Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão

Algoritmos Genéticos

uma (breve) introdução

João Paulo Ramos e Barros

Nota prévia:
Este texto corresponde a uma versão reduzida e revista de parte de um trabalho não publicado intitulado: "Optimização com Algoritmos Genéticos em problemas NP-Hard: o problema do Transporte e o problema do Horário". Esse trabalho foi realizado em Janeiro de 1994, em co-autoria com João Pedro Guerreiro Neto, actualmente Assistente na Faculdade de Ciências da Universidade de Lisboa. Obviamente qualquer falha ou incorrecção do presente texto são da minha exclusiva responsabilidade.

João Paulo Barros
Beja, 22 de Abril de 1997

O QUE SÃO ALGORITMOS GENÉTICOS?

Muitos dos problemas para os quais não de conhecem algoritmos eficientes, são problemas de optimização. É muitas vezes possível encontrar algoritmos mais eficientes (no sentido de uma maior rapidez de execução) que embora não garantindo uma solução óptima, mas apenas próxima, permitem alcançá-la num período de tempo aceitável. Muitos problemas bem conhecidos foram já resolvidos utilizando tal tipo de algoritmos (um dos exemplos bem conhecido é o problema do viajante). Exemplos típicos dessa classe de algoritmos são o *Hill Climbing* e uma sua variante, usualmente denominada *Simulated Annealing* [Michalewicz, Z., 92]

Os algoritmos genéticos (AGs), pela primeira vez especificados na década de setenta por John Holland, são uma das técnicas de inteligência artificial, utilizadas como alternativa à busca exaustiva que se pode tornar intratável para espaços, de potenciais soluções, de grande dimensão. Os problemas de optimização podem ser vistos como buscas em espaços de soluções. Nesse caso considera-se a busca como a procura da melhor solução. Os problemas de optimização são precisamente um dos campos mais explorados na aplicação de AGs. Estes são algoritmos que buscam de forma aleatória e utilizando um modelo baseado em fenómenos naturais, donde o seu nome, uma solução óptima num espaço de potenciais soluções [Spillman, 93].

Os AGs pretendem modelar dois tipos de fenómeno natural: a teoria da evolução de Darwin e as leis da herança genética descobertas por Mendel em meados do século XIX. Assim sendo, muita da sua nomenclatura é baseada na da Biologia. Nomes como cromossomas, genes, alelos, populações, indivíduos, reprodução, etc., são usualmente utilizados na literatura. Muitas vezes, os nomes gene, indivíduo e cromossoma são referidos como constituindo uma mesma entidade. Biologicamente, são coisas distintas. De facto, um indivíduo (ser vivo) contem tipicamente vários cromossomas, cada um dos quais com diversos genes. Estas distinções não são, no entanto, grandemente importantes para a nossa discussão. De certa forma e neste caso, torna-se pois indiferente a nomenclatura utilizada desde que consistente. Optou-se pela designação cromossoma para referir cada elemento da população. Ou seja, cada individuo é uni-cromossómico. Ocasionalmente, e sempre que tal faça sentido, poderemos utilizar o termo gene como designação de cada um dos elementos constituintes de cada cromossoma. Ao estado do gene (valor) chamaremos *alelo*.

Basicamente, um AG mantem uma população de cromossomas (soluções) que evolui e se modifica devido aos cruzamentos entre estes e, em mais pequena parte, devido às modificações induzidas pelo meio-ambiente (mutações genéticas). Na Natureza, os individuos mais aptos são os que apresentam maiores probabilidades de sobrevivência. Para a simulação deste tipo de comportamento, o AG utiliza uma função de avaliação que classifica como mais ou menos aptos os vários elementos da população. As mutações genéticas que se podem verificar em cada individuo, são simuladas forçando alterações aleatórias nos individuos da nova geração. O objectivo é conseguir uma população que contenha pelo menos um indivíduo que satisfaça as condições requeridas à partida.

É importante notar que um AG mantem ao longo da sua execução um conjunto de soluções no que difere de outros algoritmos heurísticos de busca aleatória como os já citados, *Hill Climbing e Simulated Annealing*.

O funcionamento de um AG consiste, basicamente, em gerar sucessivos conjuntos de potenciais soluções (populações) que serão, em média, melhores do que as de gerações anteriores. Para tanto, um AG pressupõe métodos para:

- Criação de uma população inicial.
- Avaliação de todos os indivíduos constituintes da população.
- Selecção dos mais aptos.
- Acasalamento dos escolhidos.
- Criação de novos indivíduos a partir do cruzamento dos anteriormente seleccionados.
- Modificação aleatória de alguns elementos da população.

Em [Rawlins, 91] é apresentada uma definição restrita de um AG a que chama algoritmo genético canónico. É provavelmente o mais frequentemente citado como exemplo de um algoritmo genético padrão por oposição à diversidade de variações propostas. As suas características são:

- Todos os individuos são representados por *strings* de *bits* e têm o mesmo comprimento.
- Todas as populações geradas têm o mesmo número de indivíduos.
- As alterações induzidas pelo ambiente ("mutação") é probabilística e normalmente idependente do valor da *string*.
- A reprodução restringe-se ao *crossover* conforme adiante descrito.

Na figura 1 apresenta-se este tipo de algoritmo de forma esquemática.

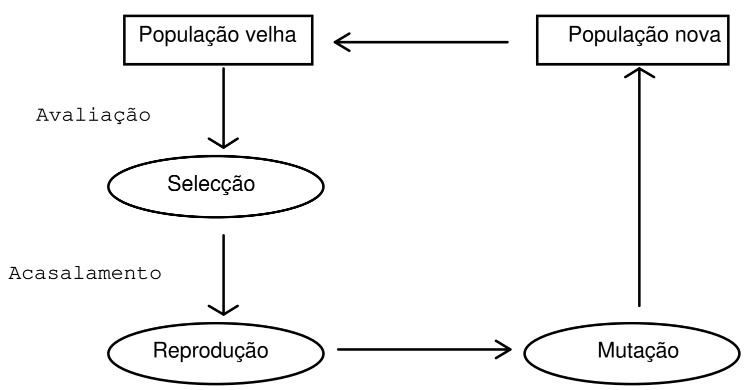


Figura 1: O Algoritmo genético canónico.

Na figura 2 apresenta-se o AG conforme o apresentado em [Michalewicz, 92]. Um AG pressupôe ainda uma prévia codificação do espaço de soluções possíveis que pretendemos explorar.

```
procedure Algoritmo Genético
begin
   tempo := 0;
   Criar população(tempo);
   Avaliar população(tempo);
   while not condição de terminação do
      tempo := tempo + 1;
      Seleccionar população(tempo) da população(tempo - 1);
      Modificar população(tempo); { operadores genéticos }
      Avaliar população(tempo);
   end
end
```

Figura 2: Pseudo-código do algoritmo genético

Quando a codificação utilizada não corresponde a uma sequência de *bits*, Michalewicz utiliza a denominação *algoritmo evolutivo* em alternativa a algoritmo genético.
A exigência, por parte do AG canónico, de uma codificação em *strings* de *bits*, implica, de facto, uma modificação do problema inicial de forma a que este possa ser suportado pela codificação.

A ineficiência, ou mesmo falha, das aplicações de AGs encontra-se ligada a dois factores primordiais: uma codificação inicial inadequada ou operadores inadequados que podem, mais cedo ou mais tarde, destruir muita da relação inicialmente imposta entre a codificação e o problema inicial. Um dos problemas ainda em aberto consiste precisamente em reconhecer quais as características da codificação inicial que podem melhor ser exploradas pelo AG. Uma codificação não adequada ao problema em causa pode só ser detectada após verificação experimental.

Os AGs foram já aplicados, com bastante sucesso, em problemas de optimização como a ligação dos nós de uma rede, escalonamento de tarefas, controlo adaptativo, optimização das queries a bases de dados, etc. Também os problemas de busca e aprendizagem constituem domínios por excelência, para a aplicação de AGs.

Em seguida, apresentamos o exemplo mais usual de utilização de um AG.

A OPTIMIZAÇÃO DE UMA FUNÇÃO

Como exemplo da aplicação de um AG iremos apresentar a descrição de um possível implemento de um optimizador de funções reais de variável real, seguindo as linhas do proposto em [Michalewicz, 92]. É provavelmente o exemplo mais típico de aplicação de um AG. O algoritmo aqui descrito pode considerar-se um exemplo do AG canónico anteriormente apresentado.

Codificação

Suponha-mos que se pretende optimizar uma função $f(x_1,x_2.....,x_n)$: $\boldsymbol{D}^n \rightarrow \boldsymbol{R}$, com $\boldsymbol{R}$ o conjunto dos números reais. Cada variável será então representada por uma *string* de *bits* que possa comportar um valor binário para todos os valores que se pretendem possíveis para a variável. Isto corresponde, no caso de uma variável do tipo inteiro x_i, ao comprimento do seu domínio $\boldsymbol{D}^i$. Para uma variável real o comprimento da *string* de *bits* necessária resulta da precisão com que se pretende indicar os valores das variáveis. Assim sendo, para uma variável real x_i com um domínio $[a_i,b_i]$ que se pretende indicada com precisão de p_i casas decimais, o número de *bits* m_i necessário é o menor valor que verifica a inequação:

$$10^p(b_i - a_i) \leq 2^{m_i} - 1$$

Cada solução potencial (individuo da população) é representada por uma *string* de *bits* resultante da justaposição das *strings* de *bits* correspondentes às variáveis. Asssim sendo, se por exemplo a variável x_1 necessitar de três *bits* para a sua codificação e a variável x_2 de dois e o número total de variáveis for

de duas, então os individuos resultantes serão constituídos por cinco *bits*. Os primeiros três relativos à variável x_1 e, os dois seguintes, à variável x_2.
Como é evidente, no caso de variáveis inteiras a codificação seria ainda mais simples.

Criação

Antes de iniciar o algoritmo iterativo é necessário criar uma população inicial com uma cardinalidade aleatória. Também os cromossomas podem ser criados aleatóriamente desde que respeitando o comprimento previamente calculado. No entanto se conhecermos valores das variáveis potencialmente mais favoráveis, poderemos utilizar tal conhecimento para criar uma melhor população. Melhor significa que contem cromossomas, em média, representativos de um melhor conjunto de soluções. Também a cardinalidade da população pode ser imposta.

Avaliação

Possuindo já uma população, o passo seguinte consiste em avaliar cada um dos cromossomas que a constituem. No presente exemplo, tal significa simplesmente calcular o valor da função para os valores das variáveis presentes em cada cromossoma. No caso geral, aplica-se uma função de avaliação a cada cromossoma.

Selecção

Em seguida é necessário seleccionar os cromossomas que irão "reproduzir-se".
Tal como explica a teoria da evolução de Darwin, os mais aptos (no nosso caso com melhores valores da função) são os que têm mais hipóteses de sobrevivência. No entanto, tal como sucede na Natureza, mesmo os menos capazes podem sobreviver, basta terem sorte. Para simular este comportamento utiliza-se um método que se pode comparar ao girar de uma roleta, onde os cromossomas possuem, cada um, um espaço proporcional ao valor que obtiveram aquando da sua avaliação. Os melhores têm mais probabilidades de ganhar, neste caso de reproduzir-se.

O método pode considerar-se dividido em duas partes: na primeira calcula-se a percentagem da roleta que vai caber a cada cromossoma; na segunda faz-se rodar a roleta tantas vezes quantas o número de cromossomas da população.
Para partilhar a roleta pelos cromossomas seguem-se os seguintes passos:

- Calcula-se o valor da função objectivo *f* para cada cromossoma v_i tal que
  (i = 1, 2, …, #População)
- Calcula-se o valor da função objectivo para a população tomada como um todo:
  $$T = \sum_{i=1}^{\#População} f(v_i)$$
- Calculam-se as probabilidades de selecção para cada cromossoma v_i:
  $$p_i = \frac{f(v_i)}{T}$$
- Calculam-se as probabilidades cumulativas q_i para cada cromossoma:
  $$q_i = \sum_{j=1}^{i} p_j$$

Para simular cada rotação da roleta escolhe-se aleatóriamente um número *r* no intervalo [0,1]. Se $r < q_1$ então adiciona-se um cromossoma igual ao primeiro à nova população que se está a gerar. Caso contrário selecciona-se o cromossoma v_i que verifique: $(2 \leq i \leq \#População) \wedge (q_{i-1} < r < q_i)$.

Daqui se conclui que provavelmente alguns cromossomas da população inicial não serão escolhidos (aqueles que apresentam maus valores para a função objectivo). Outros poderão ser escolhidos mais do que uma vez. É de facto este tipo de comportamento que se pretende simular. Os melhores terão mais possibilidades de sobreviver.

Reprodução

Neste exemplo de AG a reprodução corresponde ao cruzamento (*crossover*) entre os cromossomas seleccionados.

Para saber quais os cromossomas que se irão cruzar segue-se um método muito simples. Para cada cromossoma escolhe-se um número aleatório r_i no intervalo [0,1]. Dada uma probabilidade de cruzamento p_c(um dos parâmetos do AG), o cromossoma v_i será cruzado com outro se $r_i < p_c$.

Os cromossomas escolhidos cruzam-se aleatóriamente entre si. Se o número de cromossomas seleccionados for par a tarefa encontra-se facilitada, basta cruzá-los dois a dois. Se for ímpar, opta-se, também aleatoriamente por adicionar ou subtrair um cromossoma, ao conjunto dos seleccionados.

O cruzamento entre dois cromossomas consiste simplesmente em dividir ambos em dois segmentos e trocá-los entre si (ver figura 3). A posição onde ambos os cromossomas são partidos para posterior troca de segmentos é determinada aleatoriamente.

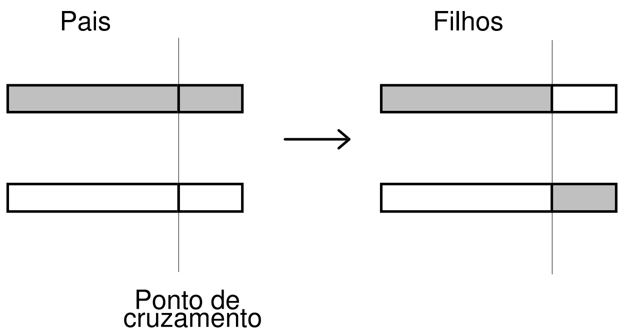


Figura 3: Cruzamento (*crossover*).

Aqui, tal como no AG canónico anteriormente descrito, foi utilizado o cruzamento simples, ou seja, com um único ponto de cruzamento. O número destes pontos determina quantos segmentos são trocados entre os cromossomas.

Mutação

Na Natureza as populações de seres vivos não se modificam apenas devido às combinações das caraterísticas genéticas dos pais. Muito raramente, acontecem pequenas e, muitas vezes, inexplicáveis modificações na herança genética dos novos seres. São estas raras mutações genéticas que se podem introduzir no modelo dos AGs através da modificação aleatória de um pequeno número de *bits* (genes) dos cromossomas. Na realidade, a mutação é a forma que os AGs têm de introduzir novas características não presentes na população na população inicial

A probabilidade de mutação p_m (mais um dos parâmetros do AG) típicamente muito baixa, serve exactamente para decidir se cada *bit* de cada cromossoma deve ou não ser modificado. Novamente se aplica um método semelhante aos anteriormente descritos na Selecção e Reprodução: a cada *bit* da população atribui-se um valor aleatório no intervalo [0,1]. Se o valor assim determinado for menor que p_m então o *bit* correspondente é mudado. Na figura 4 o terceiro e o penúltimo (a partir de cima) bits ("gene") foram modificados.

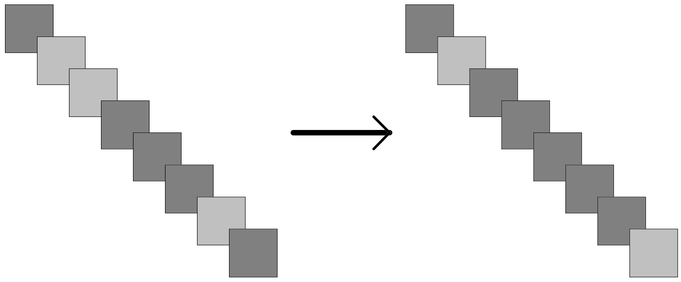


Figura 4: Mutação.

No fim deste processo obtém-se uma nova população a partir da qual já se pode efectuar uma nova selecção de cromossomas. Está criada uma nova geração.

O AG terminará quando se alcançar uma população satisfatória ou, mais frequentemente, quando se atingir um número pré-determinado de gerações.

Pode suceder que a solução mais satisfatória surja numa qualquer geração vindo a desaparecer numa geração posterior. Isto sucede devido aos erros estocasticos da amostra inerentes a considerar-se uma população finita [Michalewicz, 92]. Para obviar este problema opta-se simplesmente por guardar, numa estrutura externa ao algoritmo, a melhor solução até à data.

OS OPERADORES GENÉTICOS

Muita da investigação actual sobre AGs procura encontrar mais e variados operadores genéticos que apresentem um melhor desempenho, tanto a nível de aplicações específicas de AGs como a um nível mais generalista.

Dada a imensidade de novos operadores constantemente propostos, não é aqui apresentada a sua grande maioria. Referem-se, apenas, alguns dos mais simples, até porque permitem já obter uma ideia, mais concreta, das variações possíveis.

Em primeiro lugar descrevemos algumas variantes dos operadores canónicos já referidos: cruzamento e mutação. Em seguida apresentamos outros operadores que se adaptam particularmente bem à reordenação de sequências em que pretenda evitar o surgimento de símbolos repetidos. Como se verá, os operadores descritos destinam-se a representações de cromossomas correspondentes a *strings* de *bits* e/ou sequências de símbolos.

Cruzamento

Este operador foi inspirado no mecanismo de reprodução natural, comum a uma grande maioria de seres vivos, mecanismo essencial à sobrevivência e mesmo à evolução. Este processo necessita de dois seres para se iniciar, produzindo um ou mais descendentes.

Como foi apresentado anteriormente em termos artificiais, a operação limitava-se a introduzir um ponto de corte em dois cromossomas da população, trocando segmentos entre os eles, produzindo dois novos cromossomas a partir dos iniciais.

Este mecanismo, embora simples e directo, muitas vezes pode não originar as melhores soluções, tanto por não se adequar à estrutura do problema, como pela pouca versatilidade que o próprio processo encerra.

Apresentaremos outras formas de abordar o problema do cruzamento, que pelas suas características, simultaneamente semelhantes no objectivo e distintas na forma, podem resultar num melhor espectro de soluções.

Considere-se por exemplo que num dado problema o seguinte cromossoma seria altamente considerado: C=(1,1,_,0,0,_,_,1), em que os *underscore*, designam posições irrelevantes, além disso teríamos dois elementos da população, C_1=(**1**,**1**,0,1,1,0,0,**1**) e C_2=(0,0,1,**0**,**0**,1,1,0). Rapidamente se verifica que pela operação normal de cruzamento, o resultado C nunca se poderia verificar, qualquer que fosse o ponto de corte, no entanto se definirmos uma operação com dois pontos de corte, o cromossoma desejado já poderia ser conseguido (com um corte anterior à terceira posição e outro anteriro à ultima posição).

Este é um operador que generaliza o operador de cruzamento simples[1]. Mesmo assim conseguiríamos apresentar pares de cromossomas e resultados desejados, que mesmo este novo operador não resolveria. Continuando com o processo, o próximo operador expandiria o conceito a múltiplos pontos de corte — designa-se por *cruzamento multi-ponto* — sendo número de cortes fixo, ou mesmo aleatório.

Seguindo esta linha de raciocínio, torna-se apropriado generalizar ainda mais o conceito de cruzamento, dando-lhe, agora, um sentido menos discreto. Vamos designá-lo por *cruzamento continuo*. O mecanismo não deixa por isso de ser simples: basta associar a cada elemento de um dos cromossomas, uma probabilidade de 50%, de pertencer ao primeiro filho ou não. Os elementos do segundo cromossoma pai, serão o complementar binário dos resultados do primeiro. Faz-se a divisão, resultando em dois novos filhos, distribuídos que estão as informações genéticas dos pais. Veja-se o mesmo exemplo apresentado anteriormente.

Seja C_1=(1,1,0,1,1,0,0,1), apliquemos então o processo aleatório, dando o seguinte resultado: $C_1=(1_1,1_1,0_2,1_2,1_2,0_1,0_1,1_1)$, onde os índices associados indicam se o número vai fazer parte do cromossoma do primeiro filho ou do segundo. Obtemos, assim, o seguinte C_2: $C_2=(0_2,0_2,1_1,0_1,0_1,1_2,1_2,0_2)$. Logo, os dois filhos resultariam em F_1=(1,1,1,0,0,0,0,1) e F2=(0,0,0,1,1,1,1,0).

Apesar de todas estas novas operações, não é possível indicar qual o melhor operador de cruzamento a aplicar-se numa situação de resolução geral de problemas. Cada um possui vantagens sobre os outros em determinados tipos de questões.

Mutação

A mutação falada até aqui, pressupõe uma uniformidade relativamente à posição dos alelos dentro do cromossoma, ou seja a probabilidade da mutação é igual para todos os elementos do cromossoma. Só que isto pode levar a maus resultado, se persistirmos nesta abordagem, porque para muitos problemas, o conceito do alelo mais significativo é um facto[2]. Nesse tipo de problemas não devemos considerar idêntica a importância dos vários alelos. Com efeito, a função de avaliação pode ser muito mais sensível para alguns alelos do que para outros. Os grandes "saltos", devem ser mais raros que os menores, para que a evolução de faça de uma forma mais gradual, mais consentânea portanto, com o que sucede na Natureza.

Assim se introduz o conceito de *mutação não-uniforme*, que tal como o nome indica, apresenta, uma não uniformidade na probabilidade da mutação para os diferentes elementos constituintes do cromossoma a ser afectado (os seus alelos).

Podemos definir uma função $P_m: I \rightarrow R$, que dado um inteiro correspondente à posição no cromossoma, nos dá uma probabilidade específica de mutação. Por exemplo, se $P_m(2) = 0.4$ então o bit no alelo 2 teria 40% de probalilidade de ser modificado por mutação.

---

[1]Se os dois pontos de corte fossem na mesma posição, voltariamos ao operador inicial.

[2]Um dos casos mais conhecidos, é o da optimização de funções, onde o *bit* mais significativo é várias ordens de magnitude mais importante que o *bit* menos significativo.

Aprofundando ainda mais esta ideia, verifica-se que o número de gerações já passadas, pode também ser um factor a considerar, já que num algoritmo genético, é esperado que ao fim de algumas gerações, os resultados comecem a convergir para valores próximos do óptimo. Por sua vez, a banda de variação deve diminuir para aumentar a velocidade de convergência e diminuir os riscos de um "salto" demasiado grande, impróprio nessa fase de processamento. Para que a função P_m, possa entrar em linha de conta com este efeito de "abrandamento" da evolução, podemos inserir-lhe uma nova dimensão: o número de gerações, ou, mais simplesmente, o tempo. Logo $P_m(pos, T)$ indica a probabilidade de um dado alelo, na posição *pos* e decorridos *T* períodos, sofrer uma mutação.

Outros operadores genéticos

Vamos dividir os operadores, em seguida descritos, em unários e binários, conforme necessitem de um ou dois pais.

Unários

Os operadores unários não têm correspondência na Natureza e são, de um modo geral, pouco ricos em novas combinações dado não introduzirem novos elementos em termos de herança genética de pai para filho. Por outro lado, podem provocar grandes mudanças na avaliação do cromossoma, especialmente se o contexto dos alelos for sensível e determinante para a validade do indivíduo face à função de avaliação.

O operador mais simples é provavelmento o operador *troca* — de dois genes. Ele simplesmente troca a posição de dois símbolos no cromossoma. Uma generalização deste operador leva-nos a considerar a troca de dois segmentos separados por um ponto de corte. Por exemplo: C=(1,1,1,1,0,0,0,0), seja o ponto de corte entre o quarto e o quinto elemento, isto produzirá o seguinte descendente, C'=(0,0,0,0,1,1,1,1). Também este operador pode ser generalizado, para múltiplos pontos de corte, em que o extremo — *troca continua* — significa reordenar ao acaso todos os alelos do cromossoma, mantendo a sua distribuição intacta. De qualquer forma, de pouco parecem servir estes extremos, dada à enorme mistura que provocam nos cromossomas .

Outro operador unário é o *splice*. Este origina um novo cromossoma resultante da mudança de um segmento, delimitado por dois pontos de corte, para o fim do cromossoma original. Como exemplo consideremos o cromossoma seguinte C em que o segmento a movimentar se encontra assinalado: C=(1,0,**0**,**0**,**1**,**0**,1,1). O cromossoma C' resultante será: C'=(1,0,1,1,**0**,**0**,**1**,**0**).

O operador *inversão*, é inverte a sequeência de genes entre os dois pontos de corte. Para o memso vector C= (1,0,**0**,**0**,**1**,**0**,1,1) obteríamos, agora, C'=(1,0,**0**,**1**,**0**,**0**,1,1).

Binários

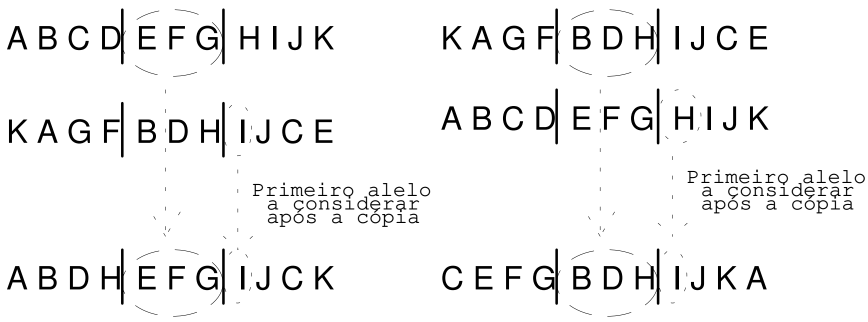
Dado obterem informação de dois pais, os operadores binários são mais poderosos no sentido de possuirem uma maior capacidade de transmitirem características boas, ou simplesmente comuns, da população em causa.

Como primeiro e mais simples exemplo de umm operador binário, temos a *troca binária*, que simplesmente escolhe um alelo do cromossoma, e o troca com o outro pai, originando dois filhos. De certa forma, é uma mistura entre um cruzamento simples (*crossover*) e uma mutação simples. Por exemplo: C_1=(1,1,1,1,1,1,1) e C_2=(0,0,0,0,0,0,0), seja a quarta posição a escolhida. Os dois filhos serão: F_1=(1,1,1,**0**,1,1,1) e F_2=(0,0,0,**1**,0,0,0).

Para aplicações em que os cromossomas correspondam a sequências de símbolos em que a sua ordenação constitui o cerne do problema um operador adequado poderá ser o *cruzamento ordenado*[3]. Este cria filhos que preservam a ordem original de um segmento de um dos pais, e mantem a ordem relativa dos elementos restantes provenientes do outro pai. A operação começa por seleccionar aleatoriamente dois pontos de corte. Os segmentos contidos entre os dois pontos de corte, em cada um

[3]Este operador e outros mais sofisticados encontram-se descritos em [Fox et al., 91].

dos cromossomas, são copiados para os filhos. Para completar o primeiro filho procede-se da seguinte forma começando no primeiro alelo do segmento à direita do segundo ponto de corte do segundo pai: copia-se o gene desse pai para a correspondente posição no filho, desde que ele ainda não conste no filho. Procede-se de idêntica forma para o alelo seguinte do pai. Quando se chega ao fim deste, continua-se no seu primeiro alelo. Prossegue-se até chegar ao primeiro ponto de corte. Para gerar outro filho, trocam-se os papéis dos pais (figura 5).



a) b)

Figura 5: Cruzamento ordenado. Observe-se como no caso b) o gene H não foi copiado dado já constar devido à cópia anterior do segmento B D H.

Bibliografia

[Fox et al., 91] Fox, B. R. e McMahon, M.B., "Genetic Operators for Sequencing Problems", in [Rawlins, 91], pp.284-300.

[Michalewicz, 92] Michalewicz, Z., *Genetic Algorithms + Data Strutures = Evolution Programs,* Springer-Verlag Berlin Heidelberg, 1992 .

[Rawlins, 91]Rawlins, G., *Foundations of Genetic Algorithms*, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems, Morgan Kaufman Publishers, Los Altos, CA, 1991.

[Spillman, 93] Spillman, R. - "Genetic Algorithms", pp.26-30, Dr. Dobbs Journal, February 1993.