

# 1. Computação Evolutiva

**Prof. Renato Tinós**

Programa de Pós-Graduação Em  
Computação Aplicada

Depto. de Computação e Matemática  
(FFCLRP/USP)



# 1.2. Algoritmos Genéticos

1.2.1. Introdução

1.2.2. Elementos de Algoritmos Genéticos (AGs)

1.2.3. Projeto de AGs

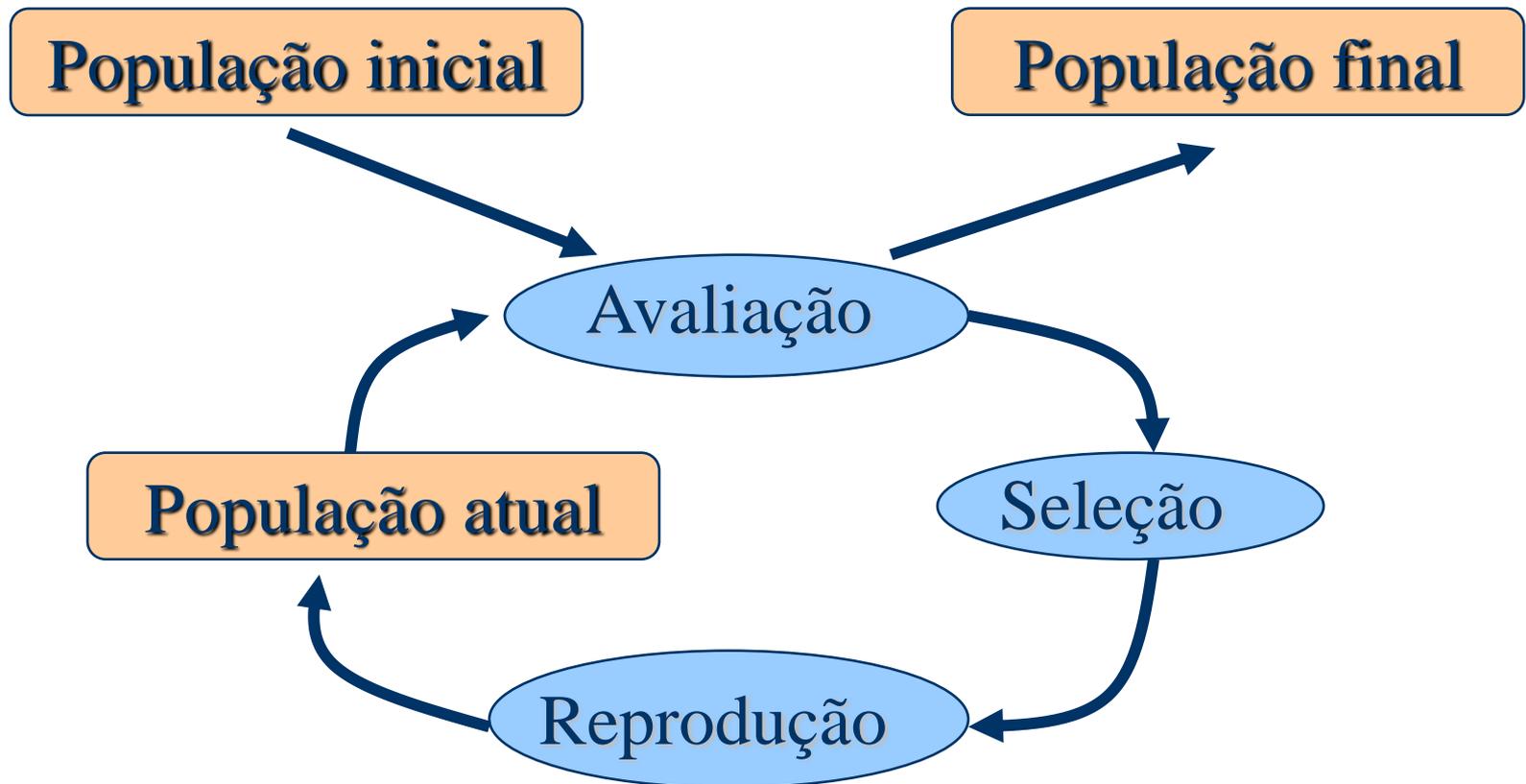
1.2.4. Exemplos

1.2.5. Implementação Computacional

## 1.2.1. Introdução

- Início
  - Desenvolvido inicialmente por John Holland
  - Uso em problemas de otimização: Keneth DeJong
  - Popularizado por David Goldberg
  - Aplicado tipicamente em problemas discretos
    - Foi originalmente desenvolvido para lidar com cromossomos binários

## 1.2.1. Introdução



## 1.2.1. Introdução

### Algoritmo Genético Básico

Início

inicialize a população  
avale a população inicial

repita

se critério de convergência for satisfeito  
interrompa

fim se

selecione indivíduos para a nova população  
aplique mutação e cruzamento nos indivíduos

selecionados

avale os indivíduos da nova população

fim repita

Fim

## 1.2.2. Elementos de AGs

- Elementos
  - População
  - Codificação
  - Função de avaliação
  - Reprodução

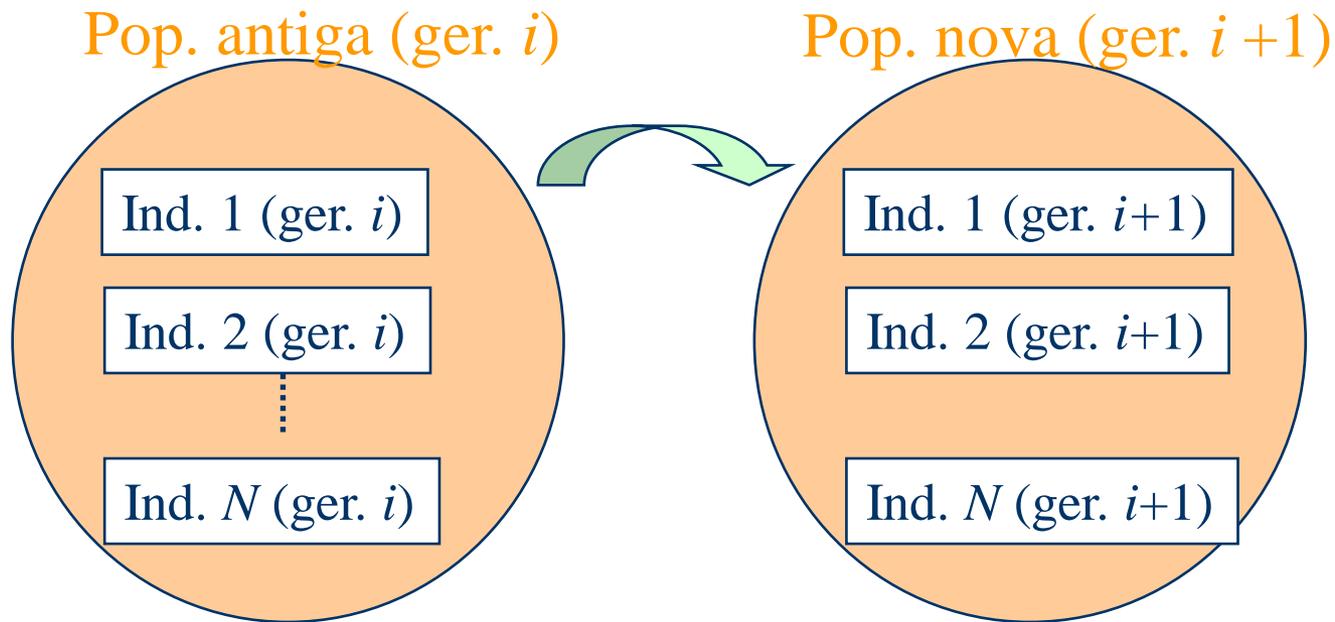
## 1.2.2. Elementos de AGs

- População
  - Formada por **indivíduos**
    - Possíveis soluções para um dado problema
    - Codificados em **cromossomos**
      - *strings* (vetores)
      - Apesar de algumas implementações considerarem indivíduos com mais de um cromossomo, o AG padrão considera que um indivíduo é formado por apenas um cromossomo
    - Cada elemento do cromossomo do AG é chamado de **gene**
      - Os genes podem assumir valores ou símbolos ( **alelos** )

## 1.2.2. Elementos de AGs

- População

- Existe geralmente um número fixo de indivíduos em uma população
  - Em cada geração, a população velha é substituída por uma população nova (com novos indivíduos)



## 1.2.2. Elementos de AGs

- População
  - AG Padrão usa o modelo geracional
    - Cada indivíduo sobrevive por exatamente uma geração
    - Toda a população de pais é substituída pelos filhos
  - No outro extremo, está o modelo estacionário (*Steady-state*) no qual um filho é gerado por geração para substituir um pai

## 1.2.2. Elementos de AGs

### Algoritmo Evolutivo Geracional

#### Início

$t \leftarrow 1$

inicializePopulacao(  $\mathbf{P}_t$  )

avaliePopulacao(  $\mathbf{P}_t$  )

**enquanto** ( criterioConvergencia == 0 )

$\mathbf{P}_{t+1} \leftarrow \text{selecao}( \mathbf{P}_t )$

$\mathbf{P}_{t+1} \leftarrow \text{transformePopulacao}( \mathbf{P}_{t+1} )$

avaliePopulacao(  $\mathbf{P}_{t+1}$  )

$t \leftarrow t + 1$

**fim enquanto**

**Fim**

## 1.2.2. Elementos de AGs

### Algoritmo Evolutivo Estacionário

#### Início

$t \leftarrow 1$

inicializePopulacao(  $\mathbf{P}_t$  )

avalePopulacao(  $\mathbf{P}_t$  )

**enquanto** ( criterioConvergencia == 0 )

$\mathbf{Q}_t \leftarrow$  transformePopulacao(  $\mathbf{P}_t$  )

    avalePopulacao(  $\mathbf{Q}_t$  )

$\mathbf{P}_{t+1} \leftarrow$  selecao(  $\mathbf{P}_t \cup \mathbf{Q}_t$  )

$t \leftarrow t + 1$

**fim enquanto**

**Fim**

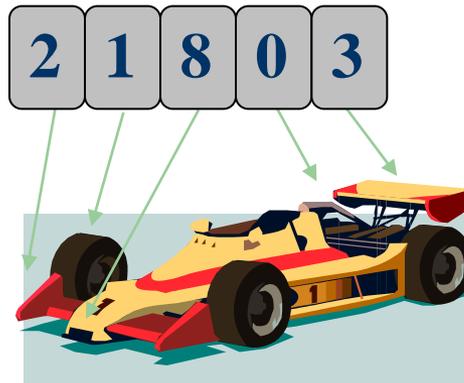
## 1.2.2. Elementos de AGs

- Codificação
  - Cada indivíduo é codificado por um conjunto de genes que definem as características do indivíduo
    - Genótipo
      - Conjunto de parâmetros (genes) que define um indivíduo
    - Fenótipo
      - Produto da interação de todos os genes

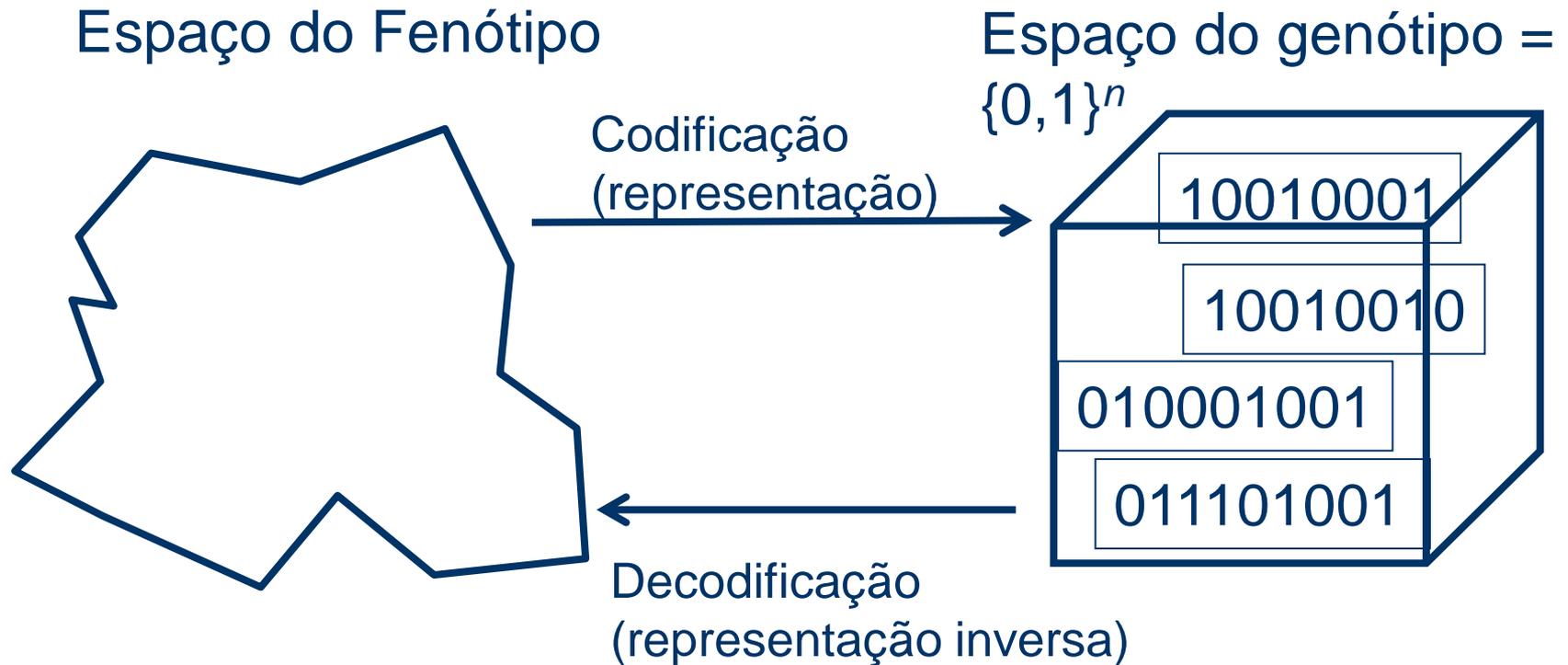
## 1.2.2. Elementos de AGs

- Codificação
  - Genes são combinados para formar *strings* ou vetores
    - Exemplo:

$$x_i = [ 2 \ 1 \ 8 \ 0 \ 3 ]^T$$



## 1.2.2. Elementos de AGs



## 1.2.2. Elementos de AGs

- Codificação

- Genes podem ser representados por:

- Números Binários ( 0 ; 1 )

- São tradicionalmente usados

- Exemplo:  $x_i = [ 0 \ 1 \ 0 \ 1 \ 1 \ 1 ]^T$

- Podem ser utilizados para codificar outras representações

- Inteiros. Exemplo:  $x_i = [ 0 \ 1 \ 0 ]^T$  pode codificar o número inteiro 2

- Reais

- Caracteres: BCD, ASCII, ...

- Etc...

## 1.2.2. Elementos de AGs

- Codificação
  - Genes podem ser representados por:
    - Números Inteiros (... ; -1 ; 0 ; 1 ; 2 ; ...)
      - Exemplo:  $x_j = [-1 \ 10 \ 2 \ -3 \ -98 \ 1]^T$
    - Números Reais
      - Exemplo:  $x_j = [-1,23 \ 10,65 \ 2,99]^T$
    - Caracteres ( A ; B ; ... )
      - Exemplo:  $x_j = [t \ e \ s \ t \ e]^T$
    - Outros
      - combinação de outras representações
      - números complexos
      - etc...

## 1.2.2. Elementos de AGs

- Codificação
  - Permutação
    - Tarefas no qual uma sequência de objetos aparece em uma certa ordem
      - Se existe  $n$  variáveis, então a representação é uma lista de  $n$  inteiros, cada qual aparecendo apenas uma vez

## 1.2.2. Elementos de AGs

- Codificação
  - Permutação
    - Exemplo: Problema do Caixeiro Viajante
      - Nomeie as cidades como  $1, 2, \dots, n$
      - Um *tour* é uma permutação
        - Ex.: para  $n=4$   
[1,2,3,4], [3,4,2,1]
      - Espaço de busca é muito grande
        - Ex.: para 30 cidades existe  $30!$   
 $\approx 10^{32}$  possíveis *tours*



## 1.2.2. Elementos de AGs

- Função de Avaliação
  - Também conhecida como **Função de Avaliação**, **Função de Aptidão** ou **Função de *Fitness***
  - Mede o grau de aptidão (*fitness*) da solução (indivíduo)
    - É aplicada ao fenótipo do indivíduo
      - O genótipo deve ser decodificado para que a aptidão do indivíduo seja calculada. Exemplo
        - genótipo:  $\mathbf{x}_i = [0\ 1\ 0\ 1]^T$
        - fenótipo:  $z_i = 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 5$
        - aptidão:  $f(z_i) = 1 / (1 + z_i^2) = 0,0385$

## 1.2.2. Elementos de AGs

- **Função de Avaliação**
  - Cada problema tem sua própria função de avaliação
    - Dada de acordo com os requisitos para a solução do problema
      - Exemplo: projeto de ponte
        - Menor custo
        - Menor tempo de construção
        - Maior capacidade de carga

## 1.2.2. Elementos de AGs

- Função de Avaliação
  - Muitas vezes, a função de avaliação não é conhecida (ou é difícil de ser computada com precisão)
    - Deve ser possível, no entanto, obter a aptidão do indivíduo através do seu genótipo
  - A aptidão de um indivíduo é importantíssima no processo de otimização
    - Define quais indivíduos serão selecionados para se reproduzirem, gerando a nova população

## 1.2.2. Elementos de AGs

- Seleção
  - Existem diversos métodos para a seleção de indivíduos para a fase de reprodução
    - Exemplos:
      - Elitismo
      - Método da Roleta
      - *Rank Selection*
      - Seleção por Torneio

## 1.2.2. Elementos de AGs

- Seleção
  - Elitismo
    - Indivíduos com maior aptidão são automaticamente selecionados
    - Utilizado para que os melhores indivíduos não desapareçam do processo de otimização
    - Geralmente, além de selecionar os melhores indivíduos, evita que estes sofram modificações pelos operadores genéticos

## 1.2.2. Elementos de AGs

- Seleção
  - Método da Roleta
    - A seleção natural
      - Trabalha com probabilidades
        - As vezes, um indivíduo muito bom não sobrevive porque algum processo externo o afetou
      - Escolhe preferencialmente, embora não exclusivamente, indivíduos com maior aptidão
        - Indivíduos mais aptos têm mais chances de serem reproduzidos
        - As vezes, um indivíduo muito bom pode ser filho do cruzamento de um indivíduo bom com um ruim

## 1.2.2. Elementos de AGs

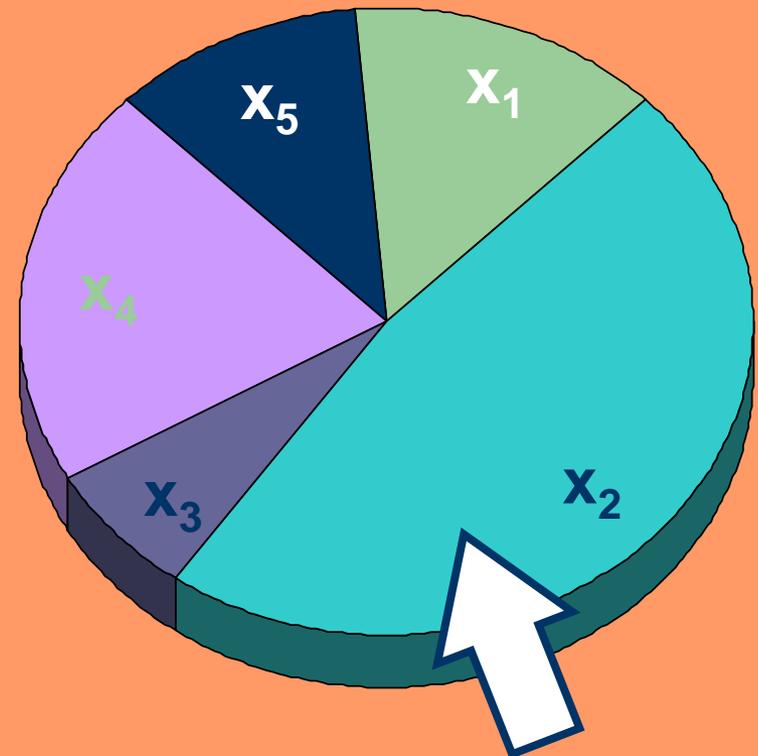
- Seleção
  - Método da Roleta
    - Indivíduos com maior aptidão tem mais chances de serem reproduzidos
    - A probabilidade de um indivíduo ser escolhido para se reproduzir é dada por sua aptidão relativa
      - Aptidão do indivíduo normalizada pela soma das aptidões de todos os indivíduos da população

$$f_r(z_i) = \frac{f(z_i)}{\sum_{j=1}^N f(z_j)}$$

## 1.2.2. Elementos de AGs

### Método da Roleta baseado em Aptidão Relativa

$x_i$	$f(z_i)$	$f_r(z_i)$
$x_1$ 10110	2,23	0,14
$x_2$ 11000	7,27	0,47
$x_3$ 11110	1,05	0,07
$x_4$ 01001	3,35	0,21
$x_5$ 00110	1,69	0,11



## 1.2.2. Elementos de AGs

- Seleção
  - Método da Roleta
    - Indivíduos são escolhidos pelo método da roleta até que o número máximo de indivíduos permitido em uma população seja alcançado
    - O nível de diversidade da população é maior quando o método da roleta é empregado (em relação ao elitismo)
      - No entanto, mesmo no método da roleta, o nível de diversidade da população decresce no decorrer do processo de otimização

## 1.2.2. Elementos de AGs

- Seleção

- **Rank Selection**

- Indivíduos são primeiramente ordenados de acordo com o *fitness*
- São então ranqueados, sendo que o pior indivíduo tem rank 0 e o melhor tem rank  $\mu-1$ , sendo  $\mu$  o número de indivíduos na população
- Probabilidade de seleção do  $i$ -ésimo indivíduo é dada por

$$P_{lin-rank}(i) = \frac{(2-s)}{\mu} + \frac{2i(s-1)}{\mu(\mu-1)}$$

na qual o parâmetro  $s$ :  $1,0 < s \leq 2,0$  controla a pressão seletiva

## 1.2.2. Elementos de AGs

- Seleção

- *Rank Selection*

- Exemplo com três indivíduos

	Fitness	Rank	$P_{selFP}$	$P_{selLR} (s = 2)$	$P_{selLR} (s = 1.5)$
A	1	<b>0</b>	0.1	0	0.167
B	5	<b>2</b>	0.5	0.67	0.5
C	4	<b>1</b>	0.4	0.33	0.33
Sum	10		1.0	1.0	1.0

## 1.2.2. Elementos de AGs

- Seleção
  - Torneio
    - No modelo mais simples,  $k$  indivíduos são escolhidos aleatoriamente na população
      - O indivíduo de maior aptidão entre eles é escolhido
    - O parâmetro  $k$  controla a pressão seletiva
    - É mais simples (computacionalmente) do que os métodos de seleção proporcionais à aptidão, como o método da roleta

## 1.2.2. Elementos de AGs

- Reprodução
  - Aplicado após a seleção de indivíduos
  - Permite a obtenção de novos indivíduos
  - Os operadores genéticos de reprodução mais comuns são
    - *Crossover* (cruzamento ou recombinação)
    - *Mutação*

## 1.2.2. Elementos de AGs

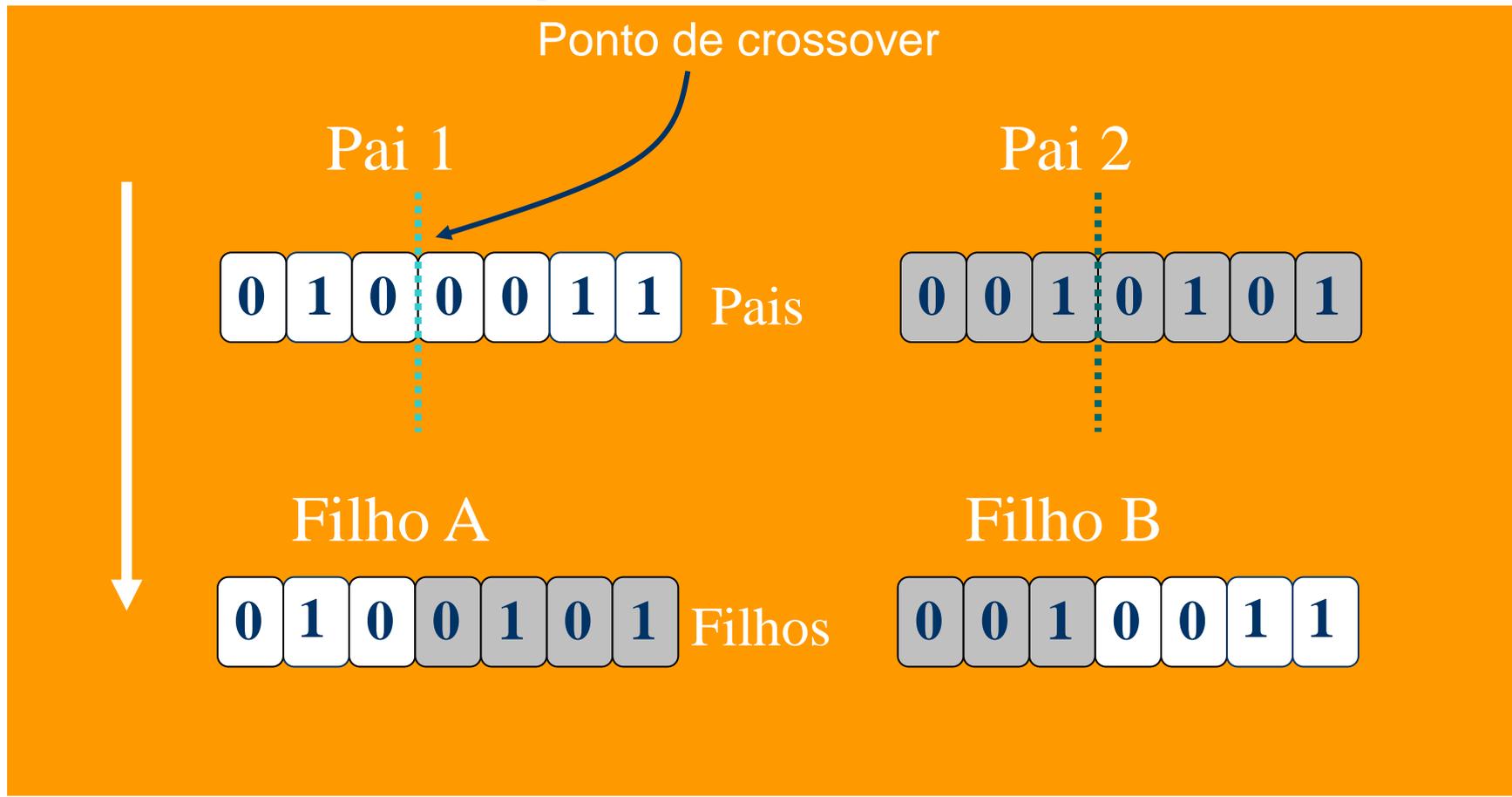
- Reprodução
  - *Crossover*
    - Recombinação de características dos pais durante a reprodução
      - Permite que as próximas gerações herdem essas características
    - Troca trechos dos cromossomos de dois indivíduos escolhidos durante a seleção
    - Ocorre com uma probabilidade definida pela *taxa de crossover*  $p_c$ 
      - Para cada par de indivíduos selecionados, gera-se um número aleatório com distribuição uniforme
      - Ocorre *crossover* no par dado se o valor deste número for menor que  $p_c$
      - Tipicamente entre 0,6 e 0,9

## 1.2.2. Elementos de AGs

- Reprodução
  - *Crossover*
    - Permite a exploração rápida do espaço de busca
    - Tipos
      - Um ponto
        - Troca trechos (entre os dois cromossomos) delimitados por um ponto escolhido aleatoriamente
      - Dois pontos
        - Troca trechos (entre os dois cromossomos) delimitados por dois pontos escolhidos aleatoriamente
      - Uniforme
        - Troca trechos (entre os dois cromossomos) gerados por uma máscara, geralmente gerada aleatoriamente

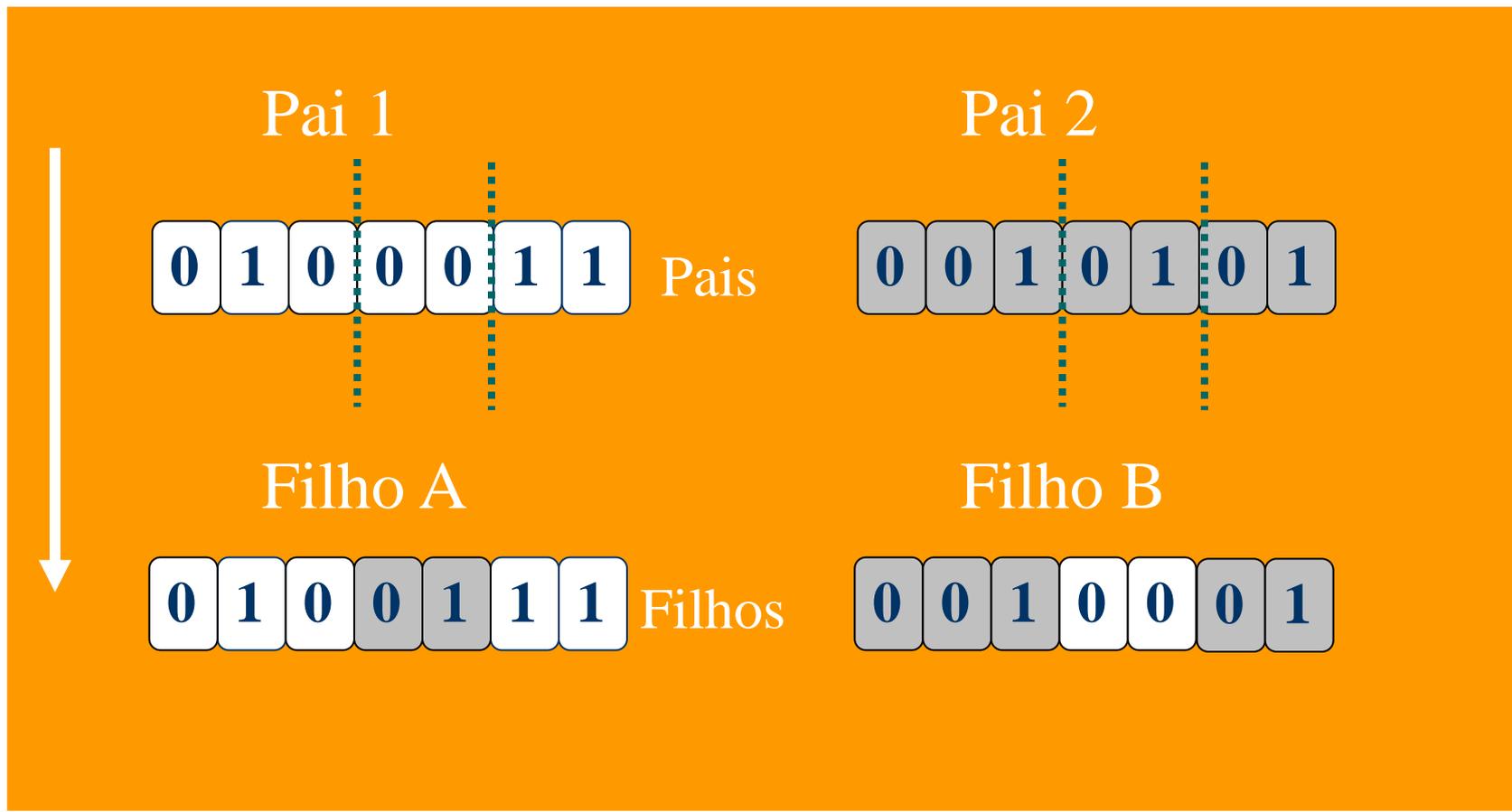
## 1.2.2. Elementos de AGs

### Crossover de 1 ponto



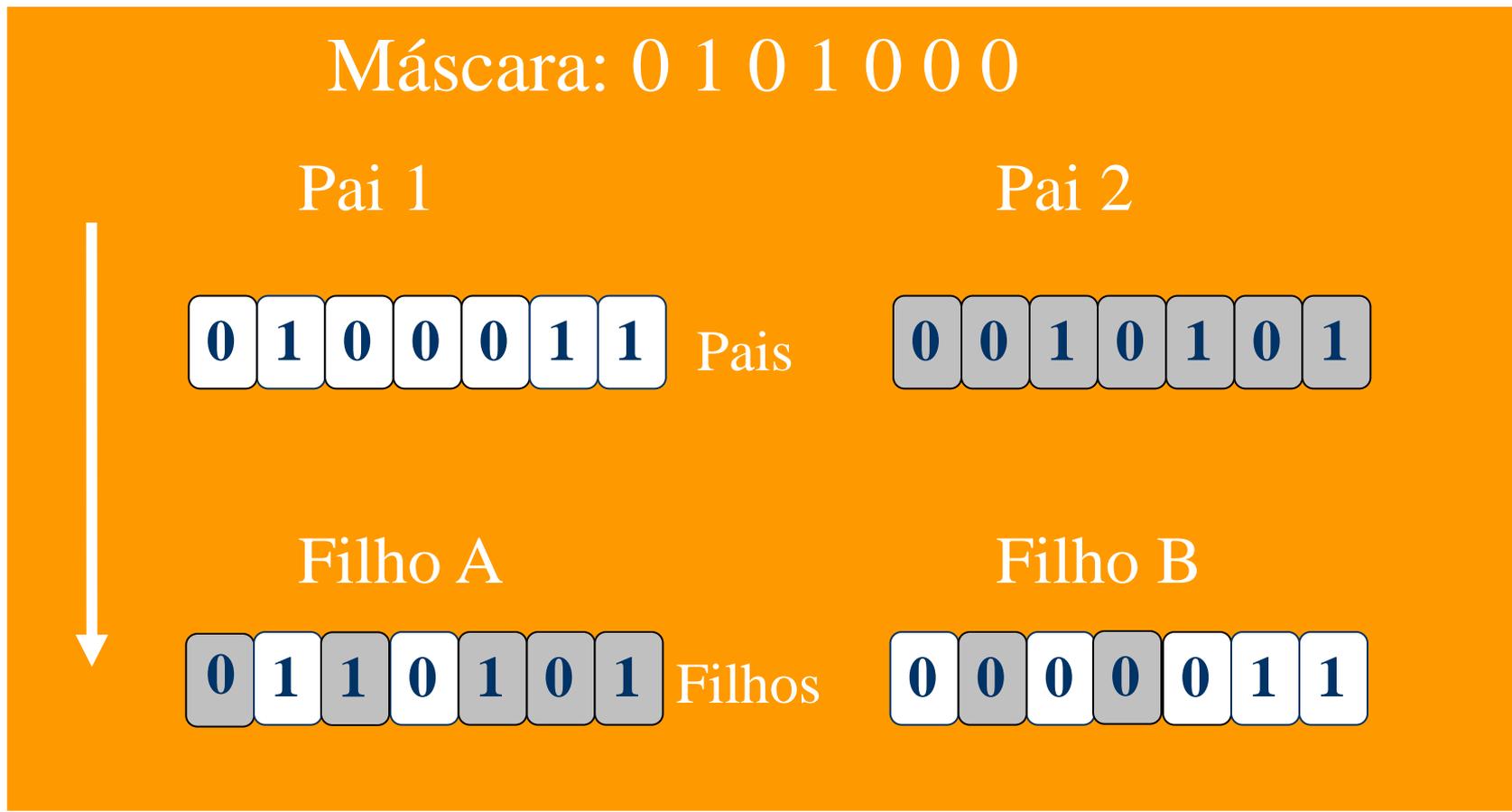
## 1.2.2. Elementos de AGs

### Crossover de 2 pontos



## 1.2.2. Elementos de AGs

### Crossover uniforme



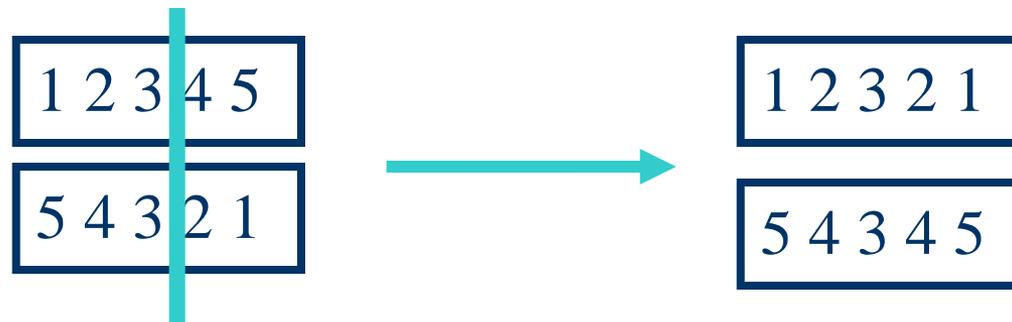
## 1.2.2. Elementos de AGs

- Reprodução

- *Crossover*

- Problemas de Permutação

- Crossover comum não pode ser aplicado



- Operadores especializados têm assim sido propostos para combinar a informação de dois pais

## 1.2.2. Elementos de AGs

- Reprodução
  - *Crossover*
    - Problemas de Permutação
      - *Exemplo: Order 1 Crossover*
        1. Copie parte arbitrária do primeiro pai para o primeiro filho
        2. Copie elementos (ex.: números) que não estão nesta primeira parte para o primeiro filho:
          - Começando a direita do corte da parte copiada
          - Usando a ordem do segundo pai
          - Voltando para o começo, caso chegue no fim do cromossomo
        3. Faça o mesmo para o segundo filho

## 1.2.2. Elementos de AGs

- Reprodução

- *Crossover*

- Problemas de Permutação

- *Order 1 Crossover* (exemplo)

- Copie parte aleatória do pai 1

1 2 3 4 5 6 7 8 9



  4 5 6 7

9 3 7 8 2 6 5 1 4

- Copie os números restantes na ordem do pai 2 (1,9,3,8,2)

1 2 3 4 5 6 7 8 9



3 8 2 4 5 6 7 1 9

9 3 7 8 2 6 5 1 4

## 1.2.2. Elementos de AGs

- Reprodução
  - Mutação
    - Gera diversidade genética
    - Altera aleatoriamente um ou mais genes no cromossomo
    - Assegura que a probabilidade de atingir qualquer ponto do espaço de busca nunca será zero

## 1.2.2. Elementos de AGs

- Reprodução

- Mutação

- Aplicada a cada gene de cada indivíduo após o *crossover* com uma taxa de mutação  $p_m$ 
      - Para cada gene de cada indivíduo selecionado, gera-se um número aleatório com distribuição uniforme
      - Ocorre mutação no gene dado se o valor deste número for menor que  $p_m$
    - Taxa de mutação é geralmente pequena
      - Tipicamente entre  $1/\text{tamanho\_populacao}$  e  $1/\text{tamanho\_cromossomo}$
      - Ex.:  $p_m = 0,01$ 
        - Gene tem probabilidade de 1% de sofrer mutação

## 1.2.2. Elementos de AGs

- Reprodução
  - Mutação
    - A alteração depende da representação do gene
      - Binária (*Bit Flip Mutation*)
        - O gene mutado recebe a negação de seu antigo valor (ou seja, se era igual a 0 fica igual a 1 e se era igual a 1 fica igual a zero)

## 1.2.2. Elementos de AGs

### Mutação Binária

Antes da mutação

0 1 0 0 0 1 1



Após a mutação

0 1 1 0 0 1 1

## 1.2.2. Elementos de AGs

- Reprodução
  - Mutação
    - Inteiro ou Real
      - O gene pode ser mutado para receber qualquer valor entre os seus limites máximos e mínimos (distribuição uniforme)
      - O gene pode ser mutado adicionando um valor aleatório com distribuição normal com média zero e desvio padrão  $\sigma$
      - O gene pode ser mutado para receber um acréscimo ou decréscimo no seu valor corrente

## 1.2.2. Elementos de AGs

- Reprodução

- Mutação

- Problemas de Permutação

- Mutação inteira comum não pode ser aplicada

- [ 4 2 3 1 } -> [ 3 2 3 1 ]

- Algumas soluções

- Trocar dois valores aleatoriamente

1 2 3 4 5 6 7 8 9      →      1 5 3 4 2 6 7 8 9

- Inversão

1 2 3 4 5 6 7 8 9      →      1 5 4 3 2 6 7 8 9

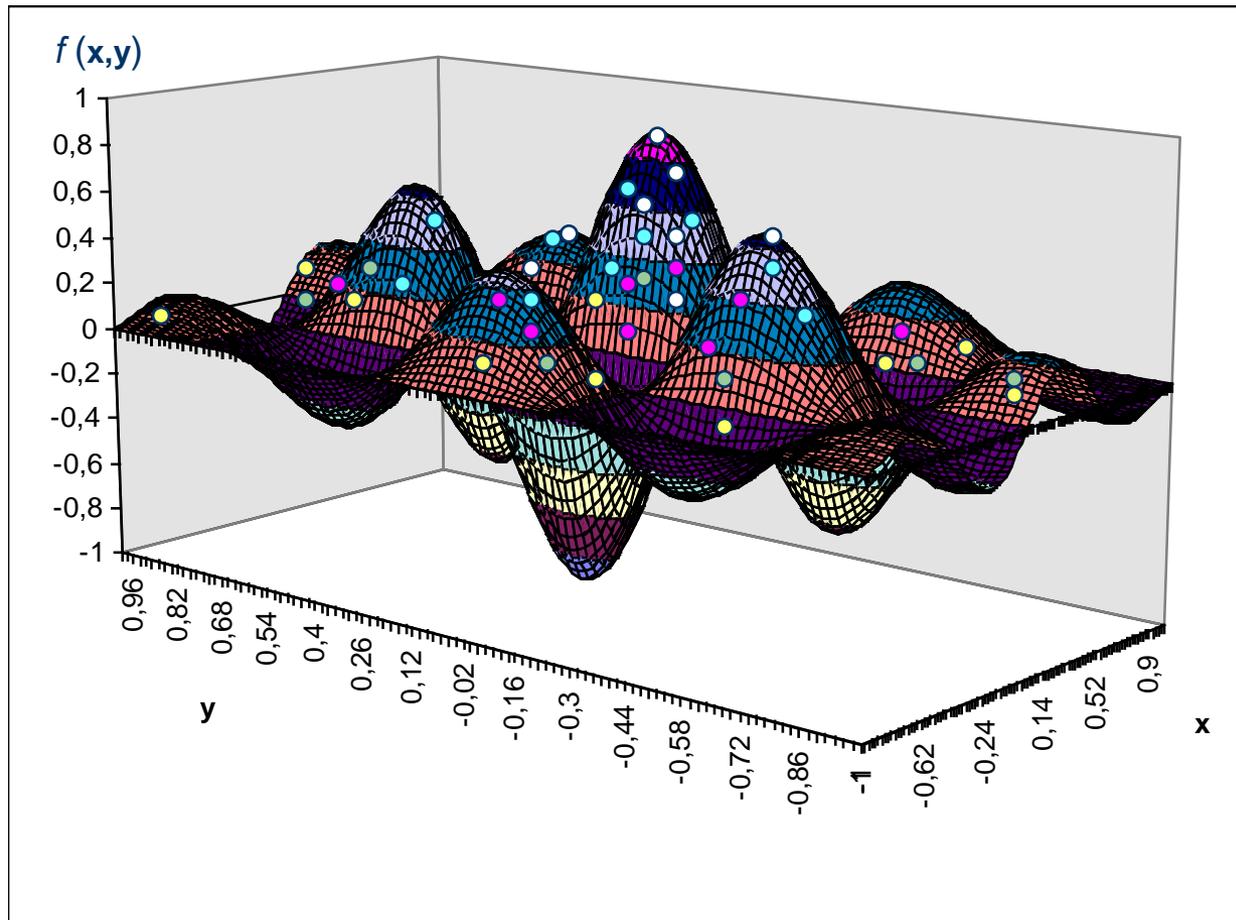
- Rearranjo

1 2 3 4 5 6 7 8 9      →      1 3 5 4 2 6 7 8 9

## 1.2.2. Elementos de AGs

- Convergência
  - Se o AG estiver corretamente implementado, a população deve evoluir em gerações sucessivas
  - Aptidão do melhor indivíduo e da média da população devem aumentar em direção a um ótimo global

## 1.2.3. Projeto de AGs



Genótipo:

$x$	$y$
-----	-----

## 1.2.3. Projeto de AGs

- Critérios de Parada
  - Tempo de execução
  - Número de gerações
  - Valor de aptidão mínimo e/ou médio
  - Convergência
    - Nas últimas  $k$  iterações não houve melhora nas aptidões

## 1.2.3. Projeto de AGs

- Escolha dos parâmetros
  - Quantos indivíduos em uma população?
    - Poucos: diminuição rápida da diversidade
    - Muitos: aumento do tempo de computação
  - Qual a taxa de mutação?
    - Baixa  $\Rightarrow$  mudanças lentas
    - Alta  $\Rightarrow$  instabilidade, pois os traços não são mantidos por um período suficiente
  - Qual a taxa de crossover?

## 1.2.3. Projeto de AGs

- Quais tipos de operadores genéticos devem ser utilizados?
  - Depende do problema
  - Em geral crossover e mutação
    - Mutação sozinha pode ser utilizada, mas crossover sozinho não deve ser utilizado
- Crossover x Mutação
  - *Exploration*: descoberta de áreas promissoras no espaço de busca
    - ganho de informação sobre o problema
    - Crossover
      - Em geral, produz grandes saltos em uma área entre as regiões ocupadas pelos dois pais
  - *Exploitation*: otimização dentro de uma área promissora
    - uso da informação sobre o problema
    - Mutação
      - Em geral, cria pequenos desvios aleatórios na solução dada pelo pai (vasculha as soluções vizinhas à solução pai)
  - Em geral, deve haver um compromisso entre *exploration* e *exproitation*
    - Mas qual?

## 1.2.3. Projeto de AGs

- Qual codificação deve ser utilizada?
  - Inteira, real ou binária?
  - Binária ou Código Gray?
    - Exemplo: 3 bits

<b>Decimal</b>	<b>Gray</b>	<b>Binária</b>
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111

## 1.2.3. Projeto de AGs

- População Inicial
  - Geralmente a população inicial é aleatória
    - **Necessidade de executar o algoritmo várias vezes**  
(com diferentes sementes aleatórias)
  - Conhecimento pode ser inserido

## 1.2.3. Projeto de AGs

- Quando AGs geralmente não devem ser utilizados
  - Em problemas que podem ser solucionados por algoritmos deterministas em tempo razoável (e geralmente menor do que para os AGs)
    - Ou seja em que o ótimo global com certeza será encontrado
    - Exemplos
      - Grande parte dos problemas da Classe Polinomial ( $O(n^k)$ )
      - Grande parte dos problemas unimodais

## 1.2.3. Projeto de AGs

- Dificuldades
  - Custo computacional geralmente alto quando comparado com algoritmos tradicionais
  - Convergência prematura
    - AGs comportam-se como algoritmos de busca local

# Comentários

- Referências
  - Mitchell, M. *An introduction to genetic algorithms*. MIT Press, 1996.
    - Capítulos 1 e 5
  - Goldberg, D. E. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989
    - Capítulo 1
- Agradecimentos
  - Parte do material desta apresentação foi obtida através de
    - Material de apoio do livro Eiben, A. E. & Smith, J. E. *Introduction to Evolutionary Computation*. Springer, 2003.