# Algoritmos Genéticos

# Roteiro

- Introdução
  - Otimização
- Algoritmos Genéticos
  - Representação
  - Seleção
  - Operadores Genéticos
- Aplicação
  - Caixeiro Viajante

# Introdução

- Algoritmos Genéticos (AGs), são métodos de otimização inspirados em evolução
  - J. Holland (1975), D. Goldberg (1989)

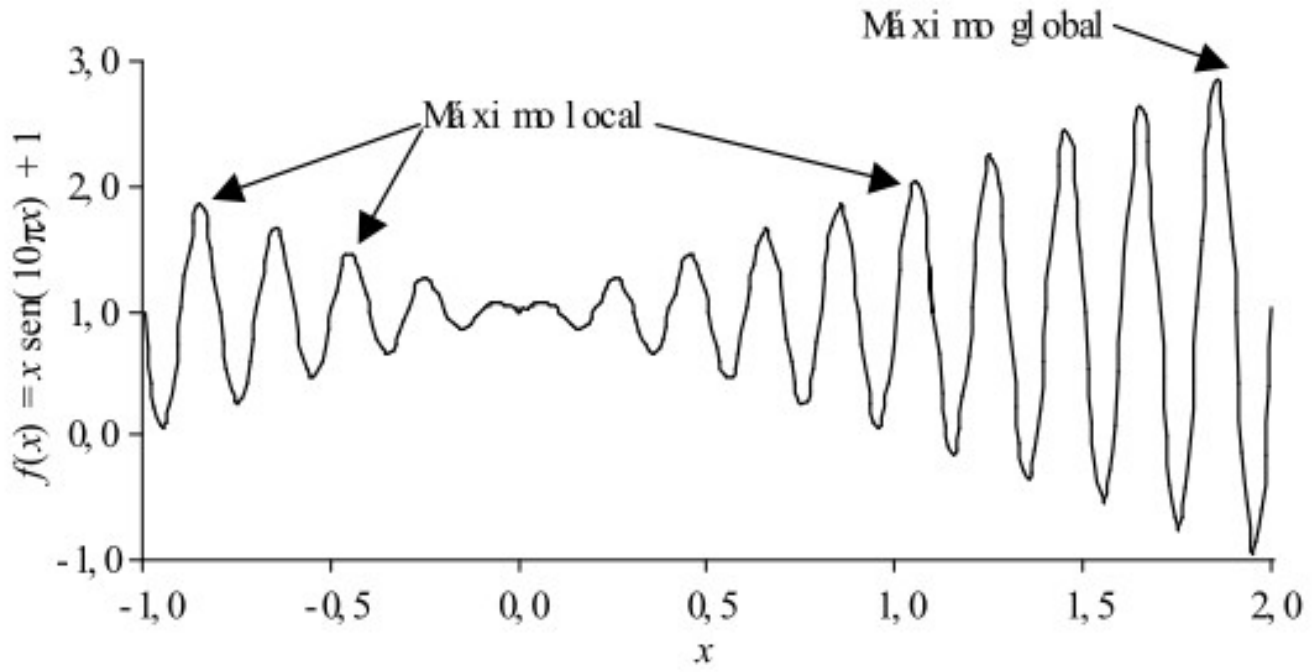
- Teoria da Evolução
  - Indivíduos mais adaptados sobrevivem e transmitem suas características para as gerações seguintes
  - Charles Darwin (Origem das Espécies, 1859)

# Otimização - Definição

- Espaço de Busca
  - Possíveis soluções de um problema
- Função Objetivo
  - Avalia cada solução com uma *nota*
- Tarefa:
  - Encontrar a solução que corresponda ao ponto de máximo (ou mínimo) da função objetivo

# Otimização - Exemplo

- Achar ponto máximo da função
  - $f(x) = x\,\mathrm{sen}(10\pi x) + 1, \quad -1 \leq x \leq 2$



# Otimização - Dificuldades

- Alguns problemas podem ter espaços de busca muito grandes

- Muitos algoritmos não são capazes de localizar ótimo global na presença de múltiplos ótimos locais
  - Ex.: Hill Climbing

# Algoritmos Genéticos

- Geração de um conjunto inicial de soluções que são iterativamente melhoradas
  - ***População*** de ***indivíduos (cromossomos)***

- Busca de soluções seguem um processo evolutivo
  - ***Seleção*** dos mais aptos +

    ***Transmissão*** de características

# Algoritmos Genéticos

- Passo 1: Geração de uma população inicial com indivíduos escolhidos aleatoriamente
- Passo 2: Avaliação dos indivíduos
  - Cálculo da função de ***fitness*** (usando a função objetivo)
- Passo 3: Seleção de indívíduos mais aptos
- Passo 4: Geração de uma nova população a partir dos indivíduos selecionados e ir para Passo 2
  - Operadores de busca (***crossover*** e ***mutação***)

# Algoritmos Genéticos

Seja $S(t)$ a população de cromossomos na geração $t$.

$t \leftarrow 0$
inicializar $S(t)$
avaliar $S(t)$
**enquanto** o critério de parada não for satisfeito **faça**
    $t \leftarrow t+1$
    selecionar $S(t)$ a partir de $S(t-1)$
    aplicar *crossover* sobre $S(t)$
    aplicar mutação sobre $S(t)$
    avaliar $S(t)$
**fim enquanto**

# Algoritmos Genéticos

- AGs são algoritmos de busca Meta-Heurística
  - I.e., algoritmo de alto nível customizável a uma ampla quantidade de problemas

- Pontos importantes a definir:
  - Representação dos invivíduos
  - Estratégia de seleção
  - Operadores de busca

# Representação de Indivíduos

- Um cromossomo representa (codifica) um conjunto de parâmetros da função objetivo
  - E.g., na função $f(x) = x\text{sen}(10\pi x) + 1$, um cromossomo codifica um valor do parâmetro x
- A representação de uma solução do espaço de busca é dependente do problema de otimização
  - Porém, alguns esquemas de representação podem ser reaproveitados

# Representação Binária

- Cromossomo representado por uma cadeia de bits (0 ou 1)
  - Cada sequência de bits é mapeada para uma solução do espaço de busca

- Representação tradicional, fácil de manipular através de operadores de busca

# Representação Binária - Exemplo

- Codificação de $-1 \leq x \leq 2$ com 22 bits
  - $2^{22}$ valores possíveis (tamanho do espaço)
  - $S_1$ = 1000101110110101000111 na base 10 seria igual a 2288967
  - Mapeado para intervalo [-1; 2] representaria a solução:
    - $x_1 = \min + (\max - \min) * b_{10} / (2^{22}-1) =$

      $-1 + (2+1) * 228896 / (2^{22}-1) = 0{,}637197$

# Representação Real

- Para otimização de parâmetros contínuos a representação binária não é adequada
  - Muitos bits para obter boa precisão numérica

- Parâmetros numéricos podem ser codificados diretamente nos cromossomos
  - Ex.: $S_1 = 0{,}637197$

# Seleção

- AGs selecionam indivíduos aptos de uma população para gerar novos indivíduos
  - ***Cromossomos filhos*** (novas soluções)

- Em geral, indivíduos pais são selecionados com uma probabilidade proporcional a seus valores de fitness
  - Probabilidade de seleção $p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$

# Seleção – Roda da Roleta

| Ind. i | Aptidão fi | Aptidão Acumulada Σ(fi) |
|---|---|---|
| 1 | 2,0 | 2,0 |
| 2 | 1,6 | 3,6 |
| 3 | 1,4 | 5,0 |
| 4 | 0,7 | 5,7 |
| 5 | 0.3 | 6,0 |

1. Ordenar aptidões da população

2. Calcular aptidões acumuladas

3. Gerar número aleatório entre [0; Última aptidão acumulada]

4. Indivíduo selecionado é o primeiro com aptidão acumulada maior que o número aleatório gerado

Exemplo: gerar número aleatório entre [0; 6]. Se 4.2 for o número gerado selecione indivíduo 2

# Seleção – Roda da Roleta

- Observação importante:
  - Não funciona para valores negativos da função de objetivo
  - Nesse caso, deve-se usar uma função de aptidão para valores positivos ou realizar ***Seleção por Torneio***

# Seleção por Torneio

- Passo 1: Escolher inicialmente com a mesma probabilidade *n* indivíduos
- Passo 2: Selecionar o cromossomo com maior aptidão dentre os *n* escolhidos
- Passo 3: Repetir passos 1 e 2 até preencher população desejada

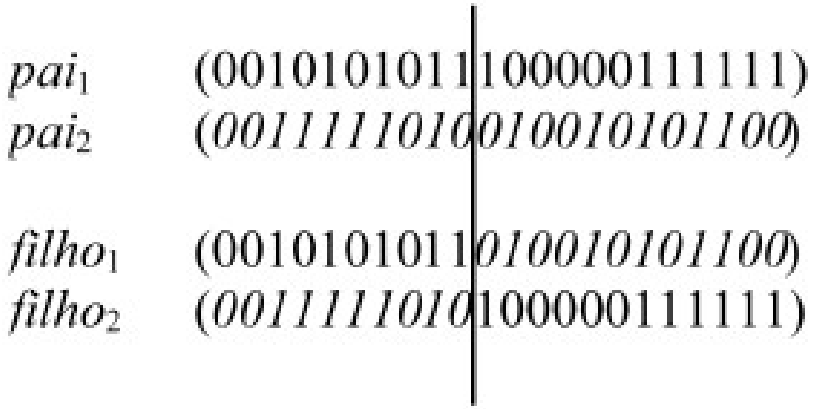
# Operadores Genéticos

- A etapa de seleção, gera uma população intermediária de potenciais cromossomos pais
- Na nova geração, escolhe-se aleatoriamente dois pais para aplicação de operadores genéticos (***crossover*** e ***mutação***)
- Produção de filhos é feita até completar o tamanho da população desejada

# Operador Crossover – Representação Binária

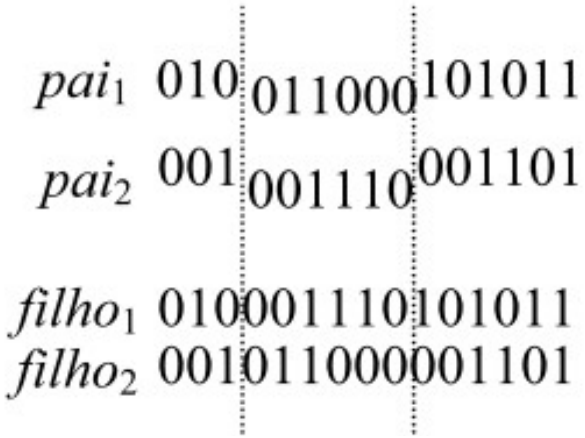
- Aplicado a um par de cromossomos retirados da população intermediária para gerar filhos
  - Filhos herdam características dois pais

- Crossover de um ponto
  - Cortar pais em uma posição aleatória e recombinar as partes geradas



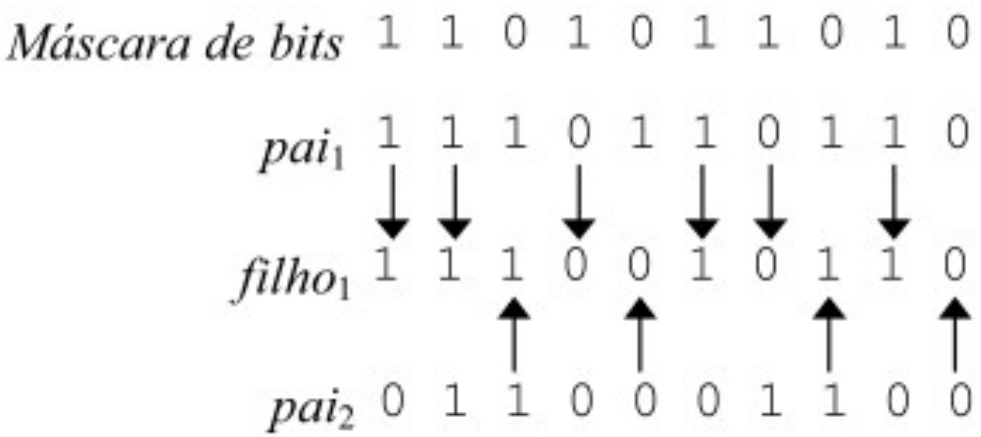
# Operador Crossover – Representação Binária

- Crossover de dois pontos
  - Cortar pais em duas posições aleatórias e recombinar as partes geradas



# Operador Crossover – Representação Binária

- Crossover uniforme
  - Gerar uma máscara de bits aleatórios e combinar os bits dos pais de acordo com a máscara gerada



# Operador Crossover – Representação Real

- Na representação real, crossover é obtido por meio de operações aritméticas sobre os pais

- Crossover média aritmética
  - Filho = (pai1 + pai2)/2

- Crossover média geométrica
  - Filho = raiz(p1*p2)

# Operador Crossover – Representação Real

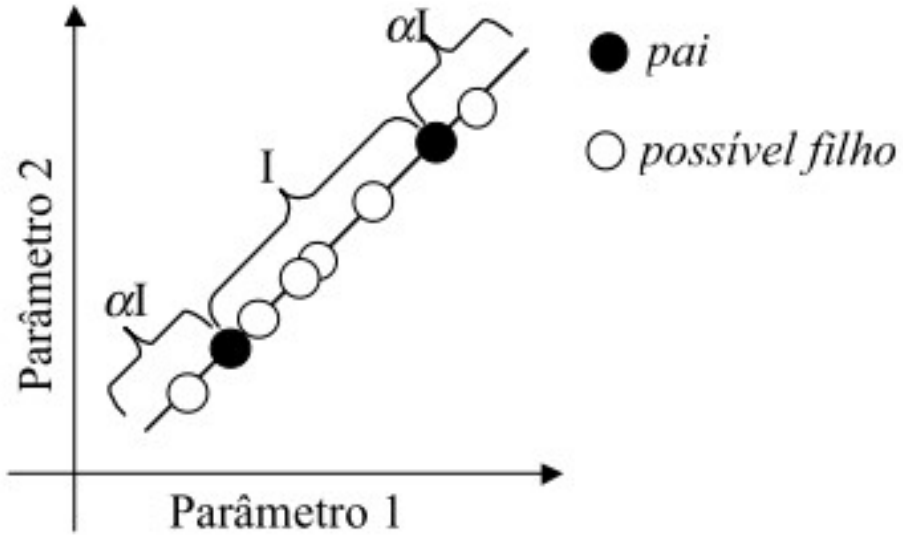
- Operadores de média tendem a diminuir muito a diversidade dos filhos
  - Filhos sempre vão estar no meio do intervalo dos pais

- Operador BLX-$\alpha$
  - Filho = pai1 + $\beta$*(pai2 – pai1)

    onde $\beta$ é um número aleatório entre [-$\alpha$, 1+ $\alpha$]
  - Parâmetro $\alpha$ controla o diversidade dos filhos

# Operador Crossover – Representação Real

- Operador BLX-α
  - α = 0 equivale a gerar filhos aleatoriamente no intervalo numérico entre os pais (I = pai2 – pai1)
  - Se α > 0, o intervalo dos possíveis filhos é estendido em α*I em ambos os lados



# Operador Crossover

- Geralmente, crossover é aplicado somente com uma dada probabilidade (***taxa de crossover***)
  - Taxa de crossover é normalmente alta (entre 60% e 90%)

- Durante a aplicação do operador, é gerado um número aleatório *r* entre 0 e 1 e aplica-se o teste:
  - Se r < taxa de crossover, então operador é aplicado
  - Senão, os filhos se tornam iguais aos pais para permitir que algumas boas soluções sejam preservadas

# Operador Mutação – Representação Binária

- A mutação é aplicada sobre os cromossomos filhos para aumentar a variabilidade da população

- Operador para representação binária:
  - Para cada bit realize ***teste de mutação*** e troque o valor do bit caso o teste seja satisfeito

| | | |
|---|---|---|
| Antes | $filho_1$ | (0010101010010010101100) |
| | $filho_2$ | (0011111011100000111111) |
| Depois | $filho_1$ | (0010**0**0101001001011**1**1100) |
| | $filho_2$ | (0011**1**1101**0**0000011111) |

Obs.: Taxa de mutação deve ser pequena (< 5%) apenas o suficiente para aumentar diversidade

# Algoritmos Genéticos – Observações Importantes

- Operador **Crossover** considera características importantes presentes nos pais
  - Aplicado a uma taxa relativamente alta, mas cuidado com efeitos destrutivos
- Operador **Mutação** explora novas características nos indivíduos que seriam possivelmente úteis
  - Aplicado a uma taxa relativamente baixa, mas dependendo do problema e operador use taxas mais altas

# Algoritmos Genéticos – Observações Importantes

- Convergência Prematura
  - Em algumas execuções, AG pode convergir para soluções iguais
    - Cromossomos com boa aptidão (mas ainda não ótimos) que geram filhos com pouca diversidade
  - Nesses casos, aconselha-se:
    - Aumento da taxa de mutação e crossover
    - Evitar a inserção de filhos duplicados

# Algoritmos Genéticos – Observações Importantes

- Critérios de Parada
  - Número máximo de gerações
  - Função objetivo com valor ótimo alcançado (quando esse valor é conhecido)
  - Convergência na função objetivo (i.e., quando não ocorre melhoria significativa da função)

# Algoritmos Genéticos – Observações Importantes

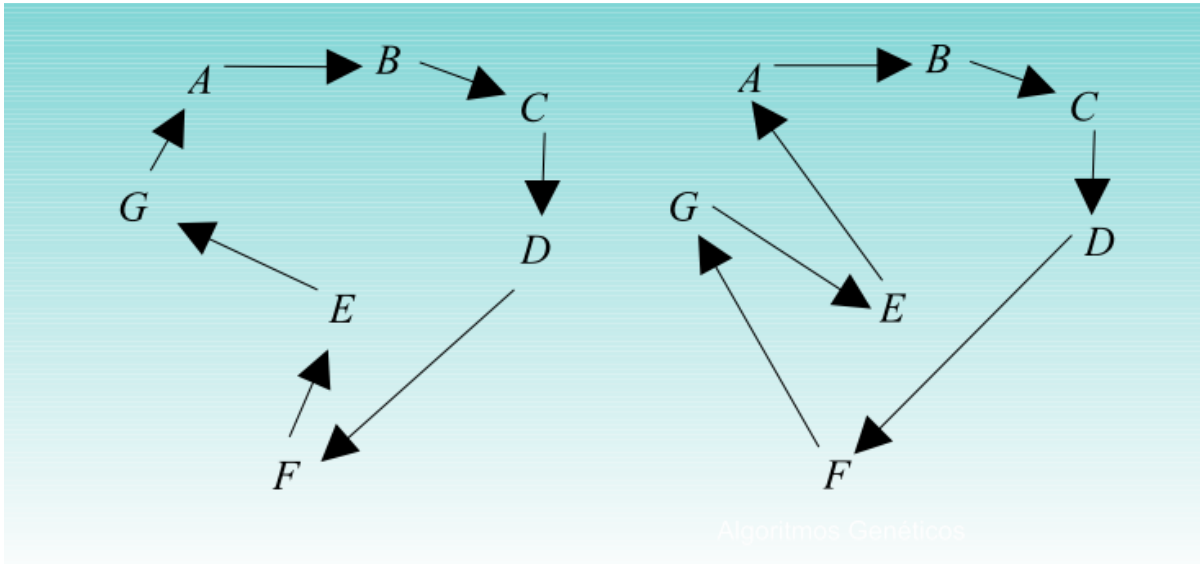
- População inicial
  - Não pode ser excessivamente pequena
    - Pouca representatividade do espaço de busca
  - Não pode ser excessivamente grande
    - Demora na convergência
  - Para melhorar a representatividade população inicial pode possuir indivíduos igualmente espaçados no espaço de busca

# Algoritmos Genéticos

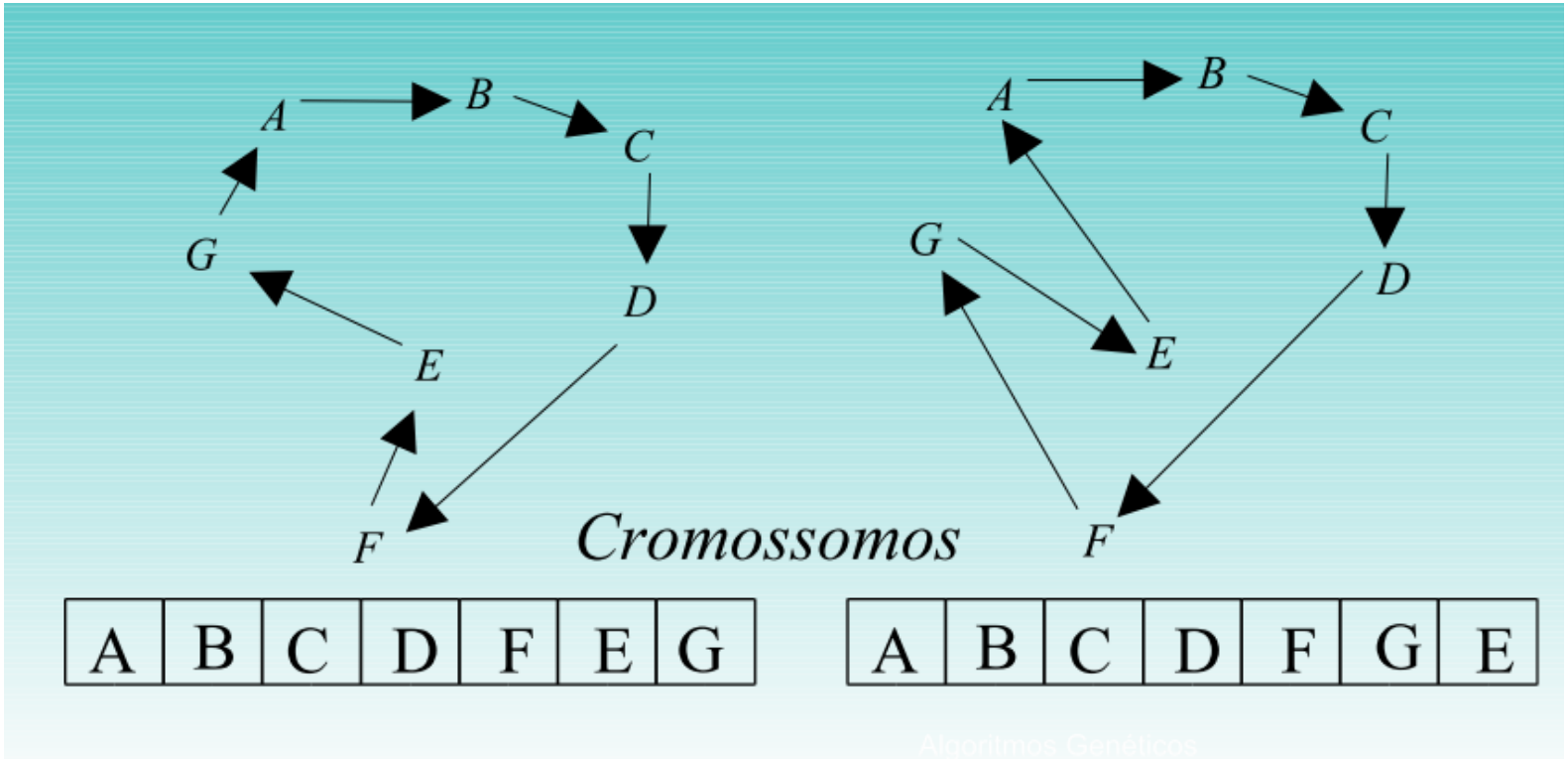
Caixeiro Viajante

# O Problema

- Dado um número de cidades, encontrar o caminho mais curto passando por todas as cidades uma única vez
  - Função Objetivo = Distância Total Percorrida

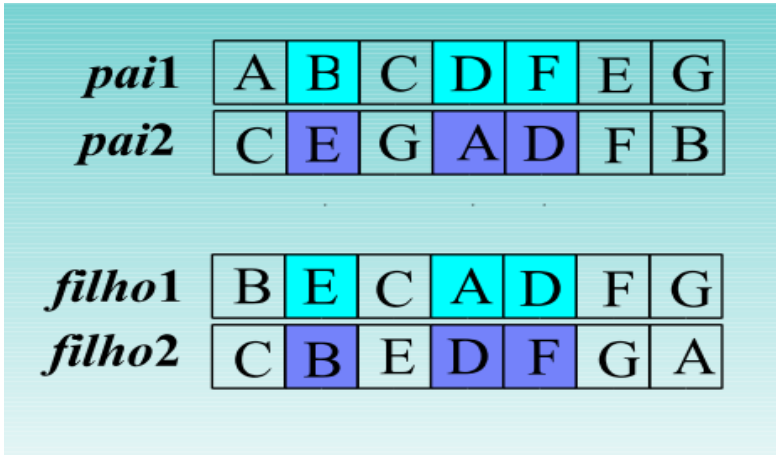


# Representação



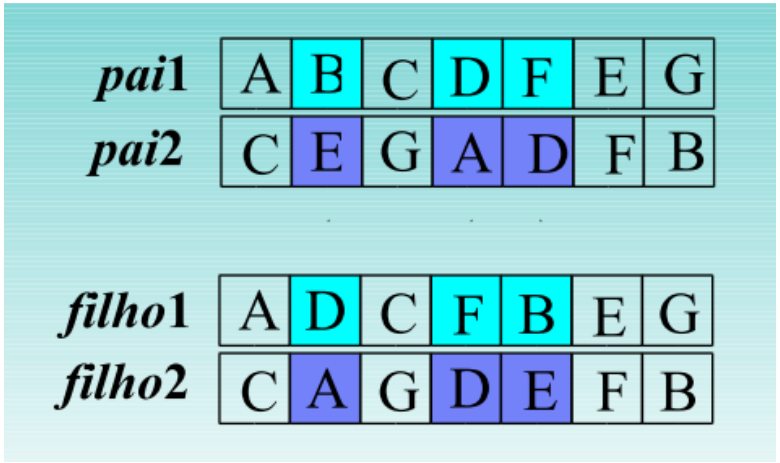
# Crossover

- Crossover baseado em posição
  - São selecionadas *n* cidades. Cada filho mantém a posição das cidades selecionadas de um pai



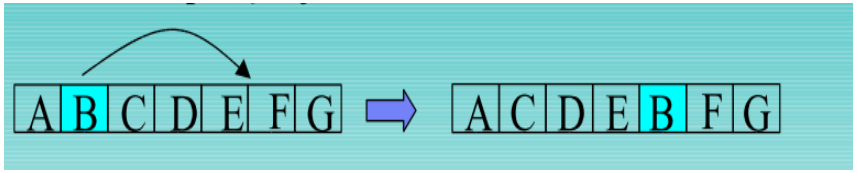
# Crossover

- Crossover baseado em ordem
  - São selecionadas *n* cidades. Cada filho herda a ordem das cidades selecionadas de um pai

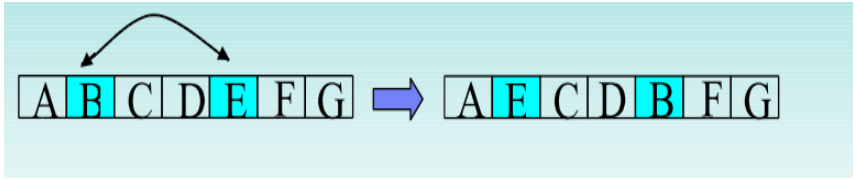


# Mutação

- Mutação baseada na troca de posição de uma cidade




- Mutação baseada na troca da ordem de duas cidades



# Algoritmos Genéticos (revisão do algoritmo)

Seja $S(t)$ a população de cromossomos na geração $t$.

$t \leftarrow 0$
inicializar $S(t)$
avaliar $S(t)$
**enquanto** o critério de parada não for satisfeito **faça**
    $t \leftarrow t+1$
    selecionar $S(t)$ a partir de $S(t-1)$
    aplicar *crossover* sobre $S(t)$
    aplicar mutação sobre $S(t)$
    avaliar $S(t)$
**fim enquanto**

# Algoritmos Genéticos – Referência Básica da Aula

- Estefane Lacerda – Introdução aos Algoritmos Genéticos. Em *Sistemas Inteligentes – Aplicações a Recursos Hídricos e Ciências Ambientais*, 1999
  - http://www.dca.ufrn.br/~estefane/metaheuristicas/index.html