6

TÉCNICAS ALGORÍTMICAS BASEADAS NA EVOLUÇÃO NATURAL E A TECNOLOGIA DAS RESTRIÇÕES

A evolução natural é responsável pelo aparecimento de todos os organismos vivos no nosso planeta tais como os conhecemos actualmente. É um processo muito lento que ocorre há milhões de anos e que continua neste preciso momento. É o processo de adaptação de todas as espécies de organismos vivos que lhes permite sobreviver num ambiente sujeito a mudanças regulares. Por vezes algumas espécies desaparecem por falta de adaptação a esse meio ambiente. Acerca de algumas dezenas de anos atrás surgiu o interesse pelo estudo da possibilidade de simular computacionalmente o processo da evolução natural e de usar estas simulações como ferramentas de optimização.

Neste capítulo pretende-se dar uma visão global das técnicas algorítmicas baseadas na evolução natural, com especial destaque para os algoritmos genéticos

quando aplicados à optimização de problemas. É dada uma atenção especial, atendendo à natureza do trabalho desenvolvido, ao uso dos algoritmos genéticos em colaboração com o paradigma da programação lógica por restrições. Finalmente, é apresentada uma metodologia desenvolvida que combina o paradigma da programação lógica por restrições com os algoritmos genéticos, para a resolução de problemas de optimização.

6.1 Algoritmos Genéticos: Técnica Baseada em Princípios Biológicos

Os algoritmos genéticos (AG) constituem um conjunto de métodos adaptativos que podem ser usados na resolução de problemas de pesquisa e optimização. Estes métodos foram inspirados pelas descobertas efectuadas pelos biólogos evolucionários e pretendem simular os processos que são essenciais à evolução natural de populações que, em termos biológicos, são constituídas por organismos vivos. Ao longo de muitas gerações, populações de organismos vivos evoluem de acordo com os princípios da selecção natural, que foram pela primeira vez descritos por Darwin[1].

Na natureza, os indivíduos de uma população competem entre si na obtenção de recursos tais como alimentação, água e abrigo. Os membros da mesma espécie também competem frequentemente para atrair um companheiro para reprodução. Os indivíduos com maior capacidade de sobrevivência e de atracção de companheiros para reprodução são os que terão mais probabilidades de virem a ter um maior número de descendentes. Os menos capazes terão um menor número de descendentes. Desta forma, os genes dos indivíduos mais bem adaptados ao meio ambiente serão espalhados por um crescente número de indivíduos a cada sucessiva geração. A combinação de boas características de diferentes antepassados pode em alguns casos produzir descendentes bem adaptados ao meio ambiente, cujo grau de

[1] Charles Darwin (1809-1882) escreveu a obra “Sobre a Origem das Espécies por Meio da Selecção Natural” publicada em 1859.

adaptação é potencialmente melhor que o dos progenitores. Com este processo e ao longo de sucessivas gerações, uma dada espécie evolui de forma a que cada vez mais indivíduos estarão melhor adaptados ao seu ambiente.

Os AG usam uma analogia directa com o processo de evolução natural (ver Tabela 6-1). Estes trabalham com uma população de soluções (indivíduos) do problema considerado. A cada indivíduo é atribuído um valor de aptidão[2] que corresponde à sua capacidade de sobrevivência e reprodução. Os indivíduos com os maiores valores de aptidão possuem maiores oportunidades de se combinarem com outros indivíduos da população. Na reprodução são gerados novos indivíduos que herdam algumas características de cada um dos progenitores. A combinação de boas características dos progenitores permite a geração de indivíduos (soluções) melhores.

Tabela 6-1: Analogia entre a evolução natural e os algoritmos genéticos.

Analogia com a Natureza		
Evolução Natural	Û	**Algoritmos Genéticos**
Indivíduo	—	Solução
Genótipo (cromossomas)	—	Representação
Reprodução Sexual	—	Operador Cruzamento
Mutação	—	Operador Mutação
População	—	Conjunto de Soluções
Gerações	—	Ciclos

6.2 Paradigmas da Computação Evolucionária

A *computação evolucionária* (CE) refere-se ao estudo dos fundamentos e das aplicações de determinadas técnicas heurísticas baseadas nos princípios da evolução natural. Estas técnicas são usualmente agrupadas em três categorias principais indicadas na Figura 6-1. Esta classificação tem origem essencialmente em factores históricos relacionados com o aparecimento de diferentes linhas de investigação.

Os algoritmos evolucionários (AE) são estocásticos e iterativos, não garantindo

[2] Na língua inglesa o valor de aptidão é conhecido por *fitness*.

a convergência para a melhor solução. A conclusão do processo iterativo pode ser obtida atingindo um número máximo de gerações pré-definido ou obtendo uma solução aceitável. Estes operam sobre um conjunto de indivíduos que usualmente se designa por população. Cada indivíduo representa uma potencial solução para o problema considerado. Esta solução é obtida por meio de um mecanismo de codificação e descodificação. Inicialmente, a população é gerada aleatoriamente e a cada indivíduo na população é atribuído um valor, através de uma função de aptidão. Este valor, conhecido por valor de aptidão, é uma medida da sua qualidade relativamente ao problema considerado e é usado para orientar a pesquisa.

CE	**=**	**AG**	**+**	**EE**	**+**	**PE**
Computação Evolucionária		*Algoritmos Genéticos* (Holland, 1975)		*Estratégias de Evolução* (Rechenberg, 1973)		*Programação Evolucionária* (Fogel *et al*, 1966)

Figura 6-1: Os três paradigmas da computação evolucionária.

Do esqueleto do AE indicado na Figura 6-2 observa-se que o algoritmo compreende três etapas principais. São elas a selecção, a reprodução e a substituição. Durante a etapa de selecção, é criada uma população provisória com alguns dos indivíduos da população principal. Os indivíduos mais aptos conseguem estar presentes com maior frequência na população provisória do que os menos aptos, tal como acontece no mecanismo da selecção natural. Aos indivíduos da população provisória são aplicados os operadores reprodutivos de forma a gerar uma nova população. Finalmente, os indivíduos da população principal são substituídos pelos novos indivíduos. Geralmente, esta substituição tenta manter os melhores indivíduos e remove os menos aptos. O processo inteiro é repetido até que uma determinado condição de finalização se torne verdadeira.

Os princípios básicos dos AG foram estabelecidos por Holland (1975). Actualmente, o algoritmo original proposto por Holland é designado por AG canónico ou AG simples (Goldberg, 1989). Estes algoritmos serão descritos com mais detalhe na secção seguinte.

A *Programação Evolucionária* (PE), originalmente introduzida por Fogel (1966), apresenta algumas características comuns com os AG. A diferença fundamental, em termos da sua funcionalidade, relaciona-se com a forma como os descendentes são

gerados. Não existe qualquer preocupação em imitar os operadores naturais, sendo o operador de mutação o único responsável pela geração de descendentes. A PE simula a evolução como um processo baseado nas soluções, que enfatiza o comportamento na ligação entre os progenitores e os descendentes em vez da sua ligação genética, tal como se verifica nos AG, pelo que a PE apresenta menos restrições relativamente à representação das soluções. Como na PE não existe um operador de recombinação, ao contrário do que acontece com os AG, é possível usar um qualquer tipo de representação desde que se defina um operador de mutação adequado.

```
Gerar P
t ← 0
enquanto ¬ Condição de final P_t faz
    Avalia P_t
    P'_t    ← Selecciona P_t
    P''_t   ← Aplica operadores de reprodução P'_t
    P_t+1   ← Substitui P_t, P''_t
    t ← t + 1
fim
retorna Melhor_Solução
```

Figura 6-2: Esqueleto de um algoritmo evolucionário.

Relativamente às Estratégias de Evolução (EE), estas foram originalmente introduzidas na Alemanha por Rechenberg (1973), tendo sido alvo de posteriores desenvolvimentos por Schwefel (1981), foram inicialmente usadas para optimização de parâmetros contínuos. A sua primeira versão considera apenas dois indivíduos, um progenitor e um descendente. Estes utilizam usam uma representação decimal para a sua codificação. As EE consideram, ainda, um operador de mutação e um mecanismo de selecção. Este tipo de EE é designado por EE-(1+1) na actual noção de EE (Bäck *et al*, 1991).

Mais tarde surgiram generalizações destas técnicas. A primeira passa pela sua aplicação a mais do que um indivíduo, nas chamadas EE-(m+1). Neste caso, consideram-se m indivíduos progenitores que estão potencialmente envolvidos na geração de um descendente, o que permite a imitação de comportamentos sexuados e a utilização de operadores de recombinação. Posteriormente, foram desenvolvidas outras estratégias evolutivas designadas de EE-($m+l$) e EE-(m,l). Em ambos os casos são gerados l descendentes a partir de m progenitores, enquanto que os

símbolos '+' e ',' definem o método de selecção. O primeiro método determina que os indivíduos da população na geração seguinte são escolhidos de um universo que contém os m progenitores e os l descendentes. No outro método os indivíduos para a geração seguinte são seleccionados do universo dos l descendentes, o que significa que nenhum indivíduo sobrevive para além de uma geração. As EE actuais incluem esquemas de recombinação de vários progenitores e de auto-adaptação pela inclusão dos parâmetros da EE na codificação dos indivíduos (Bäck e Schwefel, 1993). Uma referência para as EE é a obra escrita por Schwefel (1995).

6.3 Algoritmos Genéticos

Os AG constituem, de longe, o grupo mais extenso dos métodos representativos da aplicação das ferramentas de CE. Estes trabalham com uma população de indivíduos, que representam as soluções para o problema, usando operadores de recombinação, mutação, selecção e substituição. Um AG cria gerações sucessivas de indivíduos cada vez mais aptos. A busca é guiada apenas pelo valor de aptidão associado a cada indivíduo na população. Quando se planeia a resolução de um dado problema utilizando AG é usualmente necessário ter em linha de conta a codificação das soluções e a função de avaliação, sendo estas dependentes do problema. Os princípios básicos dos AG foram estabelecidos com rigor pela primeira vez por Holland (1975), tendo ainda introduzido as fundações teóricas, capazes de justificar que estes realizam de forma efectiva processos de pesquisa e optimização de soluções. Esta secção destina-se a apresentar uma breve introdução aos aspecto práticos de implementação dos AG, embora uma descrição mais completa, incluindo os fundamentos teóricos, possa ser encontrada em muitos outros textos, como, por exemplo, os de Goldberg (1989), Davis (1991) ou Michalewicz (1996).

6.3.1 Codificação

A codificação passa por se encontrar uma representação para as possíveis soluções do problema, ao nível do material genético de cada indivíduo. Estas soluções podem ser representadas por um conjunto de parâmetros. Estes são

agrupados para formar uma cadeia de valores, frequentemente referidos como cromossomas. Cada parâmetro codificado num cromossoma é designado por *gene*. Holland mostrou que idealmente se deve usar um alfabeto binário para codificar os valores de cada parâmetro, embora, actualmente, esta opinião não seja partilhada com unanimidade. Por exemplo, se o objectivo do problema for o de maximizar uma função de três variáveis, *f(x, y, z)*, a representação de cada uma das variáveis pode ser efectuada usando valores binários de 10 *bits* com uma escala adequada. O cromossoma teria assim três genes e um total de 30 dígitos binários.

Fazendo o paralelo com a ciência genética, ao conjunto de parâmetros representados nos cromossomas é dado o nome de *genótipo*, que constitui o material genético do indivíduo correspondente. O genótipo possui a informação necessária para construir um organismo, o qual é referido como *fenótipo*. Um fenótipo caracteriza os atributos de um indivíduo, conhecidos que são os valores dos seus genes e o ambiente que o rodeia. Os mesmos termos são usados pelos AG. Por exemplo, numa tarefa de projecto de uma ponte, o conjunto de parâmetros que especificam um projecto em particular constitui um genótipo, enquanto a construção final é o fenótipo. O valor de aptidão de um indivíduo depende do desempenho do fenótipo. Este valor pode ser inferido do genótipo, ou seja, pode ser calculado a partir do cromossoma, usando uma função de avaliação. Os cromossomas estão organizados por sequências lineares de genes. A cada um dos valores possíveis de um dado gene é dado o nome de *alelo*. O conjunto de alelos para um dado gene define o domínio dos valores possíveis para o correspondente parâmetro.

A representação que tem vindo a ser usada com maior frequência, para os valores admissíveis para cada parâmetro, é a codificação binária. No entanto, e como já se referiu, verifica-se uma crescente falta de unanimidade quanto a esta representação ser a mais adequada e natural. Um exemplo óbvio, quanto à sua falta de adequação, relaciona-se com a resolução de problemas que necessitam de valores com uma elevada precisão. Estes problemas requerem a utilização de um elevado número de *bits* o que requer um maior esforço computacional para explorar um vasto espaço de potenciais soluções. Uma representação mais natural para estes problemas passa pela utilização de números reais para a representar as soluções. Este tipo de representações é usado em trabalhos apresentados por Mühlenbein *et al* (1991) e

Michalewicz (1996).

Algumas instâncias dos problemas de optimização combinatória, constituem também problemas que apresentam dificuldades ao nível da representação, quando se tenta utilizar os AG para os solucionar. O problema do caixeiro viajante e o problema do circuito *hamiltoniano* são dois exemplos. As representações mais adequadas para estes problemas são as representações genéticas baseadas em permutações ou na ordem. Nestas representações, cada cromossoma, que constitui o genótipo de um indivíduo, é definido como uma permutação de todos os símbolos do alfabeto, ou seja, uma sequência ordenada de todos os símbolos, não sendo permitidas repetições de símbolos. Verifica-se que o comprimento dos cromossomas é igual ao número de símbolos do alfabeto e que este alfabeto define o conjunto de alelos para cada um dos genes.

6.3.2 Medida da Aptidão

O valor de aptidão de cada indivíduo (ou solução) é calculado através duma função de avaliação que considera a representação escolhida. Esta função, que efectua a avaliação de modo a atribuir um valor de aptidão às potenciais soluções através duma função de aptidão, precisa de ser definida para cada problema que se pretende solucionar. Dado um cromossoma, a função de avaliação retorna um valor numérico que espelha o seu mérito para o problema alvo, dada a interpretação fenotípica do genótipo do indivíduo. Quando os problemas que se pretendem solucionar são casos que têm origem no mundo real, a função de avaliação não é fácil de escolher, dado que os factores a considerar são, por norma, numerosos e difíceis de avaliar com rigor.

As noções de avaliação e de aptidão são por vezes intercambiáveis. No entanto, deve-se distinguir a noção de função de avaliação e função de aptidão usadas nos AG. A primeira proporciona uma medida do desempenho em relação a um conjunto particular de parâmetros, sendo que, a avaliação de um dado indivíduo é independente da avaliação dos outros indivíduos. A segunda transforma essa medida de desempenho na atribuição de oportunidades de reprodução, sendo, portanto, definida em relação aos outros indivíduos da população.

A definição da função de aptidão apropriada é muito importante para a execução correcta de um AG. Esta função representa o ambiente do problema. Refira-se que no caso ideal, a função de aptidão deve ser definida de modo a ser suave e regular no sentido em que os cromossomas que partilham uma parte importante do material genético tenham valores de aptidão próximos. Adicionalmente, deve-se procurar que esta função contenha um número mínimo de máximos locais e ao mesmo tempo o máximo global não seja um ponto em que os vizinhos possuam valores de aptidão muito menores. Quando o cálculo do valor de aptidão é demasiado complexo e pesado, é frequente a utilização de funções que dão um valor de aptidão aproximado aceitável e cujo cálculo seja mais simples.

6.3.3 Operadores Genéticos

Como foi anteriormente referido, os AG são processos iterativos e estocásticos que, durante uma dada iteração, usualmente designada por *geração*, mantêm constante uma população de soluções potenciais, representadas pelos seus cromossomas. Cada solução é avaliada para fornecer uma medida da sua *aptidão*. De seguida, uma nova população é produzida, pela selecção de indivíduos de acordo com uma probabilidade de selecção proporcional à sua aptidão. Alguns membros da nova população sofrem alterações através das operações de recombinação e mutação. Um esqueleto para um AG típico é apresentado na Figura 6-3.

```
Gerar P0
t ← 0
Avalia Pt
enquanto ¬ Condição de final Pt faz
    P't    ← Selecciona Pt
    P''t   ← Aplica operadores de recombinação P't
    P'''t  ← Aplica operadores de mutação P''t
    Avalia P'''t
    Pt+1   ← Selecciona Sobreviventes Pt, P'''t e Substitui
    t ← t + 1
fim
retorna Melhor_Solução
```

Figura 6-3: Esqueleto típico de um algoritmo genético.

Note-se as semelhanças relativamente ao esqueleto dos AE apresentado na

Figura 6-2. A primeira geração dos AG, e uma boa parte dos actuais, utiliza como estratégia de renovação a substituição em cada geração de toda a população pelos novos indivíduos gerados. No entanto, actualmente, é frequente serem utilizadas outras estratégias diferentes que serão apresentadas adiante.

Recombinação

O principal operador dos AG é o operador de recombinação. Dois indivíduos progenitores são escolhidos da população actual, por um método de selecção bem definido, para produzir dois novos indivíduos, denominados descendentes. Quando os cromossomas são apenas cadeias, o operador de recombinação produz os dois descendentes ao escolher um ou mais pontos de corte nos cromossomas dos progenitores e depois cria uma combinação diferente das partes resultantes para gerar cada um dos cromossomas dos descendentes. Neste caso particular, este operador é usualmente designado por operador de cruzamento. Tal como na recombinação biológica, um descendente herda genes de ambos os progenitores.

Alguns dos operadores de cruzamento mais frequentemente usados, quando se opta por uma representação dos cromossomas que utilizam cadeias de dimensão fixa, são os seguintes:

- **cruzamento de ponto único** – faz a selecção aleatória de uma posição de corte criando quatro sequências que aparecerão cruzadas nos descendentes. Estes recebem uma sequência de cada um dos progenitores (ver Figura 6-4);
- **cruzamento de dois pontos** – selecciona aleatoriamente duas posições de corte, criando seis sequências que aparecerão cruzadas nos descendentes. Neste caso, os descendentes recebem uma sequência de um dos progenitores e duas do outro (ver Figura 6-5);
- **cruzamento uniforme** – usa uma mascara binária, gerada aleatoriamente, de comprimento igual ao dos cromossomas. Os genes herdados por cada um dos descendentes dependem desta máscara (ver Figura 6-6).

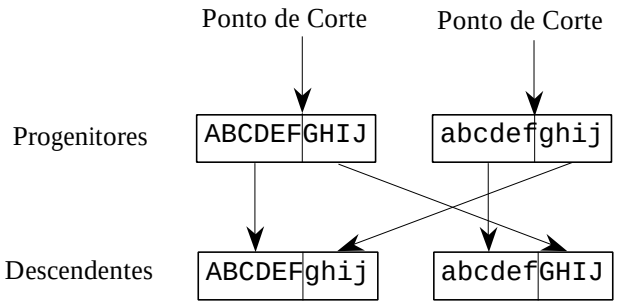

Figura 6-4: Um cruzamento de ponto único.

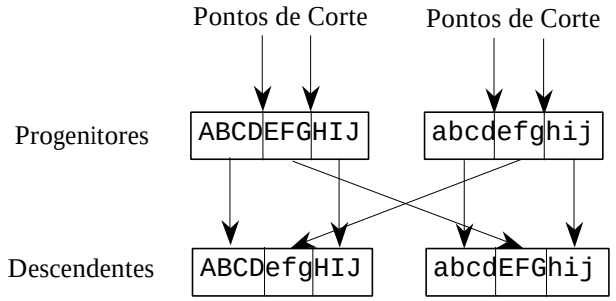

Figura 6-5: Um cruzamento de dois pontos.

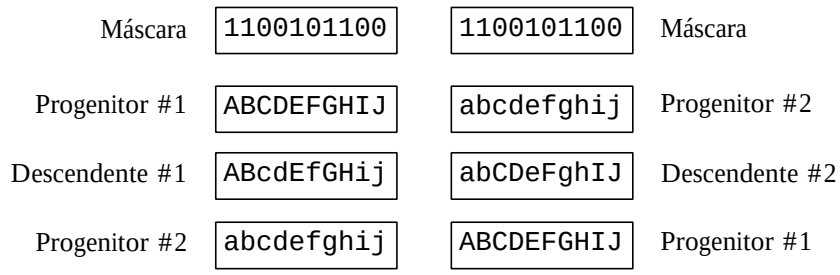

Figura 6-6: Um cruzamento uniforme.

Estes operadores foram originalmente desenvolvidos para representações binárias, embora possam ser também usados para representações que utilizam um alfabeto de símbolos diferente. No entanto, para determinadas representações estes operadores não se podem aplicar, sendo necessário definir outros operadores específicos para o alfabeto utilizado, tanto para a operação de recombinação como para o operador que será descrito na secção seguinte. Este é o caso, por exemplo, das representações genéticas baseadas em permutações usadas para os problemas como o do caixeiro viajante. Uma descrição mais detalhada destas representações assim como

dos respectivos operadores pode ser encontrada em Michalewicz (1996).

Mutação

O operador de mutação usado nos AG tem o objectivo de trazer de volta para a população os genes perdidos durante o processo de selecção de modo a que possam ser testados num novo contexto. Serve ainda para proporcionar novos genes que não estavam originalmente presentes na população inicial (Gen e Cheng 1997). Este operador permite dotar a pesquisa efectuada por um AG com uma componente aleatória, a qual se tem revelado de grande importância para este tipo de problemas.

Quando são utilizadas representações binárias, o operador de mutação escolhe aleatoriamente uma posição do cromossoma e altera-o de acordo com o procedimento ilustrado na Figura 6-7. Para outras representações usam-se outros procedimentos, como, por exemplo, a substituição do símbolo indicado pelo ponto de mutação por outro símbolo do alfabeto gerado aleatoriamente. No que se refere às representações baseadas em permutações, uma forma simples deste operador passa pela troca de posições de dois símbolos.

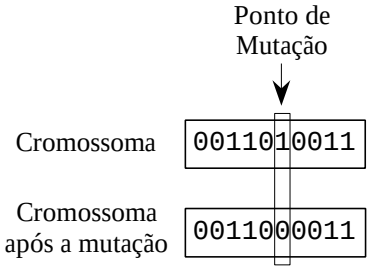

Figura 6-7: Uma simples operação de mutação.

6.3.4 Técnicas de Selecção

Em geral, diferentes técnicas de selecção são usadas para a escolha de progenitores de modo a efectuar a reprodução e para a escolha dos indivíduos que devem sobreviver para a geração seguinte. O procedimento de selecção começa por efectuar a conversão do valor de aptidão (f) de cada um dos indivíduos na população em valores designados por valores de selecção (f'). A etapa seguinte

consiste num processo de amostragem que, em função do valor de selecção, realiza a escolha dos indivíduos.

Existem vários métodos para a conversão dos valores de aptidão. Alguns dos métodos mais comuns são (Michalewicz, 1992):

- **Conversão de escala linear** (6-1);

$$f' = a \times f + b \tag{6-1}$$

- **Conversão de escala por truncagem sigma** (6-2) — $\overline{f}$ é o valor médio dos valores de aptidão e σ é o seu desvio padrão;

$$f' = max\left(0,\ f - \overline{f} + s\right) \tag{6-2}$$

- **Conversão de escala com a função potência** (6-3);

$$f' = f \tag{6-3}$$

Por vezes torna-se interessante classificar a população de acordo com o valor de aptidão dos indivíduos. Esta classificação é então usada para aplicar um plano reprodutivo em vez dos valores de aptidão. Este mecanismo apresenta-se especialmente atractivo para funções de aptidão pouco precisas ou quando são esperados valores de aptidão muito semelhantes. Este mecanismo proporciona, ainda, uma pressão de selecção constante na população (Whitley, 1989). A classificação pode ser efectuada por ordenação dos valores de aptidão. O valor de selecção é então calculado por (6-4), em que *i* identifica o indivíduo presente na posição *i* da população e a função *ordem* dá a posição desse indivíduo na classificação efectuada. Considera-se ainda que o indivíduo com maior valor de aptidão está na primeira posição da classificação efectuada.

$$f'(i) = \frac{n + 1 - ordem(i)}{\sum_{j=1}^{n} j} \tag{6-4}$$

Todos os métodos referidos são denominados por métodos de conversão *explícita*. Por oposição, existem também os métodos de conversão *implícita*. O mais comum destes métodos é conhecido por *selecção por torneio* (Goldberg e Deb, 1991).

Este escolhe aleatoriamente pares de indivíduos da população em função dos seus valores de aptidão. Esta escolha aleatória é feita normalmente com uma técnica de amostragem estocástica. Tendo escolhido um par, o indivíduo que é efectivamente seleccionado é aquele que apresenta um maior valor de aptidão. Este método de torneio pode ser generalizado para *n* indivíduos.

Quanto às técnicas de amostragem, estas podem ser essencialmente classificadas em *determinísticas* ou *estocásticas.* As primeiras escolhem os indivíduos da população que possuem maiores valores de selecção. As técnicas de amostragem estocásticas usam os valores de selecção para calcular a probabilidade de selecção de um dado indivíduo. Uma das técnicas de amostragem estocásticas mais frequentemente utilizada é conhecida por *Roleta* (Baker, 1987). Esta realiza uma escolha aleatória dos indivíduos com uma probabilidade proporcional aos valores de selecção. A Figura 6-8 mostra um gráfico em forma de tarte cuja área das fatias é proporcional ao respectivo valor de selecção, para uma população exemplo de dez indivíduos.

Indivíduo	Valor de Selecção
1	6,7
2	5,5
3	3,9
4	3,2
5	2,4
6	1,7
7	1,3
8	1,3
9	0,9
10	0,5

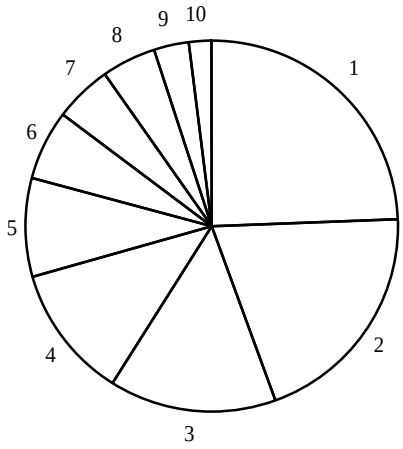

Figura 6-8: Um método de amostragem estocástica.

6.3.5 Modelos de Evolução

Nos AG tradicionais considera-se como etapa básica de evolução a geração dos indivíduos, e portanto, a uma etapa atómica corresponde uma geração. Nesta aproximação toda população da geração seguinte é constituída pelos novos indivíduos resultantes da reprodução. Existe uma outra alternativa, denominada por

AG de *estado estacionário*, em que a etapa atómica consiste em determinar um ou dois novos indivíduos (Whitley, 1989). Estes, após a geração da população inicial, prosseguem seleccionando, em cada ciclo, dois progenitores para reprodução (recombinação e mutação). Os novos indivíduos gerados são inseridos na população por troca com outros indivíduos da população actual. Geralmente, os indivíduos substituídos pelos novos indivíduos resultantes da reprodução são os piores (possuem o menor valor de aptidão) ou então são escolhidos aleatoriamente. Por vezes, se os novos indivíduos são piores que todos os indivíduos da população actual, não é efectuada qualquer substituição.

Um modelo intermédio de evolução, usado frequentemente, consiste em seleccionar uma percentagem da população actual para reprodução. Sendo este valor um parâmetro dos AG, é possível considerar este modelo como uma generalização dos diferentes modelos de evolução, dado que inclui as duas aproximações referidas, bem como todos os casos intermédios. Uma das principais diferenças do modelo original relativamente aos restantes modelos deve-se ao facto de nos últimos existir a possibilidade de competição entre indivíduos de gerações diferentes. É possível encontrar um estudo aprofundado acerca dos efeitos da substituição de uma percentagem de indivíduos na população para a geração seguinte no trabalho de DeJong e Sarma (1992).

Por vezes, existe interesse, na resolução de determinados problemas, em que os melhores indivíduos permanecem na população, o que implica que estes devem ser sempre escolhidos como sobreviventes. O *valor de elistimo* é um parâmetro de AG que segue esta aproximação e que define o número de indivíduos com o maior valor de aptidão que devem passar para as gerações seguintes. O conceito de elitismo foi pela primeira vez introduzido por DeJong (1975).

6.3.6 Parâmetros dos Algoritmos Genéticos

Um AG depende essencialmente de um conjunto de parâmetros para um bom funcionamento. Alguns destes parâmetros relacionam-se com o tamanho da população, a definição e aplicação dos operadores genéticos, a escolha dos critérios de selecção e substituição.

Tamanho da População

O *Tamanho da População* é um parâmetro de grande importância para qualquer AG. O seu valor afecta quer a qualidade da solução final quer o tempo de processamento. Uma população pequena apresenta uma diversidade genética pobre em termos de diversidade dos seus elementos constituintes, proporcionado uma menor cobertura do espaço de soluções, pelo que as soluções geradas tendem a ser fracas. Por outro lado, com o aumento do tamanho da população obtém-se um aumento da probabilidade de produzir melhores soluções, através duma maior cobertura do espaço de soluções e prevenindo a convergência prematura, embora à custa de um maior esforço computacional.

Taxa de Cruzamento ou Recombinação

A *Taxa de Cruzamento* define-se como a medida da possibilidade de aplicação do operador de cruzamento a um dado par de indivíduos. Os valores típicos para esta taxa situam-se no intervalo de 0,6 a 1,0. Quanto maior for esta taxa, maior é a quantidade de indivíduos introduzidos na população. Sendo o tamanho da população normalmente fixo, mais indivíduos tenderão a ser substituídos, logo pode haver a tendência para a perda de indivíduos com aptidão elevada. Para valores baixos desta taxa, gerar-se-à menos indivíduos em cada geração, o que pode originar um aumento do número de gerações para obter os mesmos resultados.

Taxa de Mutação

A *Taxa de Mutação* é uma medida da taxa de ocorrência da operação mutação sobre o genótipo de um dado indivíduo. Dado que uma taxa de mutação elevada tenderá a tornar o AG num algoritmo essencialmente aleatório, é usual esta taxa assumir valores relativamente baixos que, tipicamente, estão no intervalo de 0,001 a 0,1.

Taxa de Substituição

O parâmetro *Taxa de Substituição* define qual a proporção de indivíduos da

população substituída em cada geração. Se a percentagem de indivíduos a substituir for de 100% todos os indivíduos da população actual são substituídos pelos novos indivíduos resultantes da reprodução. Quanto menor for o valor desta taxa, menor será a diferenciação genética entre gerações e deste modo existirá uma convergência do algoritmo mais lenta.

Critérios de Paragem

O critério para a paragem do AG depende do problema em causa e do esforço computacional que é exigido. Em face do tempo e dos recursos disponíveis, é necessário definir qual a qualidade da solução que se pretende. Um critério usado com frequência passa por definir o número máximo de gerações em que a evolução deve ocorrer. Um segundo critério possível passa pela definição de um valor mínimo para o desvio padrão do valor de aptidão dos indivíduos na população. Uma vez atingido esse valor mínimo o algoritmo pára. Ainda um outro critério bastante comum de paragem consiste em fazer evoluir o algoritmo até se verificar que não se registam melhorias significativas das soluções ao longo de um dado número de gerações. No entanto, se for possível avaliar a qualidade das soluções encontradas, o critério de paragem pode ser o de encontrar uma "boa" solução.

6.3.7 Algoritmos Genéticos Paralelos

Uma parte da metáfora biológica motivadora para o uso de métodos de pesquisa baseados na evolução natural é devida ao facto de estes métodos serem implicitamente paralelos. Nas populações existentes na natureza, existem em paralelo milhares, ou mesmo milhões de indivíduos. Este facto sugere um grau de paralelismo que é directamente proporcional ao tamanho da população usado na pesquisa genética. Uma das classificações mais aceites para os AG paralelos relaciona-se com o grau de paralelismo, ou seja, a relação entre o processamento e a comunicação. Com esta classificação pode-se efectuar a distinção entre os modelos de granularidade fina e os modelos de granularidade grossa. Em geral, há três diferentes formas de exploração do paralelismo dos AG: os AG paralelos com populações globais, o modelo das ilhas e os AG celulares, também denominados por modelo de

granularidade fina massivamente paralelo.

Populações Globais com Paralelismo

A forma mais directa de implementar um algoritmo paralelo consiste numa implementação próxima de um AG canónico em que a selecção é realizada pelo método da selecção por torneio. O modelo usa um processador central que realiza as operações de selecção enquanto um conjunto de processadores escravos realizam as operações de recombinação, mutação e avaliação da aptidão (ver Figura 6-9). Para que todas as operações sejam realizadas em paralelo são necessários $n=m/2$ processadores escravos, onde m é o tamanho da população. Portanto, m deve ser par e apenas dois indivíduos residem em cada processador.

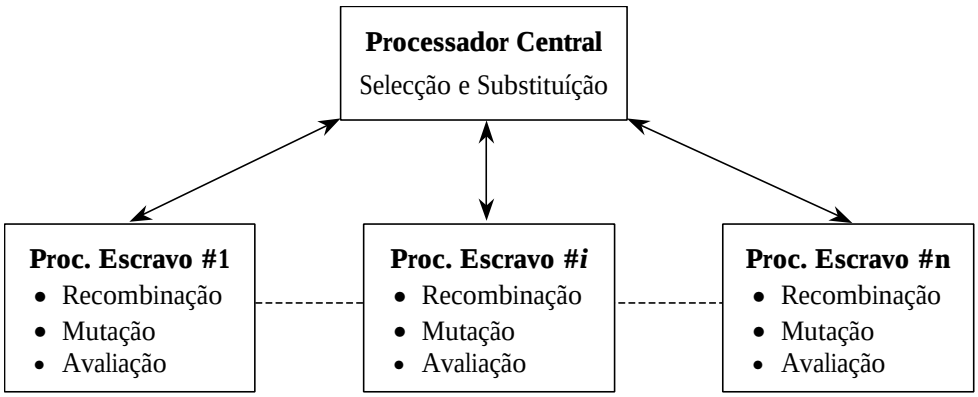

Figura 6-9: Um AG que executa os operadores genéticos em paralelo.

Modelos das Ilhas

O modelo das ilhas é tipicamente um modelo de granulosidade grossa, muita vezes designado por AG distribuído. Este modelo é normalmente usado em situações em que existem p processadores e o AG possui uma população de tamanho t, onde $p<<t$. Este considera um conjunto de sub-populações que evoluem em paralelo e de forma independente. Periodicamente, as sub-populações trocam indivíduos entre si. Cada sub-população evolui da mesma forma que num AG sequencial e o resultado é o melhor indivíduo encontrado no conjunto das sub-populações. Para a troca de indivíduos entre as sub-populações que residem nas

'*ilhas*', estas são organizadas segundo uma topologia em anel ou similar. Refira-se que a troca de indivíduos entre as ilhas, denominada por *migração*, permite a partilha de material genético entre as sub-populações (Starkweather *et al*, 1991).

Os AG distribuídos requerem a definição de alguns parâmetros adicionais. Um deles é a *taxa de migração*, que define a periodicidade com que os indivíduos migram. Outros parâmetros passam pela definição do número de indivíduos que migram por cada migração e qual o critério a usar para decidir quais os indivíduos que devem migrar. O último parâmetro relaciona-se com a topologia e estabelece quais são as ilhas de destino numa migração.

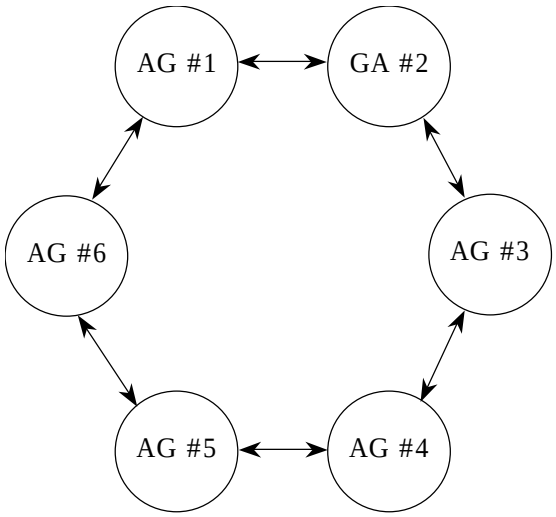

Figura 6-10: Um modelo para um AG distribuído baseado nas ilhas.

Algoritmos Genéticos Celulares

Considere-se que está disponível uma máquina com 2500 processadores simples organizados segundo uma grelha bidimensional (50×50) e em que a comunicação entre os processadores apenas se efectua com os seus vizinhos imediatos. Os AG celulares tiram partido deste tipo de arquitecturas para efectuar uma disposição espacial dos indivíduos de uma população, em que cada indivíduo é atribuído a um processador. Neste modelo, a operação de recombinação limita-se aos pares de indivíduos imediatamente vizinhos, usando uma selecção local para a escolha do parceiro de reprodução. Esta selecção poderá ser determinística ou probabilística.

O indivíduo descendente fica residente no processador que realizou a operação. Este modelo computacional foi proposto por Hillis (1990) e Collins e Jefferson (1991). Nos AG celulares, a propriedade de isolamento pela distância permite uma alta diversidade e uma pressão de selecção fraca devido ao operador de selecção local.

6.4 Os Algoritmos Genéticos e a Programação Lógica por Restrições

A relação entre os AG e a PLR que se discute nesta secção prende-se com a utilização dos AG como método de optimização para aplicações que recorrem a meta-interpretadores de PLR(*DF*). A abordagem proposta nesta secção, tendo em mente a combinação destas duas técnicas sob esta perspectiva, foi, essencialmente, inspirada por três trabalhos: o primeiro é o trabalho de Davis (1991) que propõe algumas linhas gerais de orientação para a hibridização dos AG com outras técnicas de optimização; outro é o trabalho descrito por Cotta *et al* (1995) onde se discute algumas alternativas para a hibridização dos AG com algoritmos *B*&*B* para a resolução de problemas do caixeiro viajante; e por último, o trabalho de Barnier e Brisset (1998) que apresenta uma forma de optimização por hibridização de AG com as técnicas de satisfação de restrições.

Antes de efectuar a descrição da metodologia proposta, que foi designada por *GeRL* (algoritmos **Ge**néticos, **R**estrições e **L**ógica), interessa, no entanto, começar por descrever a abordagem seguida por Barnier e Brisset com algum detalhe, dado que, como se demostrará adiante, esta pode ser implementada utilizando a metodologia *GeRL*.

6.4.1 Optimização por Hibridização de AG com a PLR

O trabalho desenvolvido por Barnier e Brisset (1998) teve como objectivo o desenvolvimento de um método de optimização baseado nos AG para a resolução de problemas com a PLR(*DF*). Segundo os autores, o método foi desenvolvido para os problemas combinatórios que possuem espaços de pesquisa vastos que,

possivelmente, possuem funções objectivo demasiado complexas quando se usam as técnicas de satisfação de restrições usuais e, também, para os casos em que as restrições do problema são demasiado complexas para um AG convencional.

O método desenvolvido é independente dos problemas, pelo facto de que, os genes dos indivíduos baseiam-se nos domínios das variáveis do problema em causa. Para o utilizador, entendido como programador de aplicações, as facilidades embebidas para as tarefas de optimização usando meta-interpretadores de PLR(*DF*), que normalmente usam um algoritmo *B&B* padrão, poderão ser substituídas por um AG. Com o método desenvolvido, os autores pretendem, por um lado, usar as técnicas de satisfação de restrições para obter soluções válidas num subconjunto do espaço de pesquisa, e por outro, usar os AG para explorar o espaço formado pelo conjunto dos sub-espaços e realizar a optimização.

Representação das Soluções

Os cromossomas usados neste método para representar as soluções são formados por uma cadeia de genes $G_1G_2...G_n$. A cada gene está associado uma variável de domínio e cada indivíduo pode constituir uma solução para o problema em questão ou então uma parcela do espaço de soluções. O tamanho dos cromossomas é dado pelo número de variáveis de domínio do problema. O gene G_i representa um sub-domínio (ou sub-conjunto) de $D(X_i)$, em que $D(X_i)$ representa o domínio da variável X_i.

Para além dos parâmetros de um AG comum, é ainda usado um parâmetro adicional, designado por *taxa de hibridização* (r), sendo este definido pela razão entre o tamanho de G_i e o tamanho de $D(X_i)$, ou seja, pela expressão (6-5).

$$r = \frac{|G_i|}{|D(X_i)|}, \quad 0 \le r \le 1 \tag{6-5}$$

De notar que quando r =1, o problema é equivalente a um problema de satisfação de restrições puro. No extremo oposto, quando r =0, o problema reduz-se a ser solucionado por um AG cuja representação dos indivíduos usa um alfabeto de número inteiros. O valor deste parâmetro r é idêntico para todas as variáveis, no

entanto, é possível usar valores diferentes.

Conhecendo o valor do parâmetro r, calcula-se o tamanho dos sub-domínios das variáveis com a expressão (6-6)[3]. Esta garante que cada gene possua pelo menos um valor. Os indivíduos da geração inicial são gerados durante a inicialização do AG com os sub-domínios para cada gene gerados de forma aleatória.

$$|G_i| = max\{1, round(r \times |D(X_i)|) \quad (6\text{-}6)$$

Função de Aptidão

O cálculo do valor de aptidão de cada indivíduo é realizado durante a resolução do sub-problema de satisfação de restrições correspondente. Cada sub-problema resulta da colocação das restrições (6-7). Estes sub-problemas são solucionados por um método tradicional de etiquetagem de todas as variáveis. Se for encontrada uma solução para um sub-problema então o valor de aptidão do respectivo indivíduo resulta da função objectivo. Por outro lado, se o sub-problema não contiver nenhuma solução então o indivíduo é rejeitado ou é-lhe atribuído um valor de aptidão baixo, possivelmente nulo.

$$X_i \in G_i \quad (6\text{-}7)$$

Por vezes, é necessário um grande esforço computacional para provar que um sub-problema não tem nenhuma solução. Nestes casos, torna-se necessário um mecanismo que interrompa o processo de etiquetagem das variáveis ao fim de um dado período de tempo pré-determinado, para que a avaliação da aptidão dos indivíduos não seja demasiado demorada.

Refira-se que basta encontrar a primeira solução para cada sub-problema de satisfação de restrições quando r <1, não sendo necessário realizar qualquer tarefa de optimização, dado que o AG por si já executa essa tarefa.

[3] A função *round* efectua o arredondamento de um valor real para o inteiro mais próximo.

Operadores Clássicos

Tendo em conta a representação escolhida para os indivíduos, os autores deste método argumentam que se pode adaptar os operadores de cruzamento e mutação clássicos, para as representações binárias, com apenas algumas modificações.

A Tabela 6-2 mostra um pequeno exemplo que ilustra como são executadas as operações clássicas de cruzamento e mutação modificadas para a representação escolhida. Este pequeno exemplo corresponde a um problema com três variáveis de decisão (X_1, X_2 e X_3), cada uma com o respectivo domínio finito. Os genes dos indivíduos indicados na tabela possuem um subconjunto do domínio original da respectiva variável. O operador de cruzamento coloca os pontos de corte de modo a separar os genes (não separa os valores do mesmo gene), enquanto que o operador de mutação modifica os valores de um gene escolhido de forma aleatória.

Tabela 6-2: Os operadores de cruzamento e mutação clássicos redefinidos.

Variáveis	X_1	X_2	X_3
Domínios	1..7	1..9	1..5
Progenitor P_1	1,2,3	4,7,8	1,3,4
Progenitor P_2	1,5,7	1,4,8	1,2,5
Descendente D_1	1,2,3	4,7,8	1,2,5
Descendente D_2	1,5,7	1,4,8	1,3,4
Mutação de P_1	1,2,3	1,7,8	1,3,4

Operador de Recombinação Orientado para os Conjuntos

Segundo os autores (Barnier e Brisset, 1998), os operadores clássicos para representações binárias modificados são robustos mas, no entanto, nem sempre são os mais eficientes. Dado que os sub-domínios das variáveis são sub-conjuntos de valores inteiros presentes no domínio original, os operadores orientados para os conjuntos seriam os mais adequados para a estrutura dos genes usada. No entanto, o uso de operadores orientados para conjuntos, como a união, a intersecção e a complementaridade podem, de forma drástica, modificar o tamanho dos genes, e por

isso não se revelam os mais adequados ao método. Deste modo, desenvolveram um novo operador de recombinação orientado para os conjuntos mais adequado ao método e que se encontra ilustrado na Tabela 6-3. Este operador é definido da seguinte forma:

- para cada gene G_i, determina-se o conjunto ($G_i^{\cup} = G_i^{P1} \cup G_i^{P2}$) resultante da união entre os sub-domínios dos progenitores;
- a partir do conjunto $G_i^{\cup}$ é construído um sub-conjunto G_i^{D1} do tamanho adequado, sendo que os seus valores são escolhidos aleatoriamente, para o primeiro descendente;
- os restantes valores ($G_i^{\cup} - G_i^{D1}$) destinam-se à primeira parte do segundo descendente G_i^{D2};
- se o tamanho de G_i^{D2} é demasiado pequeno escolhem-se aleatoriamente valores de G_i^{D1} até G_i^{D2} possuir o tamanho adequado.

Tabela 6-3: O operador de recombinação com os domínios finitos.

	X_1	X_2	X_3
Progenitor	1,2,3	4,7,8	1,3,4
Progenitor	1,5,7	1,4,8	1,2,5
União	1,2,3,5,7	1,4,7,8	1,2,3,4,5
1º Descendente	2,3,5	1,4,8	2,3,4
Resto	1,7	7	1,5
2º Descendente	1,5,7	4,7,8	1,2,5

6.4.2 Uma Metodologia que Combina a PLR com os AG

A metodologia *GeRL* desenvolvida para combinar os AG com a PLR, que se descreve nesta secção, tem como objectivo essencial a optimização de problemas combinatórios, em particular os PPLI, que envolvem um grande número de restrições com meta-interpretadores de PLR(*DF*) pelo uso de AG em alternativa ao algoritmo *B&B* tradicional. Esta metodologia segue as linhas gerais de orientação

defendidas por Davis (1991) para a combinação dos AG com outras técnicas de optimização. Os algoritmos que seguem estas linhas de orientação foram designados por AG híbridos.

Linhas Gerais de Orientação para a Hibridização

Davis refere que, embora as representações binárias possam codificar quase todos os tipos de problemas e que os operadores genéticos para estas representações são independentes do conhecimento da estrutura do problema, é desejável que, a aplicação dos AG para resolução de problemas do mundo real utilize o conhecimento da estrutura do problema em questão. Para além disso, devem incorporar características das técnicas específicas usadas para os solucionar. Com base nestes factos, Davis propôs algumas linhas gerais de orientação para implementar um AG híbrido e que se baseiam nos três princípios seguintes:

1. Usar a representação semelhante às estruturas de dados do algoritmo conhecido (algoritmo corrente) para solucionar o problema (que não é necessariamente óptimo) para codificar as soluções;
2. Incorporar as características positivas do algoritmo corrente no AG híbrido. Um exemplo consiste no facto de que o algoritmo corrente pode fornecer uma ou mais soluções para a população inicial se este for suficientemente rápido;
3. Criar ou adaptar os operadores genéticos de recombinação e mutação para a nova representação e incorporar heurísticas relacionadas com o domínio do problema como operadores.

A Metodologia *GeRL*

Em geral, uma aplicação escrita segundo o paradigma da PLR(*DF*) é constituída por três etapas: a identificação das variáveis do problema e o seu respectivo domínio, a colocação das restrições do problema envolvendo algumas das variáveis do problema e, finalmente, a enumeração das soluções ou a pesquisa da melhor solução segundo uma dada função objectivo. Esta última etapa, quando realiza uma tarefa de optimização, usa normalmente um algoritmo *B&B* embebido no meta-interpretador

de PLR(*DF*). É precisamente nesta etapa de optimização que se pretende usar um AG em vez de um algoritmo *B&B* padrão. A metodologia desenvolvida segue as linhas gerais de orientação propostas por Davis.

Em termos de arquitectura, a metodologia *GeRL* possui algumas características semelhantes relativamente ao método de hibridização proposto por Cotta *et al* (1995) em que um algoritmo *B&B* trabalha para um AG na resolução de problemas do caixeiro viajante. No seu método de hibridização, os operadores genéticos desenvolvidos implementam uma forma restrita de um algoritmo *B&B.*

A metodologia *GeRL*, e ao contrário do método de hibridização proposto por Cotta *et al*, tem por objectivo solucionar todos os tipos de problemas de optimização baseados em restrições pelo uso de meta-interpretadores PLR(*DF*). Esta metodologia assenta nas linhas gerais de orientação propostas por Davis e impõe que os operadores genéticos do AG devem ser implementados segundo o paradigma da PLR(*DF*), embora não defina nenhum método em particular para os implementar. O desenvolvimento de uma aplicação para solucionar problemas de optimização que segue esta metodologia requer que se efectue a escolha da representação mais adequada das soluções e o desenvolvimento, para essa representação, de métodos baseados no paradigma da PLR(*DF*) que implementem os operadores genéticos.

Uma aplicação baseada na metodologia *GeRL*, tal como se ilustra na Figura 6-11, divide-se em duas partes (Tavares et al, 2000a; Tavares et al, 2000b): a primeira é a aplicação principal e contém o AG para as tarefas de optimização, recebendo a designação de *processo principal*; a segunda corresponde à parte responsável pela execução dos operadores genéticos bem como o tratamento das restrições, sendo designada por *motor* da PLR(*DF*) ou abreviadamente por *motor* da PLR. De certa forma, pode-se considerar o processo principal como um processo cliente dos serviços de um servidor baseado no motor da PLR. O processo principal necessita de, em primeiro lugar, iniciar este servidor para poder usar os seus serviços. A inicialização do motor da PLR, começa por criar as variáveis de decisão do problema com os respectivos domínios e coloca as restrições que estabelecem as relações do problema a solucionar. A inicialização do motor da PLR termina retornando o seu estado inicial Π para o processo principal, sendo constituído,

normalmente, pelas variáveis de domínio, pelo armazém de restrições e pela função objectivo na forma de um termo linear. Com o estado inicial Π resultante da inicialização do motor da PLR, o processo principal torna-se capaz de criar a sua população inicial, efectuar as operações genéticas e avaliar a aptidão de cada indivíduo da população. Todas estas tarefas são executadas pelo motor da PLR. Todos os novos indivíduos gerados, como resultado da execução de uma operação genética (recombinação ou mutação) ou pela criação de novos indivíduos por parte do motor da PLR, deverão ser, normalmente, consistentes com as restrições do problema.

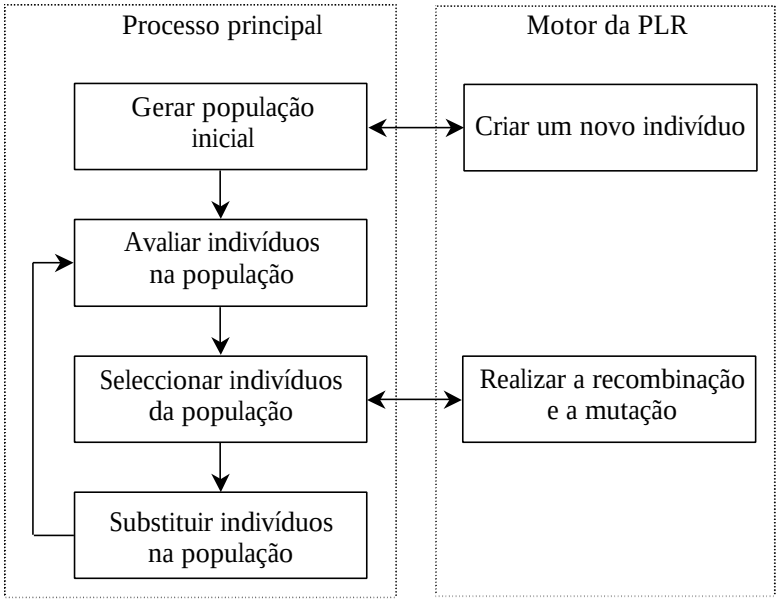

Figura 6-11: Esquema da combinação da PLR(*DF*) com os AG.

Como se referiu, esta metodologia *GeRL* não define qual a representação a usar nem como devem ser implementados os operadores genéticos. Desta forma, é possível escolher qual a melhor representação para o problema a solucionar, bem como permite o desenvolvimento de um vasto conjunto de operadores genéticos que possuem diferentes níveis de complexidade. Este operadores podem ir desde os simples operadores capazes de gerar apenas uma solução até a operadores mais complexos que usam o conhecimento específico da estrutura do problema para realizarem alguma forma restrita de optimização. Todos estes operadores, independentemente do nível de complexidade, devem ser capazes de gerar soluções

consistentes e o recurso à PLR pode constituir a forma mais natural para o conseguir. Em resumo, esta metodologia define uma divisão de tarefas que se encontra indicada na Tabela 6-4.

Tabela 6-4: Divisão de tarefas entre o motor da PLR e um AG.

Motor da PLR	**Processo principal (AG)**
• Gerar indivíduos consistentes de acordo com as restrições do problema; • Implementar o operador de cruzamento; • Implementar o operador de mutação; • Calcular o valor de aptidão para cada indivíduo gerado ou resultante de operações de cruzamento ou mutação.	• Responsável pelas tarefas relacionadas com as operações de selecção e substituição de indivíduos; • Em cada geração decide quais os indivíduos que serão envolvidos nas operações de recombinação e mutação; • Delegação da execução dos operadores genéticos no motor da PLR.

Questões Práticas de Implementação

Em termos práticos, uma aplicação desenvolvida seguindo a metodologia *GeRL* pode ser implementada usando dois tipos de ferramentas computacionais. O primeiro tipo consiste numa ferramenta de desenvolvimento de aplicações segundo o paradigma da PLR. Estas são, normalmente, um interpretador ou compilador da linguagem de programação Prolog com facilidades embebidas para a programação por restrições. O outro tipo é, normalmente, um pacote de *software* para a construção de aplicações baseadas nos AG, sendo, usualmente, por razões de desempenho, bibliotecas escritas numa linguagem procedimental ou orientada aos objectos. É essencial que o código escrito para ambas as ferramentas possa ser ligado. Este facto não é normalmente um factor limitativo, dado que os compiladores da Prolog possuem facilidades para ligar o código fonte escrito em Prolog com o código fonte escrito na linguagem de programação C ou C++, sendo estas últimas as linguagens mais frequentemente escolhidas para escrever as bibliotecas para a construção de aplicações baseadas nos AG.

Considerando que o processo principal corresponde em larga medida a um AG, que poderá ser baseado numa biblioteca escrita numa linguagem procedimental, será

lógico que este processo principal seja escrito usando uma linguagem procedimental. Um esqueleto para a rotina principal encontra-se ilustrado na Figura 6-12. Basicamente este começa por criar o estado inicial Π do motor da PLR em função do problema a solucionar e depois efectua a optimização com um AG.

```
procedimento processo_principal
começa
    <código de inicialização>
    Π ← PLR_criar_estado_inicial
    Min ← PLR_AG( Π, Parametros )
    <código de saída>
fim
```

Figura 6-12: O pseudo-código do processo principal escrito numa linguagem procedimental.

No entanto, mesmo que o AG usado se baseie numa biblioteca escrita numa linguagem procedimental, é sempre possível escrever o código para o processo principal usando apenas o paradigma da PLR(*DF*). Este poderá ser semelhante ao apresentado na Figura 6-13. Neste caso, o procedimento responsável pelo AG deve ser totalmente configurável, em termos dos seus parâmetros e dos operadores genéticos. Um AG destes será reutilizável, permitindo que o desenvolvimento de aplicações possa ser feito apenas escrevendo código segundo o paradigma da PLR(*DF*), nomeadamente na linguagem Prolog (ver Figura 6-14).

```
processo_principal ←
        inicialização_da_aplicação,
        criar_estado_inicial_plr( Π ),
        PLR_AG( Π, Parametros, Min ),
        Terminar_aplicação.
```

Figura 6-13: O pseudo-código do processo principal num linguagem lógica como a linguagem Prolog.

Com um AG configurável, como foi descrito, é necessário, antes de se iniciar a tarefa de optimização realizada pelo AG, registar a extensão dos predicados que serão invocados para a criação dos novos indivíduos, execução dos operadores genéticos e, eventualmente, a extensão do predicado que faz a avaliação da aptidão dos indivíduos na população. Podem, ainda, ser registados parâmetros adicionais que para o AG configurável não terão qualquer significado, mas que podem ser importantes para as diferentes operações.

```
funçao PLR_AG( Π, Parametros )
começa
        registar( Parametros )
        t ← 0
        P0 ← PLR_criar_população( Π, Parametros )
        PLR_avalia ( Π, P0 )
        enquanto ¬ Condição Final faz
                Pt' ← selecciona_de Pt
                Pt'' ← PLR_recombinação( Π, Pt' , Parametros )
                Pt''' ← PLR_mutação ( Π, Pt'' , Parametros )
                PLR_avalia ( Π, Pt''' )
                Pt+1 ← substitui ( Pt, Pt''' )
                t ← t + 1
        fim
        retornar melhor_indivíduo_de Pt
fim
```

Figura 6-14: O pseudo-código do AG na metodologia GeRL (todas as rotinas iniciadas por PLR_ são executadas no motor da PLR(*DF*)).

Representação das Soluções

Tal como Davis sugere nas suas linhas gerais de orientação para a hibridização, a representação das soluções com a metodologia *GeRL* segue, normalmente, uma representação directa, ou seja, a estrutura de dados do problema. No entanto, é possível usar uma qualquer representação indirecta, incluindo as representações que usam o alfabeto binário. Dado que os operadores genéticos são implementados segundo o paradigma da PLR, na situação mais comum, um cromossoma de um indivíduo pode tratar-se de uma lista que possui um comprimento *n*, sendo *n* igual ao número de variáveis do problema. O formato dos cromossomas pode então ser o seguinte:

$$[G_1, \ldots, G_i, \ldots, G_n]$$

Cada gene G_i (posição i do cromossoma) corresponde à variável X_i do problema. As soluções representadas são atribuições possíveis a todas as variáveis de domínio de valores que satisfazem as restrições do problema e, portanto, são soluções válidas.

De referir que os AG usados numa aplicação que segue a metodologia *GeRL* é independente da representação das soluções. Este facto deve-se a que, todas as

operações sobre os indivíduos na população de um AG são realizadas pelo motor da PLR. Para este AG, cada indivíduo na população faz referência à representação desse indivíduo no motor da PLR. Este facto permite que o AG usado numa aplicação desenvolvida com a metodologia *GeRL* seja independente da representação usada, desde que seja configurável e se escreva o código para os operadores genéticos.

Criação de Novos Indivíduos

Criar novos indivíduos e executar os operadores genéticos modifica o estado inicial Π do motor da PLR devido à instanciação das variáveis e, portanto, é preciso garantir que, após a finalização destas operações, o estado inicial do motor da PLR seja reposto. Por este facto, para que o estado do motor da PLR se mantenha, antes e após a realização de cada uma das operações, ter-se-à como consequência uma quebra da semântica inerente à programação em lógica em termos globais, embora localmente a cada operação a semântica da programação lógica se mantenha.

Para garantir que o estado do motor da PLR não se altere entre operações, sempre que o processo principal necessita de criar um novo indivíduo, este convoca a extensão do predicado *plr_nevo_indivíduo*/3 (ver Figura 6-15). Este tem como argumento de entrada o estado inicial Π e como argumento de saída um novo indivíduo. O novo indivíduo deve corresponder a uma solução válida para o problema. Na verdade, um novo indivíduo é criado ao convocar a extensão do predicado *novo_indivíduo*/2. A extensão do predicado *plr_novo_indivíduo*/3 serve apenas para repor o estado do motor da PLR tal como este é inicialmente, ou seja o estado inicial Π.

```
plr_novo_indivíduo( Π, Parâmetros, Novo_indivíduo ) ←
        (
                novo_indivíduo( Π, Parâmetros ),
                assert( indivíduo( Π )),
                fail
        ;
                retract( indivíduo( Novo_indivíduo ))
        ).
```

Figura 6-15: Criação de um novo indivíduo na metodologia *GeRL*.

O processo de criação de um novo indivíduo pode ser visto como um procedimento de etiquetagem comum, onde os valores atribuídos às variáveis são escolhidos aleatoriamente a partir dos seus respectivos domínios, sendo suficiente a primeira solução encontrada. No entanto, é possível o desenvolvimento de outros formas diferentes para geração de novos indivíduos, incluindo aquelas que obtêm soluções geradas por outros métodos distintos.

Operadores Genéticos

Tal como se referiu anteriormente, os operadores genéticos também modificam o estado inicial Π do motor da PLR e, portanto, deve-se garantir que a sua implementação seja também capaz de o repor. Por outro lado, deve-se considerar que estes operadores serão executados muitas vezes, especialmente o operador de recombinação, e deste modo, devem ser simples e de grande eficiência. A Figura 6-16 e a Figura 6-17 mostram o pseudo-código para a extensão dos predicados *plr_recombinação*/6 e *plr_mutação*/4 que, respectivamente, executam as operações de recombinação e mutação. Estes predicados consideram os requisitos para Π.

```
plr_recombinação( Π, Parâmetros, P1, P2, D1, D2 ) ←
        parâmetros_controlo_recombinação ( Parâmetros, Controlo ),
        (
                recombinação( Π, Controlo, P1, P2 ),
                assert( descendente( Π )),
                fail
        ;
                retract( descendente( D1 ))
        ),
        (
                recombinação( Π, Controlo, P2, P1 ),
                assert( descendente( Π )),
                fail
        ;
                retract( descendente( D2 ))
        ).
```

Figura 6-16: O operador de recombinação na metodologia *GeRL*.

Usualmente, o operador de recombinação constroi um descendente (nova solução) pela cópia de alguns genes de um dos seus progenitores e os restante do outro progenitor. A decisão de quais os genes que vêm de cada progenitor é

determinada pelos parâmetros de controlo. Geralmente, isto significa que os descendentes herdam a maior quantidade possível de informação genética dos seus progenitores. Este facto tem de ser seguido no desenvolvimento do operador de recombinação, assegurando que as novas soluções resultantes (indivíduos) são consistentes com as restrições do problema.

O operador de mutação é um operador mais simples que o operador de recombinação. Basicamente, altera aleatoriamente o valor de um ou mais genes. Os parâmetros de controlo definem quais são os genes que devem ser modificados. É também condição necessária que a solução resultante seja consistente com as restrições do problema.

```
plr_mutação( Π, Parâmetros, Indivíduo, Mutado ) ←
        parametros_controlo_mutação ( Parâmetros, Controlo ),
        (
                mutação( Π, Controlo, Indivíduo ),
                assert( indivíduo_mutado( Π )),
                fail
        ;
                retract( indivíduo_mutado( Mutado ))
        ).
```

Figura 6-17: O operador de mutação na metodologia *GeRL*.

Representação Baseada em Sub-Conjuntos dos Domínios Finitos na Metodologia GeRL

A abordagem de Barnier e Brisset (1998) anteriormente descrita, na hibridização de AG com a PLR para a optimização de problemas, pode ser facilmente implementada com a metodologia *GeRL*. Existindo um AG genérico configurável como o que foi referido na secção anterior, um utilizador apenas precisa de escrever o código fonte para a criação de novos indivíduos, para os operadores genéticos e para a avaliação dos indivíduos. Naturalmente, este código fonte é escrito na linguagem Prolog com extensões para a PLR(*DF*), embora não seja uma condição obrigatória. O parâmetro adicional proposto por Barnier e Brisset, designado por taxa de hibridização (r), apenas entra nos procedimentos de geração de novos indivíduos e nos procedimentos que implementam os operadores genéticos e, como

tal, o AG genérico configurável apenas recebe o seu valor e reencaminha-o para os procedimentos definidos pelo utilizador.

Usando a Prolog como linguagem para a escrita das aplicações, os indivíduos podem ser representados por uma lista de genes ([G_1, G_2, ... , G_n]), em que cada gene é também uma lista, contendo neste caso um sub-conjunto dos valores do domínio da respectiva variável. Por exemplo, o progenitor P_1 indicado na Tabela 6-2 será representado na seguinte forma:

[[1, 2, 3], [4, 7, 8], [1, 3, 4]]

A criação de novos indivíduos usa o parâmetro r para determinar o tamanho de cada um dos genes, ou seja, o número de valores que estes devem possuir em função do tamanho original dos domínios das respectivas variáveis. Os valores escolhidos para cada um dos genes serão normalmente escolhidos de forma aleatória.

Os operadores genéticos implementam simples operações sobre conjuntos. O operador de mutação necessita, também, do estado inicial Π para poder modificar, de forma parcial ou total, um ou mais genes. O parâmetro r também é necessário para estes operadores.

Finalmente, a avaliação dos indivíduos é efectuada pela colocação das restrições (6-7) anteriormente referidas. Aqui colocam-se todas as questões, já discutidas na secção 6.4.1, relativamente ao facto de o sub-problema representado por um indivíduo ter ou não soluções e também relativamente ao desempenho.