Algoritmos Genéticos

André Ricardo Gonçalves
andreric [at] dca.fee.unicamp.br
www.dca.fee.unicamp.br/~andreric

Sumário

1 Algoritmo Genético

Charles Darwin afirmava que o mecanismo de evolução é uma competição que seleciona os indivíduos mais bem adaptados em seu ambiente podendo assegurar descendentes, transmitindo as características que permitiram sua sobrevivência (DRÉO et al., 2005).

Com base na teoria Darwiniana, pesquisadores tentaram imitar tal mecanismo de evolução. Pesquisas essas que levaram a criação de vários métodos de busca e otimização em problemas complexos, os quais são intratáveis por métodos clássicos. O **AG!** (**AG!**) pertencente a essa classe de métodos embasados na teoria de evolução das espécies é o método mais difundido na comunidade científica e também como ferramenta para resolução de diversos problemas da engenharia, matemática, computação, entre outros.

Este capítulo apresenta os principais conceitos de Algoritmos Genéticos, especificando suas etapas de execução, detalhes de implementação além da descrição de aspectos teóricos.

1.1 Computação Evolucionária

A **CE!** (**CE!**) é uma sub-área da Inteligência Artificial, na qual diversos "algoritmos evolucionários" estão presentes. Algoritmos que possuem como inspiração a teoria da evolução das espécies descrita por Charles Darwin em 1859 no famoso livro "A Origem das Espécies", teoria essa que introduziu o conceito de *seleção natural*.

Os algoritmos evolucionários são constituídos de quatro elementos básicos:

1. Uma população de indivíduos, onde cada indivíduo corresponde a uma solução candidata do sistema;
2. Uma maneira de criar novos indivíduos, a partir de indivíduos já existentes;
3. Uma forma de medir a qualidade da solução que cada indivíduo corresponde;
4. Um método de selecionar os melhores indivíduos, aplicando assim o princípio da seleção natural.

Os algoritmos evolucionários que pertencem a CE são listados abaixo.

- Algoritmo Genético;

- Programação Genética;
- Sistemas Classificadores;
- Estratégias Evolutivas;
- Programação Evolucionária.

Mesmo com algumas diferenças, estes algoritmos possuem a mesma idéia geral, trabalham com uma população de indivíduos, os quais representam possíveis soluções do problema. Esses indivíduos passam por algumas modificações genéticas e os mais aptos são selecionados para gerar descendentes, criando uma nova população de indivíduos provavelmente mais aptos. Este processo se repete até que encontre um indivíduo que represente a solução ótima ou próxima dela.

1.2 Algoritmo Genético

O Algoritmo Genético (AG) constitui um modelo matemático que simula a teoria da evolução Darwiniana, no qual existe inicialmente um conjunto de indivíduos (população inicial), que representam possíveis soluções para um determinado problema e a cada iteração, chamada de *geração* em AG. Os indivíduos são avaliados em relação ao seu nível de adaptabilidade com o meio externo e os mais aptos são selecionados para gerar descendentes.

Uma nova população é gerada através da aplicação de operadores genéticos, como recombinação e mutação de genes. Este processo é realizado até um determinado número de gerações e o indivíduo mais apto encontrado é dito ser a solução do problema (LOPES, 2006).

O algoritmo genético foi apresentado inicialmente por John Holland em seu trabalho intitulado de *"Adaptation in Natural and Artificial Systems"* em 1975, com o objetivo de formalizar matematicamente e explicar os processos de adaptação de processos naturais e desenvolver sistemas artificiais que mantenham os mecanismos originais encontrados em sistemas naturais (IYODA, 2000).

Em AG os indivíduos são representados por cromossomos e cada símbolo do cromossomo é chamado de *gene*. Os genes por sua vez armazenam informações, os *alelos*.

Os cromossomos são geralmente implementados como uma cadeia de *bits*, comumente utiliza-se um vetor como estrutura de armazenamento. O nível de adaptabilidade do indivíduo em relação ao meio é calculado com base na função objetivo, que nos casos mais simples é própria função que se quer maximizar. Em AG esta função é denominada função de *fitness*.

Seja $P(t)$ a população de indivíduos na geração t, o algoritmo 1 apresenta

o pseudo-código do AG canônico descrito em (MICHALEWICZ, 1996).

Algoritmo 1: Algoritmo Genético Canônico

```
1  begin
2      t ← 0;
3      inicia P(t);
4      avaliar P(t);
5      while não condição de parada do
6          t ← t + 1;
7          selecione P(t) a partir de P(t − 1);
8          altere P(t);
9          avalie P(t);
10     end
11 end
```

O primeiro passo de um AG é a geração da população inicial, que é formada por um conjunto de cromossomos que representam possíveis soluções de um determinado problema, chamadas de *soluções-candidatas* (LACERDA; CARVALHO, 1999). Este processo é realizado de maneira aleatória na maioria das aplicações.

Durante o processo evolutivo os melhores indivíduos são selecionados com base em seu valor de *fitness*, ou seja, quanto maior o nível de adaptabilidade de um indivíduo em relação ao ambiente, maiores serão suas chances de sobreviver e gerar descendentes.

No decorrer da execução do AG, os cromossomos podem sofrer trocas de genes (*crossover*) e mutações, com o intuito de explorar regiões desconhecidas no espaço de busca. Este processo é repetido até que uma solução satisfatória seja encontrada (LACERDA; CARVALHO, 1999).

De acordo com (LACERDA; CARVALHO, 1999) alguns dos principais critérios de parada são:

1. utilização de um número máximo de gerações;
2. obtenção do valor ótimo da função objetivo, se for conhecido;
3. quando não houver mais melhorias significativas no cromossomo de maior aptidão, ou seja, o sistema convergiu.

Segundo (TANOMARU, 1995) o algoritmo genético não afirma que as soluções sejam ótimas, mas os processos naturais, principalmente relacionados aos seres vivos são soberbamente adequados ao nosso mundo.

1.2.1 Codificação de Indivíduos

O processo de codificação de indivíduos é onde ocorre a representação de cada possível solução, de um problema qualquer, como uma seqüência de símbolos gerados a partir de um alfabeto finito (TANOMARU, 1995).

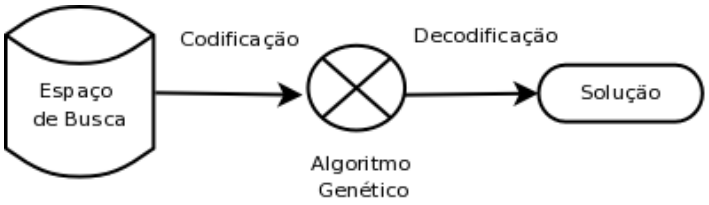

Figura 1: Processo de codificação e apresentação dos dados ao AG

Holland (1975) inicialmente utilizou a codificação binária e afirma que qualquer problema pode ser satisfatoriamente representado pela codificação binária.

Além da codificação binária, existem outras codificações que dependendo do problema, obtêm resultados melhores utilizando tais codificações. Como a codificação inteira, no qual os genes são representados por número inteiros, codificação de *string*, onde os cromossomos são letras do alfabeto, entre outras.

Para cada tipo de codificação existem operadores de *crossover* e mutação específicos, os quais serão apresentados na seção 1.2.3.1 e 1.2.3.2.

Codificação Binária

Em grande parte dos problemas práticos tratados com AG é utilizada a codificação binária (LOPES, 2006). Isto se deve ao fato de que geralmente as variáveis do problema trabalhado podem ser representadas em números binários.

De acordo com (TANOMARU, 1995) um indivíduo x, em uma codificação binária utilizando a notação binária posicional, é representado por uma seqüência $s = [b_n....b_2b_1]$, onde n é o número de bits necessários para representar x e cada $b_i \in$ 0,1. Por exemplo, se o espaço de busca é o intervalo [0,100], um vetor binário de 7 posições é necessário para representar esta variável.

Para a utilização do alfabeto binário é necessária uma discretização das variáveis, ou seja, um mapeamento entre o intervalo real da variável, [x_{min},x_{max}] em um intervalo binário [0,2^n], onde n é o número de bits necessários para representar o maior número no intervalo real (LOPES, 2006).

Segundo Michalewicz (1996) quando a codificação binária é utilizada para representar valores reais, o tamanho do cromossomo dependerá da precisão requerida. Sendo t o tamanho do intervalo real da variável e p a precisão requerida. O tamanho do cromossomo necessário para representar esta variável é calculado conforme a Eq. (1.1).

$$n = \lfloor log_2(t * 10^p) \rfloor + 1 \tag{1.1}$$

Por exemplo, se o intervalo de busca é [1,3], temos que o tamanho do intervalo é igual a 2 ($t = 3 - 1 = 2$), se utilizarmos uma precisão de 5 casas decimais ($p = 5$), seria necessário $\lfloor log_2(2 * 10^5) \rfloor + 1 = 18$ bits. Valores inteiros podem ser mapeados da mesma maneira, utilizando $p = 0$.

Com o valor do tamanho do cromossomo obtido, é necessário mapear o intervalo real da variável no intervalo binário. Este processo é realizado em duas etapas: inicialmente aplicamos a função de mapeamento sobre o valor real da variável e posteriormente convertemos o valor obtido para base binária. Sendo x o valor real da variável e $\hat{x}$ o valor mapeado, a função de mapeamento é descrita pela Eq. (1.2).

$$\hat{x} = \frac{(x - x_{min}) \cdot (2^n - 1)}{t} \tag{1.2}$$

Assim o cromossomo é representado pela cadeia de bits obtida com a conversão de $\hat{x}$ para base binária.

Já o processo de decodificação é realizado em duas etapas: inicialmente converte-se o cromossomo da base 2 para base 10 de acordo com 1.3, na qual o resultado é um valor discreto $\hat{x} \in \mathbf{N} \mid 0 \leq x \leq 2^n - 1$.

$$s = [b_n b_{n-1} .... b_2 b_1 b_0] = \sum_{i=0}^{n} b_i \cdot 2^i = \hat{x} \tag{1.3}$$

Na etapa seguinte $\hat{x}$ é mapeado de volta no espaço de busca, de acordo com 1.4, onde x_{min} e x_{max} indicam os limites do intervalo de busca.

$$x = x_{min} + \left(\frac{x_{max} - x_{min}}{2^n - 1} \right) \cdot \hat{x} \tag{1.4}$$

Codificação Real

Segundo (MICHALEWICZ, 1996), a codificação binária possui algumas desvantagens quando é aplicada à problemas de alta dimensionalidade e com alta precisão numérica. Isto ocorre pelo fato de quando há um grande número de variáveis, obtêm-se longas cadeias de bits que podem fazer o algoritmo convergir vagarosamente (LACERDA; CARVALHO, 1999). Por exemplo, para 100 variáveis com domínio no intervalo [-500,500], com precisão de 6 casas decimais, a cadeia de bits possuirá um tamanho igual a 30, gerando um espaço de busca igual a 10^{1000}, o que é intratável na prática. Um AG utilizando codificação binária obtém um pobre desempenho para este tipo de problema (MICHALEWICZ, 1996).

Na codificação real, o cromossomo é representado por um vetor de números reais, no qual cada posição do vetor (gene) contém o valor de uma variável do problema. A figura 2 mostra um cromossomo utilizando codificação real.

1.029	0.215	0.984	0.128	1.854	1.024

Figura 2: Exemplo de cromossomo na codificação real

Os cromossomos gerados são menores, em relação a codificação binária e também esta codificação é de melhor compreensão para o ser humano (LACERDA; CARVALHO, 1999).

1.2.2 Seleção

De modo a escolher os melhores indivíduos da população, emulando o processo de seleção natural, um método de seleção é utilizado.

Existem vários métodos de seleção presentes na literatura, porém não há um senso comum em relação a qual método é melhor para determinado tipo de problema, isto ainda é uma questão aberta em AG (MITCHELL, 1996).

Holland (1975) inicialmente utilizou um método de seleção baseado na proporção do valor de *fitness* do indivíduo em relação a soma do *fitness* de toda da população (MITCHELL, 1996). Sendo assim a probabilidade de um indivíduo i, ser escolhido é dado por

$$p_i = \frac{f_i}{\sum_{k=1}^{N} f_k} \tag{1.5}$$

onde f_i é o valor do *fitness* do indivíduo i e N o número de indivíduos na população.

De acordo com (TANOMARU, 1995) se não houvesse o processo de seleção, além do AG perder a grande parte do caráter evolutivo, o mesmo seria um processo ineficiente similar a uma busca aleatória.

Método da Roleta

O método mais utilizado e também o mais simples, é conhecido como método da roleta (*roulette wheel*). Nesta abordagem utiliza-se uma roleta fictícia, na qual cada cavidade ("casa") da roleta representa um indivíduo, sendo a área da cavidade proporcional ao valor do *fitness* do indivíduo (TANOMARU, 1995). Dessa forma indivíduos com maior *fitness* têm maior chance de ser escolhido para gerar descendentes. Exemplo de uma roleta fictícia é mostrada na figura 3, onde os indivíduos com maior valor de *fitness* possui uma cavidade maior.

O método da roleta apresentado por (LACERDA; CARVALHO, 1999) pode ser visto pelo algoritmo 2.

Algoritmo 2: Algoritmo da Roleta

```
1  begin
2      total ← Σ_{i=1}^{N} f_i;
3      rand = randomico(0,total);
4      total parcial ← 0;
5      i ← 0;
6      repeat
7          i ← i + 1;
8          total parcial ← total parcial + f_i;
9      until total parcial ≥ rand ;
10     return indivíduo s_i
11 end
```

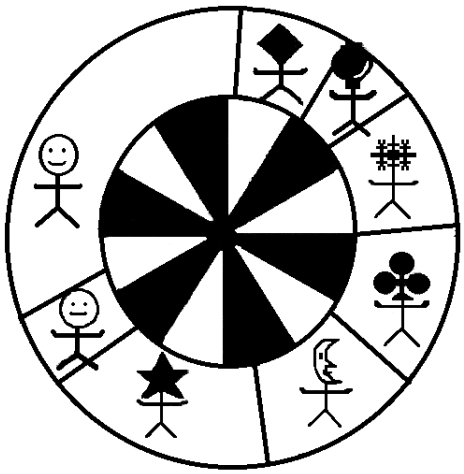

Figura 3: Exemplo de roleta de seleção

Segundo (LOPES, 2006) o método da roleta é muito "agressivo", pois discrimina indivíduos de melhor *fitness* em detrimento de indivíduos com menor valor, ocasionando uma convergência em poucas gerações e provavelmente para um máximo local. Métodos de ordenamento, em especial o linear reduzem este tipo de problema, o método baseado em *rank* é um exemplo desta classe.

Método baseado em Rank

Inicialmente os indivíduos são ordenados por seus valores de *fitness*, objetivando determinar as probabilidades de seleção, as quais podem ser determinadas por mapeamentos lineares ou não-lineares. Mapeamentos estes que modificam os valores dos *fitness* antes da seleção afim de aumentar a pressão seletiva, a pressão seletiva é discutida na seção 1.2.5.

Seleção por torneio

Neste método são escolhidos n cromossomos aleatoriamente, com probabilidade de escolha iguais e cromossomo com maior *fitness* é selecionado para gerar descendentes. Segundo (LOPES, 2006; LACERDA; CARVALHO, 1999) valores de $n = 2$ ou $n = 3$ são comumente utilizados.

Elitismo

Operadores de *crossover* e mutação são essenciais para aperfeiçoamento dos cromossomos e a exploração de outras regiões no espaço de busca. Mas vale ressaltar que os melhores cromossomos podem ser perdidos de uma geração para outra com a aplicação destes operadores. Para que seja possível a preservação dos melhores indivíduos, uma estratégia de mantê-los na população foi criada, a qual é denominada de *elitismo*. Nesta abordagem os

melhores cromossomos são transferidos para a próxima geração sem alterações.

O desempenho do AG com elitismo e sem elitismo é mostrada na figura 4.

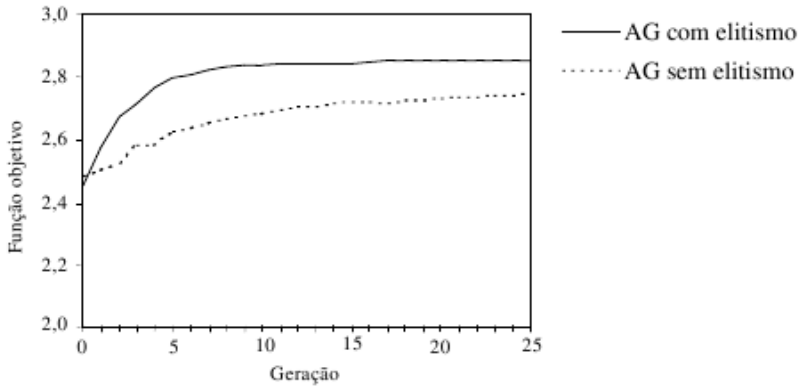

Figura 4: Desempenho do AG com e sem elitismo. [Fonte: (LACERDA; CARVALHO, 1999)]

De acordo com (TANOMARU, 1995) geralmente supervisiona e preserva apenas um indivíduos, mas em grandes populações pode ser interessante garantir uma certa percentagem dos melhores indivíduos.

1.2.3 Operadores Genéticos

Operadores genéticos são responsáveis pela geração de novos indivíduos (nova população) a partir de uma população inicial. A cada geração novos indivíduos vão emergindo, os quais, em teoria, são melhores (mais aptos) do que seus genitores. A figura 5 mostra onde os operadores são aplicados no AG.

Outro ponto a destacar é que os operadores são inseparavelmente ligados ao tipo de codificação utilizada (LOPES, 2006). Neste trabalho serão apresentados os operadores mais difundidos na literatura do AG, o *crossover* (ou recombinação) e a mutação.

Assim nos processos biológicos os fenômenos da recombinação e da mutação, podem ou não ocorrer, havendo certa probabilidade de ocorrência. Assim sendo, em AG é aplicada uma probabilidade de ocorrência a cada operador, geralmente um valor fixo, mas de acordo com (LOPES, 2006) vários autores reportam que esta probabilidade deve ser modificada ao longo da execução do AG. Valores dos parâmetros do AG são discutidos na seção 1.2.4.

1.2.3.1 Crossover

O operador genético do *crossover* cria novos indivíduos para a população através da recombinação de partes diferentes de dois cromossomos-pai escolhidos através do método de seleção (LOPES, 2006). No *crossover* a partir de dois cromossomos-pai dois novos indivíduos são gerados, os quais são chamados de *descendentes*.

Figura 5: Aplicação dos operadores genéticos

Este operador é aplicado com uma determinada probabilidade de ocorrência, para cada par de cromossomos selecionados, denominada taxa de *crossover*, a qual é definida pelo usuário. Uma discussão sobre valores ideais é feita na seção 1.2.4. Caso não ocorra o *crossover* os descendentes serão iguais aos pais, permitindo a preservação de algumas soluções (LACERDA; CARVALHO, 1999).

Diferentes operadores de *crossover* tem sido apresentados na literatura, a escolha de qual aplicar é dependente do tipo de codificação escolhida. Neste trabalho são apresentados *crossover* para codificação binária e real.

Crossover para representação binária

Os operadores de *crossover* binários são simples e de fácil implementação, os mais utilizados são: *crossover* de um ponto e *crossover* de dois ou vários pontos.

Dados dois cromossomos-pai P_1 e P_2, no operador de *crossover* de um ponto, um "corte" é realizado em uma posição randomicamente escolhida de ambos os cromossomos P_1 e P_2, esta posição de corte é denominada ponto de *crossover*, dois novos indivíduos F_1 e F_2 são gerados a partir da concatenação das partes alternadas dos cromossomos-pai (LOPES, 2006), esse processo é mostrado pela figura 6.

A abordagem de dois ou vários pontos é uma extensão do *crossover* de um

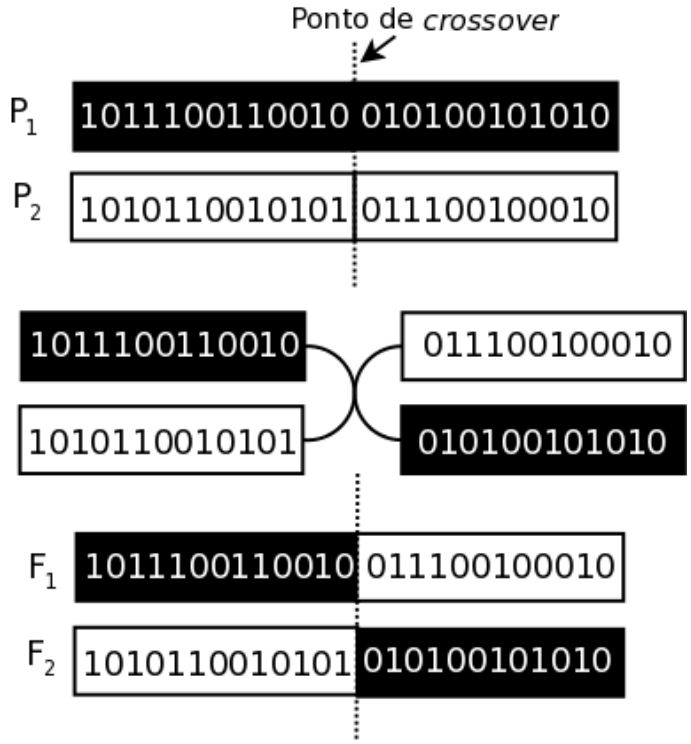

Figura 6: Operador de *crossover* de 1 ponto para codificação binária

ponto, na qual dois ou mais pontos de *crossover* são escolhidos e os cortes são então realizados. A figura 7 mostra o operador *crossover* de 2 pontos.

De acordo com (MITCHELL, 1996) a escolha de qual operador de *crossover* utilizar depende da função de *fitness*, codificação utilizada e outros detalhes do AG, não havendo uma definição, o autor supracitado ainda afirma que normalmente a escolha é realizada de maneira empírica.

Crossover para representação real

Para (OBITKO, 1998) todos os operadores da representação binária podem ser aqui aplicados. (MICHALEWICZ, 1996) destaca três operadores, os quais são definidos a seguir.

- **Crossover Simples** Este operador é a extensão da versão binária do *crossover* de um ponto, para a representação real.
- **Crossover Aritmético** Neste operador os cromossomos descendentes são gerados a partir de uma combinação linear dos cromossomos-pai (P_1 e P_2). A combinação linear que gera os cromossomos filhos F_1 e F_2 é dada pelas Eqs. 1.6 e 1.7 respectivamente.

$$F_1 = \alpha \cdot P_1 + (1 - \alpha) \cdot P_2 \tag{1.6}$$

$$F_2 = \alpha \cdot P_2 + (1 - \alpha) \cdot P_1 \tag{1.7}$$

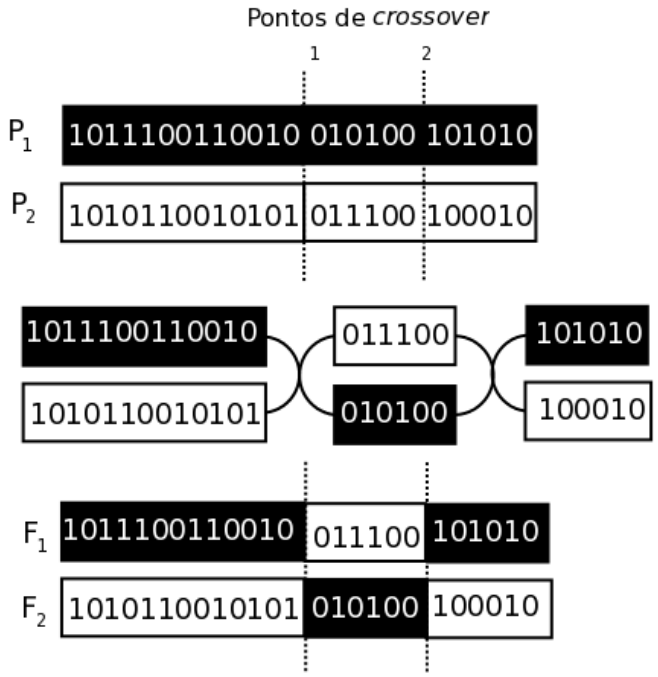

Figura 7: Operador de *crossover* de 2 pontos para codificação binária

Neste operador o valor de α é aleatoriamente obtido dentro do intervalo [0,1].

- **Crossover Heurístico**

  Neste operador, a partir de dois cromossomos-pai P_1 e P_2 um ou nenhum cromossomo filho é gerado. O cromossomo-filho (F_1) é gerado de acordo com a seguinte regra:

$$F_1 = r \cdot (P_2 - P_1) + P_2 \tag{1.8}$$

  desde que P_2 tenha um melhor valor de *fitness* do que P_1 e r é um número aleatório dentro do intervalo [0,1]. (MICHALEWICZ, 1996) afirma que este operador contribui para um melhor refinamento local e busca em direções mais promissoras.

  Muitos outros operadores podem ser encontrados na literatura, vários destes podem ser encontrados em (LACERDA; CARVALHO, 1999).

1.2.3.2 Mutação

O operador de mutação é responsável pela exploração global do espaço de busca introduzindo novo material genético em indivíduos já existentes (LOPES, 2006). (IYODA, 2000) observa que a idéia intuitiva por trás do operador de mutação é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido com a busca. Para que a mutação não prejudique os resultados até então obtidos, uma pequena taxa de mutação é geralmente aplicada, valores dos parâmetros são discutidos na seção 1.2.4.

Mutação para representação binária

O operador de mutação mais simples e mais comumente utilizado apenas seleciona randomicamente um gene do cromossomo e inverte seu valor, como mostrado na figura 8.

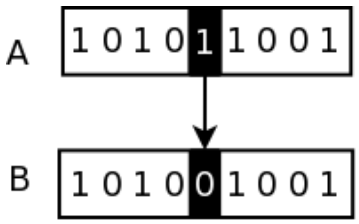

Figura 8: Operador de mutação para codificação binária

Uma variação é a seleção e inversão dos valores de vários genes do cromossomo (OBITKO, 1998).

Mutação para representação Real

De acordo com (IYODA, 2000) os operadores mais utilizados são o uniforme e o gaussiano, mas havendo uma gama de outros operadores na literatura.

- **Mutação Uniforme**

  Apresentado inicialmente por (MICHALEWICZ, 1996), este operador aplica uma simples substituição de um gene por um número aleatório, dentro do intervalo [v_{min},v_{max}], onde v_{min} e v_{max} representam o limite do intervalo permitido para o gene descendente. O número aleatório é gerado com base na distribuição uniforme.

- **Mutação Gaussiana**

  É uma variação da mutação uniforme, onde substitui um gene por um número aleatório de uma distribuição normal.

$$c_i = \begin{cases} N(p_i, \sigma) & \text{, se } i = j \\ p_i & \text{, caso contrário} \end{cases} \tag{1.9}$$

  onde $N(p_i, \sigma)$ é a distribuição normal com média p_i e desvio padrão σ.

- **Mutação Limite**

  Nesta abordagem um gene aleatório é selecionado e substituído por um dos limites do intervalo, [v_{min},v_{max}], com igual probabilidade de escolha. Este operador é utilizado para evitar a perda de diversidade dos filhos gerados pelo *crossover* aritmético, que tende a trazer os genes para o centro do intervalo permitido (LACERDA; CARVALHO, 1999).

1.2.4 Parâmetros do Algoritmo Genético

A escolha de bons parâmetros para o AG é de extrema importância para um bom desempenho do mesmo. Parâmetros esses como o número de indivíduos na população, as probabilidades de execução dos operadores genéticos como o *crossover* e mutação.

De acordo com (LOPES, 2006) não existe uma unanimidade entre os autores em relação à valores ideais de probabilidade e alguns processos de auto-ajuste dos parâmetros estão sendo propostos na literatura recente, objetivando não apenas uma melhor eficiência do AG, mas também reduzindo a responsabilidade do usuário em determinar os parâmetros corretos.

Mesmo sem a existência de um padrão universal, alguns autores têm sugerido algumas abordagens para adaptação dos parâmetros. Segundo (MITCHELL, 1996) as pessoas usualmente utilizam valores, os quais são conhecidos por trabalharem bem em casos semelhantes reportados na literatura.

Alguns autores realizaram uma bateria de testes a fim de identificar como a variação dos parâmetros afetam o desempenho do AG. Configurações largamente utilizadas na comunidade de AG foram obtidas por (DEJONG, 1975 apud MITCHELL, 1996), os testes foram realizados com diversas funções objetivo. Com base nos testes (DEJONG, 1975) indica que entre 50-100 indivíduos em uma população são capazes de cobrir satisfatoriamente o espaço de busca, para a probabilidade de ocorrência de *crossover* (de um ponto) a melhor taxa foi em torno de 60% já a mutação a melhor taxa foi de 0.1% por bit.

Outros autores também sugerem bons parâmetros para AG, para (LACERDA; CARVALHO, 1999) a taxa de *crossover* varia entre 60% e 90% e taxa de mutação entre 0.1% e 5%. Para (TANOMARU, 1995) uma população entre 50-200 indivíduos são suficientes para resolver a maioria dos problemas e ainda afirma que populações maiores podem ser necessárias para problemas mais complexos.

1.2.5 Pressão seletiva

Entende-se por pressão seletiva a razão entre o maior *fitness* da população e o *fitness* médio (LACERDA; CARVALHO, 1999), a qual segundo (LOPES, 2006) é uma das causas da perda de diversidade genética.

Quando a pressão seletiva é muito baixa, ou seja, o *fitness* é praticamente igual para toda a população, o algoritmo genético apresenta um comportamento aleatório, pois no processo de seleção os indivíduos possuem praticamente a mesma probabilidade de serem escolhidos. Quando há uma alta pressão seletiva, alguns indivíduos com valor de *fitness* muito acima da média, denominados de *super-indivíduos*, terão vários descendentes nas próximas gerações, que em casos extremos, indivíduos próximos do ótimo global mas com baixo valor de *fitness* poderão ser extintos. Como conseqüência disto o AG tenderá a convergir rapidamente e provavelmente para um máximo local (TANOMARU, 1995).

Lopes (2006) observa que é desejável em gerações iniciais que a pressão seletiva seja mínima, permitindo que indivíduos com baixo valor de *fitness* tenham chances de serem escolhidos do processo seletivo. Já na fase final do AG os indivíduos tendem a ter valores de *fitness* muito próximos devido à convergência do AG, e nesta etapa é preferível que

a pressão seletiva seja máxima, de modo a induzir a evolução no sentido do ótimo global.

Métodos de seleção como o Método da Roleta que utiliza o valor de *fitness* dos indivíduos como base para o processo de escolha, criam uma pressão seletiva de maneira inversa, fornecendo uma alta pressão seletiva no início e uma baixa na etapa final.

Para solucionar este tipo de problema é possível a utilização da abordagem de escalonamento, o qual realiza um remapeamento do valor de *fitness* dos indivíduos antes da seleção. Uma das diferentes técnicas de escalonamento mais utilizada é a linear proposta por (GOLDBERG, 1989), que consiste em manter o *fitness* médio, atribuir ao *fitness* mínimo zero e o valor máximo seja a média multiplicada por uma constante (valor entre 1,20 e 2,00). Com isso os valores de *fitness* da população sempre estarão dentro de um intervalo predeterminado, independente do momento evolutivo (LOPES, 2006).

1.2.6 Teorias sobre AG

Algoritmos Genéticos são aplicados em inúmeros problemas de diversas áreas do conhecimento, pelos quais comprovou-se que o AG realmente funciona. Mas porque ele funciona? Em qual embasamento teórico ele se sustenta? Algumas pesquisas foram realizadas e em 1975 John Holland apresentou a Teoria dos Esquemas.

A Teoria dos Esquemas foi elaborada sobre a codificação binária, utilizando a *string* de bits e o alfabeto binário $\{0,1\}$.

Um esquema é um modelo (*template*) de um cromossomo que representa um conjunto de cromossomos. Segundo (MICHALEWICZ, 1996) um esquema (*schema*) é construído pela introdução do símbolo *don't care* (*) no alfabeto de genes, representando que naquela posição do esquema pode ser atribuído qualquer símbolo do alfabeto. O autor supracitado afirma que um esquema representa um hiperplano ou um subconjunto do espaço de busca.

Por exemplo, o esquema [01*1] representa os cromossomos [0101] e [0111]. Fica claro que o esquema [0001] representa um único esquema, [0001], já o esquema [***] representa todos os cromossomos de tamanho 3. Em termos gerais um esquema pode representar 2^r cromossomos, onde r é o número de símbolos *don't care* presentes no esquema (IYODA, 2000).

Alguns outros conceitos foram definidos por Holland (HOLLAND, 1975), como o conceito de ordem, o comprimento e *fitness* de um esquema. A ordem $o(H)$ é a quantidade de 0's e 1's contidos no esquema. O comprimento δ(H), é a diferença entre a última posição que contém 1 ou 0 e a primeira posição que contém um 1 ou 0. Por exemplo, o esquema H $=$ [01**1*1*] possui um comprimento δ(H) $=$ 7-1 $=$ 6 e ordem igual a o(H) $=$ 4.

O *fitness* de um esquema H em uma geração t, $eval(H,t)$ é a média dos valores de *fitness* dos cromossomos representados pelo esquema H (MICHALEWICZ, 1996).

A partir destas definições é possível prever a variação do número de esquemas H entre duas gerações consecutivas. Seja $m(H,t)$ e número de cromossomos representados pelo esquema H na geração t, α a média do valor de *fitness* para toda a população e $eval(H,t)$ o *fitness* do esquema H na geração t, a Eq. (1.10) é conhecida segundo (MICHALEWICZ, 1996) como equação do crescimento reprodutivo do esquema (*reproductive schema growth equation*), que define o número esperado de cópias do esquema H para a próxima geração.

$$m(H, t+1) = \frac{\alpha}{eval(H,t)} \cdot m(H,t) \tag{1.10}$$

Com base na Eq. (1.10) é possível concluir que haverá um aumento na amostragem de esquemas H se o *fitness* de H ($eval(H,t)$) for maior do que a média dos valores de *fitness* da população (α), ou seja, haverá um crescimento se H representar bons cromossomos (LACERDA; CARVALHO, 1999). E ainda, os operadores de *crossover* e mutação não afetaram, de maneira destrutiva, o esquemas curtos e de baixa ordem (IYODA, 2000). Como conseqüência desta Eq. (1.10) foi formulado o Teorema dos Esquemas.

Teorema 1.2.1 *(Teorema dos Esquemas) Esquemas curtos, com baixa ordem e com fitness acima da média da população, aumentam exponencialmente sua participação em gerações subseqüentes.*

Os esquemas bons e curtos recebem o nome de *blocos construtivos*. As informações destes blocos serão combinadas com outros blocos bons e curtos no decorrer das gerações e esta combinação obterá indivíduos de alta aptidão (LACERDA; CARVALHO, 1999). Esta hipótese é conhecida como *Hipótese dos Blocos Construtivos*.

Hipótese 1.2.1 *(Hipótese dos Blocos Construtivos) Um algoritmo genético apresenta um desempenho quase-ótimo através da justaposição de esquemas curtos, de baixa ordem e de alto desempenho chamados de blocos construtivos.*

Em alguns problemas a Hipótese dos Blocos Construtivos não é confirmada, nestes casos dois blocos construtivos combinados resultam em um cromossomo ruim. Este fenômeno é conhecido como decepção (*deception*) (MICHALEWICZ, 1996).

De acordo com (LACERDA; CARVALHO, 1999) este tipo de fenômeno ocorre com cromossomos com alta *epistasia*. Em AG entende-se por epistasia como o nível de interação entre genes de um cromossomo, quando o valor de um gene influencia o valor de outros genes. Problemas com esta característica são raramente encontrados na prática.

Iyoda (2000) afirma que a hipótese dos blocos construtivos não fornece uma explicação do motivo pelo qual o AG funciona, apenas indica os motivos pelos quais ele funciona para uma determinada classe de problemas.

Com o intuito de contornar este problema algumas alternativas foram sugeridas, como apresenta (MICHALEWICZ, 1996). Entre elas, sugere-se, caso haja um conhecimento prévio sobre a função objetivo realizar uma codificação apropriada, de modo a estimular a criação de blocos construtivos.

1.3 Aplicações

Algoritmo Genético é basicamente aplicado em problemas de otimização, estendendo a outros problemas apenas modelando o mesmo como um problema de otimização.

Aplicações em problemas de Processamento de Imagem foram investigados por (CENTENO et al., 2005) e (RAHNAMAYAN; TIZHOOSH; SALAMA, 2005). AG's foram aplicados à problemas de Engenharia de software por (DAI et al., 2003) e (WEGENER et al., 1997). Já (NODA; FREITAS; LOPES, 2000) e (FIDELIS; LOPES; FREITAS, 2000) utilizaram AG para mineração de dados.

Outras aplicações de AG foram propostas na literatura, algumas com sucesso menor comparado a outras técnicas, mas em outras obteve sucesso, mostrando que AG é fortemente indicado em problemas reais. AG são também combinados com outras técnicas como redes neurais, formando sistemas híbridos altamente robustos.

1.4 Conclusão

Neste capítulo foram apresentados vários conceitos da Computação Evolucionária, área essa que tem despertado grande interesse na comunidade científica devido sua grande aplicabilidade em problemas reais. Um foco maior foi dado nos Algoritmos Genéticos, descrevendo sua estrutura básica, suas etapas intermediárias, foram identificadas suas fragilidades e alternativas de contornar estas dificuldades, além da explanação de alguns aspectos teóricos do AG.

Mesmo com o grande avanço dos algoritmos genéticos nos últimos anos, algumas questões ainda permanecem em aberto, como identifica (MICHALEWICZ, 1996). O autor apresenta algumas direções que segundo ele terão uma grande atividade e significantes resultados em um futuro breve, são elas: Fundamentação Teórica, Otimização de Funções, Representação de Indivíduos e Operadores, Sistemas Auto-Adaptativos e AG Paralelos. Essas pesquisas têm por objetivo explicar e melhorar cada vez mais esta técnica.

Referências

CENTENO, T. M. et al. Applications on evolutionary computing. In: _____. [S.l.]: Springerlink, 2005. cap. Object Detection for Computer Vision Using a Robust Genetic Algorithm, p. 284–293.

DAI, Y. S. et al. Optimal testing-resource allocation with genetic algorithm for modular software systems. *Journal of Systems and Software*, v. 66, n. 1, p. 47–55, 2003.

DEJONG, K. A. *An Analisysof the Behavior of a Class of Genetic Adaptive Systems*. Tese (Doutorado) — University of Michigan, 1975.

DRÉO, J. et al. *Metaheuristics for Hard Optimization: Methods and Case Studies*. [S.l.]: Springer, 2005.

FIDELIS, M. V.; LOPES, H. S.; FREITAS, A. A. Discovering comprehensible classification rules with a genetic algorithm. In: *Proceedings of the 2000 Congress on Evolutionary Computation*. [S.l.: s.n.], 2000. p. 805–810.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. [S.l.]: Addison-Wesley, 1989.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. 2. ed. [S.l.]: The MIT Press, 1975.

IYODA, E. M. *Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas*. Dissertação (Mestrado) — Universidade Estadual de Campinas (Unicamp), Campinas, São Paulo, 2000. Disponível em: <http://www.dca.fee-.unicamp.br/~vonzuben/research/emi_mest.html>.

LACERDA, E. G. M.; CARVALHO, A. C. Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais. In: _____. Porto Alegre, RS: Universidade/UFRGS, 1999. cap. Introdução aos Algoritmos Genéticos, p. 99–150. Disponível em: <http://www.dca.ufrn.br-/~estefane/metaheuristicas/ag.pdf>.

LOPES, H. S. *Fundamentos da Computação Evolucionária e Aplicações*. Bandeirantes,Paraná, 2006. 52-106 p.

MICHALEWICZ, Z. *Genetics Algorithms + Data Structures = Evolution Programs*. 3. ed. New York: Springer-Verlag Berlin Hildelberg, 1996.

MITCHELL, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. London, England: The MIT Press, 1996.

NODA, E.; FREITAS, A. A.; LOPES, H. S. Comparing a genetic algorithm with a rule induction algorithm in the data mining task of dependence modeling. In: *Genetic and Evolutionary Coomputation Conference*. [S.l.: s.n.], 2000. p. 1080.

OBITKO, M. *Introduction to Genetic Algorithms*. 1998. Website. Disponível em: <http://obitko.com/tutorials/genetic-algorithms>.

RAHNAMAYAN, S.; TIZHOOSH, H.; SALAMA, M. Adaptive and natural computing algorithms. In: _____. [S.l.]: Springerlink, 2005. cap. Learning Image Filtering from a Gold Sample Based on Genetic Optimization of Morphological Processing, p. 478 – 481.

TANOMARU, J. Motivação, fundamentos e aplicações de algoritmos genéticos. *Anais do II Congresso Brasileiro de Redes Neurais*, 1995.

WEGENER, J. et al. Testing real-time systems using genetic algorithms. *Software Quality Journal*, v. 6, n. 2, p. 127–135, 1997.