UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

COMPUTAÇÃO EVOLUCIONÁRIA
ALGORITMOS GENÉTICOS (1)

Cristiano Leite de Castro
crislcastro@ufmg.br

Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil

UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Introdução

- **Algoritmo Genético** (AG) é um método de busca probabilístico inspirado na Teoria da Evolução de Darwin;
- concebido por John Holland[1] e seus alunos na Universidade de Michigan na década de 1970.
  - objetivo inicial: simular/estudar o comportamento adaptativo dos seres vivos;
  - no entanto, tem sido tipicamente aplicado para resolver problemas de otimização, principalmente com variáveis de decisão discretas (otimização combinatória).

1. John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA. (1975)

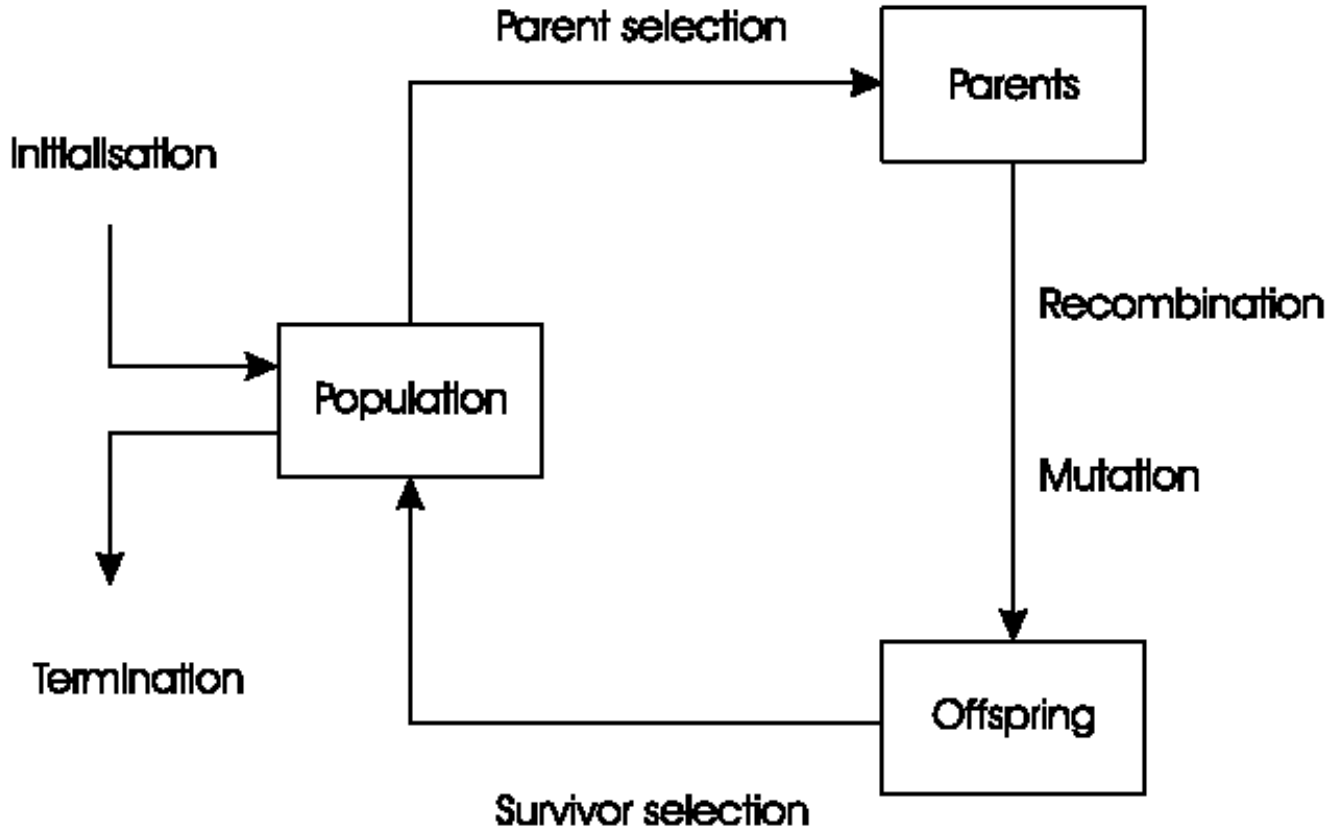
Introdução

- AG original proposto por Holland ficou conhecido como **AG simples (SGA)** ou canônico;

- Desde então, outros AGs têm sido propostos e se diferenciam em:
  - formas de representação das soluções candidatas;
  - operadores de recombinação e mutação;
  - mecanismos de seleção;
  - etc.

AG Simples (SGA)

- Esquema Geral:



UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

AG Simples (SGA)

- Características:

Forma de Representação	Codificação binária
Operador de Recombinação	crossover de 1-ponto
Operador de Mutação	*Bit-Flip* com baixa probabilidade
Seleção dos Pais	Proporcional ao *Fitness*
Seleção dos Sobrevivente	Geracional

Componentes do SGA proposto por Holland et al., 1975

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Componentes de um AG

1. Representação;
2. Operadores de Variação:
    1. Recombinação e Mutação;
3. Mecanismos de Seleção:
    1. Seleção dos Pais
    2. Seleção dos Sobreviventes (substituição);
4. Modelos de População;

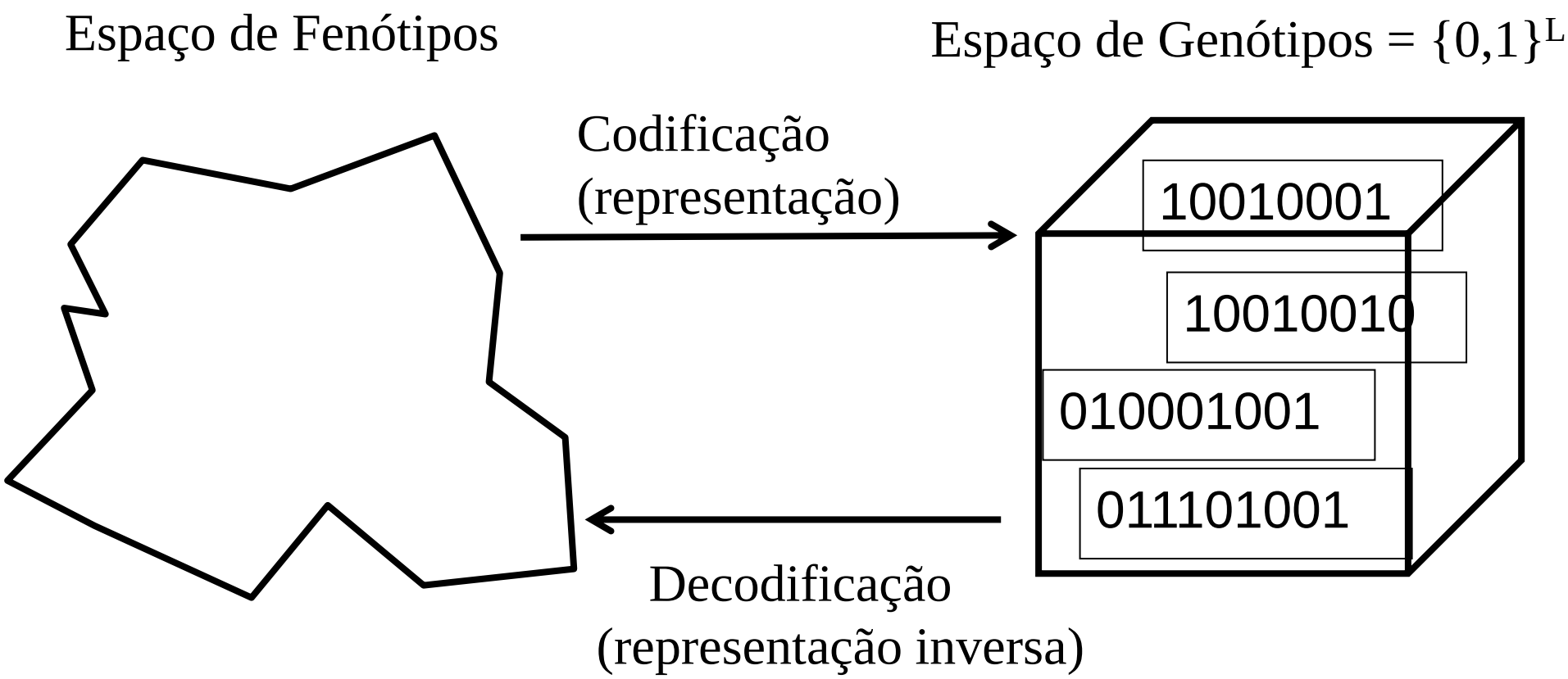
UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Componentes de um AG

1. **Representação**;
2. Operadores de Variação:
    1. Recombinação e Mutação;
3. Mecanismos de Seleção:
    1. Seleção dos Pais
    2. Seleção dos Sobreviventes (substituição);
4. Modelos de População;

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Representação

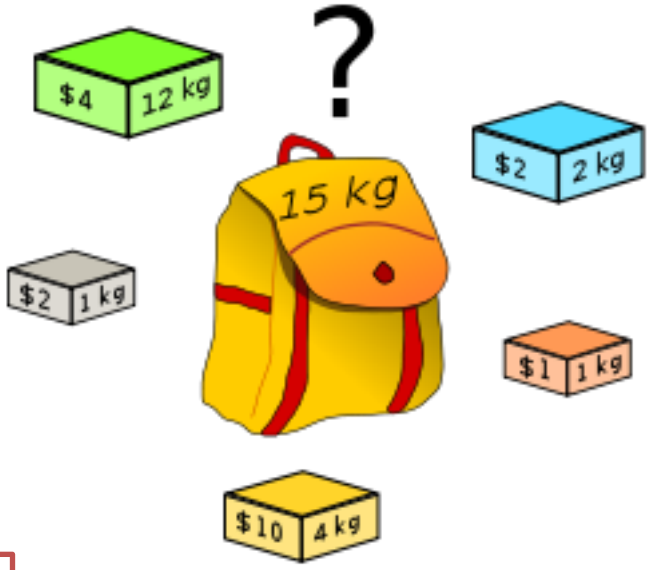


Obter a representação mais apropriada para as soluções candidatas do problema é a etapa mais importante no projeto de um AG.

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Representação Binária

- mais utilizada;
- mais adequada para problemas que envolvem variáveis de decisão *booleana*;
  - Exemplo: **Problema da Mochila 0/1**:



obj1 obj4 obj6

[1 0 0 1 0 1 0 0]

Indivíduo = solução candidata

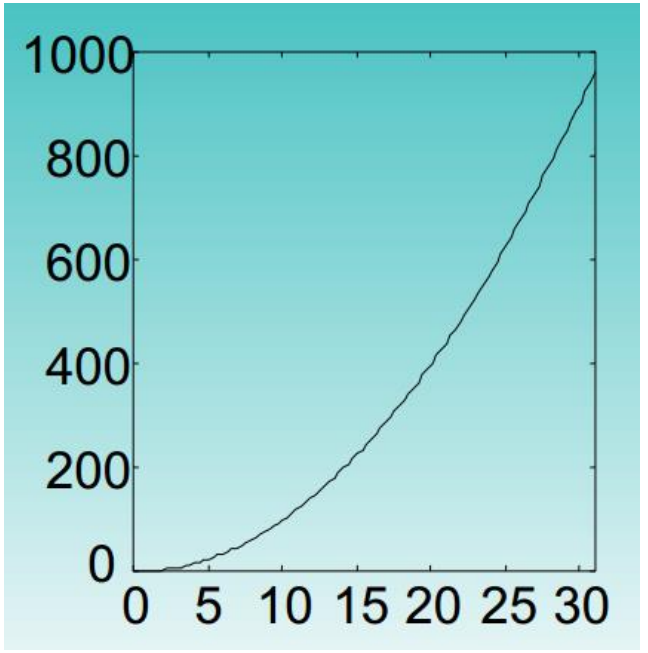
- Solução candidata:

- *string* binária de tamanho n (número de objetos):
  - 1 - indica que o objeto foi incluído na mochila;
  - 0 - indica que o objeto não foi incluído;

Representação Binária

- no entanto, pode ser usada para codificar informação não-binária;
  - Exemplo:

$\max f(x) = x^2 \quad 0 \leq x \leq 31$



String no.	Initial population	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 1	13	169
2	1 1 0 0 0	24	576
3	0 1 0 0 0	8	64
4	1 0 0 1 1	19	361
Sum			1170
Average			293
Max			576

Representação Binária

- **Mapeamento Real → Binário**:
  - seja uma variável de decisão real x_i no intervalo $[L_i, U_i]$;
  - sua representação correspondente em binário é b_i;
  - l_i = número de bits usado para codificar a variável x_i;
  - intervalo entre dois valores adjacentes: $\Delta_{x_i} = \frac{U_i - L_i}{2^{l_i} - 1}$
  - o valor de x_i que corresponde ao binário b_i é dado por:

$$x_i = L_i + (U_i - L_i)\frac{k_i}{2^{l_i} - 1}, \text{com } k_i = \sum_{j=1}^{l_i} b_{ij} 2^{l_i - j}$$

$k_i \in [0\ , 2^{l^i} - 1]$ é o valor decimal que corresponde ao binário b_i.

Representação Binária

- **Mapeamento Real → Binário**:
  - Exemplo: $x_i \in [2,5]$ representado por $l_i = 10$ bits.

0000000000 maps to 2.0, 1111111111 maps to 5.0

Precision = (5.0 - 2.0) / 1023 ≈ 0.00293

0001011011 maps to 2.0 + (5.0 - 2.0) 91 / 1023 ≈ 2.26686

Decimal equivalente a 0001011011

  - um vetor de *n* variáveis de decisão vai requerer $\sum_{i=1}^{n} n * l_i$ bits no total:

x_1 x_2 x_3

1 1 0 0 1 0 1 0 1 1 0 0 0 1 1

$l_1 = l_2 = l_3 = 5$, logo o indivíduo possui 15 bits

Representação Binária

- **Mapeamento Real → Binário**:
  - Qto maior o valor de l_i (número de bits para representar um valor real), melhor a acurácia da representação;
  - Entretanto, isso aumenta o custo computacional do AG.

Representação Binária

- **Principais problemas**:
  - dificuldade em se alcançar uma precisão arbitrária;
  - limites das variáveis de decisão podem ser desconhecidos;
  - *Hamming Cliffs*:
    - uma pequena diferença entre valores no espaço de fenótipos pode significar uma grande diferença entre estes valores no espaço dos genótipos:
    - valores no espaço de fenótipos: 7 e 8 → $|8 - 7| = 1$
    - valores codificados: 7 = 0111 e 8 = 1000 → $dH^1(8,7) = 4$;
    - **Consequência**: para um operador de mutação baseado em *bit-flip*, *por exemplo*, a probabilidade de mudança de 7 (0111) para 8 (1000) não é a mesma de 7 (0111) para 6 (0110);

1. distância de Hamming (dH) = número de bits diferentes entre 2 strings binárias. Exs: 0001 e 0111 tem dH = 2.

Representação Binária

- **Código de *Gray*** (*Frank Gray*, 1953)
  - solução alternativa para codificação binária que resolve o problema dos *Hamming Cliffs*;
  - se k_1 e k_2 são dois inteiros adjacentes representados no código de *Gray* por g_1 e g_2 então

$$|k_1 - k_2| = 1 \rightarrow dH(g_1, g_2) = 1$$

  - para ilustrar, veja o exemplo:

0	1	2	3	4	5	6	7	Valor
000	001	010	011	100	101	110	111	binário
000	**001**	**011**	**010**	**110**	**111**	**101**	**100**	**Gray**

Representação Binária

- Conversões: **Binário ↔ Gray**

  - seja um cromossomo $\vec{b} = (b_1, b_2, \ldots, b_{l_i})$ no código binário e seu correspondente $\vec{g} = (g_1, g_2, \ldots, g_{l_i})$ no código Gray.

  - A conversão entre $\vec{b} \leftrightarrow \vec{g}$ pode ser feita com base no operador "ou-exclusivo":

$$0 \otimes 0 = 0; 0 \otimes 1 = 1; 1 \otimes 0 = 1; 1 \otimes 1 = 0.$$

Representação Binária

- Conversões: **Binário ↔ Gray**

function binaryToGray ($\vec{b}$)
{

$g(1) = b(1);$

for (i $= 2;$ i $\leq l_i; i++$)

$g(i) = b(i-1) \otimes b(i);$

return $\vec{g}$;

}

function grayToBinary ($\vec{g}$)
{

b$(1) = g(1);$

for (i $= 2;$ i $\leq l_i; i++$)

b$(i) = b(i-1) \otimes g(i);$

return $\vec{b}$;

}

Representação Binária

- Conversões: **Binário ↔ Gray**

  – Binário p/ Gray:

  $7_{10} = 00111_2 \quad l_i = 5 \qquad \vec{b} = 00111 \rightarrow \vec{g} = 00100$

  – Gray p/ Binário:

  $\vec{g} = 00111 \rightarrow \vec{b} = 00101$

Representação Binária

- Conversão **Gray p/ número real**:
  - Seja $\overrightarrow{g_i}$ = cromossomo em código Gray representando a variável real x_i;
  - Passo 1: $\overrightarrow{b_i} = \text{grayToBinary}(\overrightarrow{g_i})$;
  - Passo 2: $k_i = \sum_{j=1}^{l_i} b_{ij} 2^{l_i - j}$;
  - Passo3: $x_i = L_i + (U_i - L_i)\frac{k_i}{2^{l_i} - 1}$

Representação Binária

- Conversão **Gray p/ número real**:
  - seja o indivíduo: $[\underset{x_1}{11001}\ \underset{x_2}{01101}\ \underset{x_3}{00010}]$

1) conversão de Gray para real, sendo $x_2 \in [0, 31]$.

$\overrightarrow{g_2} = 01101 \rightarrow \overrightarrow{b_2} = 01001$ $x_2 = 0+(31/31)[0*2^4+ 1*2^3+ 0*2^2+ 0*2^1+ 1*2^0]=9$

1) converta de Gray para real, sendo $x_3 \in [-2, 2.5]$.

$\overrightarrow{g_3} = 00010 \rightarrow \overrightarrow{b_3} = 00011$ $x_3 = -2+(4.5/31)[0*2^4+ 0*2^3+ 0*2^2+ 1*2^1+ 1*2^0]$

$= -2+0.145[3] = -1.565$

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Representação Real

- comumente usada em problemas de otimização com variáveis contínuas $f: \mathbb{R}^n \rightarrow \mathbb{R}$;
- a solução candidata é representada através de um vetor de números reais:

$$\vec{x} = [x_1, x_2, \ldots, x_n] \text{com } x_i \in \mathbb{R}$$

- cada gene corresponde diretamente a uma variável de decisão x_i do problema de otimização;
- o número de variáveis *n* determina a dimensão do espaço de busca.

Representação Real

- Exemplo:

$$f(x_1, x_2) = 100({x_1}^2 - x_2)^2 + (1 - x_1)^2$$

$$-2048 \leq x_1, x_2 \leq 2048$$

  - representação de uma possível solução para o problema:

$$\vec{x} = [-1.235, 0.023\ ]$$

Representação por Permutação

- comumente usada em problemas de otimização combinatória em que é necessário definir a ordem na qual uma sequência de eventos ocorre.

- Exemplos:
  - *Job Shop Scheduling*;
  - *Traveling Salesman Problem* (TSP);
  - etc.

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Componentes de um AG

1. Representação;
2. **Operadores de Variação:**
    1. **Recombinação e Mutação;**
3. Mecanismos de Seleção:
    1. Seleção dos Pais
    2. Seleção dos Sobreviventes (substituição);
4. Modelos de População;

Recombinação (*Crossover*)

- mais importante operador para geração de novos indivíduos no espaço de busca;

- p_C = probabilidade de *crossover*:
  - determina a chance de um par de pais ser submetido à recombinação de seus genes;
  - um número t entre [0,1[é sorteado e comparado a p_C:
    - se $t < p_C$ então 2 filhos são gerados via *crossover* dos pais;
    - caso contrário, 2 filhos são gerados como cópias dos pais.

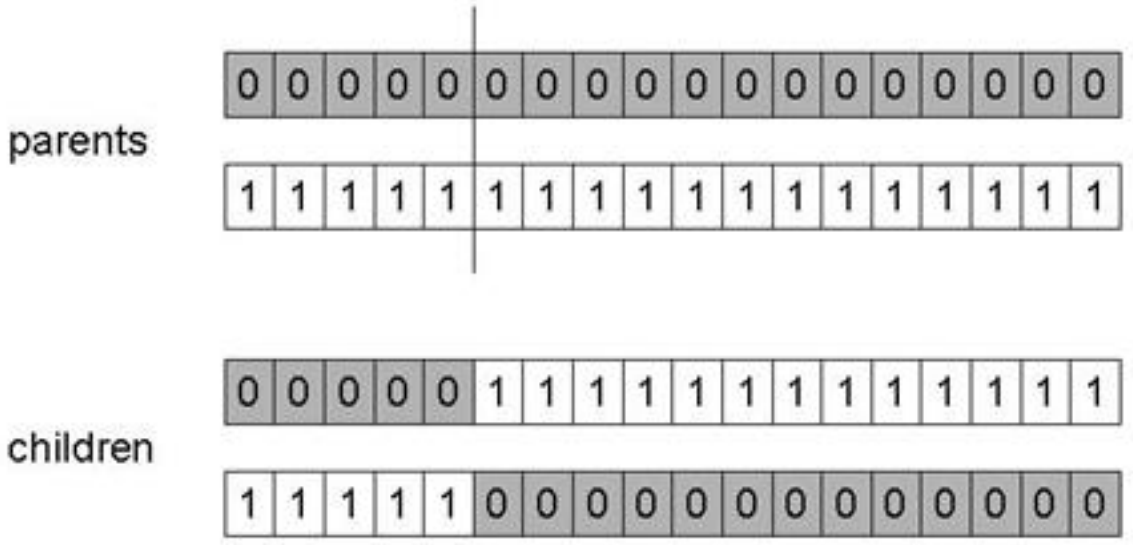
Recombinação (*Crossover*)

- **Métodos de *crossover* para codificação binária**:
  - crossover com *1* ponto de corte;
  - crossover com *n* pontos de corte;
  - crossover uniforme;
  - crossover por variável.
  - etc;

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

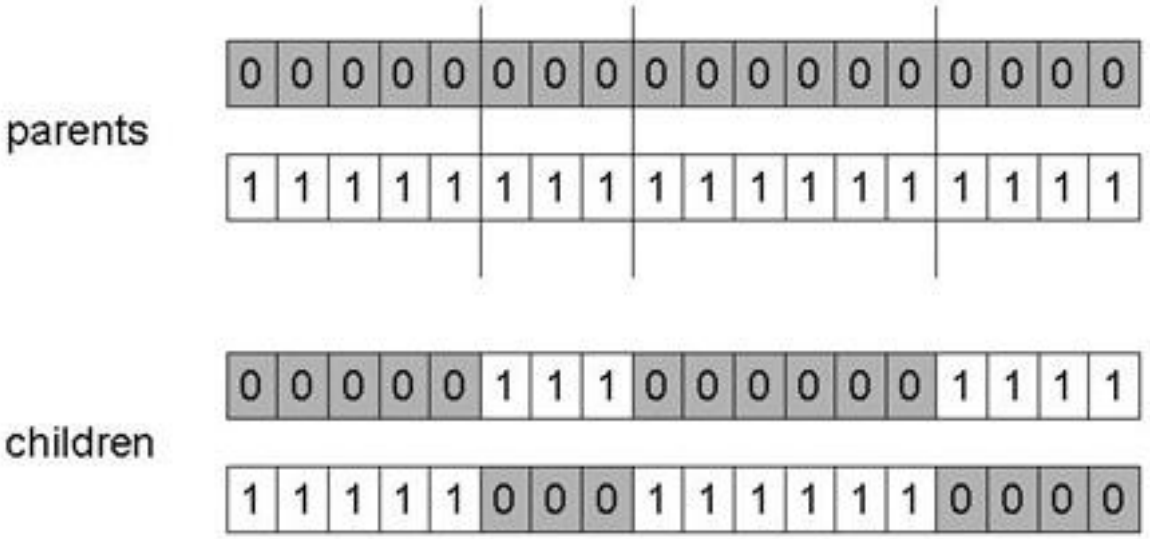
Recombinação (*Crossover*)

- ***Crossover* com 1 ponto de corte**:
  - sorteie um número aleatório no intervalo $[1, l_i - 1]$;
  - estabeleça cortes nos cromossomos pais no *ponto* sorteado;
  - gere os filhos a partir da troca das caudas dos pais;
  - tipicamente, $p_C \in [0.6, 0.9]$.



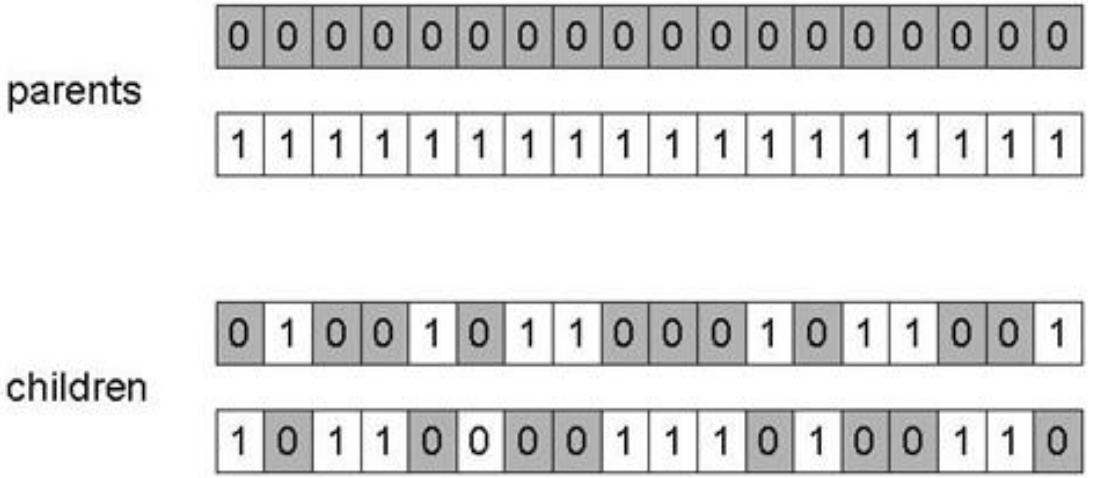
Recombinação (*Crossover*)

- ***Crossover* com *n* pontos de corte**:
  - sorteie *n* números aleatórios no intervalo $[1, l_i - 1]$;
  - estabeleça cortes nos cromossomos pais nos pontos sorteados;
  - gere os filhos a partir de trocas alternadas entre as partes;



Recombinação (*Crossover*)

- ***Crossover* uniforme**:
  - atribua "cara" p/ um dos pais e "coroa" para o outro;
  - para cada um dos genes do 1o filho: "jogue a moeda" $(prob = 0.5)$;
  - faça uma cópia inversa para os genes do 2o filho.

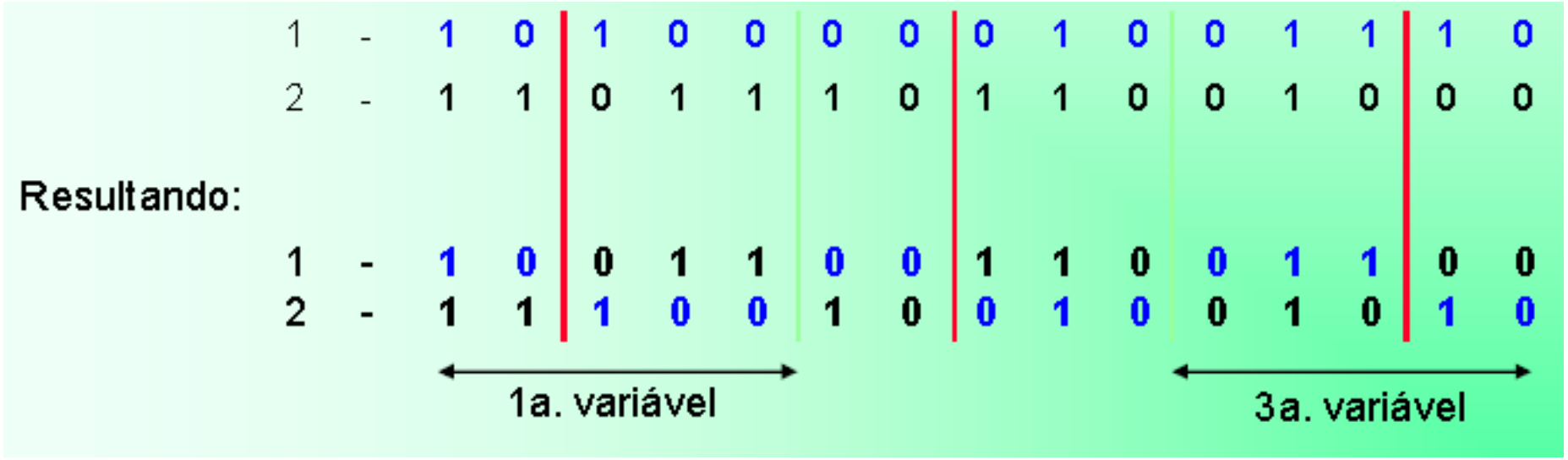


Resultado do Sorteio:

$[H\ T\ H\ H\ T\ H\ T\ T\ H\ H\ H\ T\ H\ T\ T\ H\ H\ T]$

Recombinação (*Crossover*)

- ***Crossover* por variável**:
  - atribua um ponto de corte por variável de decisão;
  - **Exemplo:** suponha que o indivíduo tenha 3 variáveis. Considere que os pontos de corte sorteados sejam 2, 2 e 3 para as variáveis respectivamente apresentadas.



Recombinação (*Crossover*)

- **Métodos de *crossover* para codificação real**:
  - *crossover* discreto:
    - sejam $\vec{x}$ *e* $\vec{y}$ dois cromossomos pais selecionados.
    - então, cada gene do filho $\vec{z}$ vem de um dos pais com igual probabilidade, ou seja $z_j = x_j$ ou y_j
    - o *crossover* discreto não insere novos valores à população.
  - *crossover* aritmético:
    - cada gene do filho $\vec{z}$ é uma combinação convexa dos genes dos pais

$$z_j = \alpha\, x_j + (1 - \alpha)\, y_j\,, \quad \alpha \in [0, 1]$$

Recombinação (*Crossover*)

- ***Crossover* Aritmético:**
  - cada gene do filho $\vec{z}$ é uma combinação convexa dos genes dos pais

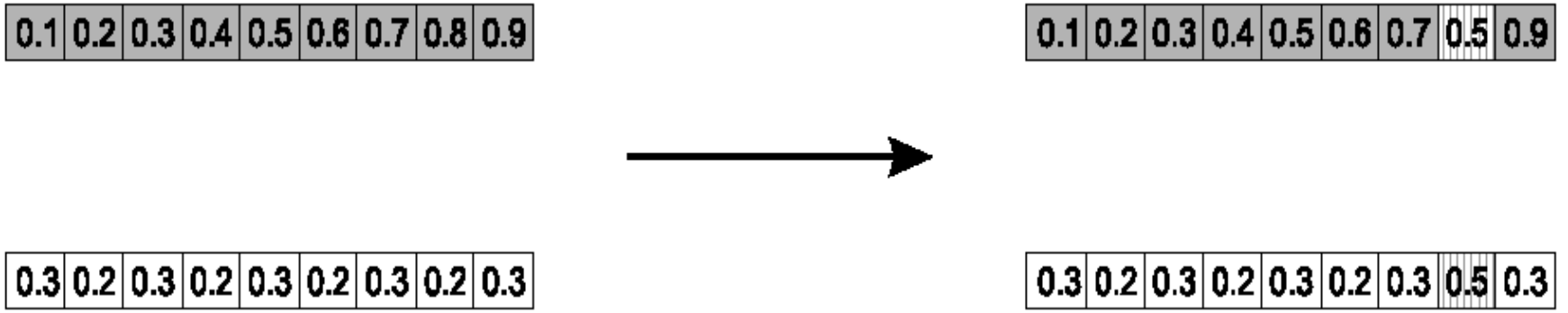
$$z_j = \alpha\, x_j + (1 - \alpha)\, y_j\,, \quad \alpha \in [0, 1]$$

  - o parâmetro α pode ser:
    - constante → *crossover* artimético uniforme;
    - variável → dependente da "idade" da população;
    - escolhido aleatoriamente a cada vez;
  - tipos de *crossover* aritmético:
    - único, simples e total;

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

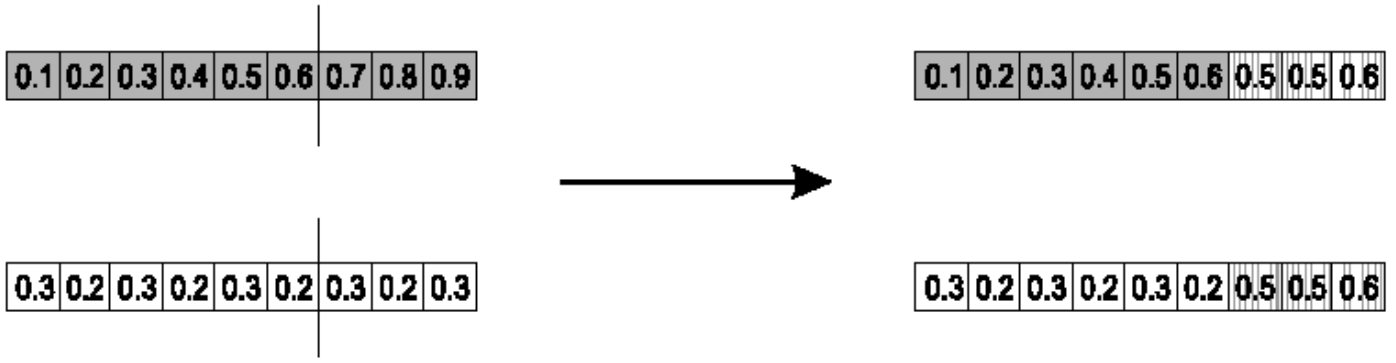
Recombinação (*Crossover*)

- ***Crossover* Aritmético Único:**
  - Pais: $\vec{x} = \langle x_1, \ldots, x_n \rangle$ e $\vec{y} = \langle y_1, \ldots, y_n \rangle$;
  - Escolha aleatoriamente um único gene *k*;
  - Filho$_1$ é: $\langle x_1, \ldots, x_{k-1}, \alpha \cdot y_k + (1-\alpha) \cdot x_k, \ldots, x_n \rangle$
  - faça a operação inversa para o outro filho.
  - Ex: supondo $\alpha = 0.5$



Recombinação (*Crossover*)

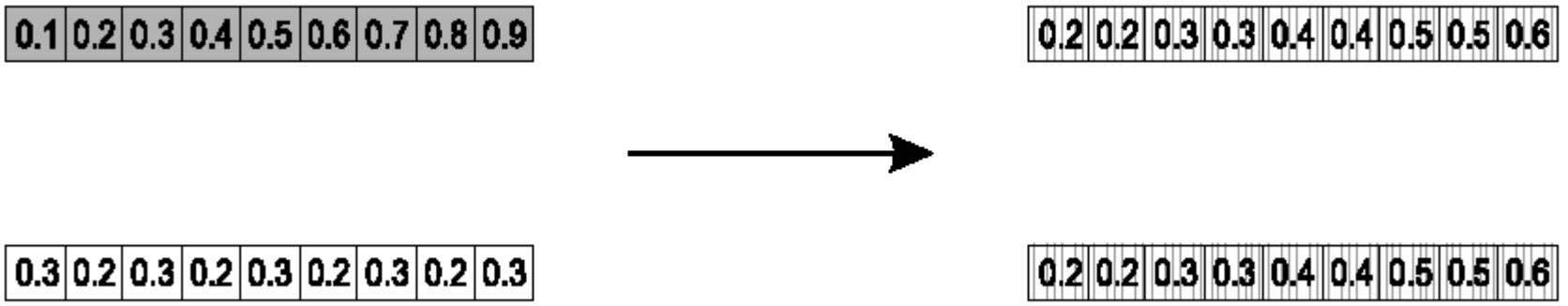
- ***Crossover* Aritmético Simples:**
  - Pais: $\vec{x} = \langle x_1, \ldots, x_n \rangle$ e $\vec{y} = \langle y_1, \ldots, y_n \rangle$;
  - Escolha aleatoriamente um único gene *k e*, após este ponto combine os genes dos pais;
  - Filho$_1$ é: $\left\langle x_1, \ldots, x_k, \alpha \cdot y_{k+1} + (1-\alpha) \cdot x_{k+1}, \ldots, \alpha \cdot y_n + (1-\alpha) \cdot x_n \right\rangle$
  - faça a operação inversa para o outro filho.
  - Ex: supondo $\alpha = 0.5$



UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Recombinação (*Crossover*)

- ***Crossover* Aritmético Total:**
  - mais comumente usado;
  - Pais: $\vec{x} = \langle x_1, \ldots, x_n \rangle$ e $\vec{y} = \langle y_1, \ldots, y_n \rangle$;
  - Filho$_1$ é: $\vec{z} = \alpha\vec{y} + (1 - \alpha)\vec{x}$
  - Filho$_2$ é: $\vec{z} = \alpha\vec{x} + (1 - \alpha)\vec{y}$
  - Ex: supondo $\alpha^1 = 0.5$



1. p/ $\alpha = 0.5$, os dois filhos serão idênticos !!!

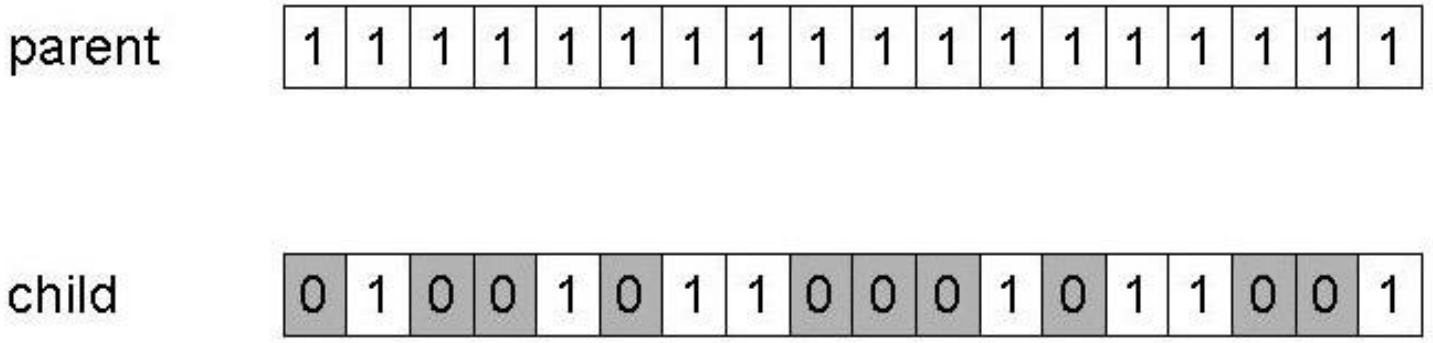
Recombinação (*Crossover*)

- **outro método de *crossover* para codificação real:**
  - *Simulated binary crossover (SBX)* (Deb and Agrawal, 1995).
    - Ver tutorial sobre *SBX* em:

      http://pt.slideshare.net/paskorn/self-adaptive-simulated-binary-crossover-presentation

Mutação

- **Mutação para codificação binária**:
  - *Bit-flip*: altera cada gene ($0 \rightarrow 1; 1 \rightarrow 0$) com uma probabilidade p_M;
  - tipicamente, $\frac{1}{popSize} \leq p_M \leq \frac{1}{chromossomeLength}$
  - para um cromossomo de tamanho l, na média $l \times p_M$ bits são alterados;



Mutação

- **Mutação para codificação real**:
  - Esquema Geral:

$$\vec{x} = \langle x_1, \ldots, x_n \rangle \rightarrow \langle x'_1, \ldots, x'_n \rangle, \quad x_i, x'_i \in [L_i, U_i]$$

  - **Mutação Uniforme**:
    - x'_i é amostrado uniformemente de $[L_i, U_i]$;
    - análoga a *bit-flipping* para codificação binária;
  - **Mutação não-uniforme**;

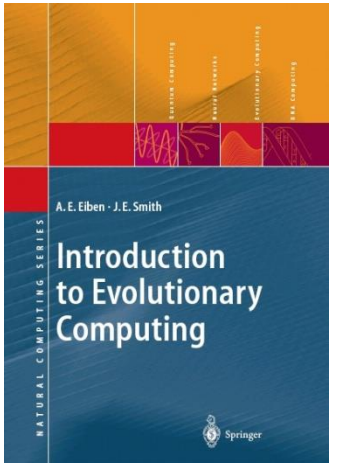
Mutação

- **Mutação não-uniforme:**
  - adiciona a cada gene x_i uma quantidade obtida de uma distribuição:
    - geralmente adota-se $\sim N(0, \sigma)$ onde σ deve ser definido a priori.
    - se o valor resultante x'_i ultrapassa $[L_i, U_i]$ então cortes são necessários para adequação ao domínio da variável;
    - desde que em uma distribuição Gaussina 2/3 das amostras estão entre $[-\sigma, \sigma]$
      - a maior parte das alterações são pequenas;
      - no entanto, existe probabilidade não-nula de se gerar alterações grandes;

UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Recombinação (*Crossover*)

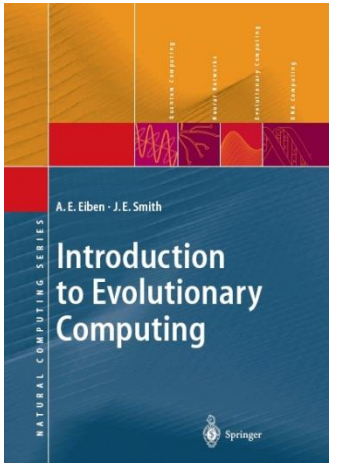
- **Principais operadores de *crossover* para codificação por Permutação**:
  - *Partially Mapped Crossover*;
  - *Edge Crossover*;
  - *Order Crossover*;
  - *Cycle Crossover*.



  - ver Seção 3.5.4 (Capítulo 3) do Livro:
    - A.E. EIBEN, J.E. SMITH, Introduction to Evolutionary Computing (Natural Computing Series), Springer.

Mutação

- **Principais operadores de mutação para codificação por Permutação**:
  - *Swap Mutation*;
  - *Insert Mutation*;
  - *Scramble Mutation*;
  - *Inversion Mutation*.



  - ver Seção 3.4.4 (Capítulo 3) do Livro:
    - A.E. EIBEN, J.E. SMITH, Introduction to Evolutionary Computing (Natural Computing Series), Springer.

UFMG

UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Referências

- **Leitura Recomendada**:
  - Capítulo 3 do Livro:

  A.E. EIBEN, J.E. SMITH, Introduction to Evolutionary Computing (Natural Computing Series), Springer.