela 5 um exemplo.

Default	All	Every	Any	Defeito	Todos	Qualquer
Event	Module	Lecture	Lesson	Exam	Examination	Tutorial
Teacher	Teachers	Invigilator	Supervisor	Lecturer	Docente	Vigilante
Student	Students	Aluno	Alunos			
Day	Days	Dia	Dias			
Time	Times	Hour	Hours	Minute	Minutes	Hora
Slot	Slots	Time Slot	Per. Tempo			
Place	Places	Room	Rooms	Sala	Salas	
Preference	Weight	Priority	Penalty	Preferência	Peso	Prioridade
Teaches	Invigilates	Supervises	Lecciona	Vigia		
Hold	Holds	Have capacity	Has capacity	Capacity	Capacidade	
Duration	Length	Duração	Tamanho			
Contains	Comprises	Has	Have	Contém	Tem	
Is	Are	É	São			
At Least	No less than	No fewer than	Pelo menos	Não menos do que		
At most	No more than	No máximo				
Exactly	Precisely	Exactamente				
Double_Bookings	Clashes	Conflicts	Conflitos			

Tabela 5: Uma tabela exemplo de sinónimos

Isto permite a definição de ficheiros de entrada com uma leitura mais fácil para os utilizadores. Desta forma é 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.

4.3.4 REPRESENTAÇÃO DO TEMPO

Cada ficheiro contém um problema relacionado com determinado período de tempo: Um ano escolar. 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 *UNILANG* permite a definição do número de períodos usados (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 que tem o mesmo 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 não definição 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 para resolver, as capacidades dos *slots* não são necessárias definir explicitamente.

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.

4.3.5 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, isto tornam-se em restrições 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

4.3.6 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>]

4.3.7 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 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 *time 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

4.3.8 DOCENTES E VIGILANTES

Neste modelo, os docentes podem ser vistos como tal bem 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. Então 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

4.3.9 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 numa linha (consecutiva) ou num dado dia. Normalmente, isto aplica-se à maioria das restrições exactamente (*exactly*) e pelos menos (*at least*) algumas restrições 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

events {<E>} [preference <P>]

Outra restrição típica estabelece que um dado número de eventos ocorre ou não pode ocorrer simultaneamente no mesmo dia.

events {<E>} are [not] simultaneously | on_the_same_day [preference <P>]

4.3.10 EXTENSÕES À *UNILANG*

4.3.10.1 Representação da Organização da Universidade

Após o estudo da linguagem *UNILANG*, e da compreensão da sua simplicidade de representação de problemas e utilização da mesma, sentiu-se a necessidade de criar extensões à linguagem, por forma a adaptá-la ao tipo de problemas a analisar neste trabalho. As extensões criadas referem-se, fundamentalmente, à representação da estrutura de uma universidade, composta pelos seguintes itens:

- *Departamentos*
- *Áreas Científicas*
- *Cursos*

Partindo do pressuposto que a estrutura base de uma universidade assenta em departamentos, torna-se necessário representá-la. Como tal, utilizando uma notação o mais parecida possível com a da linguagem *UNILANG*, ficaria:

departments are {<DEP>}

Na instituição existe um conjunto diverso de áreas científicas e a sua definição será a seguinte:

scientific_areas are {<SA>}

Outro dos elementos em que assenta a estrutura da universidade são os cursos, cuja representação pode ser efectuada de forma análoga aos anteriores:

degrees are {<DEG>}

Cada um dos departamentos possui um conjunto de cursos e um conjunto de áreas científicas:

department {<DEP>} has degrees {<DEG>} | default [preference <P>]

department {<DEP>} has scientific_areas {<SA>} | default [preference <P>]

Cada um dos cursos tem associado um conjunto de eventos que deverão ser representados, utilizando a seguinte notação:

degree {<DEG>} has events {<E>} | default [preference <P>]

Cada um dos eventos está também associado (de forma rígida ou flexível) a uma ou várias áreas científicas:

Scientific_area {<SA>} has events {<E>} | default [preference <P>]

Cada um dos estudantes e/ou grupos de estudantes (turmas) pertencem a um dado curso:

degree {<DEG>} has student | group_student {<ST> | <GST>} | default [preference <P>]

Finalmente, cada um dos docentes ou grupos de docentes pertencem a uma dada área científica:

scientific_area {<SA>} has teacher | group_teachers {<TE> | <GTE>} | default [preference <P>]

4.3.10.2 Standards e Designações

É possível definir um problema de *Construção de Horários* a partir da notação até agora apresentada, mas seria interessante que a aplicação desenvolvida tivesse um conjunto de informação *standard*, por defeito, que o utilizador não teria necessidade de definir no seu problema, devido à mesma suprir as suas necessidades básicas de representação do problema. Por exemplo, ao nível dos dias, tempos e períodos de leccionação. A seguir apresenta-se um exemplo de alguma notação *standard* por defeito, que poderia existir numa aplicação.

```
days are Mon Tue Wed Thu Fri Sat
times are t1 t2 t3 t4 t5 t6 t7 t8 t9
slots are s1 s2 s3 s4 s5 s6 s7 s8 s9
slots are s10 s11 s12 s13 s14 s15 s16 s17 s18
slots are s19 s20 s21 s22 s23 s24 s25 s26 s27
slots are s28 s29 s30 s31 s32 s33 s34 s35 s36
slots are s37 s38 s39 s40 s41 s42 s43 s44 s45
slots are s46 s47 s48 s49 s50 s51 s52 s53 s54
```

Exemplo 2: Exemplo de notação *standard*, por defeito, a utilizar numa aplicação

Quanto às restrições rígidas (*hard constraints*) fica definido que nos níveis de preferência, o valor 0 (zero) indica não existir qualquer restrição associada. Qualquer outro valor significa o peso atribuído pelo desrespeito da restrição.

Outra questão referente à simplicidade da *UNILANG*, prende-se com a inexistência de notação representativa de designações de determinados componentes, e que são importantes para uma maior facilidade de compreensão e tratamento (gráfico) dos dados do problema, por parte do utilizador, aquando da utilização da aplicação. Ou seja, por exemplo, na *UNILANG* é possível definir as abreviaturas dos cursos, mas não as designações dos mesmos, acontecendo o mesmo com as salas, os docentes, as turmas, entre outros. Uma das formas possíveis de ultrapassar esta questão seria representar as designações e outra informação associada a cada componente, num outro ficheiro textual. Para cada componente (identificado pela abreviatura respectiva) encontrar-se-ia a informação a si associada, que poderá ser de diverso tipo. A seguir apresenta-se um exemplo simples de associações que podem ser definidas.

```
day Mon Monday
day Tue Tuesday
day Wed Wednesday
day Thu Thursday
day Fri Friday
day Sat Saturday
time t1 09:00
time t2 10:00
time t3 11:00
time t4 12:00
time t5 13:00
time t6 14:00
time t7 15:00
time t8 16:00
time t9 17:00
```

Exemplo 3: Exemplo de notação a utilizar para representação de dados na aplicação

As extensões apresentadas foram definidas por forma a responder a algumas necessidades de um *Problema de Construção de Horários*, mas como é evidente, dependendo do tipo de problema, outras extensões poderão ser definidas.

4.3.11 FUNÇÃO DE AVALIAÇÃO E SOLUÇÃO FINAL

A função de avaliação desta linguagem encontra-se descrita implicitamente nos dados do problema e respectivas restrições. As restrições rígidas (em que não é incluída a palavra chave *preference*) têm de ser respeitadas. As restrições flexíveis (nas quais a palavra chave *preference* aparece)

devem ser respeitadas para obter uma solução de boa qualidade. Para cada uma das restrições flexíveis violada, o valor de preferência (<P>) é adicionado à penalidade total da solução. Assim, esta representação implica que tenhamos um valor numérico (qualidade) associado a uma dada solução. Quanto menor este valor for, melhor será a solução final.

A representação da solução final de qualquer problema de construção de horários é composta por: para cada parte de um evento, um conjunto de time slots (quando as partes do evento têm lugar), um conjunto de salas (onde elas são dadas), um conjunto de docentes (ou grupos de docentes) que irão leccionar (supervisar ou participar) o evento e um conjunto de alunos (grupos de alunos) que irão assistir ao evento:

event_part <EP> | event_section <ES> | event <E> is in slots {<S>}

event_part <EP> | event_section <ES> | event <E> is in rooms {<R>}

event <E> | event_section <ES> is taught by {<TE | GTE>}

event <E> | event_section <ES> has students {<ST | GST>}

Para cada um dos eventos, ofertas de eventos (se existirem) ou partes de eventos (se especificadas), um conjunto de time slots e um conjunto de salas pode ser especificado. A solução fica completa com um conjunto de docentes (ou grupos de docentes) e um conjunto de alunos (ou grupos de alunos) para cada um dos eventos (ou ofertas de eventos se especificadas). É assumido que cada oferta de um evento tem os mesmos docentes e alunos para cada uma das suas partes.

4.3.12 CONCLUSÕES E EVOLUÇÃO DA LINGUAGEM

O desenvolvimento desta linguagem é inovador, no sentido de que após estudo de diversa bibliografia associada à questão da definição de problemas de construção de horários, esta parece ser a primeira tentativa de usar a identificação de subclasses do problema, para obter um *standard* dos dados e restrições que definem qualquer problema de construção de horários. É evidente que ficam em aberto diversos pormenores nesta definição *standard* e faltam algumas coisas na linguagem que permitam a representação de um dado problema específico mas em geral, a grande maioria dos problemas de geração de horários (e associados), pode ser representado utilizando esta linguagem. É o caso dos problemas de construção de horários em universidades analisados neste projecto. A representação destes problemas na linguagem *UNILANG* não constituiu problema de maior e permitiu que a sua especificação fosse efectuada de forma completa.

4.4 SISTEMA DE LEITURA E CONVERSÃO DOS DADOS

O sistema de leitura e conversão dos dados desenvolvido consiste no seguinte: um problema definido, utilizando a linguagem de especificação de problemas de construção de horários (*UNILANG*) é colocado num ficheiro de texto, formato ASCII. De seguida, é lido, linha a linha e a partir da consulta do dicionário de termos da linguagem, cada uma das palavras chave da linguagem é identificada. A sintaxe de cada uma das frases é então identificada e o seu conteúdo vai ser colocado então nas estruturas de dados internas do programa correspondentes.

No sistema desenvolvido está garantido o controle de alguns erros que possam ocorrer aquando da construção do ficheiro de dados. Por exemplo, a variação de uma letra numa palavra chave, é considerada válida desde que só exista um palavra chave nestas condições. Separação de palavras chave, coladas é também utilizada. No caso de ser encontrada uma frase (linha do ficheiro *UNILANG*) que não é valida, o utilizador é informado e a sua intervenção é solicitada.

4.5 BASE DE DADOS PROJECTADA

4.5.1 INTRODUÇÃO

Um processo de construção de horários passa pela gestão de diversa informação associada. Como tal, é necessário definir uma base de dados suficientemente abrangente que permita:

- *Suportar toda a informação envolvida no processo;*
- *Gerir de forma rápida e eficiente essa mesma informação.*

Existem no mercado diversos *Sistemas de Gestão de Bases de Dados (SGBD)* que poderiam ser utilizados num problema de construção de horários, dadas as suas elevadas capacidades ao nível de estruturação e armazenamento de informação. Mas, a questão é que estes *SGBD* encontram-se vocacionados para a gestão de dados, e como tal, não se adequam ao problema de construção de horários, no qual, apesar da importância da estruturação dos dados, o mais importante é a rapidez de acesso para leitura e escrita, dos mesmos. Sendo assim, o ideal seria que os dados se encontrassem permanentemente disponíveis para utilização. Dessa forma, e pela quantidade de dados que um problema de construção de horários contém, o suporte mais adequado para a sua manutenção é a memória RAM do computador.

Estabelecido, à partida, que será utilizada a memória como suporte aos dados envolvidos num problema de construção de horários, procedeu-se à definição das estruturas que os irão suportar. Foi definido um conjunto de registos (associação de campos de dados de diversos tipos), cada qual

com a informação respeitante a cada componente. De seguida foram definidas matrizes, (tabelas de dados) que suportarão os dados envolvidos no *Problema de Construção de Horários* a tratar.

Dessa forma, apresenta-se a seguir, os registos definidos, para cada um dos componentes de um problema.

4.5.2 ESTRUTURA GERAL DA INSTITUIÇÃO

Normalmente, uma instituição de ensino universitário encontra-se estruturada em Departamentos (*Departments*), Áreas Disciplinares (*Scientific Areas*) e Cursos (*Degrees*). As Áreas Disciplinares e os Cursos encontram-se, por vezes, afectos a Departamentos. Para o armazenamento da informação associada foram criados 4 matrizes de registos: *tt_department*, *tt_scientific_area*, *tt_degree*, *tt_degree_event*. Nas 3 primeiras matrizes encontra-se definida a informação geral associada aos Departamentos, Áreas Científicas e Cursos, nomeadamente a identificação, a designação e o director. Na 4ª matriz são definidos os eventos associados a cada um dos cursos.

Campos do Registo	Descrição
id	Identificação do Departamento (Ex: DCBC).
designation	Designação do Departamento (Ex: Departamento de Ciências Básicas e Computacionais).
director	Identificação do Coordenador do Departamento, através do código de docente.

Tabela 6: Representação de Departamentos - tt_department (N, 3)

Campos do Registo	Descrição
id	Identificação da Área Científica (Ex: Fis).
designation	Designação da Área Científica (Ex: Física).
director	Identificação do Responsável pela Área Científica, através do código de docente (Ex: LPR).
department	Identificação do Departamento (Ex: DCBC).

Tabela 7: Representação de Áreas Científicas - tt_scientific_area (N, 4)

Campos do Registo	Descrição
id	Identificação do Curso (Ex: EEC).
designation	Designação do Curso (Ex: Engenharia Electrotécnica e de Computadores).
director	Identificação do Coordenador do Curso, através do código de docente.
department	Identificação do Departamento (Ex: DCBC).

Tabela 8: Representação de Cursos - tt_degree (N, 4)

Campos do Registo	Descrição
id	Identificação do Evento associado ao Curso (Ex: IA-EI).
designation	Designação do Evento associado ao Curso.
degree	Identificação do Curso.
year	Identificação do Ano.
period	Identificação do Período.
event	Identificação do Evento.

Tabela 9: Representação de Planos Curriculares - tt_degree_event (N, 6)

4.5.3 REPRESENTAÇÃO DO TEMPO

A representação do tempo será realizada utilizando 5 matrizes de registos, por forma a armazenar a informação respeitante aos dias, horas, *slots*, períodos e períodos de tempo: *tt_day*, *tt_time*, *tt_slot*, *tt_period*, *tt_time_period*.

Campos do Registo	Descrição
id	Identificação do dia (Ex: 3).
designation	Designação do dia (Ex: quarta).

Tabela 10: Representação dos Dias - tt_day (N, 2)

Campos do Registo	Descrição
id	Identificação da hora (Ex: 16).
designation	Designação da hora (Ex: 15:30-16:00).

Tabela 11: Representação das Horas - tt_time (N, 2)

Campos do Registo	Descrição
id	Identificação do slot (ex: S32, S33, ...).
designation	Designação do slot.
day	Identificação do dia (ex: 2).
time	Identificação do tempo (ex: 17).
usable	Se o slot está ou não disponível para alocação (ex: 'S' ou 'N').
preference	Nível de preferência.
seat_capacity	Número de lugares possível.
event_capacity	Número de eventos possível.

Tabela 12: Representação dos Tempos ("Slots") - tt_slot (N, 8)

Campos do Registo	Descrição
id	Identificação do Período (Ex: An, S1, S2, T1, ...).
designation	Designação do Período (Ex: 1º Semestre).
num_weeks	Número de Semanas.
weeks	Semanas (Array).
start_date	Data Inicial.
start_day	Dia Inicial.
end_day	Dia Final.

Tabela 13: Representação dos Períodos - tt_period (N, 7)

Campos do Registo	Descrição
id	Identificação do Período de Tempo: (ex: Mseg - Manhã de Segunda, Tter – Tarde de Terça, Asex - Almoço de Sexta).
designation	Designação do Período de Tempo.
slots	Lista dos slots associados ao período de tempo (Array).

Tabela 14: Representação de Períodos de Tempo - tt_time_period (N, 3)

4.5.4 REPRESENTAÇÃO DO ESPAÇO

A representação do espaço físico utilizado (salas) será realizada utilizando 3 matrizes de registos: *tt_room, tt_room_type, tt_room_distance*. Na primeira matriz é definido um espaço para guardar o identificador da sala, a designação, o número máximo de alunos que comporta, o número máximo de eventos que é possível realizar na sala ao mesmo tempo (geralmente um), o nível de preferência (entre 0 e 3), os tipos de sala associados, a possibilidade de decorrerem dois eventos em simultâneo (*double bookings*), os slots e respectivas preferências associadas e as restrições de tempo. Na segunda matriz são definidos os tipos de salas existentes através de um identificador próprio e a designação respectiva. Na terceira matriz são armazenados os identificadores de duas salas próximas e a distância entre elas.

Campos do Registo	Descrição
id	Identificação da Sala (ex: S1.1).
designation	Designação da Sala.
seat_capacity	Número máximo de alunos.
event_capacity	Número máximo possível de eventos a realizar ao mesmo tempo.
preference	Nível de preferência.
room_types	Tipos de Sala (Array).
double_bookings	Double Bookings.
slot_pref	Preferências de Slots (Array).
daytime_constraints	Restrições de Dia e Tempo (Array).

Tabela 15: Representação das Salas - tt_room (N, 9)

Campos do Registo	Descrição
id	Identificação do tipo de sala (Ex: T).
designation	Designação do tipo de sala (Ex: Teórica).

Tabela 16: Representação de Tipos de Salas - tt_room_type (N, 2)

Campos do Registo	Descrição
id1	Identificação da 1ª sala (ex: A1.1).
id2	Identificação da 2ª sala (ex: S2.1).
slots	Distância entre as 2 salas.

Tabela 17: Representação de Distâncias entre Salas - tt_room_distance (N, 3)

4.5.5 DESCRIÇÃO DE EVENTOS

Existem diversos eventos a escalonar, envolvendo diversos recursos. O armazenamento temporário desses eventos será realizado fazendo uso de 2 matrizes de registos: *tt_event_description*, *tt_event_group*. Na primeira, serão armazenadas as características básicas de cada evento, nomeadamente: a identificação do evento, o número de alunos inscritos, o número de docentes necessário, o número de salas necessário para a concretização do evento, o tipo de sala que é necessário para que o evento se concretize, o nível de preferência desse tipo de sala, o número de secções (ofertas), o número de partes (aulas) em que se divide o evento e a Área Disciplinar associada ao evento. Na segunda matriz é armazenado o conjunto de eventos que compõem um grupo.

Campos do Registo	Descrição
id	Identificação do evento (ex: IA).
designation	Designação do evento (ex: Inteligência Artificial).
duration	Duração do evento.
periods	Períodos em que o evento ocorre (Array).
num_students	Número de alunos inscritos no evento.
students	Lista das identificações dos alunos inscritos (Array).
group_students	Lista das identificações dos grupos de alunos inscritos (Array).
num_teachers	Número de docentes escalonados para leccionar o evento.
teachers	Lista das identificações dos docentes inscritos (Array).
group_teachers	Lista das identificações dos grupos de docentes inscritos (Array).
num_rooms	Número de salas escalonadas para o evento.
rooms	Lista de identificações das salas escalonadas para o evento (Array).
preference	Grau de preferência do evento.
scientific_area	Lista das identificações das áreas científicas associadas (Array).
degree	Curso ao qual pertence o evento.
num_sections	Número de ofertas existentes.
sections	Lista das identificações das ofertas existentes (Array).
num_parts	Número de partes que compõem uma oferta do evento.
part_duration	Lista das durações das partes que compõem uma oferta do evento (Array).
slot_pref	Lista de preferências dos slots (Array).
room_type	Lista de preferências dos tipos de sala associados (Array).
event_constraints1	Restrições de Eventos 1.
event_constraints2	Restrições de Eventos 2.

Tabela 18: Representação dos Eventos - tt_event_description (N, 23)

Campos do Registo	Descrição
Id	Identificação do grupo de eventos.
designation	Designação do grupo.
events	Os eventos associados a cada grupo.

Tabela 19: Representação de Grupos de Eventos - tt_event_group (N, 3)

4.5.6 RECURSOS HUMANOS

Existem dois tipos de recursos humanos básicos num processo de construção de horários. São eles, os alunos e os docentes, que podem associar-se em grupos, de acordo com determinados interesses e pré-definições. Como tal, foi definido um tipo de dados (*t_resources*) que armazena diversas características comuns aos recursos, bem como outras de carácter específico de acordo com o tipo do recurso. De seguida, foram construídas quatro matrizes do tipo de dados definido e são elas: *tt_student*, *tt_student_group*, *tt_teacher* e *tt_teacher_group*.

Campos do Registo	Descrição
id	Identificação do recurso.
designation	Designação do recurso.
resource_type	Tipo de recurso (ex: 1 - Docente; 2 - Aluno, ...).
preference	Grau de preferência do recurso.
event_pref	Lista de preferências de eventos (Array).
slot_pref	Lista de preferências de slots (Array).
daytime_constraints	Lista de restrições de tipo *daytime_constraints* (Array).
spreading_constraints	Lista de restrições de tipo *spreading_constraints1* (Array).
spreading_constraints	Lista de restrições de tipo *spreading_constraints2* (Array).
double_bookings	Grau de preferência de *double_bookings*.
degree	Identificação do curso (aplica-se somente a alunos e turmas).
teacher_id	Número de alunos (aplica-se somente a turmas).
num_students	Identificação do docente (aplica-se somente a docentes).
students	Lista de identificações dos alunos (Array) (aplica-se somente a turmas).
scientific_area	Área científica associada (aplica-se somente a docentes e grupos de docentes).
teachers	Lista de identificações dos docentes (Array) (aplica-se somente a grupos de docentes).

Tabela 20: Representação de Recursos Humanos (Estudantes, Turmas, Docentes e Grupos de Docentes) - t_resources (N, 16)

4.5.7 SOLUÇÃO FINAL

A representação da solução final é realizada utilizando a matriz *tt_event_allocation*, em que se armazena a alocação dos diversos eventos.

Campos do Registo	Descrição
id	Identificação do evento.
id_section	Identificação da oferta.
teachers	Lista de identificações de docentes (Array).
group_teachers	Lista de identificações de grupos de docentes (Array).
students	Lista de identificações de alunos (Array).
group_students	Lista de identificações de grupos de alunos (Array).
event_parts_rooms	Lista de identificações de salas atribuídas a partes do evento (Array).
event_parts_slots	Lista de identificações de salas atribuídas a partes do evento (Array).
total_penalties	Total de penalidades (Array).

Tabela 21: Representação da Solução Final - tt_event_allocation (N, 9)

De uma forma geral, pode concluir-se que as estruturas acima definidas são suficientes para suportar todos os dados que compõem um problema geral de construção de horários, podendo, no entanto, estenderem-se as suas estruturas, de acordo com a especificidade do problema.

4.6 CONCLUSÃO

Neste capítulo foi apresentada a arquitectura base desenvolvida, do sistema de dados a utilizar neste projecto. Foram apresentados diversos *standards* e linguagens de representação de problemas de construção de horários. A seguir, foi introduzida e descrita a linguagem de representação de *Problemas Completos de Construção de Horários* (*UNILANG*), tendo sido apresentadas extensões à linguagem, a usar no projecto. De seguida, foi referida a utilização e adaptação do sistema de leitura e conversão de dados para a base de dados definida. Finalmente, foi apresentada a base de dados desenvolvida e incluída no projecto realizado.

No capítulo seguinte é descrita a implementação do sistema desenvolvido. É apresentada a linguagem de programação utilizada, a interface gráfica construída para apoiar o utilizador da aplicação na resolução de um problema de construção de horários. É descrito um algoritmo, baseado em *Arrefecimento Simulado*, possível de usar na resolução de problemas de construção automática de horários, tendo sido usado num estudo computacional realizado. O *Arrefecimento Simulado* foi escolhido devido à sua simplicidade e aos bons resultados encontrados na literatura da sua aplicação a problemas semelhantes. Finalmente, são apresentados resultados computacionais obtidos.

Capítulo 5

Implementação do Sistema

5.1 INTRODUÇÃO

Foi desenvolvido de raiz um pacote de *software*, com vista a integrar o sistema de dados, atrás apresentado, e apoiar o utilizador no estudo de problemas de construção de horários de forma automática, utilizando um algoritmo baseado em *Arrefecimento Simulado*.

O ambiente de desenvolvimento utilizado foi o *Microsoft Windows*, utilizando a linguagem de programação *Delphi v5.0*. A interface com o utilizador apresenta as características comuns de uma aplicação Windows, baseando-se em janelas sobreponíveis, menus, botões e caixas de diálogo.

A aplicação desenvolvida contém potencialidades de leitura de ficheiros de texto, com os diversos dados necessários à resolução do problema. Esses dados são então colocados em memória, em estruturas devidamente definidas, de forma a que a sua manutenção seja o mais simples possível.

O pacote de *software* desenvolvido contém, entre outros, diversos tipos de formulários que permitem a apresentação e manutenção dos dados lidos. Isso inclui também, formulários de manutenção de preferências de diversas entidades, nomeadamente docentes, salas e turmas.

Na aplicação construída é possível a resolução de qualquer *Problema de Construção de Horários*, através de uma meta-heurística implementada: o *Arrefecimento Simulado*. É possível, também, visualizar a resolução do problema para uma dada entidade, por exemplo o horário de um docente.

5.2 PLATAFORMA DE DESENVOLVIMENTO SELECCIONADA

5.2.1 SISTEMA OPERATIVO UTILIZADO

O projecto do sistema de construção de horários foi desenvolvido sobre o sistema operativo *Windows*, procurando utilizar ao máximo as potencialidades gráficas deste sistema. A escolha do ambiente *Windows* baseou-se essencialmente em 3 factores:

- *Disponibilidade de hardware adequado ao trabalho, compatível com o ambiente Windows;*

- *Disponibilidade de linguagem de programação neste sistema, adequada ao desenvolvimento de raiz de um sistema de construção de horários (Delphi v5.0);*
- *Potencialidades gráficas disponibilizadas (definição de menus, caixas de diálogo, botões, tratamento de janelas, etc.) para apresentação da informação.*

Acrescido a estes factores, foi ainda tomada em consideração a cada vez maior generalização e facilidade de utilização do ambiente *Windows*, o que permite que haja um maior número de potenciais utilizadores da aplicação desenvolvida.

5.2.2 LINGUAGEM DE PROGRAMAÇÃO

Para o desenvolvimento da aplicação de geração de horários foi seleccionada a linguagem *Delphi v5.0* da *Borland*. Esta escolha baseou-se na análise das linguagens mais poderosas, disponíveis[3] para o ambiente de desenvolvimento seleccionado (*Microsoft Windows*).

Nesta análise verificou-se que o *Delphi* apresentava como vantagens:

- *Programação orientada aos objectos;*
- *Utilização de todas as potencialidades oferecidas pelo Pascal;*
- *Facilidade de adaptação para esta linguagem de programação e ambiente, de diversos tipos de componentes, que permitem desenhar, de forma extremamente simples, recursos Windows, tais como menus, caixas de diálogo, icons ou grelhas;*
- *Enorme facilidade de obtenção de componentes de diversos tipos, nomeadamente grelhas, para esta linguagem de programação;*
- *Facilidade de tratamento de dados em memória, oferecida pela utilização de arrays dinâmicos;*
- *Enormes potencialidades do ambiente a nível de teste;*
- *Facilidades de gestão de um projecto de software, oferecidas pelo ambiente Delphi.*

De um modo geral, o *Delphi v5.0* mostrou-se uma linguagem muito fiável, prática e intuitiva, que permite um fácil desenvolvimento de aplicações gráficas, para o sistema operativo *Windows*.

[3] Em Março de 2000, data de início da implementação do sistema.

5.3 POTENCIALIDADES REQUERIDAS AO SISTEMA

No desenvolvimento do sistema procurou-se incluir um conjunto de funcionalidades específicas de um sistema de construção de horários, sendo basicamente as seguintes:

- *Possibilidade de definição, o mais simples possível, de um Problema de Construção de Horários (utilizando a linguagem UNILANG);*
- *Armazenamento dos dados de forma correcta, no sentido de facilitar a manutenção dos mesmos, por qualquer pessoa responsável pelo processo de construção de horários, numa instituição de ensino;*
- *Permitir a manipulação dos dados de forma simples e intuitiva, para que a informação pretendida seja mais facilmente obtida;*
- *Possuir uma interface gráfica que permita total controlo, por parte do utilizador, do funcionamento da aplicação;*
- *Grafismo que possibilite a definição de parâmetros do algoritmo utilizado e o acompanhamento da evolução do mesmo;*
- *Possuir opções de saída da informação, nomeadamente dos horários construídos, gráficos da solução e resultados obtidos.*

As potencialidades do sistema desenvolvido encontram-se direccionadas, o mais possível, para o utilizador da aplicação, nomeadamente ao nível da facilidade de representação de um problema, da simplicidade de manutenção dos dados e da facilidade de obtenção e percepção da informação resultante do processo de construção dos horários.

5.4 INTEGRAÇÃO DO SISTEMA

O sistema desenvolvido e aqui descrito nesta dissertação, encontra-se integrado num amplo projecto de desenvolvimento de um sistema de planeamento e escalonamento universitário: o *UNIPS – University Planning and Scheduling System.* Os objectivos desse projecto passam pela integração, no mesmo, de todo o trabalho até agora desenvolvido e pela sua continuidade, evoluindo no sentido da resolução dos outros dois problemas associados: a *Calendarização de Exames* e a *Distribuição de Serviço Docente*, bem como o estudo de outras técnicas de resolução dos problemas. Para tal, poderá:

- *Fazer uso da linguagem UNILANG, atrás descrita, para representação dos problemas;*
- *Utilizar a base de dados completa, já previamente definida;*

- *Dar continuidade ao desenvolvimento do sistema de "parsing" construído;*
- *Fazer uso da interface gráfica desenvolvida;*
- *Utilizar o algoritmo desenvolvido de Arrefecimento Simulado.*

Finalmente, o projecto *UNIPS* terá uma implementação distribuída, podendo os diversos problemas ser resolvidos por diversos utilizadores ou agentes inteligentes em substituição desses utilizadores, em determinadas partes da instituição de ensino.

A interface gráfica desenvolvida e o estudo computacional efectuado, serão apresentados a seguir, neste capítulo.

5.5 INTERFACE GRÁFICA

5.5.1 INTRODUÇÃO

Actualmente, o desenvolvimento de uma qualquer aplicação de construção de horários deverá ser realizado de forma a que a apresentação gráfica seja considerada como um dos parâmetros fundamentais, para o correcto funcionamento da mesma. Isto torna-se ainda mais verdadeiro quando se percebe a importância da interactividade com o utilizador. Como tal, deverá ser desenvolvida uma interface amigável, prática e funcional, para o utilizador da aplicação, através da qual poderá executar a manutenção dos diversos dados que compõem o problema. É necessário ter em mente um conjunto de requisitos no sentido de ser construída uma aplicação prática e funcional para construção de horários.

Em primeiro lugar, a representação de dados deverá seguir, o mais possível, a natureza dos dados do mundo real. Deverá haver suficiente representação de disciplinas, docentes e salas, bem como as suas propriedades individuais. Estes dados deverão permitir a sua alteração por vontade do utilizador por forma a reflectir as mudanças do mundo real. De seguida, as restrições relacionadas com estes dados deverão ser impostas ao longo da representação do problema. Isto conduz a uma forma de produção de horários correctos.

O passo seguinte encontra-se relacionado com a geração de uma solução quase óptima do problema. Isto envolve a satisfação, o mais possível, dos critérios de qualidade. Mas, como qualquer utilizador atribui importância diferente aos *standards* de qualidade, deverá ser fornecida uma qualquer forma de atribuir prioridades a esses *standards*. Quanto maior a prioridade, maior esforço deverá ser realizado pela aplicação, para a satisfação de resultados.

O re-escalonamento é outro aspecto a ter em consideração. Deve ser criado de determinada forma que um dado horário pode ser visto como um *input* adicional. A seguir, deve tentar re-escalonar esse horário, efectuando o menor número possível de alterações, se para tal as houver.

Uma aplicação automática não se consegue aproximar da complexidade (ou simplicidade, talvez) como a qual um humano lida, como é o caso do problema complexo de construção de horários. Isto, em conjunção com o facto de que a solução final pode necessitar de facto de pequenas alterações, torna necessário a possibilidade de edição posterior do horário gerado.

Sendo um dos objectivos de desenvolvimento, aquando da definição do projecto, a aplicação construída possuí uma interface gráfica, prática e funcional, que permite a devida apresentação dos dados e cuja interacção com o utilizador é fácil e intuitiva. O sistema apresenta uma interface desenvolvida de acordo com os princípios básicos de interfaces homem-máquina *[Cox e Walker 1993]*, *[Pyle 1995]*, *[Reis 1995a]*. E como tal, apresentam-se as características gerais do sistema:

- *Preocupações estéticas de forma a tornar a interface agradável;*
- *Utilização de menus, caixas de diálogo e botões para permitir a introdução de comandos do utilizador;*
- *Utilização de janelas sobreponiveis, na visualização da informação;*
- *Utilização simultânea do teclado e do rato na introdução de comandos;*
- *Utilização de terminologia "standard" em todos os menus e caixas de diálogo;*
- *Entrada de dados robusta minimizando a introdução de valores errados;*
- *Permissão de total controlo do funcionamento da aplicação ao utilizador.*

Como anteriormente referido, a aplicação encontra-se direccionada para permitir a sua fácil utilização, e como tal, é importante a interface gráfica construída, que vai definir o nível de interactividade do sistema através do qual o utilizador estará em posição de introduzir, gravar, ler e, de uma forma geral, manipular os dados do problema e os resultado obtidos.

Sendo assim, tendo em atenção a relevância da interface gráfica, foi desenvolvido um conjunto de formulários para facilitar a manipulação dos dados por parte do utilizador. Todos eles apresentam uma estrutura idêntica, ao nível dos componentes construídos, no sentido de familiarizar o mais depressa possível, o utilizador com a aplicação desenvolvida.

5.5.2 ECRÃ INICIAL

Como anteriormente referido, o projecto *UNIPS* passa pelo desenvolvimento de uma aplicação prática que envolve uma interface gráfica e cujo ecrã inicial é apresentado na Figura 14.

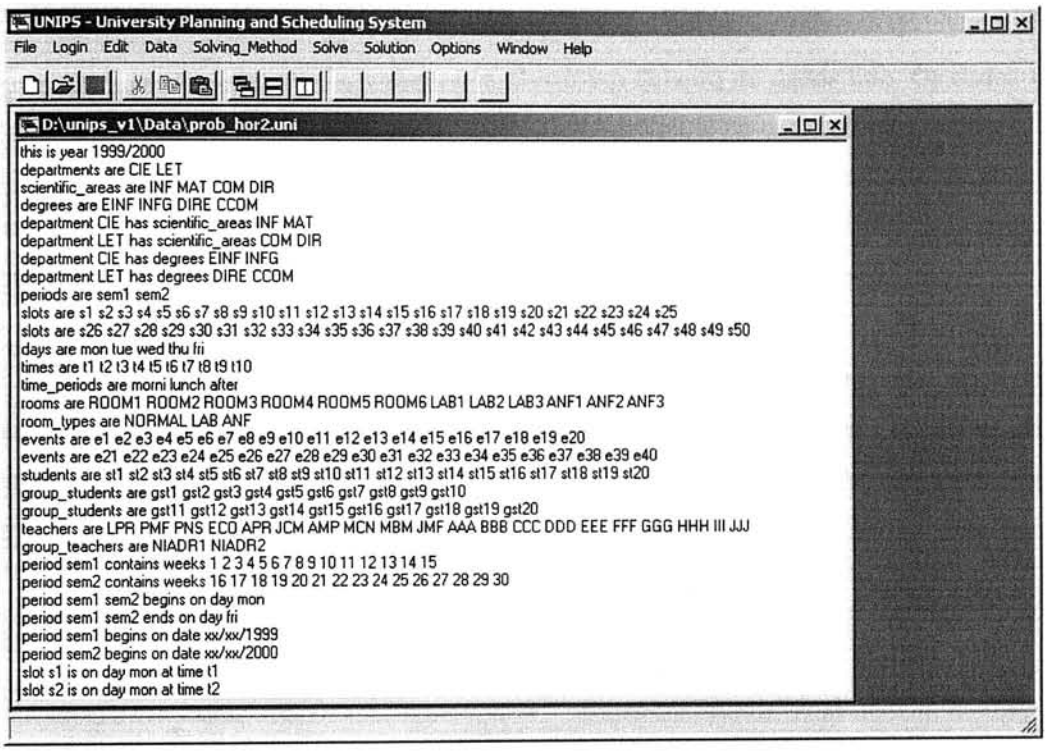


Figura 14: Ecrã Inicial

O referido ecrã possui um menu, com diversas opções para, nomeadamente:

- *Manutenção (criação, leitura, gravação, impressão) dos ficheiros de dados a utilizar na aplicação;*
- *Conexão (Login) de utilizadores na aplicação de forma distribuída;*
- *Edição de texto;*
- *Tratamento dos dados (lidos do ficheiro de dados ou criados num novo problema definido);*
- *Definição de um método a utilizar na resolução de um dado problema;*
- *Resolução de um problema (Horário Turma-Docente, Horário das Disciplinas, Calendarização de Exames, Definição de Ofertas de Disciplinas, Horário de um Aluno, Alocação de Staff, Alocação de Vigilantes, Alocação de Salas) através de um dos métodos escolhido;*
- *Apresentação das soluções dos diversos problemas resolvidos;*

- *Definição da apresentação da aplicação (cores, opções de display, distribuição das janelas abertas, entre outras opções);*
- *Desenvolvimento de opção de ajuda, para o utilizador poder obter informação que o apoie na utilização da aplicação.*

Na Figura 14, acima apresentada, encontra-se aberto um ficheiro de dados lido. Os dados estão representados no formato da linguagem *UNILANG* e fazem parte de um problema definido. Esse problema é possível resolver pelo sistema desenvolvido, encontrando-se essa forma de resolução descrita e analisada neste capítulo.

5.5.3 ESTRUTURA GERAL DA INSTITUIÇÃO

Foi desenvolvido um formulário para representar a estrutura da instituição de ensino, ao nível dos departamentos, áreas científicas e cursos que a compõem. Neste formulário é possível criar e eliminar departamentos, áreas científicas e cursos, bem como alterar a informação associada. Na Figura 15 é apresentado o referido formulário.

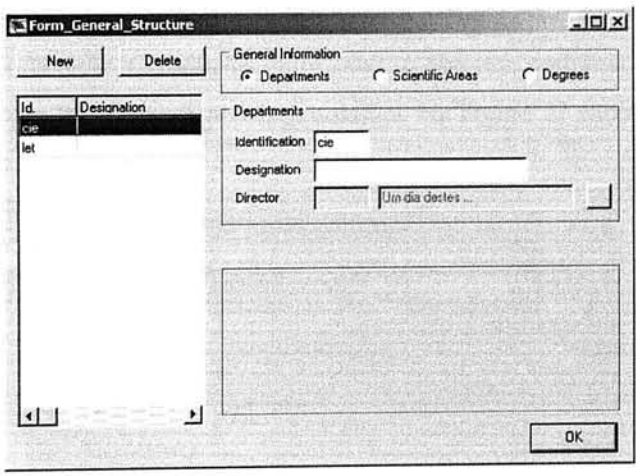


Figura 15: Estrutura Geral da Instituição

5.5.4 PERÍODOS LECTIVOS

O funcionamento de uma instituição pode dividir-se em períodos lectivos, cuja duração (anual, semestral, quadrimestral, ou outra) pode ser definida pelo utilizador. Na Figura 16 apresenta-se o formulário respectivo.

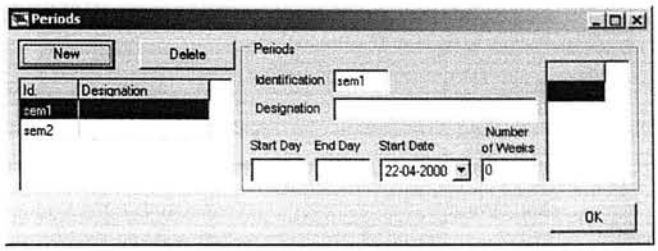


Figura 16: Períodos

Utilizando este formulário podem-se definir semestres, trimestres ou outro tipo de período (tal como férias de Natal) e as suas características tais como a data de início e fim e o número de semanas.

5.5.5 ESPAÇO

Uma instituição de ensino tem um determinado espaço, composto por diversas salas, que permite a leccionação dos diversos eventos existentes nos planos curriculares dos cursos que a instituição ministra. Essas salas têm uma determinada capacidade e permitem um dado número máximo de eventos, poderão ser de diversos tipos, ter algum tipo de preferência e permitir a sobreposição de eventos. É possível ainda definir distâncias entre salas, o que tem particular interesse em eventos que se deseje que tenham lugar o mais próximo possível. Na Figura 17 encontra-se representado o formulário respectivo.

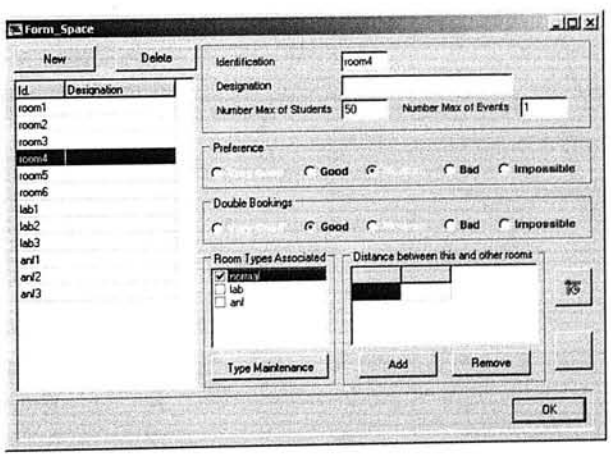


Figura 17: Espaço

5.5.6 TEMPO

É possível definir os dias e horas de funcionamento da instituição de ensino, utilizando, para tal, o formulário desenvolvido e que se encontra representado na Figura 18.

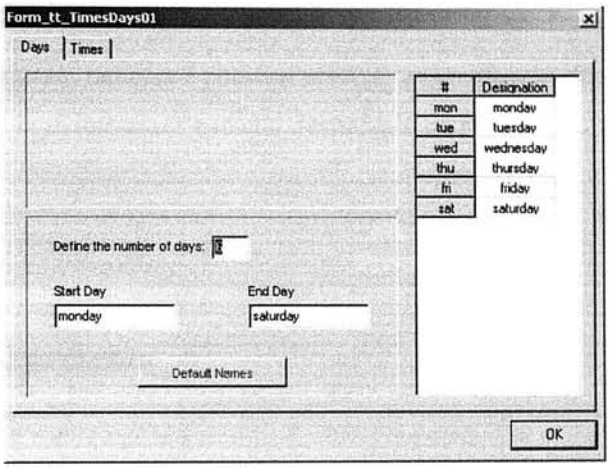


Figura 18: Definição de dias e horas

Caso esta informação não se encontre definida no ficheiro de texto que contém o problema, é possível, no entanto, utilizar dias e horas definidas por defeito na aplicação.

5.5.7 GRUPOS DE ALUNOS E ALUNOS

É possível associar alunos de uma instituição em grupos, usualmente designados por turmas, e definir as suas características associadas. Na Figura 19 encontra-se o formulário desenvolvido para tal.

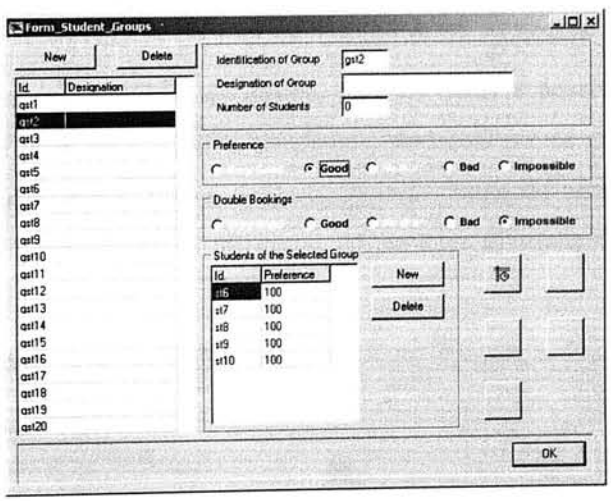


Figura 19: Alunos e Grupos de Alunos

É possível definir grupos de estudantes (turmas) e para cada um deles, é possível definir os alunos que a compõe (ou unicamente qual o seu número).

5.5.8 DOCENTES E GRUPOS DE DOCENTES

De forma análoga aos alunos, também é possível efectuar a manutenção dos docentes da instituição e associá-los em grupos com interesses comuns (formulário da Figura 20).

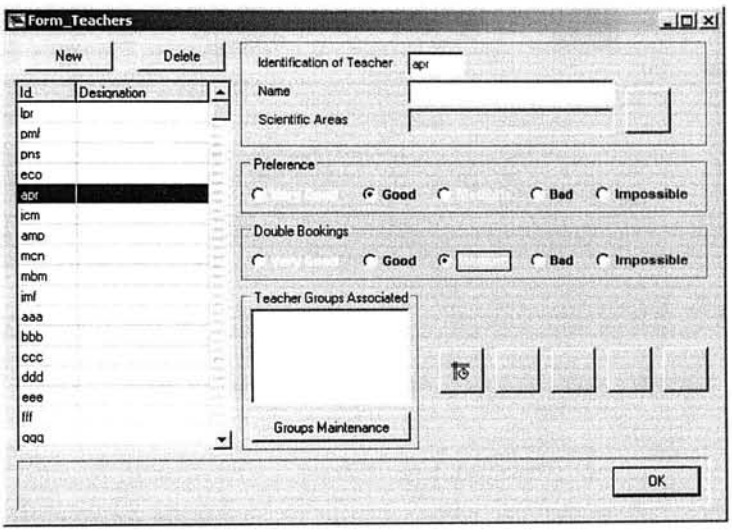


Figura 20: Docentes e Grupos de Docentes

No formulário é possível definir novos docente e associá-los a áreas científicas e grupos (caso estes sejam utilizados).

5.5.9 PREFERÊNCIAS TEMPORAIS DE SALAS, GRUPOS DE ALUNOS E DOCENTES

Existem cinco recursos da instituição de ensino, para os quais é possível definir preferências temporais. São eles: as salas, as turmas, os estudantes, os docentes e os grupos de docentes. Para cada um deles é possível definir o nível de preferência ao longo do período de tempo definido (tipicamente uma semana), representado cada qual, por uma cor diferente. O formulário respectivo encontra-se representado na Figura 21.

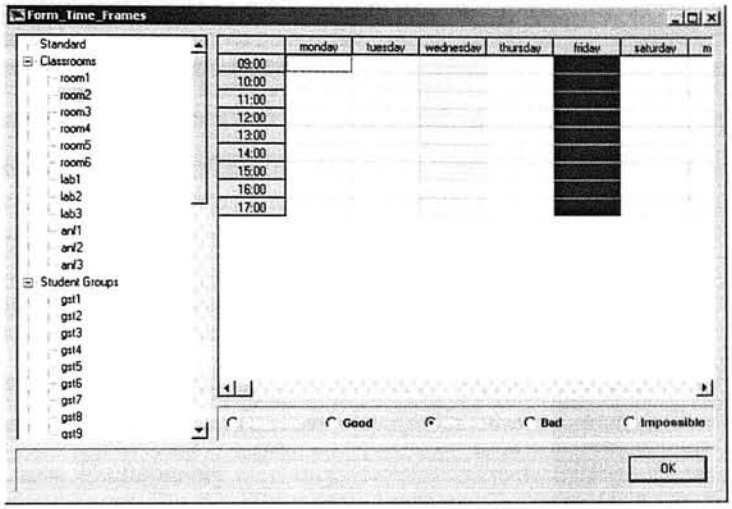


Figura 21: Preferências de tempos de salas, grupos de alunos e docentes

5.5.10 EVENTOS

Uma instituição de ensino lecciona um diverso conjunto de eventos, com determinadas características e preferências e para os quais se encontram associados alunos (ou turmas), docentes (ou grupos de docentes) e salas. Na Figura 22 apresenta-se o formulário de desenvolvido para a manutenção dos diversos eventos.

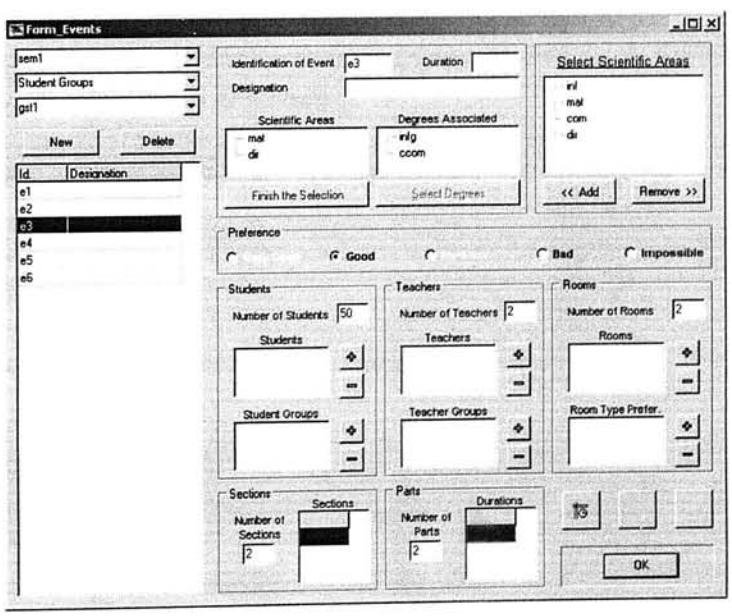


Figura 22: Eventos

Para cada evento pode haver um determinado conjunto de ofertas (diferentes turmas desse evento) de leccionação (por exemplo: o docente "*lpr*" e o docente "*pmf*" podem leccionar turmas diferentes

do evento "*e18*"). Cada evento pode ainda subdividir-se em diversas partes (por exemplo: uma parte de 2 tempos e outra de 1 tempo de duração). O formulário apresentado permite ainda que para cada evento não sejam definidos quais os estudantes, docentes ou salas específicas mas sim unicamente qual o seu número. O formulário permite ainda associar eventos a áreas científicas e cursos.

5.5.11 MANUTENÇÃO DO ALGORITMO

Foi desenvolvido um formulário com vista permitir ao utilizador a manutenção do algoritmo escolhido e implementado para a resolução do *Problema de Construção de Horários*. Aí, é permitido um conjunto de operações:

- *A definição do valor dos parâmetros mais importantes que irão determinar o comportamento do algoritmo;*
- *Inicializar o algoritmo calculando uma possível solução aleatória do problema;*
- *Executar o algoritmo utilizando os parâmetros seleccionados;*
- *Aceder a um formulário que apresenta, em tempo real, a construção do horário de determinado recurso (sala, docente ou turma), com as sucessivas atribuições e desatribuições dos eventos associados, nos respectivos slots;*
- *Visualizar a evolução do algoritmo através de gráficos da Solução, da Qualidade da solução, da Temperatura e da Probabilidade de Aceitação.*

Na Figura 23, abaixo apresentada, é possível observar o referido formulário.

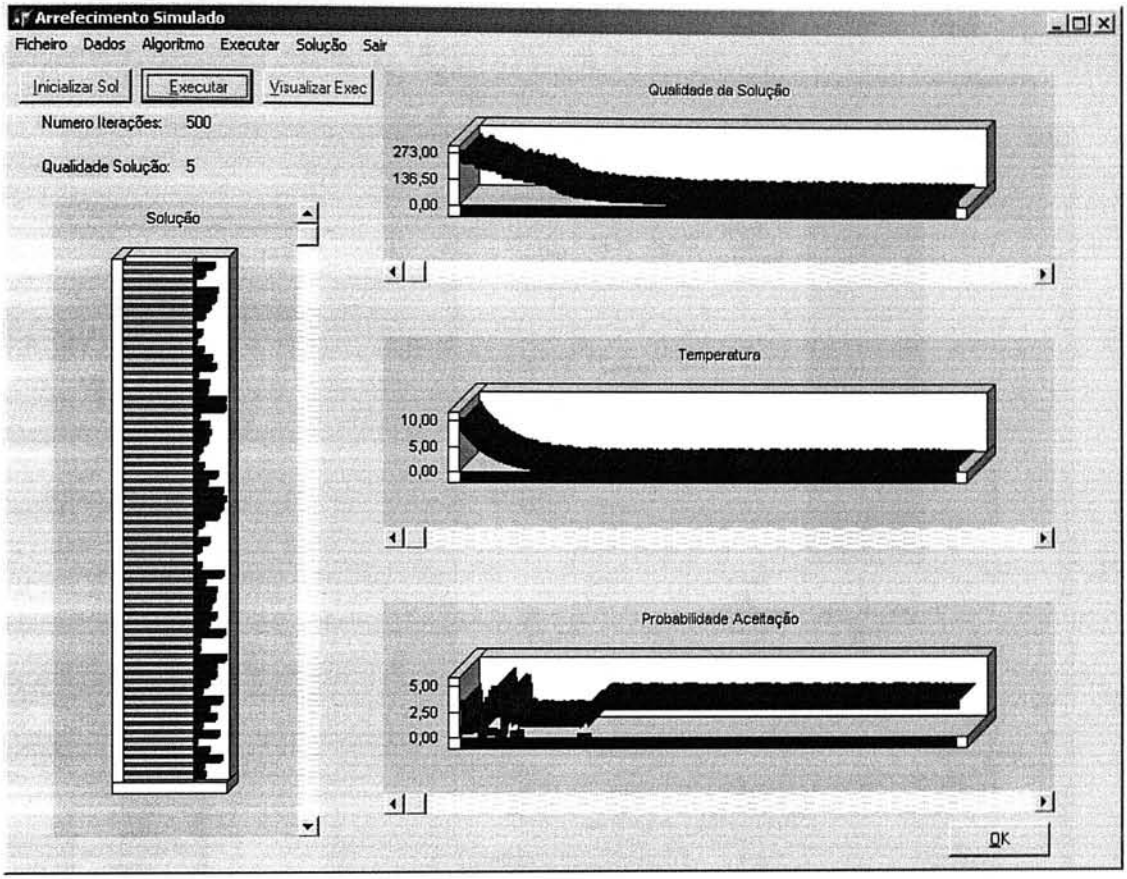


Figura 23: Manutenção do algoritmo de Arrefecimento Simulado

5.5.12 VISUALIZAÇÃO DOS RESULTADOS FINAIS

Finalmente, é apresentado o formulário que permite a observação dos horários, obtidos de forma automática, pela execução do algoritmo. Aí, é permitido visualizar os horários dos diversos recursos existentes, bem como as respectivas preferências temporais de cada um. É possível ainda, para um determinado recurso, observar, em tempo real, a evolução do seu horário, quando é escolhida a opção de execução do algoritmo.

Concluído o processo automático de construção dos horários, é permitido ao utilizador, de forma interactiva (com a simples utilização do rato e dos sues botões esquerdo e direito), a mudança de eventos das posições temporais (*slots*) atribuídas pelo algoritmo. Isto é permitido no sentido de dar, ao utilizador, total liberdade de escalonar os horários da forma mais conveniente e possível de efectuar. Existem cores atribuídas às células onde estão escalonados os eventos. Quando surge uma cor amarela numa célula, significa que existe uma sobreposição de um evento, podendo o utilizador tentar mudar um dos eventos para outro *slot* que esteja disponível. A cor vermelha numa célula

indica a sobreposição de mais do que um evento. Representado a azul encontram-se as preferências temporais do recurso escolhido. Na Figura 24 seguinte é apresentado o referido formulário.

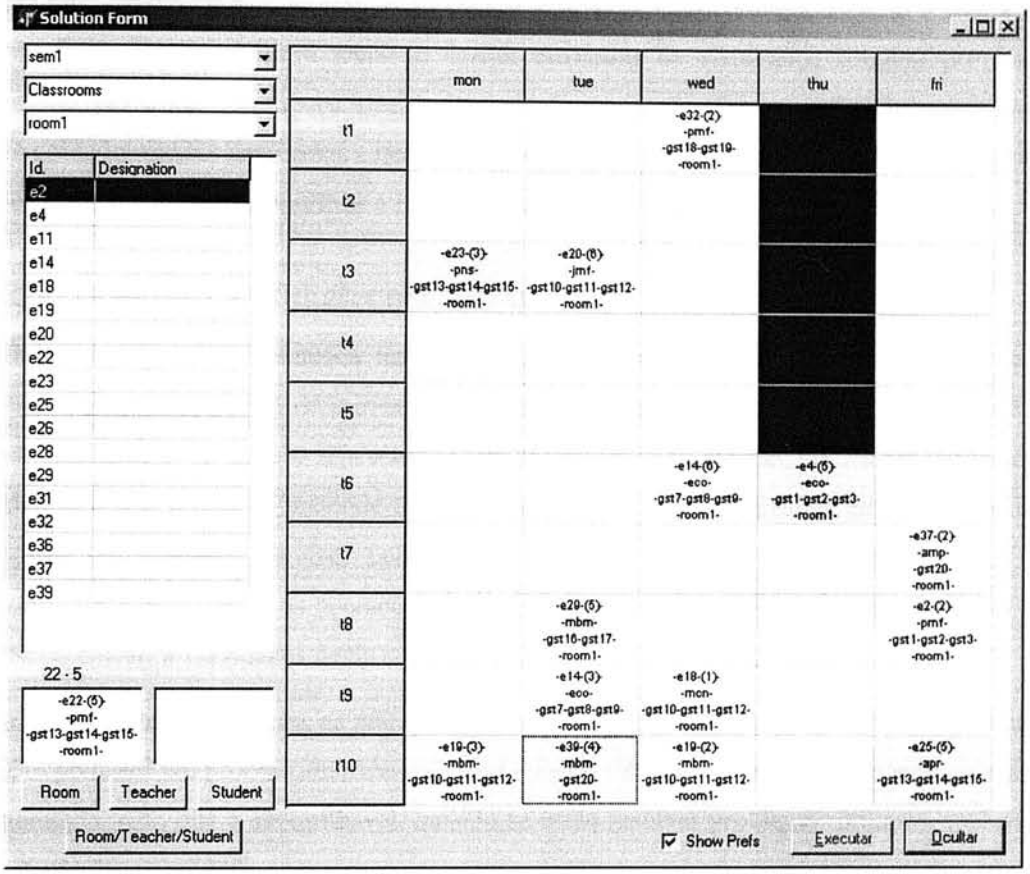


Figura 24: Visualização e tratamento da solução

No canto superior esquerdo do formulário pode ser seleccionado o período (no caso semestre) a visualizar, o tipo de recursos que se pretendem visualizar (salas, docentes ou turmas) e qual o recurso específico. Após efectuar esta selecção, no quadro do lado esquerdo, são visualizados todos os eventos desse recurso e no quadro do lado direito, é visualizada a distribuição temporal desses eventos.

5.6 ESTUDO COMPUTACIONAL

5.6.1 INTRODUÇÃO

Foi realizado um estudo computacional para avaliar o desempenho de uma das meta-heurísticas apresentadas, tendo sido escolhido o *Arrefecimento Simulado*. Neste estudo, houve a preocupação de avaliar não só a qualidade das soluções obtidas, ou seja a eficácia do algoritmo, mas também a

sua eficiência. Acrescente-se que, pelo facto de se estar a trabalhar com uma meta-heurística, é necessário desenvolver um algoritmo ("construtivo") que produza uma solução inicial e a seguir permita que, iterativamente, se pesquise o espaço de soluções, procurando melhorar a qualidade da solução final. Como tal, deve tentar-se definir estruturas de vizinhança simples, por forma a representar um esforço computacional baixo, por forma a não afectar a eficiência das mesmas. Foram cuidadosamente analisados e implementados os parâmetros do algoritmo, nomeadamente no que se refere ao cálculo de conflitos e impossibilidades, bem como de avaliação da solução.

5.6.2 O ALGORITMO DE ARREFECIMENTO SIMULADO UTILIZADO

Há vários aspectos importantes a ter em atenção na concepção de uma meta-heurística. Em primeiro lugar, é necessário definir um ponto de partida para a pesquisa a efectuar, isto é, uma solução inicial (embora tal não seja estritamente necessário, uma solução poderá ser "admissível"). Atendendo a que o que se pretende é obter boas soluções, então definida essa solução inicial, começa-se um processo iterativo com vista a melhorar a qualidade da solução. Para tal, é necessário especificar quais as operações de modificação a que uma solução será sujeita, ou seja é necessário definir a vizinhança dessa solução.

Outro aspecto a ter em atenção, na prática, é a afinação dos parâmetros da heurística. O processo de afinação pode ser um processo moroso e de importância vital para a obtenção eficiente de soluções de qualidade, pelo que é aconselhável, quando se estão resolver problemas práticos, realizar esta afinação de forma cuidada.

De referir que um dos objectivos principais deste trabalho foi estudar em termos gerais a adequabilidade de uma meta-heurística para resolver problemas de construção de horários. Mais do que afinar o modelo para um hipotético problema, aumentando a sua eficiência para esse problema em particular (recorde-se que o valor dos parâmetros pode variar de problema para problema) pretendia-se verificar se, mesmo num estado mais ou menos bruto, uma meta-heurística é uma ferramenta adequada a utilizar na resolução deste tipo de problemas.

Apresenta-se abaixo, o algoritmo implementado para a resolução de problemas de construção de horários.

```
procedimento arrefecimento_simulado;
Inicio
    av1 := calcula_avaliacao_inicial;
    temp := PA_TemperaturaInicial;
    niter := 0;
    repetir
      c := 0;
      repetir
        vizinho := calcula_vizinho(event, part, slot);
        av2 := calcula_avaliacao(av1, event, part, slot);
        delta := av2 - av1;
        se (delta <= 0) ou (valor_aleatório < exp(-abs(delta/temp))) então
        Inicio
          nova_solução := vizinho ;
          av1 := av2;
        Fim;
        c := c + 1;
      até c = PA_NumeroRepeticoes;
      niter := niter+1;
      diminui_temperatura(temp);
      actualiza_formulário_da_solução;
      escreve_solucao(niter, av1, delta, temp);
    até criterio_paragem(temp, niter, av1);
Fim;
```

Algoritmo 4: Algoritmo de *Arrefecimento Simulado* utilizado

A seguir são apresentadas algumas das partes mais importantes que compõem o algoritmo.

5.6.2.1 SOLUÇÃO INICIAL

O algoritmo implementado neste estudo, não tem especificamente como objectivo obter "boas" hipóteses de soluções inicias. Na verdade, a definição de uma "boa" hipótese de solução inicial não é óbvia. Essa hipótese de solução pode ser "boa" quando comparada com determinada solução x, mas péssima se comparada com outra solução y. Além disso, ter uma "boa" hipótese de solução de partida não parece ser, em geral, uma característica importante no desempenho das meta-heurísticas, quando se está a trabalhar com vários objectivos. Outro aspecto importante a salientar, no que diz respeito às características das soluções iniciais obtidas, é a admissibilidade da solução. Tendo em atenção que um dos objectivos a ser avaliado é o da violação de restrições, embora tenha sido definida uma regra de prioridade, com a qual se procura obter soluções admissíveis, não existe nenhum processo de *backtracking*, no caso das restrições serem violadas.

Para o algoritmo implementado, a hipótese de solução inicial é gerada aleatoriamente, para todos as partes que compõem os eventos do problema. Apesar de ser possível considerar outras regras de

prioridade, possivelmente mais interessantes noutras situações, como atrás referido, a obtenção de "boas" hipóteses de soluções inicias não constitui, no âmbito deste trabalho, uma preocupação.

5.6.2.2 ESTRUTURA DE VIZINHANÇA

No contexto dos algoritmos de pesquisa local, e em particular das meta-heurísticas, a estrutura de vizinhança de uma solução assume particular importância. Para a sua definição, devem ser tidos em atenção alguns aspectos, como os seguintes:

- *A obtenção de uma solução vizinha deve ser simples, do ponto de vista do esforço computacional envolvido;*
- *Todo o espaço de soluções deve ser acessível a partir de qualquer solução, para que não hajam restrições na pesquisa;*
- *O valor da função objectivo de soluções vizinhas de uma solução* x*, deve ser tão próximo quanto possível do valor em* x*.*

No algoritmo implementado, o cálculo da solução vizinha é efectuado da forma mais simples possível, ou seja, aleatoriamente. Um vizinho é então calculado, seleccionando aleatoriamente uma parte de um evento e em seguida calculando um novo slot e uma nova sala para essa parte desse evento. Todo o espaço de soluções de partes de eventos é acessível utilizando esta vizinhança.

5.6.2.3 FUNÇÃO DE AVALIAÇÃO, ESQUEMA DE ARREFECIMENTO E CRITÉRIO DE PARAGEM

Existe um conjunto de parâmetros importantes para o desempenho do algoritmo definido, e como tal é importante que o utilizador os possa redefinir. Para tal, foi desenvolvido um pequeno formulário onde o utilizador pode escolher os valores dos parâmetros, de acordo com o problema que pretende resolver. Os referidos parâmetros a que o utilizador tem acesso são: *Número Máximo de Iterações* e *Número de Repetições* (por iteração). Quanto à temperatura, os parâmetros são: *Temperatura Inicial*; *Diminuição da Temperatura* e *Temperatura Mínima*. O utilizador pode ainda escolher o tipo de diminuição da temperatura que pretende: *Multiplicativa* ou *Subtractiva*. Apresenta-se na Figura 25, o referido formulário.

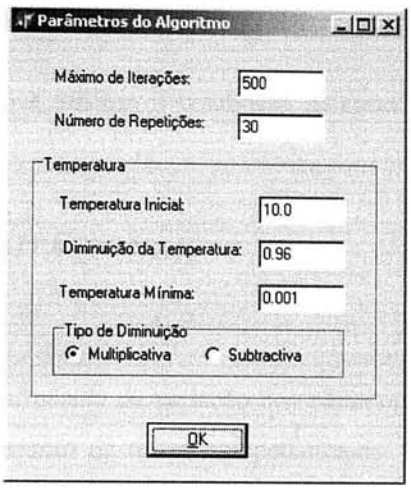


Figura 25: Formulário de manutenção dos parâmetros do algoritmo

A execução do algoritmo passa pela avaliação da solução inicial, tendo sido desenvolvida uma função para obter esse valor. Tanto essa função como a função de avaliação geral da solução, utilizam funções de cálculo de impossibilidades (restrições rígidas, ao nível de preferências dos recursos) e de conflitos entre eventos com recursos em comum.

$$FAval = \text{Im}\,possibilidades + Conflitos$$

$$\text{Im}\,possibilidades = \sum_{slot=1}^{nslots} \left(\sum_{teacher=1}^{nteachers} \text{Im}\,possibilidades_{teacher} + \sum_{groupstudent=1}^{ngroupsstudents} \text{Im}\,possibilidades_{groupstudent} + \sum_{room=1}^{nrooms} \text{Im}\,possibilidades_{room} \right)$$

$$Conflitos = \sum_{slot=1}^{nslots} \left(\sum_{teacher=1}^{nteachers} Conflitos_{teacher} + \sum_{groupstudent=1}^{ngroupsstudents} Conflitos_{groupstudent} + \sum_{room=1}^{nrooms} Conflitos_{room} \right)$$

As impossibilidades referem-se a aulas que são escalonadas em períodos em que os docentes, turmas ou salas estão indisponíveis, enquanto os conflitos referem-se a sobreposição de aulas nos horários dos docentes, turmas ou salas.

O valor da temperatura inicial, definido por defeito, é de 10.0, podendo este parâmetro ser alterado pelo utilizador. Com o objectivo de, no final, "maus movimentos" terem uma probabilidade pequena de serem aceites, a temperatura definida no início da execução do algoritmo vai diminuindo ao longo do tempo. Existem dois tipos de diminuição: a multiplicativa (por defeito) e a subtractiva. Na diminuição multiplicativa, o valor da temperatura em cada iteração é multiplicado pelo valor do parâmetro *Diminuição de Temperatura*. Se a temperatura descer abaixo do valor mínimo então assume esse valor mínimo. Na diminuição de temperatura subtractiva, o valor do parâmetro *Diminuição de Temperatura* é subtraído, em cada iteração, ao valor da temperatura. O utilizador pode seleccionar o tipo de diminuição que pretende, de acordo com o problema a resolver.

Diminuição Multiplicativa: $T_{n+1} = T_n * DimT, \quad Se\ T_{n+1} < MinT\ Então\ T_{n+1} = MinT$

Diminuição Subtractiva: $T_{n+1} = T_n - DimT, \quad Se\ T_{n+1} < MinT\ Então\ T_{n+1} = MinT$

O critério de paragem definido a utilizar, é o número máximo de iterações, valor possível de redefinir pelo utilizador.

5.7 RESULTADOS COMPUTACIONAIS

5.7.1 INTRODUÇÃO

Com o objectivo de avaliar a influência da variação dos parâmetros associados ao arrefecimento simulado, foi realizado um conjunto de testes computacionais. Para a realização dos mesmos, foram definidos diversos problemas, de vários graus de dificuldade, sendo analisada a capacidade do algoritmo, variando os seus parâmetros, na resolução dos problemas, tendo sido analisado também o tempo necessário para a obtenção de resultados. Todos os testes foram realizados numa máquina equipada com um processador Pentium III a 550 Mhz, com 256 megabytes de memória RAM e um disco rígido com 6 gigabytes de capacidade.

5.7.2 DEFINIÇÃO DE PROBLEMAS

O processo de definição de problemas foi realizado da forma previamente apresentada e explicada, ou seja, através da construção de ficheiros de textos. Em cada ficheiro, é definido um problema de construção de horários utilizando a linguagem *UNILANG*, com todos os componentes necessários à definição completa do mesmo. De seguida, a partir da aplicação gráfica desenvolvida, é efectuada a leitura do ficheiro e os dados que compõem o problema são imediatamente colocados em memória, nas estruturas de dados respectivas. Neste estudo foram considerados três problemas, cujos níveis de dificuldade variam de forma crescente. Estudaram-se diversos comportamentos, nomeadamente, a variação da temperatura ao longo da execução do algoritmo e a evolução da qualidade da solução. Tendo em conta que a avaliação da solução é realizada através de uma função que calcula o total de restrições violadas pela solução, a qualidade da solução é melhor quanto menor o seu valor, sendo zero o óptimo.

5.7.3 PROBLEMA DE NÍVEL DE DIFICULDADE DE RESOLUÇÃO SIMPLES

5.7.3.1 Definição

Foi definido um problema, cujo nível de dificuldade apresentado é relativamente reduzido. O ficheiro com a descrição do problema apresenta-se em anexo (*Anexo 1: Problema nº 1*). As principais características do problema são:

Características	*Valores*
Número de Dias	3
Número de Horas	5
Número de *Slots*	15
Número de Salas	5
Número de Turmas	4
Número de Docentes	5
Número de Eventos	10
Número de Partes de Eventos (Aulas)	28

Tabela 22: As principais características do Problema nº 1

Tendo em conta que a resolução do problema implica atribuir a cada uma das 28 aulas, um slot (tempo) e uma sala e que temos 15 slots e 5 salas possíveis, a dimensão do espaço de pesquisa deste problema é:

$$Espaço\_Pesquisa_1 = Aulas^{Slots*Salas} = 28^{15*5} = 28^{75} = 3.4*10^{78}$$

Existe ainda um conjunto de propriedades das características, que atribuem mais alguma dificuldade ao problema, como sejam:

- *Todos os eventos encontram-se divididos em 3 partes (aulas), à excepção de 2 eventos, cujo número de partes é 2, o que perfaz as referidas 28 aulas;*
- *Cada docente lecciona 2 eventos;*
- *Salas, Grupos de Alunos e Docentes têm preferências temporais de aceitação de eventos.*

5.7.3.2 Parâmetros Iniciais e Temperatura

Os valores dos parâmetros iniciais, definidos para a resolução deste problema, são os apresentados na Tabela 23, tendo sido experimentados os dois tipos de diminuição de temperatura (Multiplicativa e Subtractiva).

Parâmetros	*Multiplicativa*	*Subtractiva*
Número Máximo de Iterações	500	500
Número de Repetições	30	30
Temperatura Inicial	10.0	10.0
Diminuição da Temperatura	0.99	0.05
Temperatura Mínima	0.001	0.001

Tabela 23: Valores iniciais dos parâmetros do algoritmo

Os valores atribuídos aos parâmetros resultaram da realização de diversos testes, obtidos através de um processo de tentativa/erro.

Os gráficos resultantes da variação da temperatura, segundo os dois tipos de diminuição, apresentam-se na Figura 26 e Figura 27.

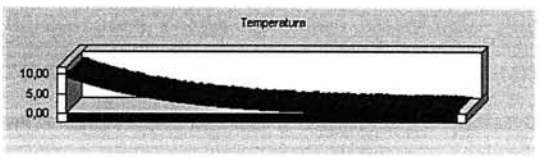


Figura 26: Diminuição Multiplicativa da Temperatura de 0.99

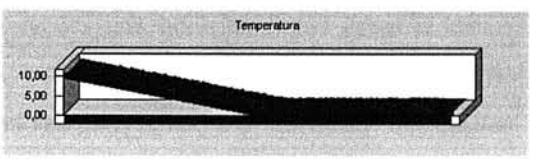


Figura 27: Diminuição Subtractiva da Temperatura de 0.05

Enquanto a diminuição multiplicativa da temperatura apresenta valores suavizados, ao longo das iterações definidas, a diminuição subtractiva apresenta um decréscimo acentuado (tipo "rampa") até ao valor mínimo, a partir do qual, a temperatura não se altera.

5.7.3.3 Qualidade da Solução

Apresentam-se, na Figura 28 e na Figura 29, os gráficos da evolução da qualidade da solução, utilizando variação multiplicativa e subtractiva da temperatura, respectivamente. Analisando-os, deduz-se que, através da variação subtractiva, a solução melhora mais rapidamente, acompanhando a diminuição da temperatura (Figura 27), e como tal, a solução óptima é atingida com um menor número de iterações e consequentemente em menor tempo. Facilmente se conclui que, como se trata de um problema cuja resolução é relativamente simples, provavelmente, não seria necessário o algoritmo de *Arrefecimento Simulado* para a sua resolução. A solução óptima poderia ser obtida através de um método simples de pesquisa local.

Figura 28: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura

Figura 29: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura

Nos diversos testes realizados, a qualidade da solução para este problema foi, na maior parte dos casos, de 0, mas houve algumas situações (muito raras) em que os resultados obtidos foram 1 e 2, sendo estes valores atingidos relativamente cedo, quando falamos em número de iterações efectuadas. Tal deve-se a que, apesar de se tratar de um problema simples, o espaço de atribuição de soluções é limitado, não permitindo grandes alterações nos horários finais construídos. As soluções iniciais obtidas "prendem" determinados eventos, não permitindo a sua posterior movimentação, devido à temperatura que se torna cada vez mais baixa, e o "tamanho" dos "saltos" que é permitido efectuar, torna-se pequeno.

O tempo médio gasto para a resolução do problema foi de 1.5 segundos.

5.7.4 PROBLEMA DE NÍVEL DE DIFICULDADE DE RESOLUÇÃO MÉDIO

5.7.4.1 Definição

Foi definido um problema, cujo nível de dificuldade apresentado é médio, quando comparado com o problema anteriormente apresentado. O ficheiro com a descrição do problema apresenta-se em anexo (*Anexo 2: Problema nº 2*). As principais características que compõem o problema são:

Características	*Valores*
Número de Dias	5
Número de Horas	10
Número de *Slots*	50
Número de Salas	12
Número de Turmas	20
Número de Docentes	10
Número de Eventos	40
Número de Partes (Aulas)	288

Tabela 24: As principais características do Problema nº 2

A dimensão do espaço de pesquisa deste problema é bastante superior à do anterior:

$$Espaço\_Pesquisa_2 = Aulas^{Slots*Salas} = 288^{50*12} = 288^{600} = 4.3*10^{1475}$$

Existe ainda um conjunto de propriedades das características, que atribuem alguma dificuldade adicional ao problema, como sejam:

- *Todos os eventos encontram-se divididos em 8 partes (aulas), à excepção de 4 eventos, cujo número de partes é 5 e outros 4 eventos, que foram divididos em 3 partes, o que perfaz as referidas 288 aulas;*
- *Cada docente lecciona 4 eventos;*
- *Salas, Grupos de Alunos e Docentes têm preferências temporais de aceitação de eventos.*

5.7.4.2 Parâmetros Iniciais e Temperatura

Os valores dos parâmetros iniciais, definidos para a resolução deste problema, são os apresentados na Tabela 25, tendo sido experimentados os dois tipos de diminuição de temperatura (Multiplicativa e Subtractiva) e para cada uma delas, foi tentado resolver o problema com 50 e 100 repetições.

Parâmetros	*Multiplicativa*	*Subtractiva*
Número Máximo de Iterações	1000	1000
Número de Repetições	50/100	50/100
Temperatura Inicial	10.0	10.0
Diminuição da Temperatura	0.96/0.99	0.05/0.2
Temperatura Mínima	0.001	0.001

Tabela 25: Valores iniciais dos parâmetros do algoritmo

Os valores atribuídos aos parâmetros resultaram da realização de diversos testes, obtidos através de um processo de tentativa/erro.

Os gráficos resultantes da variação da temperatura, segundo o tipo multiplicativo, e com diminuição da temperatura de 0.96 e 0.99, respectivamente, encontram-se representados na Figura 30 e na Figura 31.

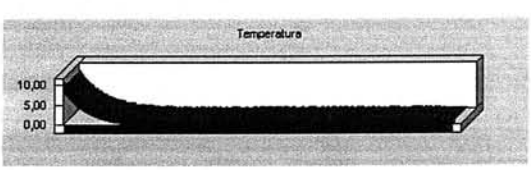


Figura 30: Diminuição Multiplicativa da Temperatura de 0.96

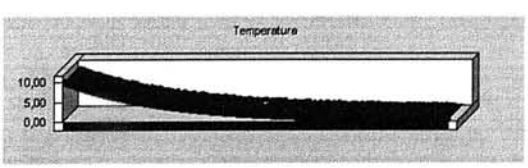


Figura 31: Diminuição Multiplicativa da Temperatura de 0.99

Na diminuição multiplicativa da temperatura, utilizando no parâmetro um valor de 0.96, a temperatura decresce de forma suavizada, mas rapidamente, enquanto que utilizando um valor de 0.99, esse decréscimo é mais lento.

Na resolução dos problemas, aplicando a variação da temperatura segundo o tipo subtractivo, e com diminuição de 0.05 e 0.02, respectivamente, os gráficos obtidos são os representados nas figuras Figura 32 e Figura 33.

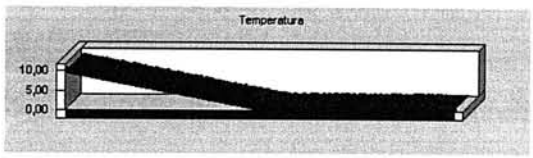


Figura 32: Diminuição Subtractiva da Temperatura de 0.05

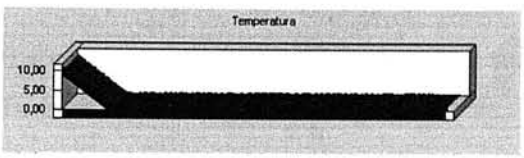


Figura 33: Diminuição Subtractiva da Temperatura de 0.2

5.7.4.3 Qualidade da Solução

A qualidade da solução para este problema foi analisada para ambos os casos (diminuição da temperatura multiplicativa e diminuição subtractiva).

Diminuição Multiplicativa

Para os casos em que a diminuição foi multiplicativa, são apresentados nas figuras Figura 34, Figura 35, Figura 36 e Figura 37 os resultados obtidos, de acordo com a variação de determinados parâmetros.

Para o caso em que o número de repetições foi de 50, e o valor de diminuição da temperatura foi de 0.96 e 0.99, apresentam-se, respectivamente as figuras Figura 34 e Figura 35.

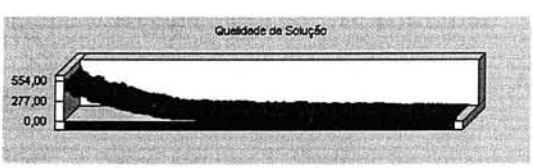


Figura 34: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura de 0.96 e 50 repetições

Figura 35: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura de 0.99 e 50 repetições

Para o caso do valor de diminuição da temperatura ser 0.96, a qualidade da solução final obtida é quase sempre 0. Quando tal não acontece, a qualidade é 1 ou 2, e estes valores surgem,

normalmente, bem antes de terminar o número máximo de iterações definido (1000, para este problema). Isto significa que, determinadas alocações iniciais efectuadas (quando a temperatura é elevada) "prendem" determinados movimentos, que mais tarde se tornam impossíveis de efectuar, devido ao menor grau de liberdade que este valor de diminuição da temperatura oferece e como tal, é impossível alterar, de forma profunda, as alocações efectuadas, mesmo havendo ainda bastantes iterações a efectuar. A solução fica desta forma presa num óptimo local, não sendo possível obter o óptimo global.

Para o caso em que o número de repetições foi de 100, e o valor de diminuição da temperatura foi de 0.96 e 0.99, apresentam-se, respectivamente os gráficos nas figuras Figura 36 e Figura 37.

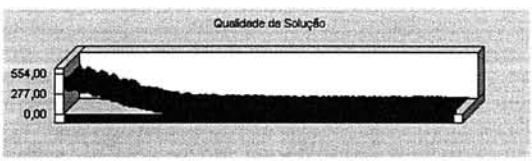


Figura 36: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura de 0.96 e 100 repetições

Figura 37: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura de 0.96 e 100 repetições

Para o caso do valor de diminuição da temperatura ser 0.99, a qualidade da solução final obtida é quase sempre de 1, 2 ou 3. Por vezes, a qualidade atinge o valor 0, necessitando praticamente do número total de iterações definido (1000, para este problema), para que isso aconteça. Conclui-se assim que, o maior grau de liberdade oferecido pelo valor de diminuição de 0.99, origina que exista maior probabilidade de serem efectuadas determinadas alocações iniciais que, mais tarde, são impossíveis de alterar, ou então, para que tal seja possível para algumas delas, é necessário um número maior de iterações de forma a que sejam realizadas iterações suficientes com baixa temperatura para que a solução estabilize.

Diminuição Subtractiva

Para os casos em que a diminuição foi subtractiva, são apresentados nas figuras Figura 38, Figura 39, Figura 40 e Figura 41 os resultados obtidos, de acordo com a variação de determinados parâmetros. Para o caso em que o número de repetições foi de 50, e o valor de diminuição da temperatura foi de 0.05 e 0.2, apresentam-se, respectivamente os gráficos nas figuras Figura 38 e Figura 39.

Figura 38: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura de 0.05 e 50 repetições

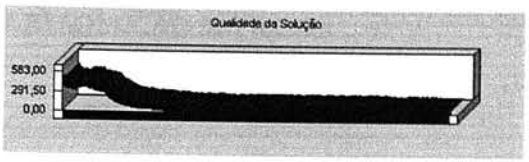


Figura 39: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura de 0.2 e 50 repetições

Utilizando variação Subtractiva da temperatura, com 50 repetições, tanto para o caso do valor da variação ser 0.05 ou 0.2, o valor obtido da Qualidade da Solução final é quase sempre 1 ou 2, conseguindo-se, no entanto, por vezes, atingir o melhor valor da solução. A principal diferença está no número de iterações que é necessário para obter os resultados. Comparando os gráficos, apercebemo-nos que, quando o valor da variação é de 0.05, o número de iterações necessário é cerca de três vezes superior do que quando a variação é de 0.2. Assim, conclui-se que, com 50 repetições torna-se muito difícil atingir a melhor solução (com valor 0), independentemente do valor de variação da temperatura.

Para o caso em que o número de repetições foi de 100, e o valor de diminuição da temperatura foi de 0.05 e 0.2, apresentam-se, respectivamente os gráficos nas figuras Figura 40 e Figura 41.

Figura 40: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura de 0.05 e 100 repetições

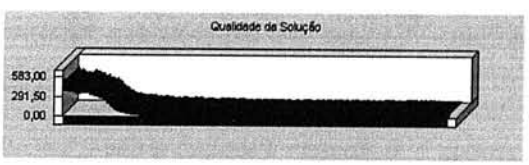


Figura 41: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura de 0.2 e 100 repetições

Os resultados aqui obtidos, utilizando 100 repetições, são muito bons, pois tanto para uma variação da temperatura de 0.05 como de 0.2, consegue-se obter, praticamente sempre, a melhor solução final (com valor 0), dependendo apenas, por vezes, do maior ou menor número de interacções necessário. Ou seja, estes parâmetros surgem como os mais adequados à resolução deste problema definido.

O tempo médio gasto para a resolução do problema, tanto na variação Subtractiva como na Multiplicativa foi, respectivamente, de 20.5 e 34.2 segundos, para 50 e 100 repetições.

5.7.5 PROBLEMA DE NÍVEL DE DIFICULDADE DE RESOLUÇÃO SUPERIOR

5.7.5.1 Definição

Foi definido um problema, cujo nível de dificuldade apresentado é superior, quando comparado com os problemas anteriormente apresentados. O ficheiro com a descrição do problema apresenta-se em anexo (*Anexo 3: Problema nº 3*). As principais características que compõem o problema são:

Características	*Valores*
Número de Dias	5
Número de Horas	10
Número de *Slots*	50
Número de Salas	22
Número de Turmas	50
Número de Docentes	40
Número de Eventos	150
Número de Partes (Aulas)	736

Tabela 26: As principais características do Problema nº 3

A dimensão do espaço de pesquisa deste problema é um número com mais de 3000 zeros:

$$Espaço\_Pesquisa_3 = Aulas^{Slots*Salas} = 736^{50*22} = 736^{1100} = 3.7*10^{3153}$$

Existe ainda um conjunto de propriedades das características, que atribuem uma maior dificuldade a este problema relativamente aos problemas anteriores, como sejam:

- *Todos os eventos encontram-se divididos, por defeito, em 5 partes (aulas), à excepção de 7 eventos, cujo número de partes é 3, o que perfaz as referidas 736 aulas;*
- *Os docentes definidos leccionam entre 1 e 5 eventos cada um;*
- *Salas, Grupos de Alunos e Docentes têm preferências temporais de aceitação de eventos;*
- *Grande parte dos horários (docentes e turmas) estão muito cheios, existindo inclusivamente diversos horários que estão totalmente cheios;*
- *O número de salas e suas disponibilidades não é muito superior às necessidades (a taxa de ocupação final das salas vai ser superior a 75%).*

5.7.5.2 Parâmetros Iniciais e Temperatura

Os valores dos parâmetros iniciais, definidos para a resolução deste problema, são os apresentados na Tabela 27, tendo sido experimentados os dois tipos de diminuição de temperatura

(Multiplicativa e Subtractiva) e para cada uma delas foi decidido tentar resolver o problema com 300 repetições.

Parâmetros	*Multiplicativa*	*Subtractiva*
Número Máximo de Iterações	3000	3000
Número de Repetições	300	300
Temperatura Inicial	10.0	10.0
Diminuição da Temperatura	0.99	0.05
Temperatura Mínima	0.001	0.001

Tabela 27: Valores iniciais dos parâmetros do algoritmo

Os valores atribuídos aos parâmetros resultaram da realização de diversos testes, obtidos através de um processo de tentativa/erro, tendo sido decidido apresentar os resultados de dois dos testes cujos resultados são os mais representativos da tentativa de obtenção da solução do problema.

Os gráficos resultantes da variação da temperatura, segundo os dois tipos de diminuição, apresentam-se nas figuras Figura 42 e Figura 43.

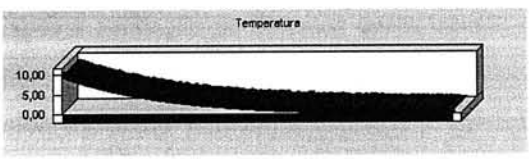


Figura 42: Diminuição Multiplicativa da Temperatura de 0.99

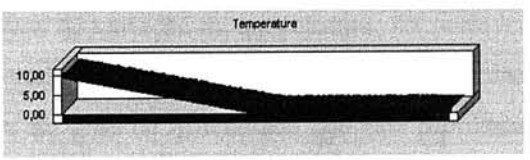


Figura 43: Diminuição Subtractiva da Temperatura de 0.05

A diminuição multiplicativa da temperatura apresenta valores suavizados, ao longo das iterações definidas enquanto a diminuição subtractiva apresenta um decréscimo tipo "rampa" até ao valor mínimo, a partir do qual, a temperatura não se altera.

5.7.5.3 Qualidade da Solução

Apresentam-se nas figuras Figura 44 e Figura 45, os gráficos da evolução da qualidade da solução, utilizando variação multiplicativa e subtractiva da temperatura, respectivamente.

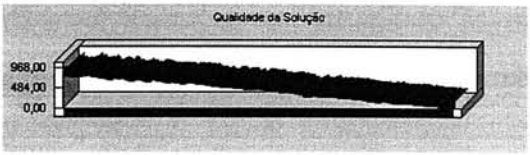


Figura 44: Evolução da Qualidade da Solução obtida, utilizando variação Multiplicativa da temperatura de 0.99 e 300 repetições

Figura 45: Evolução da Qualidade da Solução obtida, utilizando variação Subtractiva da temperatura de 0.05 e 300 repetições

Nos diversos testes realizados, a qualidade da solução para este problema foi, praticamente sempre 1, surgindo, num caso ou outro, o valor 2. Usando variação Multiplicativa da temperatura, a qualidade da solução foi melhorando de uma forma crescente, relativamente constante, como se pode constatar pelo gráfico da Figura 44. Por outro lado, no caso da variação Subtractiva da temperatura, a qualidade da solução foi melhorando de uma forma crescente, mas mais rapidamente, até cerca de metade do número de iterações definido para a resolução do problema, como se pode constatar pelo gráfico da Figura 45.

Após a execução de diversos testes usando os dois tipos de variação da temperatura, foi possível obter o valor 0 na solução final (num teste usando variação Subtractiva). Ou seja, a partir dos diversos testes realizados é possível apercebermo-nos do grau de dificuldade que este problema apresenta, e que mesmo para um algoritmo eficiente muitas vezes não será possível obter uma solução óptima, sem relaxar alguma das restrições do problema.

O tempo médio gasto para a resolução do problema, tanto com variação Multiplicativa como Subtractiva da temperatura, foi de cerca de12 minutos.

5.7.6 RESULTADOS OBTIDOS

Os diversos resultados obtidos são muito satisfatórios e demonstram a eficiência do algoritmo de *Arrefecimento Simulado*, utilizado na resolução dos problemas descritos. Para todos os problemas definidos, o algoritmo utilizado demonstrou extrema robustez na resolução dos mesmos, e conseguiu obter a solução óptima ou uma solução muito próxima da óptima.

5.8 CONCLUSÃO

Neste capítulo foi apresentada a implementação do sistema projectado de geração automática de horários. Descreveu-se a plataforma de desenvolvimento seleccionada, que inclui o sistema operativo e a linguagem de programação escolhidos. Foram apresentadas as potencialidades requeridas ao sistema desenvolvido e a seu âmbito de integração num amplo projecto de

desenvolvimento de um sistema de planeamento e escalonamento universitário: o *UNIPS – University Planning and Scheduling System*. Apresentou-se, a interface gráfica desenvolvida e descreveu-se o estudo computacional efectuado. Finalmente, apresentaram-se os resultados computacionais obtidos na resolução de alguns problemas com uma dimensão variável (desde problemas muito simples até problemas de elevada dimensão e extremamente complexos), utilizando um algoritmo baseado no *Arrefecimento Simulado*.

No capítulo seguinte, são apresentadas as conclusões retiradas do trabalho desenvolvido e apresentado nesta dissertação, e é efectuada uma proposta de continuidade deste projecto, com apresentação dos desenvolvimentos futuros pelos quais pode passar.

Capítulo 6

Conclusões e Desenvolvimentos Futuros

A primeira conclusão a retirar do trabalho que conduziu à elaboração desta dissertação, consiste no facto dos objectivos propostos no início do projecto terem sido satisfatoriamente alcançados.

Neste trabalho foi apresentado o estado da arte referente ao *Problema da Construção de Horários* em universidades, tendo sido analisadas diversas vertentes do problema, entre as quais, os problemas associados de *Calendarização de Exames* e *Distribuição de Serviço Docente*, as diferenças entre horários escolares e universitários, a problemática da construção de horários semi-automáticos vs. automáticos, os diversos tipos de restrições existentes num problema, para finalmente se poder avaliar a qualidade de um horário. Foram ainda analisados alguns sistemas comerciais para resolver o problema, disponíveis no mercado de aplicações do género.

Foram introduzidas algumas técnicas que podem ser usadas no processo de geração de horários, nomeadamente a *Coloração de Grafos*, a *Programação Inteira*, e diversas meta-heurísticas como os *Algoritmos Genéticos*, o *Arrefecimento Simulado* e a *Pesquisa Tabu*. Finalmente, foi efectuada uma breve introdução à *Programação em Lógica* e à *Programação em Lógica com Restrições*, bem como aos *Sistemas Periciais* e aos *Sistemas Interactivos*. Do estudo da aplicação destas técnicas em casos práticos, percebeu-se que o *Arrefecimento Simulado* é uma das mais utilizadas, notando-se também um número crescente de adeptos da *Programação em Lógica com Restrições*.

Apresentou-se a arquitectura base de um sistema de dados, desenvolvida a partir da análise de diversos *standards* de modelação de problemas, tendo sido escolhida, analisada e estendida (a sua sintaxe) uma linguagem simples e completa (*UNILANG*). De seguida foi definida a estrutura de uma base de dados para representação completa de problemas, e cujos dados serão colocados em memória aquando da leitura do ficheiro com o problema a solucionar. Esta arquitectura foi construída com o objectivo de ser uma plataforma flexível, para o desenvolvimento futuro de uma aplicação que permita o estudo mais aprofundado do *Problema da Construção de Horários* e sub-problemas associados, como é o caso da *Calendarização de Exames* e da *Distribuição de Serviço Docente*, possibilitando, também, a aplicação e estudo de outras técnicas de resolução dos problemas.

Foi implementado um sistema de construção de horários de forma automática, tendo sido, para tal, desenvolvida uma interface gráfica amigável, que permite, interactivamente, ao utilizador, uma posterior manutenção dos dados lidos do ficheiro de texto com o problema representado, bem como a manutenção da solução obtida a partir do algoritmo implementado: o *Arrefecimento Simulado*.

A modelo de dados desenvolvido correspondeu totalmente às expectativas iniciais do sistema projectado. A plataforma de desenvolvimento seleccionada revelou-se indicada ao desenvolvimento de raiz de um sistema de geração de horários. As rotinas elaboradas para todos os processos até à obtenção dos horários, mostraram um desempenho muito satisfatório. A interface desenvolvida para todos os processos de manutenção dos dados e informação obtida do processo de geração dos horários, pode ser considerada de grande intuitividade e facilidade de operação para o utilizador.

O algoritmo de geração automática de horários (baseado em *Arrefecimento Simulado*) implementado mostrou um desempenho bastante bom, como demonstram os resultados obtidos. Utilizando este algoritmo foi possível resolver problemas de média e elevada dimensão, obtendo-se soluções de muito boa qualidade, num muito curto intervalo de tempo. A sua extrema rapidez de execução ficou-se a dever bastante à aposta inicial do projecto, de colocar todos os dados importantes do problema em memória.

Como possíveis desenvolvimentos futuros deste trabalho, nomeadamente ao nível da aplicação prática desenvolvida, sugerem-se os seguintes:

- *Estudo e implementação de outros métodos para a resolução automática de diversos problemas, nomeadamente, Horário Turma-Docente, Horário das Disciplinas, Calendarização de Exames, Definição de Ofertas de Disciplinas, Horário de um Aluno, Distribuição de Serviço Docente, Alocação de Vigilantes e Alocação de Salas, entre outros;*
- *Possibilidade de obtenção de análises estatísticas, mais aprofundadas, à informação gerada pelos diversos métodos;*
- *Manutenção da interface gráfica desenvolvida, com melhoria das potencialidades já existentes e implementação de outras, de acordo com as necessidades do utilizador;*
- *Construção de auxílio "on-line" adequado ao contexto, de forma a guiar o utilizador no funcionamento da aplicação;*
- *Desenvolvimento de potencialidades de impressão do sistema;*
- *Construção de rotina para instalação da aplicação desenvolvida;*

O sistema implementado apresenta enormes potencialidades no sentido de uma evolução futura para um sistema mais completo e robusto, e pode tornar-se numa referência de análise e estudo de problemas complexos de construção de horários.

Bibliografia

[Abramson 1991] Abramson, D. "Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms". Management Science, 37(1), 98-113, 1991

[Azevedo e Barahona 1994] Azevedo, Francisco; Barahona, Pedro. "Timetabling in Constraint Logic Programming", In Proceedings of World Congress on Expert Systems", Mexico, 1994

[Bloomfield e McSharry 1979] Bloomfield, S. D.; McSharry, M. M. "Preferential Course Scheduling", Interfaces, vol. 9, nº 4, pp. 24-31, The Institute of Management Sciences, Agosto de 1979

[Burke et al. 1998] Burke, E.; Kingston, J.; Pepper, P. "A Standard Data Format for Timetabling Instances", In Practice and Theory of Automated Timetabling II, LNCS, Vol. 1408, pp. 213-222, 1998

[Carter 1986] Carter, M. W. "A Survey of Practical Applications of Examination Timetabling Algorithms", Operations Research, 34(2), 193-202, 1986

[Carter et al. 1994] Carter, M. W.; Laporte, G.; Chinneck, J. W. "A General Examination Scheduling System", Interfaces, vol. 24, nº 3, pp. 109-120, The Institute of Management Sciences, May-June, 1994

[Carter e Laporte 1996] Carter, M.; Laporte G. "Recent Developments in Practical Examination Timetabling", In Practice and Theory of Automated Timetabling II, LNCS, Springer Verlag, Vol. 1408, pp. 3-21, 1998

[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. Submitted to Computacional Optimization and Applications, 1992

[Cooper e Kingston 1993a] Cooper, T.; Kingston, J. "The Solution of Real Instances of the Timetabling Problems", The Computer Journal, Vol. 36, pp. 645-653, 1993

[Cooper e Kingston 1993b] Cooper, T.; Kingston, J. "A Program for Constructing High School Timetables", in Edmund Burke, Dave Corne, Ben Paetcher and Peter Ross, First International Conference on the Practice and Theory of Automated Timetabling – PATAT95, Napier University, Edinburgh, UK, 1995

[Cooper e Kingston 1996] Cooper, T.; Kingston, J. "The Complexity of Timetable Construction Problems", 1996

[Costa 1994] Costa, D. "A Tabu Search Algorithm for Computing an Operational Timetable". European Journal of Operational Research, 76, pp. 98-110, 1994

[Cox e Walker 1993] Cox, Kevin; Walker, David. "User interface Design", Practice Hall, Second Edition, 1993

[Cumming e Paechter 1995] Cumming, A.; Paechter, B. "Seminar: Standard Timetabling Data Format", in 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

[Davis 1991] Davis, L. "Handbook of Genetic Algorithms", Van Nostrand Reinhold, NY

[Dias 1999] Dias, Ricardo Osório. "Sistemas de Distribuição de Serviço Docente em Universidades", Monografia em Informática de Gestão, Universidade Fernando Pessoa, Porto, 1999

[Falcão 1990] Falcão, P.; Pereira, M.; Ribeiro, R.; Barahona, P. "Another Timetabling Approach", Universidade Nova de Lisboa, 1990

[Frangouli et al. 1995] Frangouli, Harikleia; Harmandas Vassillis; Stamatopoulos, Panagiotis. "UTSE: Construction of Optimum Timetables of University Courses - A CLP Based Approach", Third International Conference on the Practical Applications of Prolog, PAP'95, Paris, pp.225-243, Abril 1995

[Glover e Laguna 1993] Glover, F.; Laguna, M. "Tabu Search". In Reeves, C. R. (Ed.), Modern Heuristic Techniques for Combinatorial Problems. Scientific Publications, Oxford, 1993

[Gomes 1997] Gomes, Nuno Filipe. "O Problema do Sequenciamento de Tarefas em Programação Lógica por Restrições". Tese de Mestrado, Faculdade de Engenharia da Universidade do Porto, Departamento de Engenharia Electrotécnica e de Computadores, 1997

[Gorry e Morton1971] Gorry, G. M.; Morton, Scott. "A Framework for Management Information Systems", Sloan Management Review, 1971

[Greenberg 1999] Greenberg, Harvey J. "Mathematical Programming Glossary", http://www.cudenver.edu/~hgreenbe/glossary/glossary.html

[Grefory 1991] Grefory, J. E. "Foundations of Genetic Algorithms", San Mateo: Morgan Kaufmann, 1991

[Gudes et al. 1990] Ehud, Gudes; Kuflik, Tsvi; Meisels, Amnon. "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", in Edmund Burke, Dave Corne, Ben Paetcher and Peter Ross, First International Conference on the Practice and Theory of Automated Timetabling: pp. 393-408, Edinburgh, 1995

[Holland 1975] Holland, John H. "Adaptation in Natural and Artificial Systems", Ann Arbor: The University of Michigan Press, 1975

[Howe 1999] Howe, Denis. "FOLDOC – Free On-line Dictionary of Computing", http://wombat.doc.ic.ac.uk/foldoc/index.html

[Jaffar e Lassez 1987a] Jaffar, Joxan; Lazzez, Jean-Louis. "Constraint Logic Programming", In Proceedings 14th ACM Symposium on Principles of Programming Languages, pp. 111-119, Munich. ACM, 1987

[Jaffar e Lassez 1994b] Jaffar, Joxan; Lazzez, Jean-Louis. "Constraint Logic Programming – A Survey", In Journal of Logic Programming, 1994

[Johnson et al. 1991] Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; Schevon, C. "Optimization by Simulated Annealing: an Experimental Evaluation; part II, Graph Coloring and Number Partitioning", Operations Research, 39(3), pp. 378-406, 1991

[Kang e White 1992] Kang, L.; White, G. M. "A Logic Approach to the Resolution of Constraints in Timetabling", European Journal of Operational Research, 61, pp. 306-317, 1992.

[Karaboga e Pham 1998] Karaboga, D.; Pham, D. "Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks", Springer-Verlag, London, 1998

[Kingston 1999a] Kingston, Jeffrey H. "Modelling timetabling problems with STTL", Basser Department of Computer Science, The University of Sydney 2006, Australia, 1999

[Kingston 1999b] Kingston, Jeffrey H. "A User's Guide to the STTL Timetabling Language – Version 1.0", 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, 220 (4598), pp. 671-680, 1983

[Kowalski 1979] Kowalski, R. "Logic for problem solving", New York, North-Holland, 1979

[Laudon e Laudon 1997] Laudon, Kenneth C.; Laudon, Jane Price. "Fundamentals of Management Science", 6th edition, Burr Ridge, 1994

[Leighton 1979] Leighton, F. T. "A Graph Colouring for Large Scheduling Problems", J. Res. Natl. Bur. Standards, 84, 489-506, 1979

[Liatsos 1995] Liatsos, Vassilios. "An Environment for a Resource Allocation Problem in CLP", Imperial College of Science, Technology and Medicine (University of London) Department of Computing, Setembro de 1995

[Mehta 1981] Mehta, N. "The Application of a Graph Coloring Method to an Examination Scheduling Problem", Interfaces 11:57 - 64, 1981

[Melicio et al. 1998] Melicio, F.; Caldeira, J.P.; Rosa, A.C. "Timetabling implementation aspects by Simulated Annealing ", IEEE- Systems Science and Systems Engineering, Jifa Gu (Edt), pp:553-557, 1998

[Newall 1999] Newall, J. P. "Hybrid Methods for Automated Timetabling", PhD Thesis, University of Nottingham, 1999

[Oliveira 1984] Oliveira, Eugénio da Costa. "Engenharia do Conhecimento: Sistemas Periciais e Prolog", Tese de Doutoramento, Universidade Nova de Lisboa, 1984

[Oliveira J. 1999] Oliveira, José Fernando. "Optimização Combinatória e Técnicas Heurísticas – Meta-heurísticas e Aplicações", Instituto Superior de Estudos Empresariais, 1999

[Osman 1995] Osman, I.H. "An introduction to meta-heuristics", In Lawrence M, Wilsdon C (eds). Operational Research Tutorial 1995. Operational Research Society: Birmingham, pp. 92-122, 1995

[Osman e Kelly 1996] Osman, I. H.; Kelly, J. P. "Meta-Heuristics: An Overview", in Metaheuristics. Theory and Applications, ed. I.H. Osman and J.P. Kelly, (Kluwer, Boston), 1996

[Pyle 1995] Pyle, David R. "Graphical User interface design and Evaluation: A Practical Guide", Practice-Hall, 1995

[Reis 1995a] Reis, Luís Paulo. "Sistema Integrado de Detecção e Seguimento de Contornos", Tese de Mestrado, Faculdade de Engenharia do Porto, Setembro de 1995

[Reis 1996b] Reis, Luís Paulo. "Programação Lógica com Restrições: Sistemas e Técnicas", Relatório Técnico, Núcleo de Inteligência Artificial Distribuída e Robótica, Faculdade de Engenharia da Universidade do Porto, 1996

[Reis et al 1999] Reis, Luís Paulo; Teixeira, Paulo; Oliveira, Eugénio da Costa. "Examination Timetabling using Constraint Logic Programming", ECP'99, 5th European Conference on Planning, Durham, U.K., Setembro 1999

[Reis e Oliveira 1999a] Reis, Luís Paulo; Oliveira, Eugénio da Costa. "Constraint Logic Programming Approach to Examination Scheduling", AICS'99, Artificial Intelligence and Cognitive Science Conference, Cork, Ireland, Setembro 1999

[Reis e Oliveira 1999b] Reis, Luís Paulo; Oliveira, Eugénio da Costa. "Constraint Logic Programming using Set Variables for Solving Timetabling Problems", INAP'99, 12th International Conf. on the Applications of Prolog, Tokyo, Japão, Setembro 1999

[Reis e Oliveira 2000a] Reis, Luís Paulo; Oliveira, Eugénio da Costa. "A Language for Specifying Complete Timetabling Problems", Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling – PATAT2000, Konstanz, Alemanha, Agosto de 2000

[Reis e Oliveira 2000b] Reis, Luís Paulo; Oliveira, Eugénio da Costa. "Examination Timetabling using Set Variables", Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling – PATAT2000, Konstanz, Alemanha, Agosto de 2000

[Romero 1982] Romero, B. P. "Examination Scheduling in a Large Engineering School: A Computer-Assisted Participative Procedure", Interfaces, vol. 12 (2), pp. 17-23, The Institute of Management Sciences, Industrial Organization Department, Industrial Engineering Technical School, Madrid, Spain, April 1982

[Schaerf e Schaerf 1995] Schaerf, A.; Schaerf, M. "Local Search Techniques for High School Timetabling" in 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

[Stamatopoulos et al. 1998] Stamatopoulos, Panagiotis; Viglas, Efstratios; Karaboyas, Serafeim. "Nearly Optimum Timetable Construction Through CLP and Intelligent Search", International Journal on Artificial Intelligence Tools, Vol. 7, No. 4, pp. 415-442, 1998

[Tripathy 1984] Tripathy, A. "School Timetabling – A Case in Large Binary Integer Linear Programming", In Management Science, 30: pp. 1473-1489, 1984

[Turban e Meredith 1994] Turban, Efrain; Meredith, J. "Fundamentals of Management Science", 6th edition, Burr Ridge, 1994

[Uhlemann et al. 1969] Uhlemann, K. H.; Schollkopf, K. H.; Knauer, B. A. "Untersuchungen zum studenten-planproblem", Elecktron, Datenverarbeitung, 11, pp. 199-131, 1969

[Welsh e Powell 1967] Welsh, D.; Powell, M. B. "An Upper Bound to the Chromatic Number of a Graph and its Application to Timetabling Problems", The Computer Journal, 10, pp. 85-86, 1967

[Werra 1985] Werra, D. "An Introduction to Timetabling", European Journal of Operational Research, 19, pp. 151-162, 1985

[White e Chan 1979] White, G. M.; Chan, P. W. "Towards the Construction of Optimal Examination Schedules", INFOR, vol. 17, nº 3, pp. 219-229, 1979

[Winston 1991] Winston, W. L. "Operations Research: Applications and Algorithms", 2ª edição, PWS-Kent, Boston, Massachusetts, 1991

[Wren 1996] Wren, Anthony. "Scheduling, Timetabling and Rostering – a Special Relationship ?", in Selected papers from the 1st International Conference The Practice and Theory of Automated Timetabling – PATAT95, Lecture Notes in Computer Science, pp 46-75. Springer-Verlag, Berlim.

[Waltz 1972] Waltz, D. "Generating Semantic Descriptions from Drawings of Scenes with Shadows". Technical Report AI271, MIT, Massachusetts, 1972

[Yoshikawa et al. 1994] Yoshikawa, M.; Kaneko, K.; Nomura, Y.; Watanabe, M. "A Constraint-Based Approach to High-School Timetabling Problems: A Case Study", in Proceedings of the 12th National Conference on Artificial Intelligence AAAI'94, pp. 111-116, 1994

Anexos

ANEXO 1: PROBLEMA Nº 1 – NÍVEL SIMPLES

this is year 1999/2000
departments are CIE LET
scientific_areas are INF MAT COM DIR
degrees are EINF INFG DIRE CCOM
department CIE has scientific_areas INF MAT
department LET has scientific_areas COM DIR
department CIE has degrees EINF INFG
department LET has degrees DIRE CCOM
periods are sem1 sem2
slots are s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15
days are mon tue wed
times are t1 t2 t3 t4 t5
time_periods are morni lunch after
rooms are ROOM1 ROOM2 ROOM3 LAB1 ANF1
room_types are NORMAL LAB ANF
events are e1 e2 e3 e4 e5 e6 e7 e8 e9 e10
group_students are gst1 gst2 gst3 gst4
teachers are LPR PMF PNS ECO APR
group_teachers are NIADR1 NIADR2
period sem1 contains weeks 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
period sem2 contains weeks 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
period sem1 sem2 begins on day mon
period sem1 sem2 ends on day fri
period sem1 begins on date xx/xx/1999
period sem2 begins on date xx/xx/2000
slot s1 is on day mon at time t1
slot s2 is on day mon at time t2
slot s3 is on day mon at time t3
slot s4 is on day mon at time t4
slot s5 is on day mon at time t5
slot s6 is on day tue at time t1
slot s7 is on day tue at time t2
slot s8 is on day tue at time t3
slot s9 is on day tue at time t4
slot s10 is on day tue at time t5
slot s11 is on day wed at time t1
slot s12 is on day wed at time t2
slot s13 is on day wed at time t3
slot s14 is on day wed at time t4
slot s15 is on day wed at time t5
slots default have capacity 500 seats
slots default have capacity 12 events
room default holds 50 students
room LAB1 holds 30 students
room ANF1 holds 200 students
room default holds 1 events
room default cannot have double_bookings
room default excludes slots default preference 0
rooms ROOM1 ROOM2 ANF1 exclude slots s1 s2 s3
rooms ROOM1 ROOM2 ROOM3 exclude slots s4 s5 s6 preference 2
rooms ROOM1 ROOM2 ANF1 LAB1 exclude slots s8 s9 s10 preference 3
rooms ROOM1 ROOM3 exclude slots s7 s11 s12 preference 3

group_students default has 15 students
group_students gst1 gst4 has 50 students
group_students gst1 has students st1 st2 st3 st4 st5
group_students gst2 has students st6 st7 st8
group_students gst3 has students st9
group_students gst4 has students st10
group_students default cannot have double_bookings
group_students gst1 gst4 excludes slots s1 s2 s3 s4
group_students gst1 gst2 gst3 gst4 excludes slots s6 s7 s8 s9 s10 preference 1
group_students gst1 gst3 excludes slots s11 preference 2
group_students gst1 gst2 excludes slots s12 preference 3
teacher LPR teaches events e1 e6
teacher PMF teaches events e2 e7
teacher PNS teaches events e3 e8
teacher ECO teaches events e4 e9
teacher APR teaches events e5 e10
teacher default cannot have double_bookings
teacher LPR ECO excludes slots s1 s2 s3 s4
teacher LPR excludes slots s7 s8 preference 2
teacher ECO excludes slots s9 s10 preference 2
teacher ECO excludes slots s11 preference 3
teacher PMF PNS excludes slots s1 s4 s7
teacher PMF PNS excludes slots s11 s12 preference 1
teacher APR excludes slots s11 s12
teacher APR excludes slots s4 s5 s6 s7 s8 preference 1
group_teacher NIADR1 has teachers LPR ECO APR
group_teacher NIADR2 has teachers PMF PNS
event default lasts 1 slots
event e1 e2 e3 e4 e5 has group_students gst1 gst2
event e6 e7 e8 e9 e10 has group_students gst3 gst4
event default has 2 parts of duration 1 1
event e5 e6 has 3 parts of duration 1 1 1
scientific_area INF has teachers default
scientific_area INF has events default
degree EINF has group_students default
degree EINF has events default

ANEXO 2: PROBLEMA Nº 2 – NÍVEL MÉDIO

this is year 1999/2000
departments are CIE LET
scientific_areas are INF MAT COM DIR
degrees are EINF INFG DIRE CCOM
department CIE has scientific_areas INF MAT
department LET has scientific_areas COM DIR
department CIE has degrees EINF INFG
department LET has degrees DIRE CCOM
periods are sem1 sem2
slots are s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25
slots are s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48
s49 s50
days are mon tue wed thu fri
times are t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
time_periods are morni lunch after
rooms are ROOM1 ROOM2 ROOM3 ROOM4 ROOM5 ROOM6 LAB1 LAB2 LAB3 ANF1 ANF2 ANF3
room_types are NORMAL LAB ANF
events are e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 e11 e12 e13 e14 e15 e16 e17 e18 e19 e20
events are e21 e22 e23 e24 e25 e26 e27 e28 e29 e30 e31 e32 e33 e34 e35 e36 e37 e38 e39 e40
students are st1 st2 st3 st4 st5 st6 st7 st8 st9 st10 st11 st12 st13 st14 st15 st16 st17 st18 st19 st20
group_students are gst1 gst2 gst3 gst4 gst5 gst6 gst7 gst8 gst9 gst10
group_students are gst11 gst12 gst13 gst14 gst15 gst16 gst17 gst18 gst19 gst20
teachers are LPR PMF PNS ECO APR JCM AMP MCN MBM JMF
group_teachers are NIADR1 NIADR2
period sem1 contains weeks 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
period sem2 contains weeks 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
period sem1 sem2 begins on day mon
period sem1 sem2 ends on day fri
period sem1 begins on date xx/xx/1999
period sem2 begins on date xx/xx/2000
slot s1 is on day mon at time t1
slot s2 is on day mon at time t2
slot s3 is on day mon at time t3
slot s4 is on day mon at time t4
slot s5 is on day mon at time t5
slot s6 is on day mon at time t6
slot s7 is on day mon at time t7
slot s8 is on day mon at time t8
slot s9 is on day mon at time t9
slot s10 is on day mon at time t10
slot s11 is on day tue at time t1
slot s12 is on day tue at time t2
slot s13 is on day tue at time t3
slot s14 is on day tue at time t4
slot s15 is on day tue at time t5
slot s16 is on day tue at time t6
slot s17 is on day tue at time t7
slot s18 is on day tue at time t8
slot s19 is on day tue at time t9
slot s20 is on day tue at time t10
slot s21 is on day wed at time t1
slot s22 is on day wed at time t2
slot s23 is on day wed at time t3
slot s24 is on day wed at time t4
slot s25 is on day wed at time t5
slot s26 is on day wed at time t6
slot s27 is on day wed at time t7
slot s28 is on day wed at time t8

slot s29 is on day wed at time t9
slot s30 is on day wed at time t10
slot s31 is on day thu at time t1
slot s32 is on day thu at time t2
slot s33 is on day thu at time t3
slot s34 is on day thu at time t4
slot s35 is on day thu at time t5
slot s36 is on day thu at time t6
slot s37 is on day thu at time t7
slot s38 is on day thu at time t8
slot s39 is on day thu at time t9
slot s40 is on day thu at time t10
slot s41 is on day fri at time t1
slot s42 is on day fri at time t2
slot s43 is on day fri at time t3
slot s44 is on day fri at time t4
slot s45 is on day fri at time t5
slot s46 is on day fri at time t6
slot s47 is on day fri at time t7
slot s48 is on day fri at time t8
slot s49 is on day fri at time t9
slot s50 is on day fri at time t10
time_periods morni contains slots s1 s2 s3 s4 s5 s11 s12 s13 s14 s15 s21 s22 s23 s24 s25 s31 s32 s33 s34 s35 s41 s42 s43 s44 s45
time_periods lunch contains slots s5 s6 s15 s16 s25 s26 s35 s36 s45 s46
time_periods after contains slots s6 s7 s8 s9 s10 s16 s16 s18 s19 s20 s26 s27 s28 s29 s30 s36 s37 s38 s39 s40 s46 s47 s48 s49 s50
slots default have capacity 500 seats
slots default have capacity 12 events
room default holds 50 students
room LAB1 LAB2 LAB3 holds 30 students
room ANF1 holds 200 students
room ANF2 ANF3 holds 100 students
room default holds 1 events
room default cannot have double_bookings
room_type NORMAL has rooms ROOM1 ROOM2 ROOM3 ROOM4 ROOM5 ROOM6
room_type ANF has rooms ANF1 ANF2 ANF3
room_type LAB has rooms LAB1 LAB2 LAB3
room LAB1 close to room LAB2 preference 3
room ROOM1 close to room ROOM2 preference 3
rooms ROOM1 ANF1 exclude slots s31 s32 s33 s34 s35
rooms ROOM1 ROOM2 ROOM3 ROOM4 exclude slots s1 s2 s11 s12 s21 s22
rooms ROOM5 ROOM6 ANF1 ANF2 exclude slots s10 s20 s30 s40 s50
rooms ROOM1 ROOM2 ANF1 LAB1 exclude slots s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 preference 3
rooms ROOM1 ROOM3 exclude slots s36 s37 s38 s39 s40 preference 3
group_students default has 15 students
group_students gst1 gst4 has 50 students
group_students gst1 has students st1 st2 st3 st4 st5
group_students gst2 has students st6 st7 st8 st9 st10
group_students gst3 has students st11 st12 st13 st14 st15
group_students gst4 has students st16 st17 st18 st19 st20
group_students default cannot have double_bookings
group_students gst1 gst4 excludes slots s1 s2 s3 s4 s5
group_students gst1 gst2 gst3 gst4 gst5 excludes slots s6 s7 s8 s9 s10 preference 1
group_students gst1 gst3 excludes slots s16 s17 s18 s19 s20 preference 2
group_students gst1 gst2 excludes slots s26 s27 s28 s29 s30 preference 3
teacher LPR teaches events e1 e11 e21 e31
teacher PMF teaches events e2 e12 e22 e32
teacher PNS teaches events e3 e13 e23 e33
teacher ECO teaches events e4 e14 e24 e34
teacher APR teaches events e5 e15 e25 e35
teacher JCM teaches events e6 e16 e26 e36

teacher AMP teaches events e7 e17 e27 e37
teacher MCN teaches events e8 e18 e28 e38
teacher MBM teaches events e9 e19 e29 e39
teacher JMF teaches events e10 e20 e30 e40
teacher default cannot have double_bookings
teacher LPR excludes slots s1 s2 s3 s4 s5 s11 s12 s13 s14 s15
teacher ECO excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
teacher LPR excludes slots s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 preference 2
teacher ECO excludes slots s31 s32 s33 s34 s35 preference 2
teacher ECO excludes slots s36 s37 s38 s39 s40 preference 3
teacher PMF PNS excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
teacher PMF excludes slots s41 s42 s43 s44 s45
teacher PMF PNS excludes slots s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 preference 1
teacher APR JCM AMP excludes slots s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40
teacher APR JCM excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 preference 1
teacher MCN MBM JMF excludes slots s1 s2 s11 s12 s21 s22 s31 s32 s41 s42
group_teacher NIADR1 has teachers LPR ECO APR MBM MCN JMF
group_teacher NIADR2 has teachers PMF PNS JCM AMP
event default lasts 1 slots
event e1 e2 e3 e4 e5 has group_students gst1 gst2 gst3
event e6 e7 e8 e9 e10 has group_students gst4 gst5 gst6
event e11 e12 e13 e14 e15 has group_students gst7 gst8 gst9
event e16 e17 e18 e19 e20 has group_students gst10 gst11 gst12
event e21 e22 e23 e24 e25 has group_students gst13 gst14 gst15
event e26 e27 e28 e29 e30 has group_students gst16 gst17
event e31 e32 e33 e34 e35 has group_students gst18 gst19
event e36 e37 e38 e39 e40 has group_students gst20
event default has 8 parts of duration 1 1 1 1 1 1 1 1
event e7 e8 e34 e35 has 5 parts of duration 1 1 1 1 1
event e5 e6 e17 e18 has 3 parts of duration 1 1 1
scientific_area INF has teachers default
scientific_area INF has events default
degree EINF has group_students default
degree EINF has events default

ANEXO 3: PROBLEMA Nº 3 – NÍVEL SUPERIOR

this is year 1999/2000
departments are CIE LET
scientific_areas are INF MAT COM DIR
degrees are EINF INFG DIRE CCOM
department CIE has scientific_areas INF MAT
department LET has scientific_areas COM DIR
department CIE has degrees EINF INFG
department LET has degrees DIRE CCOM
periods are sem1 sem2
slots are s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25
slots are s26 s27 s28 s29 s30 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50
days are mon tue wed thu fri
times are t1 t2 t3 t4 t5 t6 t7 t8 t9 t10
time_periods are morni lunch after
rooms are ROOM1 ROOM2 ROOM3 ROOM4 ROOM5 ROOM6 ROOM7 ROOM8 ROOM9 ROOM10 ROOM11 ROOM12 ROOM13 ROOM14 ROOM15 ROOM16
rooms are LAB1 LAB2 LAB3 ANF1 ANF2 ANF3
room_types are NORMAL LAB ANF
events are e1 e2 e3 e4 e5 e6 e7 e8 e9 e10 e11 e12 e13 e14 e15 e16 e17 e18 e19 e20
events are e21 e22 e23 e24 e25 e26 e27 e28 e29 e30 e31 e32 e33 e34 e35 e36 e37 e38 e39 e40
events are e41 e42 e43 e44 e45 e46 e47 e48 e49 e50 e51 e52 e53 e54 e55 e56 e57 e58 e59 e60
events are e61 e62 e63 e64 e65 e66 e67 e68 e69 e70 e71 e72 e73 e74 e75 e76 e77 e78 e79 e80
events are e81 e82 e83 e84 e85 e86 e87 e88 e89 e90 e91 e92 e93 e94 e95 e96 e97 e98 e99 e100
events are e101 e102 e103 e104 e105 e106 e107 e108 e109 e110 e111 e112 e113 e114 e115 e116 e117 e118 e119 e120
events are e121 e122 e123 e124 e125 e126 e127 e128 e129 e130 e131 e132 e133 e134 e135 e136 e137 e138 e139 e140
events are e141 e142 e143 e144 e145 e146 e147 e148 e149 e150
students are st1 st2 st3 st4 st5 st6 st7 st8 st9 st10 st11 st12 st13 st14 st15 st16 st17 st18 st19 st20
group_students are gst1 gst2 gst3 gst4 gst5 gst6 gst7 gst8 gst9 gst10
group_students are gst11 gst12 gst13 gst14 gst15 gst16 gst17 gst18 gst19 gst20
group_students are gst21 gst22 gst23 gst24 gst25 gst26 gst27 gst28 gst29 gst30
group_students are gst31 gst32 gst33 gst34 gst35 gst36 gst37 gst38 gst39 gst40
group_students are gst41 gst42 gst43 gst44 gst45 gst46 gst47 gst48 gst49 gst50
teachers are LPR PMF PNS ECO APR JCM AMP MCN MBM JMF TE11 TE12 TE13 TE14 TE15 TE16 TE17 TE18 TE19 TE20
teachers are TE21 TE22 TE23 TE24 TE25 TE26 TE27 TE28 TE29 TE30 TE31 TE32 TE33 TE34 TE35 TE36 TE37 TE38 TE39 TE40
group_teachers are NIADR1 NIADR2
period sem1 contains weeks 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
period sem2 contains weeks 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
period sem1 sem2 begins on day mon
period sem1 sem2 ends on day fri
period sem1 begins on date xx/xx/1999
period sem2 begins on date xx/xx/2000
slot s1 is on day mon at time t1
slot s2 is on day mon at time t2
slot s3 is on day mon at time t3
slot s4 is on day mon at time t4
slot s5 is on day mon at time t5
slot s6 is on day mon at time t6
slot s7 is on day mon at time t7
slot s8 is on day mon at time t8
slot s9 is on day mon at time t9
slot s10 is on day mon at time t10
slot s11 is on day tue at time t1
slot s12 is on day tue at time t2

slot s13 is on day tue at time t3
slot s14 is on day tue at time t4
slot s15 is on day tue at time t5
slot s16 is on day tue at time t6
slot s17 is on day tue at time t7
slot s18 is on day tue at time t8
slot s19 is on day tue at time t9
slot s20 is on day tue at time t10
slot s21 is on day wed at time t1
slot s22 is on day wed at time t2
slot s23 is on day wed at time t3
slot s24 is on day wed at time t4
slot s25 is on day wed at time t5
slot s26 is on day wed at time t6
slot s27 is on day wed at time t7
slot s28 is on day wed at time t8
slot s29 is on day wed at time t9
slot s30 is on day wed at time t10
slot s31 is on day thu at time t1
slot s32 is on day thu at time t2
slot s33 is on day thu at time t3
slot s34 is on day thu at time t4
slot s35 is on day thu at time t5
slot s36 is on day thu at time t6
slot s37 is on day thu at time t7
slot s38 is on day thu at time t8
slot s39 is on day thu at time t9
slot s40 is on day thu at time t10
slot s41 is on day fri at time t1
slot s42 is on day fri at time t2
slot s43 is on day fri at time t3
slot s44 is on day fri at time t4
slot s45 is on day fri at time t5
slot s46 is on day fri at time t6
slot s47 is on day fri at time t7
slot s48 is on day fri at time t8
slot s49 is on day fri at time t9
slot s50 is on day fri at time t10
time_periods morni contains slots s1 s2 s3 s4 s5 s11 s12 s13 s14 s15 s21 s22 s23 s24 s25 s31 s32 s33 s34 s35 s41 s42 s43 s44 s45
time_periods lunch contains slots s5 s6 s15 s16 s25 s26 s35 s36 s45 s46
time_periods after contains slots s6 s7 s8 s9 s10 s16 s16 s18 s19 s20 s26 s27 s28 s29 s30 s36 s37 s38 s39 s40 s46 s47 s48 s49 s50
slots default have capacity 500 seats
slots default have capacity 12 events
room default holds 50 students
room LAB1 LAB2 LAB3 holds 30 students
room ANF1 holds 200 students
room ANF2 ANF3 holds 100 students
room default holds 1 events
room default cannot have double_bookings
room_type NORMAL has rooms ROOM1 ROOM2 ROOM3 ROOM4 ROOM5 ROOM6
room_type ANF has rooms ANF1 ANF2 ANF3
room_type LAB has rooms LAB1 LAB2 LAB3
room LAB1 close to room LAB2 preference 3
room ROOM1 close to room ROOM2 preference 3
rooms ROOM1 ROOM2 ROOM3 ROOM4 ROOM5 ROOM6 exclude slots s31 s32 s33 s34 s35
rooms ROOM1 ROOM2 ROOM3 ROOM4 exclude slots s1 s2 s11 s12 s21 s22
rooms ROOM6 ROOM7 ROOM13 ROOM14 exclude slots s6 s7 s8 s9 s10
rooms ROOM5 ROOM6 ROOM7 ROOM8 ROOM9 ROOM10 ROOM11 ROOM12 ROOM13 ROOM14 ANF1 ANF2 exclude slots s10 s20 s30 s40 s50
rooms ROOM1 ROOM2 ANF1 LAB1 exclude slots s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 preference 3

rooms ROOM1 ROOM3 exclude slots s36 s37 s38 s39 s40 preference 3
group_students default has 15 students
group_students gst1 gst4 has 50 students
group_students gst1 has students st1 st2 st3 st4 st5
group_students gst2 has students st6 st7 st8 st9 st10
group_students gst3 has students st11 st12 st13 st14 st15
group_students gst4 has students st16 st17 st18 st19 st20
group_students default cannot have double_bookings
group_students gst1 gst10 gst11 gst16 gst17 gst21 gst22 gst36 gst37 gst38 excludes slots s1 s2 s3 s4 s5
group_students gst11 gst14 gst20 gst21 gst26 gst27 gst41 gst42 gst43 gst44 gst45 excludes slots s11 s12 s13 s14 s15
group_students gst1 gst2 gst3 gst4 gst5 excludes slots s6 s7 s8 s9 s10 preference 1
group_students gst1 gst3 excludes slots s16 s17 s18 s19 s20 preference 2
group_students gst1 gst2 excludes slots s26 s27 s28 s29 s30 preference 3
teacher LPR teaches events e1 e11 e21 e31 e141
teacher PMF teaches events e2 e12 e22 e32 e142
teacher PNS teaches events e3 e13 e23 e33 e143
teacher ECO teaches events e4 e14 e24 e34 e138
teacher APR teaches events e5 e15 e25 e35
teacher JCM teaches events e6 e16 e26 e36
teacher AMP teaches events e7 e17 e27 e37
teacher MCN teaches events e8 e18 e28 e38
teacher MBM teaches events e9 e19 e29 e39
teacher JMF teaches events e10 e20 e30 e40
teacher TE11 teaches events e41 e42 e45 e46
teacher TE12 teaches events e43 e44 e47 e48
teacher TE13 teaches events e49 e50 e51 e52
teacher TE14 teaches events e53 e54 e55 e56
teacher TE15 teaches events e57 e58 e59 e60
teacher TE16 teaches events e61 e71 e81 e91
teacher TE17 teaches events e62 e72 e82 e92
teacher TE18 teaches events e63 e73 e83 e93
teacher TE19 teaches events e64 e74 e84 e94
teacher TE20 teaches events e65 e75 e85 e95
teacher TE21 teaches events e66 e76 e86 e96
teacher TE22 teaches events e67 e77 e87 e97
teacher TE23 teaches events e68 e78 e88 e98
teacher TE24 teaches events e69 e79 e89 e99
teacher TE25 teaches events e100 e101 e102 e103
teacher TE26 teaches events e104 e105 e106 e107
teacher TE27 teaches events e108 e109 e110
teacher TE28 teaches events e111 e112 e113 e114
teacher TE29 teaches events e115 e116 e117 e118
teacher TE30 teaches events e119 e120 e121 e122
teacher TE31 teaches events e123 e124 e125 e126
teacher TE32 teaches events e127 e128 e129 e130
teacher TE33 teaches events e131 e132 e133 e134
teacher TE34 teaches events e135 e136 e137
teacher TE35 teaches events e139 e140
teacher TE36 teaches events e144
teacher TE37 teaches events e147 e148
teacher TE38 teaches events e145 e146
teacher TE39 teaches events e149 e150
teacher TE40 teaches events e70 e80 e90
teacher default cannot have double_bookings
teacher LPR excludes slots s1 s2 s3 s4 s5 s11 s12 s13 s14 s15
teacher ECO excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
teacher LPR excludes slots s31 s32 s33 s34 s35 s36 s37 s38 s39 s40 preference 2
teacher ECO excludes slots s31 s32 s33 s34 s35 preference 2
teacher ECO excludes slots s36 s37 s38 s39 s40 preference 3
teacher PMF PNS excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
teacher PMF excludes slots s41 s42 s43 s44 s45

teacher PMF PNS excludes slots s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 preference 1
teacher APR JCM AMP excludes slots s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 s31 s32 s33 s34 s35 s36 s37 s38 s39 s40
teacher APR JCM excludes slots s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14 s15 s16 s17 s18 s19 s20 preference 1
teacher MCN MBM JMF excludes slots s1 s2 s11 s12 s21 s22 s31 s32 s41 s42
group_teacher NIADR1 has teachers LPR ECO APR MBM MCN JMF
group_teacher NIADR2 has teachers PMF PNS JCM AMP
event default lasts 1 slots
event e1 e2 e3 e4 has group_students gst1 gst2 gst3 gst4 gst5
event e5 e30 has group_students gst1 gst2 gst3 gst5
event e6 e7 e8 e9 e10 has group_students gst4 gst5 gst6 gst7 gst8
event e11 e12 e13 e14 e15 has group_students gst7 gst8 gst9
event e16 e17 e18 e19 e20 has group_students gst10 gst11 gst12
event e21 e22 e23 has group_students gst13 gst14 gst15
event e24 e25 has group_students gst13 gst15
event e26 e27 e28 e29 has group_students gst16 gst17
event e31 e32 e33 e34 e35 has group_students gst18 gst19
event e36 e37 e38 e39 e40 has group_students gst20 gst21 gst22
event e41 e42 e43 e44 e45 has group_students gst23 gst24
event e46 e47 e48 e49 e50 has group_students gst25 gst26
event e51 e52 e53 e54 e55 has group_students gst27 gst28
event e56 e57 e58 e59 e60 has group_students gst29 gst30
event e61 e62 e63 e64 e65 has group_students gst31 gst32
event e66 e67 e68 e69 e70 has group_students gst33 gst34
event e71 e72 e73 e74 e75 has group_students gst35 gst36
event e76 e77 e78 e79 e80 has group_students gst37 gst38
event e81 e82 e83 e84 e85 has group_students gst39 gst40
event e86 e87 e88 e89 e90 has group_students gst41 gst42
event e91 e92 e93 e94 e95 has group_students gst43 gst44
event e96 e97 e98 e99 e100 has group_students gst45 gst46
event e101 e102 e103 e104 e105 has group_students gst47
event e106 e107 e108 e109 e110 has group_students gst48
event e111 e112 e113 e114 e115 has group_students gst49
event e116 e117 e118 e119 e120 has group_students gst50
event e121 e122 e123 e124 e125 has group_students gst12
event e126 e127 e128 e129 e130 has group_students gst13
event e131 e132 e133 e134 e135 has group_students gst14
event e136 e137 e138 e139 e140 has group_students gst15
event e141 e142 e143 e144 e145 has group_students gst16
event e146 e147 e148 e149 e150 has group_students gst17
event default has 5 parts of duration 1 1 1 1 1
event e1 e2 e3 e4 e5 e11 e12 e13 e14 e15 has 4 parts of duration 1 1 1 1
event e29 e30 e31 e32 e33 e34 e35 has 3 parts of duration 1 1 1
scientific_area INF has teachers default
scientific_area INF has events default
degree EINF has group_students default
degree EINF has events default