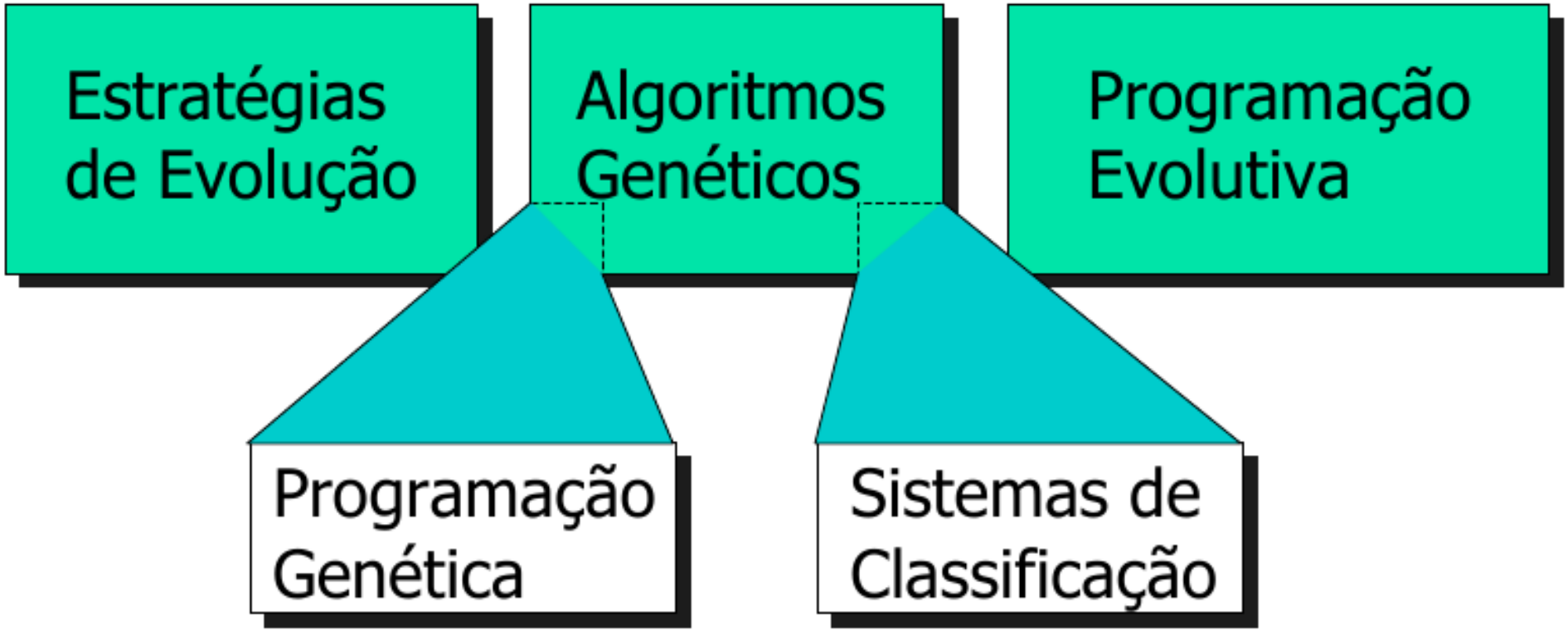
Aprendizado de Máquina: Computação Evolutiva

Prof. Arnaldo Candido Junior
UTFPR – Medianeira

Introdução

- Computação Evolutiva
  - Técnicas computacionais para resolução de problemas baseados em:
    - Genética
    - Teoria da evolução natural
  - Pesquisas tiveram início na década de 50
    - Independentemente a partir de cerca de 10 grupos em um período de 15 anos

Algoritmos Evolutivos



Algoritmos Evolutivos

- Aspectos de cada área estão sendo assimilados por outras. Difícil definir fronteiras, tudo é computação evolutiva
- Programação Evolutiva (Pes): utilizada principalmente para otimização de funções contínuas, sem usar crossover
- Estratégias Evolutivas (Ees): utilizadas principalmente para otimização de funções contínuas, utiliza crossover

Algoritmos Evolutivos (2)

- Algoritmos Genéticos (Ags): usados em geral para problemas de otimização combinatória
- Programação Genética (PG): evolui programas
- Sistemas de classificação: evolui regras de classificação

Algoritmos Genéticos (AGs)

- Propostos para resolver problemas de busca e otimização
- Baseados na genética e teoria da seleção natural
- Depois de várias gerações populações naturais evoluem de acordo com os princípios de seleção natural e sobrevivência dos mais aptos

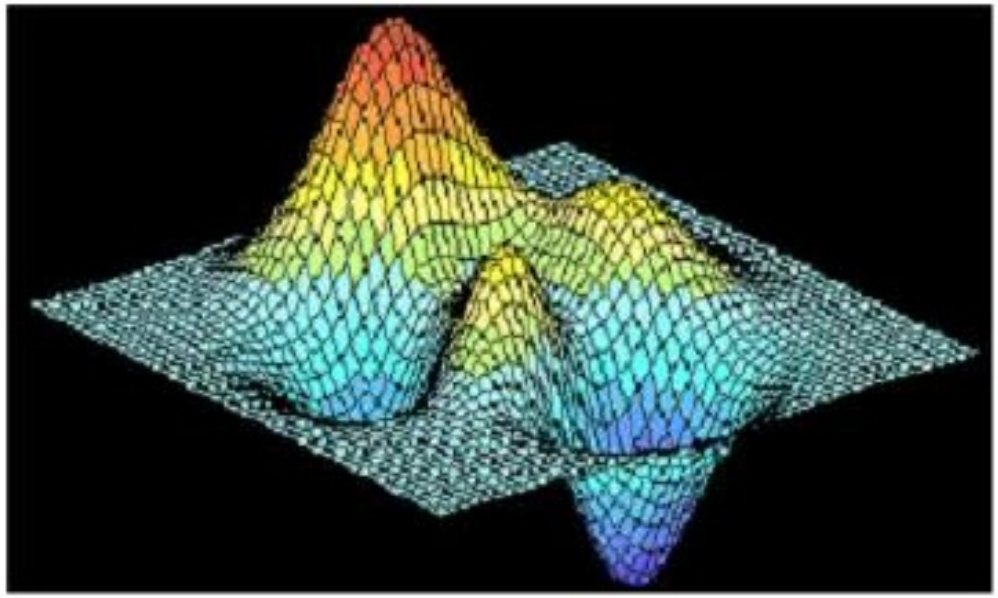
Algoritmos Genéticos (2)

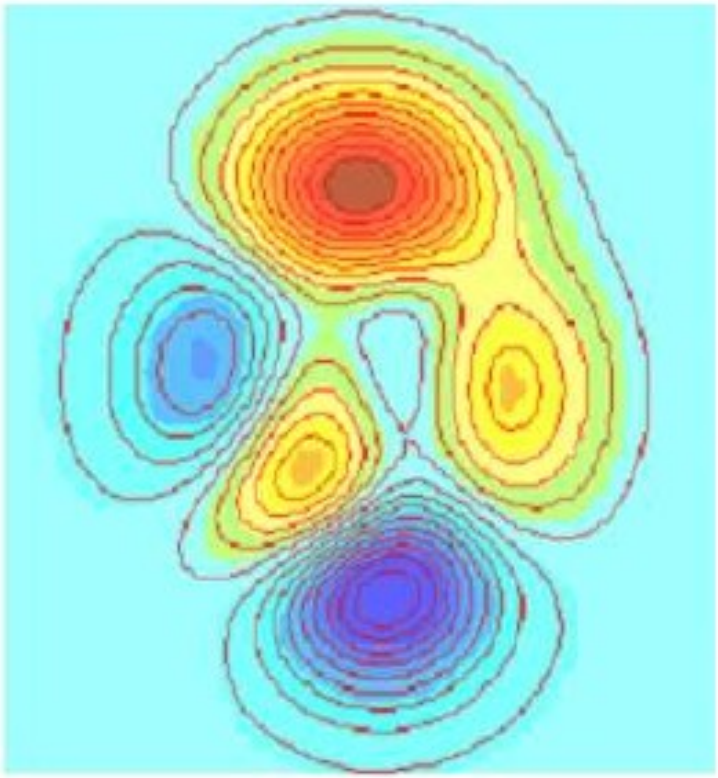
- Desenvolvido por John Holland
  - Popularizado por David Goldberg
- Objetivos:
  - Abstrair e explicar rigorosamente os processos adaptativos dos sistemas naturais
  - Desenvolver sistemas artificiais que conservam mecanismos importantes dos sistemas naturais

Algoritmos Genéticos (3)

- Utilizam uma população de soluções candidatas (indivíduos)
- Otimização ocorre em várias gerações. A cada geração
  - Mecanismos de seleção selecionam os indivíduos mais aptos
  - Operadores de reprodução geram novos indivíduos

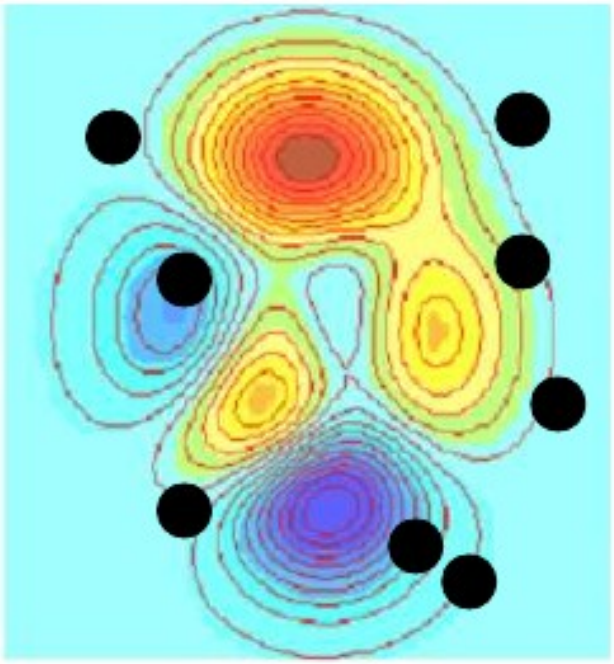
Processo Evolutivo



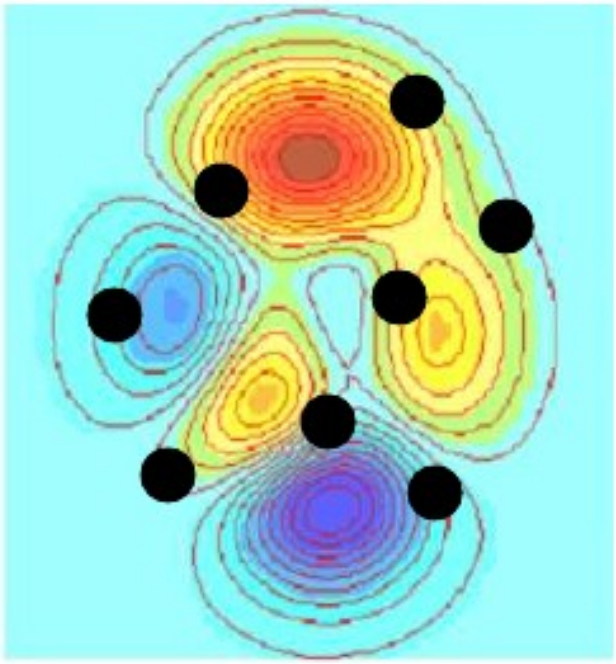


www.cs.bham.ac.uk/~axk

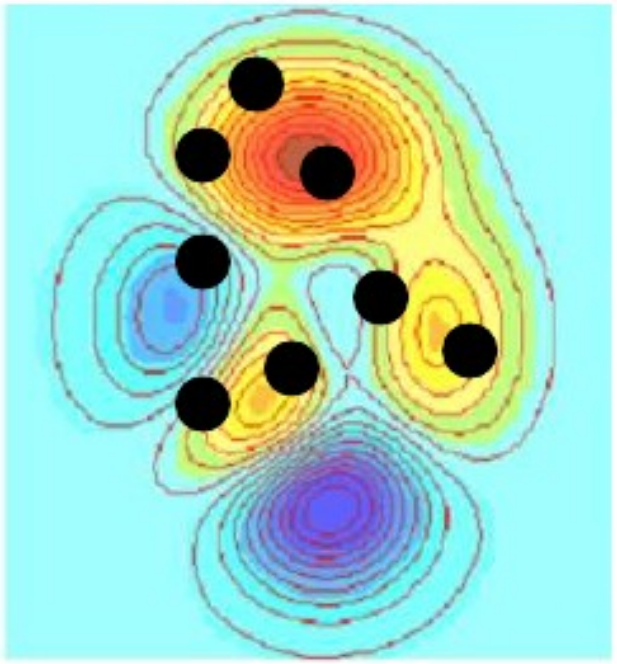
Processo Evolutivo (2)



Geração 1



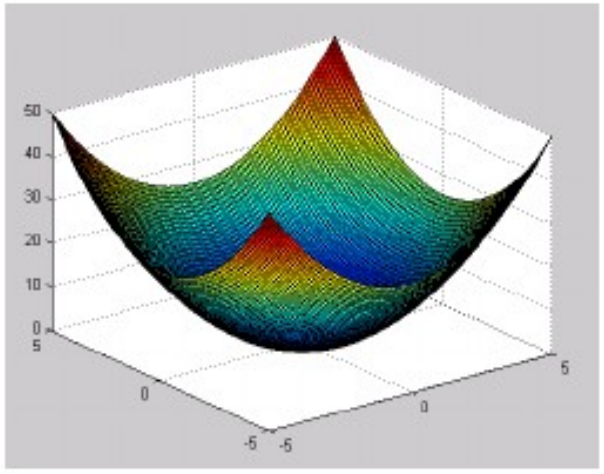
Geração 10

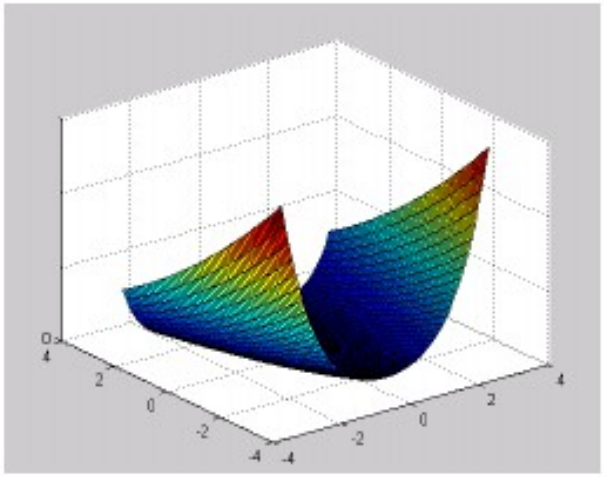


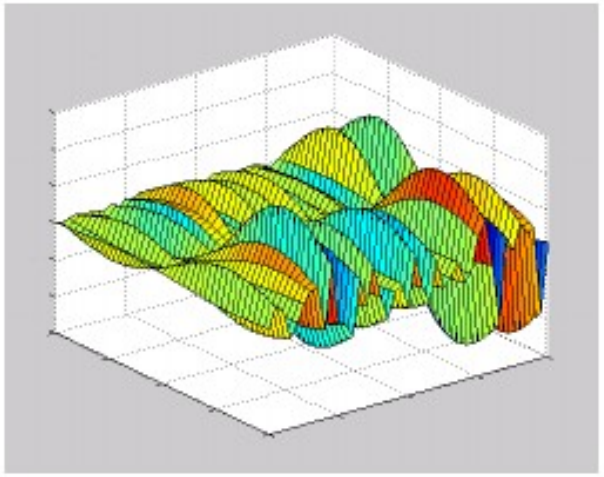
Geração 20

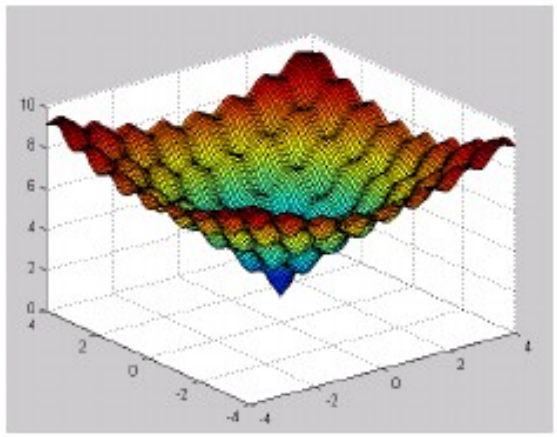
www.cs.bham.ac.uk/~axk

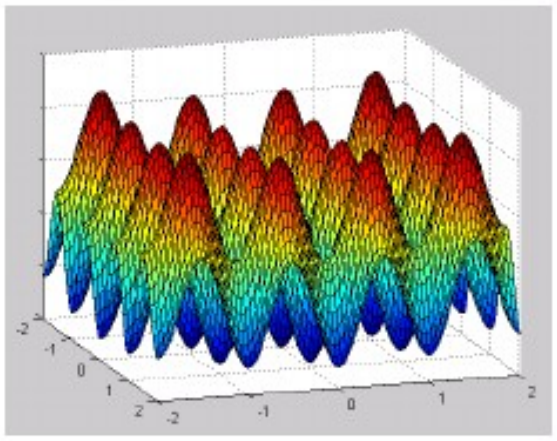
Espaço de Busca



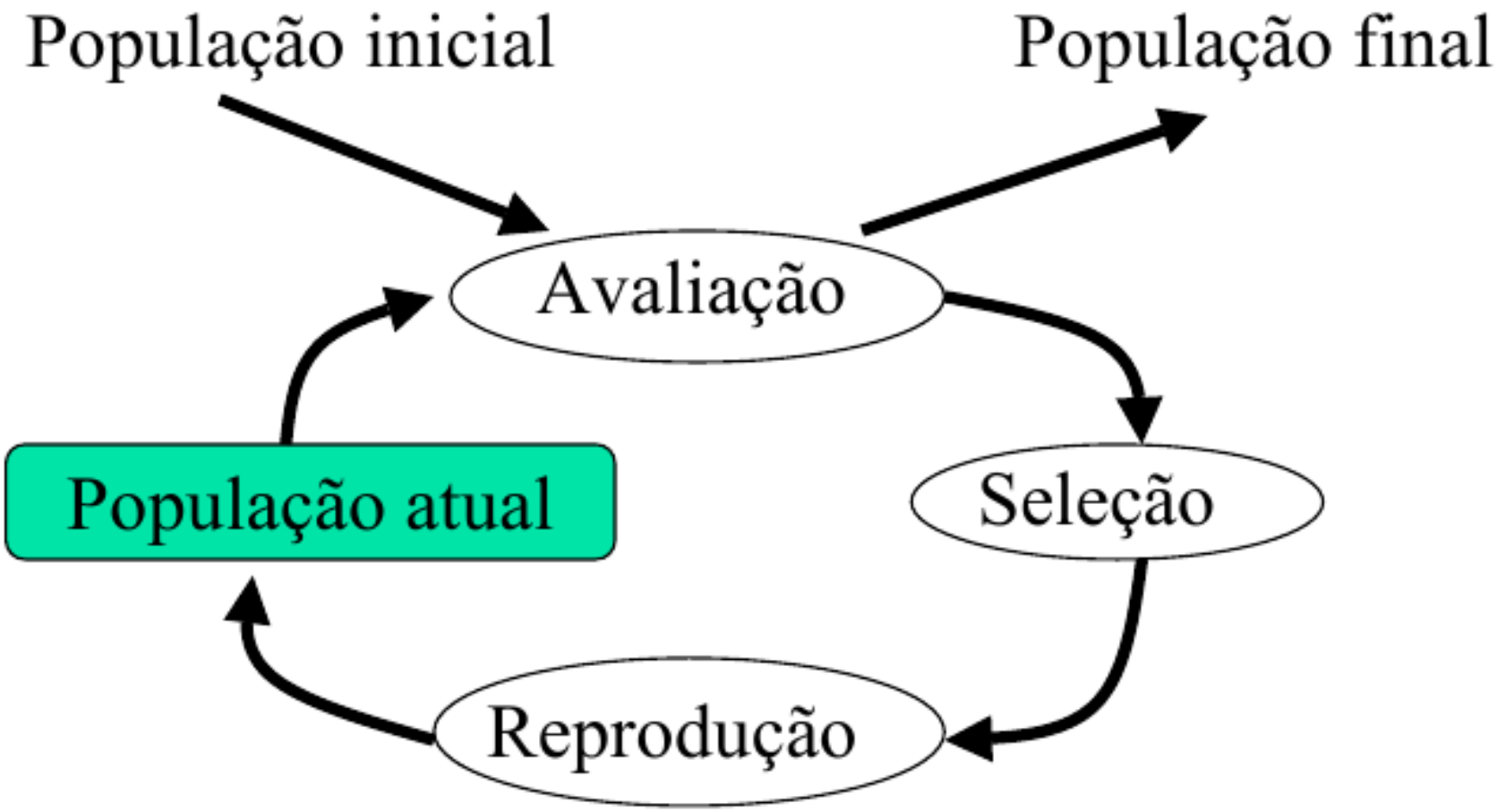








Algoritmos Genéticos



Algoritmos Genéticos (2)

- Podem “evoluir” soluções para problemas do mundo real
  - Problemas devem ser adequadamente codificados
  - Deve haver uma forma de avaliar as soluções apresentadas

Algoritmos Genéticos (3)

- Cada indivíduo representa uma possível solução para um dado problema
  - É associado a um escore de aptidão que mede qualidade da solução que ele representa
- Indivíduos mais aptos têm mais oportunidades de se reproduzir, produzindo descendentes cada vez mais aptos

Princípios Básicos

- Codificação
- Indivíduo / Cromossomo
- População / Geração
- Reprodução, Mutação
- Seleção / Função de aptidão

Indivíduo

- Possível solução para um dado problema
  - Também chamado de cromossomo, string ou genótipo
- Conjunto de indivíduos forma uma população
- Codificado como um vetor de genes ou características
  - Cada possível valor (alelo) representa um parâmetro

Codificação

- De acordo com a codificação adotada, genes podem assumir valores:
  - Binários
  - Inteiros
  - Reais (ponto flutuante)

Codificação (2)

- Genótipo: contém informação necessária para construir um organismo (fenótipo)
- Fenótipo: produto da iteração de todos os genes
  - Aptidão de um indivíduo depende do desempenho de seu fenótipo
  - Inferido do genótipo usando função de aptidão

Codificação (3)

- Tradicionalmente, os indivíduos são representados por vetores binários
  - 0, 1 (ausência, presença)
  - Universal e independente do problema
  - Permite a codificação de qualquer problema
  - Permite a utilização dos operadores de reprodução padrão

Codificação (4)

- Genes também podem assumir valores inteiros, reais ou de tipos abstratos
- Representações em níveis abstratos mais altos
  - Facilitam sua utilização em determinados domínios mais complexos
  - Necessitam de operadores específicos

Codificação (5)

- Para alguns problemas, genes assumem valores inteiros ou reais
  - Operadores específicos podem ser utilizados para esses valores
  - Ou valores podem ser convertidos para valores binários

Codificação Binária de Inteiros

- Representar um valor $g \in I$ por um vetor de binários $b_1, b_2, ..., b_N$
- Definir função $f[g] \rightarrow \{0,1\}^n$ para valor de g restrito ao intervalo $[g_{min}, ..., g_{max}]$
  - Valor de g varia entre 0 a 2^n-1
  - Valor de g varia entre m até $m+2^n-1$
  - Valor de g varia entre 0 a k, onde $k < 2^n-1$ (ex.: aplicar log)

Codificação Binária de Inteiros (2)

- Valor de g varia entre 0 a 2^n-1: converter valor de g para valor binário correspondente.
- Valor de g varia entre m até m+2^n-1: converter valor de g para valor binário correspondente
- Valor de g varia entre 0 a k-1, onde k < 2^n-1. Últimos valores referem-se a k (exemplo a seguir).

Codificação Binária de Inteiros (4)

- Corte: $n = \lfloor \log(K) \rfloor + 1$
  - Codificar todos os valores até K-2 com a codificação binária convencional
  - Usar para os demais valores g = K-1
  - Características:
    - Fácil de implementar
    - Forte *bias* para os últimos valores
- Escala

Exemplo

- $g \in \{0, 1, 2, ..., 5\}$
- $k = 6$
- $n = 3$
- $g = \text{decimal}(x)$ se $x \leq k-1$
- $g = \text{decimal}(k-1)$ se $x > k$

Valor de g	Código (x)
0	000
1	001
2	010
3	011
4	100
5	101
5	110
5	111

Código Cinza

- Códigos para dígitos consecutivos diferem em apenas um bit
- Frequentemente utilizado para transformar valores analógicos para decimal
  - Pequena mudança no valor analógico muda apenas um bit no valor decimal

Código Cinza (2)

- Existem vários códigos cinza
  - Não é único
- Um código cinza para 3 bits:
  - 000, 010, 011, 001, 101, 111, 110, 100
- Um código cinza para 2 bits:
  - 00, 01, 11, 10

Dígito	Binário	Código cinza
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Código Cinza (3)

- Algoritmo simples

1. Começa com todos os bits iguais a zero
2. Para cada novo número
   Mudar o valor do bit mais a direita que gerar uma nova sequência de bits

Código Termômetro

- Valores consecutivos diferem em apenas um bit
- Utiliza muitos dígitos binários
  - Tamanho cresce linearmente com número de valores

Dígito	Binário	Código termômetro
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0111
4	0100	1111

Função de Aptidão

- Mede o grau de aptidão de um indivíduo
  - Retorna um valor (índice) de aptidão numérico
    - Proporcional a utilidade ou habilidade do indivíduo
  - Aptidão = probabilidade do indivíduo sobreviver para a próxima geração
    - Cada aplicação tem sua própria função de aptidão

Função de Aptidão (2)

- É aplicada ao fenótipo do indivíduo
  - Genótipo precisa ser decodificado, recuperando o fenótipo associado
- Pode envolver uma (otimização de função) ou mais medidas (otimização multi-objetivo)
  - Ex. projeto de ponte
    - Custo, tempo de construção, capacidade máxima

Função de Aptidão (3)

- Métodos para calcular valor ou índice de aptidão
  - Padrão
  - Baseada em ranking
  - Ranking-espaço
    - Estimula diversidade

Função de Aptidão Padrão

- Utiliza apenas informação sobre “qualidade do cromossomo”

$$f_i = \frac{q_i}{\sum_j q_j}$$

q = índice de aptidão do cromossomo

Seleção

- Ocorre em duas etapas
  - Escolha de indivíduos para participar da reprodução
  - Escolha dos indivíduos para sobrevivem para a próxima geração

Seleção (2)

- Escolhe preferencialmente indivíduos com maiores notas de aptidão
  - Não exclusivamente
  - Indivíduos mais aptos têm mais oportunidades de gerar descendentes
    - Que serão cada vez mais aptos

Seleção (3)

- Existem vários método de seleção
  - Uniforme
    - Geralmente usada para reprodução
  - Por roleta
  - Por torneio
  - Amostragem Universal Estocástica

Seleção pela Roleta

- Método simples mais utilizado
- Escolhe indivíduos por meio de um sorteio
  - Cada indivíduo ocupa uma fatia proporcional à sua aptidão

Seleção por Torneio

- Escolhe aleatoriamente *n* indivíduos da população
  - Cromossomo com maior aptidão é selecionado
    - Processo se repete até que a população intermediária seja preenchida
  - Quanto maior o valor de *n*, maior a pressão seletiva
    - Indivíduo tem que ser melhor que uma quantidade maior de competidores

Pressão Seletiva

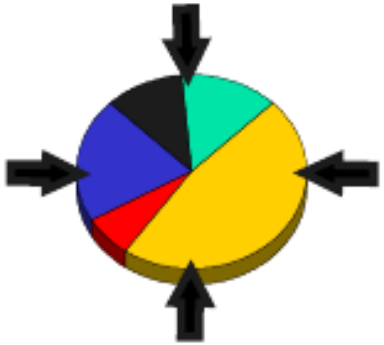
- Grau com que os melhores indivíduos são favorecidos
  - Influencia taxa de convergência do AG
    - Pressão muito baixa
      - Taxa de convergência lenta
      - Demora para encontrar boa solução
    - Pressão muito elevada
      - Convergência prematura

Diversidade

- Deve haver equilíbrio entre pressão seletiva e diversidade
- Formas de prevenir convergência prematura
  - Controlar número de oportunidades de reprodução de cada indivíduo
- Formas de promover diversidade
  - Aumento do tamanho da população
  - Aumento da taxa de mutação

Seleção por SUS

- SUS (*Stochastic Universal Sampling*)
  - Amostragem Universal Estocástica
- Variação do método da roleta



  - Ao invés de *n* vezes, a roleta é girada uma única vez e *n* indivíduos são selecionados
    - Utiliza n agulhas igualmente espaçadas ao invés de 1
  - Exibe menos variância que repetidas rodadas da roleta

Seleção por Estado Estável

- Maioria dos AGs é geracional
  - A cada geração, os indivíduos da população são formados apenas pelos filhos
  - Estado estável (*steady state*) reduz o número de substituições de pais pelos filhos
    - Existe uma intersecção entre conjuntos de indivíduos de gerações sucessivas
    - Ex.: escolher os N melhores entre os N pais e N filhos

Operadores Genéticos

- Permitem obtenção de novos indivíduos
  - Cada geração possui, geralmente, indivíduos mais aptos
  - Principais operadores genéticos
    - Crossover (cruzamento ou recombinação)
    - Mutação
    - Elitismo

Crossover

- Filhos herdam partes das características dos pais durante a reprodução
  - Permite que as próximas gerações herdem estas características
- Funcionamento
  - Escolhe dois indivíduos e troca trechos entre eles

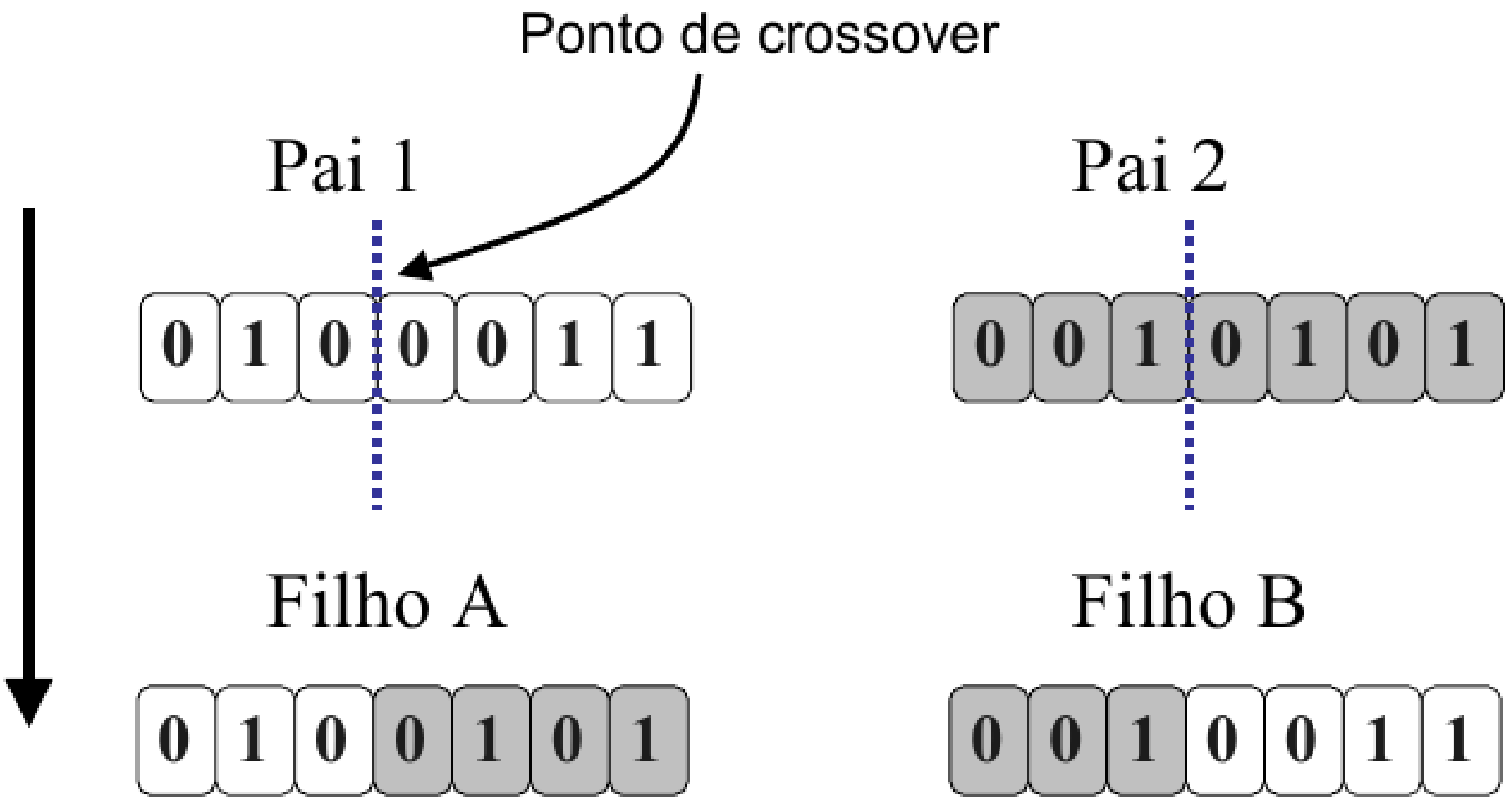
Crossover (2)

- É o operador genético predominante
  - A taxa de *crossover* deve ser maior que a taxa de mutação
  - Taxa de *crossover*: $0.6 < P_c < 1.0$
  - Caso *crossover* não seja aplicado, descendentes seriam iguais aos pais
- É a operação mais importante para exploração rápida do espaço de busca

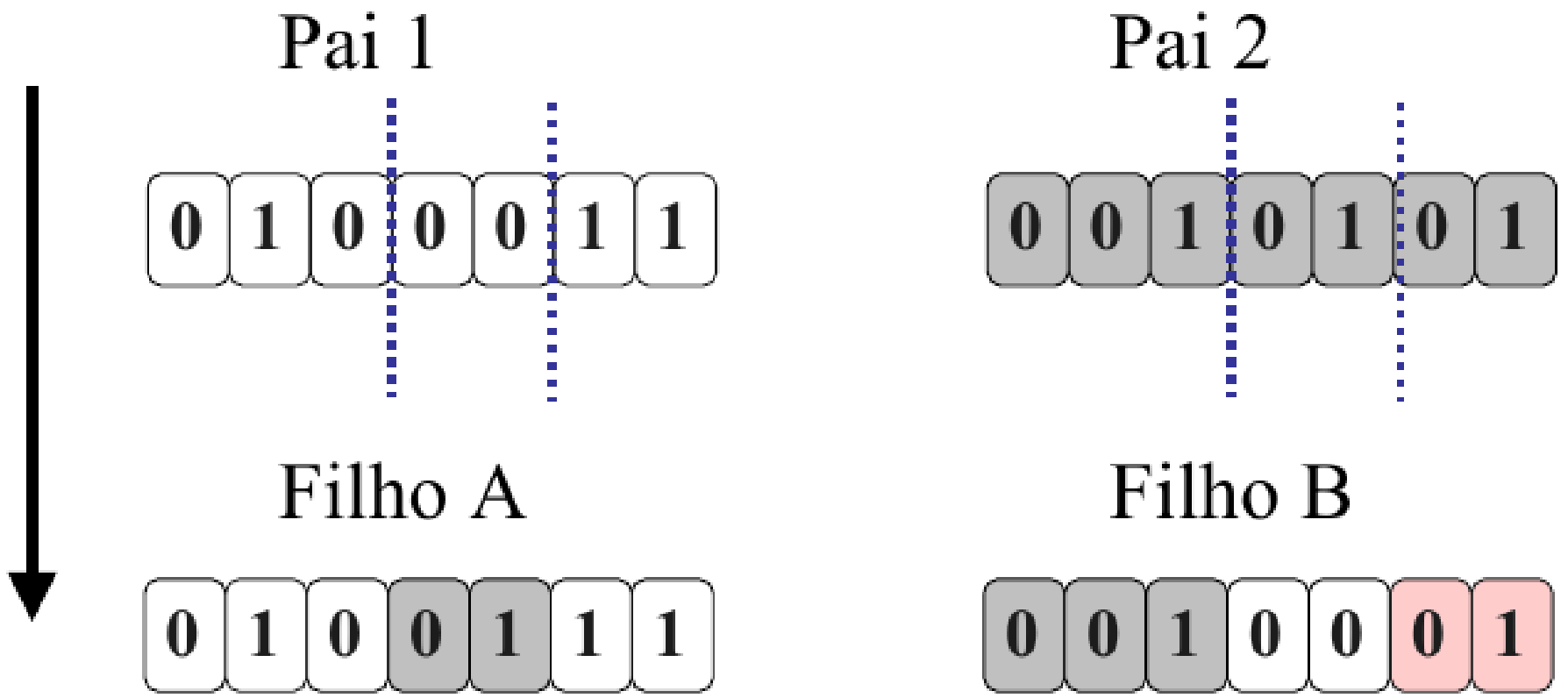
Crossover (3)

- Diversas variações
  - Um ponto
    - Mais comum
  - Dois pontos
  - Multi-pontos
  - Uniforme

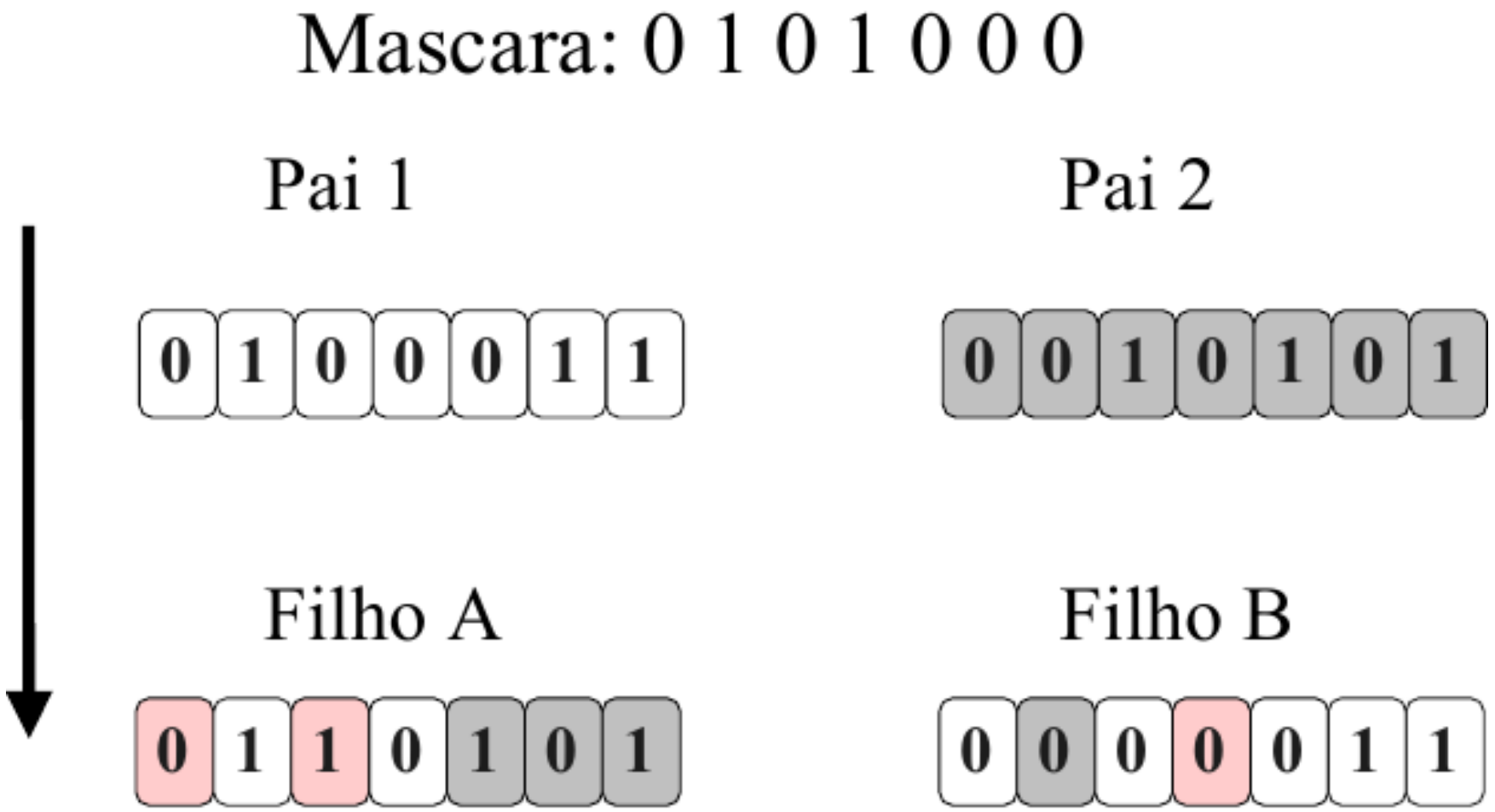
Crossover 1 Ponto



Crossover 2 Pontos



Crossover Uniforme



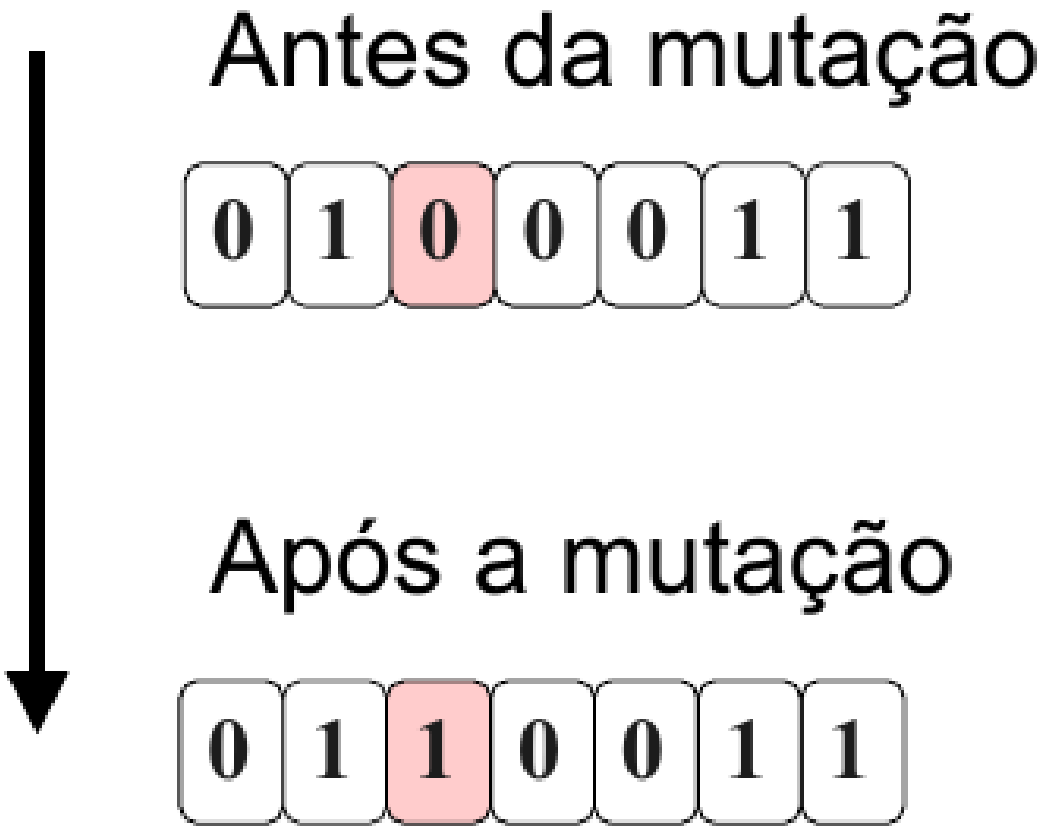
Mutação

- Permite introdução e manutenção da diversidade genética
  - Aplicado a cada indivíduo após crossover
- Funcionamento
  - Altera aleatoriamente um ou mais componentes de uma estrutura escolhida

Mutação (2)

- Probabilidade de atingir qualquer ponto do espaço de busca nunca será zero
  - Reduz chance de parada em Mínimos Locais
- Operador genético secundário
  - Taxa de mutação pequena $P_m \cong 0.001$

Mutação (3)



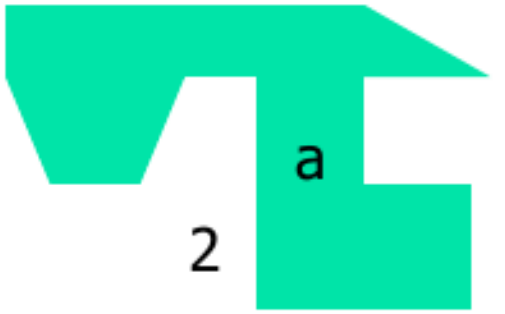
Elitismo

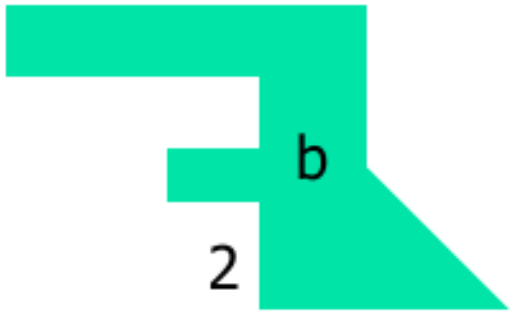
- Indivíduo de maior desempenho é automaticamente selecionado
- Evita modificações deste indivíduo pelos operadores genéticos
  - Utilizado para que os melhores indivíduos não desapareçam da população

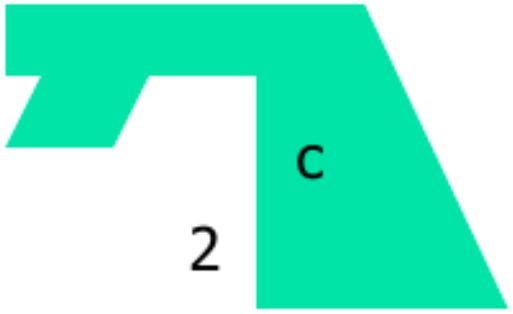
Exemplo

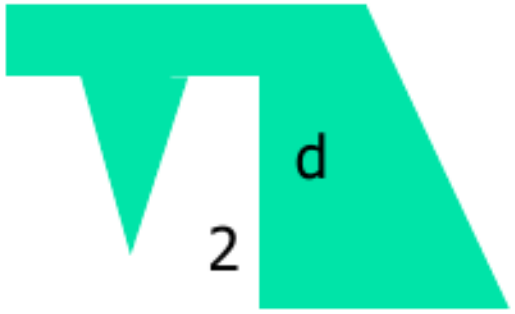
Encontrar a chave para a fechadura:

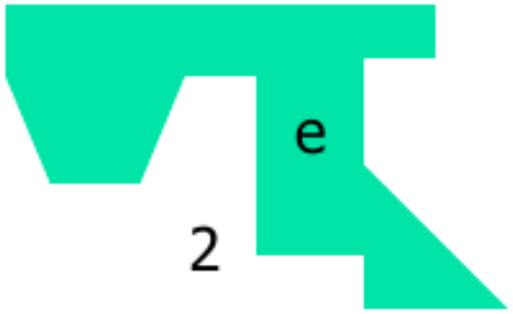
Quanto menor o valor de Aptidão, melhor a chave

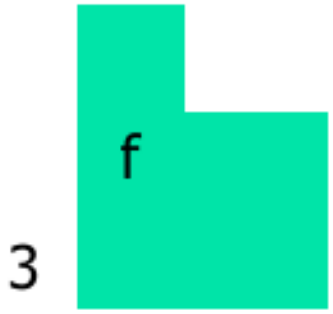






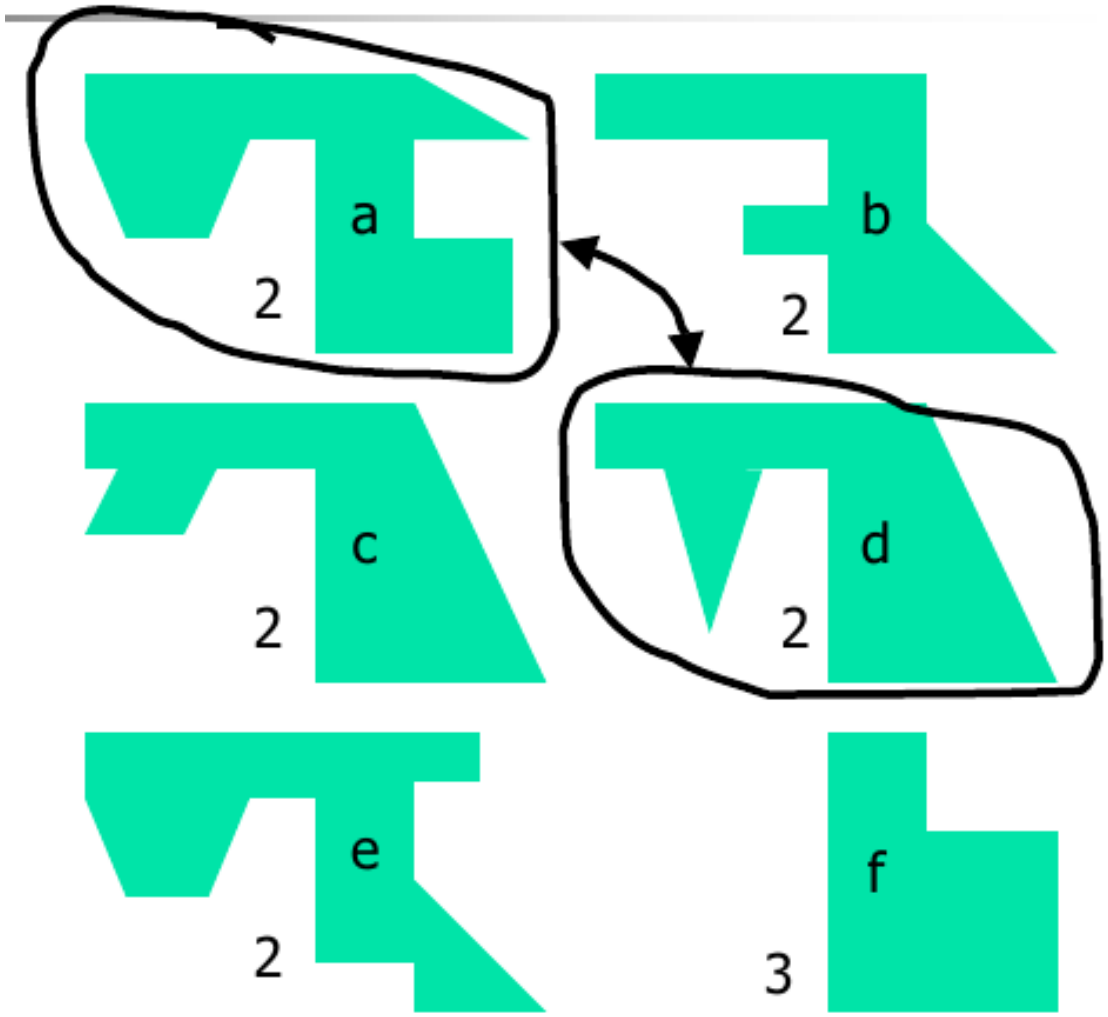






Exemplo (2)

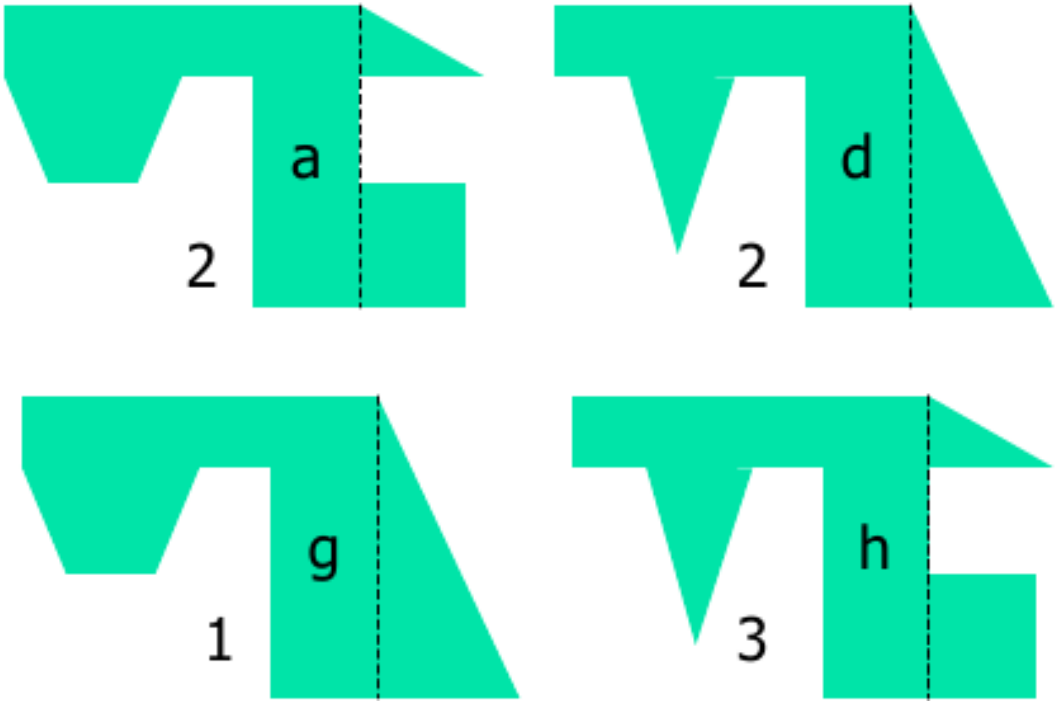
Encontrar a chave para o chaveiro:



Exemplo (3)

Encontrar a chave para o chaveiro:

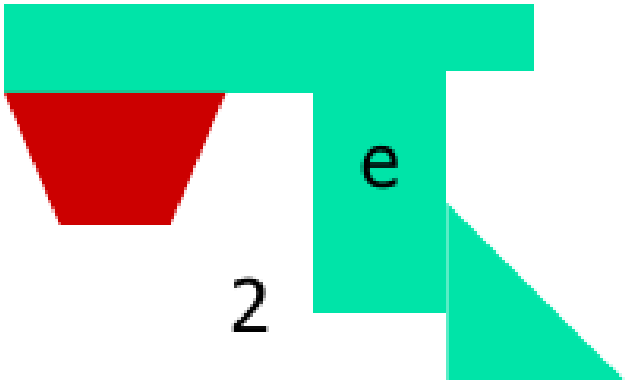
Cruzamento

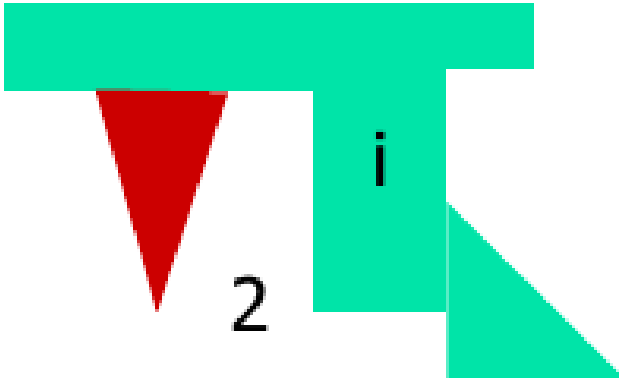


Exemplo (4)

Encontrar a chave para o chaveiro:

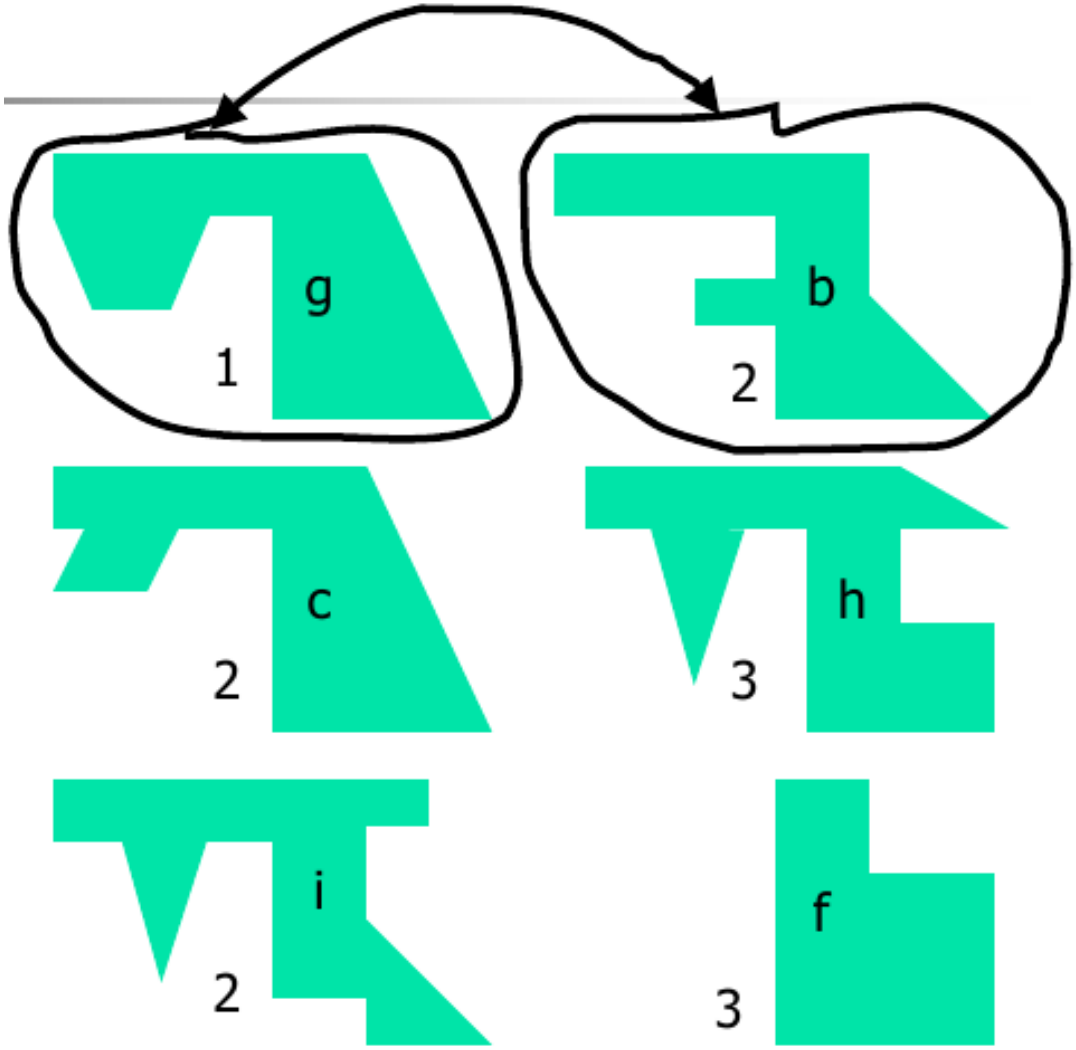
Mutação





Exemplo (5)

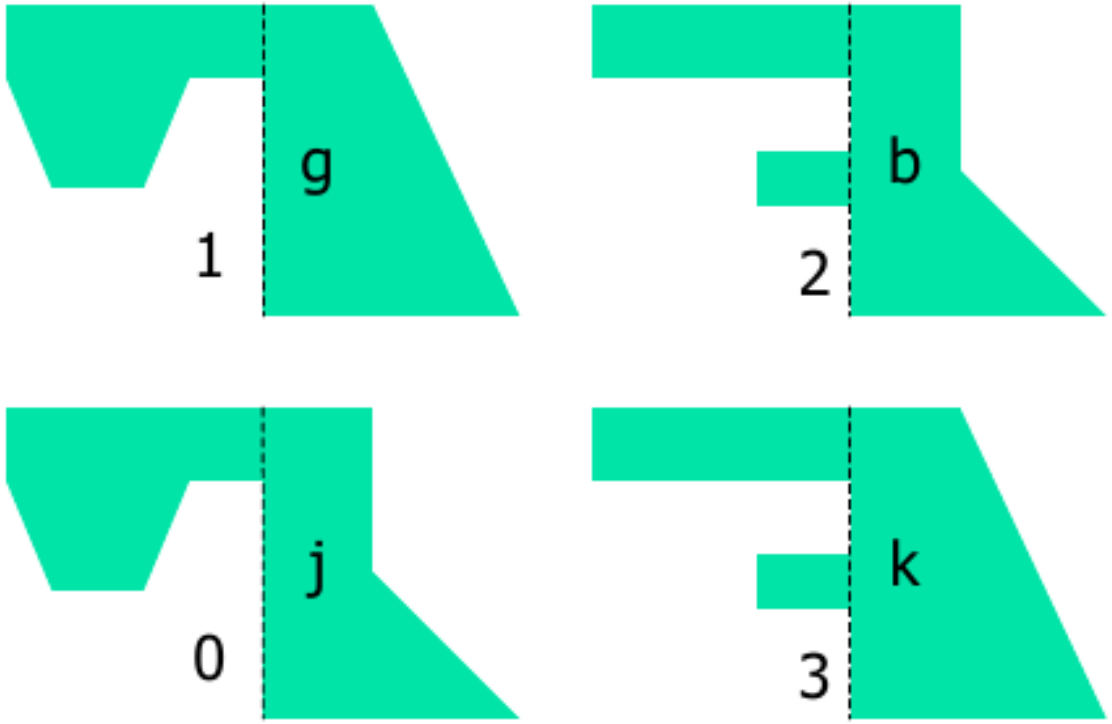
Encontrar a chave para o chaveiro:



Exemplo (6)

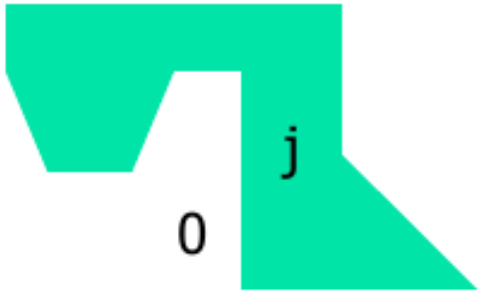
Encontrar a chave para o chaveiro:

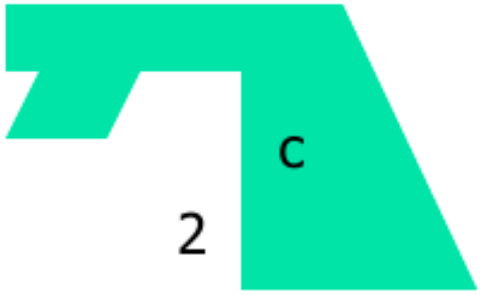
Cruzamento

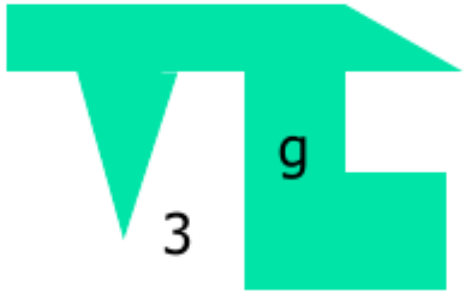


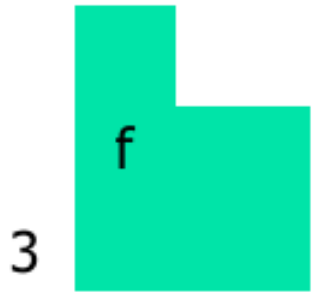
Exemplo (7)

Encontrar a chave para o chaveiro:





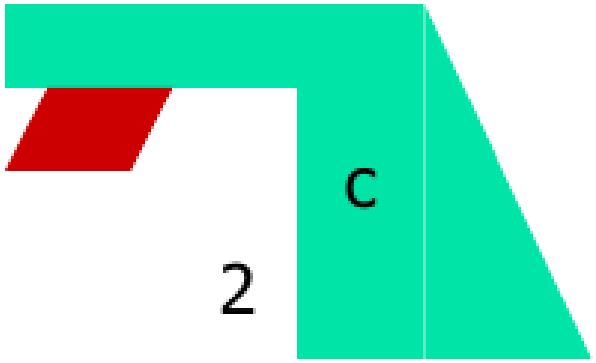


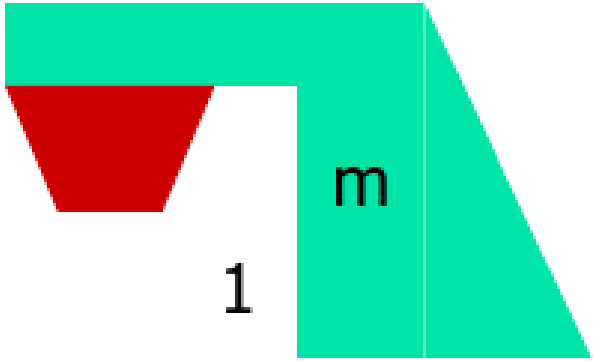


Exemplo (8)

Encontrar a chave para o chaveiro:

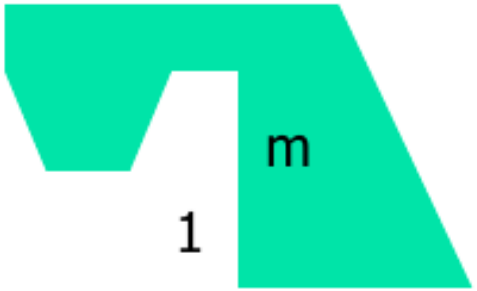
Mutação

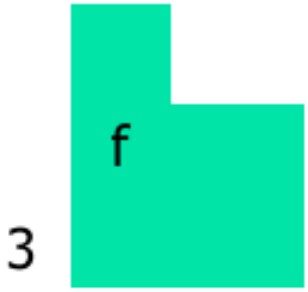




Exemplo (9)

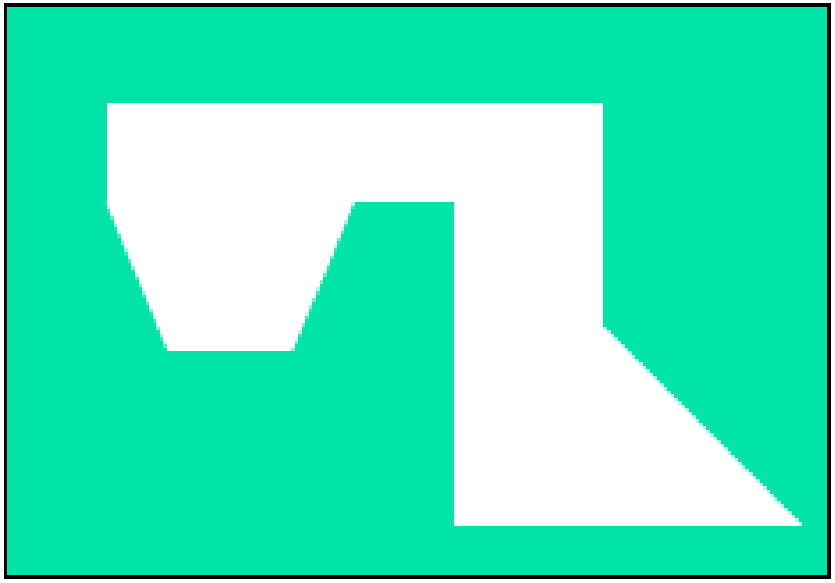
Encontrar a chave para o chaveiro:





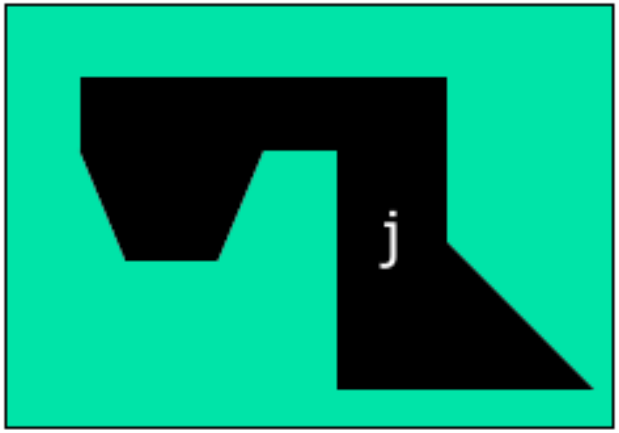
Exemplo (10)

Encontrar a chave para
o chaveiro:



Exemplo (11)

Encontrar a chave para o chaveiro:



Um Algoritmo Genético

1. Escolher população inicial de cromossomos
2. Avaliar cada cromossomo da população
3. Enquanto critério de parada não for atingido
   - 3.1 Selecionar indivíduos mais aptos
   - 3.2 Criar novos cromossomos aplicando operadores genéticos
   - 3.3 Avaliar cada cromossomo da população

Exemplo

- Utilizando Algoritmos Genéticos, achar o máximo da função :

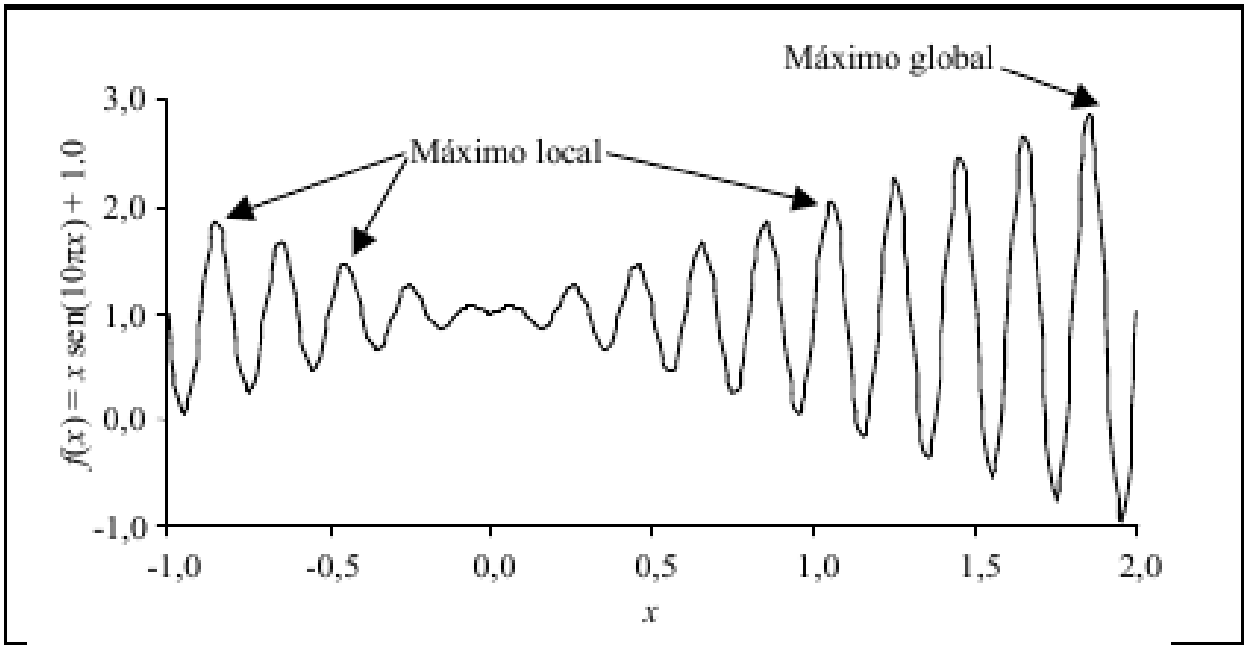
  $f(x) = x \text{ seno}(10 \pi x) + 1,0$

- Restrita no intervalo:

  $-1,0 \leq x \leq 2,0$

Exemplo (2)

- Gráfico da função f(x) = x seno(10πx) + 1.0



Máximo global:
x = 1,85055
f(x) = 2,85027

Pode ser observado que existem vários pontos de máximo

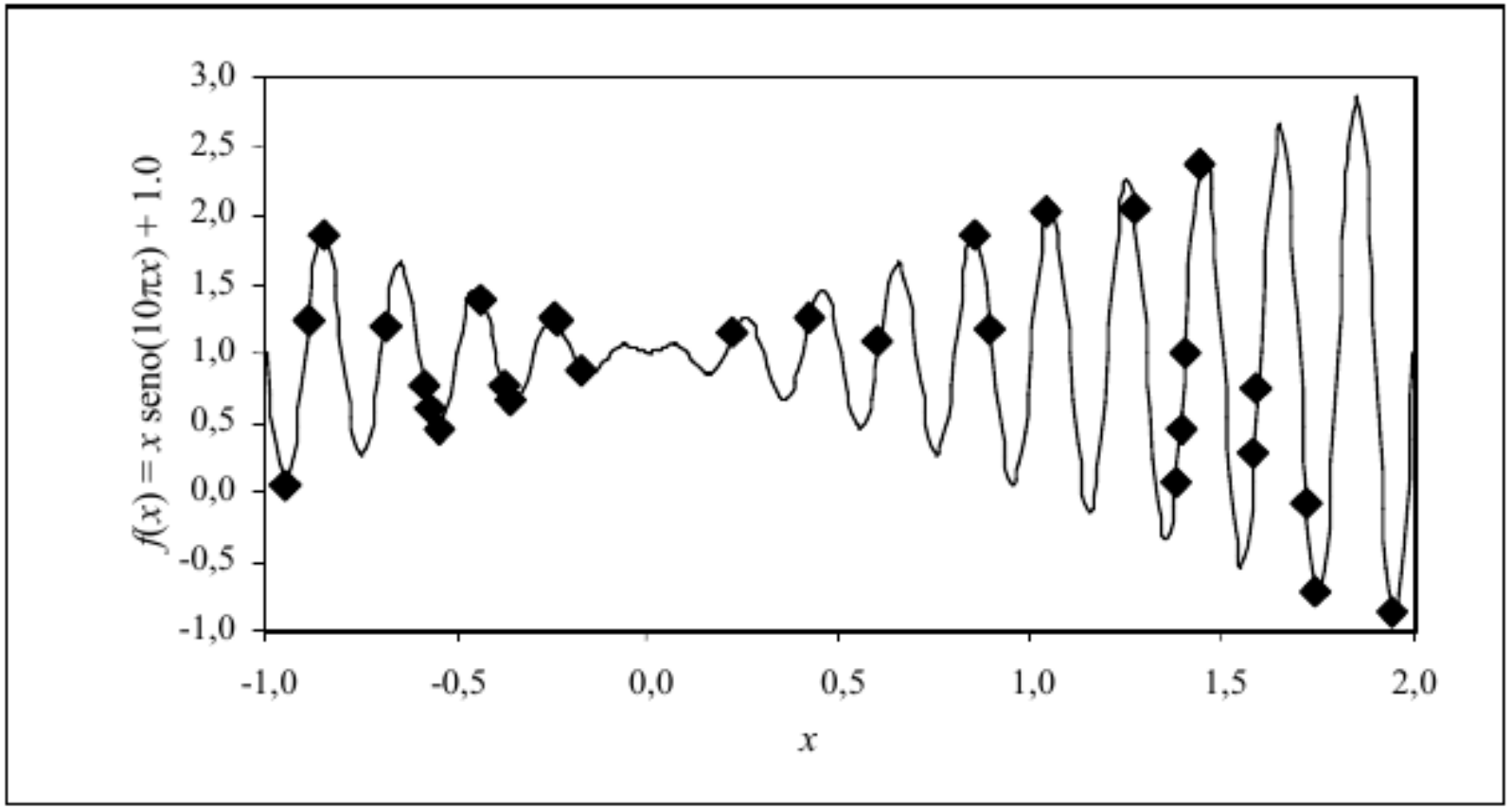
Representação dos Cromossomos

- Representar o único parâmetro deste problema (a variável x) na forma de um cromossomo
  - Binário = 10001011101101010001111
  - Decimal = $(10001011101101010001111)_2$ = 2288967
  - Valor de x precisa estar no intervalo [-1,0; 2,0]

Geração da População Inicial

Rank i	Cromossomo $\mathbf{s}_i$	x_i	Função objetivo $f(x_i)$	Aptidão f_i	Aptidão acumulada $\sum_{k=1}^{i} f_k$
1	11010000000111110110111	1,43891	2,35251	2,00000	2,00000
2	11000001101001000111111	1,26925	2,04416	1,93103	3,93103
3	10101110010101100100000	1,04301	2,01797	1,86207	5,79310
4	10011110000110010000101	0,85271	1,84962	1,79310	7,58621
5	10011101101110000011100	0,84829	1,84706	1,72414	9,31035
6	00001100111111010010110	-0,84792	1,84610	1,65517	10,96552
7	00110000000100111010010	-0,43570	1,39248	1,58621	12,55172
8	01111001010000011011100	0,42098	1,25777	1,51724	14,06897
9	01000000000110011101000	-0,24764	1,24695	1,44828	15,51724
10	01000000100011111011110	-0,24343	1,23827	1,37931	16,89655
11	00001001010000000111010	-0,89156	1,23364	1,31035	18,20690
12	00011010011000101011111	-0,69079	1,19704	1,24138	19,44828
13	10100001100110000011011	0,89370	1,17582	1,17241	20,62069
14	01101000010110110000100	0,22292	1,14699	1,10345	21,72414
15	10001000111100001000011	0,60479	1,09057	1,03448	22,75862
16	11001100110010100011110	1,39988	0,99483	0,96552	23,72414
17	01000110010000100011101	-0,17655	0,88140	0,89655	24,62069
18	00110100111110100101000	-0,37943	0,77149	0,82759	25,44828
19	00100011010011001011100	-0,58633	0,75592	0,75862	26,20690
20	11011101011011111111111	1,59497	0,74904	0,68966	26,89655
21	00110110110011011110110	-0,35777	0,65283	0,62069	27,51724
22	00100100010011111100111	-0,57448	0,58721	0,55172	28,06897
23	11001011101100111111000	1,38714	0,45474	0,48276	28,55172
24	00100110011001101001111	-0,54999	0,45001	0,41379	28,96552
25	11011100100101001000001	1,58492	0,27710	0,34483	29,31035
26	11001010110001110100011	1,37631	0,06770	0,27586	29,58621
27	00000100001001001100001	-0,95144	0,04953	0,20690	29,79310
28	11101000010000000010001	1,72169	-0,08458	0,13793	29,93103
29	11101010001111000000000	1,74494	-0,72289	0,06897	30,00000
30	11111011000000001010111	1,94147	-0,87216	0,00000	30,00000

Geração da População Inicial (2)



Exemplo 2

- Problema: encontrar valor inteiro de x que minimiza $f(x) = x^2$, $x \in [-7, +7]$
  - Representação → cada inteiro como um cromossomo de 4 bits
    3 = (0,0,1,1)
    7 = (0,1,1,1)
    -4 = (1,1,0,0)
  - População inicial → aleatória
  - População de tamanho 4
  - Função de aptidão → f(x)

Exemplo 2 (2)

- População inicial gerada:

	x	f(x)	
$A_1 \rightarrow$ 1 1 0 1	-5	25	← Pai 1
$A_2 \rightarrow$ 1 1 1 1	-7	49	
$A_3 \rightarrow$ 0 0 1 0	2	4	← Pai 2
$A_4 \rightarrow$ 0 0 1 1	3	9	

  - Escolha dos pais com maior aptidão pela roleta
  - Gera 2 filhos, que substituirão indivíduos com menor aptidão

Exemplo 2 (3)

		x	f(x)
$Pai_1 = A_1$	→ 1 1 0 1		
$Pai_2 = A_3$	→ **0 0 1 0**	-5	25
$Filho_1$	→ 1 1 **1 0**	2	4
$Mutação_1$	→ 1 1 **0 0**	-4	16
$Filho_2$	→ **0 0** 0 1	1	1

	x	f(x)
A_1 → 1 1 0 0	-4	16
A_2 → 1 1 1 1	-7	49
A_3 → 0 0 0 1	1	1
A_4 → 0 0 1 1	3	9

Exercício

- Encontrar de x para o qual a função $f(x) = x^2 - 4x + 4$ assume o valor mínimo
  - Assumir que x ∈ [-15, +15]
  - Codificar X como vetor binário
    - Usar 5 bits, primeiro bit é o sinal (1-positivo, 0-negativo)
  - Criar uma população inicial com 4 indivíduos
  - Utilizando *crossover* de um ponto e mutação em apenas um gene da população
  - Definir o valor mínimo após no máximo 10 gerações

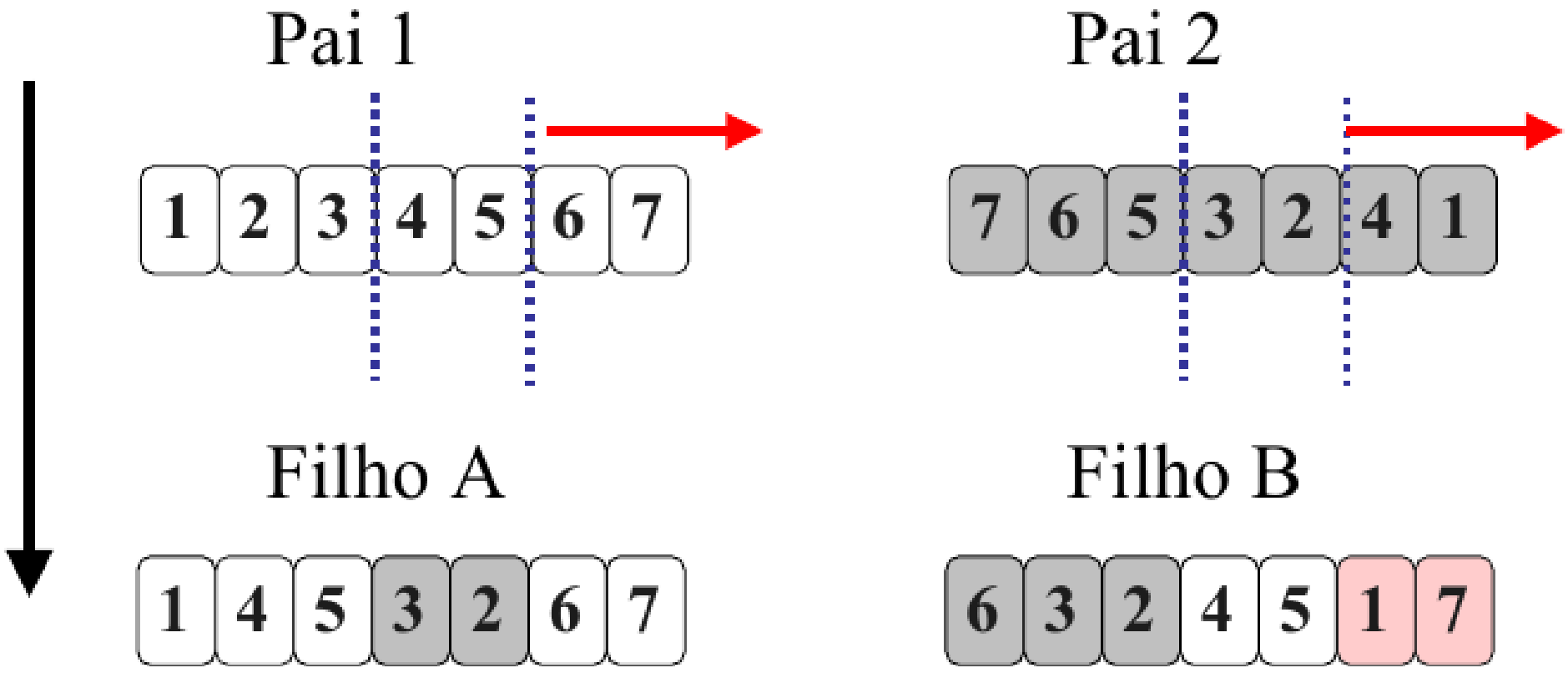
Operadores de Permutação

- Muitos problemas de otimização procuram um ordenamento eficiente de ações ou tarefas
- Exemplo
  - Problema do caixeiro viajante
  - Problemas de agendamento
  - Coloração de grafos
  - Binpacking

Operadores de Permutação (2)

- Crossover
  - De ordem (OX)
  - Baseado em posição (PBX)
  - Baseado em ordem (OBX)
  - Crossover de ciclo (CX)
  - Crossover parcialmente mapeado (PMX)
- Mutação
  - Order-Based Mutation
  - Position-Based Mutation
  - Scramble Mutation

Crossover de Ordem



Representação Real

- Para um ser humano é mais natural do que uma cadeia de bits.
  - Ex. cromossomo (-50, 35)
- Cromossomos mais compactos (menores vetores) e com melhor precisão numérica
- Vários autores têm obtido desempenho melhor com representação real do que com representação binária

Operadores para Representação Real

- Representação Real permite grande variedade de operadores
  - Crossover
    - Convencionais
    - Aritméticos
    - Baseados em gradiente
  - Mutação
    - Randômica
    - Creep

Operadores de Crossover

- Operadores mais sofisticados foram desenvolvidos para:
  - Problemas que utilizam permutações
  - Codificação com valores reais
    - Média aritmética
    - Média ponderada
    - Recombinação geométrica
    - Recombinação uniforme

Operadores de Crossover (2)

- Convencionais
  - n-Pontos, uniforme
  - Não criam novas informações (i.e. novos números reais)
- Aritméticos
  - Realizam operações aritméticas entre os parâmetros
    - Ex. crossover pela média: $filho = (pai_1 + pai_2)/2$
- Baseados em gradiente (usam derivadas)

Crossover Aritmético

- Média aritmética
- Média ponderada
  - $filho = \beta pai_1 + (1 - \beta) pai_2$
- Recombinação geométrica
  - $filho = (pai_1 * pai_2)^{1/2}$
- Recombinação uniforme
  - Para cada gene, gene de um dos pais é aleatoriamente escolhido

Crossover Média

- Tendem a levar valores dos genes para centro do espaço de busca
  - Reduz diversidade
- Não extrapola para além da população inicial
- Problema minimizado pelo *blend crossover*

Crossover Aritmético

- Blend crossover (BLX)
  mãe = $(m_1, m_2, \ldots, m_l)$
  pai = $(p_1, p_2, \ldots, p_l)$
  filho = $(a_1, a_2, \ldots, , a_l)$
  $a_i = m_i + \beta (p_i - m_i)$

  onde β é um número aleatório escolhido de uma distribuição uniforme no intervalo [-α, 1+α]

  Tipicamente α, = 0,5 ou 0,25

Crossover Aritmético (2)

- Exemplo de Blend crossover (BLX)

  mãe = (30,173; 85,342)

  pai = (75,989; 10,162)

  Supor α, = 0,5 e β = 1,262

  a_1 = 30,173 +1,262(75,989-30,173) = 87,993

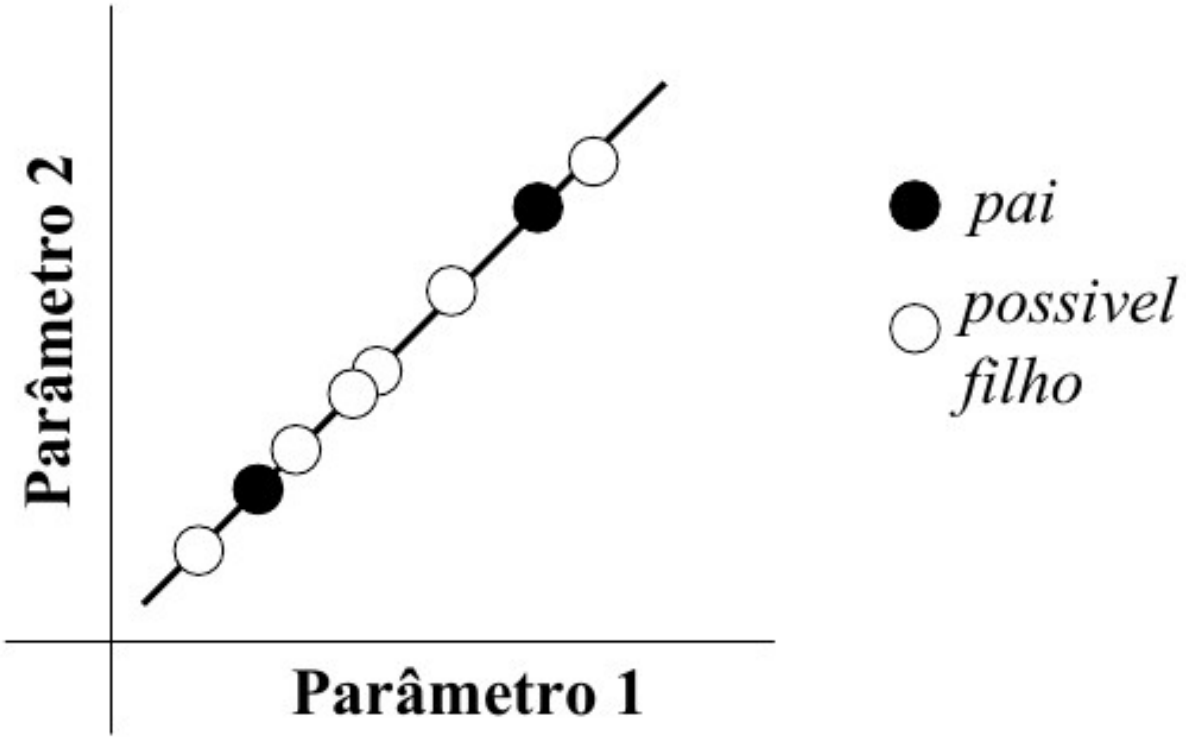
  a_2 = 85,342 +1,262(10,162-85,342) = -9,535

  assim,

  filho = (87,993; -9,535)

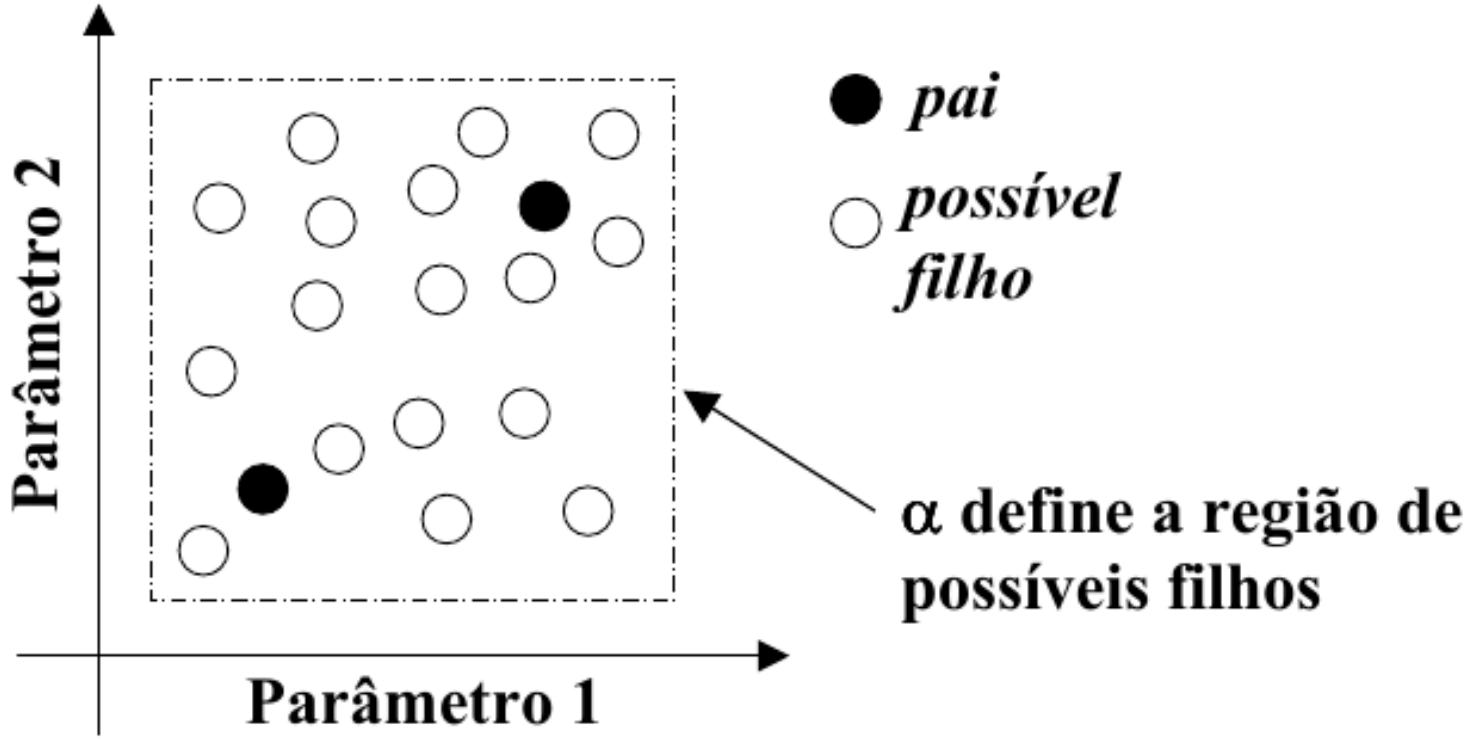
Crossover Aritmético (3)

- Blend crossover (BLX)
  - Usando β igual para cada parâmetro



Crossover Aritmético (4)

- Blend crossover (BLX)
  - Usando β diferente para cada parâmetro



Mutação Representação Real

- Mutação randômica
  - Substituição do valor do parâmetro por outro valor aleatoriamente escolhido
- Mutação creep
  - Adiciona ao parâmetro pequeno valor aleatório
    - Obtido de uma distribuição uniforme ou normal

Operadores de Michalewicz

- Crossover Simples
- Crossover Aritimético
- Crossover Heurístico
- Mutação Uniforme
- Mutação de Limite
- Mutação Não-uniforme
- Mutação Não-uniforme Múltipla

Observações

- Se o AG estiver corretamente implementado, a população geralmente evolui em gerações sucessivas
  - Até estabilizar
- Aptidões do melhor indivíduo e do indivíduo médio aumentam em direção a um ótimo global

AGs Co-Evolutivos

- Utilizam mais de uma população durante processo evolutivo
- Empregados quando
  - Domínio é muito complexo
  - Difícil ou impossível avaliar uma função de aptidão para o problema

AGs Co-Evolutivos (2)

- Podem ser
  - Cooperativos
    - Colaborações entre duas populações
      - População de possíveis soluções
      - População de instâncias do problema a ser resolvido
  - Competitivos
    - Competição entre populações
      - Predadores-Presa (Hospedeiro-Parasita)
    - Muito usados em jogos
    - Maioria dos algoritmos co-evolutivos

AGs Co-Evolutivos (3)

- Geralmente utilizados para decompor um problema grande em sub-problemas
  - Uma população (processo evolutivo) para cada sub-problema
  - Populações interagem
    - Função de aptidão de uma população pode depender do estado do processo evolutivo de outra(s) população(ções)

Métodos de Nicho

- Nicho é a área particular dentro de um habitat ocupada por uma espécie
  - Diversidade de nichos é uma das principais razões para diversidade biológica
  - Permite formar várias espécies em torno de diferentes mínimos locais
    - Mantém a diversidade da população

Métodos de Nicho (2)

- Dois importantes métodos de nicho usados por AGs são:
  - Compartilhamento de fitness
  - Crowding

Métodos de Nicho (3)

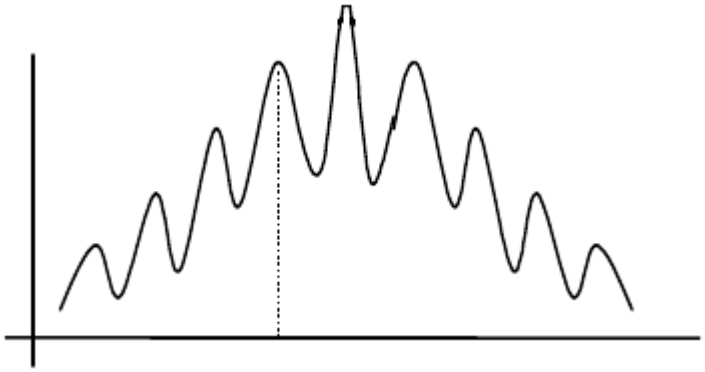
- Compartilhamento de fitness
  - Valores de aptidão de indivíduos dentro de uma mesma espécie são reduzidos
    - De acordo com o número de indivíduos da espécie
    - De modo a afetar sua probabilidade de serem selecionadas para a próxima geração

Métodos de Nicho (4)

- Crowding
  - Substitui os antigos indivíduos ruins por novos indivíduos bons próximos a eles
  - Evita que a população fico muito compacta, não destruindo a diversidade da população

Convergência Prematura

- Causas:
  - Excessivo números de filhos de um mesmo individuo (o superindividuo)
  - Perda de diversidade
- O Algoritmo converge para um mínimo/máximo local



Convergência Prematura (2)

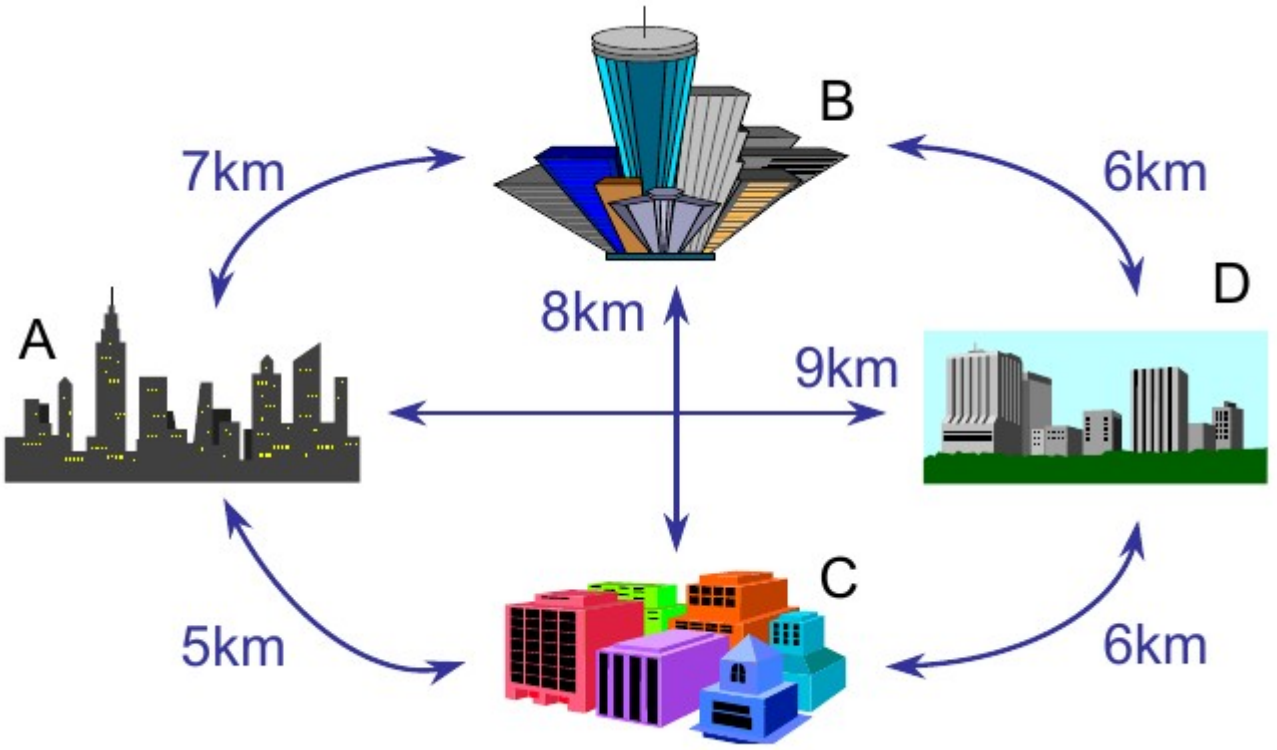
- Combatendo a perda de diversidade
  - Aumentar a taxa de mutação
  - Evitar cromossomos duplicados na população
  - Diminuir a pressão da seleção
- Controlar o número de filhos de um indivíduo
  - Ranking, escalamento, seleção por torneio

Aplicações

- Otimização de função numérica
- Otimização combinatorial
  - Problema do caixeiro viajante
  - Problema de empacotamento
  - Alocação de recursos (*job shop schedulling*)
- Projetos
  - Projeto de pontes
- Aprendizado de Máquina
  - Jogos

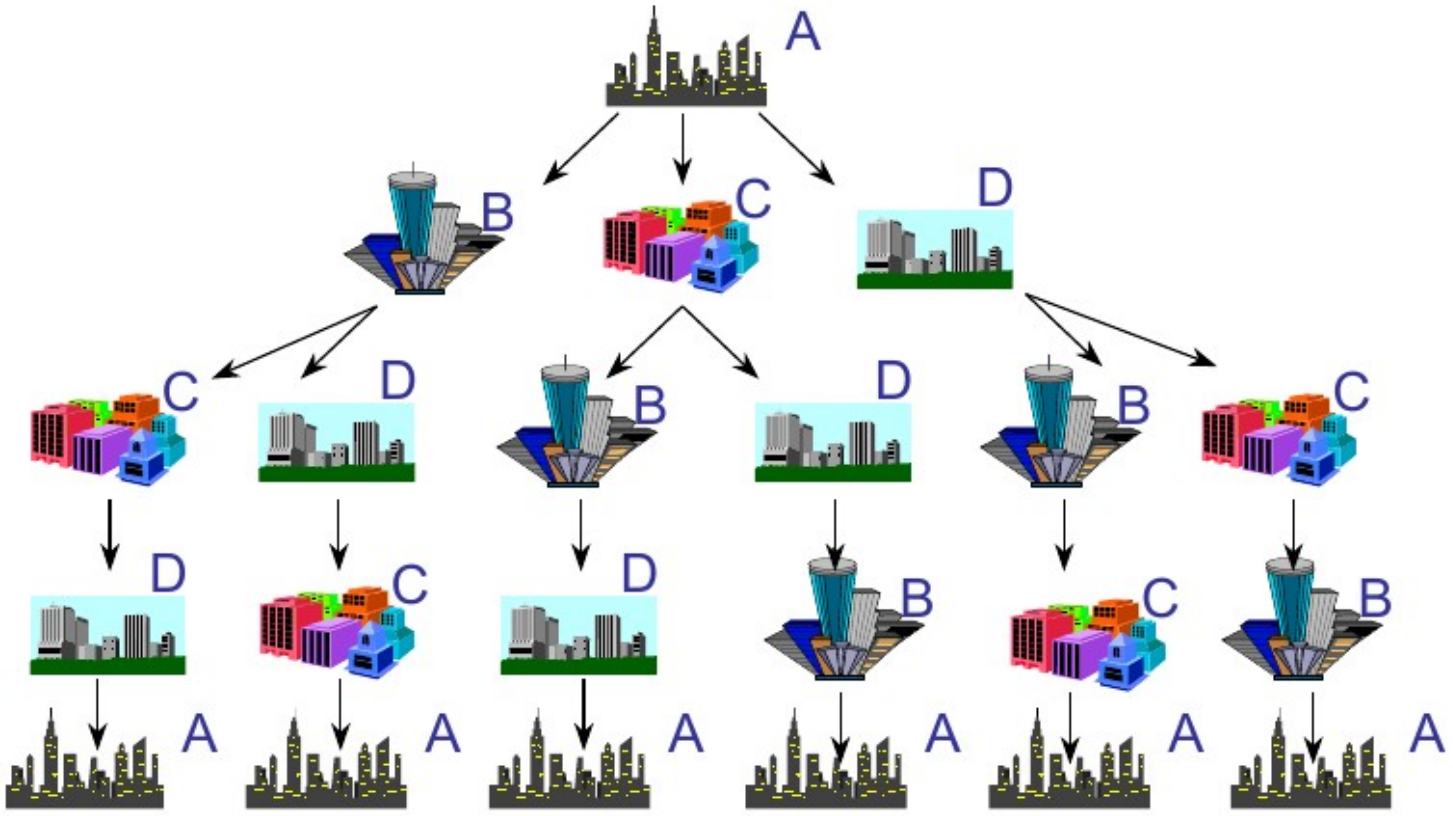
O problema TSP

- Considere as rotas definidas entre as cidades A, B, C e D:



O problema TSP (2)

- O problema do TSP representado como árvore de busca



O problema TSP (3)

- Problema: Explosão Combinatória
  - Com quatro cidades, existem 6 caminhos possíveis
  - Com dez cidades, existem 362.880 caminhos possíveis
  - Quanto mais cidades forem adicionadas ao TSP, mais caminhos possíveis vão existir
  - O que leva a uma **explosão combinatorial**

TSP

- A simplicidade da definição do problema é enganadora
  - TSP é um dos problemas mais estudados em matemática computacional e IA
  - Não existe nenhuma solução eficiente para o caso geral
  - Quem resolver este problema para o caso geral receberá um prêmio de **$1,000,000** do Clay Mathematics Institute

TSP (2)

- Embora a complexidade do TSP ainda seja desconhecida, nos últimos cinquenta anos seu estudo tem levado à melhoria de métodos de solução de vários problemas de otimização

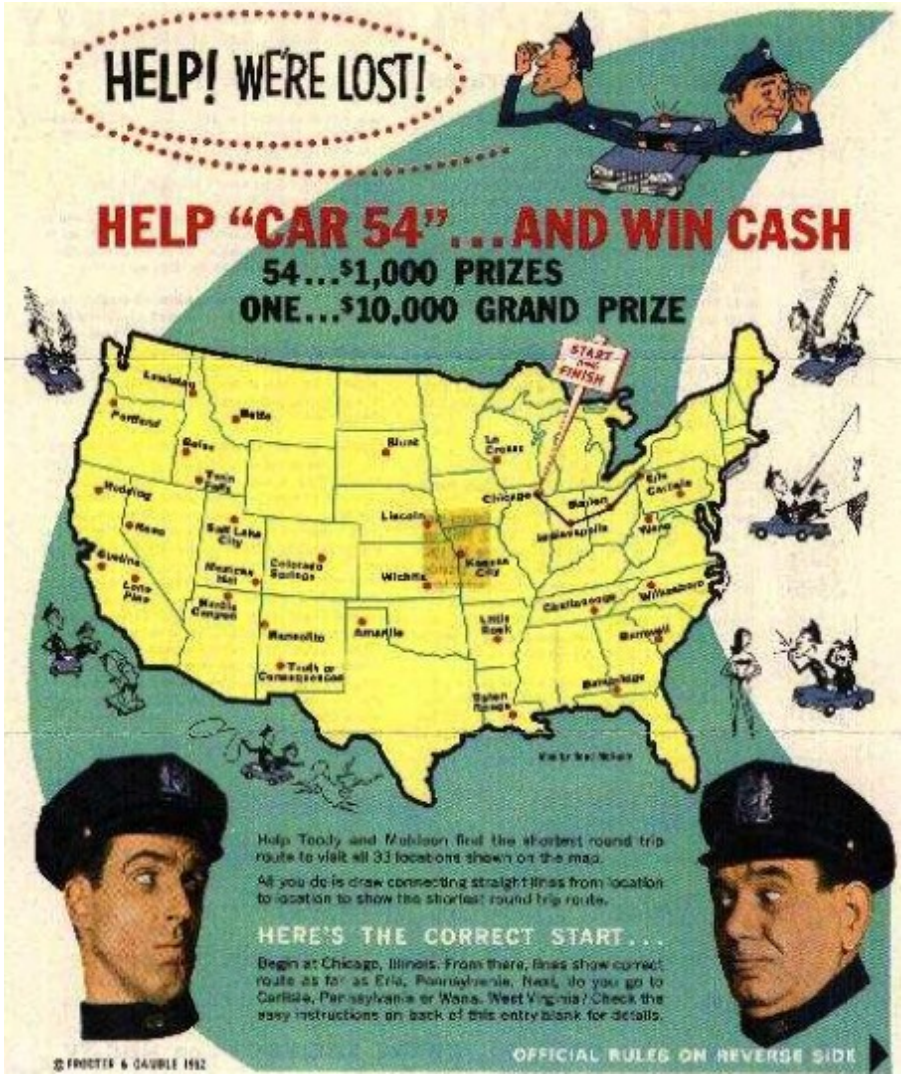
Evolução das Soluções

Ano	Grupo	No. Cidades
1954	G. Dantzig, R. Fulkerson, and S. Johnson	49
1971	M. Held and R.M. Karp	64
1975	P.M. Camerini, L. Fratta, and F. Maffioli	67
1977	M. Grötschel	120
1980	H. Crowder and M.W. Padberg	318
1987	M. Padberg and G. Rinaldi	532
1987	M. Grötschel and O. Holland	666
1987	M. Padberg and G. Rinaldi	2.392
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	7.397
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	13.509
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	15.112
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook, K. Helsgaun	24.978

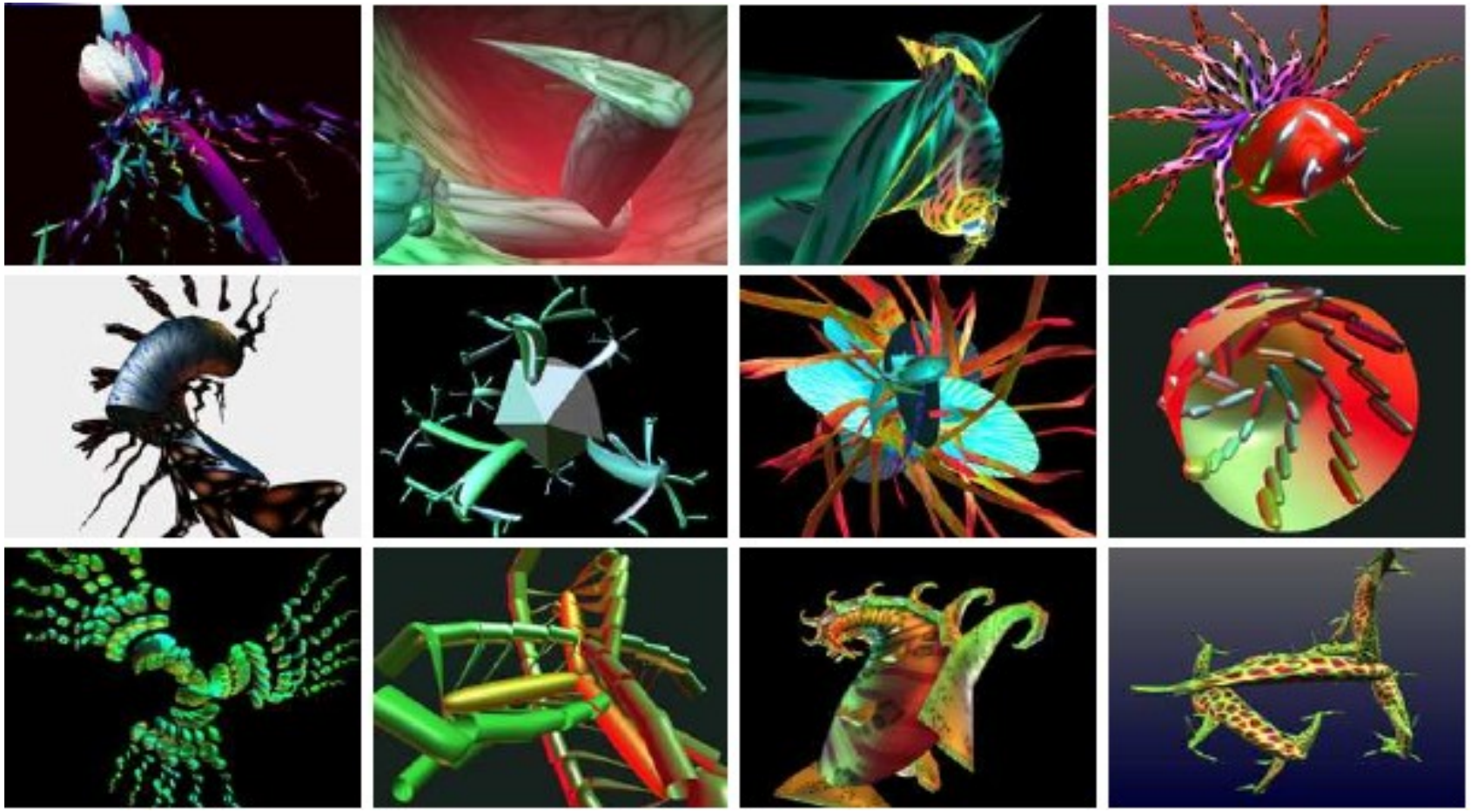
Evolução das Soluções (2)

- Problema de TSP visitando 24.978 cidades na Suécia foi resolvido em maio de 2004
  - Percurso total: aprox. 72500 Km
  - Foi provado que não existia caminho mais curto

Evolução das Soluções (3)



Arte Evolutiva



http://www.genarts.com/galapagos/galapagos-images.html

Conclusão

- Computação Evolutiva
- Algoritmos Genéticos
  - Codificação
  - Função de aptidão
  - Operadores Genéticos
  - Reprodução
  - Aplicações

Créditos

- Prof. Dr. André C. P. L. F. de Carvalho - ICMC-USP