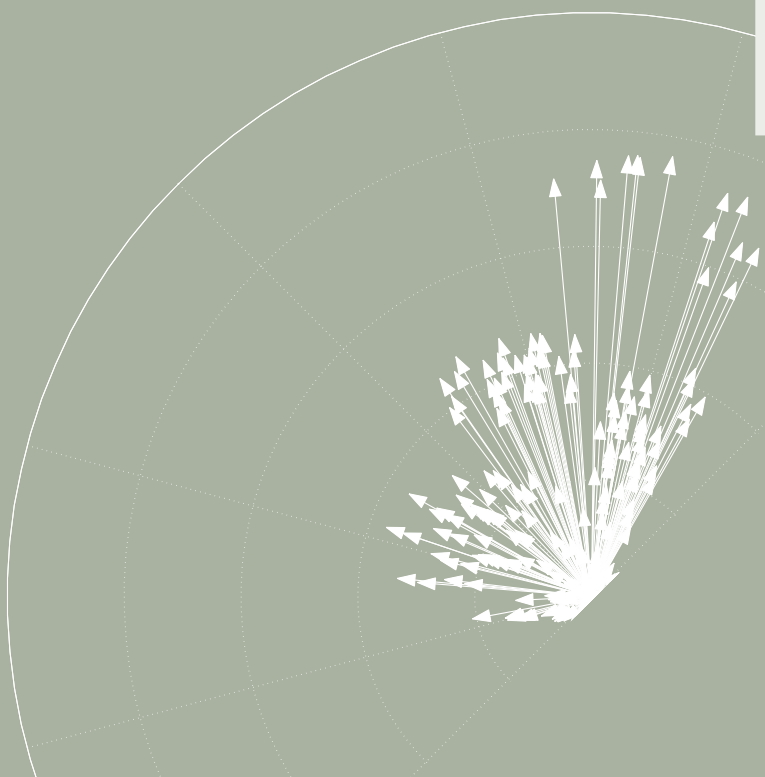
MANUAL DE COMPUTAÇÃO EVOLUTIVA E METAHEURÍSTICA

ANTÓNIO GASPAR-CUNHA
RICARDO TAKAHASHI
CARLOS HENGGELER ANTUNES
COORDENADORES



IMPRENSA DA UNIVERSIDADE DE COIMBRA
COIMBRA UNIVERSITY PRESS

EDITORA ufmg

CAPÍTULO 2

Algoritmos Genéticos

Francisco J. B. Pereira

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra

Os Algoritmos Genéticos (AGs) são métodos de pesquisa probabilísticos inspirados nos princípios da selecção natural e da genética. Foram desenvolvidos por John Holland e pelos seus alunos da Universidade de Michigan durante os anos 1960 (Holland, 1975). De acordo com Goldberg (1989), a investigação tinha um duplo objectivo: efectuar um estudo rigoroso dos mecanismos de adaptação existentes na natureza e desenvolver modelos computacionais que retivessem os princípios básicos identificados nos sistemas naturais.

1. A Inspiração Biológica

A evolução biológica efectua uma pesquisa num espaço de grande dimensão e complexidade, constituído por todas as possíveis combinações genéticas que podem ser geradas. Alguns dos elementos pertencentes a este espaço de procura poderão originar organismos viáveis. O processo evolutivo contribui para a obtenção de um conjunto de indivíduos bem adaptados ao meio ambiente em que se encontram.

De acordo com os pressupostos apresentados por Darwin, o processo de evolução natural recorre a dois mecanismos básicos: a **selecção** e a **reprodução com variação** (Darwin, 1859). A selecção

garante que os indivíduos mais aptos (os que melhor se adaptam ao meio ambiente) têm maior probabilidade de sobreviver. Graças a esta situação, têm a possibilidade de gerar um maior número de descendentes, propagando assim as suas características ao longo das gerações. A existência de um mecanismo de variação associado à reprodução garante que os descendentes gerados não são uma cópia fiel dos progenitores. A combinação destas duas forças permite que, ao longo de sucessivas gerações, a população de indivíduos evolua de forma gradual.

A formulação, no início do século XX, da teoria genética da hereditariedade permitiu integrar o conceito de gene no processo evolutivo, tornando mais claro de que forma é que as alterações são introduzidas na população. A variabilidade das espécies passa a ser explicada através da aplicação de um conjunto de operadores genéticos, como o operador de recombinação[1] ou de mutação. A moderna teoria da evolução natural, resultante da fusão das ideias originais de Darwin com a genética Mendeliana é, normalmente, designada por Neodarwinismo.

Os princípios básicos da evolução biológica serviram de inspiração para o desenvolvimento de algoritmos de pesquisa e adaptação em ambientes computacionais, tendo em vista aplicações, tanto no campo da engenharia, como no da biologia. A questão central associada ao surgimento dos AGs é colocada de forma simples por John Holland: "Como transformar as ideias de Darwin em algoritmos?"(Holland, 2000).

2. Estrutura de um Algoritmo Genético

Adoptando a inspiração natural, os AGs processam conjuntos de elementos do espaço de procura (potenciais soluções para o problema). Estes conjuntos, usualmente denominados populações, são transformados (evoluídos) ao longo de sucessivas iterações (gerações). O objectivo é encontrar uma solução de qualidade elevada (idealmente a solução óptima) para o problema que está a ser resolvido. No início são escolhidas aleatoriamente várias soluções. A partir deste conjunto inicial, e de acordo com os dois mecanismos básicos utilizados pela evolução natural referidos anteriormente, as transformações na população de soluções são efectuadas da seguinte forma:

- Os elementos mais aptos de uma determinada geração são seleccionados para servirem de progenitores das soluções que irão aparecer na geração seguinte. Na maior parte dos casos, o processo de selecção é probabilístico;
- Operadores de transformação, designados operadores genéticos, actuam sobre os elementos seleccionadas originando as novas soluções.

Existem duas decisões cruciais que é necessário tomar quando se pretende aplicar um AG na resolução de um problema:

- Escolha da representação para as soluções que fazem parte do espaço de procura: a proposta original de Holland defendia a utilização de um código binário para efectuar a representação das possíveis soluções de um problema. Desta forma, os indivíduos processados por um AG clássico são simplesmente sequências de 0's e 1's codificando a informação necessária para representar um ponto do espaço de procura;
- Definição da função de aptidão que associe a cada solução uma medida de qualidade, representando a sua capacidade para resolver o problema em causa.

Para além da inspiração biológica, os AGs adoptaram um conjunto de designações para qualificar situações e entidades relacionadas com o seu modo de funcionamento. Assim, o conjunto das

[1] O operador de *recombinação* também é chamado, por vezes, de operador de *cruzamento*.

soluções que vão sendo processadas é usualmente designado por **população**. Cada uma das iterações do processo de optimização é considerada uma **geração** e a sequência das várias populações reflecte a **evolução** ao longo do tempo. Os elementos que constituem uma determinada população são designados por **indivíduos** ou **cromossomas**. Os **genes** são os constituintes básicos dos cromossomas, podendo ser encarados como cada uma das características elementares de uma solução. Estão localizados numa determinada posição ou **locus** e cada um deles pode tomar um valor de entre um conjunto de possibilidades, denominadas **alelos**.

A estrutura genérica de um AG é ilustrada no Algoritmo 1. Nele pode ser consultado o processo iterativo que vai gerando sucessivas populações (passos 4 a 9).

Algoritmo 1 Estrutura Genérica de um Algoritmo Genético

1: $t \leftarrow 0$
2: Gerar a população inicial $P(t)$
3: Avaliar os indivíduos de $P(t)$
4: **repita**
5: Seleccionar progenitores $P'(t)$ a partir de $P(t)$
6: Aplicar operadores genéticos a $P'(t)$ obtendo a nova população $P(t+1)$
7: Avaliar P(t+1)
8: $t \leftarrow t+1$
9: **até** (critério de terminação atingido)
10: Devolver resultado final da optimização

Devido ao mecanismo de selecção, a qualidade média dos elementos que constituem a população tem tendência para aumentar ao longo do tempo. Os operadores genéticos são os responsáveis pela obtenção de novas soluções, ao mesmo tempo que tentam garantir que o processo mantenha um nível adequado de diversidade. Um AG possui dois tipos de operadores genéticos fundamentais: recombinação e mutação. Os detalhes de implementação de cada um destes operadores dependem da representação adoptada. Tal como as duas questões previamente identificadas (representação e aptidão), este é outro ponto fulcral e deve ser abordado com grande cuidado, de modo a maximizar a eficácia do processo de optimização. Existem, no entanto, princípios genéricos associados ao funcionamento destes operadores que são descritos a seguir:

- **Recombinação**: Operador estocástico que efectua a troca de material genético entre dois progenitores, conduzindo à criação de dois novos indivíduos (os descendentes). Estes serão formados por sequências genéticas parciais de cada um dos elementos originais. O operador de recombinação é considerado o principal responsável pela pesquisa efectuada por um AG. O objectivo central da sua aplicação é simples de compreender: ele actua sobre indivíduos que foram seleccionados na população actual, ou seja, são soluções com algum potencial. A combinação de características complementares de indivíduos promissores poderá original novas soluções que integrem as vantagens de ambos e que possuam uma aptidão acrescida para o problema que está a ser resolvido.

- **Mutação**: Operador unário que actua sobre as soluções resultantes da recombinação e que altera ligeiramente algumas das suas características. Num AG, a mutação é um processo completamente aleatório e tem como objectivo manter um nível de diversidade adequado na população, garantindo que alelos que eventualmente desapareçam tenham a possibilidade de reaparecer. Existem variantes de algoritmos evolutivos descritos em outros capítulos deste manual, nos quais a mutação desempenha um papel diferente.

Na proposta original de Holland (1975), é descrito um terceiro operador genético denominado **inversão**. Este operador unário tem a capacidade de proceder a um reordenamento parcial dos genes

existentes no cromossoma com o objectivo de tornar a aplicação do operador de recombinação mais eficaz. Embora faça parte da descrição original de um AG, a inversão é muito menos utilizada do que os dois outros operadores referenciados[2]. O próprio Holland, num trabalho recente (Holland, 2000), refere-se à recombinação e à mutação como os dois principais operadores genéticos de um AG. Por esse motivo, apenas estes dois operadores serão apresentados neste texto.

À medida que o número de gerações aumenta, um AG converge gradualmente para regiões do espaço de procura onde se encontram soluções promissoras. A optimização termina quando um determinado critério de terminação é atingido. Os critérios mais comuns são:

- Número limite de gerações atingido;
- Descoberta de uma solução com qualidade pretendida;
- Inexistência de melhoria durante um determinado período de tempo.

Ao atingir um critério de terminação, o AG devolve o resultado final da optimização. Existem duas alternativas para apresentação de resultados: devolver a melhor solução encontrada ao longo da optimização ou devolver um conjunto de indivíduos de qualidade elevada (por exemplo, devolver todos os indivíduos que fazem parte da última geração).

Bibliografia Básica

John Holland apresentou a sua ideia original sobre AGs no livro *Adaptation in Natural and Artificial Systems*, originalmente publicado em 1975 (Holland, 1975). É uma obra rigorosa, onde os principais conceitos relacionados com o funcionamento destes algoritmos são apresentados e analisados. É de leitura essencial para quem pretenda compreender os fundamentos teóricos associados ao funcionamento de um AG.

Ao longo dos anos têm sido publicados outros textos introdutórios que são uma excelente porta de entrada para quem pretende perceber os principais conceitos relacionados com o funcionamento de um AG. Todos estes livros descrevem exemplos práticos de aplicação, o que facilita a compreensão dos conceitos apresentados. O livro de David Goldberg *Genetic Algorithms in Search, Optimization, and Machine Learning*, apesar de ter sido publicado em 1989, mantém a sua actualidade (Goldberg, 1989). Alternativas mais recentes e igualmente relevantes são os livros *Genetic Algorithms + Data Structures = Evolution Programs* de Michalewicz (1992), *An Introduction to Genetic Algorithms* de Mitchell (1996) ou *An Introduction to Evolutionary Computing* de Eiben e Smith (2003).

A obra *Handbook of Evolutionary Computation* editada por Bäck et al. (1997) agrupa um conjunto muito abrangente de contribuições relativas a toda a área da computação evolutiva. É de consulta obrigatória sempre que se pretenda aprofundar ou procurar referências adicionais sobre um assunto específico. Por sua vez, o livro *Inteligência Artificial: Fundamentos e Aplicações* de Costa e Simoes (2008) merece uma referência. Embora seja um livro genérico de introdução à área da inteligência artificial possui uma secção importante dedicada a algoritmos evolutivos. Para alguns leitores poderá ter a vantagem adicional de ser escrito em português. Finalmente, ao longo do texto apresentamos diversas referências bibliográficas que podem servir de introdução aos diferentes tópicos abordados.

3. Exemplo: Aplicação de um AG ao Problema da Mochila

Em termos práticos, para aplicar um AG a um determinado problema é necessário definir quatro componentes essenciais: representação para as soluções, função de aptidão, método de selecção

[2] A razão para este facto está, provavelmente, relacionada com o surgimento de representações e de operadores de recombinação alternativos que tornam a sua aplicação menos importante.

Tabela 2.1: Propriedades dos objectos da instância do KSP a optimizar

	Obj_1	Obj_2	Obj_3	Obj_4	Obj_5	Obj_6	Obj_7	Obj_8
Peso	10	18	12	14	13	11	8	6
Valor	5	8	7	6	9	5	4	3

e operadores genéticos. É ainda necessário escolher valores para alguns parâmetros que controlam o funcionamento do AG, tais como o tamanho da população, o número máximo de gerações a processar (assumindo que este é o critério de terminação adoptado) ou a probabilidade de aplicação dos operadores genéticos. Para além destes, podem existir outros parâmetros específicos das componentes que constituem o AG.

O Problema da Mochila

O problema da mochila[3] (KSP) é um exemplo clássico de uma situação de optimização combinatória e pode ser enunciado da seguinte forma: dado um conjunto de objectos, cada um deles com um peso e um valor específico, determinar o subconjunto mais valioso cujo peso total não ultrapasse a capacidade de uma mochila.

O problema pode ser descrito mais formalmente: existe um conjunto S com N objectos, sendo cada objecto Obj_i caracterizado por um peso P_i e por um valor V_i, $i = 1, ..., N$ (todos os pesos e todos os valores são positivos). Existe, além disso, uma mochila com capacidade $C > 0$. O objectivo é encontrar um subconjunto de objectos que satisfaça as seguintes condições:

$$\max \sum_{i=1}^{N} x_i \times V_i \tag{2.1}$$

$$\text{sujeito a } \sum_{i=1}^{N} x_i \times P_i \leq C \tag{2.2}$$

Nas equações 2.1 e 2.2, $x_i \in \{0, 1\}$ ($x_i = 1$ se o objecto Obj_i se encontra na mochila, $x_i = 0$ caso contrário).

Implementação do AG

As características da instância do KSP que vai ser utilizada para ilustrar o funcionamento de um AG são as seguintes[4]:

- Número de objectos: 8
- Capacidade da mochila: 35
- O peso e o valor de cada um dos objectos podem ser consultados na tabela 2.1.

[3] Do inglês, *Knapsack Problem*. Na literatura, o problema descrito nesta secção é conhecido como 0-1 Knapsack Problem. Existem diversas variantes do KSP como, por exemplo, o Multidimensional Knapsack Problem.

[4] A instância escolhida é extremamente simples e serve apenas para exemplificar o funcionamento de um AG.

Representação

Uma representação comum para este problema é uma sequência binária com N posições (em que N é o número de objectos existentes). Cada posição está associada a um objecto: se tiver o valor 1 significa que foi escolhido para ser incluído na mochila. Em relação ao exemplo que estamos a considerar, a sequência {10100000} é uma solução possível que especifica que os objectos Obj_1 e Obj_3 se encontram na mochila.

A representação proposta é a mais natural para o problema do KSP. Para além disso, é uma representação binária, o que a torna especialmente apropriada para aplicação de um AG simples. Possui, no entanto, uma limitação importante, uma vez que permite a existência de muitas soluções inválidas no espaço de procura. Uma solução inválida no KSP ocorre quando o peso total do conjunto de objectos escolhidos excede a capacidade total da mochila. Uma vez que a representação não inclui nenhum mecanismo que limite a quantidade de objectos a escolher (por exemplo, a solução {11111111} faz parte do espaço de procura definido por esta representação), o processamento de soluções inválidas por parte do AG é inevitável. Existem várias alternativas para lidar com esta situação. A lista seguinte enuncia as três mais relevantes:

- **Penalizar**: A opção mais simples é tratar das soluções inválidas durante a avaliação. De acordo com este princípio, as soluções ilegais são penalizadas. A amplitude da penalização é proporcional ao grau de violação das restrições.

- **Reparar**: A reparação de uma solução inválida é uma possibilidade interessante, uma vez que permite manter apenas indivíduos legais na população. Esta abordagem obriga a desenvolver um algoritmo de reparação simples e rápido, específico para o problema a optimizar. O principal ponto fraco desta estratégia é a eventual dificuldade em encontrar um algoritmo de reparação. Em alguns casos, a correcção de uma solução ilegal pode ser uma tarefa extremamente díficil de resolver.

- **Alterar Representação**: Existem sempre várias representações possíveis para o mesmo problema. A decisão de qual a representação mais apropriada para uma determinada situação deve levar em consideração as propriedades do espaço de procura associado a cada uma delas (por exemplo, se admitem ou não soluções ilegais). No entanto, é importante referir que este não deve ser o único critério a considerar quando se procede à escolha da representação. É essencial considerar as vantagens e desvantagens associadas a cada hipótese e optar pela que oferece mais garantias em termos de eficácia.

Neste ponto optamos pela abordagem baseada em penalização. Na secção 5 serão descritas e analisadas abordagens alternativas.

Aptidão

A função de aptidão atribui a qualidade a uma solução, especificando a sua adequação para o problema em causa. A qualidade é estabelecida de forma quantitativa, o que facilita a comparação entre o mérito relativo de diferentes soluções.

No caso do KSP, o critério essencial para determinar a qualidade de uma solução é o valor total proporcionado pelos objectos escolhidos. É formalizado como um problema de maximização, pelo que a função de aptidão deve atribuir valores mais elevados aos melhores indivíduos. No entanto, e dada a representação adoptada, é necessário incluir na função de aptidão um mecanismo que penalize as soluções inválidas. Se esta componente não fosse considerada, é trivial verificar que a melhor solução para o problema seria a que escolhe todos os objectos para serem incluídos na mochila. Claramente esta é uma solução inválida, uma vez que o peso total dos objectos excede a capacidade da mochila.

A avaliação da aptidão de uma solução S é feita recorrendo à equação 2.3:

$$Qualidade(S) = \sum_{i=1}^{N} x_i \times V_i - Pen(S) \tag{2.3}$$

em que $x_i \in \{0, 1\}$ ($x_i = 1$ se o objecto Obj_i se encontra na mochila, $x_i = 0$ caso contrário) e $Pen(S)$ é a penalização associada à solução S. O valor da penalização pode ser calculado de diversas formas. Deve, no entanto, obedecer a três princípios básicos: i) a penalização é proporcional ao grau de violação da restrição de capacidade; ii) nunca deve compensar exceder a capacidade da mochila; iii) a penalização de um índivíduo válido é 0. Neste texto adoptamos uma penalização linear, calculada de acordo com a expressão apresentada na equação 2.4. Para uma descrição de outros tipos de penalização, consultar (Costa e Simoes, 2008; Michalewicz, 1992), e o capítulo 14 deste livro.

$$Pen(S) = \begin{cases} 0, & \text{se S legal;} \\ \rho \times (\sum_{i=1}^{N} x_i \times P_i - C), & \text{caso contrário} \end{cases} \tag{2.4}$$

O valor de ρ é obtido através da seguinte expressão:

$$\rho = max_{i=1,\ldots,N} \frac{V_i}{P_i} \tag{2.5}$$

AG: Simulação de uma Iteração

Estabelecidas a representação e a aptidão, o AG pode iniciar a optimização. Será ainda necessário especificar alguns valores para parâmetros, mas eles serão apresentados à medida que forem necessários.

Nesta secção apresentamos detalhadamente todas as etapas que constituem um ciclo iterativo completo do AG. De acordo com o algoritmo 1, a criação da população inicial é a primeira tarefa a ser realizada. Existe um parâmetro, o **tamanho da população**, que indica quantos indivíduos fazem parte de cada geração. Num AG simples, o seu valor mantém-se constante ao longo da optimização. A dimensão da população a escolher para um caso concreto depende de vários factores, mas os valores padrão utilizados na maioria das situações pertencem ao intervalo [100, 500]. No exemplo descrito nesta secção adoptamos uma população de apenas 10 indivíduos para manter a explicação o mais simples possível.

A população inicial é gerada aleatoriamente. Considerando a representação adoptada e a instância do KSP que está a ser optimizada, é necessário apenas gerar 10 sequências binárias de tamanho 8. Após serem gerados os 10 indivíduos da população inicial, aplica-se a equação 2.3 para calcular a qualidade de cada um deles. A tabela 2.2 apresenta um resumo dos individuos gerados e avaliados. Como se pode verificar, na população inicial existem 3 indivíduos inválidos. A melhor solução (I_5) tem qualidade 17.0, correspondendo à escolha dos objectos Obj_2 e Obj_5.

O processo que conduz à criação da população da nova geração inicia-se com a selecção dos progenitores. Neste passo, os indivíduos de melhor qualidade da geração actual têm maior probabilidade de serem seleccionados para dar origem às novas soluções. Vários métodos de selecção têm sido apresentados por diversos autores. No trabalho original de Holland é proposta a selecção por roleta (ou selecção proporcional) (Holland, 1975). Com este método, a probabilidade de um indivíduo ser seleccionado baseia-se na razão entre a sua qualidade e a soma das qualidades de todos os indivíduos da população. Diversas desvantagens têm sido associadas à selecção por roleta (Eiben e Smith, 2003). A principal é a dificuldade em controlar a pressão selectiva, i.e., em estabecer uma relação apropriada entre o mérito relativo de um indivíduo e a probabilidade de ser seleccionado como progenitor. Situações em que a pressão selectiva é demasiado elevada, i.e., quando apenas um pequeno grupo dos melhores indivíduos conseguem ser escolhidos como progenitores, podem levar o algoritmo a convergir prematuramente.

Tabela 2.2: População Inicial

Indivíduos	$\sum V_i$	$\sum P_i$	Penalização	Qualidade
$I_1 = \{10100000\}$	12	22	0	12.0
$I_2 = \{10100010\}$	16	30	0	16.0
$I_3 = \{00010011\}$	13	38	2.1	10.9
$I_4 = \{00010000\}$	6	14	0	6.0
$I_5 = \{01001000\}$	17	31	0	17.0
$I_6 = \{00000010\}$	4	8	0	4.0
$I_7 = \{01110110\}$	30	63	19.4	10.6
$I_8 = \{01000100\}$	13	29	0	13.0
$I_9 = \{10101101\}$	29	62	18.7	10.3
$I_{10} = \{00011000\}$	15	27	0	15.0

Por outro lado, se a pressão selectiva for demasiado baixa, i.e., se todos os indivíduos tiverem aproximadamente a mesma probabilidade de serem escolhidos como progenitores, o algoritmo deixa de ter a capacidade de identificar e convergir para áreas promissoras no espaço de procura.

Neste texto optamos por aplicar a **selecção por torneio** (Eiben e Smith, 2003; Mitchell, 1996). Este método apresenta duas características importantes que o ajudam a controlar a pressão selectiva: por um lado, não está excessivamente dependente dos valores concretos da qualidade dos indivíduos. A única informação de que necessita para fazer a selecção é uma hierarquização das soluções (em termos de mérito). Em segundo lugar, possui um parâmetro designado **tamanho do torneio** que ajuda a controlar explicitamente a pressão selectiva. O algoritmo 2 ilustra como a selecção por torneio escolhe um progenitor.

Num AG simples, o tamanho da população mantém-se constante ao longo das gerações. Se a população for constituída por P indivíduos, é necessário seleccionar P progenitores para dar origem aos P descendentes que formarão a nova geração. Deste modo, o processo descrito no algoritmo 2 é repetido P vezes. Num processo de selecção alguns elementos (os mais aptos) serão seleccionados várias vezes, enquanto que outros poderão nunca ser escolhidos. É esta pressão evolutiva que possibilita a convergência gradual do algoritmo para áreas promissoras do espaço de procura. Tal como foi referido anteriormente, o tamanho de torneio ajuda a controlar a pressão selectiva. Em regra, este parâmetro assume um valor entre 2 e 5, sendo raro encontrar situações em que seja superior. Numa situação de torneio binário (tamanho de torneio igual a 2), a pressão selectiva é mínima[5]. Pelo contrário, se o tamanho de torneio for 5, a pressão selectiva já é considerável[6].

No exemplo do KSP que está a ser analisado, é adoptado um torneio binário. A tabela 2.3 ilustra o processo de selecção e indica quais os indivíduos escolhidos como progenitores. Cada linha mostra os dois indivíduos envolvidos num torneio e apresenta o vencedor (o que tem melhor qualidade)

Após a selecção, os progenitores são agrupados em pares, aos quais é aplicado o operador de recombinação. Neste exemplo recorremos ao operador original proposto por Holland (1975), designado **recombinação com um ponto de corte**. De acordo com este operador, os dois indivíduos são alinhados, um ponto de corte é escolhido aleatoriamente ao longo dos cromossomas e os descendentes

[5] O número esperado de cópias do melhor elemento da população actual no conjunto de progenitores é 2 e apenas a pior solução está garantidamente afastada do processo reprodutivo.

[6] Como é evidente, o tamanho da população também influencia a pressão selectiva.

Algoritmo 2 Selecção por torneio com tamanho de torneio k

1: Escolher da população actual k indivíduos aleatoriamente
2: Seleccionar o melhor dos k indivíduos como progenitor

Tabela 2.3: Selecção dos progenitores

Torneio	Participantes	Vencedor
1	$\{I_8(13.0),\ I_3(10.9)\}$	I_8
2	$\{I_1(12.0),\ I_5(17.0)\}$	I_5
3	$\{I_4(6.0),\ I_9(10.3)\}$	I_9
4	$\{I_4(6.0),\ I_{10}(15.0)\}$	I_{10}
5	$\{I_8(13.0),\ I_3(10.9)\}$	I_8
6	$\{I_9(10.3),\ I_{10}(15.0)\}$	I_{10}
7	$\{I_7(10.6),\ I_3(10.9)\}$	I_3
8	$\{I_5(17.0),\ I_2(16.0)\}$	I_5
9	$\{I_7(10.6),\ I_5(17.0)\}$	I_5
10	$\{I_7(10.6),\ I_2(16.0)\}$	I_2

são criados a partir de sequências complementares dos dois progenitores. A figura 2.1 resume o modo de actuação do operador de recombinação com um ponto de corte. Existem outras propostas, como a recombinação com vários pontos de corte ou a recombinação uniforme, que generalizam o modo de actuação deste operador para representações binárias. A descrição de seu funcionamento pode ser consultada em (Eiben e Smith, 2003; Booker et al., 1997). A probabilidade de aplicação da recombinação é um parâmetro do AG e define a probabilidade deste operador ser efectivamente aplicado a cada um dos pares de progenitores. O seu valor é usualmente elevado (valores típicos entre 0.7 e 1.0), uma vez que é o principal responsável pela construção de boas soluções. Nos casos em que o operador de recombinação não é aplicado, os progenitores envolvidos passam para a fase seguinte sem alterações.

Na tabela 2.4 exemplifica-se o efeito da aplicação da recombinação com um ponto de corte aos elementos escolhidos anteriormente. Considera-se que os pares de progenitores são formados de acordo com a ordem pela qual foram seleccionados[7]. A barra vertical identifica o ponto de corte em cada

[7] Deve evitar-se que o operador de recombinação seja aplicado a duas cópias do mesmo indivíduo.

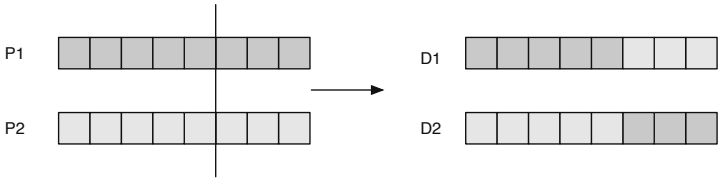


Figura 2.1: Recombinação com um ponto de corte

Tabela 2.4: Aplicação do operador de recombinação com um ponto de corte aos indivíduos seleccionados. A barra vertical identifica o local de corte em cada par de progenitores.

Progenitores	Descendentes
{0 1 0 | 0 0 1 0 0}	{0 1 0 0 1 0 0 0}
{0 1 0 | 0 1 0 0 0}	{0 1 0 0 0 1 0 0}
{1 0 1 0 | 1 1 0 1}	{1 0 1 0 1 0 0 0}
{0 0 0 1 | 1 0 0 0}	{0 0 0 1 1 1 0 1}
{0 1 0 0 0 1 0 0}	{0 1 0 0 0 1 0 0}
{0 0 0 1 1 0 0 0}	{0 0 0 1 1 0 0 0}
{0 1 0 0 0 1 0 | 0}	{0 1 0 0 0 1 0 1}
{0 0 0 1 0 0 1 | 1}	{0 0 0 1 0 0 1 0}
{0 1 0 0 1 0 | 0 0}	{0 1 0 0 1 0 1 0}
{1 0 1 0 0 0 | 1 0}	{1 0 1 0 0 0 0 0}

um dos pares. Neste exemplo assume-se que o operador de recombinação não é aplicado ao terceiro par, pelo que estes progenitores passam sem alteração para a fase seguinte. Pode ainda observar-se um fenómeno curioso na aplicação da recombinação ao primeiro par de soluções: embora os dois cromossomas sejam diferentes, a localização do ponto de corte faz com que os descendentes gerados sejam iguais aos progenitores.

Os indivíduos que resultam da recombinação são sujeitos a mutação. No exemplo que estamos a relatar adoptamos a **mutação binária**, o operador clássico para representações binárias (Holland, 1975; Goldberg, 1989). O seu modo de actuação é extremamente simples: quando aplicada a um gene do cromossoma (i.e., a um constituinte básico da solução que neste caso é um bit), troca o seu valor. A probabilidade de aplicação do operador de mutação é mais um parâmetro do AG, sendo definida em relação a genes individuais, ou seja, define qual a probabilidade de um gene sofrer uma mutação. O seu valor é bastante baixo, sendo vulgar encontrar valores entre 0.001 e 0.01. Tal como foi referido anteriormente, num AG este operador tem como função principal agitar a pesquisa, reintroduzindo diversidade e evitando que esta fique presa em óptimos locais de fraca qualidade.

Na tabela 2.5 exemplifica-se o efeito da aplicação da mutação binária no exemplo que está a ser descrito. Na coluna da esquerda surgem os indivíduos que resultam da recombinação. Os genes sublinhados são sujeitos a mutação, dando origem ao novo conjunto de indivíduos que surge na coluna da direita. Depois da aplicação do operador de mutação, o processo de criação da população da nova geração está concluído. As novas soluções são avaliadas (tabela 2.6) e, caso não tenha sido atingido nenhum dos critérios de terminação que estejam a ser considerados, pode iniciar-se imediatamente uma nova iteração do algoritmo.

Em resumo, a aplicação de um AG a uma instância do KSP obrigou a definir as seguintes componentes:

- Representação para as soluções;
- Função de aptidão;

Tabela 2.5: Aplicação do operador de mutação binária aos indivíduos que resultam da recombinação. Os genes sublinhados foram escolhidos para ser sujeitos a mutação.

Antes da mutação	Depois da mutação
{01$\underline{0}$01000}	{01101000}
{01000100}	{01000100}
{1010$\underline{1}$000}	{10100000}
{00011101}	{00011101}
{0$\underline{1}$0001$\underline{0}$0}	{00000110}
{00011000}	{00011000}
{01000101}	{01000101}
{00010$\underline{0}$10}	{00010110}
{0$\underline{1}\underline{0}$01010}	{00101010}
{1010$\underline{0}$000}	{10101000}

Tabela 2.6: Avaliação dos descendentes

Indivíduos	$\sum V_i$	$\sum P_i$	Penalização	Qualidade
$D_1 = \{01101000\}$	24	43	5.5	18.5
$D_2 = \{01000100\}$	13	29	0	13.0
$D_3 = \{10100000\}$	12	22	0	12.0
$D_4 = \{00011101\}$	23	54	13.2	9.8
$D_5 = \{00000110\}$	9	19	0	9.0
$D_6 = \{00011000\}$	15	27	0	15.0
$D_7 = \{01000101\}$	16	45	6.9	9.1
$D_8 = \{00010110\}$	15	33	0	15.0
$D_9 = \{00101010\}$	20	33	0	20.0
$D_{10} = \{10101000\}$	21	35	0	21.0

- Mecanismo de selecção;
- Operadores genéticos de recombinação e mutação;
- Critério de terminação.

Para além destas componentes, foi ainda necessário especificar valores para os seguintes parâmetros: tamanho da população, tamanho de torneio, probabilidade de aplicação da recombinação, probabilidade de aplicação da mutação e número máximo de gerações (assumindo que é este o critério de terminação adoptado).

Medidas de Desempenho

Quando se aplica um AG a um determinado problema é importante conseguir perceber se ele apresenta um bom desempenho. A avaliação pode ser feita em comparação com outro método de optimização ou pode servir para determinar qual a melhor configuração para o AG que está a ser analisado. Os AGs são métodos de pesquisa estocásticos pelo que as comparações devem ser efectuadas utilizando testes estatísticos[8]. Para que estes testes possam ser utilizados, a aplicação de um AG a um determinado problema deve ser repetida várias vezes. Só assim poderão ser obtidos dados suficientes para tornar a análise estatística significativa. O número de repetições de um AG é um parâmetro adicional que deve ser especificado[9].

As medidas mais utilizadas para aferir o desempenho de um AG são a eficiência e a eficácia. A primeira está directamente relacionada com a sua rapidez, enquando que a segunda mede a qualidade das soluções. Há três critérios comuns para estimar a qualidade:

- **Taxa de sucesso**: Se a qualidade da solução óptima for conhecida, pode contabilizar-se quantas repetições do AG a conseguem descobrir. A razão entre este valor e o total de repetições corresponde à taxa de sucesso do algoritmo.
- **Média das melhores soluções**[10]: Para cada uma das repetições realizadas determina-se a qualidade da melhor solução. Esta medida é obtida efectuando a média destes valores.
- **Melhor solução encontrada**[11]: Qualidade da melhor solução encontrada pelo AG.

Os três critérios descritos estão evidentemente relacionados. Além disso, dependendo do problema concreto e do objectivo da optimização, pode fazer mais sentido utilizar uma ou outra medida. Em (Eiben e Smith, 2003, cap. 14) pode ser consultada uma descrição detalhada das vantagens e limitações de cada um destes critérios.

4. Propriedades dos Algoritmos Genéticos

Apesar da sua simplicidade, os AGs possuem um conjunto de propriedades que os tornam especialmente robustos para efectuar pesquisas em espaços de grande dimensão e complexidade. Três das principais características que contribuem para esta situação são:

- Processamento simultâneo de uma população de potenciais soluções para o problema. Esta particularidade permite que diferentes áreas do espaço de procura sejam analisadas em paralelo, reduzindo a probabilidade de convergência prematura para um óptimo local.

[8] A apresentação dos testes existentes e das condições em que podem ser aplicados está fora do âmbito deste capítulo. Consultar (Zar, 2009) para uma apresentação detalhada deste tópico.

[9] Embora existam excepções, 30 repetições é um valor padrão normalmente aceite pela comunidade da computação evolutiva.

[10] Do inglês *mean best fitness*.

[11] Do inglês, *best-ever fitness*.

- Possuem componentes probabilísticas e não recorrem a informação específica sobre o problema que estão a resolver (a única informação de que necessitam é o resultado da aplicação da função de aptidão). Estes dois factores, combinados com a existência de uma população de soluções, tornam estes métodos particularmente versáteis e robustos.

- Equilíbrio entre exploração de novas regiões do espaço e conservação da informação já adquirida: a selecção actua como principal força de conservação, permitindo manter e consolidar a informação descoberta anteriormente. Por seu lado, através de transformações efectuadas nos elementos seleccionados, é promovida a exploração de novas regiões do espaço. Assumindo que este possui algum tipo de regularidade, as referidas transformações poderão conduzir à descoberta de novas regiões, onde se encontrem soluções de qualidade superior.

Fundamentação Teórica

Na sua proposta, Holland analisou teoricamente o desempenho dos AGs como estratégia de pesquisa. A descrição rigorosa da análise efectuada está fora do âmbito deste texto, aconselhando-se a consulta das obras (Holland, 1975; Goldberg, 1989) para um estudo mais detalhado. Neste ponto limitamo-nos a descrever de forma simplificada algumas das ideias principais. Toda a análise teórica definida por Holland é sustentada por dois conceitos básicos: a noção de **esquema** e a noção de **blocos construtores**. Um esquema é um modelo representativo de um determinado subconjunto de cromossomas que partilham os mesmos valores em algumas posições. Mais concretamente, e limitando a análise a alfabetos binários, um esquema é uma sequência constituída pelos símbolos $\{0, 1, *\}$. Posições contendo 0 ou 1 são consideradas definidas, enquanto que as que contêm o símbolo $*$ são consideradas irrelevantes para a análise[12]. A utilização de esquemas permite representar um subconjunto do espaço de procura de forma muito eficiente. Um cromossoma *ch* pertence a um determinado esquema E se e só se as sequências *ch* e E forem iguais em todas as posições em que E não contém um $*$. Considere-se o seguinte exemplo: o esquema $E = \{1 * * * 1\}$ representa todos os cromossomas binários com 5 genes que possuam um bit com valor 1 na primeira e na última posição. As sequências que se encaixam no esquema (por exemplo, 10001 ou 10101) designam-se instâncias de E. Dois atributos dos esquemas são a sua **ordem** (número posições definidas) e o seu **tamanho definido** (distância entre a primeira e última posições definidas). Analisando o esquema E, verifica-se que é de ordem 2 e que o seu tamanho definido é 4.

Cada indivíduo da população representa um número elevado de esquemas (exactamente 2^n, sendo n o número de genes do cromossoma). Como consequência, numa determinada geração, apesar de o AG apenas avaliar explicitamente os indivíduos que pertencem à população, está implicitamente a estimar a qualidade de um número muito elevado de esquemas. A qualidade de um esquema é definida como a qualidade média de todas as suas instâncias que pertencem à população. Segundo Holland (1975), é a existência deste paralelismo implícito que permite a um AG efectuar de forma eficiente a pesquisa de um espaço de procura de grande dimensão. O teorema fundamental dos AGs, também chamado **teorema dos esquemas**[13], enuncia o modo como a pesquisa é efectuada: "O número de instâncias de esquemas de ordem baixa, com tamanho definido reduzido e com qualidade acima da média aumenta de forma exponencial ao longo das sucessivas gerações de um AG". Por sua vez, a **hipótese dos blocos construtores** é uma consequência lógica do teorema anterior: "Um AG efectua a pesquisa de soluções de boa qualidade através da combinação de esquemas de pequena dimensão (i.e., ordem baixa e tamanho definido reduzido) e de qualidade acima da média. Estes esquemas são denominados os blocos construtores da solução".

A generalização da utilização de AGs conduziu a novas tentativas no sentido de entender o seu modo de funcionamento. Uma das questões centrais associadas à análise do desempenho de um AG

[12] Do inglês, *don't care symbol*.
[13] Do inglês: *schemata theorem*.

é a seguinte: "Em quais problemas é suposto um AG ser eficaz?". A consulta dos volumes da série *Foundations of Genetic Algorithms* é uma boa introdução para se obter uma ideia dos desenvolvimentos teóricos recentes desta área.

5. Extensões ao Algoritmo Genético Simples

O AG descrito na secção 3 possui as componentes básicas originalmente propostas por Holland (1975), sendo habitualmente identificado como AG simples ou AG canónico. A única excepção é o mecanismo de selecção adoptado. Uma consequência lógica da disseminação e popularização desta técnica é a introdução de alterações à proposta original. Ao longo dos anos, diversas modificações foram introduzidas na sua estrutura, tanto ao nível da representação e operadores utilizados, como ao nível da própria arquitectura do AG. Actualmente, as opções de implementação estão, na maioria das situações, dependentes do problema que se pretende resolver. Alguns exemplos de representações alternativas incluem a utilização de vectores de números reais, permutações, tabelas multidimensionais, sistemas de regras ou grafos. Uma das consequências desta situação é a necessidade de desenvolver operadores genéticos que se adaptem às novas representações (Eiben e Smith, 2003; Michalewicz, 1992). Nesta secção apresentamos alguns exemplos de alternativas ao algoritmo canónico.

Representação e Operadores Genéticos

A selecção da representação a adoptar para as soluções é crucial. Esta é provavelmente a decisão mais importante a tomar quando se efectua o desenvolvimento de um AG para aplicar num problema de optimização, até porque várias outras componentes (como, por exemplo, os operadores genéticos ou a função de aptidão) dependem desta decisão.

Alguns factores que é necessário considerar são:

- O alfabeto deve ser binário ou pode conter mais símbolos?
- A representação deve ser discreta ou contínua?
- O ordenamento dos genes no cromossoma é relevante?
- O cromossoma pode conter informação redundante?
- A representação permite a existência de soluções inválidas? Em caso afirmativo, como lidar com elas?

As decisões a tomar em relação aos tópicos enunciados estão directamente relacionadas com o problema a optimizar. Encontrar a representação apropriada para um problema, ou conjunto de problemas, é uma tarefa difícil e é um tópico importante de investigação. Quando confrontados com um caso concreto, a experiência acumulada na utilização de AGs e uma análise cuidada do problema a resolver ajudam a tomar uma decisão sobre qual poderá ser a melhor opção. Além disso, a realização de um conjunto preliminar de testes com várias alternativas pode ajudar a fundamentar melhor a escolha efectuada.

Para além da representação binária, a representação real e a baseada em permutações são as 2 alternativas que são adoptadas mais vezes por AGs. Nas próximas secções descrevemos brevemente as suas propriedades e apresentamos propostas de operadores genéticos para lidar com cada uma delas.

Representação real

Os problemas nos quais as variáveis de decisão tomam valores reais são muito comuns. Nestes casos, uma solução é representada através de uma sequência de números reais. Embora seja possível recorrer a uma sequência binária para a codificação[14], é mais natural adoptar uma representação real em que cada gene corresponde directamente uma variável de decisão. Cada um destes genes tem associado o seu próprio domínio. Quando se lida com problemas em domínios contínuos, esta representação tem a vantagem de ser mais natural, ou seja, está mais próxima das soluções que representa. Esta é uma característica importante que deve ser levada em consideração quando se está a decidir qual a representação mais apropriada para um dado problema.

Considere que se pretende minimizar a função apresentada na equação 2.6.

$$f(x_1, x_2) = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2 \qquad -2.048 \leq x_1, x_2 \leq 2.048 \tag{2.6}$$

Para este problema, a representação real de uma solução é feita recorrendo a 2 valores reais, respectivamente para as variáveis x_1 e x_2. Neste caso, uma solução possível será: $\{-1.235, 0.023\}$. O número de casas decimais a adoptar depende da precisão que se pretende, embora esteja obviamente condicionada pelas limitações físicas associadas à implementação computacional.

Os operadores de recombinação existentes para representações reais dividem-se em dois grupos: recombinação discreta vs. recombinação aritmética. Os do primeiro grupo actuam de forma análoga aos operadores de recombinação binários: determinam um ou mais pontos de corte entre genes e criam descendentes juntando secções complementares dos progenitores. A principal desvantagem dos operadores discretos é a incapacidade que têm para introduzir novos valores na população, garantindo apenas novas combinações para alelos já existentes.

Os operadores de recombinação aritméticos permitem ultrapassar esta limitação. Com estes operadores, o valor de cada um dos genes dos descendentes é encontrado dentro do intervalo definido pelos respectivos alelos que surgem nos progenitores[15]. Na equação 2.7 apresentamos um exemplo de aplicação de um operador de recombinação aritmético. Dados dois progenitores $P1$ e $P2$, os descendentes $D1$ e $D2$ são criados de acordo com as seguintes expressões:

$$\begin{cases} D_1 = \lambda \times P1 + (1 - \lambda) \times P2 \\ D_2 = \lambda \times P2 + (1 - \lambda) \times P1 \end{cases} \tag{2.7}$$

Na equação $\lambda \in [0, 1]$, o que significa que este operador efectua uma média pesada dos valores dos progenitores para gerar os descendentes.

Em traços gerais existem dois tipos de operadores de mutação para representações reais. Na mutação uniforme, o novo valor é escolhido aleatoriamente do domínio do gene em questão. Pelo contrário, na mutação não-uniforme o valor actual do gene é tido em consideração e é alterado de uma forma mais limitada. A mutação gaussiana é o exemplo mais comum, sendo que neste caso o novo valor para um gene G_i é obtido através da seguinte expressão $G_i \leftarrow G_i + N(0, 1)$.

Representação em permutação

Para apresentar as propriedades das representações baseadas em permutações regressamos ao exemplo do KSP. No que diz respeito a este problema, para além da representação binária, várias outras propostas têm sido apresentadas e analisadas. Em (Hinterding, 1999; Raidl e Gottlieb, 2005) pode

[14] Em (Michalewicz, 1992) pode ser consultado um exemplo detalhado de todos os passos envolvidos neste processo. Ver ainda (Herrera et al., 1998) para um discussão das vantagens/desvantagens da utilização de representações binárias em problemas contínuos.

[15] Vários operadores de recombinação relaxam esta restrição e possibilitam que o novo valor saia ligeiramente do intervalo (Herrera et al., 1998). Esta modificação pode ajudar a combater a convergência prematura do algoritmo.

Tabela 2.7: Descodificação da solução $\{1, 8, 7, 4, 2, 6, 3, 5\}$ utilizando a heurística *first-fit*. O símbolo $\surd$ indica que o objecto foi seleccionado para a mochila.

	1	8	7	4	2	6	3	5
Mochila	$\surd$	$\surd$	$\surd$	X	X	$\surd$	X	X
Peso Acumulado	10	16	24	24	24	35	35	35

ser consultada uma lista abrangente das várias representações. Uma das mais estudadas é a representação baseada numa permutação de valores, originalmente proposta por Hinterding (1999). As permutações são representações habitualmente utilizadas em problemas em que é necessário encontrar uma sequência ideal para um conjunto de objectos. Entre os exemplos mais comuns, incluem-se o problema do caixeiro viajante, o problema do encaminhamento de veículos ou problemas de escalonamento.

As representações em permutação codificam uma sequência ordenada de símbolos, na qual não podem existir repetições. A ordem pela qual os símbolos surgem na representação e/ou as relações de adjacência são as características mais relevantes para definir a qualidade de uma solução. No caso do KSP, a representação consiste numa permutação de todos objectos que fazem parte do problema que está a ser optimizado. Considerando o exemplo com 8 objectos apresentado na secção 3, duas soluções possíveis são $\{1, 4, 7, 8, 2, 6, 3, 5\}$ ou $\{5, 2, 4, 1, 8, 7, 3, 6\}$[16].

A representação em permutação proposta para o KSP designa-se **indirecta**, uma vez que a consulta de uma solução não especifica imediatamente quais os objectos que se encontram na mochila. É necessário aplicar um algoritmo de descodificação para interpretar a informação armazenada no cromossoma e indicar qual a solução final. A ordem pela qual os objectos surgem no cromossoma representa a sua prioridade no que diz respeito à entrada na mochila. Uma heurística simples do tipo *first-fit* considera os objectos pela ordem em que se encontram na solução e adiciona-os, se eles não provocarem a violação da restrição de capacidade. Considerando a permutação $\{1, 8, 7, 4, 2, 6, 3, 5\}$ e as propriedades indicadas na tabela 2.1, na tabela 2.7 exemplifica-se o processo de descodificação. A solução final obtida indica que os objectos $\{1, 8, 7$ e $6\}$ foram seleccionados para entrar na mochila.

Existem vantagens e desvantagens associadas à adoptação de uma representação baseada em permutações para resolver o problema do KSP. Aliás, a um nível mais geral, existem sempre pontos fortes e pontos fracos associados a cada representação que pode ser considerada para um determinado problema. Estes devem ser identificados e analisados, uma vez que isto ajudará a fazer uma escolha mais fundamentada pela representação que se acredita ser mais adequada. Regressando à proposta baseada em permutações para o KSP, podem ser identificadas as seguintes vantagens/desvantagens[17]:

- **Vantagens**:
  - A representação em permutação garante que as soluções processadas pelo AG são sempre válidas. A existência de um algoritmo de descodificação assegura que a restrição de capacidade da mochila nunca é violada.
  - Uma vez que o AG apenas processa soluções válidas, é mais simples implementar a função de aptidão. Não é necessário utilizar nenhum mecanismo de penalização para punir indivíduos

[16] Para simplificar a notação, cada objecto é representado apenas pelo seu identificador numérico.

[17] Embora a análise seja feita para um problema particular, praticamente todos os tópicos são válidos para situações em que se opta por uma representação indirecta

Tabela 2.8: Aplicação do operador de recombinação com um ponto de corte a permutações. A barra vertical identifica o local de corte em cada par de progenitores.

Progenitores	Descendentes
{3 5 4 2 | 1 7 9 8 6}	{3 5 4 2 4 3 8 6 9}
{5 1 2 7 | 4 3 8 6 9}	{5 1 2 7 1 7 9 8 6}

ilegais. A comparação entre soluções também se torna mais clara, uma vez que o único critério a considerar é o valor total proporcionado pelos objectos seleccionados.

- **Desvantagens**:
  - A principal desvantagem associada à utilização de uma representação indirecta está directamente relacionada com a necessidade de desenvolver e aplicar um algoritmo de descodificação. Em muitos casos não é trivial encontrar um algoritmo que permita descodificar o cromossoma numa solução final legal. Esta situação não se verifica no problema do KSP que está a ser utilizado como exemplo, mas é comum em muitos problemas de optimização.
  - O algoritmo de descodificação é específico do problema e da representação, o que diminui a generalidade do AG. Além disso, e embora se pretenda que o algoritmo de descodificação seja o mais simples possível, existe sempre um tempo adicional de processamento que pode condicionar a eficiência do processo de optimização.

O modo de actuar dos operadores genéticos depende da representação adoptada. É fácil verificar que a aplicação da recombinação com um ponto de corte e da mutação binária a soluções codificadas como permutações gera indivíduos ilegais (ilegais, no sentido de não pertencerem ao espaço de procura)[18]. Na tabela 2.8 ilustra-se como a aplicação da recombinação com um ponto de corte pode originar soluções ilegais.

Na literatura existem várias propostas de operadores genéticos para permutações. Tal como referimos anteriormente, esta é uma representação muito utilizada em alguns dos mais relevantes problemas de optimização que ocorrem no mundo real, o que naturalmente conduziu ao surgimento de diversas alternativas. Um operador de recombinação deve ter a capacidade de, dadas duas permutações, gerar dois descendentes tais que: i) são permutações legais (i.e., contêm todos os símbolos uma e uma só vez); ii) são constituídos por informação relevante (i.e., blocos construtores) presente em cada um dos progenitores. No caso de uma permutação, a informação relevante inclui a posição absoluta ocupada por cada um dos símbolos e as relações de precedência e de adjacência que se verificam. Dependendo do tipo de problema que está a ser resolvido, pode ser mais importante considerar a posição absoluta dos símbolos como, por exemplo, em problemas de escalonamento ou as relações de adjacência, como no caso do caixeiro viajante. Alguns dos exemplos mais conhecidos de operadores são a recombinação por ordem (*Order crossover*), recombinação por ciclo (*Cycle crossover*), recombinação por aresta (*Edge crossover*) ou o PMX (*Partially mapped crossover*). Em (Eiben e Smith, 2003; Michalewicz, 1992) pode ser consultado o funcionamento detalhado de cada um destes operadores e uma lista de problemas para os quais são mais apropriados.

Neste capítulo apresentamos o operador de recombinação por ordem, originalmente proposto por Davis (1991). Este operador é especialmente apropriado para lidar com situações em que a informação

[18] Em sentido estrito, a mutação binária não pode ser aplicada a uma representação contendo símbolos inteiros. No entanto pode ser facilmente extendida para um operador que troque o valor de um gene para outro pertencente ao domínio.

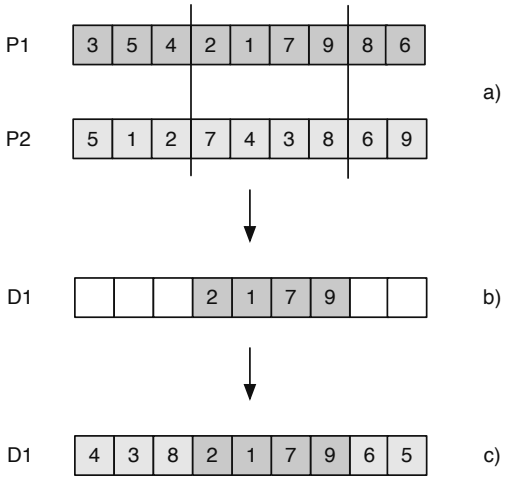


Figura 2.2: Recombinação por ordem

mais importante a transmitir aos descendentes é a ordem pela qual os símbolos aparecem nos progenitores. No algoritmo 3 podem ser consultados os passos executados por este operador para criar dois descendentes D1 e D2 a partir dos progenitores P1 e P2. A figura 2.2 apresenta um exemplo de aplicação deste operador. O painel a) da figura apresenta os dois progenitores e os locais dos pontos de corte. No painel b) pode ver-se a secção de P1 que é herdada por D1 e, finalmente, no painel c) surge o descendente já com os símbolos herdados de P2. A geração do descendente D2 fica como exercício para o leitor.

Algoritmo 3 Operador de recombinação por ordem. Gera dois descendentes D1 e D2 a partir dos progenitores P1 e P2.

1: Alinhar as sequências P1 e P2
2: Escolher aleatoriamente dois pontos de corte nas sequências P1 e P2. Os pontos de corte situam-se entre dois símbolos
3: Copiar a secção entre os pontos de corte de P1 para D1
4: Com início na posição a seguir ao segundo ponto de corte, copiar os restantes símbolos de P2 para D1. Os dois cromossomas P2 e D1 devem ser considerados circulares
5: Criar D2 repetindo os dois passos anteriores, mas trocando o papel desempenhado pelos progenitores P1 e P2.

Devido às propriedades das permutações, a aplicação da mutação não pode considerar genes individualmente. A alternativa é escolher subconjuntos de genes e alterar as suas posições. Existem vários operadores de mutação padronizados que são habitualmente aplicados a permutações: mutação por troca (*swap mutation*), mutação por inserção (*insert mutation*), mutação por deslocamento (*displacement mutation*) ou mutação por inversão (*inversion mutation*). Para uma descrição completa e detalhada do modo de funcionamento destes operadores consultar (Eiben e Smith, 2003; Michalewicz, 1992). Neste capítulo apresentamos dois exemplos simples que ilustram o modo de funcionamento de dois destes operadores. No painel a) da figura 2.3 apresentamos um exemplo de aplicação da mutação por troca: dois genes são escolhidos aleatoriamente no cromossoma e os seus valores são trocados; no painel b) da mesma figura, a mutação por inversão escolhe dois genes aleatoriamente e inverte a

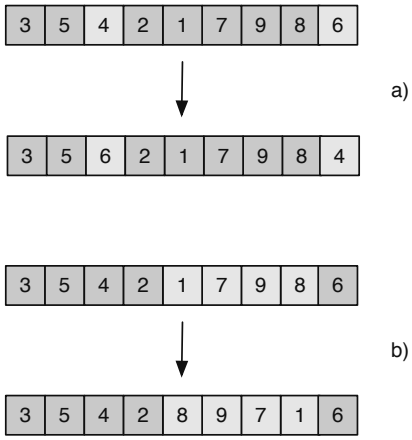


Figura 2.3: Exemplos de operadores de mutação para permutações: a) mutação por troca; b) mutação por inversão

ordem dos valores que se encontram entre essas duas posições.

Algoritmos de Reparação

Antes de propor novas formas de organização para o modo como os AGs processam soluções, é importante regressar por um momento à representação binária proposta para o KSP. Foram referidas na secção 3, três alternativas para lidar com soluções inválidas. Duas delas, penalização e desenvolvimento de representações alternativas, foram já discutidas. É altura de abordar brevemente a reparação de soluções. Esta abordagem possui vários pontos de contacto com a utilização de representações indirectas descrita anteriormente. Tal como neste caso existe uma heurística simples, neste caso designada algoritmo de correcção, que actua sobre as soluções inválidas da população, corrigindo-as[19]. O objectivo é garantir que todos os indivíduos avaliados são soluções legais. A definição de um algoritmo de correcção pode ser uma tarefa complicada. Em vários casos, poderá ser (quase) tão difícil corrigir uma solução inválida, como resolver o problema que se está a optimizar. Um caso paradigmático é o problema da construção dos horários de uma escola (Lewis, 2008). Devido a todas as restrições envolvidas, corrigir uma solução ilegal pode ser extremamente difícil.

No caso do KSP é trivial corrigir um indivíduo. Uma solução é ilegal porque a capacidade da mochila foi ultrapassada, logo, a correcção deve ir retirando objectos até que o problema esteja ultrapassado. O algoritmo 4 ilustra os passos a executar para corrigir a solução S. Embora este algoritmo não especifique como devem ser escolhidos os objectos a retirar da mochila, existem duas possibilidades: retirar os objectos de acordo com uma heurística (por exemplo, o objecto mais pesado ou o objecto com pior *ratio* entre valor e peso) ou retirá-los aleatoriamente. A primeira opção parece ser mais apropriada, mas a segunda hipótese não deve ser imediatamente descartada. Proporciona maior robustez e evita introduzir componentes sôfregas na pesquisa que poderão comprometer a eficácia global do algoritmo.

[19] Em traços gerais, as vantagens e desvantagens associadas à utilização de uma representação indirecta mantêm-se válidas.

Algoritmo 4 Reparação de uma representação binária para o KSP

1: **enquanto** $(\sum_{i=1}^{N} x_i \times P_i > C)$ **faça**
2: Escolher uma posição x_i com valor 1
3: $x_i \leftarrow 0$
4: **fim enquanto**

Arquitecturas alternativas

Num AG simples, as populações sucedem-se iteração após iteração sem que exista nenhum tipo de competição entre progenitores e descendentes. De acordo com esta organização, designada **geracional**, assim que termina a criação da nova população, a anterior desaparece. Ao longo dos anos, diversas alternativas de processamento das populações foram apresentadas com o objectivo de aumentar a eficácia dos AGs. Nas próximas secções descrevemos brevemente duas dessas propostas.

Elitismo

A qualidade média dos indivíduos que fazem parte das populações processadas por um AG aumenta ao longo das gerações. Este aumento é mais pronunciado no início da optimização e tem tendência para diminuir ou estagnar à medida que a população começa a convergir.

Pelo contrário, é relativamente comum que a qualidade da melhor solução diminua de uma geração para outra. O desaparecimento do indivíduo de melhor qualidade ocorre devido ao caracter probabilístico da selecção ou, mais provavelmente, devido à acção dos operadores genéticos. O **elitismo** evita que isto aconteça. O funcionamento desta estratégia é muito simples de descrever: quando termina a avaliação da população de descendentes verifica-se qual é a qualidade do melhor indivíduo. Se for inferior à qualidade da melhor solução da geração anterior, esta é recuperada e inserida na nova população. Como a população tem tamanho fixo, a adição de uma solução obriga a eliminar um dos descendentes que acabou de ser criado. A escolha do elemento sacrificado pode ser feita de forma aleatória ou de acordo com um critério pré-estabelecido (por exemplo, eliminar a solução de pior qualidade).

Algoritmos *steady-state*

Nos AGs do tipo *regime permanente*[20] deixa de existir a noção de gerações bem definidas em que os indivíduos são completamente substituídos em cada iteração (Whitley, 1989). Pelo contrário, os progenitores e descendentes passam a coexistir e a competir por um lugar na população. O primeiro autor a estudar esta possibilidade em AGs foi De Jong em 1975 (De Jong, 1975). No seu estudo definiu um parâmetro G (designado *generation gap*) que estabelece a fracção da população que é substituída em cada geração. O seu valor pode variar entre $G = 1$, correspondendo a uma organização geracional, até $G = 1/P$ (em que P é o tamanho da população), no qual apenas um indivíduo é substituído em cada iteração.

O algoritmo 5 ilustra o modo de funcionamento de um AG *regime permanente* para $G = 1/P$. Vários detalhes de implementação devem ser clarificados em relação ao funcionamento deste algoritmo (consultar (Lozano et al., 2004; Smith, 2007) para obter detalhes sobre as várias alternativas) :

- Selecção: o processo de selecção dos progenitores pode ser efectuado com um qualquer mecanismo de selecção (por exemplo, selecção por torneio), embora seja relativamente comum escolher as soluções aleatoriamente.

[20] Do inglês *steady-state*.

- Operadores genéticos: embora a recombinação crie habitualmente duas soluções, o algoritmo apresentado lida apenas com um descendente. Neste passo pode assumir-se que é escolhido o descendente de melhor qualidade (ou mesmo que a escolha é feita de forma aleatória).
- Domínio de substituição: existem duas possibilidades para seleccionar a solução Z que poderá ser substituída pelo descendente. A primeira restringe a escolha aos progenitores. A segunda alarga a escolha a toda a população ou a um subconjunto da população.
- Estratégia de substituição: Independentemente do domínio considerado, a escolha da solução Z a substituir pode ser feita de acordo com vários critérios. De entre as possibilidades, destacam-se a substituição por qualidade, por idade (i.e., a solução que está há mais tempo na população), por semelhança[21] ou mesmo aleatoriamente. É possível ainda considerar uma combinação de critérios para encontrar a solução Z a substituir.
- Condição de substituição: No último passo do algoritmo, a decisão sobre a entrada do descendente D na população está normalmente relacionada com a sua qualidade, ou seja, a solução é aceite se tiver melhor qualidade do que o indivíduo Z escolhido para ser substituído. Uma alternativa a esta abordagem é efectuar a substituição incondicional.

As regras genéricas enunciadas em cima permitem adoptar uma grande variedade de estratégias. Uma das mais conhecidas é a *replace worst*, proposta por Whitley no seu sistema GENITOR (Whitley, 1989) que advoga que a solução Z é a de pior qualidade na população e a que a condição de substituição é baseada no mérito, ou seja, o descendente entra população desde que tenha melhor qualidade do que o pior indivíduo que lá se encontra. Embora seja simples e intuitiva, esta estratégia provoca uma rápida diminuição de diversidade, levando o algoritmo a convergir prematuramente. Por este motivo é normalmente preferível adoptar um mecanismo que, para além da qualidade, considere medidas de semelhança entre indivíduos, de modo a manter níveis de diversidade adequados dentro da população.

Algoritmo 5 Algoritmo Genético em Regime Permanente (Steady-State)

1: Seleccionar dois progenitores P1 e P2 da população actual
2: Criar um descendente D através da aplicação de operadores genéticos
3: Avaliar o desdendente D
4: Seleccionar uma solução Z na população para substituição
5: Decidir se o descendente D substitui a solução Z

6. Aplicações Práticas

Os AGs são actualmente um método de resolução de problemas muito utilizado, tendo demonstrado a sua eficácia nos mais diferentes domínios, tanto científicos como em situações do mundo real. A seguir apresentamos alguns exemplos de áreas nas quais os AGs têm sido aplicados.

- **Optimização numérica**: Esta categoria inclui muitos problemas de engenharia que podem ser modelados através de um conjunto de parâmetros reais para os quais é necessário encontrar a melhor configuração. O que caracteriza esta classe é o facto de os domínios associados a cada parâmetro serem contínuos. Dois exemplos de áreas nas quais podem surgir este tipo de problemas são o controlo (Fleming e Purshouse, 2002) ou a química teórica e computacional (Johnston, 2004).

[21] Existem diversas medidas que procuram aferir a semelhança entre duas soluções para um dado problema. São normalmente métricas específicas de uma dada representação e/ou problema e procuram determinar até que ponto duas soluções partilham características idênticas.

- **Optimização combinatória**: Os problemas desta classe são, provavelmente, as situações de optimização às quais os AGs são aplicados com maior frequência. O que os distingue dos problemas do tópico anterior é que, neste caso, as variáveis de decisão são discretas. Existem inúmeras situações do mundo real que podem ser aproximadas por problemas de optimização combinatória, o que justifica a atenção da comunidade científica. Alguns dos exemplos mais relevantes são o problema do caixeiro viajante, o problema do encaminhamento de veículos ou problemas de escalonamento (Jensen, 2004; Lewis, 2008; Pereira e Tavares, 2009; Tsai et al., 2004).

- **Robótica evolutiva**: Nesta área recorre-se a algoritmos de inspiração biológica para evoluir a morfologia e comportamentos de robots (Floreano e Nolfi, 2004).

- **Simulação**: A capacidade de proceder à evolução/adaptação de um conjunto de entidades permite que os AGs sejam usados no desenvolvimento e estudo de modelos de simulação pertencentes a diversos domínios. Alguns exemplos relevantes são a economia (estudo da emergência de mercados económicos), o sistema imunitário (estudo das mutações somáticas), a ecologia (estudo de fenómenos do tipo hospedeiro/parasita) ou os sistemas sociais (estudo da evolução do comportamento social em colónias de insectos) (Mitchell, 1996).

A lista apresentada não pretende, de modo nenhum, ser exaustiva. A consulta de actas das conferências mais relevantes nesta área, a *Genetic and Evolutionary Computation Conference (GECCO)* [22] ou o *IEEE Congress on Evolutionary Computation (CEC)*[23] entre outras, pode ser útil para se obter uma ideia geral dos desenvolvimentos recentes e para ajudar a perceber quais as áreas de aplicação preferenciais para os AGs.

Programação Genética

Uma das aplicações mais interessantes de algoritmos de inspiração biológica é a possibilidade de evoluir programas de computador que resolvam um determinado problema. A **programação genética** pode ser considerada uma descendente directa dos AGs. Embora tenham existido algumas tentativas anteriores para desenvolver um sistema com estas características, o nascimento da área é normalmente associado à publicação, em 1992, do livro *Genetic Programming* de Koza (1992). A questão central para a qual este trabalho procura obter respostas é a seguinte: 'De que forma pode um computador aprender a resolver um determinado problema sem ser explicitamente programado para isso?"

Um algoritmo de programação genética evolui um conjunto de programas de computador ao longo de várias gerações. O objectivo é encontrar uma solução (i.e., um programa) que resolva um determinado problema. Tal como num AG, é necessário definir uma representação para as soluções, uma função de aptidão, um mecanismo de seleçcão e operadores genéticos. As abordagens iniciais recorreram a árvores para representar programas de computador. Actualmente existem diversas representações alternativas como, por exemplo, as lineares ou baseadas em grafos. O livro *A Field Guide to Genetic Programming* de Poli et al. (2008b) apresenta uma introdução clara e completa da área da programação genética, incluíndo uma descrição de aplicações práticas. Por sua vez, na página *Human Competitive Results*[24] é possível consultar um conjunto soluções obtidas por programação genética que, de acordo com um conjunto de critérios, são comparáveis ou melhores do que soluções propostas por humanos.

[22] http://www.sigevo.org/
[23] http://www.ieee-cis.org/
[24] http://www.genetic-programming.com/humancompetitive.html

7. Conclusões

Os AGs são métodos de resolução de problemas inspirados em modelos simplificados da evolução natural. O seu modo de actuação é extremamente simples de enunciar: um conjunto inicial de possíveis soluções para um determinado problema (a população inicial) vai sendo sucessivamente modificado através da aplicação de operadores inspirados nos mecanismos da selecção natural e da variação genética (i.e., o conjunto de soluções vai evoluindo ao longo de sucessivas gerações). Neste capítulo descrevemos as componentes básicas que constituem um AG, nomeadamente o método de selecção de progenitores e os operadores de transformação que dão origem a novas soluções. Identificámos ainda a definição da representação e da função de aptidão como duas decisões cruciais para o sucesso da optimização. Para tornar a exposição o mais clara possível, descrevemos, passo a passo, a aplicação de um AG a uma instância do KSP.

Uma das características que torna os AGs especialmente apelativos é a simplicidade do seu funcionamento. Considerando a metáfora biológica como ponto de partida, é trivial entender e implementar as etapas associadas ao processamento iterativo da população de soluções. Esta simplicidade está relacionada com o facto de um AG simples ser um método de optimização global. A aplicação a um determinado problema não exige conhecimento particular sobre a situação em causa, sendo suficiente definir um critério de qualidade que avalie o mérito relativo das soluções que são geradas. Esta característica torna os AGs especialmente robustos e com capacidade para lidar com espaços de procura de grande dimensão, irregulares, descontínuos e multimodais. É importante, no entanto, referir que estas características não inviabilizam que um AG possa ser combinado com um algoritmo que possua propriedades complementares. Pelo contrário, abordagens híbridas que combinem as características globais de um AG com propriedades mais locais e/ou especializadas de outros métodos podem ser extremamente eficazes na resolução de problemas difíceis. Este tópico será abordado num dos capítulos deste manual.