

**DANIEL ALEXANDRE CORDEIRO**  
**MARCOS AURÉLIO CARRERO**

**COMPUTAÇÃO EVOLUTIVA**

Trabalho da disciplina de Inteligência  
Artificial, Setor de Ciências Exatas,  
Universidade Federal do Paraná.  
Profa. Aurora R. Pozo

CURITIBA  
2002

## Índice

1 Introdução.....	5
2 Teoria Da Seleção Natural.....	6
3 Algoritmos Genéticos.....	7
4 Fundamentos dos AGs.....	7
4.1 Uma Definição.....	7
4.2 Aplicação dos Algoritmos Genéticos.....	8
4.3 Representação Cromossômica.....	11
4.4 Reprodução.....	12
4.5 Seleção.....	13
4.5.2 Seleção por Classificação.....	15
4.5.3 Seleção de Estado Seguro.....	16
4.5.4 Seleção por Torneio ou Elitismo.....	17
4.6 Cruzamento.....	17
5 Representação Binária.....	18
5.1 Operações da Representação binária.....	18
5.2 Cruzamento num Ponto.....	18
5.3 Cruzamento Multi-Ponto.....	19
5.4 Cruzamento uniforme.....	20
6 Mutação.....	20
6.1 Mutação Bit.....	21
6.2 Mutação Uniforme.....	21
7 Representação Numérica.....	22
7.1 Operações da Representação Numérica.....	22
7.2 Cruzamento por média.....	22
7.3 Mutação Aleatória.....	23
7.4 Mutação por incremento.....	23
8 Representação com Base na Ordem (Permutação).....	24
8.1 Operações da Representação baseada na ordem.....	24
8.2 Cruzamento baseado na ordem.....	24
8.3 Mutação baseada na ordem.....	25
9 Codificação em Árvore.....	25
10 Como funciona um AG.....	26
11 Um Exemplo de Algoritmo Genético.....	29
11.1 Características do Algoritmo.....	29
11.2 O Problema de Otimização.....	30
12 Uso da Tecnologia de AGs.....	33
13 Parâmetros dos Algoritmos Genéticos.....	34
14 Programação Genética.....	36
14.1 Cruzamento.....	37
14.2 Mutação.....	38
15 O Programa Dougal.....	39
15.1 Descrição do programa.....	39

## Índice de Figuras

Figura 1 - Esquema de Iteração de Um Algoritmo Genético Básico.....	9
Figura 2 - Fluxograma do ciclo de um Algoritmo Genético simples.....	11
Figura 3 - Cromossomo em binário.....	12
Figura 4 - População de strings e seus valores de fitness.....	12
Figura 5 - Método da Roleta Giradora.....	13
Figura 6 - Situação antes da Classificação.....	16
Figura 7 - Situação após a classificação.....	16
Figura 8 - Esquema gráfico do processo de cruzamento.....	18
Figura 9 - Cromossomos em binário provenientes da Seleção.....	19
Figura 10 - No cruzamento uni-ponto, são trocados os genes de um cromossomo a partir de uma posição aleatória.....	19
Figura 11 - Os Cromossomos resultantes do cruzamento.....	19
Figura 12 - No cruzamento multi-ponto são trocados os genes de um cromossomo entre duas posições aleatórias.....	19
Figura 13 - Cromossomos resultantes do cruzamento.....	20
Figura 14 - A máscara gerada aleatoriamente determina quais os genes dos Cromossomos a serem trocados.....	20
Figura 15 - Cromossomos resultantes do cruzamento uniforme.....	20
Figura 16 - Esquema do processo de mutação.....	21
Figura 17 - cromossomo com representação numérica.....	22
Figura 18 - cromossomo resultante do cruzamento por média de dois Cromossomos com representação numérica.....	23
Figura 19 - Mutação numérica aleatória. Os genes mudados são substituídos por um valor gerado ao acaso.....	23
Figura 20 - Mutação por incremento. O valor dos genes matados é obtido somando um pequeno incremento gerado aleatoriamente.....	23
Figura 21 - Exemplo de 2 Cromossomos com representação baseada na ordem.....	24
Figura 22 - Cruzamento baseado na ordem. As posições a trocar num cromossomo, são as que ficarão fixas no outro.....	24
Figura 23 - Exemplo de cromossomos codificados em árvore.....	26
Figura 24 - Foram selecionados o segundo e quarto cromossomos.....	28
Figura 25 - Dois Cromossomos antes do cruzamento (à esquerda) e depois (à direita). As secções a sombreado são aquelas que serão trocadas.....	28
Figura 26 - Analogia entre os Algoritmos Genéticos e a Seleção Natural.....	29
Figura 27 - Exemplo de um programa de computador representado em árvore. Em PG, cada árvore equivale a um indivíduo, e uma população de programas evolue de modo análogo aos AGs.....	36
Figura 28 - Duas árvores selecionadas para o cruzamento.....	37
Figura 29 - Árvores resultantes do cruzamento.....	37
Figura 30 - Exemplo de mutação onde é feita a troca de um operador aritmético.....	38
Figura 31 - Mutação onde é trocado o ramo da árvore por um outro aleatório.....	38



## 1 Introdução

As pesquisas sobre modelos computacionais inteligentes têm, nos últimos anos, se caracterizado por buscar inspiração na natureza. Existem muitas soluções para problemas complexos de adaptação encontrados na natureza, que nos fornecem modelos interessantíssimos.

Cientistas, físicos e biólogos tentam desvendar os princípios que regem os fenômenos da natureza, enquanto que os matemáticos, cientistas da computação e engenheiros buscam idéias que possam ser copiadas ou pelo menos imitadas, e então aplicadas a problemas que a ciência atual não pode resolver satisfatoriamente. Por exemplo, o fenômeno de anelamento de metais, a enorme capacidade de processamento do sistema nervoso central de animais superiores e o processo estocástico de evolução natural, fornecem inspiração para os paradigmas de computação denominados Anelamento Simulado, Redes Neurais Artificiais e Computação Evolucionária (CE), respectivamente.

A CE encara a teoria de evolução Darwiniana como um processo adaptativo de otimização, sugerindo um modelo em que populações de estruturas computacionais evoluem de modo a melhorar, em média, o desempenho geral da população.

A idéia de CE foi introduzida em 1960 por Rechenberg em seu trabalho "Estratégias Evolutivas" (o título original era: Evolutionsstrategie).

Atualmente, CE engloba um número crescente de paradigmas e métodos, dos quais os mais importantes são os Algoritmos Genéticos (AGs), Programação Evolucionária (PE), Estratégias Evolucionárias (EEs), Programação Genética (PG), e Sistemas Classificadores (SCs), entre outros. De fato, SCs e PG podem ser vistas como aplicações especiais de AGs, enquanto que PE e EEs foram desenvolvidas independentemente dos AGs.

Em 1992, John Koza, professor e pesquisador da Universidade de Stanford, utilizou Algoritmos Genéticos para solucionar certos problemas. Ele chamou seu método de Programação Genética (PG). Foi usada a linguagem de programação LISP por permitir que os programas sejam expressos na forma de "árvores analisadas", as quais são os objetos de trabalho dos Algoritmos Genéticos.

## 2 Teoria Da Seleção Natural

Em 1859 Charles Darwin, após a sua expedição às ilhas Galápagos a bordo do Beagle formulava, no seu livro *The origin of species* a Teoria da Evolução, também conhecida pela Teoria da Seleção Natural.

O meio ambiente tem a sua ação implacável sobre a evolução das espécies: sobrevivem apenas os indivíduos melhores adaptados ao seu habitat. Apenas estes têm a oportunidade de deixar descendência, perpetuando as suas características.

Esporadicamente, surge um indivíduo com alguma característica que o distingue dos restantes elementos da sua espécie - indivíduo mutado. Se esta nova característica oferecer vantagens competitivas, aumentará a probabilidade deste indivíduo sobreviver e de passá-la à sua descendência. Com o tempo, a nova característica acabará por se impor na espécie, uma vez que esta confere aos indivíduos melhores hipóteses de sobrevivência: num mundo com recursos limitados, a competição não perdoa os mais fracos.

E assim, geração após geração a espécie evolui no sentido de melhor se adaptar ao seu meio ambiente.

Mais tarde, a Genética daria o seu suporte a esta teoria. Os seres vivos têm, no núcleo da suas células, moléculas de DNA onde estão gravadas as suas características, mais precisamente em Genes organizados em cadeia - os Cromossomos.

Durante a reprodução sexual, os conteúdos dos Cromossomos dos dois pais, são misturados, gerando novos Cromossomos que combinam as características dos seus progenitores.

Menos freqüentemente ocorrem mutações no código genético. São pequenas alterações acidentais na cadeia de genes, que poderão ter repercussões nas características do novo ser. Algumas são bastante prejudiciais (como a anemia, ou o síndrome de Down), muitas são pouco influentes (pessoas esquerdinas, capacidade de dobrar a língua). Mas outras foram benéficas, são estas as responsáveis pela evolução de toda a vida na terra, incluindo a nossa espécie.

### 3 Algoritmos Genéticos

A evolução natural implementa mecanismos adaptativos de otimização que, embora estejam longe de serem uma forma de busca aleatória, com certeza envolve aleatoriedade.

Embora Charles Darwin tenha formulado a Teoria da Evolução no final do século passado, foi só recentemente que se tentou idealizar um modelo matemático do processo evolutivo. Nos anos 60, John Holland, da Universidade de Michigan, começou a definir as bases de algoritmos de otimização de inspiração genética, criando os AGs. Seu trabalho culminou na publicação do livro *Adaptação em Sistemas Naturais e Artificiais* em 1975.

Os AGs tinham dois objetivos fundamentais:

- Explicar os processos adaptativos dos sistemas naturais;
- Conceber sistemas artificiais de acordo com as propriedades de tais sistemas naturais.

Como se pode supor, os algoritmos genéticos foram criados tendo como referência à teoria de Darwin sobre a evolução dos seres vivos. Dessa forma, pode-se dizer que soluções obtidas através de algoritmos genéticos são evolutivas.

A grande popularidade que os AGs atingiram recentemente se deve a dois fatores importantes: a publicação de um livro tutorial sobre AGs por um dos alunos de doutorado de Holland, David Goldberg, e às Conferências Internacionais sobre Algoritmos Genéticos (IC- GAs) .

## 4 Fundamentos dos AGs

### 4.1 Uma Definição

*Algoritmos Genéticos (AGs) são métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em AGs, uma população de possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.*

## **4.2 Aplicação dos Algoritmos Genéticos**

John Holland, inventor dos AGs, propôs utilizar a abordagem da natureza para a resolução de problemas, tendo em conta os seguintes fatos:

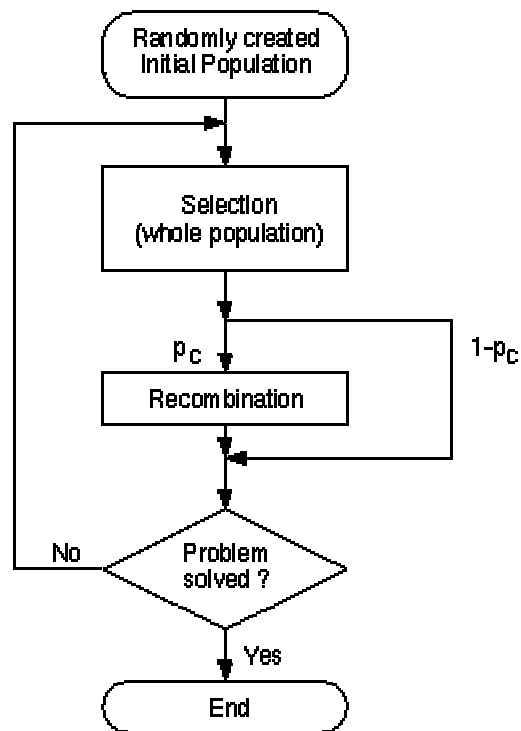
Todos os seres vivos têm codificadas as suas características nos seus cromossomos. Estes vivem em populações e, pela reprodução sexual, os cromossomos das crias são uma recombinação dos seus progenitores.

De tempos a tempos, surgem mutações nesse código, por vezes com profundas alterações para os indivíduos.

A seleção natural tem um papel decisivo na evolução das espécies, dando apenas hipóteses de sobrevivência aos indivíduos melhor adaptados (em Inglês, fit) ao seu meio ambiente.

O AG básico realiza, segundo BITTENCOURT (1998), possui seguintes funções:





*Figura 1 - Esquema de Iteração de Um Algoritmo Genético Básico*

O fluxograma acima mostra o ciclo de iteração de um algoritmo genético. Primeiramente, uma população inicial de strings é criada. O processo iterativamente seleciona os melhores indivíduos das populações que surgiram através das transformações (via recombinação) para criar novas populações. Assim, a nova população é testada para verificar se atende a um critério de parada. Se sim, então o processo pára, caso contrário, um outra iteração é realizada.

#### **Esboço de um Algoritmo Genético Básico:**

1. **[Início]** Geração aleatória de uma população de  $n$  cromossomos.
2. **[Adaptação]** Avaliação do grau de adaptação  $f(x)$  de cada cromossomo  $x$  de uma população.
3. **[Nova População]** Criação de uma nova população através da repetição dos seguintes passos até que a população esteja formada.

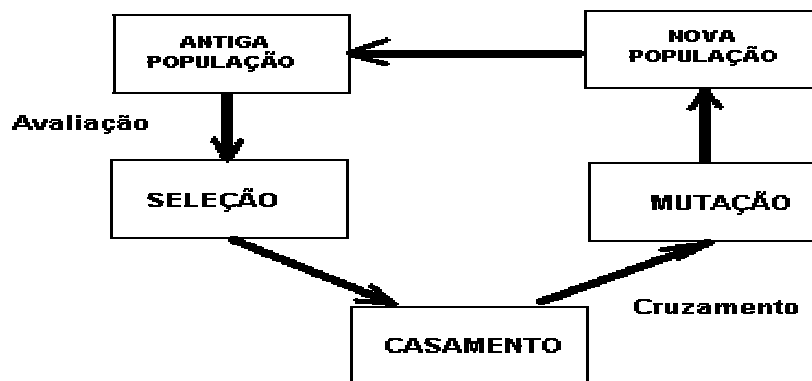
1. **[Seleção]** Selecionar dois cromossomos pais da população de acordo com seus valores de adaptação (os mais adaptados têm maior chance de serem selecionados).
2. **[Cruzamento]** Com a taxa de cruzamento, cruzar os cromossomos selecionados no item anterior para formar as novas strings filhas. Se nenhum cruzamento foi executado, as strings filhas serão cópias exatas das progenitoras.
3. **[Mutação]** Com a taxa de mutação, realizar mutações em cada locus (posição em um cromossomo) dos novos cromossomos produzidos.
4. **[Aceitação]** Inserir os novos cromossomos na nova população.
4. **[Substituição]** Substituição da antiga população pela nova população gerada.
5. **[Teste]** Verificação se a condição de parada foi satisfeita. Se sim, parar a execução e retornar a melhor solução da população atual.
6. **[Loop]** Voltar ao passo 2.

Um processo de otimização consiste em procurar melhorar a performance, com o objetivo de alcançar um ou vários pontos ótimos. É desta forma que funcionam os Algoritmos Genéticos. Eles combinam a sobrevivência do mais adaptado, com uma troca de informações ao mesmo tempo aleatória e estruturada. Os AGs trabalham sobre uma população de palavras ou cromossomos (*strings*), que são seqüências de códigos, geralmente de forma binária, que representam determinados parâmetros. Para se criar uma população de palavras, são aplicadas sucessivas operações de reprodução, cruzamento e/ou mutação.

Segundo Goldman (1989), os Algoritmos Genéticos têm quatro aspectos que os diferem dos outros métodos tradicionais de otimização, e que são:

- Trabalham sobre uma codificação de parâmetros e não diretamente sobre os parâmetros do problema;
- Operam em uma população e não em pontos isolados, reduzindo assim o risco de busca a falsos pontos;
- Usam informações da função objetiva (*payoff*) e não derivadas de outros conhecimentos auxiliares;
- Procedem a buscas utilizando operadores estocásticos e sua regra de transição é probabilística e não determinística.

A seguir vê-se um fluxograma simplificado das operações sequenciadas nos Algoritmos Genéticos simples, na *Figura 2*.



*Figura 2 - Fluxograma do ciclo de um Algoritmo Genético simples*

Para que o entendimento pleno do funcionamento dos Algoritmos Genéticos, é necessário sumarizar a apresentação destas três operações, além de aspectos da seleção e codificação (representação).

### 4.3 Representação Cromossômica

O primeiro passo para a aplicação de AGs a um problema qualquer é representar cada possível solução  $x$  no espaço de busca como uma sequência de símbolos  $s$  gerado a partir de um dado alfabeto finito  $A$ . No caso mais simples, usa-se o alfabeto binário  $A = \{0,1\}$ , mas no caso geral tanto o método de representação quanto o alfabético genético dependem de cada problema. A grande maioria dos AGs propostos na literatura usam uma população de número fixo de indivíduos, com cromossomos também de tamanho constante.

Na representação binária, o cromossomo é uma sequência de zeros e de uns.

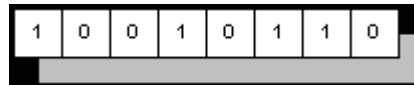


Figura 3 - Cromossomo em binário

#### 4.4 Reprodução

A reprodução em um AG simples é o processo no qual as palavras ou cromossomos (*strings*) individuais são copiados de acordo com os valores dados pela função objetivo, que na Biologia se denomina de função aptidão (*fitness*). Esta função objetivo é a medida de utilidade que se deseja maximizar. Os cromossomos (*strings*) com altos valores possuem maior probabilidade de formar a próxima geração. Este operador é um modelo artificial da seleção natural. Nas populações naturais a aptidão (*fitness*) é determinada pela capacidade que os organismos têm de lutar pela sobrevivência, resistindo a predadores, doenças, intempéries, falta de alimentos e outros obstáculos. No meio artificial, a função de avaliação é que decide quais os cromossomos (*strings*) que sobreviverão e quais morrerão.

Exemplificando, o resultado da decodificação binária da *string* 10101 representa a sua aptidão, ou *fitness*, e a função objetivo (*payoff*) poderia maximizar este valor, conforme o quadro apresentado a seguir na Figura 4:

$10101 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 21$ (fitness)	
POPULAÇÃO	"FITNESS"
0 0 0 1	1
0 1 1 0	6
1 0 1 0	12

Figura 4 - População de strings e seus valores de fitness

#### 4.5 Seleção

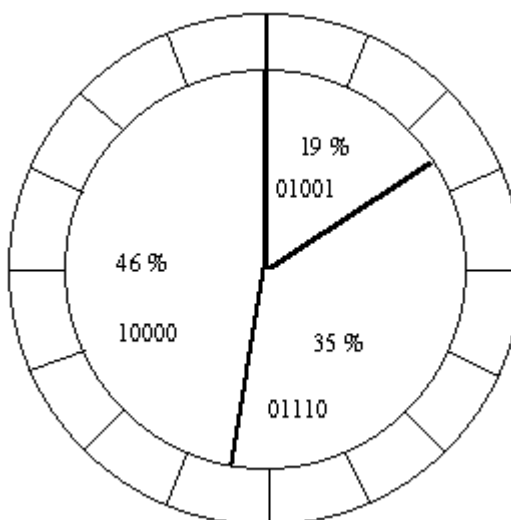
Durante a operação de reprodução, os cromossomos são selecionados para gerarem novas populações. O problema é como selecionar estes cromossomos. De acordo com a

teoria da evolução de Darwin os melhores deveriam sobreviver e formar as gerações futuras. Há vários métodos para se selecionar os melhores cromossomos, por exemplo, Seleção da Roleta Giradora, Seleção de Boltzman, Seleção por Torneio, Seleção por Classificação, Seleção de Estado Seguro entre outras.

Ver-se-á agora os métodos mais comuns de seleção em algoritmos genéticos.

#### 4.5.1 Roulette Wheel - "Roleta Giradora"

O *Método da Roleta Giradora*, "Roulette Wheel", simplesmente escolhe uma string de acordo com a porção ocupada por esta em uma roleta, isto é, de acordo com um valor de adaptação relativo a esta roleta (em porcentagem). Para visualizar o funcionamento deste método, observe a figura abaixo:



*Figura 5 - Método da Roleta Giradora*

Para selecionar três strings que serão utilizadas como reprodutoras, a roleta giradora será girada três vezes. Os resultados obtidos indicarão as strings selecionadas. É óbvio, neste caso, que o string 10000 será selecionado mais vezes. E isto é bom. Múltiplas cópias de uma mesma strings podem existir no grupo de strings reprodutoras. Isto é até desejável, pois as mais fortes começarão a prevalecer erradicando as mais fracas. Há, entretanto, certas dificuldades provenientes deste fato, pois isto pode conduzir a uma *convergência prematura a um valor ótimo local*.

Esse método pode ser dividido em duas partes: na primeira calcula-se a percentagem da roleta que vai caber a cada cromossomo; na segunda faz-se rodar a roleta tantas vezes quantas o número de cromossomos da população.

Para partilhar a roleta pelos cromossomos seguem-se os seguintes passos:

- Calcula-se o valor da função objetivo  $f$  para cada cromossomo  $v_i$  tal que  $(i = 1, 2, \dots, \#População)$
- Calcula-se o valor da função objetivo para a população tomada como um todo:

$$T = \sum_{i=1}^{\#População} f(v_i)$$

- Calculam-se as probabilidades de seleção para cada cromossomo  $v_i$ :

$$p_i = \frac{f(v_i)}{T}$$

- Calculam-se as probabilidades cumulativas  $q_i$  para cada cromossomo:

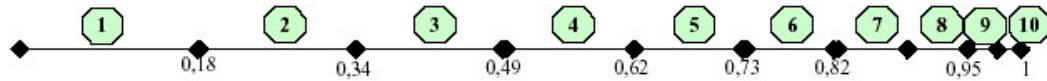
$$q_i = \sum_{j=1}^i p_j$$

Para simular cada rotação da roleta escolhe-se aleatoriamente um número  $r$  no intervalo  $[0,1]$ . Se  $r < q_1$  então adiciona-se um cromossomo igual ao primeiro à nova população que se está pra gerar. Caso contrário seleciona-se o cromossomo  $v_i$  que verifique:

$$(2 \leq i \leq \#População) \wedge (q_{i-1} < r < q_i).$$

Vejamos o seguinte exemplo de uma população com 10 indivíduos:

NÚMERO do indivíduo	1	2	3	4	5	6	7	8	9	10
Função de Mérito $=f(x)$	2.0	1.8	1.6	1.4	1.2	1.0	0.8	0.6	0.4	0.2
Probabilidade de Seleção $f(x_i)/\sum f(x_i)$	0.18	0.16	0.15	0.13	0.11	0.09	0.07	0.06	0.03	0.02



Suponhamos que se pretendem obter 6 progenitores. Começamos por gerar seis números aleatórios. Admitamos que se obtêm os seguintes:

0.81 0.32 0.96 0.01 0.65 0.42

Assim, os indivíduos selecionados para gerar a descendência serão, respectivamente.

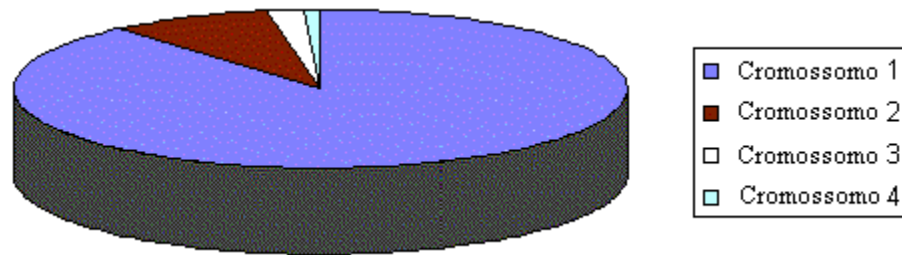
6, 2, 9, 1, 5 e 3

#### 4.5.2 Seleção por Classificação

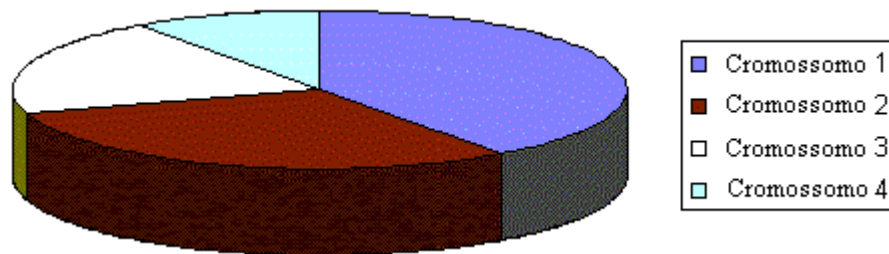
O método anterior apresentará problemas quando os valores de adaptação forem muito diferentes. Por exemplo, se o valor de adaptação do melhor cromossomo for 90% de toda a roleta, os demais cromossomos terão poucas chances de serem selecionados.

A *Seleção por Classificação* primeiramente classifica a população, depois cada cromossomo recebe um valor de acordo com esta classificação. O pior terá valor 1, o segundo pior terá valor 2 e assim sucessivamente. O melhor terá valor N igual ao número de indivíduos da população.

Pode-se ver nas próximas figuras como a situação se altera após ocorrer a classificação dos cromossomos:



*Figura 6 - Situação antes da Classificação*



*Figura 7 - Situação após a classificação*

Após a classificação todos os cromossomos têm chances de serem selecionados. Este método pode apresentar convergência muito lenta porque os melhores cromossomos não diferem muito dos outros.

#### **4.5.3 Seleção de Estado Seguro**

A principal idéia deste tipo de seleção é que partes grandes dos cromossomos devem ser transferidas para as próximas gerações.

Neste método, os AGs trabalham da seguinte forma: a cada geração são selecionados poucos (porém com altos valores de adaptação) cromossomos para gerar os filhos. Alguns cromossomos (os piores, com baixos valores de adaptação) são retirados da população e os cromossomos filhos são colocados em seus lugares. O resto da população permanece inalterada na composição da nova geração.



#### 4.5.4 Seleção por Torneio ou Elitismo

Quando se criam novas populações através de cruzamento e mutação, tem-se uma grande chance de perder o melhor cromossomo.

Neste método, primeiramente copia-se o melhor cromossomo (ou os melhores cromossomos) para a nova população. O restante é realizado do modo normal. O Elitismo pode aumentar rapidamente o desempenho do AG porque evita a perda da melhor solução.

#### 4.6 Cruzamento

O cruzamento ou corte (*crossover*) ocorre em duas etapas, sendo a primeira onde se determinam aleatoriamente quais os *strings* que casarão e, em seguida, também aleatoriamente, em que posição do *string* se processará o cruzamento (*crossover*).

Exemplificando, podem ser considerados dois strings "A" e "B", com as seguintes codificações binárias:

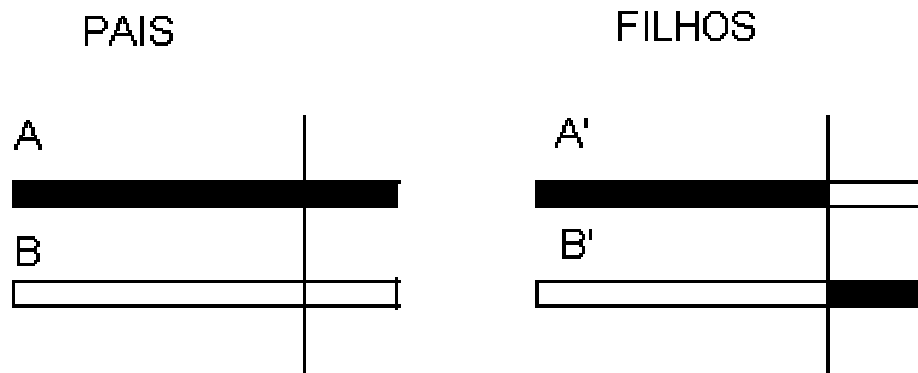
"A" = 0 1 1 0 1

"B" = 1 1 0 0 0

Se for escolhida a posição de corte em  $K = 4$ , ou seja, após a quarta posição da *string*, após o cruzamento ter-se-á a seguinte configuração, em decorrência da troca, conforme destacado em negrito:

"A" = 0 1 1 0 0

"B" = 1 1 0 0 1

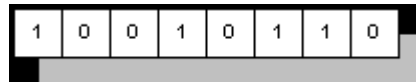


*Figura 8 - Esquema gráfico do processo de cruzamento*

O esquema acima pode igualmente ser representado por uma analogia gráfica, conforme o mostrado na *Figura B* a seguir.

## 5 Representação Binária

Na representação binária, o cromossomo é uma sequência de zeros e de uns.

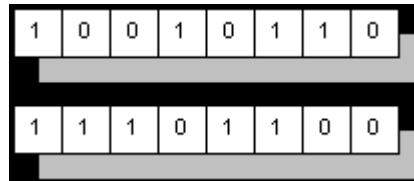


### 5.1 Operações da Representação binária

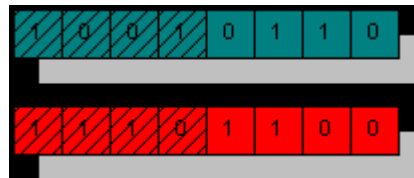
Já referimos as duas operações que se realizam no cromossomo – o cruzamento e a mutação. No cruzamento é feita a troca de algumas posições entre dois Cromossomos. A mutação, essa já é feita pela alteração de algumas posições num cromossomo.

### 5.2 Cruzamento num Ponto

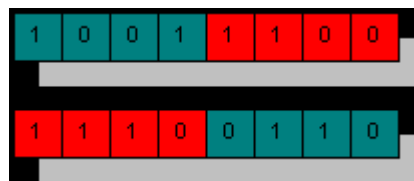
No cruzamento uni-ponto, o conteúdo dos Cromossomos é trocado a partir de uma posição escolhida aleatoriamente.



*Figura 9 - Cromossomos em binário provenientes da Seleção.*



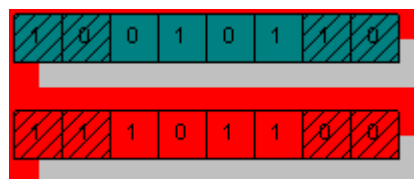
*Figura 10 - No cruzamento uni-ponto, são trocados os genes de um cromossomo a partir de uma posição aleatória.*



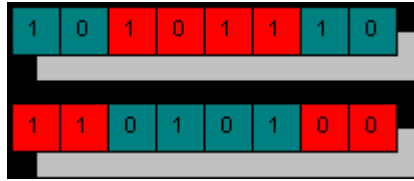
*Figura 11 - Os Cromossomos resultantes do cruzamento.*

### 5.3 Cruzamento Multi-Ponto

Neste cruzamento a troca de valores é feita entre duas posições, contrariamente ao cruzamento uni-ponto que troca tudo a partir de uma posição.



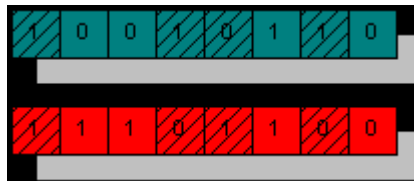
*Figura 12 - No cruzamento multi-ponto são trocados os genes de um cromossomo entre duas posições aleatórias.*



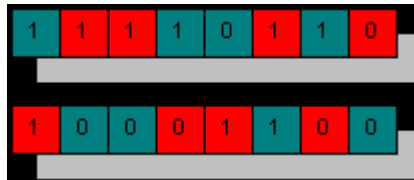
*Figura 13 - Cromossomos resultantes do cruzamento.*

#### 5.4 Cruzamento uniforme

No cruzamento uniforme, é criada uma máscara que indica os genes dos dois cromossomos que vão ser trocados.



*Figura 14 - A máscara gerada aleatoriamente determina quais os genes dos Cromossomos a serem trocados.*

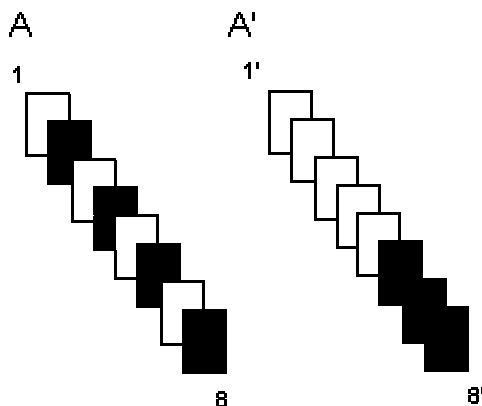


*Figura 15 - Cromossomos resultantes do cruzamento uniforme.*

### 6 Mutação

A mutação é uma alteração aleatória e ocasional do valor de uma posição qualquer da *string*. Esta alteração ocorre de acordo com uma probabilidade prefixada e, por exemplo, no caso de uma *string* binária, poderia significar a mudança de "1" para "0" ou de "0" para "1". Esta probabilidade de mutação, como a probabilidade de cruzamento, e também o tamanho da população, são elementos muito importantes nos Algoritmos Genéticos. Eles controlam todo o processo de busca, influenciando diretamente a velocidade de convergência, e evitando que aconteça a supremacia de uma determinada sub-população, o que geraria o chamado elitismo.

Como exemplo, pode ser verificado na *Figura 16*, uma analogia do processo de mutação, onde é alterada a cor dos *strings* nas posições "2", "4" e "7", da situação "A" para a situação "A'".



*Figura 16 - Esquema do processo de mutação*

## 6.1 Mutação Bit

Na mutação por bit é simplesmente alterado um gene aleatório do cromossomo.



## 6.2 Mutação Uniforme

À semelhança do cruzamento uniforme, na mutação uniforme é também criada uma máscara. Esta máscara indica os genes a serem mortos.



A codificação binária não é natural para alguns problemas e as vezes são necessárias algumas correções após as operações de cruzamento e/ou mutação.

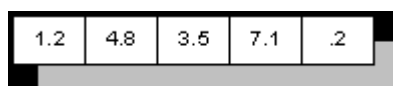
**Exemplo de Problema:** O problema da mochila.

**Descrição:** Existem vários objetos com valores associados e tamanhos conhecidos. É dada a capacidade da mochila. Deve-se selecionar um conjunto de objetos para serem colocados na mochila de modo a maximizar a soma dos valores dos objetos sem ultrapassar a capacidade da mochila.

**Codificação:** Cada bit de uma string representaria a ausência ou presença de um objeto na mochila.

## 7 Representação Numérica

Esta representação está mais bem adaptada para problemas de afinação de parâmetros numéricos. Os Cromossomos são cadeias de números.



*Figura 17 - cromossomo com representação numérica.*

### 7.1 Operações da Representação Numérica

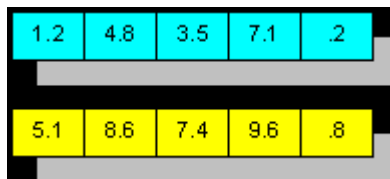
Como as operações de cruzamento se baseiam nas trocas de genes dos Cromossomos (independentemente destes serem valores binários, inteiros ou fracionários), os três tipos de cruzamento definidos anteriormente para a representação binária podem, também, ser utilizados na representação numérica.

Há, no entanto, um outro tipo cruzamento possível que aproveita a natureza numérica dos Cromossomos – o cruzamento por média.

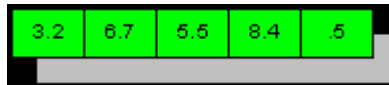
Em relação às operações de mutação, as diferenças já são muito notórias. A mutação de um gene em binário corresponde à sua inversão (de 0 para 1, ou vice versa). A mutação de um valor numérico já implica a substituição por um outro valor. Esse novo valor pode ser obtido de diferentes maneiras.

### 7.2 Cruzamento por média

Este tipo de cruzamento, parte de dois Cromossomos para produzir um único. Os genes do cromossomo filho são o resultado da média dos genes dos Cromossomos seus progenitores.



=



*Figura 18 - cromossomo resultante do cruzamento por média de dois Cromossomos com representação numérica.*

### 7.3 Mutação Aleatória

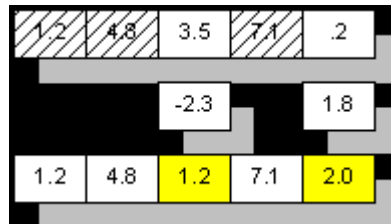
Na mutação aleatória, o valor do gene mutado é substituído por um valor aleatório (dentro de um intervalo de validade pré-definido).



*Figura 19 - Mutação numérica aleatória. Os genes mudados são substituídos por um valor gerado ao acaso.*

### 7.4 Mutação por incremento

Por sua vez, na mutação por incremento, em vez de substituir o valor de um gene por outro aleatório, soma-se um pequeno incremento (esse sim aleatório).



*Figura 20 - Mutação por incremento. O valor dos genes matados é obtido somando um pequeno incremento gerado aleatoriamente*

Esta operação altera os cromossomos de uma forma mais suave, o que a torna, em relação à mutação aleatória, melhor adaptada a otimizações mais finas. A codificação por valor é muito boa para alguns tipos de problemas. Por outro lado, para este tipo de codificação é necessário desenvolver novas operações de cruzamento e mutação para cada problema em questão.

**Exemplo de Problema:** Encontrar o peso de uma Rede Neural.

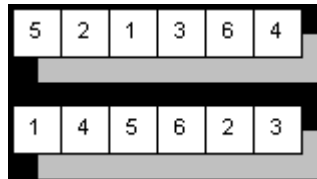
**Descrição:** Há uma rede neural com a arquitetura dada. Encontrar os pesos associados às entradas dos neurônios para treinar a rede para a saída esperada.

**Codificação:** Números reais nos cromossomos representariam os correspondentes pesos das entradas.

## 8 Representação com Base na Ordem (Permutação)

A representação baseada na ordem é utilizada para problemas de natureza combinatória, e não numérica.

Um exemplo típico é o problema do *caixeiro viajante*. Caixeiro viajante tem de percorrer uma série de cidades (uma única só vez) minimizando a distância percorrida. Neste caso, lidamos com a ordem pela qual são percorridas as cidades.



*Figura 21 - Exemplo de 2 Cromossomos com representação baseada na ordem.*

Um outro tipo de problema que sugere uma representação baseada na ordem é o do *Coloring Graph Problem*, também implementado por nós (ver adiante).

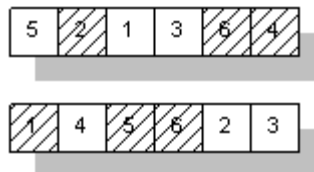
### 8.1 Operações da Representação baseada na ordem

As operações sobre uma representação baseada na ordem são diferentes daquelas definidas para a representação binária ou numérica.

Isto acontece pela natureza intrínseca desta representação. Nenhum valor pode ser repetido, pelo que os Cromossomos matados ou resultantes do cruzamento são permutações dos originais.

### 8.2 Cruzamento baseado na ordem

Inicialmente, cria-se uma máscara que indica quais os genes a manter, e aqueles cuja ordem será modificada. As posições a manter num cromossomo correspondem às que serão alteradas no outro.



*Figura 22 - Cruzamento baseado na ordem. As posições a trocar num cromossomo, são as que ficarão fixas no outro.*



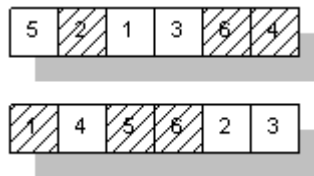
Os genes cujas posições são para alterar serão permutados de forma a ficarem na ordem que aparecem no outro cromossomo.



Os dois Cromossomos depois do cruzamento. Os genes 5, 1 e 3, que no primeiro cromossomo são para mudar, aparecem agora na ordem em que estavam no segundo cromossomo: 1, 5 e 3. Da mesma forma, os genes 4, 2 e 3, do segundo cromossomo, ficaram na ordem em que se encontravam no primeiro cromossomo 2, 3 e 4.

### 8.3 Mutação baseada na ordem

A mutação baseada na ordem é um simples baralhar das posições do gene.



A codificação por permutação somente é usada para problemas de ordenação. Mesmo para estes problemas, alguns tipos de correções, precisam ser feitas após as operações de cruzamento e mutação para deixarem os cromossomos consistentes (possuir um sequência real de números).

**Exemplo de Problema:** O problema do caixeiro viajante.

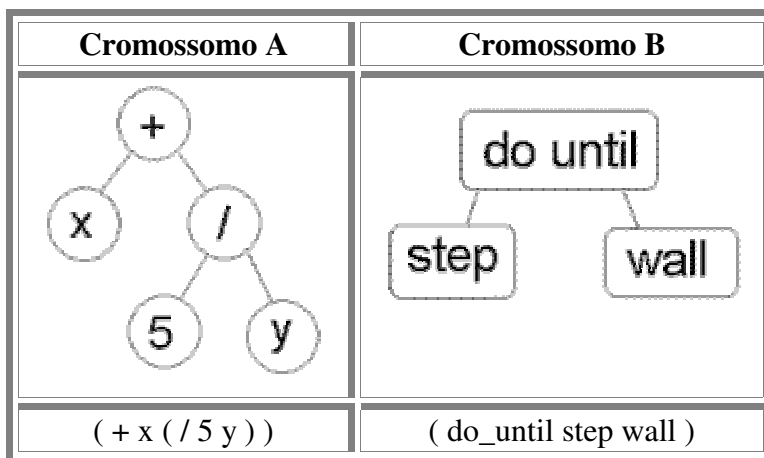
**Descrição:** Existem várias cidades e são conhecidas as distâncias entre elas. O caixeiro viajante precisa visitar todas as cidades percorrendo o menor caminho possível.

**Codificação:** Os cromossomos representariam a ordem das cidades que devem ser visitadas.

## 9 Codificação em Árvore

A codificação em árvore é usada principalmente em problemas que envolvem expressões, para programação genética.

Na codificação em árvore cada cromossomo é uma árvore de alguns objetos, tais como funções ou comandos de uma linguagem de programação.



*Figura 23 - Exemplo de cromossomos codificados em árvore*

A codificação em árvore é usada por problemas de desenvolvimentos de programas. A linguagem de programação LISP é freqüentemente usada para isto porque os programas em LISP são representados por árvores e podem ser facilmente analisados. As operações de cruzamento e mutação podem ser feitas com relativa facilidade.

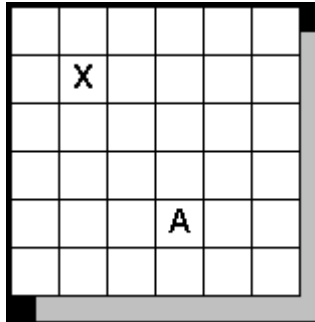
**Exemplo de Problema:** Encontrar uma função para um conjunto de valores dados.

**Descrição:** Alguns valores de entrada e saída são dados. A tarefa é encontrar uma função cuja imagem é a mais próxima dos dados de saída previamente conhecidos.

**Codificação:** Os cromossomos são funções representadas por árvores.

## 10 Como funciona um AG

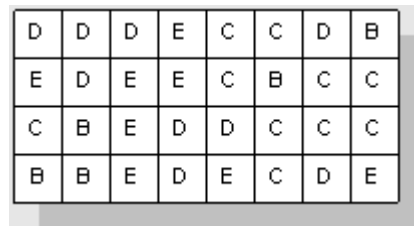
Vamos supor o seguinte problema: estamos numa posição A do mapa em baixo, e queremos atingir a posição marcada com um X.



Dispomos de quatro tipos de movimentos: andar para cima (C), para baixo (B), para a esquerda (E), e para a direita (D). Temos que executar oito movimentos. A avaliação do percurso é feita, no final dos oito movimentos, medindo a posição final em relação à posição objetivo. Assim, à semelhança dos Cromossomos dos seres vivos, as soluções de um problema usam uma codificação específica para este.



A cadeia de símbolos é chamada de cromossomo. Cada uma das suas posições é chamada de gene. Tipicamente é gerado, numa fase inicial, um conjunto de soluções aleatórias - correspondem à população inicial.



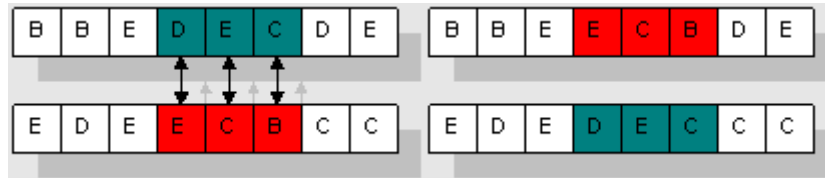
A cada uma dessas soluções (cromossomos) é dada uma pontuação que representa o grau de adaptação dos seus indivíduos ao problema (grau de fitness), ou seja, uma heurística que determina em que medida a solução é boa ou não.

Existe uma operação de Seleção que simula a Seleção Natural. A probabilidade de um indivíduo ser selecionado depende do seu fitness em relação ao fitness dos outros indivíduos da população (quanto maior for o fitness, maior é a probabilidade do cromossomo ser selecionado).



Figura 24 - Foram selecionados o segundo e quarto cromossomos.

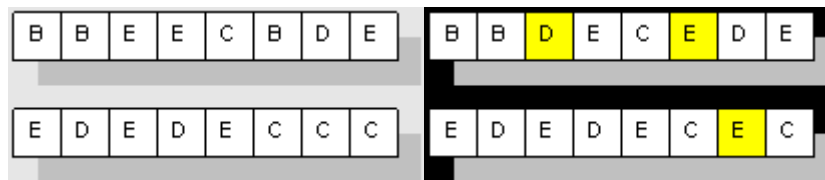
Os Cromossomos selecionados serão cruzados entre si, recombinaando as suas características em novos elementos que serão inseridos na geração seguinte. A operação de cruzamento está associada uma probabilidade de acontecer.



*Figura 25 - Dois Cromossomos antes do cruzamento (à esquerda) e depois (à direita). As secções a sombreado são aquelas que serão trocadas.*

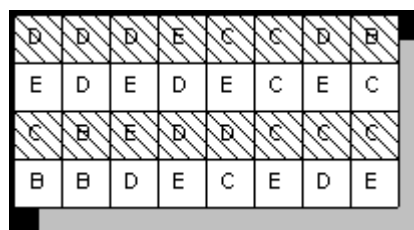
O cruzamento tende a homogeneizar a população ao longo das sucessivas gerações. Isso leva à convergência da população para uma única solução, sem possibilidade de sair daí. É necessário introduzir novos cromossomos, numa tentativa de explorar outros caminhos.

Tal como na Natureza, nos Algoritmos Genéticos a mutação é responsável por introduzir uma variedade de soluções que permita explorar vários ramos, evitando ‘encalhar’ num máximo local.

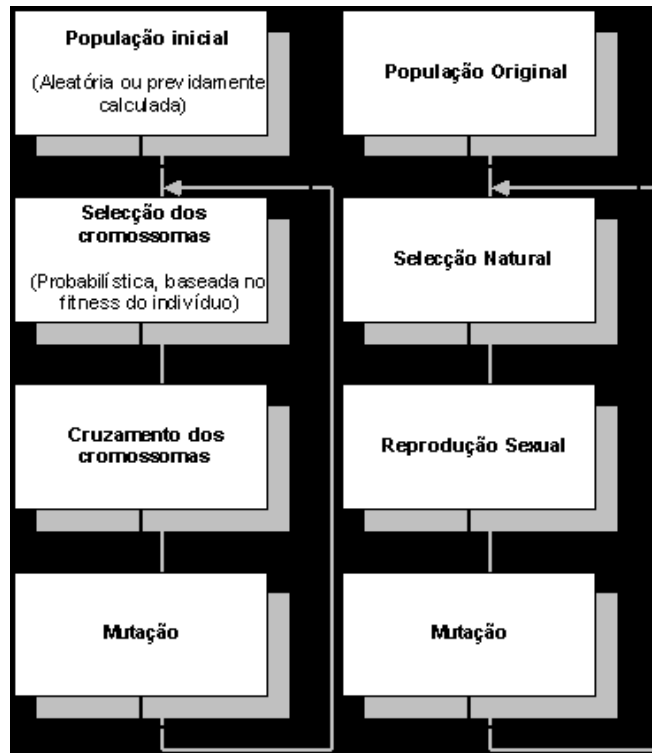


Mais uma vez, esta operação está associada a uma probabilidade (uma mutação a 100% seria mais prejudicial que benéfica, uma vez que se tende a assemelhar com uma procura aleatória).

Os dois cromossomos, resultantes das eventuais operações de cruzamento e mutação, serão parte integrante da nova geração.



Esquematizamos, em seguida, o algoritmo genético básico, e a equivalência com a que acontece na Natureza.



*Figura 26 - Analogia entre os Algoritmos Genéticos e a Seleção Natural.*

## 11 Um Exemplo de Algoritmo Genético

### 11.1 Características do Algoritmo

O tamanho da população será de 4 strings de 5 bits de comprimento. A probabilidade de cruzamento será de 0.6 e a probabilidade de mutação é de 0.001. Como esta última taxa é muito pequena, mutações não ocorrerão neste exemplo.

### 11.2 O Problema de Otimização

O problema é simples: encontrar o valor máximo da seguinte função:

$$y = -x^2 + 8x + 15$$

Por medidas de simplificação, será assumido que o ponto de máximo está compreendido entre 0 e 25 (o valor do ponto de máximo é  $x=4$ ) e que é um número inteiro.

Estabelecidas estas informações, é necessário definir um esquema de codificação para as strings. Pode-se representar números inteiros da faixa [0 .. 31] com strings de 5 bits. Alguns exemplos de strings presentes na população podem ser:

String	Valor Codificado
00001	1
00101	5
10110	22

Finalmente, pode-se determinar a função de adaptação, a qual dará os valores de adaptação relativos de cada string. O método que será empregado consiste em utilizar o valor codificado de  $x$  para calcular a coordenada  $y$  como uma taxa de adaptação. Portanto, o valor de adaptação da string  $i$ , em porcentagem, será o valor de  $y$  em  $i$  dividido pela soma de todos os valores  $y$  para cada string.

$$\text{Fitness Value}_i = \frac{f_i}{\sum f}$$

**Fitness Function for String  $i$   
in the String Population**

Por exemplo, considere a função  $y=f(x)=x^2$  para a qual deseja-se encontrar o valor máximo entre  $[0 \dots 31]$ . As seguintes strings têm seus valores relativos de adaptação indicados a baixo:

String	Valor de $x$	$f(x)$	Valor de Adaptação Relativo
00101	5	25	0.04
01101	13	169	0.25
10110	22	484	0.71

Se a função para a qual deseja-se encontrar máximo ou mínimo assumir valores negativos, a função de adaptação torna-se um pouco mais complexa, porém, em essência, continua equivalente à apresentada a cima.

## Executando um AG: Resultados

### A Primeira Iteração

Primeiramente, é necessário criar uma população de strings aleatória. Digamos que comecemos com as seguintes strings:

População de Strings
00010
00111
10110
01011

Agora realiza-se a seleção. O valor de adaptação de cada string é calculado e as strings são selecionadas o seguinte número de vezes:

String	Valor de $x$	$f(x)$	Valor de Adaptação Relativo	Número de Seleções
00010	2	27	0.35	1
00111	7	22	0.34	2
10110	22	- 293	0.008	0
01011	11	- 18	0.30	1

Com estas seleções, o grupo de strings reprodutoras fica definido da seguinte forma:

População de Strings
00111
00010
00111
01011

Finalmente, a probabilidade de cruzamento precisa ser calculada (dois cruzamentos precisam ser realizados para gerar uma nova população de duas strings). O cruzamento é aplicado tendo como resultado o seguinte:

Grupo das Strings Reprodutoras	Nova População
000110 001111	00011 00110
011011 001010	01010 00011

Então, ao final da primeira iteração, a nova população obtida é:

População das Strings	Valor de $x$
00011	3
00110	6
01010	10
00011	3

Assim, mesmo após somente uma iteração sem nenhum conhecimento além do valor de adaptação, o AG começou a convergir rapidamente para o valor ótimo 4. Isto é surpreendente, considerando que o algoritmo genético sabe nada a respeito do espaço do problema na qual pesquisa. Ele é efetivamente cego. Ainda, apenas analisando o grau de correção, o AG pesquisa eficientemente o espaço do problema para possíveis respostas.

## 12 Uso da Tecnologia de AGs

Embora as páginas anteriores tratem os algoritmos genéticos somente com uma técnica de otimização, há uma vasta diversidade de áreas nas quais a tecnologia dos AGs são empregados.

Os algoritmos genéticos são usados para a resolução de problemas difíceis, para aprendizado de máquinas, para problemas simples, etc. Eles são utilizados também para problemas relacionados à arte envolvendo imagens e músicas. A grande vantagem apresentada pelos AGs é o paralelismo.

Eles são fáceis de implementar. Uma vez que se tenha um AG é necessário simplesmente escrever um novo cromossomo, alterar a função de adaptação e talvez,



dependendo do problema, mudar a codificação das strings para resolver um outro problema. Porém, a determinação da função de adaptação e do tipo de codificação empregado pode ser complexa.

A desvantagem dos AGs está no tempo computacional requerido. Eles podem ser mais lento do que alguns outros métodos.

Para se ter uma idéia dos problemas que podem ser resolvidos utilizando AG segue a baixo uma pequena lista de algumas aplicações:

1. *Reconhecimento de Suspeitos de Crimes*
2. *Composição de Músicas*
3. *Sistemas Dinâmicos Não-Lineares*
4. *Trajetórias de Robôs*
5. *Funções para Criação de Imagens*
6. *Construção e Treinamento de Redes Neurais*
7. *Problemas de Planejamento (Problema do Caixeiro Viajante)*
8. *Jogos*
9. *Dilema dos Prisioneiros*
10. *Detecção de Epicentros de Terremotos*
11. *Otimização Estrutural*
12. *Otimização de Funções*
13. *Otimização de Consultas a Banco de Dados*
14. *Desenho de Aeronaves*
15. *Determinação das Estruturas de Proteínas*

### **13 Parâmetros dos Algoritmos Genéticos**

Esta seção apresenta algumas recomendações caso você esteja interessado em implementar seu próprio algoritmo genético. Estas recomendações são gerais. Provavelmente você irá querer experimentar seus próprios parâmetros para problemas

específicos. Atualmente não há uma teoria geral que descreva quais parâmetros devem ser adaptados para cada problema.

Estas recomendações são resultados de estudos empíricos sobre AGs que foram executados somente para codificação binária.

- **Taxa de crossover (Cruzamento)**

A taxa de deve ser alta, aproximadamente 80% - 95%. (Entretanto alguns resultados mostraram que para alguns problemas taxas de crossover próximas de 60% eram melhores).

- **Taxa de Mutação**

A taxa de mutação deve ser muito baixa. As melhores taxas relatadas são de aproximadamente 0.5% - 1%.

- **Tamanho da População**

Pode ser uma surpresa, mas populações de numerosas não melhoram o desempenho do AG, isto é, não melhoram a velocidade de determinação de uma solução. Um bom tamanho para uma população inicial é de cerca de 20 a 30 indivíduos, entretanto, as vezes populações de 50 a 100 indivíduos apresentam melhores resultados. Alguns pesquisadores mostraram que o tamanho ideal de uma população depende da forma de codificação das strings e também dos tamanhos destas últimas. Isto significa que se os cromossomos de uma população tiverem tamanho igual a 32 bits, a população deveria ter um tamanho igual a 32.

- **Seleção**

A seleção do tipo "Roleta Giradora" pode ser usada, mas algumas vezes a seleção por classificação pode ser melhor. A seção sobre Seleção possui as vantagens e desvantagens do uso destes tipos. Há também alguns métodos mais sofisticados os quais alteram os parâmetros do AG durante sua execução. Basicamente eles se comportam como

o método do "Anelamento Simulado". Mas certamente o elitismo pode ser usado (se nenhum outro método for utilizado para salvar a melhor solução). Pode-se tentar a seleção de estado estável.

- **Codificação**

A codificação depende do problema e também do tamanho da instância do problema. A seção sobre Codificação possui algumas sugestões.

- **Tipos de Crossover e Mutação**

Estas operações dependem do tipo de codificação das strings e do problema. As seções sobre estas operações possuem algumas sugestões.

## **14 Programação Genética**

O campo da Programação Genética (PG), proposto por John Koza e James Rice, de Stanford, aplica as idéias de AGs a estruturas muito mais complexas que seqüências binárias. Cada cromossomo representa uma árvore computacional de tamanho variável, em contraste com os cromossomos de tamanho fixo da grande maioria dos AGs.

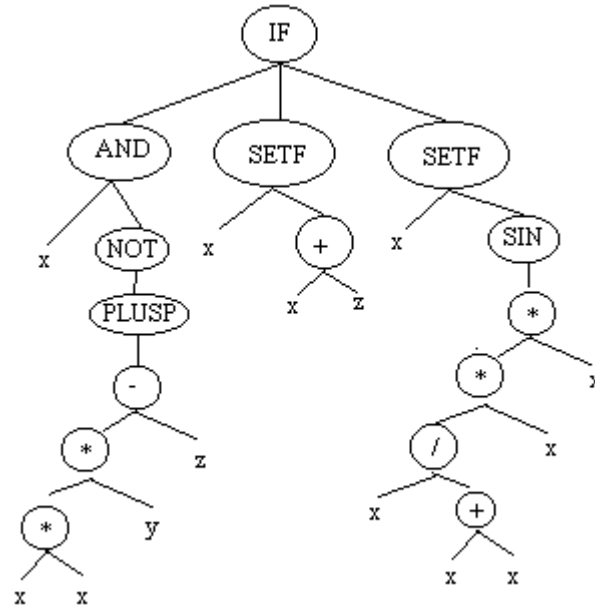
Mais ainda, cada estrutura cromossômica em PG representa um programa, geralmente uma expressão-S da linguagem LISP. LISP é uma escolha natural dos adeptos de PG porque a sintaxe de LISP pode ser representada por uma estrutura em árvore, sem distinção entre código e dados. Por exemplo, consideremos a seguinte expressão LISP:

```
(IF (AND x (NOT (PLUSP (-(*(*xx)y)z))))
```

```
(SETF x (+xz))
```

```
(SETF x (SIN (*(*(/x (+ xx))x)x))))
```

A expressão acima pode ser facilmente representada em árvore, conforme ilustrado pela Figura 27:



*Figura 27 - Exemplo de um programa de computador representado em árvore. Em PG, cada árvore equivale a um indivíduo, e uma população de programas evolue de modo análogo aos AGs.*

Em PG, cada árvore representa um programa de computador e é equivalente a um indivíduo de uma população que evolui de acordo com um AG.

Em linguagem mais acessível, a árvore da Fig. 27 é equivalente a:

IF ( $x \neq 0$ ) AND ( $x^2y - z > 0$ )

THEN  $x \leftarrow x + z$

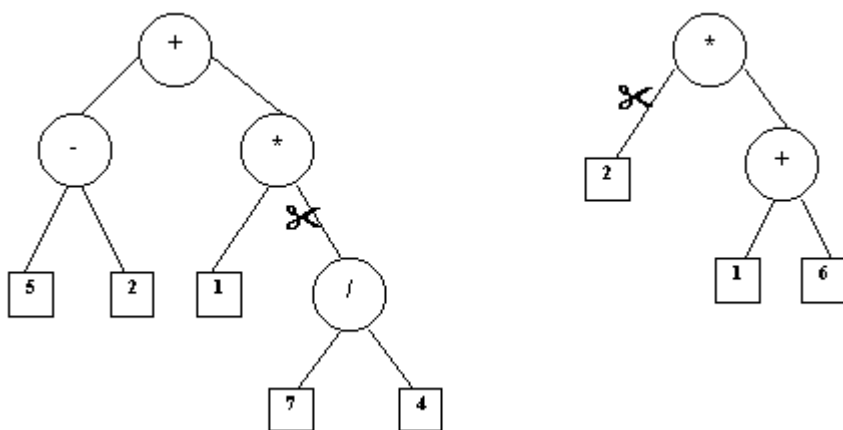
ELSE  $x \leftarrow \sin(x^2/2)$

Assim sendo, PG realiza a geração automática de programas de computador para resolver um dado problema. Embora os modelos atuais de PG ainda sejam vagarosos e ineficientes, PG tem sido aplicada a uma enorme gama de problemas incluindo regressão simbólica, reconhecimento de padrões, identificação e controle de processos, e geração automática de máquinas de estados finitos.

### 14.1 Cruzamento

Para realizar o cruzamento, escolhe-se o ponto de corte das duas árvores escolhidas, como mostra a figura 28:

*Figura 28 - Duas árvores selecionadas para o cruzamento*



Feito o cruzamento, temos os dois filhos resultantes, como é mostrado na figura 29.

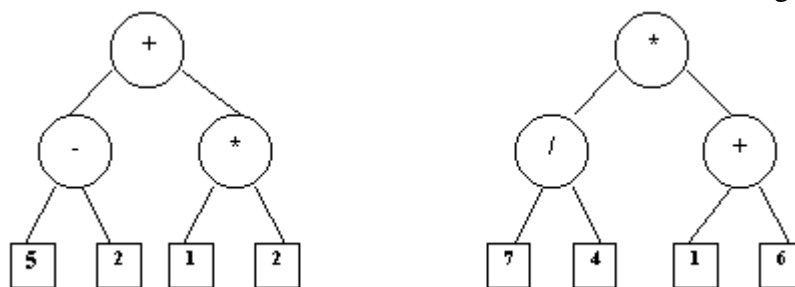


Figura 29 - Árvores resultantes do cruzamento.

## 14.2 Mutação

Neste exemplo, a mutação irá simplesmente trocar um operador aritmético de um nó da seguinte árvore gerada de uma expressão aritmética:

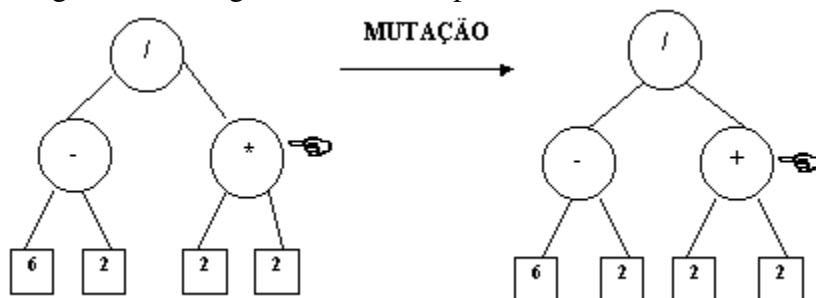


Figura 30 – Exemplo de mutação onde é feita a troca de um operador aritmético.

Um outro tipo de mutação consiste em escolher aleatoriamente um ramo da árvore, retirá-lo da mesma e substituí-lo por um outro ramo gerado pelo programa para esse efeito. Este tipo de mutação é capaz de modificar a estrutura da árvore, dado que não obriga a que o novo nó tenha o mesmo tamanho do que foi retirado. A figura seguinte pretende ilustrar o que foi descrito anteriormente:

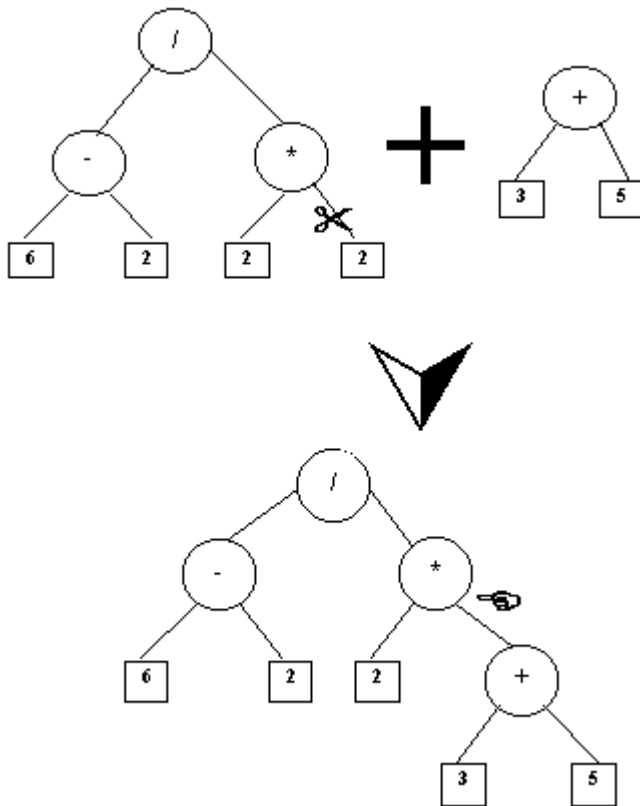


Figura 31 - Mutação onde é trocado o ramo da árvore por um outro aleatório.

Neste caso é fácil verificar que a estrutura da árvore foi modificada, passando de uma árvore de profundidade 2 para uma com profundidade 3, o que demonstra que este tipo de mutação tem a capacidade de efetuar maiores alterações sobre a informação genética de uma função do que a mutação que envolve apenas um nó da árvore.

## 15 O Programa Dougal

O Programa Dougal (Demonstration Of Using Genetic Algorithm Learning) tem por objetivo produzir um programa tutorial para ser usado no ensino de conceitos de Algoritmos Genéticos.

### 15.1 Descrição do programa

O programa foi desenvolvido para solucionar o problema do caixeiro viajante, onde um grupo de estudantes está em férias pela Europa e pretendem começar e terminar a

viagem em Londres e percorrer as seguintes cidades: Amsterdam(A), Berlim(B), Copenhagen(C), Moscou(M), Oslo(O), Paris(P), Roma(R) , Veneza(V), Varsóvia(W) e Zurique(Z). Deve-se visitar as cidades apenas uma única vez. O objetivo é traçar uma rota que contenha a menor distância a ser percorrida entre as cidades visitadas.

A solução ótima para o problema é: L A B C O M W R V Z P L , que resulta em uma distância percorrida de 4.580 milhas.

O programa utiliza os operadores de cruzamento, mutação e se desejar, pode-se informar se haverá elitismo e qual o número de indivíduos elitistas. O método de seleção utilizado é o Método da Roleta.

Em um primeiro teste feito, foi informado os seguintes parâmetros para o programa:

- População Inicial: 100
- Número de Gerações: 50
- Taxa de crossover: 60%
- Taxa de Mutação: 1%
- Número de indivíduos elitistas: 2

Ao final do processo, ou seja, na geração de número 50, obtivemos a seguinte rota que foi considerada a melhor: L P Z V R W M O B C A L, que não é a solução ótima para o problema, pois teve como resposta 6.080 milhas.

Agora, em outro teste, mudamos alguns parâmetros de entrada, a fim de que o programa possa convergir para a solução ótima. Então utilizamos as seguintes configurações:

- População Inicial: 100
- Número de Gerações: 200
- Taxa de crossover: 60%
- Taxa de Mutação: 1%
- Número de indivíduos elitistas: 2

Ao final do processo, ou seja, na geração de número 200, obtivemos a seguinte rota que é a solução ótima para o problema: L P Z V R W M O B C A L. Pode-se perceber que, aumentando o número de gerações, o programa conseguiu encontrar a solução, sendo que antes ele encontrava um ótimo local e não progredia mais.

## 16 Exercícios de AGs

### 1. Qual a diferença entre Algoritmos Genéticos e os algoritmos tradicionais de busca?

#### Resposta:

Os algoritmos genéticos utilizam uma forma codificada dos valores de uma função (conjunto paramétrico), em vez dos atuais valores. Se, por exemplo, deseja-se encontrar o mínimo da função  $f(x)=x^3+x^2+5$ , os AGs não tratariam diretamente com os valores de  $x$  e  $y$ , mas com strings que codificam estes valores. Para este caso, strings representando valores binários de  $x$  deveriam ser usadas.

Os algoritmos genéticos usam um conjunto, ou população, de pontos para conduzir uma pesquisa, não somente um ponto isolado do espaço de pesquisa. Isto dá aos AGs a capacidade de pesquisar em espaços ruidosos com vários pontos ótimos locais. Os AGs observam diferentes áreas do espaço do problema de uma só vez e usam todas estas informações para se guiarem.

Os algoritmos genéticos usam somente informações geradas por si mesmos para se guiarem pelo espaço de pesquisa. Muitas outras técnicas necessitam de uma grande variedade de informações para guiarem-se. O método hill-climbing requer derivadas, por exemplo. A única informação utilizada pelos AGs é uma medida de adaptabilidade de cada ponto no espaço (as vezes conhecida como valor da função objetiva). Uma vez conhecido este valor sobre um ponto, os AGs podem usá-lo para continuar a pesquisa pelo valor ótimo.

Os algoritmos genéticos possuem natureza probabilística, não determinística. Isto é resultado direto das técnicas randômicas usadas pelos AGs.



Os algoritmos genéticos são inerentemente paralelos. Este é um dos mais importantes e poderosos aspectos dos AGs. Algoritmos genéticos tratam um grande número de pontos (strings) simultaneamente.

**2. Sabendo-se que há uma multiplicidade impressionante de algoritmos para solução de problemas, porque a necessidade da abordagem dos algoritmos genéticos neste contexto?**

**Resposta:**

Porque problemas importantes continuam sendo difíceis de resolver. As seguintes características impedem o uso ou degradam o efeito da aplicação de algoritmos clássicos: intratabilidade matemática, não-linearidades, ausência de informação suficiente acerca do problema, observações ruidosas, variação no tempo, descontinuidade, explosão combinatória de candidatos à solução.

**3. Qual a diferença entre algoritmos genéticos e programação genética?**

**Resposta:**

Os algoritmos genéticos foram introduzidos por Holland em 1975 com o objetivo de formalizar matematicamente e explicar rigorosamente processos de adaptação em sistemas naturais e desenvolver sistemas artificiais (simulados em computador) que retenham os mecanismos originais encontrados em sistemas naturais.

A programação genética é uma extensão dos algoritmos genéticos, introduzidos por Koza, e tem por objetivo básico evoluir programas de computador usando os princípios da evolução natural.

**4. Comente sobre o método da roleta usada para a seleção de indivíduos e os problemas que podem ocorrer com este método.**

**Resposta:**

O método da roleta é um esquema para de seleção de indivíduos que farão parte da próxima geração. O método atribui para cada indivíduo de uma população uma probabilidade de passar para a próxima geração proporcional ao seu fitness medido, em relação à somatória do fitness de todos os indivíduos da população. Assim, quanto maior o fitness de um indivíduo, maior a probabilidade dele passar para a próxima geração. Um problema é que o melhor indivíduo da população seja perdido, ou seja, não passe para a próxima geração.

**5. Encontre uma nova geração buscando maximizar a função  $f(x) = 3x+5$ , para  $x$  no intervalo de  $[0,15]$ . Inicialmente é gerada uma população formada por um conjunto aleatório de indivíduos, que podem ser vistos como possíveis soluções do problema.**

0,85	0,54	0,45	0,31
0,21	0,48	0,68	0,70
0,34	0,84	0,78	0,50
0,01	0,02	0,90	0,78

Através da tabela de números aleatórios cria-se a codificação: (0 e 1)

Se o valor for menor ou igual ( $\leq$ ) a 0,5 ,então 1 (um), senão é 0 (zero).

Defina um número de quatro indivíduos, utilizando cada linha do conjunto de indivíduos dado acima e codificando para uma string de bits seguindo a regra dada, fazer a 1º geração e calcular o fenótipo, fitness e fitness total. Calcular a proporção e proporção acumulada.

**Resolução:**

Utilizando a codificação binária, pois é a mais genérica, e como  $x$  é um inteiro entre 0 a 15, então o número de bits do genótipo (string de bits) será quatro.

Em seguida, codificamos para 0s ou 1s, a primeira linha do conjunto de indivíduos gerada aleatoriamente, dado no enunciado do problema. Lembre-se, poderíamos pegar

qualquer  $x$  (fenótipo), entre 0 e 15 e decodificar os números selecionados para uma string de bits, mas no caso desse exercício, estamos usando uma população já existente, criando uma string de bits e pegando o seu fenótipo para resolver esse problema.

Primeira linha: **0,85    0,54    0,45    0,31**

**0,85** > 0,5 então **0**; **0,54** > 0,5, então **0**; **0,45** ≤ 0,5, então **1**; **0,31** ≤ 0,5, então **1**. Logo, o genótipo (string de bits) do primeiro indivíduo é **0011** e o fenótipo ( $x$ ) é **3**. Para as demais linhas, basta aplicar esses passos para encontrar o genótipo e fenótipo. O resultado é dado na tabela a seguir:

Número do Indivíduo	Genótipo (String de bits)	Fenótipo ( $x$ )
1	0011	3
2	1100	12
3	1001	9
4	1100	12

Agora, para obter o fitness, basta aplicar o fenótipo ( $x$ ) a função  $f(x)$ . Logo, para o primeiro indivíduo, teremos o seguinte:

Para  $x = 3$ , e  $f(x) = 3x+5$  temos que  $f(3) = 14$ , logo o fitness do indivíduo 1, será **14**. Igualmente para os demais indivíduos. Além do fitness, uma nova linha é acrescida na tabela, para indicar o total.

Número do Indivíduo	Genótipo	Fenótipo	Fitness (função $f$ )
1	0011	3	14
2	1100	12	41
3	1001	9	32
4	1100	12	41
<b>Total</b>			<b>128</b>

Em seguida, temos que calcular a proporção para cada indivíduo, esse proporção é calculada da seguinte forma: **fitness  $i$  /  $\Sigma$**

**fitness** (o fitness do  $i$ ésimo indivíduo dividido pelo soma dos fitness de todos os indivíduos da tabela). Para o primeiro indivíduo, temos o seguinte:

Para fitness **14** do primeiro indivíduo, e **128** a soma de todos os fitness, aplicando a fórmula, **14/128 = 0,109**, temos que a proporção desse indivíduo é **0,109 (10,9%)**. Para a proporção dos demais, basta seguir a regra do primeiro. A tabela a seguir é apresentada com as proporções e a soma que deve ser **1 (100%)**.

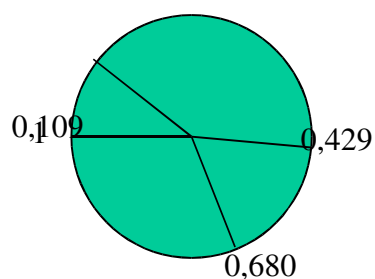
Nº do Indivíduo	Genótipo	Fenótipo	Fitness	Proporção
1	0011	3	14	0,109

2	1100	12	41	0,320
3	1001	9	32	0,251
4	1100	12	41	0,320
<b>Total</b>			<b>128</b>	<b>1</b>

Finalmente, a última coluna da tabela será a proporção acumulada. Para o primeiro indivíduo, coloco somente a própria proporção. Para o segundo, somo sua proporção mais a soma dos anteriores, no caso, somente o primeiro. E assim para os demais. A tabela resultante será a seguinte:

Nº do Indivíduo	Genótipo (str. de bits)	Fenótipo (x)	Fitness (função f)	Proporção (fit i / $\sum$ fit)	Proporção Acumulada
1	0011	3	14	0,109	0,109
2	1100	12	41	0,320	0,429
3	1001	9	32	0,251	0,680
4	1100	12	41	0,320	<b>1</b>
<b>Total</b>			<b>128</b>	<b>1</b>	

A proporção é representada graficamente da seguinte forma:



**6. Considerando a população inicial abaixo, encontre uma nova geração, buscando maximizar a função (  $x^3 / 2 - 2x^2 + 5$  ) para x no intervalo de [0,15]. Considere a probabilidade de mutação 5% e de crossover 80%. Selecione os casais para crossing-over e, a seguir, realize-o. Em seguida,**

verifique a existência de mutação nos indivíduos gerados e, a seguir, se for o caso, efetue as mutações.

Indivíduo	genótipo	fenótipo	fitness	percentual	acumulado
1	0011	3	0,5		
2		9	207,5		
3	1001 0101	5	17,5		
4	0111				

Utilize a tabela de números aleatórios em linha começando pela primeira.

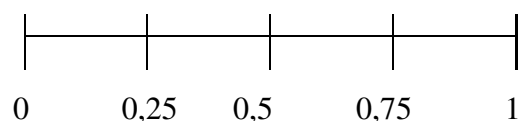
Tabela:

0,3470	0,9933	0,6565	0,2744	0,4569	0,0224	0,9026
0,0417	0,8576	0,7662	0,6435	0,2777 <sup>c</sup>	0,2834 <sup>pc</sup>	0,2119 <sup>m</sup>
0,1372 <sup>m</sup>	0,7281 <sup>pm</sup>	0,5025 <sup>m</sup>	0,7263 <sup>m</sup>	0,1118	0,1496	0,8717

Obs.:

1) Os quatro primeiros valores serão utilizados na primeira seleção, para identificar quais valores participarão do crossover, o quinto é usado para verificar se haverá ou não crossover, o sexto é usada em uma escala para se determinar o ponto de corte no crossover, e assim por diante. No exercício, você precisará de um valor aleatório, o qual buscará dessa tabela. Então, você já sabe com antecedência, em que passo do algoritmo genético foi utilizado esse valor, isso porque cada letra indica o seguinte: *c* é para crossover, *pc* é para o ponto de corte no crossover, *m* é para mutação e *pm* é para o ponto de mutação (também se usa a escala para se determinar o ponto).

2) Para a escolha do ponto de corte (ou mutação), utilize uma escala para determinar a posição. A escala pode ser assim determinada: 1 dividido pelo total de genes. Neste exercício serão quatro genes, então:  $1/4 = 0,25$ .



## Resolução:

Primeiramente, temos que completar a tabela. O fenótipo e fitness do quarto indivíduo obtém-se da seguinte forma:

Para 0111, temos  $0x2^3 + 1x2^2 + 1x2^1 + 1x2^0 = 7$  (fitness), então  $x=7$  em  $f(x) = x^3/2 - 2x^2 + 5$  temos que  $f(7) = 78,5$ , logo, esse é o fitness do indivíduo 4.

Em seguida, temos que calcular a proporção para cada indivíduo, esse proporção é calculada da seguinte forma: **fitness i /  $\Sigma$**

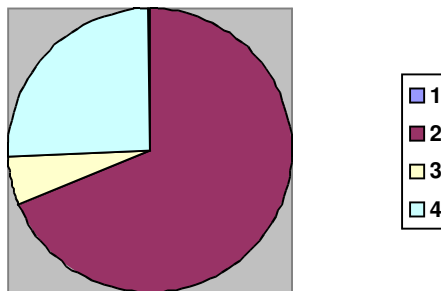
**fitness** . Para o primeiro indivíduo, temos o seguinte:

Para fitness **0,5** e **304** a soma de todos os fitness, aplicando a fórmula,  $0,5/304 = 0,0017$ , temos que a proporção desse indivíduo é **0,17%**. Para a proporção dos demais, basta seguir a regra do primeiro. A tabela a seguir é apresentada com as proporções e a soma que deve ser **1 (100%)**.

Finalmente, a última coluna da tabela será a proporção acumulada. Para o primeiro indivíduo, coloco somente a própria proporção. Para o segundo, somo sua proporção mais a soma dos anteriores, no caso, somente o primeiro. E assim para os demais. A tabela resultante será a seguinte:

Indivíduo	genótipo	fenótipo	fitness	percentual	acumulado
1	0011	3	0,5	0,0017	0,0017
2	1001	9	207,5	0,6825	0,6842
3	0101	5	17,5	0,0576	0,7418
4	0111	7	78,5	0,2582	1
			304	1	

Na roleta, os percentuais de cada indivíduo são assim representados (indivíduo 1 nem aparece):



Mas, para facilitar o entendimento, mostraremos a seleção da seguinte forma: (estamos utilizando o método da roleta, mas serão representados em uma linha e não em gráfico tipo pizza).



Queremos obter quatro progenitores. Começamos por quatro números aleatórios. Nesse exercício, foi dada uma tabela de números aleatórios, então começaremos pelos primeiros que estão selecionados:

**0,3470 0,9933 0,6565 0,2744**

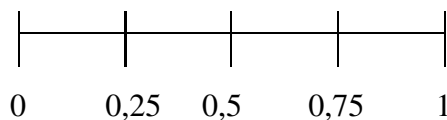
Assim, como **0,3470** pertence ao intervalo número 2, escolhe-se o **indivíduo 2**. Para os demais, segue o primeiro. Logo, os indivíduos selecionados para gerar a descendência serão, respectivamente.

**2, 4, 2, 2**

Para a fase de cruzamento, escolheremos pares aleatoriamente. Começaremos na ordem de seleção para facilitar, então, os escolhidos para o primeiro crossover serão os indivíduos **2** e **4**.

2				4			
1	0	0	1	0	1	1	1

Primeiramente, vamos verificar se haverá crossover entre esses indivíduos. A probabilidade de crossover dada é **80%**, e o próximo valor na tabela de números aleatórios é **0,4569**. Este valor é menor que o percentual de **0,800**, então vai haver cruzamento. Para determinar o ponto de corte, consultamos a tabela de número aleatórios novamente. O valor é **0,0224**.



Utilizando a escala acima, observamos que o valor indica o primeiro gene (bit) com o ponto de corte. Logo:

2				4			
1	0	0	1	0	1	1	1

E os novos indivíduos gerados, serão os seguintes:

1	1	1	1
0	0	0	1

Agora queremos saber se vai haver mutação nesses dois indivíduos gerados. Sabendo-se que a taxa de mutação é de 5%, e pegando novamente um valor da lista de números aleatórios, temos um percentual de **0,902** para o primeiro indivíduo gerado, mas esse valor é muito mais que os **0,05** de taxa, logo, não haverá mutação no **indivíduo 1**. O valor na tabela para o **indivíduo 2** é **0,0417**. Este valor é menor que a taxa de 5%, logo, haverá mutação.

Para saber em qual bit aplicaremos a mutação, pega-se o próximo valor na tabela de números aleatórios, no caso **0,8576** e verifica-se na escala de ponto de corte dada anteriormente (escala também usada no ponto de mutação).

**2**

0	0	0	1
---	---	---	---

O ponto de mutação foi escolhido no último bit e o indivíduo resultante será o seguinte:

**2**

0	0	0	0
---	---	---	---

Terminado crossover e mutação dos dois primeiros indivíduos, vamos aos próximos indivíduos. Como o terceiro e o quarto são iguais, **2** e **2**, então não haverá cruzamento, pois não haverá nenhuma troca genética. O que se pode fazer com esses dois indivíduos, é verificar se eles podem ou não, sofrer mutação.

Para o terceiro indivíduo, tomamos o próximo valor na tabela de números aleatórios. O percentual é **0,7662**, mas esse percentual está muito acima da taxa de 5%, logo não haverá mutação para esse indivíduo e nem para o próximo, pois o percentual é de **0,6435**.

Finalmente, temos os indivíduos para a próxima geração:

Indivíduo	genótipo
1	1111
2	0000
3	1001
4	1001



Daí por diante, já vimos que para obter o fenótipo e fitness, por exemplo o primeiro, pegamos seu genótipo **1111** e fazemos o seguinte,  $1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = \mathbf{15}$  (fitness), então  $x=15$  em  $f(x) = x^3/2 - 2x^2+5$  temos que  $f(15) = \mathbf{1243}$ , logo, esse é o fitness do indivíduo 1. E assim para os demais.

Em seguida, soma-se os fitness obtidos, e obtém-se o percentual para cada indivíduo, usando a fórmula : **fitness i /  $\Sigma$**

**fitness** . Para o primeiro indivíduo, temos o seguinte: para fitness **15** e **1663** a soma de todos os fitness, aplicando a fórmula,  $\mathbf{15/1663 = 0,74736842}$ , temos que a proporção desse indivíduo é **0,747%**. Assim para os demais.

A proporção acumulada, obtém-se da maneira como vimos anteriormente. E, para concluir o exercício, a tabela final para a primeira geração será a seguinte:

•

1ª Geração:

Indivíduo	genótipo	fenótipo	fitness	percentual	acumulado
1	1111	15	1243	0,74736842	0,747368
2	0000	0	5	0,00300752	0,750376
3	1001	9	207,5	0,12481203	0,875188
4	1001	9	207,5	0,12481203	1
			1663	1	

## **17 Conclusão**

Os Algoritmos Genéticos e a Programação Genética estão longe de ser a panacéia para os problemas de otimização, mas são, sem dúvida, técnicas úteis e relativamente robustas de otimização. Os primeiros trabalhos foram desenvolvidos em meados dos anos 60 e início dos anos 70 e amadureceram ao longo dos últimos anos e hoje formam não só um dos mais fascinantes ramos da ciência atual, mas também ferramentas de engenharia de comprovado valor prático e de potencial ainda inexplorado.

## **18 Referências Bibliográficas**

TANOMARU, JULIO. “Motivação, Fundamentos e Aplicações de Algoritmos Genéticos”.II Congresso Brasileiro de Redes Neurais. Curitiba, Brasil, 1995, pp. 331-441

PARKER BRETT – Dougal “Demonstration Of Using Genetic Algorithm Learning”,em <ftp.cerias.purdue.edu/pub/doc/ec/ga/scr>

[http://laseeb.ist.utl.pt/portas\\_abertas/ags/Apend\\_dncc.html](http://laseeb.ist.utl.pt/portas_abertas/ags/Apend_dncc.html)

[http://www.geocities.com/Athens/Sparta/1350/ia/a\\_genetic.html](http://www.geocities.com/Athens/Sparta/1350/ia/a_genetic.html)

<http://black.rc.unesp.br/ccomp/algoritmo/>

<http://www.ica.ele.puc-rio.br/cursos/download/CE-ga1.pdf>