FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Algoritmos Genéticos Distribuídos e Suas Aplicações

João Filipe dos Santos Teixeira Neto

(Licenciado em Engenharia Electrotécnica e de Computadores na Faculdade de Engenharia da Universidade do Porto)

Dissertação submetida para satisfação parcial dos requisitos do programa de Mestrado em Engenharia Electrotécnica e de Computadores, com especialização em Informática Industrial

004(043)/NETJ/ALG
UNIVERSIDADE DO PORTO
Faculdade de Engenharia
BIBLIOTECA
N.º 21836
CDU 004(043)
Data 11/6/1996
N.º 2923

Porto, Julho de 1995

Dissertação realizada sob a supervisão do
Professor Doutor Raul Fernando de Almeida Moreira Vidal,
Professor Associado do Departamento de Engenharia Electrotécnica e de
Computadores da Faculdade de Engenharia da Universidade do Porto

Palavras Chave

Algoritmos Genéticos Distribuídos
Algoritmos Genéticos
Sistemas Distribuídos

Os Algoritmos Genéticos são uma poderosa técnica de optimização, capaz de resolver problemas complexos de optimização em inúmeras áreas científicas. Estes algoritmos são compatíveis com uma implementação distribuída, a qual pode melhorar significativamente a sua eficiência.

Por outro lado, os Sistemas Distribuídos são um poderoso paradigma de computação, que possibilita o aumento da eficiência na execução de aplicações, utilizando e compatibilizando sistemas de computação heterogéneos.

Este trabalho avalia a possibilidade de distribuição dos Algoritmos Genéticos. São consideradas diversas características e modelos de distribuição destes algoritmos, no sentido de os adaptar da melhor forma a sistemas distribuídos.

Como parte deste trabalho, é definida a arquitectura e construído um protótipo de uma plataforma para a distribuição dos Algoritmos Genéticos, independente do problema e capaz de funcionar em sistemas computacionais de diferentes características.

A avaliação do desempenho da plataforma, em comparação com a abordagem não distribuída, é feita com a sua aplicação na resolução de um problema real, de significativa complexidade, relacionado com a análise da estabilidade de um sistema de produção e distribuição de energia eléctrica.

Keywords

Distributed Genetic Algorithms
Genetic Algorithms
Distributed Sistems

The Genetic Algorithms are a powerful optimisation technique, that can address several complex problems in many different scientific areas. These algorithms are suitable for distributed implementation, witch may significantly improve its efficiency.

On the other hand, the Distributed Systems are a powerful computational paradigm, that provides a better application's execution efficiency, using different computational systems.

This work evaluates the Genetic Algorithms distribution possibility. Several issue and distribution models are taken into consideration, when trying to adapt these algorithms to the distributed systems.

As part of this work, a distribution platform for the Genetic Algorithms are defined and implemented as a prototype. This platform is problem independent, and is able to run in different computational systems.

This platform is then applied in the resolution of a real and complex optimisation problem, related to the electric power systems stability analysis. The results are evaluated and compared to the results of the non-distributed Genetic Algorithms approach.

Agradecimentos

Gostaria de agradecer a algumas pessoas todo o apoio prestado na realização deste trabalho.

Em primeiro lugar, agradeço ao Prof. Raúl Vidal toda a orientação e auxílio prestados, quer no decorrer da fase curricular deste mestrado, quer na elaboração desta dissertação.

Agradeço ao João Ranito, que foi quem me levou a trabalhar na área dos AG, contribuindo com inúmeras sugestões e prestando um precioso apoio.

Agradeço ao Paulo Seabra, por todas as vezes que leu e releu este trabalho, detectando centenas de "grolhas".

Agradeço ao Prof. Peças Lopes pela sua contribuição em todos os aspectos relacionados com o problema de optimização utilizado para o teste da plataforma desenvolvida neste trabalho.

Por fim, e sem desmérito, agradeço a todos os meus colegas do INESC, Ademar Aguiar, Álvaro Ribeiro, Ana Paula, Cristina Ribeiro, João Correia Lopes, João Paulo Cunha, João Pascoal, Gabriel Davide, Jorge Amaral, Jorge Simões, José Correia, José Manuel Moreira, José Maria Bonnet, Luis Gama, Manuel Joaquim e Mário Santos, que de uma forma ou de outra, contribuiram para a realização deste trabalho. Agradeço também ao Paulo Alexandre Ferreira, pelo auxílio prestado.

Queria ainda referir que a realização deste trabalho seria praticamente impossível sem o apoio logístico e financeiro do INESC, e o apoio financeiro da JNICT com a bolsa BM/3401/92-IA.

1. INTRODUÇÃO 1

1.1 OBJECTIVOS DESTE TRABALHO 1
1.2 ESTRUTURA DESTE TRABALHO 2

2. ALGORITMOS GENÉTICOS NA OPTIMIZAÇÃO 5

2.1. ALGORITMOS GENÉTICOS 5
2.1.1. Evolução Natural 5
2.1.2 O Algoritmo Genético 6
2.1.3. Análise do Algoritmo Genético 7
2.1.4. Diferenças Relativamente aos Métodos Clássicos 16
2.2. MODELOS TEÓRICOS DO ALGORITMO GENÉTICO 17
2.2.1. Esquemas e Semelhanças 17
2.2.2. Outros Métodos de Análise 19
2.3. APLICAÇÃO DO ALGORITMO GENÉTICO 20
2.3.1. Codificação 20
2.3.2. Parametrização 21
2.4. DOMÍNIOS DE APLICAÇÃO 21
2.4.1. Optimização Numérica 22
2.4.2. Optimização Combinatória 22
2.4.3. Programação Genética 23

3. ALGORITMOS GENÉTICOS DISTRIBUÍDOS 25

3.1. MODELOS DE ALGORITMOS GENÉTICOS 25
3.2. COMPUTAÇÃO DISTRIBUÍDA 26
3.2.1. Modelos de Computação Paralela 26
3.2.2. Ambientes de Computação Distribuída 27
3.3. ALGORITMO GENÉTICO PARALELO 29
3.3.1. Paralelização Directa do Algoritmo Genético 29
3.3.2. Algoritmo Genético Assíncrono 31
3.3.3. Implementação do Algoritmo Genético Assíncrono 33
3.4. ALGORITMO GENÉTICO CELULAR 34
3.4.1. Definição do Modelo 34
3.4.2. Implementação do Algoritmo Genético Celular 35
3.5. MODELO DE ILHAS 36
3.5.1. Definição do Modelo 36
3.5.2. Parametrização e Topologia 37
3.5.3. Implementação do Modelo de Ilhas 40
3.6. POPULAÇÕES SOBREPOSTAS 40
3.6.1. Definição do Modelo 40
3.6.2. Implementação do Modelo das Populações Sobrepostas 41
3.7. RESUMO 42

4. SIMULAÇÃO, ANÁLISE E RESULTADOS **45**

4.1. INTRODUÇÃO 45
4.2. BASE DE TESTE 47
4.2.1. Problemas de Optimização Numérica 48
4.2.2. Problemas de Optimização Combinatória 51
4.3. APLICAÇÃO DO ALGORITMO GENÉTICO ORIGINAL 53
4.3.1. Resultados dos Problemas de Optimização Numérica 53
4.3.2. Resultados dos Problemas de Optimização Combinatória 56
4.4. APLICAÇÃO DO ALGORITMO GENÉTICO ASSÍNCRONO 58
4.5. APLICAÇÃO DO ALGORITMO GENÉTICO CELULAR 60
4.6. APLICAÇÃO DO MODELO DE ILHAS 63
4.6.1. Análise Preliminar 63
4.6.2. Análise de Desempenho 65
4.7. APLICAÇÃO DO MODELO DAS POPULAÇÕES SOBREPOSTAS 71
4.8. RESUMO 73

5. AGENDA: ALGORITMOS GENÉTICOS DISTRIBUÍDOS E ASSÍNCRONOS **75**

5.1. INTRODUÇÃO 75
5.2. ARQUITECTURA DA AGENDA 76
5.2.1. Processos Intervenientes 77
5.2.2. Protocolo de Comunicação 79
5.2.3. Eleição do PGAG 81
5.2.4. Extensões para Tolerância a Falhas e Migração de Processos 82
5.3. ESTRUTURA INTERNA DA AGENDA 84
5.3.1. Estrutura Modular 84
5.3.2. Comunicação entre Processos 86
5.3.3. Aspectos de Implementação 87
5.4. TESTE DA AGENDA 88
5.4.1. Breve Descrição do Problema de Teste 88
5.4.2. Aplicação do Algoritmo Genético Original 89
5.4.3. Aplicação e Parametrização da AGENDA 90
5.4.4. Comparação da Qualidade das Soluções 92
5.4.5. Comparação dos Tempos de Execução 93

6. CONCLUSÕES **97**

1.1 CONCLUSÕES DESTE TRABALHO 97
1.2 SUGESTÕES PARA FUTURAS EXTENSÕES 98

ANEXO A. EFEITOS DA MIGRAÇÃO NO MODELO DE ILHAS **101**

A.1. DESCRIÇÃO DAS EXPERIÊNCIAS 101
A.2. RESULTADOS OBTIDOS 102
A.2.1. Problema F6 102
A.2.1. Problema F7 103
A.2.1. Problema F8 103
A.2.1. Problema M80 104
A.2.1. Problema PCV96 104

ANEXO B. PARÂMETROS DE CONTROLE DO PROGRAMA AGENDA 105

B.1. O PROGRAMA AGENDA 105
B.2. INVOCAÇÃO E OPÇÕES 106
B.2.1. Invocação do Programa 106
B.2.2. Parâmetros Relativos ao Modelo 107
B.2.3. Parâmetros de Controle Global 107
B.2.4 Parâmetros de Controle do Algoritmo 109
B.2.5 Parâmetros Relativos ao Modelo de Ilhas 109
B.3. INVOCAÇÃO E OPÇÕES 110
B.3.1. Ficheiros *log* , *best* e *seed* 110
B.3.2. Ficheiro Definição de Topologias do Modelo de Ilhas 110

ANEXO C. PROBLEMA DE TESTE DA AGENDA 113

C.1. DESCRIÇÃO DO PROBLEMA 113
C.1.1. Enquadramento 113
C.1.2. A Ilha de Lemnos 114
C.2 APLICAÇÃO DO ALGORITMO GENÉTICO 116
C.2.1. Codificação 116
C.2.2. Avaliação 117
C.2.3. Operadores 118
C.2.4 Qualidade das Soluções Obtidas 118

BIBLIOGRAFIA 119

Figura 2.1: Algoritmo Genético 7
Figura 2.2: Superfície definida pela função f 8
Figura 2.3: Codificação de uma solução de f num cromossoma 8
Figura 2.4: Conjunto de soluções inicial 10
Figura 2.5: Operador de Cruzamento 12
Figura 2.6: Operador de Mutação 12
Figura 2.7: Conjunto de soluções na 3ª geração 13
Figura 2.8: Evolução das soluções 14
Figura 2.9: Conjunto das soluções na última geração 15
Figura 2.10: Comparação entre o AG e a pesquisa aleatória 16
Figura 2.11: Evolução dos esquemas 19

Figura 3.1: Algoritmo Genético Paralelo 30
Figura 3.2: Arquitectura de um AG Massivamente Paralelo 31
Figura 3.3: Algoritmo Genético Assíncrono (AGA) 32
Figura 3.4: Modelo do AG Celular 34
Figura 3.5: Modelo de Ilhas para o AG 36
Figura 3.6: Diferentes topologias utilizadas no Modelo de Ilhas 39
Figura 3.7: Modelo das Populações Sobrepostas 41

Figura 4.1: Versão bidimensional da superfície definida pela função de Rastringin (F6) 49
Figura 4.2: Versão bidimensional da superfície definida pela função de Schwefel (F7) 50
Figura 4.3: Versão bidimensional da superfície definida pela função de Griewangk (F8) 50
Figura 4.4: Desempenho do AG na função F6. 54
Figura 4.5: Desempenho do AG na função F7. 54
Figura 4.6: Desempenho do AG na função F8. 55
Figura 4.7: Desempenho do AG no problema M80 56
Figura 4.8: Desempenho do AG na abordagem do problema PCV96 57

Figura 5.1: Arquitectura global da AGENDA 78
Figura 5.2: Sequência típica de troca de mensagens entre PAG e PGAG 80
Figura 5.3: Estrutura interna dos elementos da AGENDA 85
Figura 5.4: Resultados da aplicação do AG original ao problema de teste 90
Figura 5.5: Comparação da qualidade das solução obtidas com a AGENDA e com o AG original 92
Figura 5.6: Evolução das soluções em função do tempo 95

Figura C.1: Diagrama do Sistema Eléctrico da ilha de Lemnos 115
Figura C.2: Representação de uma expressão numa árvore 116

Tabela 3.1: Modelos de AG distribuídos e ambientes de distribuição 43

Tabela 4.1: Soluções encontradas na resolução de F6, F7 e F8 55
Tabela 4.2: Soluções encontradas na resolução de M80 57
Tabela 4.3: Soluções encontradas na resolução de PCV96 58

Tabela 5.1: Tipos de mensagens trocadas entre os processos da AGENDA 79
Tabela 5.2: Mensagens relacionadas com a migração de PAG 83
Tabela 5.3: Máquinas aplicadas à resolução do problema 91
Tabela 5.4: Soluções encontradas na resolução de PL 92
Tabela 5.5: Coeficientes da capacidade dos processadores das máquinas A, B, C e D 94

Tabela A.1: Resultados do Modelo de Ilhas para F6 102
Tabela A.2: Resultados do Modelo de Ilhas para F7 103
Tabela A.3: Resultados do Modelo de Ilhas para F8 103
Tabela A.4: Resultados do Modelo de Ilhas para M80 104
Tabela A.5: Resultados do Modelo de Ilhas para PCV96 104

Tabela B.1: Parâmetros relativos ao modelo 107
Tabela B.2: Parâmetros de controle global 107
Tabela B.3: Problemas de teste pré-definidos 108
Tabela B.4: Parâmetros de controle do algoritmo 109
Tabela B.5: Parâmetros relativos ao Modelo de Ilhas 109
Tabela B.6: Lista de ficheiros de topologias pré-definidas 111

INTRODUÇÃO

A aplicação de Algoritmos Genéticos (AG) a problemas de optimização tem tido um significativo sucesso, em diversas áreas. Estes algoritmos caracterizam-se por uma grande facilidade de adaptação a diferentes problemas, revelando uma inerente robustez no método de pesquisa.

As suas aplicações a problemas reais tem vindo a dar resultados satisfatórios, demonstrando assim o seu potencial como disciplina de optimização [Davis91] [Goldberg89]. Actualmente, existem mesmo aplicações industriais baseadas nesta metodologia [Velasco91], o que demonstra tratar-se de algo mais do que um exercício teórico.

No entanto, para problemas complexos, o volume de computação consumido neste método pode atingir valores extremamente elevados, dificultando ou mesmo inviabilizando a sua aplicação. Nestes casos, o tempo necessário para atingir soluções com a desejada qualidade é muito grande.

Uma possível forma de resolver esta questão consiste em distribuir o trabalho a realizar por diferentes sistemas computacionais (ou diferentes processadores), que passam a operar sobre uma parte do problema, cooperando entre si. Neste contexto, surgiram diversas propostas de modelos de AG distribuídos [Gordon93], com diferentes características.

1.1. OBJECTIVOS DESTE TRABALHO

O principal objectivo deste trabalho consiste em demonstrar que a implementação distribuída do AG é possível, tentando perceber qual a melhor forma de maximizar a sua eficiência e eficácia.

Conforme se verá nas conclusões deste trabalho, até certo ponto o AG beneficia linearmente com o aumento do número de processadores do sistema computacional onde se insere, desde que seja correctamente aplicado. Ao contrário de outros algoritmos de optimização, o tempo de execução do AG é dividido pelo número de processadores.

Para distribuir o AG, de modo a possibilitar a abordagem de problemas "pesados", é concebido e implementado um protótipo de uma plataforma de distribuição para esta metodologia. Esta plataforma, designada AGENDA (acrónimo de Algoritmos GENéticos Distribuídos e Assíncronos), é capaz de tornar cooperantes diferentes sistemas computacionais, de modo a aproveitar os recursos disponíveis na resolução de qualquer problema de optimização com AG.

A concepção da AGENDA tem em conta aspectos relacionados com a heterogeneidade dos sistemas constituintes de um ambiente de computação distribuída, incluindo as diferentes arquitecturas computacionais, os diferentes sistemas operativos, as diferentes capacidades de processamento, entre outras. A estrutura da AGENDA pretende ainda conferir-lhe um elevado grau de robustez e autonomia.

Para que a AGENDA implemente correctamente o AG distribuído, foi feito um estudo dos diferentes modelos para a sua distribuição. Esta distribuição obriga a uma reestruturação do algoritmo original, implicando alterações na forma de evolução da pesquisa. No sentido de averiguar o impacto destas modificações na eficácia e eficiência do AG, foram realizadas diversas simulações dos diferentes modelos, nas quais se teve em consideração a influência de diversos parâmetros específicos de cada modelo.

Os simuladores dos modelos de AG distribuídos, que permitiram a análise do comportamento destes, foram desenvolvidos especificamente para o efeito, como parte integrante deste trabalho. As experiências realizadas por seu intermédio permitiram conhecer qual a melhor forma de distribuir o AG. Este conhecimento foi aproveitado na AGENDA, quer na sua estrutura, quer no controle da sua execução.

1.2. ESTRUTURA DESTE TRABALHO

O presente trabalho está dividido em seis capítulos, incluindo o presente. O próximo capítulo, apresenta o AG na sua forma canónica, descrevendo a sua estrutura interna e o seu modo de funcionamento, através de um exemplo detalhado de aplicação a um problema simples. Seguidamente, são considerados diferentes paradigmas de aplicação do AG, nomeadamente a optimização numérica, optimização combinatória e programação genética, dando ênfase a alguns exemplos concretos.

No terceiro capítulo são analisados os modelos de distribuição de AG mais conhecidos, entre os quais se encontram o Algoritmo Genético Assíncrono [Zeigler], o Algoritmo Genético Celular [Baluja92], o Modelo de Ilhas [Tanese90] e o Modelo das Populações Sobrepostas [Baluja92].

Neste capítulo, é feito um resumo dos principais tipos de arquitectura de sistemas de computação distribuída, classificados segundo a forma como possibilitam a distribuição da computação. Para cada um dos modelos de AG apresentados, é analisada a sua adaptação áquelas arquitecturas.

No quarto capítulo, é definida uma base de testes, constituída por cinco problemas, aos quais são aplicados os AG. É descrita a forma de adaptação do algoritmo a cada um deles, e apresentados os resultados obtidos. Seguidamente, é simulado o funcionamento de cada um dos diferentes modelos de AG já referidos.

A comparação directa dos resultados obtidos com cada um dos modelos variantes de AG evidencia as suas características e o seu valor relativo, dando relevância a um conjunto de factores operacionais.

No quinto capítulo é definido o modelo funcional e a arquitectura da plataforma de distribuição de AG desenvolvida, que engloba os conceitos analisados em capítulos anteriores, no que se refere a modelos de AG distribuídos. São descritos os pormenores de estrutura e implementação, e analisada a sua aplicação a um problema de optimização real, que serve de teste à abordagem distribuída, suportada pela plataforma.

O problema de optimização em causa relaciona-se com a análise de estabilidade de uma rede eléctrica de distribuição de energia [Lopes94], no qual se pretende encontrar, como solução, um programa ou uma expressão. Este programa, tendo como entrada um conjunto de 60 parâmetros representando grandezas medidas na rede, deve fornecer uma previsão *on-line* da estabilidade da mesma (o problema pertence à classe de programação genética).

Para finalizar, são apresentados os resultados obtidos com as abordagens normal e distribuída, assim como uma detalhada comparação, incluindo a qualidade da solução e a velocidade com que os algoritmos encontram boas soluções. A melhoria no tempo de computação é também considerada.

No último capítulo, são apresentadas as conclusões deste trabalho e referidos alguns projectos complementares a este, a realizar nesta área.

ALGORITMOS GENÉTICOS NA OPTIMIZAÇÃO

2.1. ALGORITMOS GENÉTICOS

2.1.1. Evolução Natural

Inspirado na evolução biológica, John Holland sintetizou em 1962 um novo método de optimização baseado nos mecanismos de selecção natural [Holland75]. A ideia fundamental consiste na imitação do processo seguido pela natureza, na resolução algorítmica de problemas de optimização, aplicando a selecção natural e alguns operadores de base genética [Goldberg89] [Back91].

Os mecanismos de evolução biológica, há muito estudados e conhecidos, baseiam-se na operação sobre cromossomas. Um cromossoma contém a informação necessária para a criação de um ser vivo ou indivíduo, através de um processo de descodificação [Srinivas94].

Na natureza, os cromossomas que melhor se adaptam ao meio ambiente tem uma maior probabilidade de sobrevivência e consequentemente uma maior probabilidade de reprodução. Por esta razão, os seus descendentes mantêm as suas boas características. Por outro lado, os indivíduos pior adaptados têm maior dificuldade em se reproduzirem, pelo que tendem a desaparecer. Este processo é conhecido por selecção natural [Srinivas94].

A criação de novos indivíduos é levada a cabo pela reprodução, que por intermédio dos operadores genéticos combina ou altera a informação contida em cromossomas de

indivíduos já existentes. Dois operadores conhecidos são o cruzamento, que normalmente combina dois cromossomas gerando outros dois, e a mutação, que altera aleatoriamente um ou mais elementos de um cromossoma [Goldberg89] [Srinivas94].

A evolução dá-se em populações de indivíduos, de uma forma surpreendentemente simples: aqueles que melhor se adaptam vão-se reproduzindo enquanto que os menos adaptados vão sendo eliminados naturalmente, de uma forma estocástica, sem certezas. A reprodução combina partes dos indivíduos que nela participam, criando nova informação genética.

2.1.2. O Algoritmo Genético

A implementação de um AG consiste em encontrar um mecanismo que possibilite a evolução de uma população de indivíduos da forma descrita. Cada indivíduo é representado por um cromossoma artificial, ou cadeia de informação, constituída por um conjunto de *bits*. Esta informação representa a codificação de uma eventual solução do problema de optimização em causa, em lugar da informação biológica.

Associado a cada indivíduo, existe o valor da sua adaptação, que representa uma medida da sua qualidade e que é calculado por uma função de avaliação específica de cada problema. Esta função atribui melhores valores aos indivíduos que melhor resolvem o problema em questão.

O algoritmo, representado na figura 2.1, consiste em deixar evoluir uma população inicial, habitualmente gerada aleatoriamente, por um determinado número de gerações. A evolução dá-se através da aplicação da selecção, seguida da aplicação de operadores genéticos. Estes operadores responsabilizam-se por introduzir nova informação genética através da recombinação e alteração da actual informação.

Desta forma, para adaptar o AG a qualquer problema específico, basta definir dois aspectos:

- A codificação no cromossoma para o problema.
- A função de avaliação de soluções candidatas, que mapeia um cromossoma num valor, relacionado com a sua adaptação.

Devido a esta simplicidade, este processo é capaz de endereçar eficazmente uma vasta gama de problemas [Davis91], bastando que as possíveis soluções do problema sejam codificáveis numa cadeia de informação (de uma forma geral, qualquer bloco de informação é codificável desta forma) e que seja possível avaliá-las.

1. Geração inicial de uma população de indivíduos;
2. Avaliação de todos os indivíduos da população;
3. Selecção dos indivíduos que se irão reproduzir;
4. Criação de uma nova população de indivíduos com os indivíduos seleccionados, por aplicação dos operadores genéticos;
5. Substituição da anterior população pela nova população, e voltar a **2.** caso não seja satisfeito o critério de paragem;

Figura 2.1: Algoritmo Genético

Embora extremamente genérico, o AG consegue atingir resultados ao nível das técnicas específicas para problemas de determinadas áreas, sendo ainda relativamente simples a sua fusão com estas, dando origem a algoritmos híbridos ainda melhores [Davis91].

2.1.3. Análise do Algoritmo Genético

Nesta secção é ilustrada a aplicação do AG a um problema de optimização simples. O problema consiste na maximização da função *f* definida em $\Re^2$ pela próxima equação, no intervalo especificado e com a precisão de 0.01:

$$f(x_1,x_2) = 5+3\cdot(1-x)^2\cdot e^{(-x^2-(y+1)^2)} - 10\cdot(\frac{x}{5}-x^2-y^5)\cdot e^{(-x^2-y^2)} - \frac{1}{3}\cdot e^{(-(x+1)^2-y^2)},$$
$$x \in [-5.12, 5.12]$$

Este é um problema fácil de optimizar, e conforme se verá, não constitui nenhuma dificuldade para o AG, servindo apenas de exemplo para explicar o seu funcionamento. A função tem propositadamente 2 óptimos locais, relativamente perto do óptimo global, para tentar "enganar" o processo de optimização. A sua forma é ilustrada na figura 2.2.

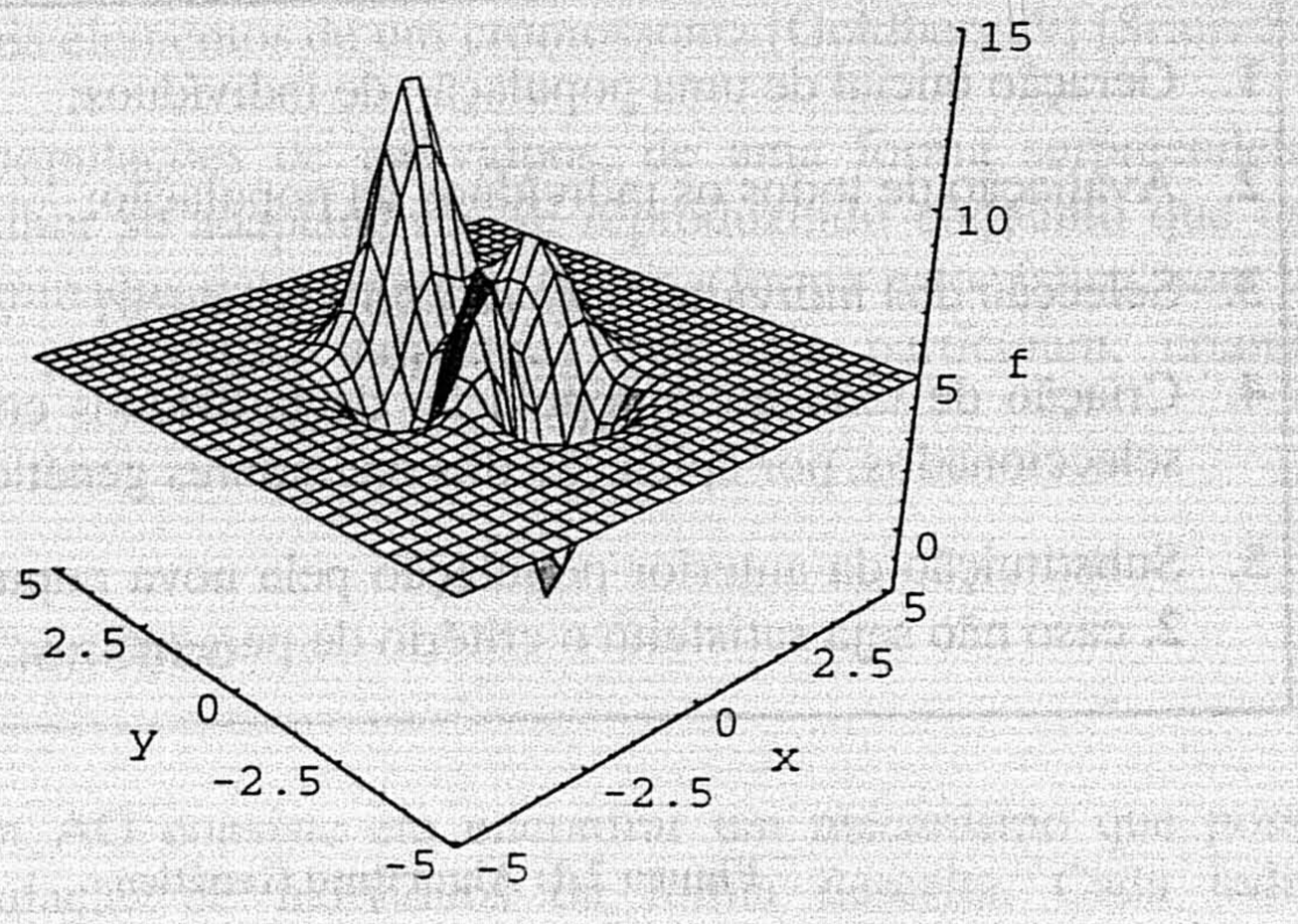


Figura 2.2: Superfície definida pela função *f*

2.1.3.1. Codificação

A estrutura do cromossoma no AG original é constituída por um conjunto de *bits* de informação, cada um dos quais podendo conter o valor binário 0 ou 1. Embora seja possível utilizar outras codificações mais elaboradas, conforme veremos mais adiante, nesta secção é preservada a sua estrutura original.

A primeira consideração consiste em definir a codificação do problema no cromossoma. Existem duas variáveis, x e y, cada uma das quais representando uma quantidade real. Essas variáveis são transformadas numa sequência de *bits*, com a forma representada na figura 2.3.

1100101001	0111100010
x_1	x_2

Figura 2.3: Codificação de uma solução de *f* num cromossoma

Considerando o intervalo e precisão definidos (ver a equação de definição de *f*), são necessários 10 *bits* para cada variável (para a precisão de 0.01 e para o intervalo entre -5.12 e 5.12, existem 5.12-(-5.12)/0.01 = 1024 possíveis valores, representáveis em $\log_2 1024 = 10$ *bits*), pelo que o cromossoma tem no total 20 *bits*.

2.1.3.2. Avaliação

A segunda e última consideração consiste na implementação da função de avaliação, que descodifica o cromossoma e lhe associa um valor de adaptação, de acordo com a equação referida. A implementação desta função deve começar pela descodificação de ambas as variáveis x_1 e x_2, o que pode ser conseguido por simples tradução do número binário representado pelo cromossoma em números reais, conforme definido na seguinte equação (**s** representa a sequência binária de todo o cromossoma):

$$x_1 = \frac{\sum_{i=0}^{i<10} s_i \cdot 2^i}{100}, \quad x_2 = \frac{\sum_{i=10}^{i<20} s_i \cdot 2^{i-10}}{100}$$

Com os valores de ambas as variáveis descodificados, resta calcular o valor da função *f*, o que não levanta qualquer problema.

2.1.3.3. Parametrização e Execução do Algoritmo

Na parametrização do AG para este exemplo, foi escolhida uma população de 32 indivíduos (mais tarde será analisado o impacto do tamanho da população na resolução do problema), o que significa que, simultaneamente, numa dada geração, existem 32 soluções candidatas à resolução do problema. As primeiras soluções são aleatórias, pelo que não são em geral satisfatórias. É devido ao trabalho do AG que irão evoluir para soluções melhores.

Nesta execução do algoritmo, as soluções iniciais (aleatórias) encontravam-se espalhadas pelo espaço de estados da função. A figura 2.4, dá uma ideia dessa dispersão, representando as referidas soluções por pequenas esferas.

Com esta população inicial de indivíduos, começa o ciclo genético. Com a primeira avaliação de todos os 32 indivíduos da população, surgem indivíduos melhor e pior adaptados, que irão competir no sentido de serem seleccionados para a reprodução. Na simulação efectuada, o conjunto inicial dos valores das 32 soluções (uma por indivíduo) teve como média o valor 5.56 e como melhor solução o valor 8.03 (a função *f* assume valores entre -2 e 13.1, aproximadamente).

A execução do algoritmo foi mantida por 100 gerações, que neste caso foram mais do que suficientes para assegurar a convergência, ou seja, atingiram um ponto a partir do qual não houve melhoria das soluções (no presente exemplo coincidiu com o atingir da solução óptima). Em cada geração, os indivíduos foram sujeitos a selecção, seguida de cruzamento e mutação.

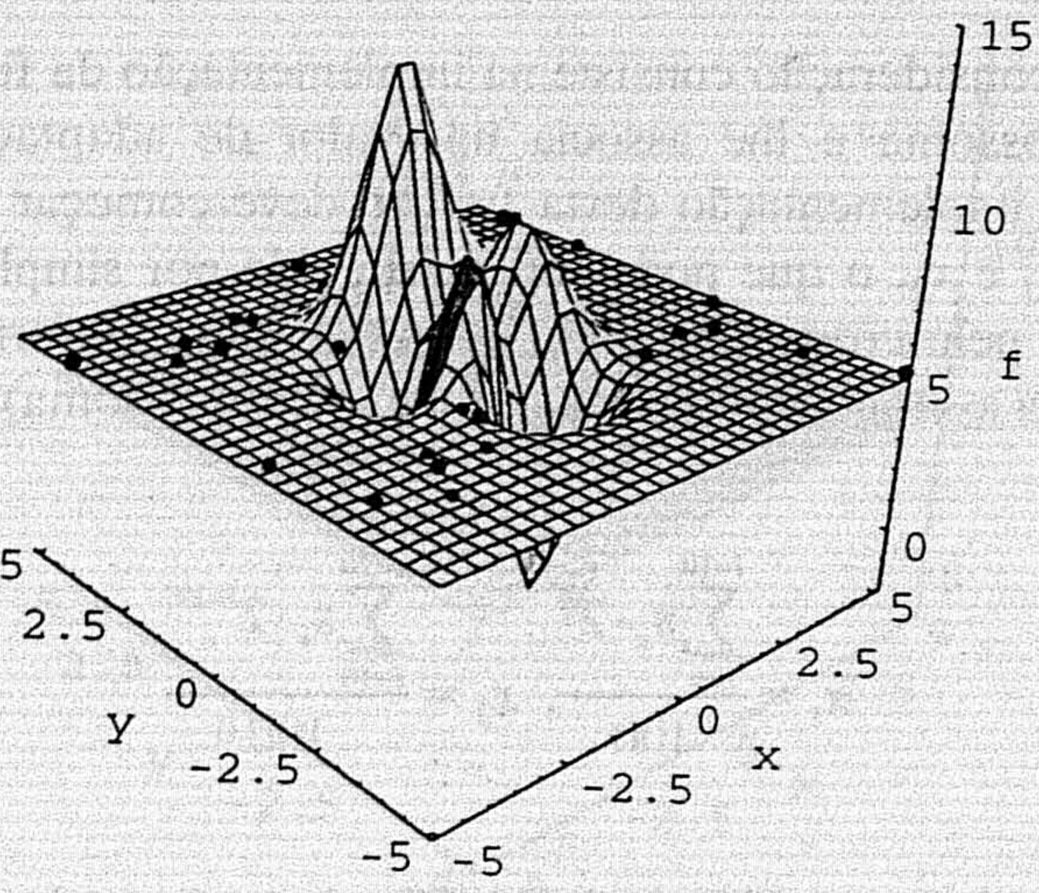


Figura 2.4: Conjunto de soluções inicial

2.1.3.4. Selecção

A selecção tem por objectivo escolher soluções para recombinação, dando preferência às melhores, mas sem certezas. Existem diversas formas de implementar o operador de selecção [Goldberg91a] [Back92], entre as quais se encontram o *torneio estocástico* e a *roleta viciada*. Ambos os operadores seleccionam um indivíduo de cada vez, de entre dois ou mais indivíduos escolhidos ao acaso na população. Em nenhum deles é garantido que o melhor indivíduo considerado seja o escolhido.

O torneio estocástico é definido pela sua ordem, que indica o número de indivíduos que irão competir entre si num só acto de selecção. No torneio de ordem dois, são escolhidos dois indivíduos ao acaso na actual população, vencendo o melhor com uma determinada probabilidade (o melhor indivíduo é aquele que contém o maior valor de adaptação, calculado pela função de avaliação do problema). A probabilidade de vencer o melhor indivíduo encontra-se tipicamente entre 0.7 e 0.9 [Goldberg91a]. Torneios de ordem superior são também utilizados.

Na roleta viciada, é associado a cada indivíduo participante no acto de selecção uma ranhura de uma roleta virtual, de tamanho proporcional ao seu valor de adaptação. Efectuando uma simulação do funcionamento da roleta, os melhores indivíduos têm maior probabilidade de serem os seleccionados.

A principal diferença entre estes dois métodos é que, no torneio estocástico, a diferença de adaptação entre os dois indivíduos em escolha é irrelevante. Neste método, se a sorte ditar que irá vencer o pior, assim será sem que a diferença entre valores de adaptação

interfira. No entanto, na roleta viciada, uma vez que a ranhura é proporcional ao valor de adaptação, é muito improvável que um indivíduo com um valor muito inferior ao adversário venha a vencer. Por outro lado, indivíduos com adaptações próximas têm quase a mesma probabilidade de escolha, ficando o melhor com uma vantagem demasiadamente fraca.

Este facto leva a algumas considerações. Em populações em que a gama de valores de adaptação é pequena, ou seja, em que os valores de adaptação são muito próximos uns dos outros, a roleta viciada transforma-se em escolha aleatória, uma vez que dois quaisquer indivíduos têm ranhuras praticamente iguais. Esta situação torna-se indesejável, uma vez que pode acontecer frequentemente [Goldberg91a] (em alguns problemas pode ser difícil mapear os resultados da função de avaliação numa gama suficientemente grande, e após iniciada a convergência das soluções há uma grande tendência para estas se aproximarem em valores de adaptação).

O torneio tem também a desvantagem de permitir com maior facilidade que soluções más passem de geração. Mas este efeito tende a diluir-se à medida que aumenta o número de gerações processadas, pelo que pouco interfere no desempenho comparativamente à roleta viciada. Por estas razões, e para que seja viável utilizar sempre o mesmo operador de selecção, é utilizado em todas as experiências deste trabalho o operador de torneio estocástico.

Após a selecção, o indivíduo vencedor poderá então vir a ser actuado por um dos operadores genéticos, ou poderá ser simplesmente passado para a geração seguinte inalterado. De qualquer das formas, as suas hipóteses de influência na evolução da população (e consequentemente na resolução do problema) ainda não se anularam. O indivíduo derrotado é reposto na população onde se encontrava, à espera de nova oportunidade de selecção, que poderá nunca vir a acontecer.

Uma vez seleccionados todos os indivíduos, são aplicados os operadores genéticos. Os operadores mais utilizados são o cruzamento e a mutação.

2.1.3.5. Cruzamento

No cruzamento, são criados dois novos indivíduos através da combinação dos cromossomas de dois indivíduos previamente seleccionados. A combinação dos cromossomas pode ser feita de diversas formas [Goldberg89] [DeJong91], sendo a mais simples o cruzamento de um ponto, que consiste em cortar os dois cromossomas num determinado ponto (obtido aleatoriamente) e cruzar as 4 partes separadas para obter dois novos cromossomas, conforme representado na figura 2.5.

O cruzamento pode ser feito na variante de dois pontos, no qual se definem dois pontos de corte entre os quais se trocam as subcadeias interiores. Levando este processo ao limite, chega-se ao cruzamento uniforme, no qual para cada ponto é decidido ao acaso qual dos descendentes o vai herdar.

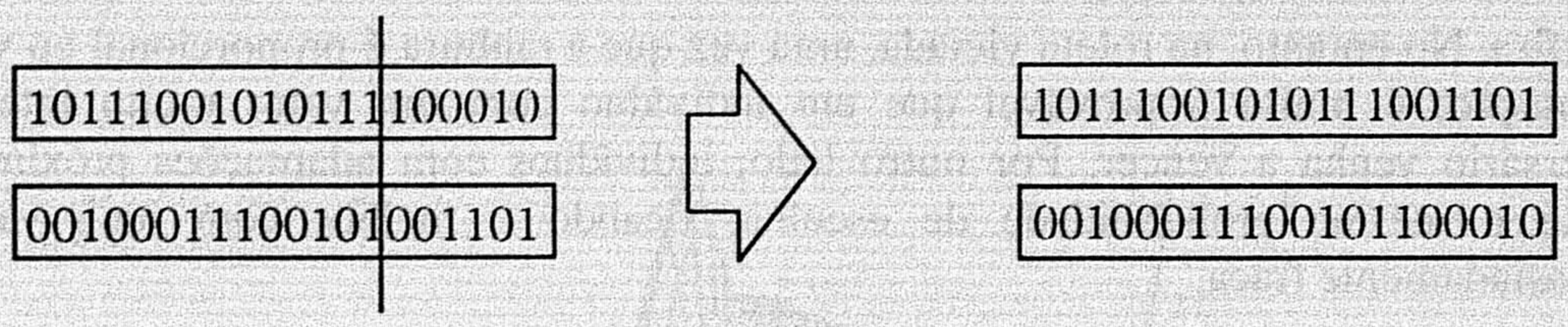


Figura 2.5: Operador de Cruzamento

Este operador é aplicado segundo o parâmetro *probabilidade de cruzamento*, cujos valores típicos se situam perto dos 0.8 [Goldberg89]. Assim, como nem sempre é aplicado, alguns indivíduos podem passar para a geração seguinte sem serem alterados por este operador.

2.1.3.6. Mutação

A mutação consiste na simples troca de elementos do cromossoma. Na sua forma mais simples, a sua acção resume-se a trocar um dos *bits* escolhido ao acaso, num qualquer elemento previamente seleccionado, conforme representado na figura 2.6.

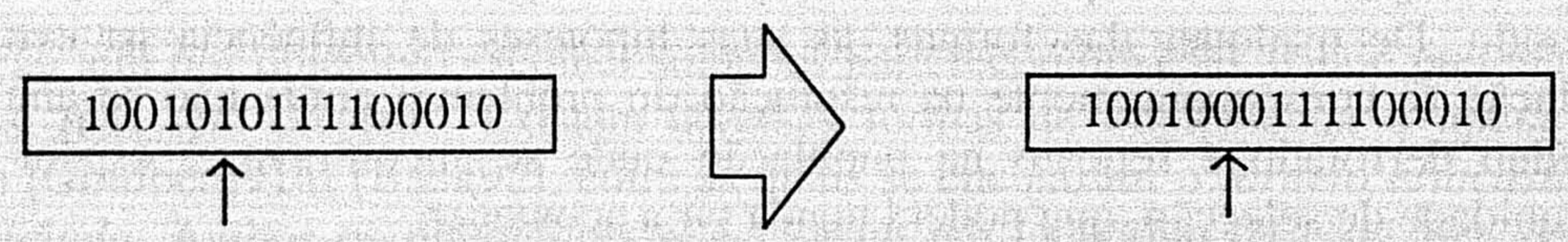


Figura 2.6: Operador de Mutação

A probabilidade de aplicar o operador de mutação deve ser muito baixa [Back92a], trocando habitualmente um só *bit* em toda a população (por geração), ou uma pequena percentagem dos *bits* da população. Desta forma, a maioria dos elementos seleccionados não chegam a ser manipulados por nenhum operador. Existem assim indivíduos que não são alterados nem pelo cruzamento nem pela mutação, sendo copiados integralmente para a geração seguinte.

2.1.3.7. Evolução das Soluções

Após algumas gerações, o conjunto de soluções inicial evoluiu no sentido da aproximação do máximo da função *f*. Na terceira geração, as soluções ocupavam posições no espaço de estados muito diferentes das soluções iniciais, conforme se ilustra na figura 2.7.

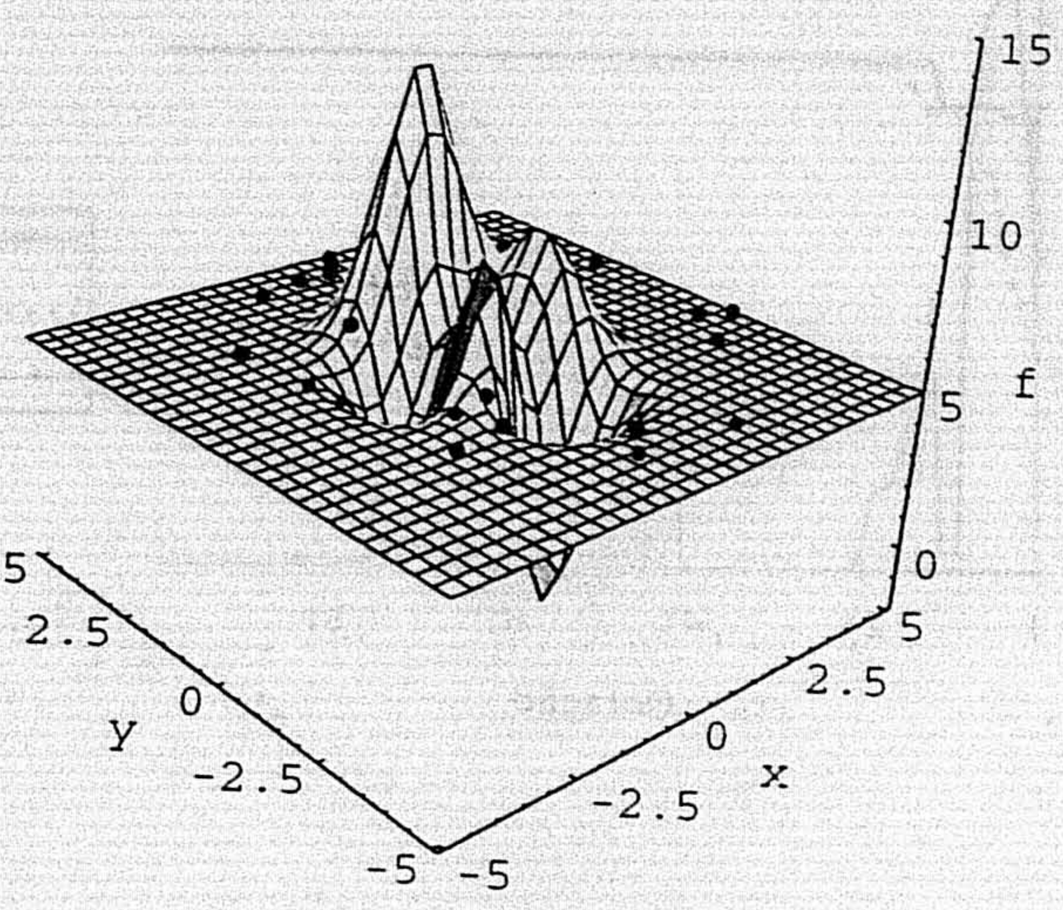


Figura 2.7: Conjunto de soluções na 3ª geração

Mesmo tendo decorrido só 3 gerações, as soluções já se encontram muito mais próximas dos pontos mais elevados do gráfico. Nesta geração, para a simulação realizada, a média da população é de 6.92 e a melhor solução atinge o valor de 11.61, valores significativamente acima dos valores iniciais já referidos (que tinham como média o valor 5.56 e como melhor solução o valor 8.03).

Para melhor ilustrar o desenvolvimento do algoritmo, são apresentadas na figura 2.8 as curvas de evolução da melhor solução, da média das soluções e do respectivo desvio padrão, obtidas em função do número de gerações.

Conforme é visível nesta figura, após algumas gerações, o algoritmo atinge o óptimo de *f*, que corresponde ao valor aproximado de 13.1. A evolução das 32 soluções dá-se principalmente nas primeiras 20 gerações. Inicialmente, a diversidade da população é elevada, conforme se pode ver pela curva do desvio padrão, e nesta altura há uma significativa diferença entre a média da população e a melhor solução.

Com o avançar do número de gerações, os melhores indivíduos vão ganhando predominância na população, uma vez que têm maior interferência na geração de novos descendentes, devido ao seu superior valor de adaptação. Por este motivo, a curva da média da população tende a aproximar-se da curva do melhor indivíduo, diminuindo assim a diversidade.

Por volta da geração número 21, foi atingida a convergência do AG. Neste momento, todos os indivíduos têm o mesmo valor de adaptação (não são necessariamente iguais, uma vez que podem existir diferentes pontos do espaço de estados com iguais valores na função de avaliação), conforme se pode ver pelo valor nulo do desvio padrão.

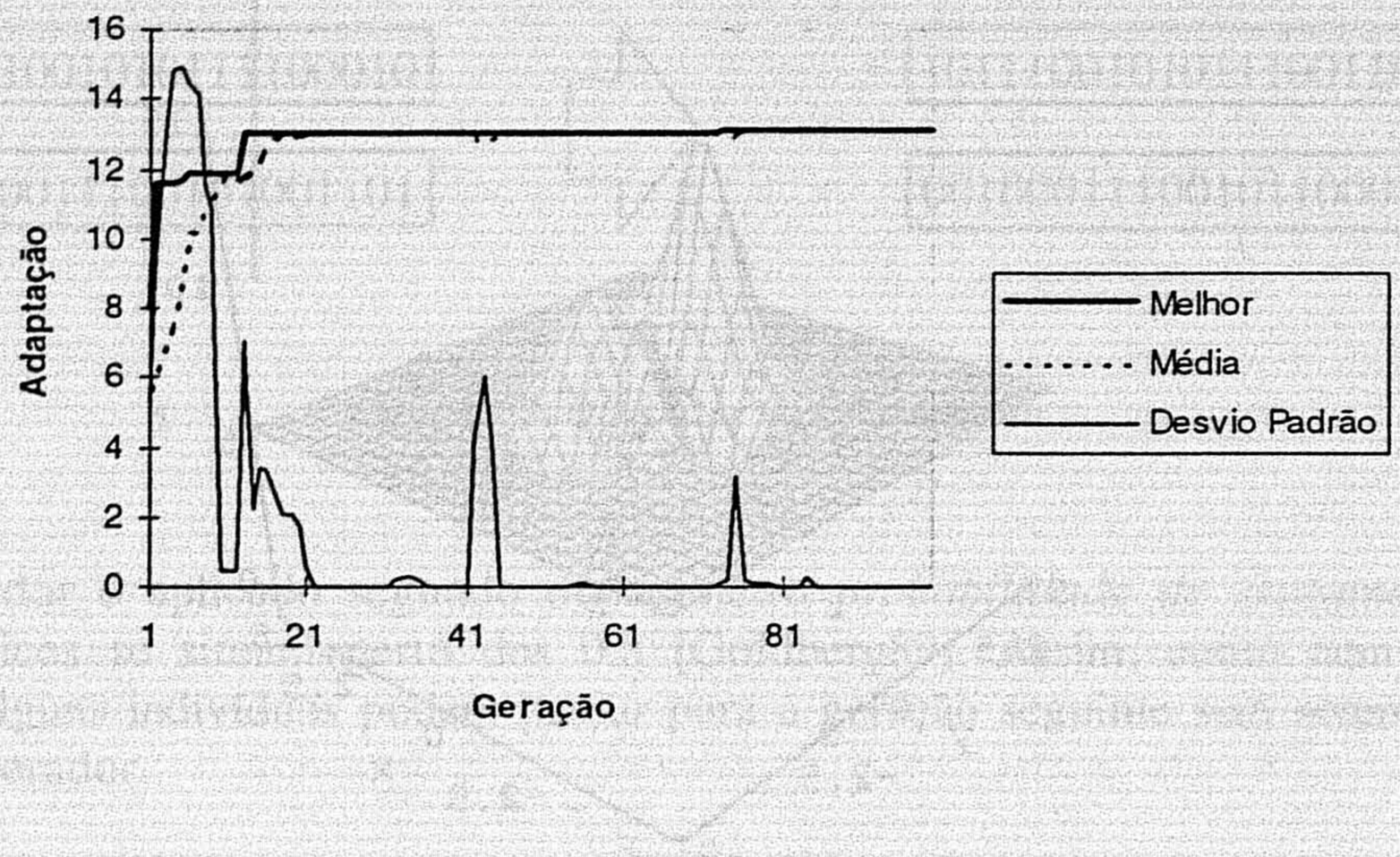


Figura 2.8: Evolução das soluções

Com os cromossomas todos iguais, o cruzamento de indivíduos não pode originar diferentes soluções. A única forma de evolução daqui em diante baseia-se na mutação, que pode introduzir novo material genético relevante, conforme acontece por exemplo perto da geração número 40. O primeiro efeito da mutação é baixar a média da população, já que a alteração aleatória de uma solução trabalhada quase de certeza que a piora. No entanto, o AG aproveita esta pequena diversidade para tentar melhorar as soluções já existentes (isto de facto acontece, algumas gerações adiante, conforme se vê na figura 2.8 pelo pequeno melhoramento final da melhor solução).

Embora este seja um problema matematicamente simples, é possível verificar a grande velocidade de convergência do AG para uma boa solução. No final da execução, todas as soluções representam o ponto $\{x;y\} = \{-0.01;1.6\}$, equivalente a um valor em f de 13.110271, que é de facto a melhor solução para a precisão inerente aos 10 *bits* de informação para as variáveis. Assim, o gráfico das soluções na última geração, representado na figura 2.9, ilustra um só ponto, onde todas as 32 soluções finais convergem.

É importante salientar que a forma como o AG foi adaptado a este problema em concreto em nada dependeu do problema em si. Do ponto de vista do AG, houve unicamente manipulação de cadeias abstractas de informação, sem estrutura e sem significado para ele. A única informação que influenciou as suas "decisões" de escolha foi o valor de adaptação de cada cadeia em particular, calculada pela função de avaliação f. A manipulação das cadeias de informação foi feita sem nenhuma heurística, cortando e cruzando cromossomas em pontos aleatórios e trocando elementos sem qualquer consideração directa.

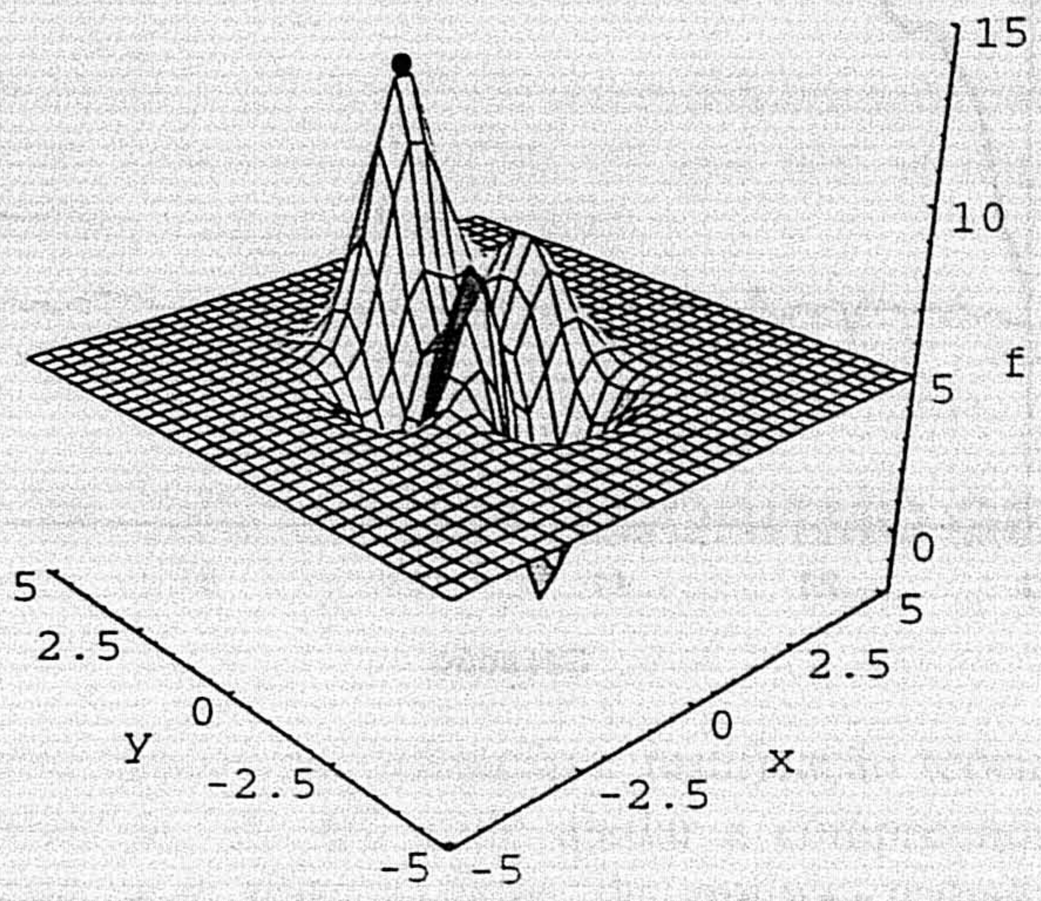


Figura 2.9: Conjunto das soluções na última geração

Esta independência do AG em relação ao problema permite uma grande generalidade e independência do seu funcionamento, incentivando o seu estudo autónomo, como uma disciplina por si só. Com esta metodologia, é possível considerar problemas para os quais pouco ou nada se conhece acerca da forma da solução [Davis91], bastando para tal saber avaliar uma dada solução em comparação com outras soluções. A forma simples como o AG conduz a pesquisa dá-lhe uma grande robustez e autonomia.

Apesar de o AG se basear em decisões estocásticas, é completamente diferente da pesquisa aleatória pura [Goldberg89]. Para evidenciar a diferença entre ambos os métodos, é ilustrada uma comparação directa na figura 2.10, na qual são desenhadas as curvas da média das soluções em função do número de geração (no algoritmo aleatório, são geradas soluções em grupos de 32, para criar a noção artificial de geração, útil na comparação).

A pesquisa aleatória sofre do "mal da dimensão" do espaço de estados, ou seja, a enorme quantidade de soluções diferentes, neste caso $2^{20} \approx 10^6$, impede que a pesquisa não orientada seja satisfatória. Gerando 32 soluções por geração, o total de 3200 ao fim de 100 gerações não se aproxima nem um pouco do número total de possíveis soluções, pelo que a probabilidade de encontrar o óptimo é muito pequena.

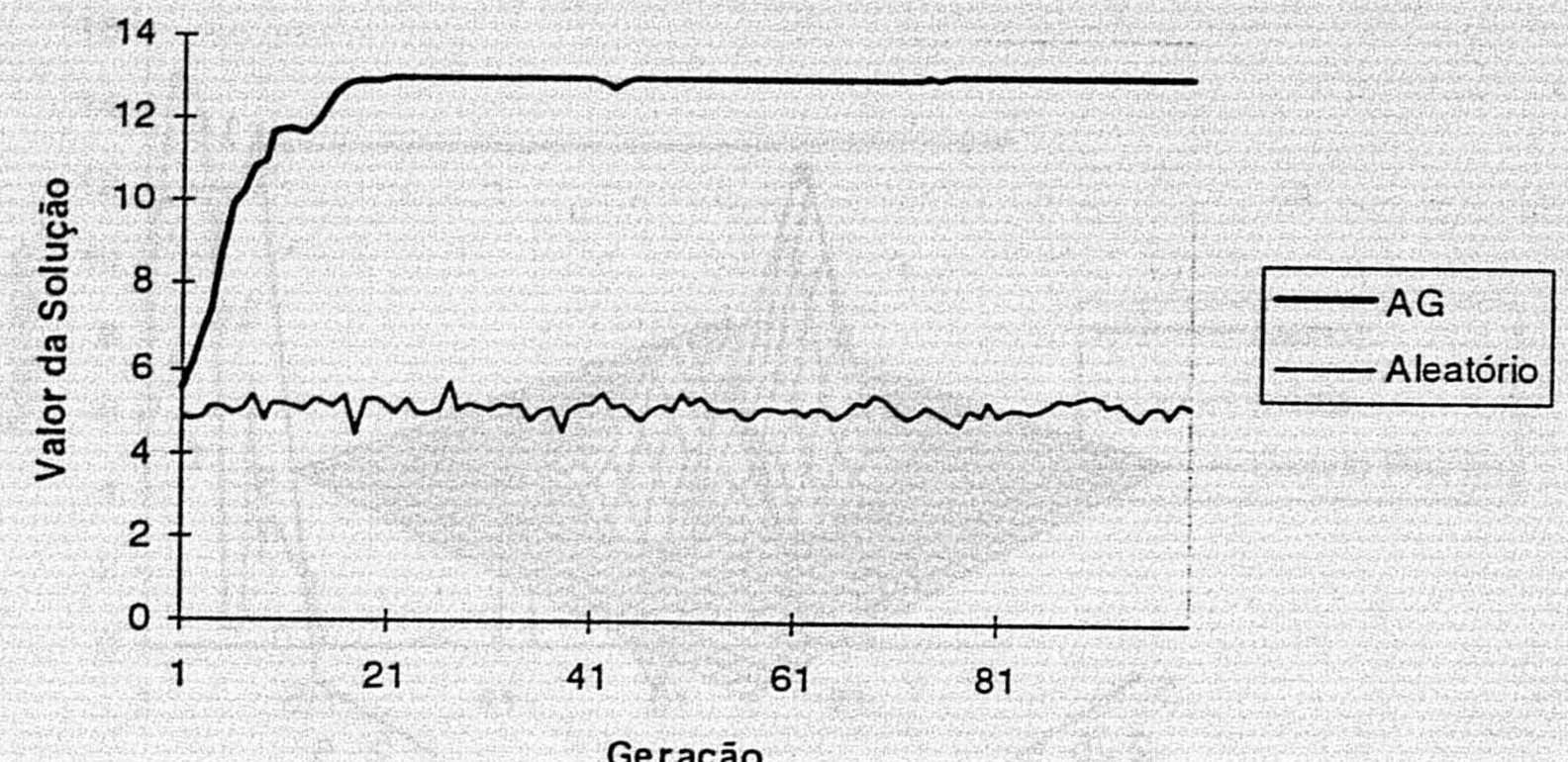


Figura 2.10: Comparação entre o AG e a pesquisa aleatória

2.1.4. Diferenças Relativamente aos Métodos Clássicos

As quatro principais características que distinguem esta metodologia relativamente aos métodos clássicos de optimização são as seguintes [Goldberg89]:

- São utilizadas codificações dos parâmetros em lugar dos próprios parâmetros dos problemas, sendo o significado desta codificação completamente irrelevante para o AG. A codificação pode ser feita em binário explícito, ou em qualquer outra forma usando alfabetos mais complexos.

- A pesquisa realiza-se a partir de um conjunto de pontos, cada um representado por um elemento da população, e não a partir de um só ponto. Esta característica diminui a probabilidade de convergência prematura para óptimos locais, uma vez que são analisados vários pontos do espaço de estados simultaneamente.

- O *feedback* é obtido directamente a partir da função de avaliação, e não através do recurso a derivadas ou outras informações parciais, nem sempre possíveis de obter. Sem esta restrição, o AG pode considerar uma maior gama de problemas.

- As transições baseiam-se em regras probabilísticas, em vez de regras determinísticas, o que também diminui a probabilidade de convergência para óptimos locais.

Os diferentes métodos clássicos têm habitualmente um muito bom desempenho numa pequena faixa de tipos de problemas, sendo muito ineficientes nos restantes. Em contrapartida, o AG consegue ter um desempenho aceitável e homogéneo em quase todo

o tipo de problemas, por vezes um pouco inferior a métodos específicos da área, mas suficientemente eficaz [Davis91] [Mitchell92].

Com o AG, é possível abordar a resolução de um problema sem conhecimento profundo da respectiva área, muitas vezes em situações em que não se conhece qualquer outro método aplicável com melhor sucesso [Liening94].

2.2. MODELOS TEÓRICOS DO ALGORITMO GENÉTICO

Existem algumas propostas para a modelização e justificação teórica do funcionamento do AG. O modelo teórico de um AG pode ajudar a compreender e melhorar o seu funcionamento, pelo que tem sido a base de muitos trabalhos de investigação [Goldberg89] [Angeline93].

Fazendo uma pré-análise intuitiva, verifica-se que o AG consegue evoluir boas soluções combinando partes de cromossomas previamente seleccionados por bom desempenho. Algumas dessas partes de cromossomas, também conhecidas por subcadeias de informação, contêm padrões que conduzem a um bom desempenho do indivíduo a que pertencem.

A acção combinada de seleccionar bons indivíduos, que contêm boas subcadeias, e a construção de novos indivíduos com essas mesmas subcadeias, aumenta a oportunidade de se criarem subcadeias ainda melhores.

Assim, o AG manipula simplesmente a informação disponível na "esperança" de construir informação ainda melhor. Por vezes, o resultado da combinação de duas boas soluções pode ser uma má solução. Nestes casos, a própria selecção natural encarrega-se de a pôr de parte.

À medida que o número de gerações vai aumentando, a diversidade nos indivíduos da população vai diminuindo, uma vez que as melhores subcadeias (que são um subconjunto de todas as cadeias) tendem a difundir-se mais do que as restantes. É neste contexto que a mutação desempenha um importante papel ao reintroduzir novo material genético (aleatório), de crucial importância para futuras recombinações, evitando muitas vezes a convergência para uma solução subóptima.

2.2.1. Esquemas e Semelhanças

A teoria mais conhecida para a modelização dos AG é a teoria dos esquemas, introduzida por Holland [Holland75] [Goldberg89]. Esta teoria modeliza as semelhanças

existentes nos diversos cromossomas de uma população, num conceito designado esquema.

Um esquema traduz um determinado padrão nos elementos dos cromossomas. A estes padrões estão associadas melhores ou piores características das soluções candidatas, e a evolução do seu número de elementos traduz a evolução do algoritmo.

A introdução dos esquemas permite considerar informação extra na análise do AG. Além das cadeias de informação em si, está a ser considerada a comparação entre elas. Esta análise passa por verificar o efeito dos operadores genéticos sobre os esquemas.

Prova-se, através do teorema dos esquemas [Glodberg89], que esquemas pequenos, com adaptação acima da média, têm uma maior probabilidade de sobrevivência, o que leva a que o seu número aumente exponencialmente de geração em geração, levando à sua rápida difusão pela população. A vantagem dos esquemas pequenos, em relação aos restantes, deve-se à sua maior imunidade à destruição provocada pelos operadores genéticos. Para uma análise mais detalhada deste assunto, deve ser consultada a bibliografia [Ranito93] [Goldberg89].

Numa população, os melhores indivíduos têm, em média, um maior número de esquemas com bom desempenho. Uma vez que a selecção e reprodução têm tendência a privilegiar os melhores indivíduos, os melhores esquemas são também beneficiados.

No anterior exemplo de aplicação do AG, é possível verificar a evolução dos esquemas através da análise do gráfico da figura 2.11, no qual se ilustra, para cada geração, a composição média dos indivíduos. Cada coluna do gráfico representa, numa dada geração, o conteúdo médio das 20 posições de cada cromossoma (uma por linha). O valor é traduzido pela percentagem de elementos igual ao valor binário 1 numa determinada posição e é representado graficamente pela tonalidade do cinzento. As 100 gerações da anterior evolução são representadas ao longo das colunas.

Inicialmente, as diferenças entre os elementos do cromossoma são insignificantes, traduzindo uma distribuição homogénea na população inicial aleatória. À medida que o número de gerações progride, algumas posições vão tomando valores fixos, ou preto ou branco, que significa que todos os elementos da população têm o mesmo valor binário nessas posições, igual a 0 ou a 1. A razão desta mudança deve-se ao facto de haver, cada vez mais, um maior número de indivíduos que vão fixando valores em determinadas posições do seu cromossoma.

Estas sequências fixas, que têm tendência a se definir cada vez melhor, são os referidos esquemas, que se vão difundindo na população. Quando um esquema já é parte integrante de todos os elementos, é representado no gráfico da figura 2.11 pelas cores fixas, branco ou preto.

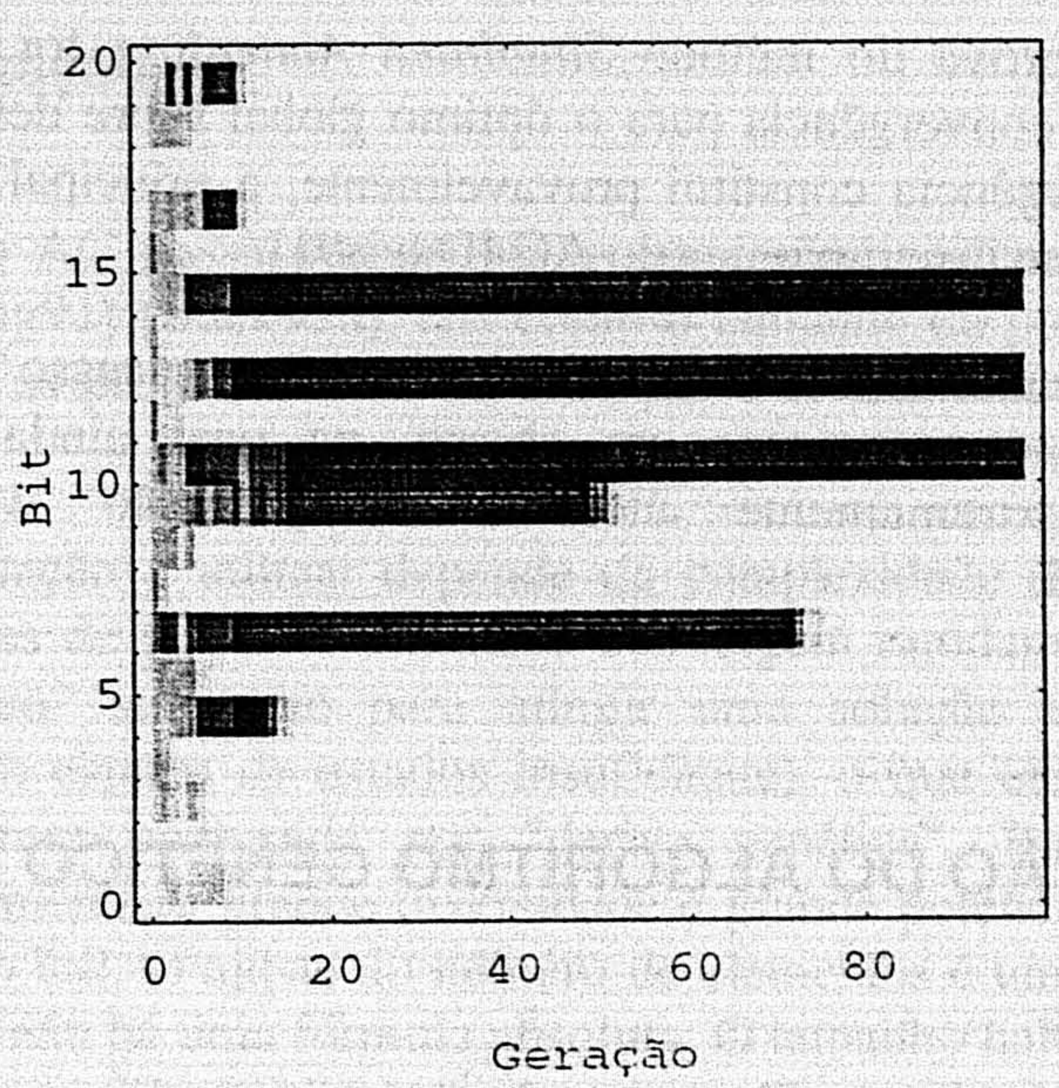


Figura 2.11: Evolução dos esquemas

No actual exemplo, após algumas gerações iniciais, quase toda a população partilha o mesmo esquema. A mutação reflecte-se por pontuais variações nos esquemas da população, que tendem a ser absorvidas. Uma mutação pode ainda originar, ao acaso, um esquema melhor do que os já existentes, o que habitualmente resulta na sua imediata difusão por toda a população. É o que acontece, conforme já visto, perto da geração número 40 da simulação analisada.

2.2.2. Outros Métodos de Análise

Embora largamente aceite, o teorema dos esquemas foi sujeito a várias críticas. As principais assentam no facto da sua inexactidão, na sua desadaptação ao comportamento dinâmico do algoritmo e na dificuldade de o adaptar a variações do AG original [Peck1993]. Deste ponto de vista, a teoria é incompleta e foram realizados diversos trabalhos no sentido de se encontrar uma teoria mais robusta e completa.

Como consequência destes trabalhos, surgiram novas teorias que, tendo em conta as várias instâncias de um determinado esquema e alguns parâmetros adicionais, conseguem obter resultados mais exactos do que a teoria dos esquemas [Peck1993]. No entanto, o grau de abstracção das operações relacionadas com o cálculo desses parâmetros reduz a sua aplicabilidade.

Relativamente à análise dinâmica do AG, as cadeias de *Markov* foram aplicadas em diversos contextos em versões muito simplificadas do algoritmo, na maioria das vezes em pormenores específicos do seu funcionamento [Mahfoud93].

O procedimento de prova do método *Simulated Annealing* foi também aplicado, na tentativa de provar a convergência para o óptimo global sobre determinadas condições. Esta prova de convergência constitui provavelmente, a principal direcção dos futuros desenvolvimentos na análise dinâmica do AG [Davis91].

Outros trabalhos específicos foram desenvolvidos, com sucesso relativo. No entanto, esta é uma área que permanece em aberto, na qual ainda se esperam muitos desenvolvimentos, extremamente úteis à compreensão e melhoramento do funcionamento do AG.

2.3. APLICAÇÃO DO ALGORITMO GENÉTICO

2.3.1. Codificação

Na aplicação do AG a problemas concretos, é necessário ter em atenção aspectos relacionados com a codificação das soluções candidatas em cromossomas. Embora a dependência da codificação seja muito inferior à maioria das técnicas clássicas de optimização, esta deve respeitar alguns aspectos.

O primeiro aspecto tem a ver com o cruzamento e a mutação. A codificação deve ser fechada relativamente a estes dois operadores [Goldberg89]. Os novos elementos gerados geneticamente devem ser válidos, evitando que surjam soluções "impossíveis", na codificação em uso. Uma forma alternativa, no caso em que não se consiga evitar a geração destas soluções, é penalizá-las fortemente com a própria função de avaliação, de forma a evitar a sua proliferação.

Por outro lado, uma boa representação deve evidenciar a semelhança entre conceitos análogos, ou seja, deve traduzir no cromossoma as semelhanças entre eles [Goldberg89]. A título de exemplo, e para ilustrar que uma codificação directa pode não ser a ideal, considere-se a optimização da função unitária $f(x)=x$. Com uma representação binária, os cromossomas 0111 e 1000 têm valores de f de 7 e de 8, respectivamente, e todos os seus elementos (*bits*) são diferentes. Por outro lado, 0000 e 1000 têm valores de f muito distintos, 0 e 8, e os seus cromossomas são muito parecidos. Este efeito denomina-se *Hamming Clifs*, no qual há uma grande diferença entre dois cromossomas (grande distância de *Hamming*) com valores na função de avaliação adjacentes. Uma forma de minimizar este efeito é a utilização de outras representações binárias, como por exemplo os códigos de *Gray* [Goldberg89]. No entanto, deve salientar-se que o AG não é crucialmente dependente da representação, e mesmo com representações directas consegue evoluir satisfatóriamente.

2.3.2. Parametrização

A parametrização do AG passa por definir os seus principais parâmetros operacionais [Goldberg91] [DeJong91] [Back92a], tais como o tamanho da população, o número de gerações a executar, as probabilidades de mutação e cruzamento, a forma de selecção, entre outras.

O tamanho da população a utilizar depende da complexidade do problema em causa [Smith93]. Populações de pequena dimensão convergem rapidamente, mas podem não ter "material genético" suficiente para atingir uma solução satisfatória (têm uma amostragem inicial do espaço de estados insuficiente), o que origina uma situação de convergência prematura conhecida por "deriva genética". Populações de elevada dimensão, embora contenham grande diversidade e grande potencial, podem não chegar a convergir em tempo útil. O ajuste do número de elementos é um processo heurístico e empírico, para o qual não há uma fórmula absoluta. O tamanho habitual das populações é cerca de 100 elementos, embora seja normal, para problemas de elevada complexidade, a utilização de 1000 ou mais indivíduos.

O número de gerações depende do grau de convergência do algoritmo, que está relacionado com a natureza do problema e com o tamanho da população. Até certo ponto, a evolução é muito acentuada, tendendo posteriormente a estabilizar.

A probabilidade de cruzamento tem de ser correctamente dimensionada. Grandes valores neste parâmetro aumentam a recombinação de soluções, mas também aumentam a probabilidade de destruição de bons esquemas já construídos (efeito conhecido por disrupção). Pequenos valores têm os efeitos contrários. Um valor típico é 0.8.

Uma grande probabilidade de mutação reintroduz muito material genético no processo, mas tende a transformar o AG num algoritmo de pesquisa aleatória, diminuindo drasticamente o seu desempenho. O valor deste parâmetro deve ser conservado em pequenos valores.

Todos estes parâmetros dependem também de quais os operadores genéticos em uso. Por outro lado, um conjunto de valores para os operadores pode funcionar bem no início do algoritmo, mas degenerar ao fim de algumas gerações. Por esta razão, é comum efectuar algumas alterações aos parâmetros que controlam a execução, no decorrer da própria execução [Smith93].

2.4. DOMÍNIOS DE APLICAÇÃO

Os AG foram já aplicados a um grande conjunto de problemas, alguns dos quais fora da esfera meramente teórica. Existem inúmeros exemplos de aplicação, incluindo a

resolução de problemas de outras áreas científicas, entre os quais se salientam os seguintes:

- Geração de topologias de redes neuronais [Belew90], em que os AG definem as ligações entre os neurónios constituintes de uma rede neuronal, de forma a maximizar o seu desempenho na resolução de um problema concreto;

- Optimização do fluxo de cargas em redes eléctricas de potência [Yin91], em substituição dos métodos numéricos utilizados na área;

- Controle operacional de uma central de geração de energia eléctrica por conversão de combustível fóssil [Velasco91];

- Desenho de circuitos em placas VLSI [Liening93], no qual se pretende minimizar o espaço de silício ocupado e a distância total de ligações entre os componentes, espalhados por várias camadas.

- Optimização de perguntas relacionais [Ranito93], na qual se utiliza o AG para reescrever uma expressão relacional da forma mais eficiente, do ponto de vista do seu tempo de processamento no acesso aos dados;

São seguidamente descritas três grandes áreas de aplicação de AG, no que respeita à forma e estrutura interna da codificação e dos respectivos operadores: optimização numérica, optimização combinatória e programação genética.

2.4.1. Optimização Numérica

A optimização numérica envolve a maximização de problemas que são traduzidos por uma função real multi-dimensional. Os cromossomas codificam as diferentes variáveis, que levam a função objectivo a um determinado valor.

A optimização numérica aplicada a funções matemáticas puras não constitui novidade [Goldberg89], tendo já sido visto na secção anterior a forma de codificar as soluções candidatas num cromossoma. Ao contrário daquele exemplo, para muitos destes problemas, não existe uma solução conhecida, nem sequer um método garantidamente bom, pelo que se justifica a aplicação do AG nesta área.

2.4.2. Optimização Combinatória

A optimização combinatória relaciona-se com problemas que consistem em encontrar sequências ou reordenações a partir de conjuntos iniciais. Estão também incluídos nesta classe os problemas de partição de grafos ou de definição de subconjunto sob determinados critérios.

Um grande conjunto de problemas conhecidos caem na classe dos problemas de optimização combinatória [Khuri94], envolvendo permutações, e que consistem em encontrar sequências ordenadas ou subconjuntos de um conjunto inicial. Os seguintes exemplos são típicos desta classe de problemas, e a todos eles foram já aplicados os AG com sucesso:

- Problema da Mochila 01 (PM01), que implica a escolha de um subconjunto de objectos a incluir numa mochila, cada qual com o seu valor de benefício e peso, de forma a maximizar o benefício total sem ultrapassar um peso limite [Khuri93];
- Problema do Caixeiro Viajante (PCV), no qual se pretende encontrar uma sequência de visita a um conjunto de cidades, de forma a minimizar o percurso global [Davis91];

2.4.3. Programação Genética

Uma das mais recentes variantes do AG é a Programação Genética (PG), desenvolvida por Koza [Koza91]. Conforme indicado pelo próprio nome, os indivíduos que constituem a população são programas, ou sequências de acções. Estes programas são criados e desenvolvidos pelo processo genético no decorrer das gerações.

As linguagens utilizadas nestes programas não devem ser as convencionais linguagens procedimentais como C ou Fortran. Estas linguagens baseiam-se fortemente numa sintaxe rígida, que dificulta a geração de novos programas por recombinação genética (os programas assim gerados têm uma enorme probabilidade de estarem sintacticamente incorrectos). Um outro problema deve-se ao facto de serem linguagens de baixo nível, o que implica uma grande lista de instruções mesmo na resolução de problemas simples.

No outro extremo, a utilização de linguagens especificamente concebidas para um problema em particular sofre de falta de generalidade. Neste caso, a escolha da linguagem tem um enorme impacto no desempenho do algoritmo, pondo em questão se os resultados se devem ao paradigma da PG ou à linguagem em si, que pode capturar quase todos os conceitos relevantes do problema em particular, mas impede a sua aplicação a outros problemas.

Um bom compromisso entre o exposto, são as linguagens derivadas ou semelhantes ao LISP. O LISP é largamente utilizado em problemas de inteligência artificial, não é uma linguagem procedimental e é facilmente representável numa estrutura em árvore. As estruturas em árvore são, por sua vez, facilmente representáveis em cromossomas. Estas características fazem com que seja o principal candidato à utilização em PG [Angeline93].

Para uma descrição detalhada da estrutura dos cromossomas e dos operadores genéticos neste paradigma, deve ser consultada a bibliografia [Ranito93] [Spears91].

ALGORITMOS GENÉTICOS DISTRIBUÍDOS

3.1. MODELOS DE ALGORITMOS GENÉTICOS

A implementação distribuída do AG obriga a uma redefinição do algoritmo original. O principal objectivo deste capítulo é apresentar um conjunto de modelos para a sua distribuição.

A principal motivação para a distribuição do AG centra-se na diminuição do tempo total gasto na sua execução. A aplicação do AG a problemas complexos envolve grandes volumes de cálculo, inerentemente demorados. Muitas vezes, a resolução satisfatória de um determinado problema baseia-se em grandes populações e num elevado número de gerações, podendo não ser atingida qualquer solução razoável em tempo útil. Por outro lado, a utilização de grandes populações consome recursos computacionais, nem sempre disponíveis de uma forma centralizada.

Algumas das soluções descritas neste capítulo propõem a divisão do AG original num conjunto de subalgoritmos de menor complexidade, que vão operar em subpopulações próprias, de tamanho comportável, trocando informação entre si, de forma a melhorar o mais possível o desempenho global.

Na primeira secção são apresentadas as principais arquitecturas e ambientes de computação distribuída, com a associada nomenclatura. Nas secções seguintes, são analisados os mais conhecidos modelos de AG distribuídos, nomeadamente o AG Assíncrono, o AG Celular, o Modelo de Ilhas e o Modelo das Populações Sobrepostas, assim como alguns dos aspectos de implementação associados.

3.2. COMPUTAÇÃO DISTRIBUÍDA

A constante necessidade de melhoria da capacidade de computação e redução dos custos associados foram alguns dos principais factores que motivaram o aparecimento e desenvolvimento de sistemas de computação paralela ou distribuída [Mullender92]. Estes sistemas podem assumir inúmeras formas, justificando a apresentação de um conjunto de modelos capazes de classificar as arquitecturas mais conhecidas, às quais será analisada a viabilidade de adaptação do AG.

3.2.1. Modelos de Computação Paralela

Na tentativa de obter um aumento da capacidade de computação proporcional ao aumento dos recursos de computação emergentes, foram desenvolvidas diversas arquitecturas paralelas para os sistemas de computação. Nestas arquitecturas incluem-se, entre outras, os sistemas de computadores paralelos, vectoriais, multicomputador e multiprocessador [Hwang93].

Segundo a classificação de *Flynn* [Flynn72], os sistemas podem dividir-se segundo as noções de *Instruction Stream* e *Data Stream*. O sistemas sequenciais convencionais designam-se por sistemas *Single Instruction Single Data* (SISD), nos quais existe um só processador, e os dados ou operandos manipulados por esse elemento são elementares (em contraste com operandos vectoriais).

Os sistemas de computação vectorial podem assumir a forma *Single Instruction Multiple Data* (SIMD), ou seja, a mesma instrução é executada sobre um conjunto de dados em paralelo, habitualmente na forma de um vector. Nestes sistemas, embora exista um só processador, a quantidade de informação processada é muito superior ao caso anterior, uma vez que os dados ou operandos manipulados numa só instrução constituem eles próprios um conjunto de operandos elementares. Em contrapartida, nem todos os problemas se adaptam bem a esta arquitectura (que foi inicialmente desenvolvida para problemas de cálculo científico), o que leva a que a melhoria de desempenho esperada possa ficar comprometida [Hwang93].

Uma outra forma de sistemas SIMD constitui um conhecido paradigma de computação, no qual uma matriz de processadores executa a mesma sequência de instruções sobre diferentes dados. Os processadores são habitualmente de média complexidade e coexistem no mesmo sistema em grande número, muitas vezes atingindo os milhares. Existem diversos exemplos de supercomputadores com esta arquitectura [Sharp95], muitas vezes designada por sistema massivamente paralelo. Nestes sistemas, o custo de desenvolvimento de aplicações eficientes é elevado, pelo que são vocacionados para problemas específicos [Hwang93].

Os sistemas capazes de executar diferentes sequências em paralelo dividem-se em duas classes: *Multiple Instruction Single Data* (MISD) e *Multiple Instruction Multiple Data* (MIMD). Os primeiros são uma raridade, não sendo comum a sua existência [Hwang93].

Os sistemas MIMD estão bastante divulgados, sendo constituídos por sistemas com mais do que um processador, totalmente independentes, cada qual executando a sua própria sequência de instruções. Os actuais sistemas de elevado desempenho tendem a seguir esta arquitectura, que oferece as vantagens de maior expansibilidade e generalidade [Hwang93].

Nos sistemas MIMD, é comum surgir em complemento uma classificação relacionada com o grau de acoplamento dos diversos processadores e com o modelo de memória inerente. Existem, principalmente, duas conhecidas variantes: os sistemas multiprocessador e os sistemas multicomputador.

Os sistemas multiprocessador distinguem-se por terem toda a memória partilhada por todos os processadores sob o mesmo espaço de endereçamento. Desta forma, a comunicação entre os diversos processadores do sistema é feita por intermédio de partilha de memória, sendo o grau de acoplamento do sistema muito elevado, com tempos de comunicação muito reduzidos.

Os sistemas multicomputador, em contrapartida, são constituídos por diversos nós interligados por uma infraestrutura de comunicação. Cada nó representa um computador autónomo, contendo memória própria, o que proporciona um maior desacoplamento do sistema. A comunicação é feita segundo um paradigma de troca de mensagens entre nós, sendo o tempo de comunicação habitualmente superior ao caso anterior.

Esta última classe inclui um popular paradigma de computação, muitas vezes designado por *Workstations Cluster*, no qual todos os elementos de um conjunto de *workstations* cooperam por intermédio de uma eficiente ligação em rede.

3.2.2. Ambientes de Computação Distribuída

No contexto dos sistemas multiprocessador e multicomputador, surge a necessidade de facilitar a cooperação entre os diversos processadores. Esta cooperação, genericamente designada por computação distribuída [Schroeder92], assenta na possibilidade de comunicação entre os diversos elementos e nós do sistema.

No paradigma da computação distribuída, um dos objectivos é a minimização dos efeitos de dispersão provocados pela distribuição, possibilitando que o sistema funcione como um todo, fazendo com que exteriormente tudo se pareça com o tradicional sistema centralizado [Mullender92].

No sentido de facilitar o desenvolvimento de aplicações, têm surgido diversos ambientes de computação distribuída. Estes ambientes exploram de diferentes formas o inerente paralelismo do sistema, oferecendo variadas soluções para o desenvolvimento de aplicações.

3.2.2.1. Sistemas Multiprocessador

No que respeita a sistemas multiprocessador, um dos principais ambientes de programação consiste na possibilidade de paralelizar a execução dentro de um mesmo processo. Um dos mais populares mecanismos que viabiliza esta facilidade designa-se por *threads*, no qual é possível criar um conjunto de linhas de execução independentes, semelhantes a subprocessos, que serão alocadas pelo sistema operativo a diferentes processadores do sistema [Tanenbaum90]. Todos os *threads* partilham o mesmo espaço de endereçamento, pelo que a comunicação está implicitamente facilitada (o tempo de comunicação é muito baixo).

Os *threads* foram conhecidos por *light weight processes* [Sun90], por constituírem um mecanismo extremamente económico na utilização de recursos de processamento, em comparação com os processos convencionais.

Nesta classe de sistemas, a comunicação inter-processo é também facilitada, existindo habitualmente diversos mecanismos oferecidos pelos sistemas operativos, que se servem do facto de a memória ser partilhada por todo o sistema.

3.2.2.2. Sistemas Multicomputador

Nos sistemas multicomputador, onde os sistemas se caracterizam por um maior desacoplamento dos seus elementos, a forma directa de conseguir a cooperação baseia-se no paradigma de troca de mensagens. Não havendo em geral memória partilhada, um processo num determinado nó tem de enviar mensagens explicitas para os processos de outros nós, contando para isso com alguns atrasos originados pelos tempos de comunicação.

Um popular mecanismo de troca de mensagens entre processos de diferentes nós é conhecido por *sockets*. Os *sockets* [Stevens90] são utilizados na grande maioria de redes de *workstations*, como o principal mecanismo de comunicação, possibilitando o envio e a recepção de mensagens através de um conjunto de serviços de comunicação disponibilizados pelo sistema.

Com base neste mecanismo, foi desenvolvida e largamente divulgada uma outra forma de interacção entre processos mais elaborada, designada por *Remote Procedure Call* (RPC) [Tanenbaum90]. Neste ambiente, um processo em execução num determinado processador, efectua uma chamada normal a uma função, podendo esta ser executada num outro processador (eventualmente de outro nó) de uma forma transparente. Todos os pormenores de comunicação, que incluem a negociação de troca de mensagens, habitualmente assentes em *sockets*, são escondidos pelo sistema de RPC. O sistema de RPC facilita o desenvolvimento de aplicações em ambientes distribuídos, fornecendo uma alternativa de mais alto nível.

Outros ambientes de distribuição têm surgido [Douglas93], com maior ou menor popularidade. Entre estes encontra-se o ambiente *Parallel Virtual Machine* (PVM) [Beguelin91], que disponibiliza um poderoso paradigma baseado em troca de mensagens, com facilidades acrescidas de controle e sincronização.

Para algumas aplicações, a troca explícita de mensagens pode obrigar a esforços adicionais de adaptação a ambientes distribuídos. No sentido de minimizar esses esforços, alguns dos ambientes de computação distribuída simulam a existência de memória partilhada, em ambientes em que a comunicação se faz por pura troca de mensagens. Um dos mais populares exemplos desta classe é o sistema de programação paralela conhecido por *Linda* [Gelernter85], [Gelernter88].

3.3. ALGORITMO GENÉTICO PARALELO

3.3.1. Paralelização Directa do Algoritmo Genético

Devido à sua estrutura e modo de funcionamento, o AG é directamente paralelizável. O ciclo genético, designado por *época*, é constituído pela selecção, manipulação genética e avaliação das novas soluções, podendo ser facilmente executado em paralelo, sem qualquer alteração funcional ao algoritmo original.

A paralelização directa das sucessivas épocas não altera nem a estrutura nem qualquer resultado intermédio ou final, incidindo o seu efeito unicamente na velocidade de execução. A única alteração necessária é a modificação do ciclo principal do algoritmo, para que este efectue as operações genéticas em paralelo [Whitley92]. Uma possível forma de paralelizar o algoritmo é mostrada na figura 3.1.

1. Geração inicial de uma população de indivíduos;
2. Avaliação, **em paralelo**, de todos os elementos da população;
3. Efectuar **em paralelo** N/2 vezes (N é o tamanho da população):
   3.1. Seleccionar 2 indivíduos
   3.2. Criar 2 novos indivíduos, por manipulação genética, colocando-os na nova população
   3.3 Avaliar os 2 novos indivíduos criados
4. Substituição da anterior população pela nova população, e voltar a 3. caso não seja satisfeito o critério de paragem;

Figura 3.1: Algoritmo Genético Paralelo

Neste algoritmo alterado, é paralelizada a avaliação inicial de todos os elementos da população e a geração de novos elementos através do processo de selecção e manipulação genética. A avaliação dos novos elementos é efectuada assim que estes são gerados, em contraste com o AG original.

Importa salientar que na maioria dos problemas reais, o tempo de computação necessário por época deve-se principalmente à avaliação dos indivíduos [Davis91], uma vez que as operações genéticas em si não são complexas nem morosas, salvo algumas excepções (como no caso de operadores de cruzamento rebuscados, utilizados em casos específicos). Nos problemas que envolvam a avaliação de soluções por recurso a simulação, como por exemplo nos problemas típicos da programação genética [Angeline93], ou nos problemas relacionados com a definição de topologias de redes neuronais [Belew90], a situação ainda se torna mais assimétrica, sendo o tempo de computação praticamente exclusivo do processo de avaliação.

A anterior solução é facilmente implementável, tendo sido tema de diversos trabalhos, relacionados com a implementação do AG em sistemas computacionais multiprocessador [Bianchini93]. É habitualmente designada por AG Massivamente Paralelo, pelo facto de se adaptar bem a sistemas com grandes quantidades de processadores. Uma arquitectura típica destes sistemas é ilustrada na figura 3.2, na qual o processador de controle, *PC*, é o responsável por seleccionar os indivíduos a processar, e escalonar os respectivos processadores.

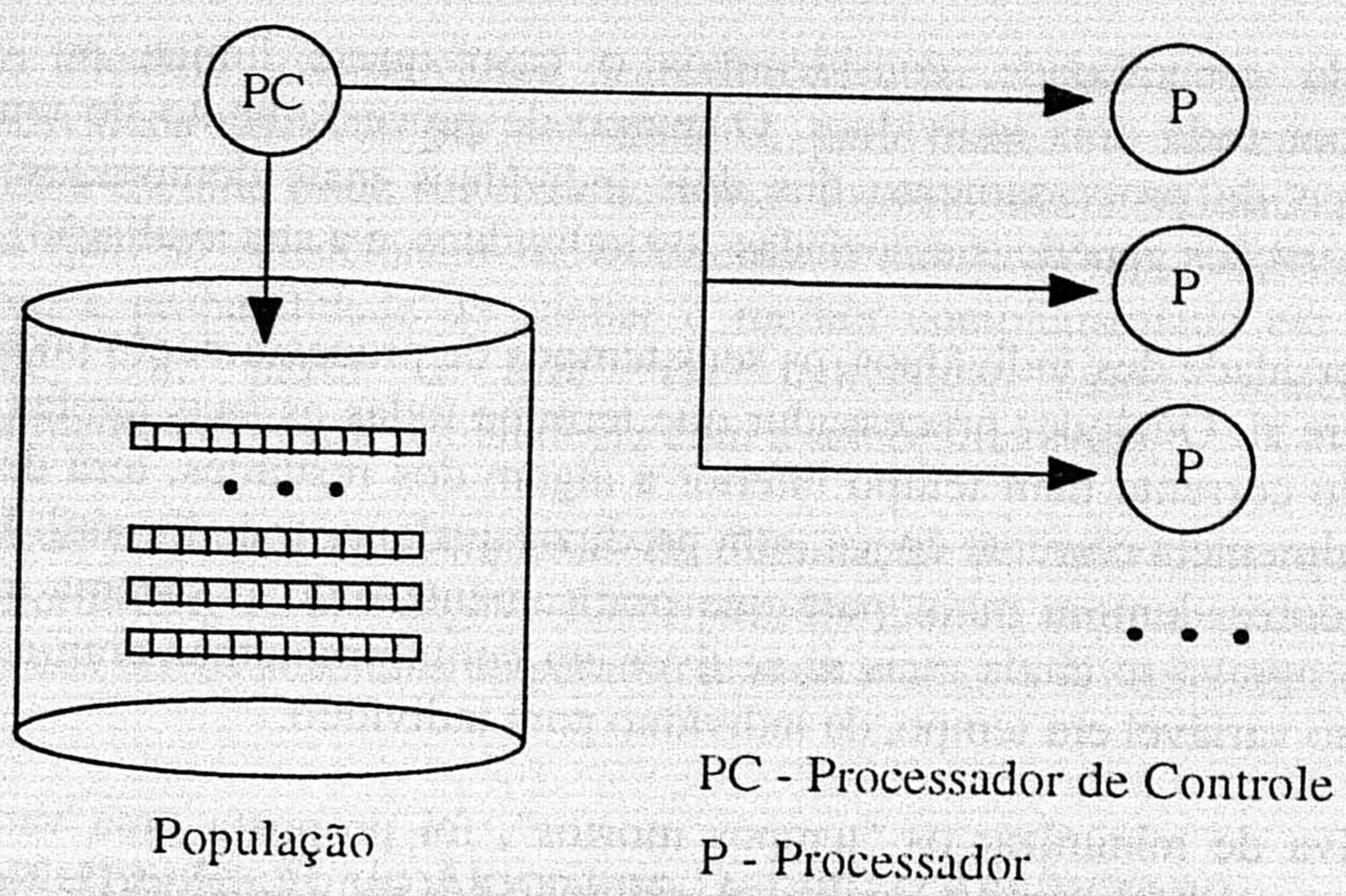


Figura 3.2: Arquitectura de um AG Massivamente Paralelo

A anterior variante do AG, embora tenha a vantagem de ser extremamente simples de implementar, não é a melhor solução do ponto de vista do aproveitamento de recursos. Conforme se verá a seguir, é possível encontrar alternativas que proporcionem um menor tempo total de execução.

3.3.2. Algoritmo Genético Assíncrono

No AG original [Goldberg89], o ciclo de selecção, cruzamento e mutação desenvolve-se por surtos, um por cada geração, não havendo mistura de indivíduos de diferentes gerações. Só após a realização de todas as operações na geração corrente é que se considera a criação síncrona da seguinte. Neste algoritmo, um novo indivíduo só pode interferir no desenvolvimento genético após terminar a geração corrente.

Uma desvantagem deste método é evidenciada quando se considera a paralelização directa do AG. Considerando o modelo anterior, cada processador livre selecciona dois indivíduos, processa-os geneticamente, avalia os novos cromossomas e insere-os na nova população.

Tendo que processar e avaliar todos os elementos da população na geração corrente, para então passar à geração seguinte, o que determina o tempo total de execução é o pior caso, ou seja, o tempo gasto pelo processador mais demorado (ou com mais "trabalho").

Sem perda de generalidade, consideremos o caso quase limite, no qual há um processador por cada dois indivíduos. O tempo de processamento de uma geração é igual ao tempo de processamento dos dois indivíduos mais demorados (este último engloba as operações genéticas associadas aos indivíduos, e a sua avaliação).

Devido à diversidade dos indivíduos, os seus tempos de processamento podem ser muito diferentes entre si. Qualquer processador que termine todas as suas tarefas relacionadas com a geração corrente num tempo inferior a algum dos restantes, terá de esperar que termine globalmente a corrente época, sem produzir qualquer trabalho útil. Basta que um dos processadores demore mais, para que praticamente todo o sistema esteja parado. Esta situação agrava-se ainda mais se se tiver em conta problemas com a avaliação de soluções muito variável em tempo, de indivíduo para indivíduo.

Numa tentativa de minimizar os "tempos mortos", foi proposta uma variante ao AG original, designada Algoritmo Genético Assíncrono (AGA) [Zeigler91]. Nesta variante, representada na figura 3.3, é possível ver que após a selecção de dois quaisquer indivíduos, seguida do respectivo cruzamento e mutação, os novos cromossomas gerados são imediatamente introduzidos na população corrente, prosseguindo o ciclo ininterruptamente. Neste contexto, não existem os anteriores conceitos de população corrente e nova população. A população é uma só, e o processamento genético efectua-se assincronamente com o número de geração, possibilitando um melhor aproveitamento de tempo em caso de paralelização do AG da forma anteriormente descrita, já que o processador mais sobrecarregado não obriga a que os restantes esperem pelo final da sua execução.

1. Geração inicial de uma população de indivíduos;
2. Avaliação **em paralelo** de todos os elementos da população;
3. Efectuar **em paralelo** N/2 vezes (N é o tamanho da população)
   - 3.1. Seleccionar 2 indivíduos para reprodução, I1 e I2
   - 3.2 Seleccionar 2 indivíduos para eliminação, I3 e I4
   - 3.2. Criar 2 novos indivíduos, por manipulação genética, I5 e I6
   - 3.3. Avaliar os novos indivíduos, I5 e I6, repondo-os na população, substituindo I3 e I4.
4. Voltar a 3. caso não seja satisfeito o critério de paragem;

Figura 3.3: Algoritmo Genético Assíncrono (AGA)

A reposição imediata dos indivíduos acabados de criar obriga a que sejam escolhidos quais os indivíduos que vão ser substituídos, para manter o tamanho da população constante. Esta escolha pode ser feita de várias formas, mas a utilizada foi a "selecção inversa", na qual se seleccionam os indivíduos a substituir aplicando o operador de selecção com a probabilidade de ganhar o melhor complementada em 1 (sendo essa probabilidade 0.8, passa a 0.2). Os indivíduos assim seleccionados são tendencionalmente os piores, por analogia com a selecção normal.

Existem outras formas mais complexas de eliminação de indivíduos, que envolvem o operador de *crowding* [Goldberg89]. Este operador elimina preferencialmente indivíduos semelhantes aos novos acabados de gerar, tentando manter a diversidade na população.

3.3.3. Implementação do Algoritmo Genético Assíncrono

O AGA adapta-se directamente a sistemas multiprocessador [Zeigler91]. A sua implementação directa pode ser feita por intermédio de um mecanismo de *threads*, no qual é criado um conjunto destes elementos de execução que repetidamente processam as épocas genéticas, conforme definidas neste algoritmo. O sistema operativo é o responsável pela alocação destes *threads* aos processadores do sistema. A comunicação é simplificada, uma vez que a população reside em memória partilhada por todos os processadores, não existindo assim atrasos de comunicação.

A implementação do AGA em sistemas multicomputador não é directa nem comum. Uma possível adaptação poderia consistir num modelo no qual um determinado nó seria o responsável por atribuir a execução remota de épocas a outros nós (através de um mecanismo de RPC, por exemplo). A população teria que residir toda no mesmo local, sendo enviados os cromossomas aos nós remotos e recolhidos os seus resultados.

Esta questão levanta uma dificuldade, relacionada com a perda de eficiência devido ao tempo gasto na comunicação entre os nós. A situação piora no caso das redes de *workstations* (em comparação com outros modelos mais acoplados de sistemas multicomputador), mas os atrasos podem não ser significativos, admitindo infraestruturas de comunicação de elevado desempenho e classes de problemas com demoradas funções de avaliação (relativamente ao tempo gasto na comunicação).

3.4. ALGORITMO GENÉTICO CELULAR

3.4.1. Definição do Modelo

O AG Celular [Whitley92] [Baluja92] caracteriza-se pela definição de uma matriz de indivíduos, na qual o cruzamento de um indivíduo está condicionado à sua vizinhança directa. A matriz representa toda a população, conforme ilustrado na figura 3.4.

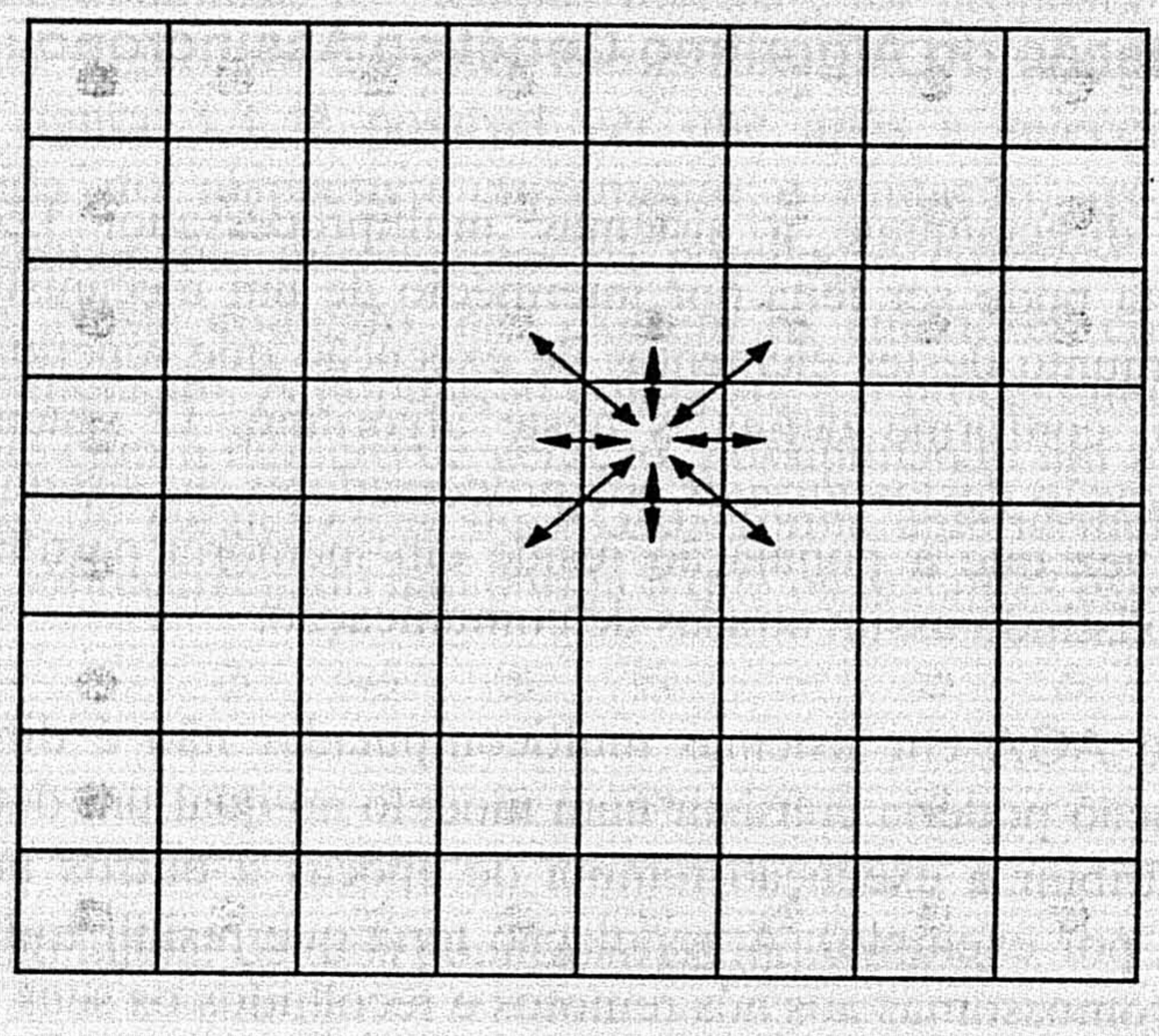


Figura 3.4: Modelo do AG Celular

Em cada elemento da matriz, que contém um dos indivíduos da população, corre um AG elementar semelhante ao AG original. A principal diferença é que a selecção está limitada a um dos 8 elementos da vizinhança, e o indivíduo a ser substituído é o próprio. A acção combinada de todos estes AG elementares, executados em paralelo em todos os elementos da matriz, conduz à evolução genética de toda a população.

A matriz deve ser entendida como a superfície de uma esfera, no sentido que os elementos da última coluna têm ligação directa com os indivíduos da primeira, e vice-versa. O mesmo se passa para as linhas. Existem algumas variantes a este algoritmo, nas quais a noção de vizinhança sofre algumas alterações. Em alguns casos, considera-se simplesmente 4 elementos da vizinhança directa, noutros a vizinhança é entendida num modelo tridimensional.

De qualquer forma, neste modelo é impossível a combinação de informação genética de indivíduos muito distanciados, a não ser após o decorrer de algumas gerações e por influência indirecta de todos os vizinhos intermédios. Por outro lado, todo o processamento é local, o que pode constituir uma vantagem em alguns casos relacionados com a implementação.

3.4.2. Implementação do Algoritmo Genético Celular

O AG Celular consiste em executar um conjunto de AG elementares, um por cada elemento da matriz. Para cada AG elementar é necessário um contexto de execução distinto, de preferência associado a um processador dedicado.

A adaptação do AG Celular a sistemas MIMD do tipo multiprocessador com um ambiente de *threads* é conseguida sem alterações ao algoritmo. Várias alternativas são possíveis, sendo um exemplo a atribuição de um *thread* a cada elemento da matriz. O sistema operativo, por sua vez, associa os *threads* aos processadores do sistema.

Como este modelo não tem tanta dependência da partilha de memória como o modelo do AGA (a computação neste caso é local), a sua adaptação a sistemas MIMD multicomputador não é tão penalizada como no caso do AGA. Aqui, cada nó poderia ficar responsável por uma área da matriz, trocando informação com os restante nós sempre que necessário (ao processar os cromossomas da "periferia" dessa área).

A implementação do AG Celular em sistemas massivamente paralelos do tipo SIMD é comum [Penfold91] [Talbi91] [Baluja92], devido ao facto de ser utilizada unicamente informação local. Em muitos destes sistemas, os processadores têm acesso privilegiado a memória local, onde pode ser guardado o cromossoma do respectivo indivíduo.

Deste ponto de vista, o AG Celular oferece uma vantagem relativamente ao AGA anteriormente descrito, já que neste último é maior a dificuldade de implementação em sistemas do tipo SIMD. Estes sistemas favorecem o processamento local, não oferecendo tantas vantagens à implementação de algoritmos que se baseiem em informação global como o AGA, que necessita de aceder a toda a população em cada ponto de execução.

Por fim, considerando implementações do AG Celular em sistemas computacionais com elevado número de processadores, o aumento da velocidade de execução do algoritmo está directamente relacionado com esse número. Desprezando tempos de comunicação e sincronização entre contextos de execução, num sistema com n processadores, o tempo de execução do algoritmo é dividido por n.

Por outro lado, nestes sistemas pode-se chegar ao limite em que se associa um processador a cada elemento da matriz de indivíduos. Assim, para uma população de

1024 indivíduos, num sistema semelhante ao descrito e com 1024 processadores, o tempo de computação em comparação ao AG original seria de aproximadamente 1024 vezes menor.

3.5. MODELO DE ILHAS

3.5.1. Definição do Modelo

Uma outra forma de distribuir o processamento de um AG é a divisão da população original em diversas subpopulações, que serão manipuladas localmente, partilhando informação genética através de trocas periódicas de informação. Este modelo é conhecido por Modelo de Ilhas [Whitley92], e é representado na figura 3.5..

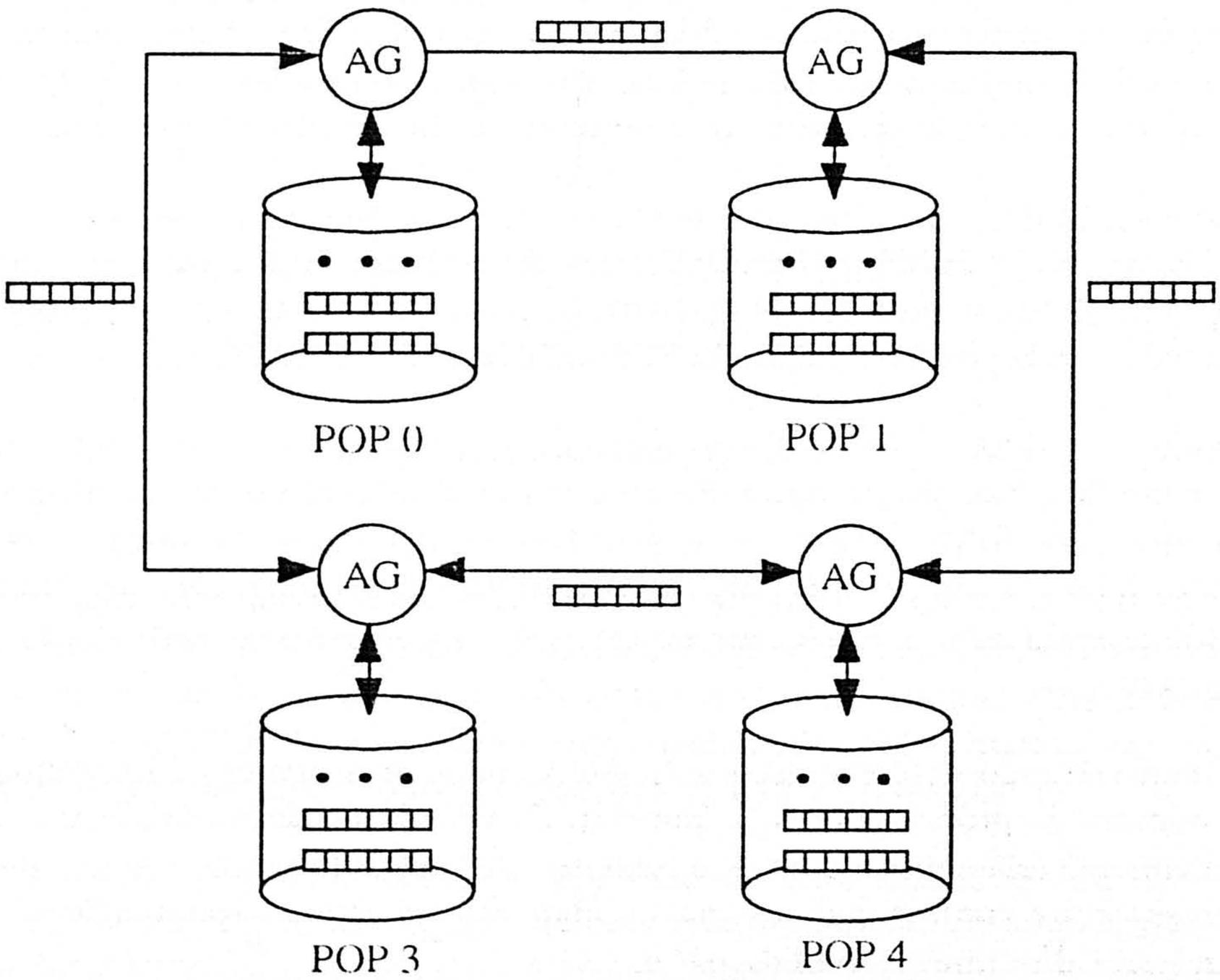


Figura 3.5: Modelo de Ilhas para o AG

A cada uma das subpopulações (associada ao respectivo algoritmo) dá-se o nome de *ilha*, na qual quase tudo se processa como no AG original. A única diferença é que, periodicamente, há troca de informação genética com as outras ilhas. Esta troca é realizada através da migração de indivíduos, quer da ilha para o exterior, emigração, quer do exterior (outras ilhas) para a própria ilha, imigração.

Cada um dos algoritmos evolui da forma já conhecida, até atingir um determinado número de gerações. Nessa altura, são escolhidos alguns indivíduos da população para emigrarem para outras populações. Na realidade, são cópias desses indivíduos que emigram, já que estes não são eliminados da população original.

Da mesma forma, periodicamente, o algoritmo verifica se existem alguns indivíduos provenientes de imigrações de outras ilhas, para integração local. Caso existam, estes são inseridos na população local, eliminando alguns dos que já lá se encontrem (no sentido de manter o tamanho da população constante).

A parametrização do modelo envolve um novo conjunto de factores, nomeadamente o número de indivíduos que emigram, o intervalo de tempo entre migrações, a escolha dos indivíduos que têm direito à migração e a escolha dos indivíduos substituíveis pelos imigrantes que vão chegando. Envolve também a definição da topologia das ligações entre as diferentes ilhas, assim como o número destas.

Estes factores serão analisados no próximo capítulo, no decorrer de um conjunto de experiências que compararam o desempenho deste modelo com outros modelos. Conforme se verá, a utilização do Modelo de Ilhas não introduz degradação na eficácia do processo genético.

Uma grande vantagem deste modelo relaciona-se com a facilidade de o combinar com outros modelos de AG distribuídos, como o AGA. Cada uma das ilhas executa um AG, podendo este ser o AG original, o AGA, ou o AG Celular, conforme as características do sistema computacional.

3.5.2. Parametrização e Topologia

Os diferentes aspectos que controlam o desenvolvimento dos AG cooperantes segundo um Modelo de Ilhas, relacionam-se com a escolha da informação genética a trocar, o seu percurso, a sua quantidade e a sua periodicidade.

O número de indivíduos a migrar é habitualmente definido em termos de percentagens da população total [Tanese90]. Valores demasiadamente pequenos levam a que o efeito da troca de informação seja reduzido, tendendo o sistema para um conjunto de populações de menor dimensão, um por cada ilha, completamente isolados. Os resultados das experiências realizadas e descritas no próximo capítulo demonstram que esta situação

introduz uma grande degradação da eficácia do algoritmo, justificando a necessidade de troca de informação entre as subpopulações.

Por outro lado, valores demasiadamente elevados no número de emigrantes tendem a aproximar demasiadamente as populações das diferentes ilhas, levando o sistema para um conjunto de subpopulações muito parecidas, que não teria muito melhor desempenho do que uma só dessas subpopulações. Neste caso, será também demonstrado que a diversidade conservada nas diferentes subpopulações é perdida, e o resultado é pouco satisfatório.

Os efeitos do número de emigrantes devem ser analisados tendo também em conta o intervalo entre as migrações, e eventualmente a topologia das ligações. A estratégia de escolha dos indivíduos a emigrar pode ser definida de diversas formas. As duas estratégias desenvolvidas, no decorrer deste trabalho, são a escolha aleatória e a escolha por selecção adaptada, com a probabilidade de ganhar o melhor igual a 1. Esta selecção equivale a escolher 2 indivíduos à sorte e a considerar definitivamente para migração o melhor deles.

Analogamente, a escolha dos indivíduos a substituir na população pelos imigrantes acabados de chegar foi executada por selecção com a respectiva probabilidade a 0, pelo que de entre dois quaisquer indivíduos escolhidos ao acaso por cada imigrante, é substituído definitivamente o pior deles.

Diversas combinações dos anteriores parâmetros e os seus resultados no AG serão analisados. É também considerado o impacto da topologia das ilhas, que consiste em definir qual o seu número, quantas ligações tem cada ilha e quais as ligações.

Valores típicos para o número de ilhas são 8, 16 e 32 ilhas. Números muito elevados de ilhas dão origem a subpopulações muito pequenas (assumindo a população global constante), o que origina convergências prematuras [Goldberg95]. Números muito pequenos de ilhas podem levar a subpopulações demasiadamente grandes, diminuindo as vantagens da distribuição do processamento.

Para a análise do impacto da forma de ligação entre ilhas neste modelo, foi criado um conjunto de 9 topologias de ligação entre 8 ilhas distintas, que pretende englobar diferentes características, úteis na análise dos resultados das simulações a desenvolver no próximo capítulo. As diferentes topologias são esquematizadas na figura 3.6.

A topologia Aleatória é representada na figura 3.6 como um possível exemplo, uma vez que esta varia de simulação para simulação. As ligações simples, representadas a tracejado, são unidireccionais (por exemplo, na topologia em Anel Simples). As ligações bidireccionais são representadas em linhas finas, e as ligações duplas, na qual há dois percursos de migração coincidentes, são representadas por linhas mais grossas. Estas últimas só ocorrem na topologia Aleatória.

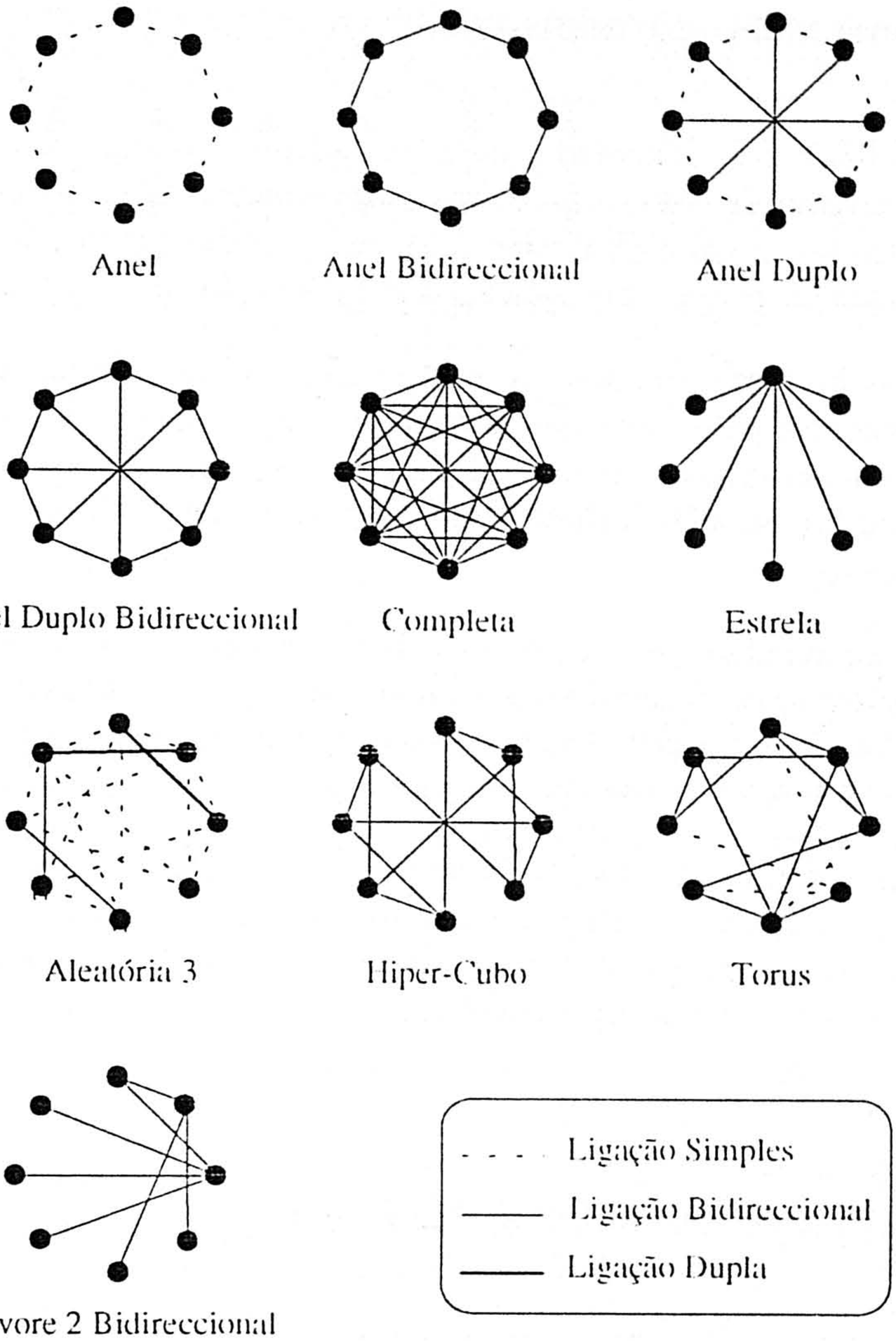


Figura 3.6: Diferentes topologias utilizadas no Modelo de Ilhas

Numa primeira análise, espera-se que uma boa topologia possibilite uma boa expansibilidade das soluções locais, ou seja, não deve criar nem grandes obstáculos nem grandes assimetrias na continua redistribuição global do material genético. Claramente, uma topologia que divida as populações em grupos sem comunicação não faz sentido, da mesma forma que uma topologia em Estrela Unidireccional (a da figura é bidireccional) condiciona negativamente a evolução, já que a única vantagem que permite é a colecção de soluções numa só população, sem permitir a partilha entre as restantes.

Não se conhece nenhuma regra absoluta para a escolha de uma boa topologia, e a sua influência depende muito do tipo do problema e do número de populações. De qualquer modo, a análise das simulações executadas permite tirar algumas conclusões.

3.5.3. Implementação do Modelo de Ilhas

O Modelo de Ilhas foi concebido para se adaptar a sistemas multicomputador, especialmente em redes de *workstations*, nos quais em cada nó é processada uma ou mais ilhas. Nestes ambientes, como os diversos nós estão desacoplados, contendo memória e gestão de processos próprios, a comunicação é estabelecida por troca de mensagens.

Como este modelo tem por base o processamento local, dentro de cada ilha, o isolamento provocado pelos nós autónomos não tem impacto negativo [Bianchini93] [Gordon93]. A cooperação entre ilhas é feita por troca periódica de informação, sempre entre duas ilhas, o que se adapta directamente a paradigmas de computação baseados em trocas de mensagens.

Não há assim a necessidade de memória partilhada entre os processadores, ao contrário de outros modelos de AG distribuídos. Neste modelo, um determinado nó não tem necessidade de "ver" toda a população, necessitando apenas da sua população local, que é periodicamente reforçada com cópias de elementos de outras populações.

Os ambientes de computação distribuída úteis para a implementação desta arquitectura baseiam-se no paradigma de troca de mensagens. Alguns exemplos são os *sockets*, sistemas RPC e outros menos genéricos, como o PVM. A adaptação deste modelo a outras arquitecturas de computação distribuída é pouco comum, embora possível.

3.6. POPULAÇÕES SOBREPOSTAS

3.6.1. Definição do Modelo

O modelo das Populações Sobrepostas [Baluja92] engloba um conjunto de populações, cada qual com o seu ciclo genético, cuja comunicação se efectua através da sobreposição de uma parte das populações vizinhas. As sobreposições fazem-se em cascata, na qual a população i partilha uma parte com a população $i+1$, fechando o ciclo através da ligação da última população à primeira, conforme ilustrado na figura 3.7.

Neste modelo, o número de indivíduos realmente partilhados depende do processo de selecção e da quantidade de população que é sobreposta. Em termos médios, sendo s a percentagem de população sobreposta e n o tamanho da população, cada um dos subalgoritmos "vê" entrar e sair $n \cdot s$ indivíduos por geração, uma vez que o acréscimo de população é metade do intervalo de sobreposição em cada um dos dois extremos.

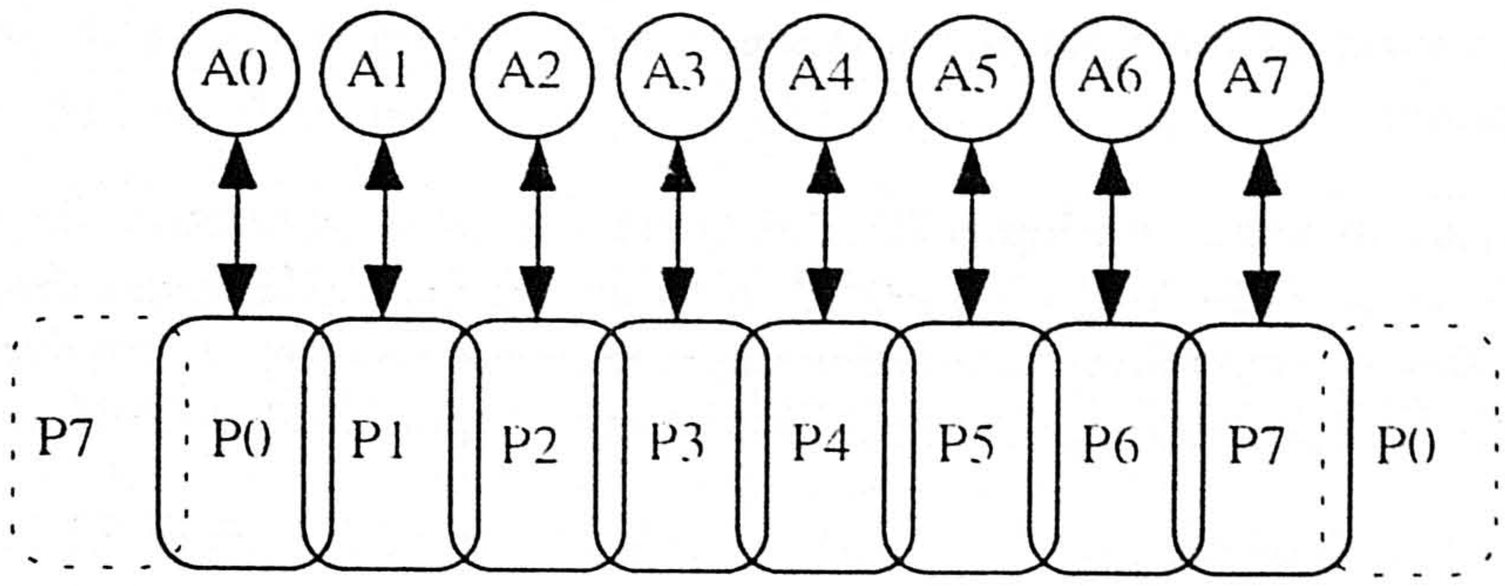


Figura 3.7: Modelo das Populações Sobrepostas

Na comparação com o Modelo de Ilhas, é lógico considerar a topologia de Anel Bidireccional, por ser a que mais se assemelha com o corrente modelo. No entanto, neste caso, a "migração" ou troca de informação entre populações é muito mais frequente, uma vez que ocorre em todas as gerações. Para haver uma equivalência entre a quantidade de informação que flui nos dois modelos, a percentagem de sobreposição deve ser equivalente a:

$$s = \frac{N}{T}$$

Assumindo os valores mais indicados para N e T, considerados no próximo capítulo, de 10% e 30 gerações, respectivamente, a percentagem de sobreposição é cerca de 0.33%, um valor muito baixo. Mesmo com valores bem diferentes para aquelas quantidades, N=30% e T=10, o valor de s não passa de 3%. De qualquer forma, os valores de s que asseguram os melhores resultados, situam-se por vezes próximos dos 10% (conforme se verá no próximo capítulo).

Nas experiências a realizar com este modelo, são considerados os efeitos comparados da variação da percentagem de sobreposição e do número de subpopulações. Em cada subpopulação é executado o AG original, ou uma sua variante, como por exemplo o AGA, possibilitando desta forma a combinação deste modelo com outros modelos de AG distribuídos.

3.6.2. Implementação do Modelo das Populações Sobrepostas

O Modelo de Populações Sobrepostas baseia-se no conceito de memória partilhada. Ao contrário do Modelo de Ilhas, um subpopulação tem necessidade de aceder a elementos de populações vizinhas em qualquer instante, não sendo possível faze-lo somente em

de populações vizinhas em qualquer instante, não sendo possível faze-lo somente em intervalos regulares.

Assim sendo, os sistemas multiprocessador oferecem uma alternativa directa para o modelo. Cada processador fica responsável por uma ou mais subpopulações, estando a população global na memória partilhada. Em cada processador, é possível o acesso directo aos seus indivíduos assim como aos indivíduos de populações vizinhas.

No que se refere à implementação em sistemas multicomputador, em particular em redes de *workstations*, a situação é mais complicada. A adaptação deste modelo a ambientes baseados no paradigma de troca de mensagens não é directa, obrigando a esforços adicionais de implementação.

Uma alternativa a esta situação é o recurso a ambientes de computação distribuída que simulem uma memória global partilhada entre os diversos nós do sistema. Estes ambientes mapeiam todos os acessos à memória global em trocas de mensagens, escondendo todos os pormenores e possibilitando uma mais fácil adaptação deste modelo.

No que respeita à fracção do tempo total de computação gasto na comunicação para este último caso, tudo depende do tipo de problema. Se a função de avaliação for complexa, e atendendo a que a percentagem de sobreposição de populações é pequena, esse tempo pode ser desprezado em sistemas com eficientes infraestruturas de comunicação.

3.7. RESUMO

O conjunto de modelos de AG distribuídos apresentados neste capítulo são implementáveis em diversos ambientes, com maior ou menor facilidade. No sentido de clarificar as associações mais lógicas, é apresentada uma tabela, na qual são apenas considerados os sistemas multicomputador e multiprocessador mais divulgados, ou seja, as redes de *workstations*.

Nestes sistemas, as *workstations* podem conter mais do que um processador, inserindo-se assim na classe dos sistemas multiprocessador. O sistema global, constituído por todas as *workstations* e pela infraestrutura de comunicação, é designado por sistema multicomputador. A tabela 3.1 mostra as associações possíveis entre modelos de AG e sistemas de computação, evidenciando o ambiente mais apropriado.

Modelo/Sistema	Multiprocessador	Multicomputador
AGA	**Threads**	Sockets, RPC, PVM
AG Celular	**Threads**	Linda
Modelo de Ilhas	**Multiprocesso+Sockets**	**Sockets**, RPC, PVM
Modelo das Pop. Sobrepostas	**Multiprocesso**, Threads	**Linda**

Tabela 3.1: Modelos de AG distribuídos e ambientes de distribuição

Os modelos de AG que se baseiam numa só população, que são o AGA e o AG Celular, adaptam-se directamente a sistemas multiprocessador, de preferência sobre um mecanismo de *threads* (dada a sua granularidade, devem ser implementados em estruturas de processamento "leves"). A sua adaptação a sistemas multicomputador é possível, mas obriga a custos adicionais de implementação e comunicação, muitas vezes socorridos por ambientes que proporcionem memória global virtual (por exemplo, o ambiente *Linda*).

Por outro lado, os modelos que envolvem mais de uma população, o Modelo de Ilhas e o Modelo das Populações Sobrepostas, têm de ser analisados isoladamente. O primeiro, adapta-se directamente a qualquer um dos dois sistemas de computação referidos. Tanto os sistemas multiprocessador como os sistemas multicomputador têm mecanismos de troca de mensagens entre processos, sendo os *sockets* os mais genéricos, por funcionar da mesma forma em ambos.

O segundo destes modelos, o Modelo das Populações Sobrepostas, adapta-se melhor a sistemas multicomputador, uma vez que estes possuem memoria global partilhada. Neste caso, a implementação é simples, com o recurso a *threads* ou simplesmente aos processos convencionais do sistema operativo. A sua adaptação a sistemas multiprocessador é também possível com algum custo adicional, por recurso aos já referidos ambientes de computação que proporcionam memória global virtual.

Por fim, salienta-se que é possível combinar diferentes modelos de AG distribuídos, conseguindo assim um melhor aproveitamento dos recursos de computação. Por exemplo, a utilização do Modelo de Ilhas em conjugação com o AGA (cada ilha corre um AGA), explora directamente o sistema na globalidade (por se adaptar a um ambiente de *workstations* em rede) e o inerente paralelismo das *workstations* multiprocessador (por explorar a possibilidade de multiprocessamento destas).

SIMULAÇAO, ANALISE E RESULTADOS

4.1. INTRODUÇÃO

No capítulo anterior foi apresentado um conjunto de modelos do AG, adaptáveis a ambientes de computação distribuída. Todos esses modelos divergem do AG original, seguindo diferentes processos de pesquisa de soluções, o que implica diferenças na sua eficiência e eficácia.

Estas diferenças devem-se a dois factores distintos, nomeadamente à alteração do algoritmo em si e à sua implementação distribuída. O primeiro factor leva a que o processo de geração de soluções seja próprio de cada modelo, o que se reflecte no tempo necessário para atingir as soluções (traduzido pelo número de gerações) e na sua qualidade.

O segundo factor relaciona-se com a implementação dos referidos modelos de AG em ambientes de computação distribuída. Como estes modelos foram concebidos para tirar proveito do processamento distribuído, o tempo necessário para a sua execução pode ser dividido pelos diversos processadores do sistema, podendo assim diminuir em grande escala.

O objectivo deste capítulo é analisar a qualidade dos diversos modelos de AG propostos, sob o ponto de vista do primeiro dos factores referidos, ou seja, focando os aspectos relacionados com a forma da pesquisa genética sem considerar aspectos dependentes da implementação distribuída (estes serão analisados no próximo capítulo). O efeito da variação de um conjunto de factores de controle e parametrização são também tidos em consideração.

A análise aqui desenvolvida tem sempre como base de comparação o AG original, sendo a qualidade de cada modelo aferido em termos relativos, comparando a qualidade das soluções obtidas (eficácia) e o tempo necessário para as obter (eficiência).

O conceito de tempo em todo este capítulo está directamente relacionado com o número de gerações desenvolvidas pelo algoritmo de cada modelo. Desta forma, as comparações evidenciam sempre a qualidade do algoritmo em si, uma vez que reflectem o número de processamentos genéticos (proporcional ao número de gerações) desenvolvidos até ser atingida uma determinada qualidade de solução. O número de gerações indica o tempo gasto no processamento genético, ou seja, o tempo gasto na optimização.

O tempo real de processamento não é aqui considerado. Este tempo depende não só da qualidade da optimização em si, mas também da implementação distribuída de cada modelo (que engloba aspectos como a infraestrutura de comunicação). A implementação de cada um destes modelos em ambientes de computação distribuída pode influir significativamente no seu tempo de processamento, sendo a sua consideração independente da comparação da qualidade de pesquisa. No entanto, não deixa de ser um factor de importância fundamental, que será considerado em pormenor no próximo capítulo, conforme já referido.

Para facilitar a análise dos diversos modelos, independentemente do subsistema de computação distribuída, foram desenvolvidos neste trabalho um conjunto de simuladores de AG distribuídos. Os simuladores proporcionam uma distribuição fictícia, própria de cada modelo, na qual o algoritmo é executado como se estivesse num ambiente de computação distribuída. A única diferença sentida na sua execução é o tempo real de processamento, que não é divido pelos processadores que eventualmente existam no sistema.

A comparação dos diversos modelos é feita com o recurso a uma base de teste, que define um conjunto de problemas de optimização conhecidos. Os diferentes resultados obtidos e a sua própria evolução, em função de diversos factores de controle de execução, serão os aspectos a considerar.

No que respeita à estrutura do capítulo, na primeira secção, é apresentada a base de teste, que é constituída por um conjunto de problemas de optimização pertencentes a duas classes distintas (problemas de optimização numérica e problemas de optimização combinatória). Para cada problema, é descrita a forma de codificação das soluções em cromossomas e a forma da função de avaliação.

A secção seguinte, analisa os resultados da aplicação do AG original aos problemas apresentados, salientando alguns aspectos de parametrização. Os resultados assim obtidos são a base de comparação para todos os restantes modelos analisados.

Seguidamente, são apresentados e analisados, em secções separadas, os resultados obtidos pela execução dos diversos modelos de AG distribuídos quando aplicados, pelos respectivos simuladores, à referida base de teste, tendo em conta diversos factores de

controle. Os modelos em questão são o AG Assíncrono, o AG Celular, o Modelo de Ilhas e o Modelo das Populações Sobrepostas. Por fim, na última secção, é resumida a análise comparativa entre todos eles.

4.2. BASE DE TESTE

A utilização de funções de optimização bem conhecidas, na análise de modelos alternativos de AG, é frequentemente utilizada por permitir uma análise comparativa da qualidade dos algoritmos, deixando o problema de optimização em si para segundo plano [Gordon93] [Baluja92]. Desta forma, os resultados habituais já são conhecidos, pelo que é facilitada a avaliação do desempenho da variante do algoritmo em análise.

Os problemas mais utilizados são funções matemáticas características [Goldberg89] [Gordon93], muitas vezes concebidas para evidenciar determinadas características do AG, e conhecidas classes de problemas de optimização [Davis91] [Baluja92], para os quais já existem diversos resultados de análises anteriores.

Os problemas considerados neste capítulo, que constituem uma base para o teste dos diversos modelos de AG distribuídos a analisar, são problemas de optimização numérica e combinatória. Nos primeiros, incluem-se três conhecidas funções reais multidimensionais com diferentes características e graus de dificuldade. Nos segundos, incluem-se dois conhecidos problemas de optimização combinatória.

Com a diversidade existente nesta base de teste, pretende-se que os resultados tenham alguma generalidade, evitando influenciar as conclusões com resultados de problemas específicos, que contêm sempre características particulares. Os problemas têm diferentes graus de dificuldade, sendo os dois primeiros problemas de optimização numérica relativamente simples e o terceiro complexo, enquanto que os problemas de optimização combinatória incluem um problema muito simples e um muito complexo.

O grau de dificuldade de um problema é aferido com base no volume de processamento necessário para obter o óptimo, ou uma solução aceitável em alternativa. Para muitos problemas, é fácil chegar perto de boas soluções, sendo no entanto muito difícil melhorá-las, ainda que residualmente.

Em todas as experiências realizadas, os resultados referem-se a médias de 20 execuções, número suficiente para minimizar a influência de resultados pouco comuns que possam eventualmente surgir numa ou noutra execução em particular.

4.2.1. Problemas de Optimização Numérica

Os problemas de optimização numérica da base de teste consistem em encontrar o ponto óptimo de uma função real multidimensional. A optimização numérica aplicada a funções matemáticas puras não constitui novidade, tendo sido descrito no segundo capítulo (secção 2.1.3) a forma de adaptar o AG a estes problemas. Ao contrário daquele exemplo, para muitos problemas de optimização numérica não existe uma solução óptima conhecida à priori, nem sequer um método garantidamente bom de a encontrar, justificando assim a aplicação do AG.

No que se segue, é aplicado o AG a funções reais bem conhecidas, no sentido de estudar o seu comportamento. As funções a seguir apresentadas foram utilizadas em diversos trabalhos de análise de AG [Gordon93]. As suas curvas servem para evidenciar as características do método de pesquisa desenvolvido por estes algoritmos, sendo aqui designadas por *F6*, *F7* e *F8*. As suas equações de definição, com os respectivos domínios, são as seguintes:

$$\text{F6:}\quad f(x_i|_{i=1,20}) = 200 + \sum_{i=1}^{20} x_i^2 - 10\cdot\cos(2\pi\cdot x_i),\quad x_i \in [-5.12, 5.12]$$

$$\text{F7:}\quad f(x_i|_{i=1,10}) = V + \sum_{i=1}^{10} -x_i\cdot\sin(\sqrt{|x_i|}),\quad x_i \in [-512, 512]$$

$$\text{F8:}\quad f(x_i|_{i=1,10}) = \sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} \cos(\frac{x_i}{\sqrt{i}}) + 1,\quad x_i \in [-512, 512]$$

Para todas as funções, pretende-se a minimização no domínio definido. As respectivas funções de avaliação têm de levar o AG na direcção da minimização das funções, rodeando a sua inerente procura do máximo (o AG é um algoritmo de maximização). Para tal, basta considerar o simétrico das funções reais, efectuando uma minimização indirecta, apresentando todos os resultados também pelo seu simétrico (tendo em conta que para uma qualquer função real *f*, *min(f)=-max(-f)*).

A codificação das soluções candidatas nos cromossomas é equivalente à forma usada no segundo capítulo. A função *F6* contém 20 variáveis, cada uma das quais podendo assumir valores entre -5.12 e 5.12. Assumindo a precisão de 0.01, são necessários 10 *bits* por variável, o que totaliza 200 *bits* por cromossoma. Pelo mesmo raciocínio, as funções *F7* e *F8* necessitam, respectivamente, de 100 e 200 *bits*, sendo nestes casos os valores limite de -512 e 512 e a precisão igual a 1.

Qualquer uma das três funções apresenta um espaço de estados significativamente grande (2^{200} para *F6* e 2^{100} para *F7* e *F8*). A função *F6*, conhecida pela função de *Rastringin*, caracteriza-se por um grande conjunto de óptimos locais, que colocam uma barreira intransponível a alguns métodos clássicos de optimização baseados em pesquisa por gradiente, como o *Hill Climbing* [Mitchell92]. É um problema relativamente fácil para o AG. A sua versão bidimensional é representada na figura 4.1.

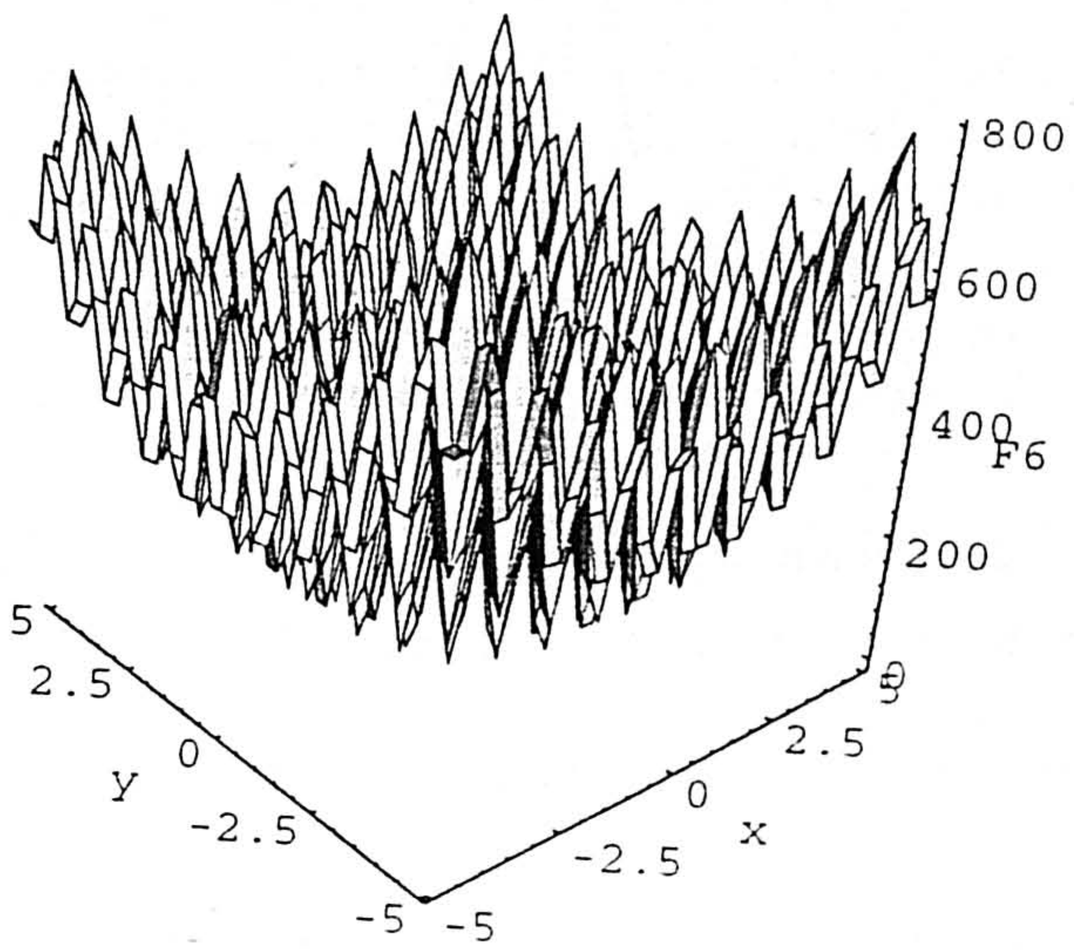


Figura 4.1: Versão bidimensional da superfície definida pela função de *Rastringin* (*F6*)

A função *F7*, função de *Schwefel*, é também relativamente simples para o AG, mas contém um segundo óptimo, muito distanciado do óptimo global. O valor de *V* (ver equação de definição de *F7*) ajusta o valor do óptimo global, e nestas experiências foi fixado em 4189.829, para colocar o mínimo da função em zero. A função, na versão bidimensional, tem a forma representada na figura 4.2.

A função *F8*, função de *Griewangk*, é uma função complicada para o AG, devido à grande interdependência entre subcadeias, provocada pelo produtório cruzado das suas variáveis. Este facto origina a existência de uma grande quantidade de soluções quase óptimas, dificultando o isolamento da melhor solução. A figura 4.3 dá uma ideia da sua forma (na versão bidimensional).

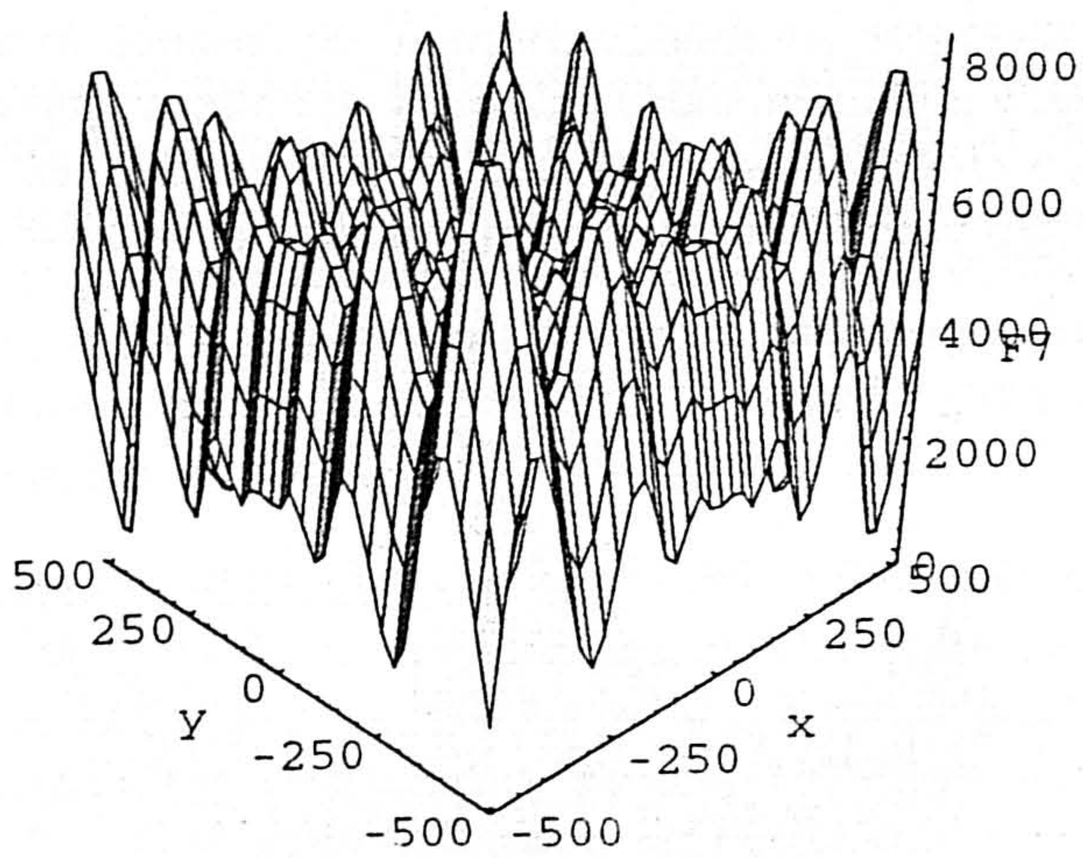


Figura 4.2: Versão bidimensional da superfície definida pela função de *Schwefel* (*F7*)

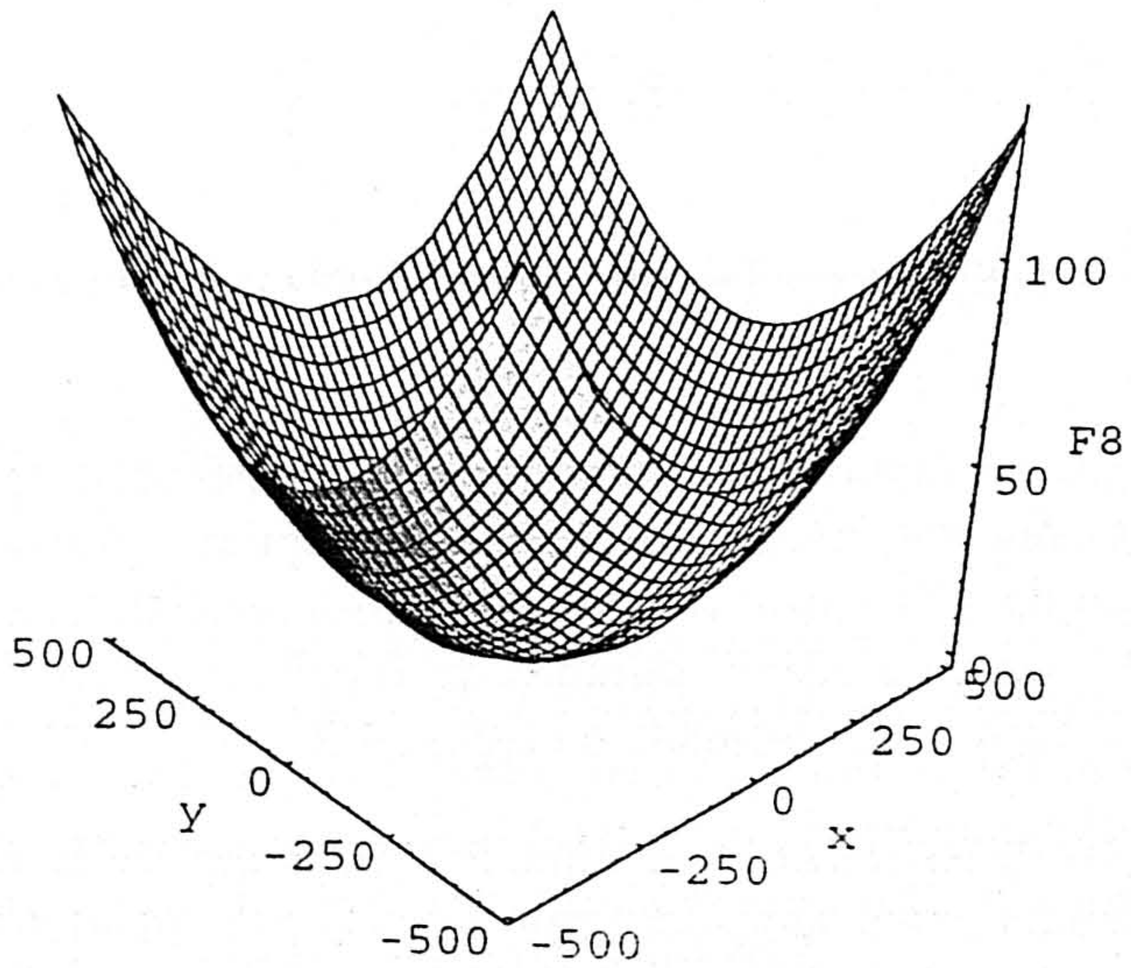


Figura 4.3: Versão bidimensional da superfície definida pela função de *Griewangk* (*F8*)

4.2.2. Problemas de Optimização Combinatória

Os dois problemas de optimização combinatória analisados são os conhecidos problemas da Mochila 01 e do Caixeiro Viajante. A aplicação do AG a estes problemas não constitui novidade [Khuri94].

4.2.2.1. Problema da Mochila 01

O Problema da Mochila 01 (PM01), representa uma conhecida classe de problemas de optimização, abordada em diversos trabalhos por variadas técnicas [Khuri94]. Esta classe de problemas é definida por um conjunto de objectos, cada um dos quais caracterizado por um valor de benefício, e por um valor de custo (neste caso interpretado como peso). O objectivo consiste em encontrar um subconjunto de objectos que maximize o benefício, tendo como restrição um determinado valor limite para o custo (ou peso). Assim, a cada objecto do conjunto inicial, é associada uma variável binária de decisão, que indica a sua inclusão ou não inclusão nesse subconjunto.

Considerando P_i e W_i o benefício e o custo (ou peso), respectivamente, do objecto i do conjunto original de n objectos, e tendo em conta que X_i, como variável de decisão binária, representa a decisão de incluir ou não o objecto i no conjunto final, o problema pode definir-se formalmente da seguinte forma:

$$\text{Maximizar } \sum_{i=1}^{n} X_i \cdot P_i, \text{ sujeito a } \sum_{i=1}^{n} X_i \cdot W_i \leq M$$

A codificação deste problema num cromossoma é directa. Para tal, associa-se a cada elemento i do cromossoma a variável de decisão X_i. Utilizando cromossomas binários, não há necessidade de transformação entre os valores das variáveis de decisão e os elementos da cadeia de informação. O mapeamento é directo, tendo o cromossoma o tamanho de n *bits*.

A função de avaliação é também muito simples. Percorrendo a cadeia, acumula-se o valor de benefício de todos os elementos considerados para o conjunto final. Caso o valor ultrapasse M, a solução é inválida e deve ser penalizada ou ignorada. Uma das formas de penalizar más soluções é subtrair um determinado valor, suficientemente elevado para evitar que esta seja propagada em futuras recombinações.

Outra forma de evitar estas situações, é conseguida ao efectuar a soma só até se atingir o valor máximo, ignorando os subsequentes elementos. Esta foi a solução escolhida, por ser a mais simples e atingir bons resultados, de acordo com experiências realizadas para o efeito. A respectiva função de avaliação soma os benefícios dos elemento com X_i activo, até que seja atingido M. A partir desse instante, todos os restantes elementos são ignorados, ficando uma parte do cromossoma por avaliar.

Note-se que mesmo em problemas em que as soluções envolvem a escolha de objectos perto do fim do cromossoma, o algoritmo não as exclui. Nestes casos, os cromossomas que contêm mais zeros nos primeiros *bits* acabam por ter um desempenho superior, possibilitando a inclusão daqueles objectos pretendidos, próximos do final.

O problema concreto do tipo PM01 considerado neste trabalho é definido por um conjuntos de objectos, com 80 elementos, com os valores de benefício e custo gerados aleatoriamente. O problema é bastante simples para o AG e passa a ser designado por *M80*. Os cromossomas têm 80 *bits* de comprimento, sendo o espaço de estados de 2^{80}.

4.2.2.2. Problema do Caixeiro Viajante

Uma outra classe de problemas, amplamente conhecida e abordada na disciplina de optimização, é o Problema do Caixeiro Viajante (PCV). O seu objectivo é encontrar o menor percurso global, que passe por todos os pontos (ou cidades, na sua versão original) de um dado conjunto, sabendo todas as distâncias entre eles.

O PCV pode ser reduzido a um problema de permutações, uma vez que consiste em encontrar uma sequência correcta num conjunto de elementos [Davis91]. Deste modo, a resolução do problema é uma reordenação do conjunto original dos pontos que o definem, de forma a minimizar a soma das distâncias de cada passo do percurso.

No problema analisado, o percurso é constituído por 96 pontos, que representam um conjunto real de cidades em África [Orlib94]. As distâncias entre os pontos são distâncias reais, medidas em linha recta (em kilómetros). O problema é bastante complexo, seja qual for o método, e será designado no texto que se segue por *PCV96*.

A codificação do problema *PCV96* em cromossomas consiste em representar uma permutação do seu conjunto dos 96 pontos. A codificação de permutações não é tão directa como nos problemas anteriores e foram experimentadas duas abordagens. A primeira, utiliza uma codificação em cromossomas binários, servindo-se de um artifício de mapeamento [Ranito92]. A segunda forma, utiliza o que se designa por cromossomas numéricos [Davis91].

A codificação escolhida foi a codificação em cromossomas numéricos, por ser a mais comum. Nesta codificação, a representação é simplificada à custa da redefinição dos operadores de cruzamento e mutação. Nesta representação, os elementos do cromossoma não são *bits*, mas sim quantidades numéricas inteiras, sendo estes conhecidos por cromossomas numéricos em vez de cromossomas binários. No presente exemplo, cada elemento do cromossoma pode guardar um pequeno número entre 0 e 255, que representa o código de uma cidade (este elemento é constituído por 8 *bits*, mas a diferença é que o elemento em si não pode ser subdividido pelos operadores genéticos).

Os cromossomas contêm assim as permutações candidatas à solução. A principal diferença, relativamente a outros cromossomas já analisados, consiste no facto de estes não poderem conter repetições de elementos. A repetição de um elemento numa permutação significa uma dupla passagem pelo mesmo ponto na solução representada, o que constitui uma situação ilegal.

Neste contexto, os operadores de cruzamento e mutação têm de ser alterados, de modo a evitar a geração de cromossomas ilegais. Em primeiro lugar, devem operar sobre quantidades numéricas e não binárias. Em segundo lugar, devem preservar a lógica das permutações, ou seja, os cromossomas gerados devem ser permutações válidas, sem repetições e contendo todos os elementos.

Existem várias propostas para a implementação deste tipo de operadores [Davis91], sendo as mais conhecidas aqui utilizadas, que consistem numa analogia ao cruzamento uniforme para cromossomas numéricos e mutação por troca de posição de pares de elementos na sequência.

4.3. APLICAÇÃO DO ALGORITMO GENÉTICO ORIGINAL

4.3.1. Resultados dos Problemas de Optimização Numérica

A parametrização do AG, quando aplicado às três anteriores funções reais, *F6*, *F7* e *F8*, teve em conta questões relacionadas com a complexidade do problema e a dimensão do espaço de estados de soluções. Em todos os casos foram utilizados parâmetros convencionais [Goldberg89] [Goldberg91], com a probabilidade de cruzamento de 0.8 e a probabilidade de mutação de 0.005 (para uma população de 1024 indivíduos, em média 5 indivíduos são alterados por cada geração). A selecção foi implementada por torneio estocástico com a probabilidade de 0.8 de vencer o melhor.

A análise do impacto destes valores está fora do âmbito deste trabalho. No entanto, o efeito do tamanho da população é significativo, justificando alguma atenção. Os gráficos das figuras 4.4, 4.5 e 4.6 demonstram os diferentes desempenhos do algoritmo para cada uma das funções, ilustrando a evolução da melhor solução em função do número de geração, para três diferentes populações, com 128, 1024 e 8192 indivíduos (as curvas ilustram médias de 20 execuções).

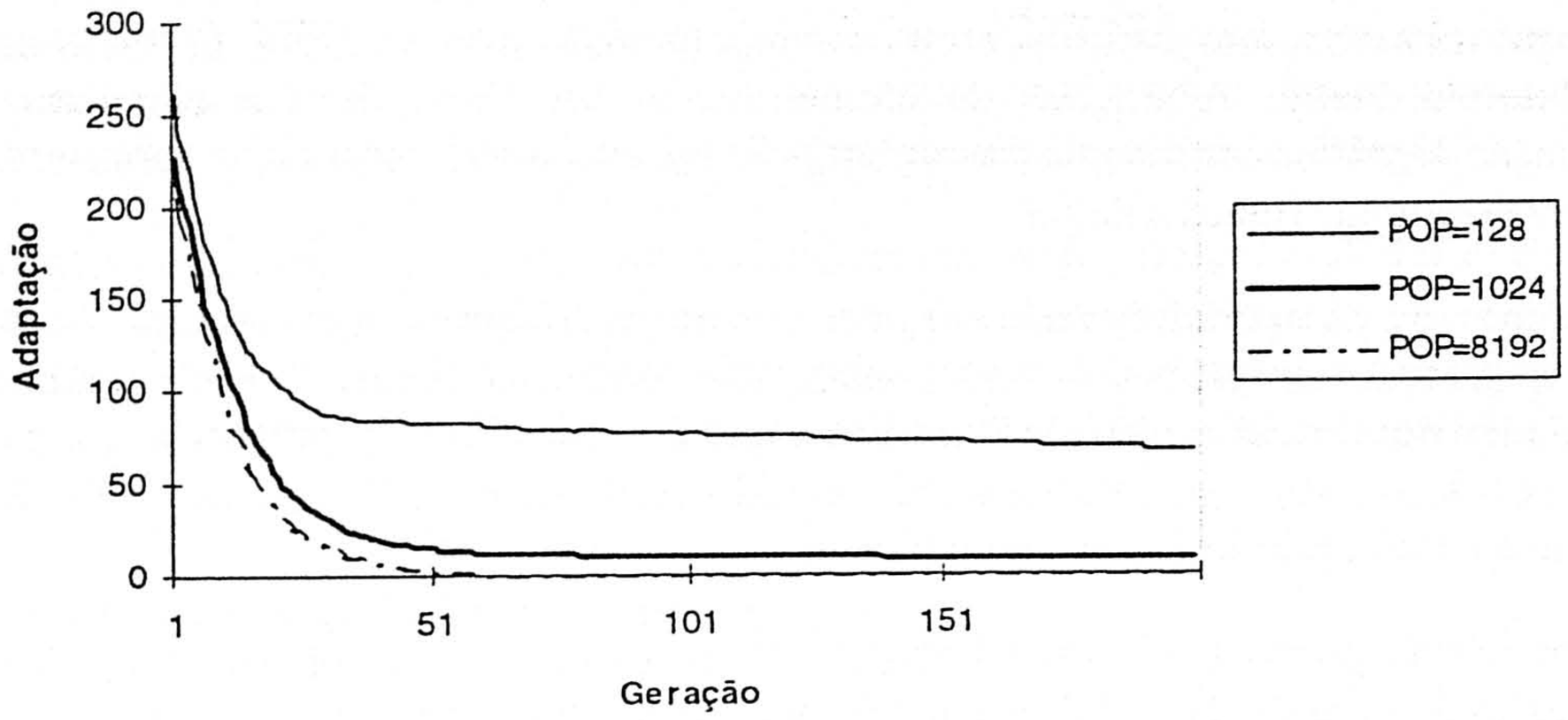


Figura 4.4: Desempenho do AG na função *F6*.

Em qualquer dos gráficos, é visível que quanto maior a população, mais rápida é a convergência (traduzida pela grau de aproximação da curva em relação ao valor zero de adaptação), e melhor é a solução final. Para melhor salientar os resultados finais obtidos nos três problemas, a tabela 4.1 representa as soluções encontradas nas anteriores simulações, ao fim de 600 gerações.

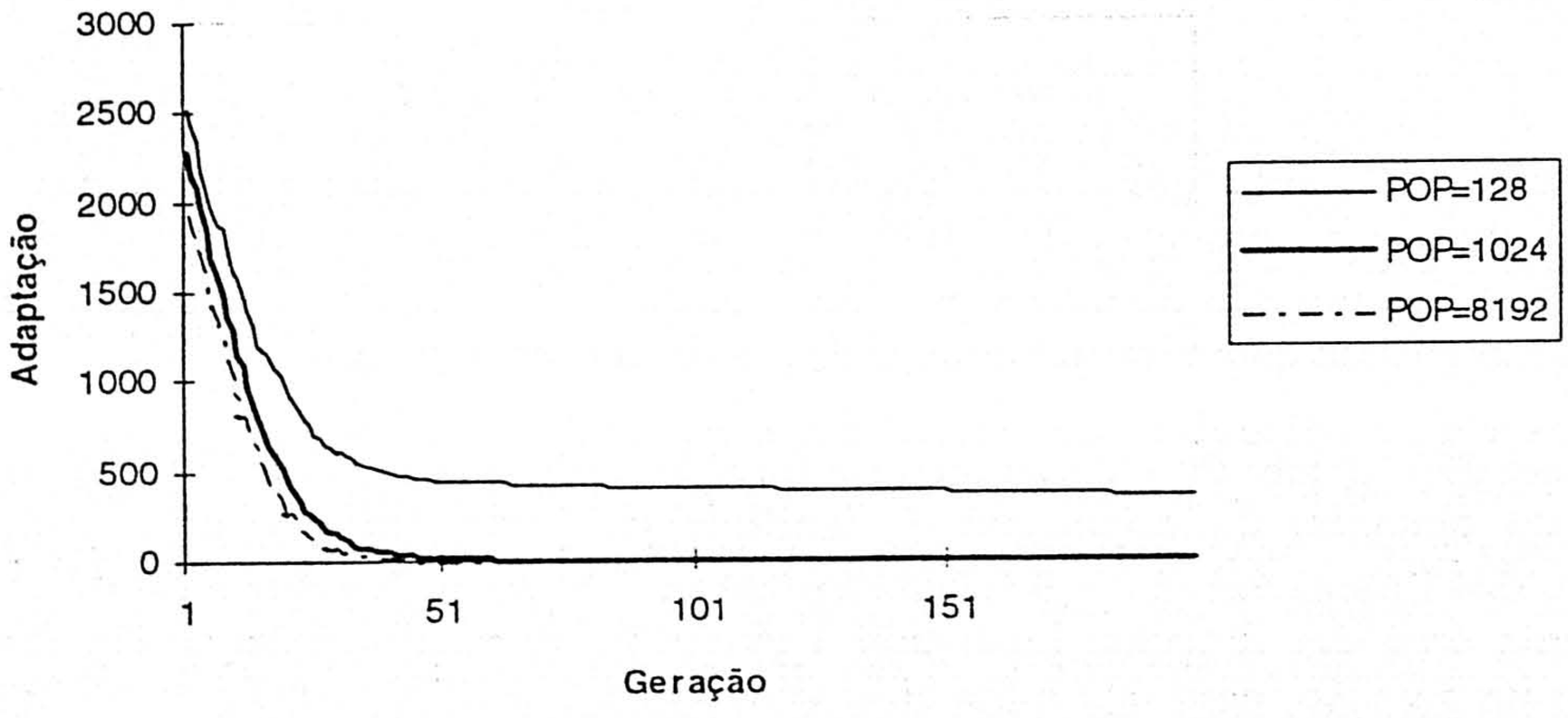


Figura 4.5: Desempenho do AG na função *F7*.

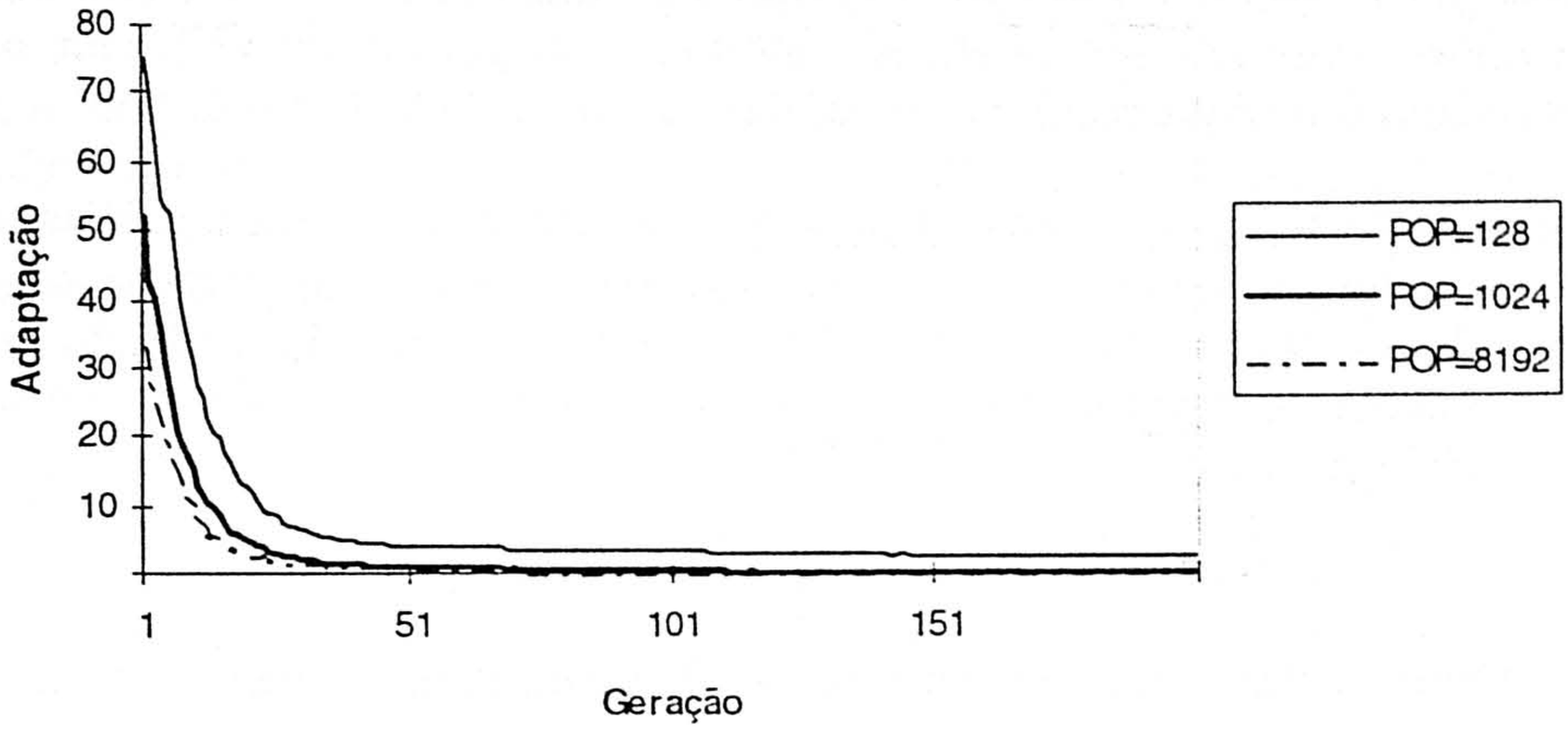


Figura 4.6: Desempenho do AG na função F8.

	POP = 128	POP = 1024	POP = 8192
F6	44.70	6.06	0.00
F7	451.46	2.99	0.00
F8	1.28	0.17	0.01

Tabela 4.1: Soluções encontradas na resolução de *F6*, *F7* e *F8*

Em qualquer das funções, a população de 128 indivíduos fica muito longe do óptimo e das outras duas curvas. A diferença entre a população de 1024 indivíduos e de 8192 não é tão acentuada.

A escolha do tamanho da população depende da qualidade do resultado que se pretende atingir. De facto, o óptimo (que é zero, para as 3 funções) foi atingido para *F6* e *F7*, por todas as execuções da população mais numerosa de 8192 indivíduos. No que respeita à função mais complexa, *F8*, o óptimo não chegou a ser atingido. A escolha do tamanho a utilizar em futuras simulações destes problemas, para comparações com outros modelos de AG, é o de 1024 indivíduos, por atingir resultados aceitáveis e não consumir demasiado tempo de execução.

Repare-se ainda que embora o aumento da população tenha o efeito de melhoria da qualidade final da solução, é de uma forma cada vez mais penosa que o consegue, no

que respeita ao tempo de processamento do algoritmo. A partir de determinada altura, um pequeno acréscimo de qualidade implica um grande aumento no esforço de processamento (que é proporcional ao número de indivíduos da população).

O número de gerações necessárias para atingir a convergência varia com o problema e com o tamanho da população. Para os anteriores exemplos, embora as execuções tenham sido mantidas por 600 gerações, a partir de um determinado número de gerações a evolução é muito pequena. Por esta razão, os gráficos ilustram somente as primeiras 200 gerações.

4.3.2. Resultados dos Problemas de Optimização Combinatória

4.3.2.1. Resultados do Problema da Mochila 01

O resultado da aplicação do AG original ao problema *M80* é ilustrado na figura 4.7, para diferentes tamanhos da população de indivíduos. Mais uma vez, as curvas representam a melhor solução, em função do número de geração, para diversos tamanhos da população.

A função de avaliação foi definida de forma a transformar o problema numa minimização (analogamente ao exposto nos problemas *F6*, *F7* e *F8*). Desta forma, o óptimo de *M80* é o valor zero, correspondente ao máximo do valor de benefício.

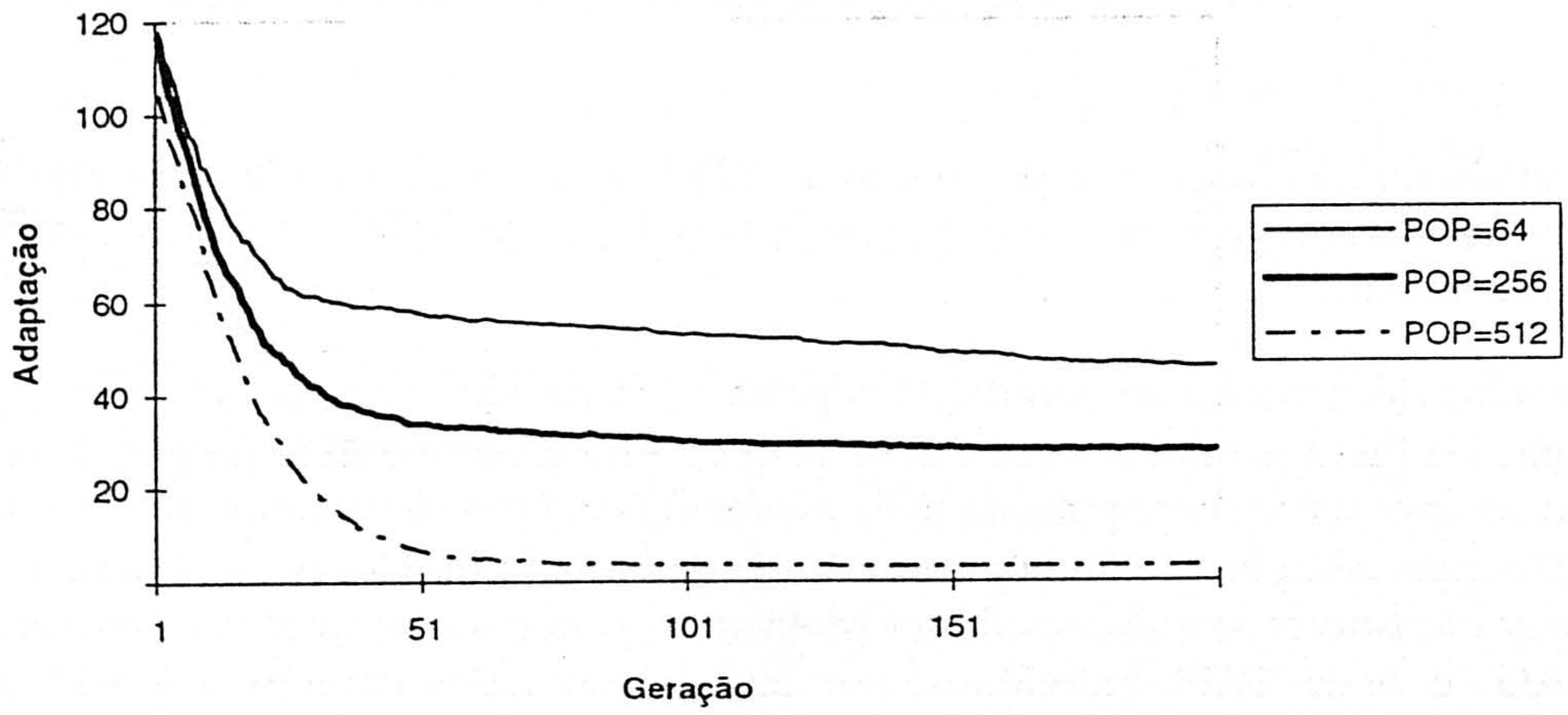


Figura 4.7: Desempenho do AG no problema *M80*

Uma vez mais, o aumento do tamanho da população melhora a evolução da curva referente à qualidade da melhor solução. Para atingir o óptimo, deve ser utilizada uma população de 1024 indivíduos. No sentido de procurar melhorias na qualidade final alcançada por outros modelos de AG, será tomada como referência a curva de 256 indivíduos. A tabela 4.2 evidencia as diferenças das soluções finais em função do tamanho da população, ao fim de 600 gerações.

	POP = 64	POP=256	POP = 512
M80	22.0	2.6	0.9

Tabela 4.2: Soluções encontradas na resolução de *M80*

4.3.2.2. Resultados do Problema do Caixeiro Viajante

Também para o problema *PCV96*, são evidenciadas na figura 4.8 as curvas da melhor solução conseguida em função do número de geração, para três diferentes tamanhos da população e para médias de 20 simulações.

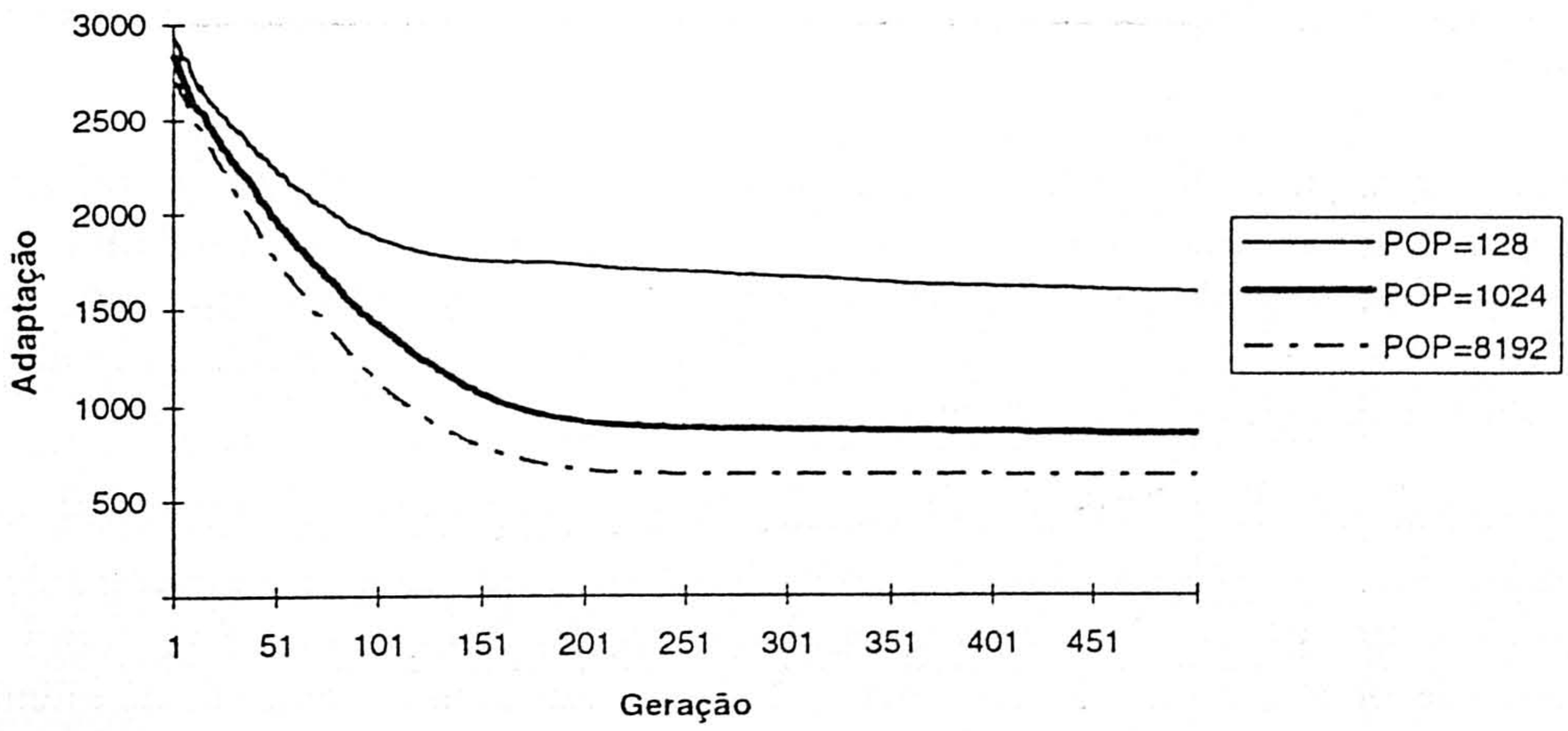


Figura 4.8: Desempenho do AG na abordagem do problema *PCV96*

A função de avaliação calcula o tamanho total do percurso. A melhor solução conseguida foi de 593 Km, mas os diferentes resultados, em função do tamanho da população, podem ser vistos na tabela 4.3.

	POP = 128	POP = 1024	POP = 8192
PCV96	1536.15	864.75	626.90

Tabela 4.3: Soluções encontradas na resolução de *PCV96*

O problema *PCV96* é bastante complexo, e mesmo com a utilização de 8192 indivíduos não é atingido o óptimo. Todas as abordagens a este problema terão sempre de se basear em populações numerosas.

4.4. APLICAÇÃO DO ALGORITMO GENÉTICO ASSÍNCRONO

Esta secção avalia o desempenho do AGA quando aplicado à base de teste anteriormente definida. Anteriores análises deste modelo, reportam que a alteração efectuada no algoritmo não afecta negativamente a recombinação de esquemas, inerente ao processo de pesquisa genético [Zeigler91], pelo menos no que respeita aos problemas típicos.

Para comparar directamente o desempenho relativo do AG e do AGA, foi simulada a execução deste modelo num ambiente de teste (através de um simulador, conforme referido no início deste capítulo). O objectivo não é minimizar o tempo total de cálculo (não há de facto paralelismo) mas sim possibilitar a análise do AGA do ponto de vista da qualidade da evolução da pesquisa genética.

Para possibilitar a comparação directa entre os dois modelos, o AG original e o AGA, é necessário criar a noção de geração nestes últimos. Uma vez que o conceito de geração no AGA é um pouco vago (ver capítulo 3, secção 3.3.2), define-se como sendo o conjunto de operações genéticas repetido tantas vezes como o número de elementos da população. Cada uma destas operações pode dar origem a um cruzamento, a uma mutação, a ambas ou a nenhuma, tal como na versão original. É assim criada a noção artificial de geração, sem interferir no algoritmo, de forma a que o número médio de operações genéticas, por geração, seja igual nos dois modelos.

Conforme já referido, nesta comparação o tempo de cálculo é medido em termos do número de gerações, ou número de processamentos genéticos. Está a ser propositadamente excluída a vantagem que o AGA proporciona ao explorar as diferenças nos tempos de avaliação de diferentes indivíduos, com o objectivo de verificar se a alteração proposta afecta ou não a eficácia do processo genético em si.

Em todos os exemplos analisados, o AGA obteve uma maior velocidade de convergência do que o AG original. No entanto, também na maioria dos casos, o AGA convergiu para uma solução ligeiramente pior. O gráfico da figura 4.9 mostra a evolução típica de ambos os modelos, neste caso para o problema *F6* (o gráfico refere-se a médias de 20 simulações).

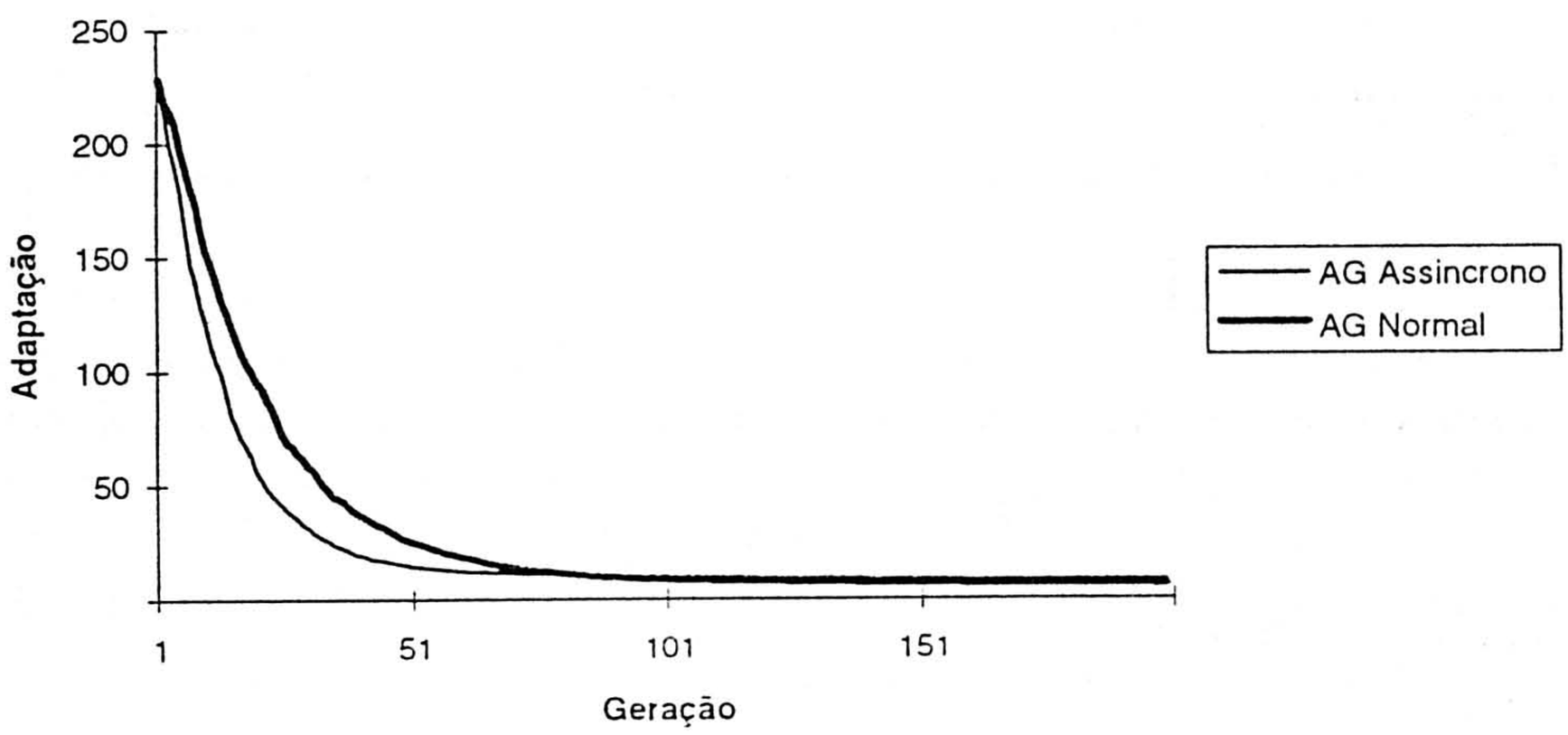


Figura 4.9: Diferenças entre o AG e o AGA para o problema *F6*

Está bem patente neste gráfico a mais rápida convergência do AGA, que até perto da geração número 70 apresenta as melhores soluções. No entanto, o cruzamento das duas linhas reflecte a convergência para uma solução ligeiramente melhor por parte do AG original. Esta situação repetiu-se praticamente em todas as experiências.

A tabela 4.4 compara ambas as abordagens em toda a base de teste, no que respeita aos valores das soluções obtidas ao fim de 600 gerações, em função de diferentes tamanhos da população.

A explicação para a melhoria no tempo de convergência por parte do AGA, tem a ver com o facto de haver uma incorporação de material genético mais cedo neste modelo do que no AG original, no qual um novo indivíduo só tem hipótese de contribuir para a evolução na geração seguinte. Existe ainda uma maior pressão na eliminação das piores

soluções, uma vez que estas estão sujeitas a serem substituídas mais cedo, assim que começa cada época (o conceito de época aqui deve ser entendido no sentido lato), sendo desta forma diminuídas as suas "fracas" contribuições.

	Tamanho POP	AG	AGA
F6	1024	6.06	8.04
F7	1024	2.99	5.43
F8	1024	0.17	0.30
M80	512	0.90	1.95
PCV96	8192	857.05	864.75

Tabela 4.4: Comparação entre o AG e o AGA na base de testes

Em contrapartida, este processo leva a uma diminuição da diversidade da população, que tem como consequência o aumento do perigo de convergência para óptimos locais [Goldberg89]. Este efeito é visível na comparação das soluções finais, obtidas pelas duas abordagens, ou seja, o AGA converge em geral para uma solução pior. No entanto, as diferenças entre os valores das soluções atingidas não são muito significativas e dependem do problema.

Os resultados apresentados demonstraram que o AGA não deteriora o processo de pesquisa genética, melhorando a velocidade de convergência à custa de uma ligeira degradação de qualidade. Esta velocidade de convergência refere-se ao número de gerações, e o facto de ser melhor do que no AG original implica que na implementação distribuída do AGA é possível diminuir o tempo de processamento pelo menos tantas vezes como o número de processadores do sistema (até certo limite, claro).

4.5. APLICAÇÃO DO ALGORITMO GENÉTICO CELULAR

Nesta secção, é avaliado o desempenho relativo do AG Celular, quando comparado com o AG original e o AGA, aplicados à conhecida base de teste. O AG Celular tem sido utilizado com sucesso em várias situações, apresentando algumas características próprias [Baluja92].

A comparação, mais uma vez, realiza-se por intermédio de um simulador apropriado ao modelo, no qual é criado um ambiente artificial para a execução do AG Celular, centrando a análise na qualidade do processo de pesquisa genética, isolando as vantagens inerentes ao sistema de computação distribuída. Para permitir uma comparação directa, foi criada a noção de geração para este modelo, de uma forma análoga à do AGA.

O gráfico da figura 4.10 mostra a evolução relativa do AG Celular, em comparação com o AG original e com o AGA, neste caso para o problema *F6* (o gráfico refere-se a médias de 20 simulações).

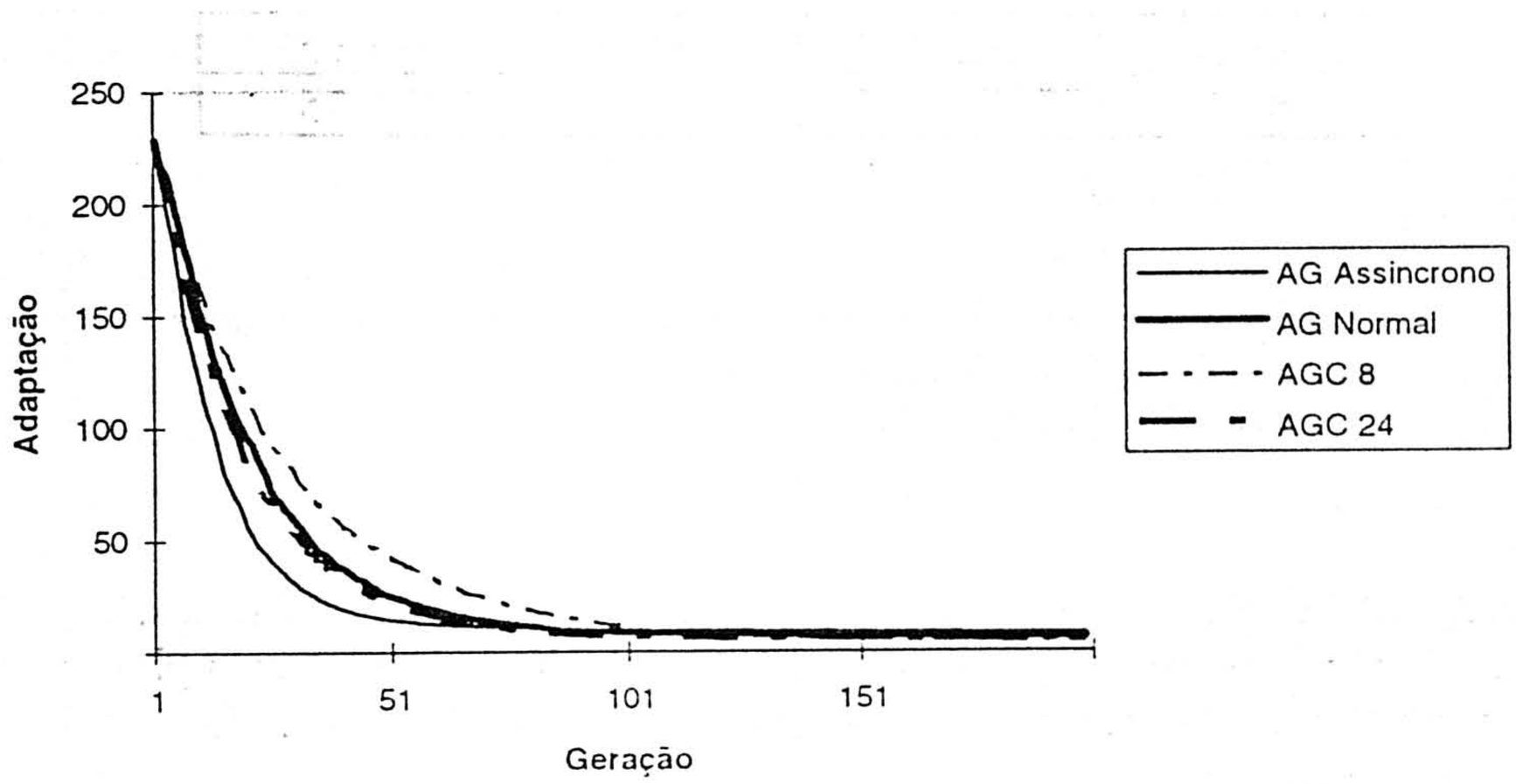


Figura 4.10: Diferenças entre o AG, o AGA e o AG Celular para o problema *F6*

Foram utilizadas duas variantes do AG Celular. A primeira, designada por *AGC8*, considera uma vizinhança normal, ou seja, cada elemento pode ser combinado com um dos 8 vizinhos directos. A segunda variante, designada por *AGC24*, considera 24 vizinhos, contando para isso com duas camadas concentricas na matriz de elementos (8 na primeira e 16 na segunda).

O *AGC8* tem pior velocidade de convergência do que o AG original. A razão desta degradação deve-se à dificuldade de fluxo da informação genética pela matriz de elementos, uma vez que todos os indivíduos estão limitados a processamento local, podendo apenas influenciar e sofrer influências dos vizinhos directos. Assim, é impossível a recombinação de informação genética de indivíduos muito distanciados, a não ser após o decorrer de algumas gerações e por influência indirecta de todos os vizinhos intermédios. A abordagem *AGC24*, por outro lado, dificulta menos a passagem de informação pela população, por ser menos local. Esse efeito é bem patente no gráfico

anterior (figura 4.10), que mostra que a velocidade de convergência desta variante é praticamente igual à do AG original, embora inferior à do AGA.

Quanto pior velocidade de convergência, maior é a diversidade da população, facilitando a geração de novas soluções tendencialmente melhores. Esse efeito é também ilustrado no gráfico anterior, no qual se verifica que a curva da variante *AGC8* cruza as restantes, atingindo uma solução melhor ao fim das 200 gerações. Pela mesma razão, a curva do AGA obtém o pior resultado. A tabela 4.5 confirma este efeito, mostrando as soluções finais obtidas nas anteriores simulações, ao fim de 600 gerações:

	Tamanho POP	AG	AGA	AGC 8	AGC 24
F6	1024	6.06	8.04	**3.52**	5.10
F7	1024	2.99	5.43	**1.42**	2.47
F8	1024	0.17	0.30	**0.07**	0.14
M80	512	0.90	1.95	**0.00**	**0.00**
PCV96	8192	857.05	864.75	837.10	**611.30**

Tabela 4.5: Comparação entre o AG, o AGA e o AG Celular, na base de testes

Nesta tabela é visível que o AG Celular atinge sempre a melhor solução, sobretudo na sua variante mais local. No entanto, conforme visto, a sua velocidade de convergência é muito mais lenta, e ao fim de um número intermédio de gerações (200 nestas experiências), a diferença na qualidade das soluções não é significativa.

Em geral, a abordagem *AGG8* convergiu mais lentamente do que a abordagem *AGC24*, mas para uma solução melhor, com excepção do problema *PCV96*. A explicação para este facto relaciona-se com o tamanho da população, que para este problema é de 8192, bastante superior à dos restantes, fazendo com que uma vizinhança de 24 indivíduos não seja excessiva, e que uma vizinhança de 8 indivíduos seja insuficiente, pelo menos para as 600 gerações.

De forma análoga, o AGA converge muito mais depressa, atingindo ao fim de muitas gerações uma solução pior do que os restantes modelos. No entanto, considerando novamente um número intermédio de gerações, a diferença de qualidade é pequena.

Com base nos resultados obtidos, e tendo em conta o compromisso entre a qualidade de solução e o esforço necessário para a obter, que depende directamente do número de gerações, o AGA oferece a melhor solução. Se existir a necessidade de obter melhores resultados, é menos penoso aumentar o número de indivíduos no AGA do que aumentar

implicitamente o número de geração ao utilizar o AG Celular ou o AG original, sobretudo porque as diferenças na qualidade das soluções não são muito grandes, mesmo ao fim de muitas gerações (em comparação com as diferenças das abordagens de diferentes tamanhos de população).

Por esta razão, aliada ao facto de o AGA se adaptar muito facilmente a ambientes de computação distribuída, este último será o núcleo dos dois restantes modelos a analisar, nomeadamente o Modelo de Ilhas e o Modelo das Populações Sobrepostas, considerados nas duas próximas secções.

4.6. APLICAÇÃO DO MODELO DE ILHAS

Esta secção analisa o desempenho relativo de uma variante do Modelo de Ilhas, na qual cada subpopulação, ou ilha, executa um AGA em vez de um AG original. A razão da substituição do AG original por este último foi apontada no final da secção anterior, e a actual variante do Modelo de Ilhas será utilizada na arquitectura da plataforma de distribuição de AG, a descrever no próximo capítulo.

Uma vez mais, a análise é centrada no processo genético em si, isolando questões relacionadas com a implementação em ambientes de computação distribuída. Todos os resultados são provenientes de um simulador de distribuição, especialmente concebido para este modelo.

A evolução do Modelo de Ilhas é directamente comparada com a evolução do AGA, que é a sua base de execução, no sentido de isolar a contribuição da arquitectura de ilhas na evolução do processo de pesquisa de soluções. Neste modelo, existe um conjunto de novos factores de parametrização, que justificam alguma atenção.

4.6.1. Análise Preliminar

Antes de efectuar uma análise mais pormenorizada do Modelo de Ilhas, que terá em conta diversos aspectos de controle, é útil enquadrar os resultados esperados, definindo alguns limites. O primeiro factor de comparação para este modelo, com um determinado número de ilhas, tem em conta os resultados obtidos por uma só ilha, ou subpopulação, evidenciando as melhorias obtidas pela comunicação entre as ilhas.

O segundo factor de comparação equaciona os resultados da execução de todas as subpopulações, mas de uma forma isolada, sem comunicação entre elas. Assim, a melhor solução esperada neste quadro, para uma topologia com k ilhas, seria a melhor das soluções obtidas em cada uma das k subpopulações ou ilhas.

Algumas experiências demonstraram que estas soluções não chegam à qualidade de uma solução obtida por uma só população, de tamanho k vezes superior, conforme se mostra na tabela 4.6, que compara a abordagem de uma população de N indivíduos com a abordagem de k populações isoladas de N/k indivíduos cada, para alguns problemas (com $k=8$). Os valores são obtidos por cálculo da média dos resultados de 20 simulações da população com tamanho N, e 20*8 simulações das populações $N/8$, com cálculo intermédio do melhor obtido de 8 em 8 simulações, nos problema *F6, F7* e *F8*.

	N	Média_{20} (N)	$\text{Média}_{20}(\text{Max}_8(\text{N/8}))$
F6	1024	8.04	36.69
F7	1024	5.43	141.17
F8	1024	0.30	0.57

Tabela 4.6: Comparação entre as abordagens de AG com *POP=N* e *k*POP=N/k*

A grande diferença na qualidade das soluções atingidas, evidenciadas na tabela 4.6, demonstram que a divisão em populações isoladas introduz uma grande degradação da eficácia do algoritmo, sendo preferível utilizar só uma população global, que englobe todos os indivíduos de todas as subpopulações, capaz de aproveitar de melhor forma o material genético, uma vez que não existe isolamento entre as diversas partes da população. Desta forma, para conseguir resultados aceitáveis com a divisão da população em subpopulações, é necessário trocar informação entre elas.

Por fim, o último dos factores de comparação considera os resultados conseguidos por um AGA na população global referida. Esta população passa a designar-se por superpopulação, e consome praticamente o mesmo volume de cálculo, pelo que é uma base justa de comparação, que evidencia as diferenças relativas somente à divisão em ilhas.

O que se espera do Modelo de Ilhas é que seja significativamente melhor do que uma só ilha, ou subpopulação, e que se aproxime de um AGA a operar sobre a respectiva superpopulação. No que se segue, é analisado o impacto dos diversos factores de controle do modelo, na evolução das soluções de todos os problemas da base de teste já conhecida. Para verificar o efeito de cada factor, é feita uma análise mais detalhada do problema *F6* com 8 subpopulações de 128 indivíduos cada, ligadas na topologia de Anel Duplo.

4.6.2. Análise de Desempenho

O desempenho do Modelo de Ilhas depende fortemente de um conjunto de factores de parametrização, que serão analisados isoladamente e em conjunto. Os principais factores são o intervalo entre migrações, o número de emigrantes e a topologia de ligações entre ilhas.

4.6.2.1. Intervalo entre Migrações

O intervalo entre migrações, *T*, influi significativamente na evolução do algoritmo. O gráfico da figura 4.11 compara o seu efeito para um modelo com 8 ilhas de 128 indivíduos cada, ligadas na topologia de Anel Duplo, para um número de emigrantes, *N*, fixo em 10% da subpopulação de cada ilha. Para referência, é desenhada a curva com a abordagem de uma só subpopulação (de 128 indivíduos) e da respectiva superpopulação de 1024 indivíduos (ambas actuadas isoladamente pelo AGA).

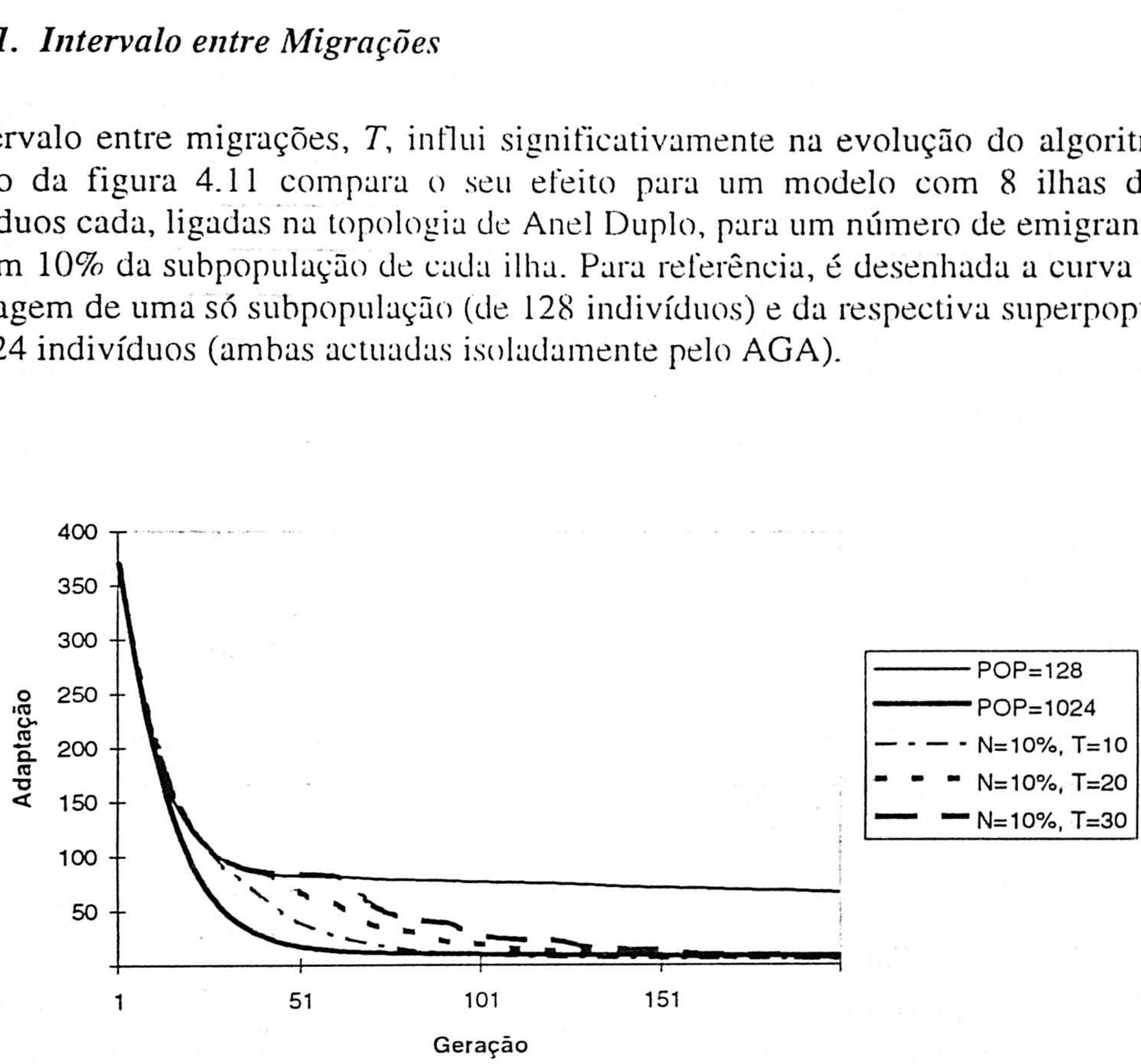


Figura 4.11: Efeito do intervalo entre migrações

A primeira conclusão a tirar desta comparação é que qualquer das variantes do período de migração, seja ele 10, 20 ou 30 gerações, é significativamente melhor do que a abordagem da subpopulação de 128 indivíduos isolada, no que respeita à qualidade final da solução. No que respeita à comparação com uma superpopulação de 1024 indivíduos, verifica-se que embora a velocidade de convergência seja mais lenta, a qualidade final é um pouco melhor.

A comparação entre as três curvas, para N=10%, revela que quanto menor o intervalo de tempo entre migrações, mais rápida é a convergência para a solução final, uma vez que as soluções locais encontradas em cada uma das populações fluem mais rapidamente para as restantes ilhas. No entanto, foi verificado que à medida que este intervalo aumenta, a solução final melhora residualmente. Este facto traduz o efeito de uma leve convergência prematura global, provocada pela migração em excesso.

É a diversidade da população global proporcionada pelo semi-isolamento das ilhas que justifica a melhoria global de qualidade, em comparação com uma só superpopulação. Cada uma das ilhas desenvolve a sua população, especializando-a numa determinada parte do problema de optimização. Os seus elementos têm características próprias, e a convergência local especializa-os numa determinada área do espaço de estados das soluções.

Em qualquer população evoluída por um AG, os bons indivíduos tendem a influenciar todos os restantes, diminuindo a diversidade da população. No Modelo de Ilhas, passa-se também este efeito, mas somente dentro de cada ilha. A diversidade global é superior porque a influência dos bons indivíduos sobre indivíduos de outras ilhas está dependente da migração, que a diminui substancialmente. O aumento do intervalo de migração leva a uma diminuição desta influência, implicando um aumento da diversidade. Quanto maior a diversidade mais facilidades existem em melhorar a qualidade das soluções (no gráfico 4.11, esta diferença é demasiado pequena para ser visualizada) e maior é o respectivo tempo de convergência, explicando as diferenças entre as 3 curvas do gráfico anterior (T=10,T=20 e T=30).

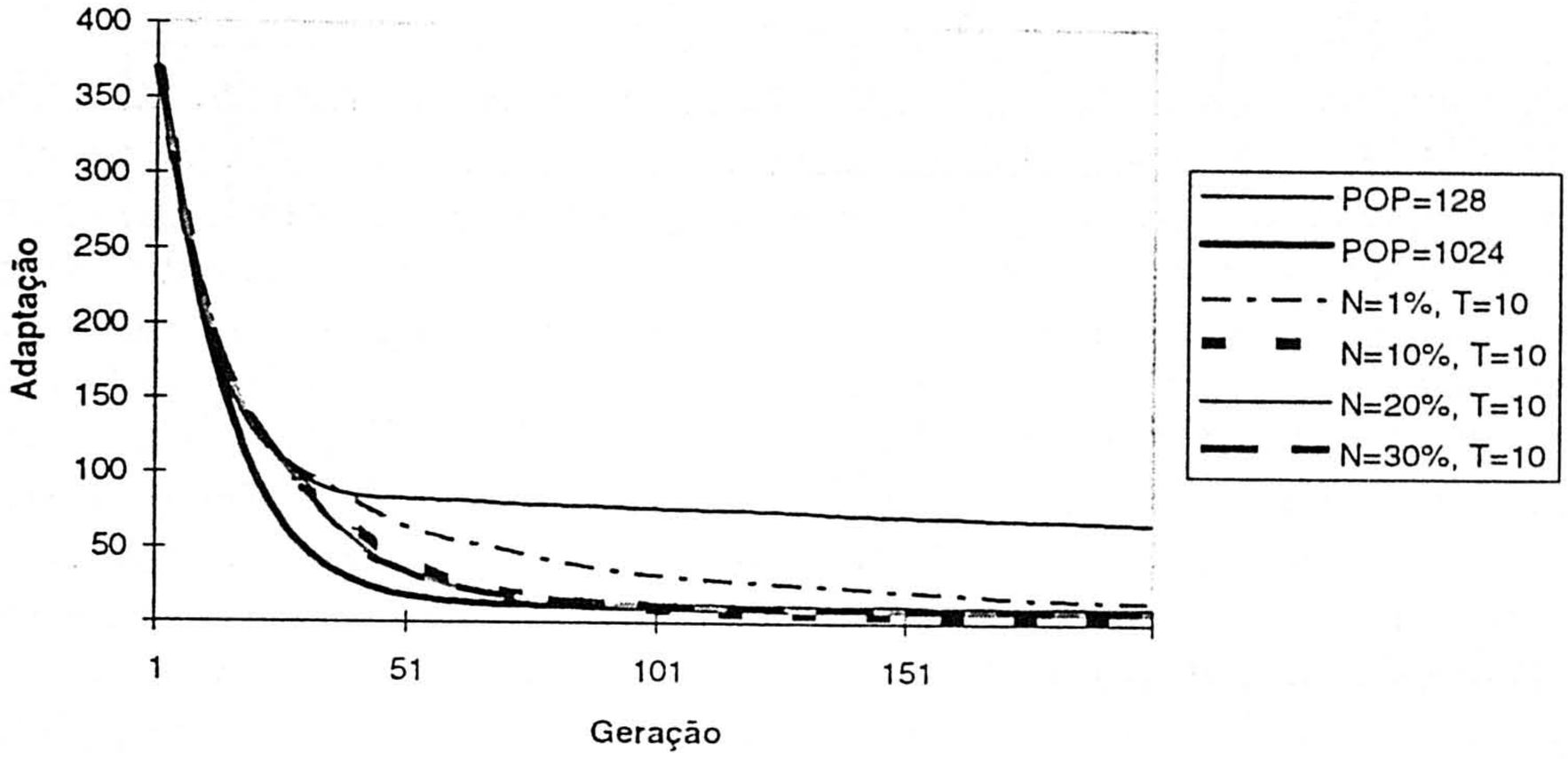
Embora a existência de ilhas provoque algum isolamento local, a partilha de informação levada a cabo pelos elementos que emigram tem o efeito de combinar soluções de conteúdo muito diferente, originando soluções ainda melhores, sem provocar uma convergência prematura.

4.6.2.2. Número de Emigrantes

O número de emigrantes, N, tem um efeito mais moderado na evolução do algoritmo deste modelo. A sua influência incide essencialmente numa ligeira variação da velocidade de convergência, sendo as soluções finais muito semelhantes para as 3 variantes analisadas. As curvas correspondentes à migração de 10%, 20% e 30% da população, para o intervalo entre migrações fixo em 10 gerações, são ilustradas no gráfico da figura 4.12.

Mais uma vez, qualquer uma das configurações obteve resultados significativamente melhores que as abordagens de uma só subpopulação, e mesmo do que a respectiva superpopulação, embora com uma velocidade de convergência mais diminuída. A variação do numero de emigrantes tem menor influência do que a variação do intervalo de migração, mas uma análise mais pormenorizada mostra que a convergência acelera com o aumento deste parâmetro. A razão para este aumento é a mesma: quantos mais

indivíduos emigram, mais decai a diversidade global e mais rápida é a convergência, mas para uma solução pior. As diferenças entre as soluções finais são mais uma vez residuais, havendo no entanto uma pequena melhoria no caso de N=10%, valor para o qual a convergência é mais lenta.



Fígura 4.12: Efeito do número de emigrantes

A utilização de um número demasiadamente pequeno de emigrantes também tem os seus efeitos negativos, quer pela diminuição exagerada da velocidade de convergência (aumentando o tempo de execução para atingir soluções úteis), quer pela insuficiência de material genético para influenciar populações remotas. Na utilização de N=1% para *F6*, com 8 subpopulações, só emigra 1 indivíduo, que sozinho tem forte probabilidade de ser eliminado pelo operador de selecção, mesmo com um bom valor relativo. Este efeito é visível no gráfico da figura 4.12, na respectiva curva.

A tabela 4.7 mostra os resultados obtidos no problema *F6*, na topologia em Anel Duplo, fazendo variar o número de emigrantes de 1% a 30% e o intervalo de migrações de 10 a 30 gerações, para realçar o efeito combinado dos dois parâmetros.

Os resultados obtidos para este problema em particular, encontraram eco em toda a base de teste. Em todos os problemas analisados, o Modelo de Ilhas atingiu melhores soluções do que a sua correspondente superpopulação. Em geral, o intervalo entre migrações que mais beneficiou a solução final foi T=30, enquanto que o número de emigrantes deve ser mantido baixo, na ordem dos 10% da população.

Para obter uma maior velocidade de convergência, devem-se utilizar intervalos entre migrações mais curtos, na ordem das 10 gerações. No entanto, estes valores também levam a uma ligeira degradação da qualidade final. A escolha dos valores apropriados deve ter em conta as características de cada situação em particular.

POP=128	55.54
POP=1024	8.04

8* POP=128	**T=10**	**T=20**	**T=30**
N=1%	6.84	8.09	9.79
N=10%	4.45	5.13	**3.66**
N=20%	7.15	6.51	5.02
N=30%	7.46	6.11	6.82

Tabela 4.7: Efeito do *N* e *T* com a topologia em Anel Duplo, para *F6*

4.6.2.3. Tamanho Global da População

Na análise do problema *PCV96*, foram realizados dois conjuntos de simulações, uma com 8 populações de 128, comparada com a superpopulação de 1024, e outra com 8 populações de 1024, comparada com a superpopulação de 8192.

No primeiro conjunto, os resultados divergiram do habitual. As melhores soluções foram as relacionadas com curtos intervalos de migração, T=10, ao contrário do valor de T=30 válido para todos os restantes problemas da base de teste. Neste caso, embora os resultados no modelo das 8 populações tenha sido muito superior a uma só, não atingiram os resultados da superpopulação.

No segundo conjunto de simulações, os resultados obtidos foram concordantes com a restante base de teste. A razão que sustenta a divergência anterior é a insuficiência do total de 1024 indivíduos para a resolução do problema. Devido à complexidade e natureza deste, é necessária uma maior população, por exemplo de 8192 indivíduos, para que o algoritmo funcione normalmente. Caso contrário, ocorrem convergências prematuras demasiado fortes em cada uma das subpopulações, que comprometem o resultado final, mesmo com a separação em ilhas. Este problema tem sido abordado em recentes trabalhos na área de AG distribuídos [Goldberg95].

Neste contexto, uma migração mais frequente melhora o desempenho global, já que atrasa as diversas convergências locais por inclusão de diferente material genético proveniente de outras subpopulações. Mesmo assim, é visível uma degradação de desempenho no modelo.

Em resumo, a chave para uma boa parametrização passa por ter uma população global suficientemente grande, grandes intervalos de migração (30) gerações) para obter

soluções melhores, pequenos intervalos para obter soluções rapidamente (10 gerações), e pequenos valores no número de emigrantes (10% da população).

4.6.2.4. Topologia

O efeito da topologia é visível, por comparação do melhor resultado conseguido nas simulações de 12 variantes do período de migração e do número de emigrantes. Na migração de uma determinada ilha, os indivíduos são sequencialmente alocados às diferentes ilhas com as quais estabelecem ligação. No caso da topologia em Anel Duplo, na qual todas as ilhas estão ligadas às suas 2 vizinhas, o primeiro emigrante de qualquer das ilhas vai para a ilha com ordem superior em 1, o segundo para a ilha com ordem inferior em 1, o terceiro de novo para a primeira, e assim sucessivamente em ciclo. Naquele exemplo, cada ilha recebe metade de *N*% de 128 indivíduos de cada uma das 2 ilhas vizinhas, de *T* em *T* gerações. A comparação das diferentes topologias para *F6* é ilustrada na tabela 4.8, evidenciando a configuração que optimiza cada uma delas (*V* é o valor do óptimo atingido pelo algoritmo). No anexo A, encontram-se descritos os resultados análogos para todos os problemas da base de teste.

Topologia	8 Ilhas	*16 Ilhas*
Anel	N=1%, T=20, V=4.09	N=1%, T=10, V=**2.57**
Anel Bidireccional	N=10%, T=30, V=3.66	N=10%, T=30, V=3.77
Anel Duplo	N=10%, T=30, V=4.97	N=10%, T=20, V=4.00
Anel Duplo Bidireccional	N=10%, T=30, V=4.06	N=10%, T=10, V=3.69
Completa	N=10%, T=20, V=4.49	N=20%, T=30, V=4.10
Estrela	N=30%, T=20, V=4.58	N=30%, T=20, V=6.42
Aleatória 3	N=10%, T=30, V=4.40	N=10%, T=20, V=3.03
Hiper-Cubo	N=10%, T=30, V=3.88	N=20%, T=30, V=3.74
Torus	N=10%, T=20, V=3.61	N=10%, T=30, V=3.84
Árvore 2 Bidireccional	N=10%, T=30, V=**3.45**	N=10%, T=10, V=3.95

Tabela 4.8 Efeito da topologia em *F6*

Os valores obtidos confirmam, em geral, a regra já concluída para os parâmetros *N* e *T* (ver anexo A). No que respeita às topologias para 8 populações, verifica-se que o Anel Bidireccional, a Árvore, o Hiper-Cubo e o Torus foram as que melhor se salientaram, com valores de *N* e *T* de 10% e 30 gerações, respectivamente. No outro extremo encontra-se a Estrela, a Aleatória e o Anel Duplo simples.

Na anterior análise de topologias, a comparação é feita segundo a melhor das 12 configurações de *N* e *T* analisadas. A outra forma de comparação desenvolvida, que avaliou as topologias segundo os seus desempenhos médios em todas as 12 configurações, não divergiu muito daquela.

A influência da topologia não se manifestou tão constante no conjunto de problemas como os anteriores parâmetros. Não houve nenhuma delas que se salientasse como inequivocamente a melhor em todos os problemas. A comparação de quais as melhores topologias, para 8 e 16 ilhas, traduz assim alguma variância, conforme é visível na tabela 4.9.

Problema	8 Ilhas	16 Ilhas
F6	Árvore, Anel Bidir., Hiper-Cubo	**Anel**, Aleatória
F7	Árvore, Anel	**Anel Bidir.**, Anel Duplo Bidir.
F8	**Torus**, Anel Bidir.	Anel Bidir.
M80	Anel Duplo, Aleatória	**Anel Bidir.**, Torus
PCV96	**Torus**, Anel Duplo	Completa, Torus

Tabela 4.9 Resumo das melhores topologias na base de testes, para 8 e 16 populações

De qualquer forma, os anéis bidireccionais e duplos revelam ter grande predominância nas boas soluções, assim como o Torus. Em contrapartida, os anéis simples são das piores soluções.

A tabela 4.9 salienta também qual dos modelos conseguiu a melhor das soluções, no que respeita ao número de ilhas. Constatou-se que nos problemas de menor complexidade, *F6*, *F7* e *M80*, a qualidade melhora com o número de ilhas (pelo menos com a passagem de 8 para 16), enquanto que nos problemas de maior complexidade, *F8* e *PCV96*, passa-se o contrário. Isto significa que problemas de baixa complexidade permitem a evolução do algoritmo em pequenas populações, aproveitando o benefício directo da existência de maior número de ilhas. Por outro lado, problemas de maior complexidade não facilitam a evolução em pequenas populações, necessitando de uma maior "massa crítica" de material genético para progredir.

4.6.2.5. Outros Parâmetros de Controle

Outros dois parâmetros de controle analisados no decurso das experiências são a estratégia de selecção dos emigrantes (ou política de emigração) e a estratégia de eliminação de indivíduos locais (ou política de imigração).

A estratégia de selecção dos emigrantes não teve grande influência nos resultados. Os resultados obtidos seleccionando os indivíduos por selecção viciada não foram muito melhores do que os obtidos por selecção aleatória, revelando que o fundamental no algoritmo não é o fluxo das boas soluções locais, mas sim o fluxo de quaisquer soluções locais, portadoras de características da ilha a que pertencem.

No que respeita à estratégia de eliminação de indivíduos locais, para dar lugar aos novos imigrantes, a diferença entre a escolha dos piores por selecção inversa viciada e a escolha aleatória, também não foi significativa.

4.7. APLICAÇÃO DO MODELO DAS POPULAÇÕES SOBREPOSTAS

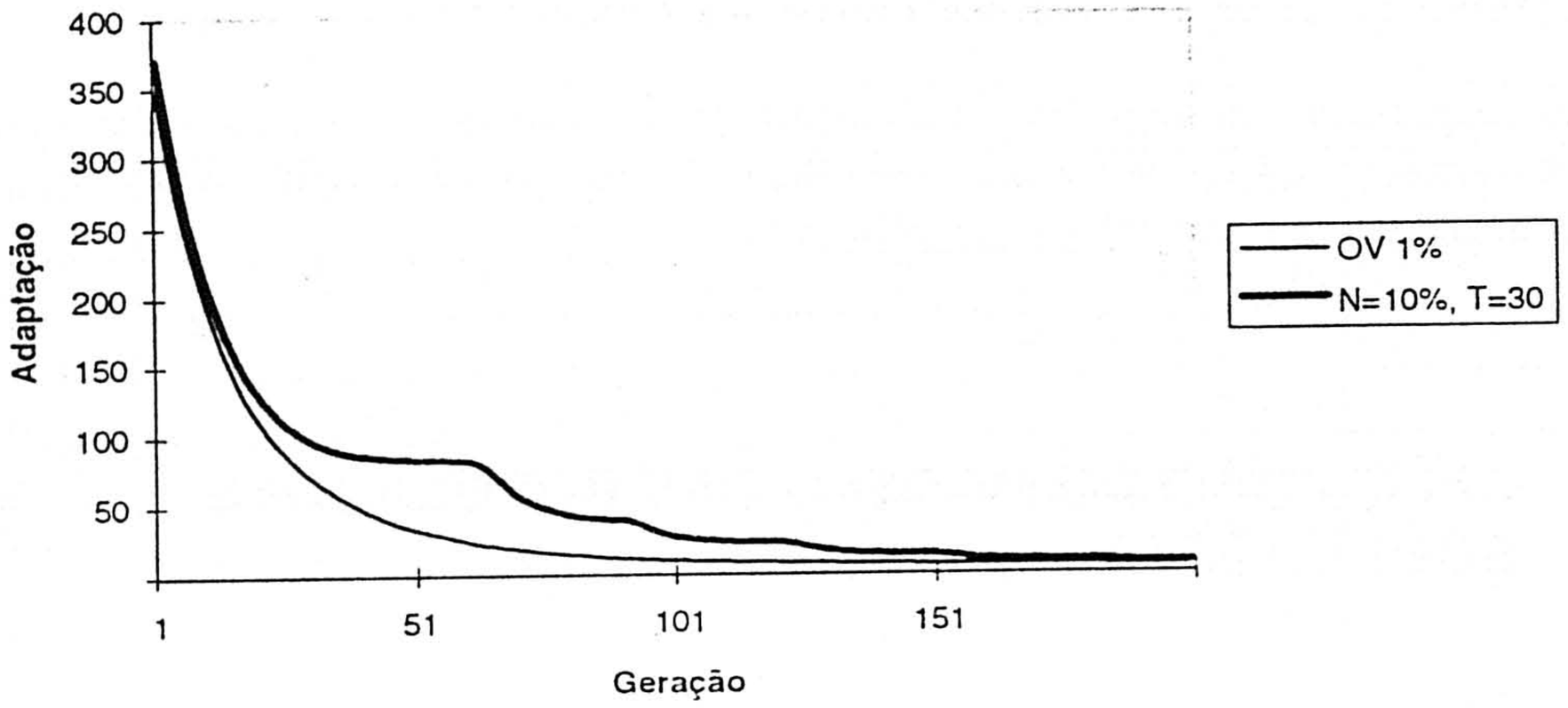
O último dos modelos analisados neste capítulo é uma variante do Modelo das Populações Sobrepostas, que de modo análogo ao Modelo de Ilhas descrito na secção anterior, tem por base o AGA em cada uma das subpopulações, em vez do AG original.

Nesta secção são focados os aspectos relacionados com o modelo, tendo por base de comparação os conceitos de subpopulação e superpopulação, descritos na secção anterior. A comparação directa com o Modelo de Ilhas é também considerada, uma vez que ambos apresentam características globais semelhantes (no que respeita à localidade do processamento e à partilha de informação entre subpopulações semi-autónomas).

A parametrização neste modelo é menos complexa do que no Modelo de Ilhas, no qual existem muitos mais graus de liberdade. Aqui, os principais factores são o número e tamanho das subpopulações e as respectivas percentagens de sobreposição.

As experiências analisadas com o simulador de distribuição para este modelo consideram a existência de 8 subpopulações de tamanho equivalente, que contêm entre 64 e 1024 indivíduos cada (consoante o problema). As percentagens de sobreposição consideradas são 1% e 10%, com excepção das situações em que a sobreposição, devido ao pequeno tamanho das subpopulações, se traduz em quantidades inferiores a um indivíduo (nestes casos, a sobreposição é de um indivíduo, aumentando residualmente a referida percentagem).

O gráfico da figura 4.13 demonstra que a qualidade da solução final atingida por ambos os métodos é semelhante, assim como a velocidade de convergência. A principal diferença é que a evolução no Modelo de Populações Sobrepostas é mais contínua, convergindo mais rapidamente do que o Modelo de Ilhas, o qual evolui aos surtos devido à migração periódica. Este último obtém soluções um pouco melhores, reforçando a regra que indica que quanto mais rápida é a convergência, pior é a solução final.



Fígura 4.13: Comparação do Modelo de Ilhas com o Modelo das Populações Sobrepostas

A tabela 4.10 resume os resultados obtidos na base de teste, para este modelo com 8 ilhas, tendo sempre como base de comparação o AGA na subpopulação, o AGA na superpopulação, e o Modelo de Ilhas na configuração de 8 ilhas em Anel Bidireccional, com N=10% e T=30 gerações (cada problema tem o seu tamanho de população próprio, designado na tabela por *POP*).

Também neste modelo os resultados obtidos foram claramente superiores ao do AGA aplicado a uma das subpopulações. Mesmo comparado com a superpopulação equivalente, as populações sobrepostas provaram a sua superioridade, no que respeita à qualidade final das soluções obtidas.

A percentagem de sobreposição que, no geral, melhores resultados obteve neste modelo foi a de 1%, conforme previsto por comparação com o modelo de ilhas. A comparação directa com aquele modelo, na topologia em Anel Bidireccional, revela que a sobreposição das populações consegue soluções finais semelhantes, evidenciando a semelhança entre os dois modelos.

Problema	POP	AGA na subpop. (POP/8)	AG na superpop. (POP)	Ilhas com Anel Bidir. N=10%, T=30	Populações Sobrepostas com s=1%	Populações Sobrepostas com s=10%
F6	1024	55.54	8.04	**3.66**	5.70	6.16
F7	1024	451.46	5.43	3.29	**2.99**	3.92
F8	1024	1.21	0.30	**0.09**	0.15	0.27
M80	512	26.10	1.95	1.50	1.08	**1.05**
PCV96	8192	1536.15	864.65	611.77	**611.05**	625.65

Tabela 4.10 Comparação do Modelo das Populações Sobrepostas com o AG Normal

A utilização de 16 subpopulações sobrepostas, com metade do tamanho das anteriores 8 subpopulações, não alterou muito os resultados já apresentados, sendo a diferença neste caso menos significativa do que no Modelo de Ilhas, talvez por haver aqui maior coesão global entre as subpopulações, motivada pela contínua troca de informação que ocorre dentro de cada geração e não apenas em intervalos fixos como naquele modelo.

4.8. RESUMO

Os modelos de AG analisados ao longo deste capítulo podem-se dividir em duas classes distintas, designadamente os modelos que operam sobre uma só população, como o AGA e o AG Celular, e os modelos que operam sobre várias populações independentes, ou semi-independentes, como o Modelo de Ilhas e o Modelos das Populações Sobrepostas.

Na primeira classe, o AGA caracteriza-se por ter uma velocidade de convergência mais rápida do que o AG original, atingindo as soluções finais em menor número de gerações, embora estas soluções tenham, em média, uma qualidade ligeiramente inferior.

De forma inversa, o AG Celular converge mais vagarosamente, mas para uma solução um pouco melhor, em comparação com AG original. O comportamento deste algoritmo pode ser manipulado através da parametrização do grau de vizinhança visto por cada elemento da matriz da população, sendo possível aumentar a velocidade de convergência à custa da diminuição da qualidade final.

Na segunda classe de modelos, ambos utilizam um conjunto de subpopulações operadas por distintos AG, com possibilidade de comunicação entre si. O algoritmo escolhido

para operar em cada uma das subpopulações foi o AGA, por reunir as características mais versáteis, constituindo um bom compromisso entre a qualidade das soluções e o esforço necessário para as alcançar.

Desta forma, estes dois modelos foram comparados com o seu algoritmo base, o AGA, quando aplicado à superpopulação equivalente, ou seja, a população que engloba tantos elementos como a soma de todas as subpopulações.

No caso do Modelo de Ilhas, os resultados demonstram que com a parametrização adequada, é possível tirar partido da distribuição em ilhas, conseguindo melhores resultados à custa de uma aceitável degradação na velocidade de convergência (medida em número de gerações). No Modelo das Populações Sobrepostas, as conclusões são idênticas.

Relembra-se que toda a análise desenvolvida neste capítulo teve em conta apenas a qualidade da pesquisa genética. A implementação de qualquer um dos modelos abordados em sistemas de computação distribuída leva a que o tempo de processamento deste seja dividido por um factor, que depende das características do modelo e do número de processadores do sistema.

Verificar que um determinado modelo não prejudica o processo de pesquisa genética, é o mesmo que concluir que o seu tempo de processamento é até certo ponto divisível pelo número de processadores do sistema, tendo em conta a granularidade do modelo. Por exemplo, o Modelo de Ilhas com 8 ilhas correctamente parametrizado e implementado num sistema de computação com 8 processadores pode ver o seu tempo de processamento dividido pelo menos por 8. Já o tempo de processamento do AGA ou o do AG Celular pode, em condições propícias, ser dividido por um número muito maior.

Por fim, refira-se que a conjugação de diferentes modelos de AG distribuídos, em especial o Modelo de Ilhas com base no AGA, pode ser benéfica, quer por explorar melhor os ambientes de computação distribuída (conforme se verá no capítulo 5), quer por conferir boas características à pesquisa genética, conforme se verificou.

AGENDA: ALGORITMOS GENÉTICOS DISTRIBUÍDOS E ASSÍNCRONOS

5.1. INTRODUÇÃO

A plataforma de distribuição de AG, desenvolvida neste trabalho, foi baptizada de Algoritmos GENéticos Distribuídos e Assíncronos (AGENDA). A AGENDA serve de base à execução dos AG distribuídos, aplicados na resolução de qualquer problema de optimização, de modo a aproveitar da melhor forma possível os recursos disponíveis num dado sistema computacional distribuído. Em contraste com os simuladores desenvolvidos para o estudo do capítulo 4, a AGENDA pretende viabilizar uma forma eficaz de distribuição real do AG.

A arquitectura da plataforma AGENDA foi concebida por forma a permitir a cooperação de diferentes sistemas computacionais entre si. Os sistemas, ligados por uma infraestrutura de comunicação, interagem com base num modelo apropriado, fazendo evoluir, em conjunto, uma população genética global. Esta população está distribuída em diversas subpopulações, espalhadas pelos diversos nós independentes do sistema, cada uma operada por um AG independente. A ligação entre as subpopulações distribuídas tem por base os modelos e princípios apresentados nos capítulos anteriores, com o objectivo de rentabilizar o melhor possível o processo global de optimização.

A AGENDA adapta-se preferencialmente a sistemas computacionais do tipo MIMD, em especial a redes de *Workstations*, explorando o elevado grau de autonomia dos seus elementos ou nós. Em cada nó, ou *Workstation*, é executado um ou mais AG, capaz de cooperar com os restantes AG remotos ou locais, por intermédio de um mecanismo de troca de mensagens, sustentado pela AGENDA. Esta, permite ainda que cada um dos

AG independentes possa tirar partido do paralelismo interno característico dos nós multiprocessador.

No que respeita a modelos de AG distribuídos (apresentados no capítulo 3), a AGENDA caracteriza-se por uma combinação do Modelo de Ilhas com o AG Assíncrono. A cada ilha está associado um AG Assíncrono independente, cuja execução pode ser internamente paralelizada (se o respectivo nó tiver mais de um processador). As diversas ilhas incorporam e disponibilizam informação genética para as restantes, através da migração de indivíduos.

Na AGENDA, a evolução quase independente das subpopulações das ilhas, confere ao sistema um elevado grau de robustez e versatilidade. O colapso ou atraso de uma das ilhas não compromete drasticamente a evolução global, sendo ainda possível substituí-la por uma nova ilha. Através da AGENDA, é também possível a cooperação de sistemas computacionais de características muito diferentes, no que respeita à arquitectura base, ao sistema operativo e à capacidade de computação.

O presente capítulo tem dois objectivos principais. O primeiro objectivo, consiste na descrição da arquitectura global da AGENDA e da estrutura interna dos seus módulos, dando ênfase ao modo de cooperação destes, no contexto do sistema de computação distribuída.

O segundo objectivo, consiste na avaliação da AGENDA. Com base num problema de optimização real, significativamente complexo, são comparadas as duas abordagens de aplicação do AG, nomeadamente o AG distribuído com a AGENDA e o AG original, não distribuído. A parametrização da AGENDA é um factor fundamental para o seu bom desempenho, pelo que é cuidadosamente preparada, tendo como base todos os resultados obtidos nas simulações realizadas no capítulo anterior.

5.2. ARQUITECTURA DA AGENDA

A AGENDA funciona como uma plataforma de suporte à distribuição dos AG. A sua arquitectura define o modo como um conjunto de processos, que implementam a optimização por AG distribuídos, devem comunicar entre si. Cada um dos processos participantes implementa um conjunto de rotinas desenvolvidas para este efeito (no âmbito da AGENDA), com as quais consegue cooperar com os seus semelhantes.

A comunicação entre processos tem por base um paradigma de troca de mensagens. Estas mensagens, com semântica própria, obedecem a um protocolo desenvolvido na AGENDA, o qual é implementado em todos os processos intervenientes. No resto desta secção, são apresentados estes processos e a relação entre eles, mostrando como conseguem cooperar na execução dos AG distribuídos. Seguidamente, é descrito o referido protocolo, que sustenta a comunicação e o funcionamento da AGENDA.

5.2.1. Processos Intervenientes

Os processos intervenientes na AGENDA funcionam em conjunto na execução distribuída de um AG. Para o conseguir, estes processos devem assegurar o desenvolvimento local de AG parciais, que operam sobre uma população interna ao processo (contida no seu espaço de endereçamento), assim como a cooperação com processos remotos. Estes, podem-se encontrar no mesmo nó ou num nó remoto do sistema computacional.

Desta forma, os referidos processos são responsáveis por duas classes distintas de actividades, ou seja, a execução local do AG e a gestão das actividades auxiliares à optimização propriamente dita. No que respeita à primeira, o AG local de cada processo (desenvolvido especificamente para o efeito, no âmbito da AGENDA) deve ser capaz de possibilitar a migração de indivíduos para outros AG remotos, executados em outros processos. Para que esta migração seja possível, é necessário que tenha sido estabelecida uma ligação entre os processos em questão.

Estas ligações entre processos têm de ser asseguradas de alguma forma. Assim, as actividades de gestão, indispensáveis ao funcionamento da AGENDA, são asseguradas por processos capazes de controlar a interacção entre os restantes, possibilitando que no início da actividade, qualquer processo responsável por um AG possa estabelecer ligações com pares que estejam no momento em execução em qualquer parte do sistema computacional (possivelmente em outros nós).

Para estabelecer estas ligações, é necessário que exista informação disponível acerca dos processos de optimização: onde se encontram (nó e endereço lógico) e qual o problema de optimização sobre o qual estão a operar. A gestão desta informação é indispensável à cooperação global entre processos.

Para clarificar o funcionamento da AGENDA, é feita aqui a distinção conceptual dos processos descritos. Aqueles unicamente relacionados com a execução do AG no âmbito da AGENDA, passam a designar-se por Processos de Algoritmos Genéticos (PAG), e os processos responsáveis pelas tarefas de gestão destes, passam a designar-se por Processos de Gestão de Algoritmos Genéticos (PGAG).

Nos PAG, é executada uma versão distribuída do AG desenvolvida para a AGENDA, capaz de cooperar com outros elementos semelhantes (através da migração) e capaz de tirar partido do paralelismo do próprio sistema onde é executado, aproveitando a eventual existência de mais de um elemento de processamento.

Nos PGAG, é guardada informação do estado dos PAG em execução, útil no estabelecimento de ligações entre estes e no controle da topologia das suas ligações. Para assegurar todo o funcionamento da AGENDA, basta que exista um só PGAG activo no sistema (entende-se por sistema o conjunto de nós ou máquinas onde é possível a execução da AGENDA).

Quando um PAG inicia a sua actividade, a sua primeira tarefa é registar-se no PGAG do sistema. Seguidamente, interroga este último, na tentativa de obter uma lista de outros PAG em execução, disponíveis para o estabelecimento de ligações. O número de ligações a estabelecer, assim como outros factores operacionais, são determinados por parâmetros de controle (ver anexo B).

Terminada esta primeira fase, o PAG inicia o AG local sobre a sua população, enviando periodicamente um conjunto de emigrantes para outros processos. Simultaneamente, verifica também periodicamente se existem alguns imigrantes a serem incorporados na sua população local, provenientes de outros PAG. No final da execução, o PAG deve anular o seu registo no PGAG, feito no início da execução. Se, por qualquer razão, durante a execução do AG local alguma das ligações se perder, o PAG requisita novas ligações ao PGAG, passando assim a cooperar com alguns novos pares.

O PGAG, por sua vez, recebe registos e respectivas anulações de diversos PAG, mantendo uma lista actualizada dos processo activos. Deste modo, aquele está apto a responder a pedidos de novas ligações, por parte de qualquer PAG que queira cooperar com outros.

Por fim, existe um terceiro tipo de processo, o Servidor de Nomes de Algoritmos Genéticos (SNAG), que consiste num servidor de nomes específico da AGENDA. Este processo mantém uma lista actualizada de todos os nós participantes no processo de optimização da AGENDA. A arquitectura global da AGENDA é esquematizada na figura 5.1.

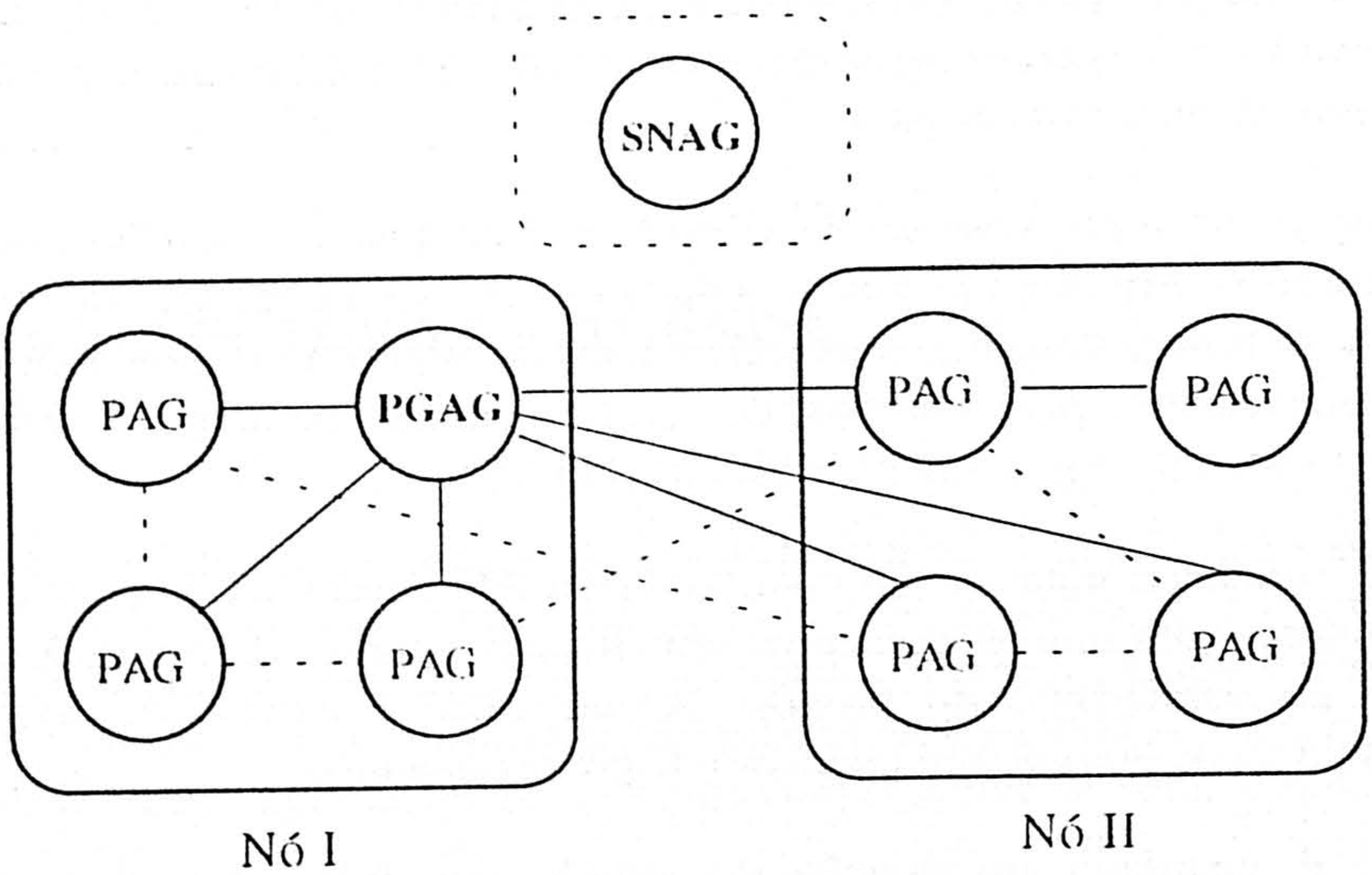


Figura 5.1: Arquitectura global da AGENDA

Com este serviço disponível, quando um PAG inicia a sua actividade, pode descobrir quais são os nós que integram o sistema computacional que serve de base à AGENDA.

Num destes nós, encontra-se o PGAG, com o qual deve ser estabelecida a comunicação inicial. A localização do SNAG tem de ser conhecida à partida, por todos os PAG.

O SNAG pode estar em qualquer nó, desde que acessível por todos os PAG (na figura 5.1 não foram representadas as ligações entre estes dois tipos de processos). Em cada sistema, deve existir um só PGAG, que controla a execução global. Na figura apresentada, as ligações entre os PAG e o PGAG estão evidenciadas, uma vez que estas ligações são obrigatórias. As restantes, representam possíveis fluxos de indivíduos em migração, definidas pelo PGAG, que indica a cada PAG quais os pares a que se deve ligar.

5.2.2. Protocolo de Comunicação

A comunicação entre os processos apresentados obedece a um protocolo, especialmente concebido na AGENDA para esse efeito. O protocolo engloba um conjunto de tipos de mensagens, que permitem a troca de informação entre os processos intervenientes, e que se encontram resumidos na tabela 5.1.

Mensagem	Origem	Destino	Descrição
gethosts	PAG	SNAG	Pedido da lista de nós do sistema
register	PAG	PGAG	Registo do PAG no PGAG
unregister	PAG	PGAG	Cancelamento do registo
getlinks	PAG	PGAG	Pedido de ligações a PAG
takeind	PAG	PAG	Envio de um indivíduo para outro PAG
iamalive	PAG	PGAG	Notificação ao PGAG que o PAG está activo
isalive	PAG	PGAG	Verificação se um PAG está activo

Tabela 5.1: Tipos de mensagens trocadas entre os processos da AGENDA

Cada tipo de mensagem contém os seus próprios argumentos, podendo ter associada uma resposta. Estas mensagens são estabelecidas entre dois processos, pertencentes aos tipos já definidos (PAG, PGAG ou SNAG), assegurando o controle e a evolução de todo o sistema. As mensagens entre os PAG e os restantes são usadas para tarefas de gestão, enquanto que mensagens entre dois PAG representam a migração da informação genética.

Para entender como se processa o estabelecimento de mensagens entre dois processos, é necessário clarificar os conceitos de endereço e porta lógica. Um processo está acessível

para comunicação com um outro, se for conhecido o endereço lógico do nó a que pertence e a porta lógica onde espera o estabelecimento da comunicação. O primeiro, identifica a máquina ou nó do sistema de computação distribuído, enquanto que o segundo estabelece o ponto de acesso do processo dentro desse mesmo nó (endereço relativo ao nó).

No início da actividade, o PAG tem de encontrar o PGAG do sistema onde se insere. Como este processo é único no sistema, podendo estar em qualquer nó, o PAG não sabe à partida qual o endereço deste. Para o saber, tem de consultar todos os nós, um por um, até o encontrar. A consulta a cada um destes nós é feita numa porta fixa, uma vez que todos os PGAG estabelecem ligação sempre na mesma porta lógica, reservada para o efeito.

Para estabelecer ligação com todos os nós, ou com um conjunto de nós, o PAG tem previamente de saber quais e quantos são estes, o que é conseguido através da requisição ao SNAG (cujo endereço é conhecido à partida) da lista dos seus endereços lógicos. Para este efeito, é enviada uma mensagem do tipo **gethosts**, do PAG para o SNAG, que origina a recepção da respectiva resposta no sentido inverso. Esta resposta contém, como argumento, a lista requisitada.

O passo seguinte consiste na ligação ao PGAG. Se esta ligação não for possível, o PAG funciona localmente sem cooperação externa. Caso contrário, a sequência inicial de troca de mensagens entre estes dois processos é a representada na figura 5.2.

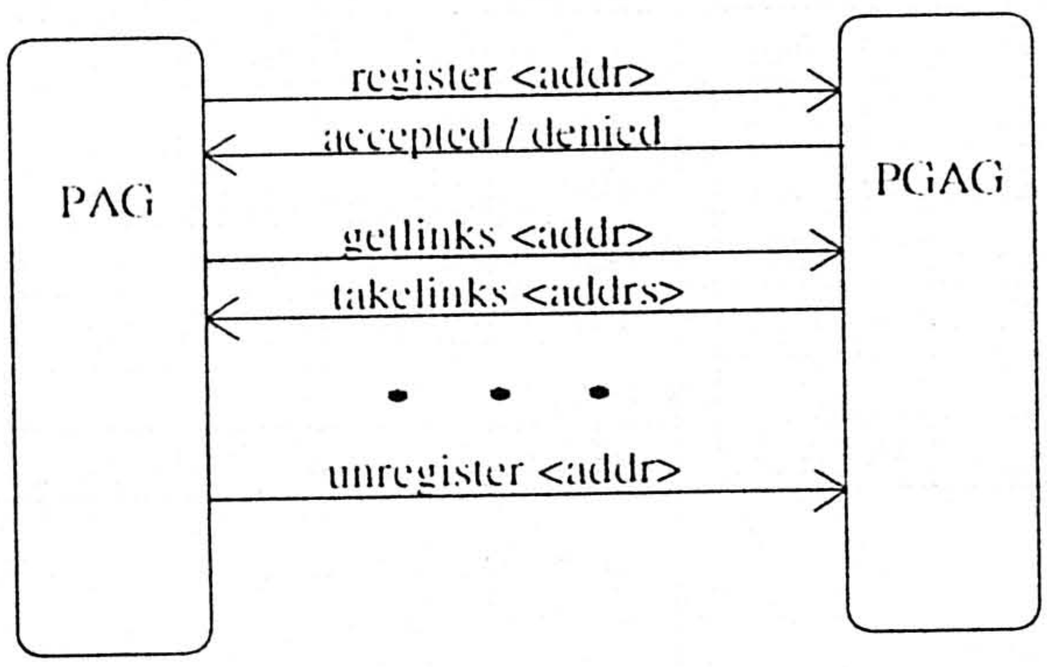


Figura 5.2: Sequência típica de troca de mensagens entre PAG e PGAG

Inicialmente, o PAG regista-se no PGAG através da mensagem **register**, que tem como argumento o endereço lógico daquele. A resposta a esta mensagem indica a aceitação ou recusa do registo. Caso o registo seja aceite, é possível o estabelecimento de ligações com este processo, por parte de outros PAG.

O passo seguinte consiste na obtenção de ligações com outros PAG, através da mensagem **getlinks**, cujo argumento inclui o número de ligações pretendidas. A resposta a esta mensagem inclui uma lista de endereços lógicos de outros PAG em execução (que

se registaram previamente), com os quais o processo pode cooperar no decurso da execução do seu AG local, através da migração periódica de indivíduos.

O envio de indivíduos entre dois PAG é feito com mensagens do tipo **takeind**, que têm como argumento a cópia de um dos seus indivíduos, devidamente convertida para uma representação apropriada ao envio. Estas mensagens não esperam resposta, diminuindo assim o tempo necessário para a troca de informação.

Quando um PAG termina a execução do seu AG local, anula o seu registo no PGAG, para que os novos PAG a iniciar a sua actividade não o incluam na sua lista de ligações. Por outro lado, quando qualquer um dos outros PAG ligados a este último verifica que este terminou a sua execução, ou que já não está a funcionar correctamente, pede ao PGAG um novo elemento ao qual se possa ligar (com a mensagem do tipo **getlinks**).

Os PAG verificam periodicamente se os outros PAG, aos quais estão ligados, se encontram em correcto funcionamento. Esta verificação é conseguida com mensagens do tipo **isalive**, enviadas ao PGAG (o argumento respectivo é o endereço lógico do processo em causa). A resposta a **isalive** informa se a ligação em causa ainda é válida.

Analogamente, todos os PAG devem comunicar periodicamente ao PGAG que se encontram em boas condições de operação, através da mensagem do tipo **iamalive**, cujo argumento é o seu próprio endereço. Desta forma, se ao fim de um determinado tempo o PGAG não receber esta mensagem, deve concluir que aquele processo não se encontra capaz de funcionar correctamente, retirando-o da lista de processos activos. O processo em falta deixa assim de poder constituir par com novos processos, e os processos a ele ligados têm a oportunidade de se ligarem a outros, ao saberem da inoperância daquele (por intermédio das mensagens do tipo **isalive**).

5.2.3. Eleição do PGAG

Em todo o sistema operado pela AGENDA, existe um só PGAG. Na arquitectura apresentada, as tarefas de gestão e de optimização propriamente ditas, foram apresentadas como sendo executadas por dois tipos distintos de processos: o PGAG e os diversos PAG. Embora conceptualmente esta distinção esteja correcta, na prática o processo responsável pelas tarefas de gestão tem um papel duplo, ou seja, também executa as tarefas de optimização.

Desta forma, um PGAG não é mais do que um PAG que assume também o papel de controle, paralelizando as tarefas de optimização com a gestão dos processos da AGENDA. Uma vez que as tarefas de gestão não consomem muito tempo, quando comparadas com a optimização em si, a dualidade de papeis neste processo não afecta significativamente a eficiência global. O PAG/PGAG conduz assim o seu próprio AG local, comunicando com outros PAG e com o PGAG, que é ele próprio. Periodicamente, quando solicitado, efectua as tarefas de gestão referidas.

A comunicação deste PAG com o próprio PGAG é feita de modo análogo à comunicação dos restantes PAG. Apesar de a comunicação ser local, é efectuada exactamente da mesma forma que uma comunicação entre dois processos distintos. Conforme se verá, este aspecto é necessário para manter a simetria da AGENDA, de modo a que seja transparente qual é o PAG que num dado momento assume o papel de PGAG.

Neste contexto, é necessário que um dos PAG seja eleito PGAG. Esta eleição é simplificada ao máximo, uma vez que quando o primeiro PAG arranca, ao verificar a inexistência de um PGAG, assume ele próprio esse papel. Os PAG seguintes, já o vão considerar como o PGAG. Se o primeiro PAG (que é o PGAG) terminar as suas tarefas de optimização antes dos restantes, mantém-se activo apenas para assegurar as tarefas de gestão.

5.2.4. Extensões para Tolerância a Falhas e Migração de Processos

Os aspectos discutidos nesta secção referem-se a possíveis extensões da AGENDA. As questões relacionadas com o aumento da tolerância a falhas no sistema em geral e a possibilidade de redistribuição da carga computacional pelos diversos nós, através da migração de PAG, são aqui analisadas. A sua implementação foi deixada fora do âmbito deste trabalho.

5.2.4.1. Substituição do PGAG

Dada a sua estrutura, a AGENDA está muito dependente do PGAG, que é fundamental para o seu funcionamento, sendo recurso único em todo o sistema. Se este falhar, ou não puder comunicar por qualquer razão, o processo de optimização distribuído pode ficar seriamente comprometido. Se a falha do PGAG ocorrer numa fase inicial, não é mesmo possível completar o estabelecimento de ligações entre os PAG. Por outro lado, se a falha ocorrer numa fase mais avançada, o impacto no sistema é menor, uma vez que os PAG, já em funcionamento, têm as sua próprias ligações estabelecidas.

A tolerância a falhas pode ser aumentada com a existência de um mecanismo de recuperação automática do PGAG, caso este falhe por qualquer razão, ou fique impossibilitado de comunicar com os restantes processos. É no sentido de diminuir esta dependência, que se define um possível protocolo de recuperação, o qual estabelece que quando um qualquer PAG não consegue comunicar com o PGAG, ao fim de um determinado número de tentativas, ele próprio assume o papel deste.

Os restantes PAG, ao verificarem a impossibilidade de ligação com o antigo PGAG, devem procurar o novo PGAG, de forma análoga à realizada no início das suas actividades. Se um dado PAG não conseguir restabelecer a ligação com o PGAG, em nenhum dos nós, conclui que este ainda não foi restabelecido e assume então o seu papel

(é necessário ter cuidado para que não surja mais do que um PGAG, em situações transitórias).

O novo PGAG não tem, à partida, qualquer informação do estado actual do sistema. Para que possa adquirir essa informação, é necessário que os restantes processos o auxiliem. A simples recepção das mensagens do tipo **iamlive**, já indica quais são e quais os endereços dos PAG activos. Estas mensagens podem ainda ser complementadas com outras informações úteis à gestão de processos. Desta forma, ao fim de algum tempo, o novo PGAG acaba por ficar a conhecer qual o estado do sistema, substituindo por completo o seu anterior.

5.2.4.2. Migração de PAG

A possibilidade de um qualquer PAG poder migrar para outro nó, isto é, interromper a sua execução no nó onde se encontra, e passar todo o seu estado para um nó remoto, continuando aí a execução a partir do ponto onde estava, pode constituir uma importante vantagem em algumas situações, para equilibrar a alocação de recursos computacionais à actividade da AGENDA. Em situações em que a carga computacional num determinado nó é muito elevada, ou é necessário diminui-la por qualquer razão (retirando o processamento genético), a migração de PAG pode ser um importante benefício. Para assegurar a migração de processos, é necessário expandir o protocolo de comunicação com mais dois tipos de mensagens, que se resumem na tabela 5.2.

Mensagem	Origem	Destino	Descrição
migrate	PGAG	PAG	Directiva de Migração
launch	PGAG	PGAG	Lançamento de um PAG remoto

Tabela 5.2: Mensagens relacionadas com a migração de PAG

Neste possível contexto (ainda não implementado), a mensagem do tipo **migrate**, com origem no PGAG, tem como efeito a migração do PAG. A migração é feita em duas fases. Na primeira fase, o PAG envia, como resposta àquela mensagem, todo o seu estado actual, ou seja, toda a informação necessária para continuar o AG no ponto onde este se encontrava. Esta informação consiste de alguns factores de parametrização (como o tamanho da população, número de gerações, entre outros), e de todos os indivíduos constituintes da população naquele momento. Após o envio desta informação, aquele PAG deve terminar a sua actividade.

A segunda fase da migração, que tem lugar através da mensagem **launch** entre dois PGAG, consiste no lançamento de um novo PAG remoto, após o envio de todo o estado

do PAG cessante. Deste modo, o novo processo inicia o seu AG local no mesmo ponto onde se encontrava o anterior, sob as mesmas condições de parametrização.

Se, no sistema computacional em questão, não for possível o lançamento remoto de novos processos, este modelo de migração obriga a que exista um PGAG em cada nó para onde seja possível migrar os PAG. Estes PGAG não precisam de coordenar todas as actividades de gestão, tendo simplesmente de assegurar parte da sequência de migração de processos, ou seja, o lançamento de novos PAG. Por esta razão, são considerados PGAG secundários ou auxiliares (na realidade, basta que um dos PAG em cada nó assuma esta responsabilidade).

5.3. ESTRUTURA INTERNA DA AGENDA

Cada um dos processos intervenientes na AGENDA, baseia-se num programa desenvolvido no decorrer deste trabalho, que implementa um AG distribuído, acrescido de um conjunto de tarefas de gestão. O programa deve ser analisado por diferentes pontos de vista, designadamente:

- Estrutura modular do AG distribuído.
- Paradigma de optimização genética.
- Processo de cooperação entre os diversos AG distribuídos.

A estrutura do AG distribuído baseia-se nos modelos apresentados nos capítulos anteriores, sendo constituída por diversas camadas em diferentes níveis, que têm por função isolar diversos pormenores de implementação. As duas camadas de mais alto nível constituem o AG em si, que neste caso implementa o Modelo de Ilhas, no qual é executado em cada ilha um AG Assíncrono (ver capítulos 3 e 4).

Do ponto de vista do paradigma de optimização, o programa implementa a optimização em três distintos paradigmas de AG: os AG Binários (AGB), os AG Combinatórios (AGC) e a Programação Genética (PG) (ver capítulo 2). Em qualquer deles, o ciclo genético consiste sempre na mesma sequência de acções, e a estrutura das camadas é equivalente. A única diferença é a estrutura dos dados manipulados e a forma dos operadores genéticos

5.3.1. Estrutura Modular

A estrutura interna do AG implementado nos programas da AGENDA é constituída por um conjunto de módulos, que operam em diferentes camadas (ver figura 5.3). A primeira camada, designada por AGEN, contém a rotina principal do AG, comum a todas as

variantes destes, assim como as funções principais de inicialização e parametrização. Esta camada existe sob a forma de três variantes, uma para cada um dos paradigmas de AG apresentados.

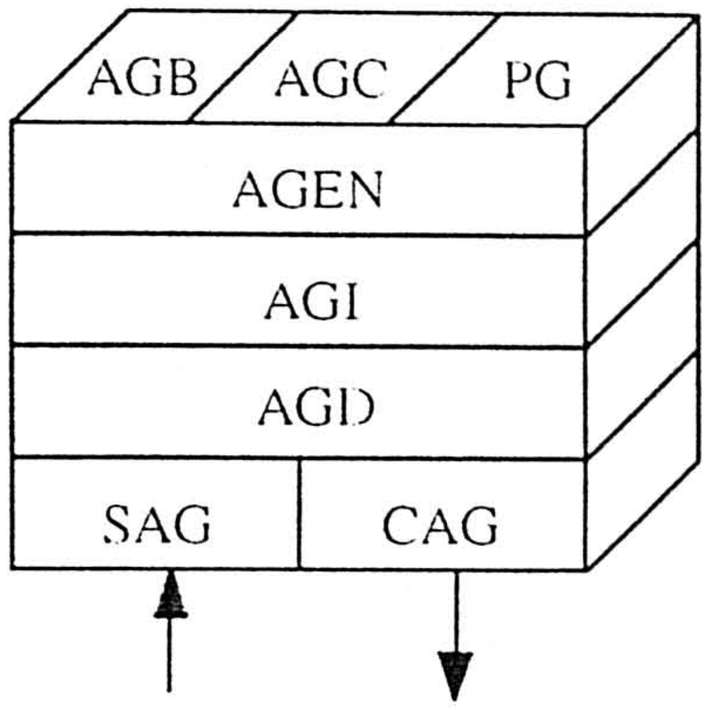


Figura 5.3: Estrututa interna dos elementos da AGENDA

A segunda camada, designada por AGI, é responsável por todos os aspectos relacionados com o Modelo de Ilhas, ao nível do processamento genético (ignorando aspectos de comunicação entre processos), entre os quais se encontram o número de ilhas a manter ligações, as políticas de emigração e imigração (selecção dos indivíduos a emigrar e selecção dos indivíduos a serem substituídos pelos imigrantes), a incorporação na população de indivíduos provenientes de outras ilhas (previamente recolhidos da infraestrutura de comunicação pelas camadas inferiores) e a indicação de envio de indivíduos locais para ilhas remotas (indicação esta dada às camadas inferiores). Nesta camada não existe nenhuma dependência relativamente ao sistema de distribuição disponível, sendo todos os aspectos relacionados com o funcionamento do AG.

A camada seguinte, Algoritmos Genéticos Distribuídos (AGD) serve de ponte de ligação entre as camadas superiores, que incorporam conhecimento relacionado exclusivamente com o AG distribuído, com a camada inferior que controla a comunicação entre os processos intervenientes na AGENDA. Nesta camada, incluem-se tarefas de manipulação de listas de indivíduos em migração, em qualquer um dos sentidos, e de adaptação da informação a trocar entre o AG e a infraestrutura de comunicação, relacionada com aspectos de representação interna e externa (esta adaptação é do nível da aplicação, sendo posteriormente completada por adaptações ao nível do subsistema de comunicação).

A última camada, com carácter dual, consiste no Servidor de AG (SAG), responsável por receber mensagens do exterior, e no Cliente de AG (CAG), responsável por enviar mensagens para o exterior (ambos estão directamente ligados à infraestrutura de comunicação). É também nesta camada que se encontram as rotinas de gestão do PGAG. A sua estrutura é descrita na próxima secção.

5.3.2. Comunicação entre Processos

A comunicação entre os processos intervenientes na AGENDA obedece ao protocolo de comunicação anteriormente descrito. A implementação deste protocolo é feita na camada inferior da estrutura apresentada, que é constituída por duas partes complementares, o SAG e o CAG. Estas últimas, estão directamente ligadas ao subsistema de comunicação entre processos. O subsistema de comunicação pode ser directamente fornecido pelo sistema operativo (por exemplo, a camada de *sockets*), ou por uma camada intermédia, baseada naquele (por exemplo, o sistema *RPC*).

A comunicação é estabelecida entre pares de processos, segundo o paradigma *cliente-servidor*. Neste paradigma, um dos processos aguarda o estabelecimento da comunicação, representando o papel de servidor, enquanto que o outro inicia a comunicação, representando o papel de cliente. A comunicação em si não é mais do que a troca de um conjunto de mensagens, definidas pelo protocolo já descrito.

O módulo CAG é o responsável por tomar a iniciativa de comunicação, quando pretende enviar mensagens a outros processos. No outro extremo do canal de comunicação, o SAG detecta um pedido de comunicação, e processa-o de acordo com o protocolo, após decifrar qual o tipo de mensagem. De aí em diante, são trocadas as mensagens necessárias de acordo com a situação, nas quais é passada informação entre os dois processos. Uma vez terminada esta sequência, o CAG devolve o controle às camadas superiores do seu processo, enquanto que o SAG fica livre para processar novos pedidos de comunicação (este último funciona em paralelo com as tarefas de optimização).

Conforme já referido, pelo menos um dos processos assume simultaneamente o papel duplo de PAG e PGAG, sendo umas vezes cliente (quando contacta o PGAG, que é ele próprio) e outras vezes servidor (quando um PAG remoto o contacta). A comunicação, nestes casos, é estabelecida da mesma maneira que no caso em que o cliente e o servidor são distintos, para que qualquer processo possa assumir o papel de PGAG transparentemente.

Só os PGAG é que assumem o papel real de servidor. A comunicação entre dois PAG é feita de forma simplificada, com mensagens do tipo **takeind** que não obrigam a resposta. A razão desta simplificação é diminuir o tempo necessário na troca de informação entre estes processos, já que esta é a mais frequente. Por outro lado, a perda de uma mensagem deste tipo não é tão grave como nas restante, uma vez que o AG é praticamente insensível à perda de uma pequena parte do fluxo de informação genética por migração (conforme visto no capítulo anterior, a diferença da evolução para pequenas variações do número de emigrantes é muito pequena).

Por fim, para que seja possível ao processo responsável pela gestão assumir os papeis de PAG/PGAG, é necessário paralelizar as tarefas de optimização com as tarefas de gestão. Para este efeito, o decorrer do ciclo de optimização genética (no qual é também assegurada a migração) tem de ser interrompido por solicitações de tarefas de gestão por parte de outros processos (ou mesmo de si próprio, no que respeita à sua faceta de

PAG). Estas interrupções podem ser feitas periodicamente, ou despoletadas por eventos de comunicação.

5.3.3. Aspectos de Implementação

O protótipo AGENDA foi implementado num conjunto de sistemas operativos *UNIX*, nomeadamente *SunOS 4.1.3*, *Mach 3.0*, ambos do tipo *BSD*, e *Solaris 5.3*, do tipo *System V*. As diferenças encontradas nestes dois tipos de sistemas operativos obrigaram a um conjunto de considerações.

5.3.3.1. Comunicação Inter-Processos

A comunicação entre processos, implementada nas camadas inferiores da estrutura do programa (CAG e SAG), pode ser adaptada com relativa facilidade a um conjunto de subsistemas de comunicação (basta alterar esta camada para adaptar o programa a um novo subsistema).

O subsistema de comunicação escolhido na implementação do protótipo da AGENDA foi o mecanismo de *sockets*. A utilização deste mecanismo assegura uma grande generalidade, permitindo um controle refinado da comunicação entre processos. Por outro lado, obriga a ter em consideração uma série de pormenores de comunicação, entre os quais a compatibilização da informação trocada (a representação dos vários tipos de informação varia de máquina para máquina) e a sincronização da comunicação.

A utilização de sistemas *RPC* foi também considerada. Com estes subsistemas, ficam resolvidos à partida os pormenores da comunicação referidos, mas o controle da comunicação não é tão refinado. Por outro lado, a ausência de multiprocessamento inter-processo (por exemplo, os *threads*) pode levantar alguns problemas, nomeadamente na implementação da dualidade cliente-servidor nos processos com papel duplo (PAG/PGAG).

Nestes casos, um processo tem de assegurar simultaneamente o papel de servidor, aguardando comunicação, e o papel de cliente, iniciando a comunicação. Se existirem *threads* no sistema, e o sistema de *RPC* for compatível com eles, tudo se resolve com facilidade (basta lançar dois *threads*, um para cada um dos papeis). Caso contrário, como os habituais sistemas de *RPC* não se compadecem da melhor forma com este paralelismo interno, a paralelização tem de ser feita por recurso a outros artifícios menos directos, como por exemplo a existência de dois processos directamente ligados, um responsável pelas tarefas de gestão (servidor) e outro pelas tarefas de optimização (cliente).

A implementação deste protótipo sobre *PVM* (ver capítulo 3, secção 3.2.2.2) é também simplificada pela estrutura em camadas do programa, sendo relativamente simples a adaptação da camada de baixo nível (CAG/SAG) a este subsistema de comunicação. Os pormenores da adaptação não foram ainda considerados.

5.3.3.2. Paralelização Intra-Processo

No que respeita à paralelização das tarefas de optimização e gestão dentro de um processo convencional, foram utilizados mecanismos de comunicação assíncrona. Nos sistemas do tipo *BSD*, é possível interceptar interrupções ao processo originadas por iniciativas de comunicação de outros processos [Stevens90]. Nestes casos, quando existe um pedido pendente de comunicação, o processo é sinalizado, interrompendo as suas tarefas para processar o pedido.

Nos sistemas do tipo *System V*, este artifício não é possível, pelo que se recorreu aos tradicionais *signals* do sistema operativo UNIX [Stevens90], que periódicamente verificam se existem pedidos de comunicação pendentes nos *sockets* respectivos (estes foram colocados no modo assíncrono [Stevens90]).

Por fim, a paralelização interna da optimização em cada nó foi assegurada pelo mecanismo de *threads*, disponível em *Mach 3.0* e *Solaris 5.3*, embora em versões diferentes.

5.4. TESTE DA AGENDA

Esta secção analisa os resultados obtidos na aplicação do protótipo da AGENDA, desenvolvido no decorrer deste trabalho, a um problema de optimização real, de considerável complexidade. Toda a análise tem por base de comparação a abordagem ao problema pelo AG original, não distribuído, no sentido de evidenciar as diferenças originadas pela AGENDA.

Em primeiro lugar, é feita uma descrição resumida do problema, na qual se apresentam as suas principais características. Em seguida, são apresentados os resultados da aplicação do AG a este problema, segundo as abordagens não distribuída (AG original) e distribuída pela AGENDA. Esta última levanta algumas considerações de parametrização, que são devidamente analisadas.

Com base nestes resultados, é feita uma comparação de ambos os métodos, segundo a qualidade das soluções obtidas e segundo o tempo real de processamento necessário para as obter, evidenciando claramente quais as vantagens da distribuição.

5.4.1. Breve Descrição do Problema de Teste

O problema que serviu de teste a toda a discussão desta secção consiste num problema real (não é um simples exercício teórico), no qual se pretende encontrar um programa capaz de efectuar a análise dinâmica da estabilidade de um sistema de produção e

distribuição de energia eléctrica [Lopes94]. Este problema, detalhadamente descrito no anexo C, enquadra-se no paradigma da Programação Genética (ver capítulo 2).

O que se pretende é que o AG consiga criar um conjunto de programas em LISP, capazes de avaliar, em poucos instantes, o grau de estabilidade do sistema referido, tendo como entrada um conjunto de parâmetros operacionais (que consistem em grandezas eléctricas e ambientais).

Este problema foi escolhido para o teste da plataforma devido a um conjunto de factores, em especial pelo seu interesse efectivo e pela sua elevada complexidade, útil na avaliação da AGENDA (cujo principal objectivo é resolver problemas complexos).

5.4.2. Aplicação do Algoritmo Genético Original

Na aplicação do AG original a este problema, foi adoptado o paradigma da Programação Genética. A parametrização neste caso é um pouco mais complexa do que nos habituais cromossomas binários, sendo necessário considerar um conjunto de novos factores, fundamentais no controle da evolução das soluções e relacionados com o ajuste da sua complexidade.

As soluções geradas podem ser vistas como expressões convencionais de LISP. Cada expressão é aplicada ao sistema, sob circunstâncias especiais, sendo possível avaliar a qualidade das suas previsões de estabilidade (ver anexo C).

Estas expressões, guardadas nos cromossomas dos indivíduos da população, indicam para cada situação do sistema, a estabilidade ou não estabilidade deste. Os indivíduos têm um valor de adaptação (ou desempenho) igual à percentagem de acertos conseguida nas previsões realizadas para um conjunto de situações conhecidas à priori, para as quais se sabe antecipadamente o resultado correcto (o resultado só pode assumir dois valores: *sistema estável* ou *sistema não estável*).

Um dos problemas que surge na aplicação do AG a este problema, é a forma de controlar a complexidade dos programas de LISP gerados. O tempo de avaliação das soluções (que neste caso consiste em praticamente todo o tempo de execução do algoritmo) cresce exponencialmente com o tamanho destes programas. Para equilibrar este efeito, foram criados mecanismos que controlam a complexidade das soluções geradas [Lopes95].

No entanto, é necessária uma certa complexidade nos programas para assegurar a sua qualidade, pelo que o tempo de processamento do AG é muito elevado, conforme se verá adiante.

Os resultados obtidos demonstram que o AG é capaz de encontrar boas soluções para problemas desta complexidade, conforme se pode ver na figura 5.4. O melhor valor da solução final na abordagem com uma população de 128 indivíduos foi de 85.4% de acertos nas previsões realizadas, enquanto que na população de 1024 indivíduos foi de

98.3% de acertos. O gráfico da figura 5.4 mostra a evolução média de 10 execuções do AG, para 128 e 1024 indivíduos, em função do número de geração.

Conforme se pode ver no gráfico, existe uma significativa diferença entre as abordagens com os dois tamanhos da população. A qualidade da solução exigida pelas condições do problema leva a que se considere a utilização de 1024 indivíduos na aplicação do AG.

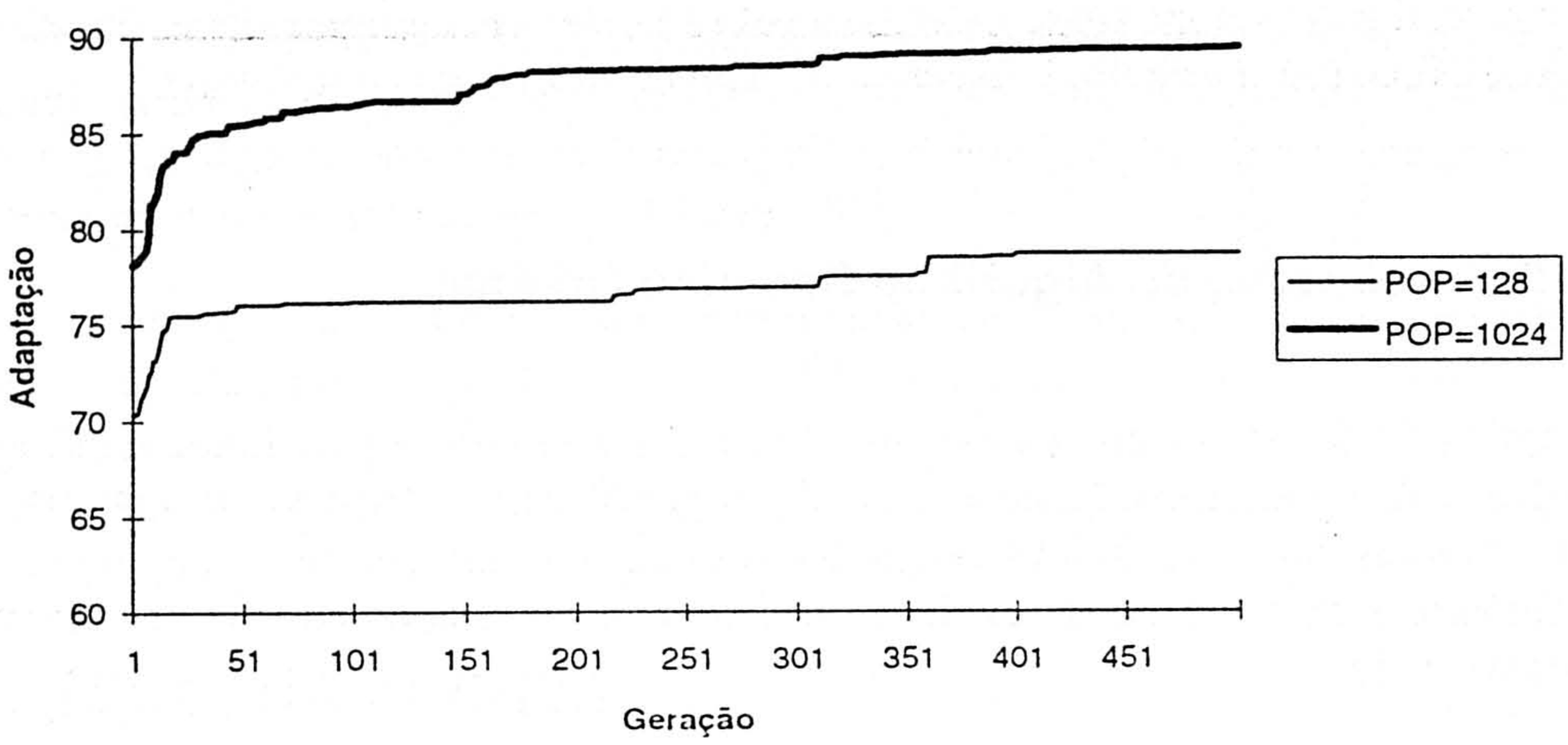


Figura 5.4: Resultados da aplicação do AG original ao problema de teste

O número de gerações considerado teve em conta o compromisso entre a qualidade das soluções, a convergência do algoritmo, o tempo de processamento e o grau de crescimento da complexidade das programas gerados. Ponderados estes factores, foram adoptadas 500 gerações, cuja execução demora em média cerca de 60 horas de processamento.

5.4.3. Aplicação e Parametrização da AGENDA

Na aplicação da AGENDA ao problema de teste, tem de ser considerado um novo conjunto de parâmetros de controle do algoritmo, nomeadamente o número de ilhas e o tamanho das suas subpopulações, o número de emigrantes, o intervalo entre migrações e a topologia das ligações entre as ilhas.

No que respeita ao número de ilhas e ao tamanho das suas subpopulações, deve ser tido em consideração que é necessário um número elevado de indivíduos na população global para contrapor a complexidade inerente ao problema, evidenciada na figura 5.4. Por esta razão, são considerados no total 1024 indivíduos, divididos por 8 subpopulações.

A razão para a escolha de 8 subpopulações é motivada por dois factores. O primeiro, tem a ver com o facto de problemas complexos necessitarem de subpopulações numerosas nas ilhas, para que haja material genético suficiente na evolução local. Este efeito foi considerado no capítulo 4, na aplicação ao problema *PCV96*, que devido à sua complexidade reagia melhor a subpopulações relativamente grandes. Esta questão é foco de análise de recentes publicações no campo da análise de AG [Goldberg95].

O segundo factor relaciona-se com razões de ordem prática. Existiam disponíveis para realizar a experiência 8 processadores, em 4 máquinas diferentes, pelo que foi atribuído a cada processador um AG local, responsável por uma só das subpopulações. A tabela 5.3 descreve resumidamente as máquinas envolvidas na experiência, evidenciando o número de processadores e o tipo de infraestrutura de comunicação que assegura a ligação às restantes. Saliente-se que a máquina D se encontra ligada por um *Modem* de muito baixo débito (19200 *baud*).

Máquina	Número de Processadores	Sistema Operativo	Infraestrutura de Comunicação
A	1	SunOs	Ethernet (10 *Mbits*/s)
B	1	SunOs	Ethernet (10 *Mbits*/s)
C	4	Solaris	Ethernet (10 *Mbits*/s)
D	2	Solaris	Modem (19200 *baud*)

Tabela 5.3: Máquinas aplicadas à resolução do problema

No que respeita ao número de emigrantes, foi escolhido o valor de 10%, tendo em conta que valores maiores não trazem grande benefício e aumentam o tempo gasto na comunicação (ver capítulo anterior).

O intervalo entre migrações utilizado foi de 10 gerações. Este valor acelera a convergência à custa de uma pequena degradação da qualidade. Tendo em conta que este problema leva muito tempo a convergir (dada a sua complexidade), o compromisso entre a qualidade da solução e o tempo para a obter, obriga à utilização deste valor. Conforme visto no capítulo anterior, a utilização de valores mais elevados, embora atinjam melhores soluções, demoram significativamente mais tempo.

A escolha da topologia foi simples. Como não existe nenhuma regra absoluta para a sua escolha, foi utilizada a Topologia Aleatória, uma vez que não deu maus resultados nas simulações do capítulo anterior. O facto de esta topologia ser aleatória, facilita o estabelecimento de ligações entre os diversos processos da AGENDA, diminuindo a complexidade das tarefas de gestão relacionadas com o reestabelecimento de ligações perdidas.

5.4.4. Comparação da Qualidade das Soluções

A análise da abordagem distribuída com a AGENDA considera dois factores. O primeiro, detalhado nesta secção, compara a qualidade das soluções obtidas com as equivalentes do AG original. O gráfico da figura 5.5 salienta os seus valores relativos, tendo como base de comparação a subpopulação de 128 indivíduos e a superpopulação equivalente de 1024 indivíduos, ambas operadas pelo AG original (ver capítulo anterior).

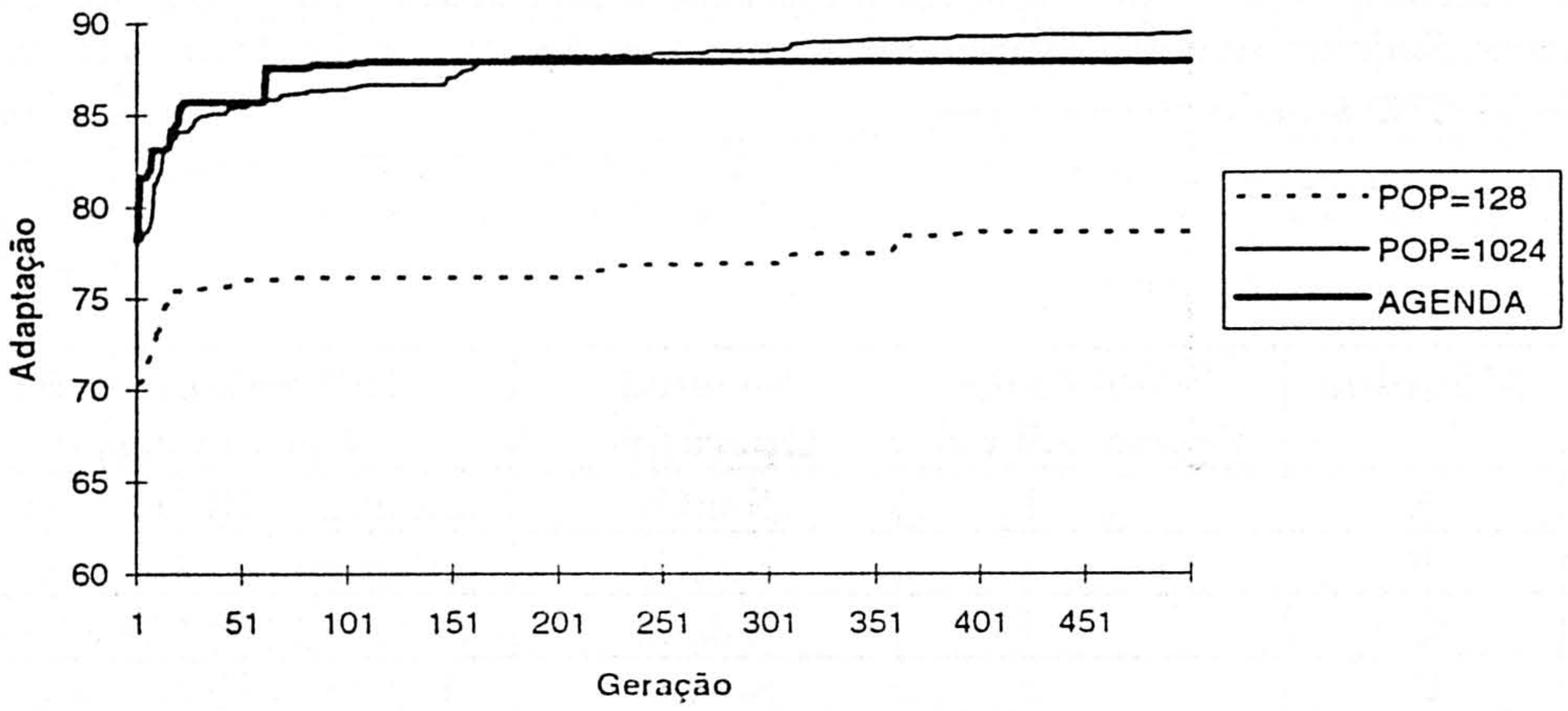


Figura 5.5: Comparação da qualidade das soluções obtidas com a AGENDA e com o AG original

Conforme se verifica na figura 5.5, a abordagem com a AGENDA consegue obter valores significativamente melhores do que a abordagem convencional de 128 indivíduos (conforme seria de esperar) e aproxima-se da curva do AG original com a superpopulação equivalente. A AGENDA converge mais rapidamente, mas não chega a atingir os valores finais obtidos nesta curva. Os valores finais obtidos nas abordagens consideradas são evidenciados na tabela 5.4.

Abordagem	Resultado
AG POP=128	78.62
AG POP=1024	89.46
AGENDA 8xPOP=128	87.93

Tabela 5.4: Soluções encontradas na resolução de PL

As diferenças relativamente à simulação do Modelo de Ilhas do capítulo anterior devem-se ao assincronismo entre as diversas ilhas. Naquelas simulações, as diferentes ilhas em cooperação convergiam para valores melhores, mas demorando mais tempo, ao contrário do que se passa na AGENDA, na qual a convergência é mais rápida, mas para valores piores.

A razão desta diferença deve-se ao facto de a evolução dos diversos AG locais, em cada processador, se dar a velocidades muito diferentes, existindo assim processadores muito mais rápidos do que outros. Desta forma, algumas subpopulações avançam "sózinhas", não conseguindo incorporar material genético exterior com diversidade suficiente para evitar a convergência prematura, uma vez que as restantes subpopulações ainda não evoluíram o suficiente.

Estas diferenças de velocidade degradam um pouco a qualidade global da evolução da pesquisa, mas oferecem uma importante vantagem em termos do tempo de processamento, conforme se descreve na próxima secção.

5.4.5. Comparação dos Tempos de Execução

Para comparar a eficiência da AGENDA, em relação ao AG original, foram realizadas uma série de experiências, no conjunto das máquinas apresentadas na tabela 5.3. A cada processador deste conjunto (algumas das máquinas têm mais do que um processador) foi atribuída uma ilha, para ser operada por um AG local da AGENDA. O tempo de execução de cada um destes AG locais dependeu da capacidade do processador que lhe foi atribuído, e também da respectiva carga computacional a que este estava sujeito, por força de outras tarefas exteriores ao problema.

Numa situação de distribuição ideal, seria de esperar que o tempo de execução do AG original fosse dividido pelo número de processadores do sistema. Se os processadores tivessem todos a mesma capacidade e disponibilidade, a distribuição pelos 8 processadores do sistema apresentado faria com que este tempo fosse 8 vezes inferior.

No entanto, a capacidade e disponibilidade para o processamento das tarefas da AGENDA neste sistema difere muito de processador para processador, sendo necessário evidenciar as diferenças para fazer uma comparação justa. Assim, a tabela 5.5 atribui um coeficiente a cada processador, que traduz o seu desempenho em relação ao processador mais eficiente (processador A). Estes coeficientes foram calculados com base na observação dos tempos de execução das diversas experiências realizadas. Para as máquinas multiprocesador, C e D, os coeficientes são a média dos coeficientes directos dos seus processadores.

Máquina	Coeficiente
A	1
B	0.34
C	0.29
D	0.24

Tabela 5.5: Coeficientes da capacidade dos processadores das máquinas A, B, C e D

Desta forma, o sistema pode ser visto como contendo um determinado número de processadores, iguais ao processador mais eficiente (ou mais disponível), que é o processador da máquina A. Esse número de processadores não é mais do que a soma de todos os coeficientes apresentados, considerando que as máquinas C e D são multiprocessador (o coeficiente neste caso é previamente multiplicado pelo número de processadores da máquina). A soma dos coeficientes aproxima-se de 3, ou seja, os 8 processadores do sistema têm um desempenho semelhante a 3 processadores iguais ao melhor deles, nas mesmas circunstâncias.

O gráfico da figura 5.6 compara a evolução da qualidade das soluções obtidas no AG original e na AGENDA, em função do tempo real de processamento. O gráfico contem 3 curvas, a curva *AG 1024*, a curva *AGID* e a curva *AGENDA*. A primeira, representa a execução do AG original com 1024 indivíduos, no processador da máquina A.

A segunda curva, referida neste gráfico por *AGID* (AG Idealmente Distribuído), representa a evolução que se poderia esperar num sistema distribuído, em condições ideais, contendo os já referidos 3 processadores semelhantes ao processador da máquina A, nas mesmas condições de operação. Esta curva, não é mais do que a redução da curva *AG 1024* por um factor de 3, na escala do tempo. Por fim, a terceira curva ilustra o desempenho da AGENDA.

Repare-se ainda que o eixo do tempo, no gráfico da figura 5.6, representa o tempo real de processamento e não o número de gerações. Este eixo tem uma escala não linear, para contrabalançar a melhoria cada vez mais lenta da qualidade das soluções, representando inicialmente intervalos de um minuto, seguidos de intervalos de dez minutos e terminando com intervalos de uma hora.

Os tempos considerados na avaliação da AGENDA já incluem o tempo gasto na comunicação entre processos, que pode ser demorado, sobretudo no caso da máquina D por estar ligada às restantes com uma linha de comunicação de muito baixo débito.

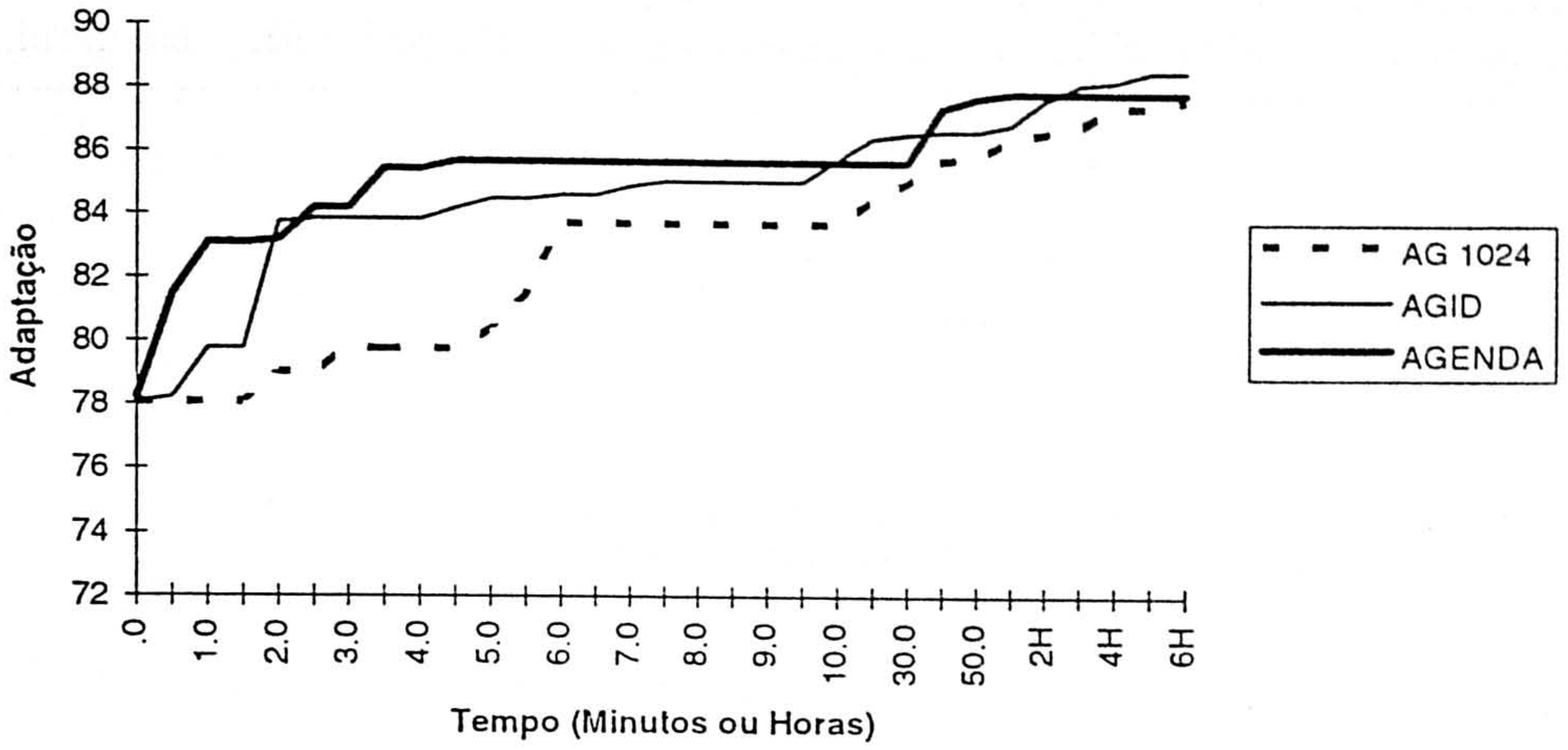


Figura 5.6: Evolução das soluções em função do tempo

O gráfico anterior (figura 5.6) mostra que a AGENDA conseguiu resultados melhores do que se poderia esperar, no que respeita ao tempo de processamento. O facto de a sua curva nesse gráfico se situar, na maior parte do tempo, acima da curva esperada para o AG distribuído em condições ideais (referida na figura por *AGID*), significa que a AGENDA não degradou a eficiência do processo de pesquisa genética, chegando mesmo a melhorá-la, mesmo tendo em conta o tempo gasto na comunicação entre os diversos AG locais.

Esta melhoria de eficiência foi conseguida à custa de uma pequena degradação de qualidade das soluções finais (ver figura 5.5). As diferentes velocidades de evolução nas várias ilhas, embora desfavoreçam esta qualidade, melhoram o tempo necessário para as atingir.

Assim, com a abordagem da AGENDA, foi possível obter melhores soluções, pelo menos nas primeiras 6 horas de execução. A qualidade das soluções finais, atingidas ao fim de 60 horas, foi ligeiramente melhor no AG original, mas o tempo necessário para a atingir é demasiado grande.

Para melhorar a qualidade das soluções da AGENDA, são possíveis duas alternativas. A primeira solução consiste em utilizar um sistema simétrico, sob o ponto de vista das velocidades de processamento nas diferentes ilhas. Estes sistemas, conforme visto na simulação do Modelo de Ilhas no capítulo anterior, consegue melhores resultados do que o AG original, em termos da qualidade final das soluções, neste caso à custa de alguma degradação no tempo necessário para as obter.

A segunda solução é talvez a mais indicada, consistindo em aumentar ligeiramente o tamanho das populações locais das ilhas operadas pela AGENDA, provocando uma melhoria da qualidade das soluções obtidas, aumentando um pouco o tempo de

processamento. É possível atingir um bom compromisso entre estes dois factores, possibilitando uma melhor aplicação da AGENDA e dos AG distribuídos, em geral.

CONCLUSÕES

Este capítulo está dividido em duas secções. A primeira secção apresenta as conclusões deste trabalho, tendo em conta os objectivos propostos. A segunda secção refere algumas possíveis extensões a realizar, com a intenção de aproveitar e melhorar os resultados aqui alcançados.

6.1. CONCLUSÕES DESTE TRABALHO

Os resultados alcançados com este trabalho demonstram que foi atingido o principal objectivo, ou seja, foi possível implementar eficientemente o AG num sistema de computação distribuída, sem degradar o seu desempenho.

Conforme se verificou no capítulo 5, o protótipo da plataforma de distribuição AGENDA, aqui concebida, possibilitou a execução distribuída do AG aplicado a um problema real, de considerável complexidade.

A principal vantagem da abordagem distribuída ficou bem patente nos resultados obtidos, ou seja, foi possível dividir o tempo de processamento do algoritmo pelos processadores do sistema, diminuindo assim muito satisfatoriamente o tempo necessário para a resolução do problema.

As condições do sistema de computação em que foi aplicada a AGENDA ofereciam algumas dificuldades, relacionadas com a sua elevada carga (devido a outras tarefas exteriores a este trabalho). Mesmo assim, a análise detalhada dos resultados demonstra que a AGENDA adaptou satisfatóriamente o AG a este ambiente, em certa medida ultrapassando os resultados esperados.

Mesmo tendo em conta o tempo necessário às tarefas de comunicação entre os diversos elementos do processo de optimização distribuída, a AGENDA possibilitou que, até certo ponto, o tempo de processamento necessário à obtenção de boas soluções fosse dividido pelo menos por um factor igual ao número de processadores do sistema.

No que respeita à qualidade das soluções finais obtidas com a AGENDA, estas foram ligeiramente pior do que o esperado, mas aproximaram-se muito da qualidade das soluções obtidas com o AG convencional, nas mesmas condições de volume de processamento.

A ligeira diminuição desta qualidade é devida à assimetria na velocidade de execução dos AG locais das diferentes ilhas, mas só se reflete nas últimas gerações do algoritmo, que ocupam uma parte significativa de todo o processamento. Este efeito pode ser perfeitamente contrabalançado com a utilização de um sistema com velocidades de processamento semelhantes em todos os processadores (conforme se verificou na simulação do Modelo de Ilhas, no capítulo 4).

Mesmo nas condições adversas referidas, a qualidade das soluções finais pode ser melhorada, com uma correcta parametrização da AGENDA, na qual é possível encontrar um bom compromisso entre tempo de processamento e qualidade pretendida. Conforme se verificou no capítulo 4, o grau de convergência de um AG distribuído pode ser controlado, influenciando a relação entre a qualidade das soluções e o tempo necessário para as obter, sendo assim possível utilizar o tempo da forma mais conveniente.

Ficou assim demonstrado que o AG não é prejudicado pela reestruturação necessária à sua implementação num sistema de computação distribuída. Verificou-se ainda que, mesmo em situações adversas, como diferentes velocidades de processamento dos AG locais das diversas ilhas e atrasos de comunicação entre estes, o AG distribuído consegue atingir um óptimo desempenho.

Mais uma vez, o AG demonstrou o seu enorme poder de adaptação, neste caso não a um novo problema complexo, mas sim à sua própria implementação distribuída, a qual alterou de forma significativa as suas condições de operação.

6.2. SUGESTÕES PARA FUTURAS EXTENSÕES

A extensão directa a este trabalho, consiste na conclusão do protótipo da plataforma de distribuição AGENDA, no sentido de implementar a tolerância a falhas e a migração de processos, da forma definida no capítulo 5.

Seguidamente, existem diversas direcções possíveis para futuros desenvolvimentos. Seria interessante verificar o comportamento do AG distribuído em condições diferentes, que

complementem os resultados aqui obtidos, utilizando outros parâmetros de controle de execução, e se possível, outros problemas de optimização, com diferentes características.

Teria também interesse avaliar o benefício do AG distribuído em problemas cujo ponto óptimo varia com o tempo. Estes problemas oferecem dificuldades adicionais aos AG com uma só população.

A análise do efeito da divisão da população global em subpopulações também pode ser desenvolvida, verificando qual o impacto da variação de um conjunto de factores que a controlam. Este tema é objecto de recentes publicações na área da análise de AG [Goldberg95].

Por fim, seria também interessante adaptar outros modelos de AG a sistemas de computação distribuída, nomeadamente a implementação do AG Celular ou do AGA em sistemas de computação massivamente paralelos.

EFEITOS DA MIGRAÇÃO NO MODELO DE ILHAS

1. DESCRIÇÃO DAS EXPERIÊNCIAS

As experiências realizadas para o Modelo de Ilhas tiveram especial atenção, devido à diversidade de parâmetros de controle deste modelo. Os principais factores analisados tiveram em atenção o efeito combinado desses parâmetros, entre os quais se incluem o intervalo entre migrações (T), a percentagem da população que emigra (N) e a topologia e quantidade de ilhas.

As tabelas apresentadas neste anexo mostram quais as melhores combinações de N e T obtidas, para cada uma das topologias descritas, para 8 e 16 populações. Os resultados são repetidos para cada problema da base de teste (capítulo 4), e servem de base à discussão do capítulo 5.

As experiências com os parâmetros N e T foram realizadas para todas as combinações possíveis, com N a assumir valores de 1, 10, 20 e 30% da população e T a assumir valores de 10, 20 e 30 gerações.

BIBLIOTECA F.E. PORTO

2. RESULTADOS OBTIDOS

2.1. Problema *F6*

Topologia	8 Ilhas	16 Ilhas
Anel	N=1%, T=20, V=4.09	N=1%, T=10, V=2.57
Anel Bidir.	N=10%, T=30, V=3.66	N=10%, T=30,V=3.77
Anel Duplo	N=10%, T=30, V=4.97	N=10%, T=30, V=4.00
Anel Duplo Bidir.	N=10%, T=30, V=4.06	N=10%, T=10, V=3.69
Completa	N=10%, T=20, V=4.49	N=20%, T=30, V=4.10
Estrela	N=30%, T=20, V=4.58	N=30%, T=20, V=6.42
Aleatória 3	N=10%, T=30, V=4.40	N=10%, T=20, V=3.03
Hiper-Cubo	N=10%, T=30, V=3.88	N=20%, T=30,V=3.74
Torus	N=10% , T=20, V=3.61	N=10%, T=30,V=3.84
Árvore 2 Bidir.	N=30% , T=30, V=3.45	N=10%, T=10, V=3.95

Tabela A.1: Resultados do Modelo de Ilhas para *F6*

2.2. Problema F7

Topologia	8 Ilhas	16 Ilhas
Anel	N=1%, T=20, V=1.17	N=1%, T=10, V=1.84
Anel Bidir.	N=20%, T=20, V=2.40	N=10%, T=30, V=0.90
Anel Duplo	N=30%, T=10,V=3.20	N=10%, T=30, V=1.44
Anel Duplo Bidir.	N=20%, T=10, V=2.59	N=10%, T=30, V=0.92
Completa	N=30%, T=30, V=2.07	N=20%, T=30, V=2.70
Estrela	N=30%, T=30, V=2.07	N=20%, T=30, V=2.18
Aleatória 3	N=30%, T=30, V=2.29	N=10%, T=20, V=1.04
Hiper-Cubo	N=30%, T=20, V=2.28	N=10%, T=30, V=1.51
Torus	N=30%, T=20, V=2.22	N=10%, T=20, V=1.56
Árvore 2 Bidir.	N=20%, T=10, V=1.41	N=10%, T=30, V=1.39

Tabela A.2: Resultados do Modelo de Ilhas para *F7*

2.3. Problema *F8*

Topologia	8 Ilhas	16 Ilhas
Anel	N=1%, T=20, V=0.10	N=1%, T=10, V=0.09
Anel Bidir.	N=10%, T=30, V=0.09	N=10%, T=30, V=0.07
Anel Duplo	N=10%, T=20,V=0.14	N=10%, T=30, V=0.08
Anel Duplo Bidir.	N=10%, T=30, V=0.09	N=10%, T=30, V=0.08
Completa	N=10%, T=20, V=0.13	N=20%, T=30, V=0.10
Estrela	N=10%, T=30, V=0.13	N=20%, T=30, V=0.10
Aleatória 3	N=10%, T=30, V=0.14	N=10%, T=30, V=0.08
Hiper-Cubo	N=20%, T=20, V=0.11	N=20%, T=30, V=0.08
Torus	N=30%, T=30, V=0.07	N=10%, T=20, V=0.08
Árvore 2 Bidir.	N=10%, T=30, V=0.15	N=20%, T=30, V=0.11

Tabela A.3: Resultados do Modelo de Ilhas para *F8*

2.4. Problema *M80*

Topologia	8 Ilhas	16 Ilhas
Anel	N=1%, T=10, V=1.05	N=10%, T=30, V=0.20
Anel Bidir.	N=10%, T=20, V=1.15	N=10%, T=30, V=0.05
Anel Duplo	N=10%, T=30, V=0.60	N=20%, T=20, V=0.20
Anel Duplo Bidir.	N=10%, T=10, V=0.90	N=10%, T=20, V=0.05
Completa	N=10%, T=10, V=0.85	N=30%, T=20, V=0.05
Estrela	N=30%, T=10, V=1.50	N=30%, T=20, V=0.30
Aleatória 3	N=10%, T=10, V=0.85	N=30%, T=20, V=0.05
Hiper-Cubo	N=10%, T=30, V=0.90	N=30%, T=10, V=0.05
Torus	N=10%, T=10, V=1.00	N=20%, T=30, V=0.05
Árvore 2 Bidir.	N=30%, T=20, V=1.35	N=20%, T=30, V=0.30

Tabela A.4: Resultados do Modelo de Ilhas para *M80*

2.5. Problema *PCV96*

Topologia	8 Ilhas	16 Ilhas
Anel	N=10%, T=10, V=859.9	N=10%, T=10, V=874.7
Anel Bidir.	N=20%, T=10, V=849.9	N=20%, T=10, V=880.6
Anel Duplo	N=10%, T=10, V=842.4	N=10%, T=10, V=885.1
Anel Duplo Bidir.	N=10%, T=10, V=933.5	N=20%, T=10, V=861.3
Completa	N=10%, T=10, V=844.2	N=20%, T=10, V=849.7
Estrela	N=30%, T=10, V=866.6	N=30%, T=10, V=901.2
Aleatória 3	N=10%, T=10, V=844.7	N=10%, T=10, V=885.7
Hiper-Cubo	N=20%, T=10, V=853.2	N=30%, T=10, V=863.7
Torus	N=10%, T=10, V=841.5	N=10%, T=10, V=852.4
Árvore 2 Bidir.	N=30%, T=10, V=875.4	N=20%, T=10, V=894.7

Tabela A.5: Resultados do Modelo de Ilhas para *PCV96*

PARÂMETROS DE CONTROLE DO PROGRAMA AGENDA

B.1. O PROGRAMA AGENDA

O programa AGENDA, desenvolvido ao longo deste trabalho, permite a execução dos diversos modelos de AG analisados, tanto por simulação de distribuição (ver capítulo 4) como pela distribuição real definida na plataforma AGENDA (ver capítulo 5).

Os modelos implementados por simulação são os seguintes:

- AG original
- AGA
- Modelo de Ilhas com AG canónico ou AGA
- Modelo das Populações Sobrepostas com AG canónico ou AGA

No que respeita à distribuição real, esta é definida pela plataforma AGENDA e baseia-se no conjugação do Modelo de Ilhas com o AGA. O programa AGENDA permite a comunicação entre ilhas remotas, em execução em diferentes nós do sistema computacional. A paralelização interna em cada ilha é também disponibilizada por este programa, por intermédio do mecanismo de *threads*.

O programa AGENDA é controlado por uma conjunto de parâmetros e ficheiros de controle, descritos na secção seguinte.

B.2. INVOCAÇÃO E OPÇÕES

B.2.1. Invocação do Programa

A forma de invocar o programa depende do paradigma de AG em causa, designadamente os AG binários, os AG combinatórios ou a Programação Genética. Na sua forma geral, a simulação é invocada numa das seguintes formas:

```
agenda.bga <opt> <opt> ...
agenda.oga <opt> <opt> ...
agenda.lga <opt> <opt> ...
```

Os AG binários são executados por `agenda.bga` no qual `bga` representa *binary genetic algorithms*. Os AG combinatórios, por sua vez, são executados por `agenda.oga` no qual `oga` representa *order genetic algorithms*. Por fim, `agenda.lga` implementa a Programação Genética. Neste caso, `lga` representa *lists genetic algorithms*. Em qualquer dos casos, `<opt> <opt>...` representa a lista de parâmetros de controle.

Os parâmetros de controle, descritos seguidamente, foram divididos em quatro classes: os parâmetros relativos ao modelo, que definem qual o modelo de AG a utilizar, os parâmetros de controle global, que definem quais os ficheiros de controle e qual o problema a optimizar, os parâmetros de controle do algoritmo, que definem o conjunto de parâmetros operacionais do AG e, por fim, os parâmetros relacionados com o Modelo de Ilhas, que definem um conjunto de parâmetros de controle relacionados com este modelo.

B.2.2. Parâmetros Relativos ao Modelo

Opção	Argumentos	Descrição
-I	nlinks	Directiva para utilizar a plataforma com *nlinks* ligações
-T		Directiva para utilizar *threads*
-P	iconscode	Directiva para utilizar o simulador do Modelo de Ilhas, com a topologia definida no ficheiro *icons.iconscode*
-O	overlap	Directiva para utilizar o simulador do Modelo das Populações Sobrepostas com a percentagem de sobreposição igual a *overlap*
-C	mode	Directiva para utilizar o simulador de AG Celulares, com o parâmetro *mode* a definir o grau de vizinhança (0 para 8 vizinhos e 1 para 24 vizinhos)
-2		Directiva para utilizar o AG original (o AGA é usado por defeito)

Tabela B.1: Parâmetros relativos ao modelo

B.2.3. Parâmetros de Controle Global

Opção	Argumentos	Descrição
-Z	pcode	Escolha do problema a optimizar, indentificado por *pcode* (ver na próxima tabela a lista de problemas de teste pré-definidos)
-E		Directiva para só avaliar uma solução
-D		Directiva para criar uma imagem da população no final da simulação (no ficheiro *gpop*)
-l	logfile	Define o ficheiro *log* como *logfile*
-b	bestfile	Define o ficheiro *best* como *bestfile*
-f	seedfile	Define o ficheiro *seed* como *seedfile*

Tabela B.2: Parâmetros de controle global

B.2.3.1. Problemas de Teste Pré-definidos

Código	Descrição
0	Problema "tudo uns"
6	Função *F6*
7	Função *F7*
8	Função *F8*
9	Função de teste
20	Problema da Mochila 01 com 20 objectos
80	Problema da Mochila 01 com 80 objectos
69	Problema do Caixeiro Viajante com cromossomas binários e cruzamento normal
70	Problema do Caixeiro Viajante com cromossomas binários e cruzamento uniforme
71	Problema do Caixeiro Viajante com cromossomas numéricos e cruzamento "normal"
72	Problema do Caixeiro Viajante com cromossomas numéricos e cruzamento "uniforme"
90	Problema do Desvio de Frequência em redes eléctricas de distribuição de energia, com geradores éolicos
91	Problema do Colapso de Tensão em redes eléctricas de distribuição de energia

Tabela B.3: Problemas de teste pré-definidos

B.2.4. Parâmetros de Controle do Algoritmo

Opção	Argumentos	Descrição
-p	psize	Tamanho da população
-g	gen	Número de gerações
-c	pcross	Probabilidade de cruzamento (valores entre 0 e 1)
-m	pmut	Probabilidade de mutação (valores entre 0 e 1)
-e	pe	Directiva para utilizar elitismo

Tabela B.4: Parâmetros de controle do algoritmo

B.2.5. Parâmetros Relativos ao Modelo de Ilhas

Opção	Argumentos	Descrição
-n	nmig	Define a percentagem da população que emigra (valores em percentagem, entre 0 e 100)
-t	tmig	Define o número de gerações entre cada migração
-y	imigcode	Define a política de imigração, identificada pelo código *imigcod* (0 para selecção aleatória de emigrantes e 1 para selecção viciada de emigrantes)
-w	emigcode	Define a política de emigração, identificada pelo código *emigcod* (0 para selecção aleatória de substituíveis e 1 para selecção viciada de substituíveis)

Tabela B.5: Parâmetros relativos ao Modelo de Ilhas

B.3. FORMATO DOS FICHEIROS DE CONTROLE

Um conjunto de ficheiros de controle auxiliam o controle e recolha de dados do programa AGENDA. Os principais ficheiros são os ficheiros de resultados, *log* e *best*, o ficheiro de *seed* e o conjunto de ficheiros relacionados com a topologia do Modelo de Ilhas, *icons* (acrónimo de *Islands Connections*).

B.3.1. Ficheiros *log* , *best e seed*

Ao longo da execução, o programa AGENDA vai registando os resultados alcançados em dois ficheiros distintos. No ficheiro *log* (opção *-l*), regista a estatística do valor de adaptação da população, em função do número de geração, cujo formato de cada linha é o seguinte:

```
<nºgeração> <melhor> <média> <mediana> <desv.padrão> <int. interq.>
```

No ficheiro *best* (opção *-b)*, vão sendo registadas as melhores soluções alcançadas, de forma sequencial.

O ficheiro *seed* (opção *-f*), por sua vez, é utilizado para forçar o AG a iniciar a execução com um conjunto de indivíduos já definidos (guardados sequencialmente neste ficheiro). Se o número de indivíduos contidos em *seed* for inferior ao tamanho da população, são gerados os indivíduos necessários (de forma aleatória, como habitualmente).

B.3.2. Ficheiro Definição de Topologias do Modelo de Ilhas

Na simulação do modelo de Ilhas, é necessário definir qual a topologia de ligação entre as diversas ilhas. O ficheiro *icons* define uma estrutura própria para o efeito, na qual se indica, para cada ilha, quais as que têm ligações directas com ela. O seguinte exemplo ilustra a definição do Anel Bidireccional, para 8 ilhas:

```
0: 1 7
1: 2 0
2: 3 1
3: 4 2
4: 5 3
5: 6 4
6: 7 5
7: 0 6
```

Neste ficheiro, cada ilha com código de 0 a 7, está ligada à ilha seguinte e à ilha anterior, conforme se representa linha a linha.

Na invocação da simulação do Modelo de Ilhas, com o parâmetro *-P<iconscode>*, é definido qual a topologia a utilizar. O ficheiro de topologia consultado tem o nome *icons.<iconscode>*. Existe um conjunto de ficheiros de topologias pré-definidos para 8 ilhas, conforme ilustrado na tabela B.6

Ficheiro	**Topologia**
icons.0	Anel Simples
icons.1	Anel Bidireccional
icons.2	Anel Duplo
icons.3	Anel Duplo Bidireccional
icons.4	Completa
icons.5	Estrela
icons.6	Aleatória Ordem 3
icons.7	Hiper-Cubo
icons.8	Torus
icons.9	Árvore Bidireccional

Tabela B.6: Lista de ficheiros de topologias pré-definidas

As topologias semelhantes às anteriores, mas para 16 ilhas em vez de 8, são definidas nos ficheiros com nomes entre *icons.10* e *icons.19*.

PROBLEMA DE TESTE DA AGENDA

No sentido de testar e avaliar convenientemente a AGENDA, foi abordada a resolução de um problema real, da área da produção e distribuição de energia eléctrica, ao qual foi aplicado o AG distribuído com base nesta plataforma. Este anexo descreve o problema em pormenor, servindo de apoio ao capítulo 5.

C.1. DESCRIÇÃO DO PROBLEMA

C.1.1. Enquadramento

Devido a questões económicas e ambientais, os sistemas de energia eléctrica são levados, cada vez mais, perto dos limites operacionais. Assim, é necessário o desenvolvimento de sistemas de controle que avaliem o grau de segurança do sistema, em cada instante, podendo prever antecipadamente situações de colapso. No entanto, a tradicional avaliação deste grau de segurança, envolve a resolução de complexos sistemas de equações diferenciais não lineares, por resolução numérica, que implicam elevados tempos de computação.

A segurança dinâmica do sistema de energia é um importante factor no seu planeamento e operação. A situação ainda se torna mais crítica nos casos em que o parque de geradores inclui uma parte significativa de produção de energia por geradores éolicos. Nestes casos, as súbitas variações de vento podem originar quebras bruscas na produção de energia nestes geradores, levando a bruscos desvios da frequência eléctrica, que aumentam o perigo de colapso do sistema.

Por estas razões, em sistemas isolados, como no caso de pequenas ilhas, a produção de energia por geradores éolicosé mantida muito abaixo dos níveis máximos possíveis, para tentar minimizar o risco de colapso, à custa de um grande desperdício de energia

renovável. A utilização de políticas de exploração tão conservativas são incompatíveis com a necessidade do aumento da utilização deste tipo de energias.

O principal objectivo da investigação científica neste campo é conseguir a maximização da produção de energia por parte de geradores éolicos, assegurando simultaneamente que bruscas variações de produção por este método não afectem a segurança do sistema. Para tal, é necessário que seja possível prever, a curto prazo, a evolução do desvio de frequência do sistema, que reflecte directamente o grau de segurança. Com esta capacidade de previsão, é possível aumentar atempadamente a carga dos geradores convencionais, evitando desequilíbrios do sistema, sem no entanto comprometer significativamente o aproveitamento da energia eólica. O sistema funciona assim muito mais perto do limite, optimizando a produção global.

Todo este processo implica a capacidade de avaliação da estabilidade e segurança do sistema em poucos instantes. O convencional processo de análise, dada a sua complexidade e volume de cálculo, é incompatível com este controle dinâmico.

Assim, esta necessidade de avaliação *on-line* da estabilidade, levou à aplicação de diversas técnicas de Inteligência Artificial, que se baseiam em métodos de aprendizagem por exemplos [Lopes94]. Nestas, o comportamento do sistema é inferido pela análise de um conjunto de pontos operacionais previamente analisados, para os quais de conhecem todos os parâmetros relevantes e o respectivo grau de estabilidade.

As abordagens utilizadas incluem técnicas de Reconhecimento de Padrões, Redes Neuronais e métodos de inferência indutiva aplicados na criação de Árvores de Decisão [Lopes94]. Em todas as abordagens, o sistema de controle abstrai o modelo de funcionamento do sistema pela análise de um conjunto de situações anteriores, conseguindo efectuar previsões para novas situações em curtos espaços de tempo e com significativo sucesso. Desta forma, o complexo volume de cálculo necessário para a análise de estabilidade, é substituído por um método eficiente de avaliação, que "aprendeu" o funcionamento do sistema, por intermédio de exemplos. A capacidade de generalização destes métodos permite a sua aplicação a novas situações.

C.1.2. A Ilha de Lemnos

No âmbito do programa europeu de investigação científica, JOULE-II, foi desenvolvido um sistema de controle avançado para sistemas de produção e distribuição de energia de pequena e média dimensão, aplicado a uma pequena ilha grega, conhecida pela ilha de Lemnos, que contém uma significativa componente de energia éolica [Lopes94a], conforme de pode ver no diagrama da figura C.1.

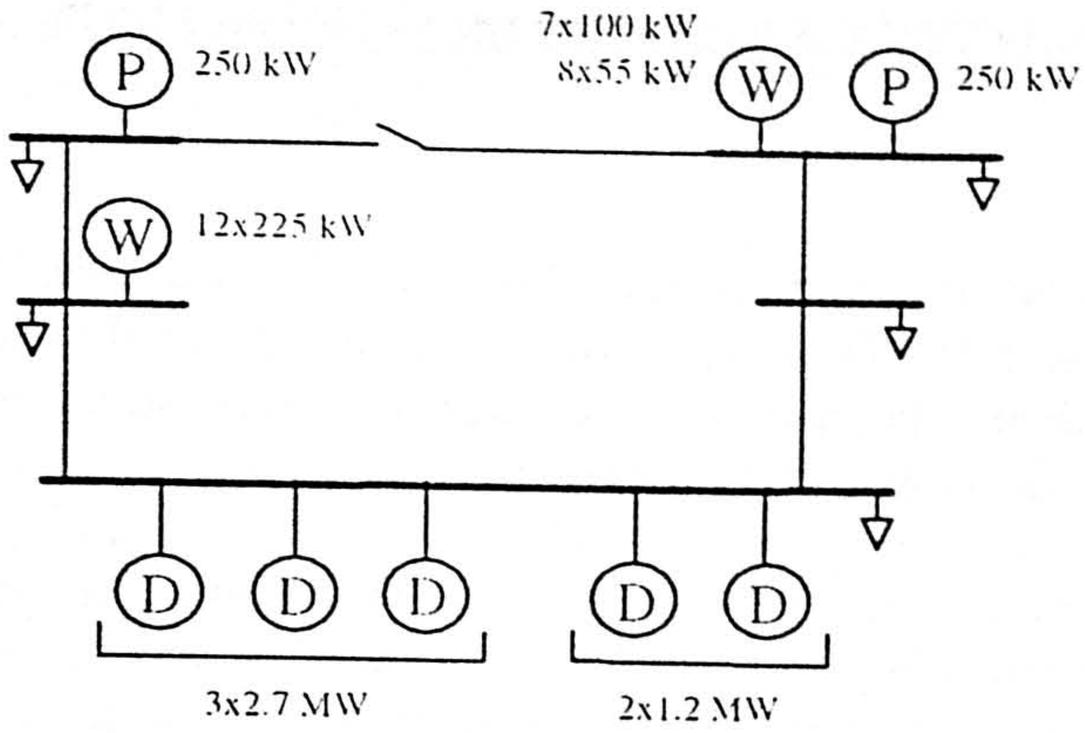


Figura C.1: Diagrama do Sistema Eléctrico da ilha de Lemnos

Neste diagrama, existem dois parques de geradores convencionais de combustível fóssil (designados por P) e dois parques de geradores eólicos (designados por W), que alimentam um conjunto de cargas (designadas por D):

O sistema de controle foi refinado pela aplicação de técnicas de aprendizagem por exemplos [Lopes94]. Estas técnicas têm por base um conjunto de 450 pontos de operação do sistema, previamente recolhidos, que traduzem 450 situações ou estados possíveis, para os quais se calculou *off-line* o grau de estabilidade. Este conjunto inicial foi dividido em duas partes: o conjunto de treino, com 300 pontos, a partir do qual o comportamento do sistema é abstraído pelos já referidos métodos, e o conjunto de teste, com 150 pontos, que é usado para aferir a qualidade das soluções encontradas, quando aplicadas a situações nunca "vistas" anteriormente.

Os pontos de operação deste sistema são definidos por um conjunto de parâmetros, que traduzem o valor de 58 grandezas relevantes, que reflectem, directa ou indirectamente, o estado global do sistema. Estas grandezas representam quantidades eléctricas e ambientais, nomeadamente, entre as quais se encontram a potencia activa e reactiva dos geradores de combustível fóssil de ambos os parques, e as previsões a curto prazo da variação do vento [Lopes94a].

Os métodos de aprendizagem por exemplos utilizados derivam estruturas de classificação do sistema, capazes de indicar a estabilidade ou não estabilidade deste, com base nos valores dos referidos parâmetros. Os resultados atingidos pelo método das Redes Neuronais e Árvores de Decisão, tendo em conta a utilização dos 58 parâmetros, aproximaram-se de 95% de acertos no conjunto de teste, ou seja, a classificação da segurança do sistema (estável ou não estável) para todas as 150 diferentes situações do conjunto de teste, esteve correcta 95% das vezes.

C.2. APLICAÇÃO DO ALGORITMO GENÉTICO

A presente secção analisa as questões relacionadas com a aplicação do AG ao anterior problema. A resolução por AG do problema de análise de estabilidade de um sistema de produção e distribuição de energia é o tema de um outro trabalho, que teve a colaboração do autor desta tese [Lopes95].

Como sempre, a aplicação do AG a um problema envolve duas questões principais: a codificação do problema em cromossomas e a função de avaliação das soluções. Neste caso, como se pretende encontrar uma estrutura de classificação, que com base nos valores de um conjunto de parâmetros consiga prever a estabilidade de um sistema eléctrico de produção e distribuição de energia, não é possível a utilização de cromossomas binários nem numéricos.

C.2.1. Codificação

Uma possível forma de representar uma estrutura de classificação é codificá-la sob a forma de um programa, numa linguagem com características adequadas. A linguagem escolhida para este efeito foi o LISP, não só pela sua flexibilidade e simplicidade, mas também pela sua directa adaptação a uma representação interna sob a forma de árvore de expressões.

A utilização do AG com árvores de expressões constitui a base de um conhecido paradigma, conhecido por Programação Genética (PG), já apresentada no segundo capítulo. Neste paradigma, os cromossomas não são sequências binárias, mas sim estruturas em memória representando árvores de expressões. As expressões utilizadas são constituídas por nós, que podem ser funções com subexpressões, ou nós terminais com variáveis ou constantes, conforme se pode ver no exemplo da figura C.2.

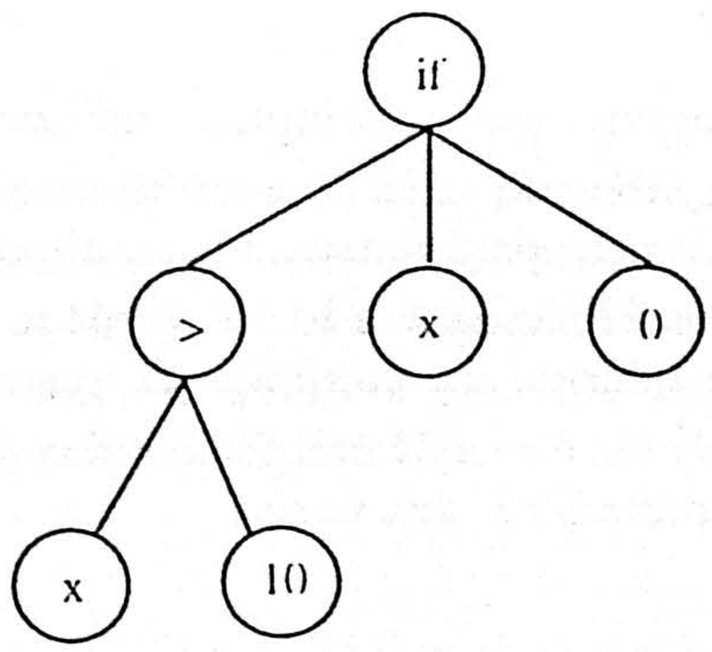


Figura C.2: Representação de uma expressão numa árvore

A expressão representada na árvore da figura C.2, pode ser apresentada numa forma mais compacta através do seu programa equivalente em LISP:

```
if ( (> x 10) x 0)
```

Todos os programas deste género retornam um valor, resultado da avaliação da expressão que representam. Neste pequeno exemplo, o valor da variável *x* é comparado com a constante 10. No caso de *x* ser superior a 10, a expressão retorna o valor de *x*, caso contrário retorna 0.

Na codificação das estruturas de classificação em programas de LISP, são utilizadas para o presente problema algumas funções lógicas e matemáticas elementares. Essas funções são combinadas com o conjunto de todos os parâmetros operacionais do sistema e com algumas constantes genéricas. As funções utilizadas são:

- Função de decisão: `if`
- Funções lógicas: `and, or, not`
- Funções lógicas de comparação: `>, <, ==, >=, <=`
- Funções trigonométricas, logaritmo e exponencial: `sin, cos, log, exp`
- Funções aritméticas simples: `+, -, *, /`

Os 58 parâmetros do sistema são representados por um conjunto de variáveis, indexadas desde x_1 até x_{58}. Os programas ou expressões assim construídos combinam livremente as funções apresentadas com estas variáveis e com valores constantes gerados ao acaso. A geração dos programas é conseguida mediante um algoritmo que constrói aleatoriamente árvores de expressões, com uma determinada profundidade média, previamente definida.

C.2.2. Avaliação

O AG adaptado a este problema inicia o seu ciclo com a geração aleatória de uma população de programas ou expressões, representados sob a forma de árvores de expressões. A avaliação de cada um destes indivíduos é feita "executando" o programa por ele representado, em cada um dos pontos de operação do conjunto de treino, comparando o resultado obtido com o resultado esperado.

Esta "execução" do programa representado pelo indivíduo, não é mais do que a substituição das variáveis nele contidas pelos correspondentes valores do ponto de operação em questão, seguida da sua avaliação. A avaliação, conforme referido, retorna um valor real, que é transformado numa quantidade binária que representa a estabilidade ou não estabilidade do sistema. Esta transformação é feita por simples comparação do valor real retornado com um valor limite arbitrado: se este for maior do que aquele, o resultado binário é verdadeiro, caso contrário é falso O valor desta

constante é irrelevante, devido à forma como o AG evolui as soluções (o algoritmo adapta as soluções ao valor escolhido para a constante).

Para cada elemento do conjunto de treino, a previsão de estabilidade dada pelo programa em avaliação é comparada com o verdadeiro estado do sistema na respectiva situação (que foi previamente calculado para os pontos de operação deste conjunto). No caso de o resultado do programa coincidir com o valor correcto (estável ou não estável), o programa sucede na previsão. Caso contrário, o programa falha na previsão. O valor de um indivíduo não é mais do que a percentagem de acertos, e a função de avaliação limita-se a "executar" o programa candidato em todas as situações, ou pontos de operação, do conjunto de treino, atribuindo-lhe um valor de adaptação igual à sua percentagem de acertos.

C.2.3. Operadores

Após a geração do conjunto inicial de indivíduos, e respectiva avaliação, o AG inicia o seu ciclo principal, no qual manipula a informação genética na esperança de encontrar melhores soluções, utilizando os operadores genéticos de cruzamento e mutação.

Estes operadores foram completamente redefinidos, para se adaptarem à representação em árvore de expressões. No actual contexto, o cruzamento deve ser capaz de recombinar duas estruturas em árvore, o que é feito escolhendo dois pontos arbitrários de corte, seguido da fusão cruzada das quatro partes resultantes.

A mutação, por sua vez, deve ser capaz de alterar uma determinada estrutura em árvore. Para uma análise mais detalhada destes operadores deve ser consultada a bibliografia [Koza91].

C.2.4. Qualidade das Soluções Obtidas

No final da execução do AG, o melhor indivíduo é aplicado ao conjunto de teste, o qual contém pontos de operação nunca utilizados na evolução das soluções. Verificou-se que as soluções obtidas por este método têm uma grande capacidade de generalização, ou seja, obtêm resultados satisfatórios nas novas situações (tão bons como nas situações de treino).

Os resultados obtidos, quer pela abordagem pelo AG original, quer pela abordagem da AGENDA, foram muito encorajadores. A qualidade das soluções finais atingiu 98.3% de acertos, o que ultrapassa os resultados atingidos em situações análogas por outros métodos de optimização [Lopes95].

[Angeline93] Evolutionary Algorithms and Emergent Intelligence
Peter Angeline
PhD Dissertation, The Ohio State University, 1993

[Back91] Optimization by Means of Genetic Algorithms
Thomas Back
University of Dortmund, 1991

[Back92] Extended Selection Mechanisms in Genetic Algorithms
Thomas Back, Frank Hoffmeister
University of Dortmund, 1992

[Back92a] Optimal Mutation Rates in Genetic Search
Thomas Back, Frank Hoffmeister
University of Dortmund, 1992

[Baluja92] A Massively Distributed Parallel Genetic Algorithm
Shumeet Baluja
School of Computer Science, Carnegie Mellon University, 1992

[Belew90] Evolving Networks: Using Genetic Algorithms with Connectionism Learning
Richard K. Belew, John McInerney, Nicol N. Schraudolph
University California, San Diego, 1990

[Beguelin91] A user's guide to PVM: Parallel Virtual Machine
A. Beguelin, J. Dongarra, A. Geist, R. Mancheck, K. Moore, R. Wade
Technical Report, ORNL/TM-11826, Mathematical Sciences Section, Oak Ridge National Laboratory, September 1991

[Bianchini93] Parallel Genetic Algorithms on Distributed-Memory Architectures
Ricardo Bianchini, Christopher Brown
The University of Rochester, Computer Science Department, 1993

[Davis91] Handbook of Genetic Algorithms
Lawrence Davis *et al.*
Van Nostrand Reinhold, 1991

[Davis91a] A Simulated Annealing Like Convergence Theory for the Simple Genetic Algorithm
T. E. Davis
Procedings of the Fourth International Conference of Genetic Algorithms, Morgan Kaufmann Publishers, 1991

[DeJong91] An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms
Kenneth De Jong, William M. Spears
Computer Science Department, George Mason University, 1991

[Douglas93] Parallel Programming Systems for Workstation Clusters
Craig C. Douglas, Timothy G. Mattson, Martin H. Schultz
Yale University, Computer Science Department, 1993

[Flynn72] Some Computer Organizations and Their Effectiveness
M. J. Flynn
IEEE Trans. Computers, 21(9):948-960, 1972

[Gelernter85] Generative Communications in Linda
David Gelernter
ACM Transactions on Programming Languages and Systems, 7(1):80-112, January 1985

[Gelernter88] Getting the Job Done
David Gelernter
BYTE, November 1988

[Goldberg89] Genetic Algorithms in Search, Optimization and Machine Learning
David E. Goldberg
Addison Wesley, 1989

[Goldberg91] Sizing Populations for Serial and Parallel Genetic Algorithms
David E. Goldberg
Genetic Algorithms in Search, Optimization and Machine Learning
Summer School, Stanford, 1991

[Goldberg91a] A Comparative Analysis of Selection Schemes Used in Genetic Algorithms
David E. Goldberg
Genetic Algorithms in Search, Optimization and Machine Learning
Summer School, Stanford, 1991

[Goldberg95] Critical Deme Size for Serial and Parallel Genetic Algorithms
David E. Goldberg, Hillol Kargupta, Jeffrey Horn, Erick Cantu-Paz
Illigal report No. 95002, January 1995

[Gordon93] Serial and Parallel Genetic Algorithms as Function Optimizers
V. Scott Gordon, Darrel Whitley
Colorado State University, 1993

[Holland75] Adaptation in Natural and Artificial Systems
J. H. Holland
The University of Michigan Press, 1975

[Hwang93] Advanced Computer Architecture
Kay Hwang
McGraw-Hill, International Editions, Computer Science Series, 1993

[Khuri93] The Zero/One Multiple Knapsack Problem and Genetic Algorithms
Sami Khuri, Thomas Back, John Heitkotter
San José State University, 1993

[Khuri94] An Evolutionary Approach to Combinatorial Optimization Problems
Sami Khuri, Thomas Back, John Heitkotter
ACM Press, 1994

[Koza91] The Genetic Programming Paradigm: Gennetically Breeding Populations of Computer Programs to Solve Problems
John R. Koza
Genetic Algorithms in Search, Optimization and Machine Learning,
Summer School, Stanford, 1991

[Liening94] A Genetic Algorithm for Channel Routing in VLSI Circuits
Jens Liening, K. Thulasiraman
Concordia University, Montreal, 1994

[Lopes94] Application of Learning from Exemples Methods for On-line Dynamic Security Assessement of Electric Power Systems - State of the Art
J.A. Peças Lopes, N. Hatziarggyriou
Proc. IV SEROPE, Foz do Iguaçu, Brazil, Maio 1994

[Lopes94a] Fast Dynamic Security Assessment of Autonomous Power Systems With a Large Penetration From Wind - The Lemmos Study Case
J.A. Peças Lopes, J.N. Fidalgo, V. Van Acker
INESC and University of Porto, 1994

[Lopes95] Derivation of Classification Structures for Fast Evaluation of Dynamic Security Assessment in Power Systems using Genetic Algorithms
J. Peças Lopes, J. Ranito, J. Neto
INESC and University of Porto, 1995

[Mahfoud93] Finite Markov Chain Models of an Alternative Selection Strategy for the Genetic Algorithm
Samir W. Mahfoud
University of Illinois, 1993

[Mitchell92] When Will a Genetic Algorithm Outperform Hill-Climbing
Melanie Mitchell, John H. Holland
Santa Fe Institute, 1992

[Mullender92] Distributed Systems
Sape Mullender
Addison Wesley, 1992

[Orlib94] OR-Library: Distributing Tests Problems by electronic mail
o.rlibrary@ic.ac.uk

[Penfold91] Deriving Application-Specific Neural Nets Using a Massively Parallel Genetic Algorithm
H. B. Penfold, U. Kohlmorgen, H. Schmeck
The University of Newcastle, Australia, 1991

[Pérez91] Algoritmo para Evaluation de Reglas en un Sistema de Control con Aprendizage
Juan Ramon Velasco Pérez
Asóciacion Espanola para la Inteligencia Artificial - AEPIA 1991

[Ranito92] Algoritmos Genéticos na Optimização de Perguntas Relacionais
João V. Ranito
Faculdade de Engenharia da Universidade do Porto, Departamento de Engenharia Electrotécnica e de Computadores, 1992

[Schroeder92] A State-of-the-Art Distributed System: Computing with BOB
Michael D. Schroeder
Distributed Systems, Sape Mullender, Addison Wesley, 1992

[Sharp95] The Grand Challanges
Oliver Sharp
Byte Magazine, February 1995

[Smith93] Adaptively Resizing Populations: Algorithm, Analysis, and First Results
Robert E. Smith, Ellen Smuda
The University of Alabama, 1993

[Spears91] A Study of Crossover Operators in Genetic Programming
William M. Spears, Vic Anand
Naval Research Laboratory, Washington,1991

[Srinivas94] Genetic Algorithms: A Survey
M. Srinivas, Lalit M. Patnaik
IEEE Computer, 06/1994

[Stevens90] UNIX Network Programming
W. Richard Stevens
Prentice Hall, 1990

[Sun90] Network Programming Guide
Sun Microsystems, 1990

[Talbi91] A Parallel Genetic Algorithm for the Graph Partition Problem
G. Talbi, P. Bessière
Laboratoire of Génie Informatique, University of Grenoble, 1991

[Tanese90] Distributed Genetic Algorithms
Reiko Tanese
Department of Electrical Engineering and Computer Science, University of Michigan, 1990

[Tanenbaum90] Distributed Operating Systems
A. S. Tanenbaum
Prentice Hal, 1990

[Velasco91] Modeling and Control of Water Resources Systems and Global Changes
Juan R. Velasco
The Society for Computer Simulation International, 1991

[Whitley92] A Genetic Algorithm Tutorial
Darrel Whitley
Computer Science Department, Colorado State University, 1992

[Yin91] Investigations on Solving The Load Flow Problem by Genetic Algorithms
Xiadong Yin, Noël Germay
Catholic University of Louvain, Belgium, 1991

[Zeigler91] Characteristics of an Asynchronous Parallel Genetic Algorithm
Bernard P. Zeigler, Jinwoo Kim
AI and Simulation Group, Department of Electrical and Computer Eng.
University of Arizona, Tucson, 1991

BIBLIOTECA F.E. PORTO

FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO
BIBLIOTECA
0000002923