Sistemas Inteligentes – if684

- Germano Vasconcelos – gcv@cin.ufpe.br
- Página da Disciplina: www.cin.ufpe.br/~îf684/EC

Algoritmos Genéticos

Algoritmos Genéticos

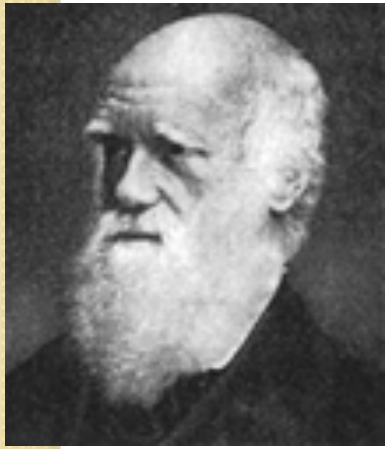
- Técnicas de busca e otimização
- Metáfora da evolução das espécies de Charles Darwin
- Desenvolvido por Holland (1975) et al
- Popularizado por Goldberg (1989)

História – Pré-darwinianos

- T. Malthus (~1800) – Pastor anglicano e economista
  - Idéia de economia da natureza
- J. Lamarck – Milico e naturalista (1809)
  - Criou a expressão "biologia"
  - Evolução, baseada em uso e desuso e hereditariedade

História: Teoria da Evolução

- 1859 – Charles Darwin publica o livro "*A Origem das Espécies*":



Charles Darwin

"As espécies evoluem pelo principio da seleção natural e sobrevivência do mais apto"

História: Teoria da Evolução

Gregor Mendel

- 1865- Gregor Mendel apresenta experimentos do cruzamento genético de ervilhas
  - Pai da genética
- A Teoria da Evolução começou a partir da conceituação integrada da seleção natural com a genética

Conceitos de AG

- Indivíduo
  - Simples membro da população
  - Nos AGs, uma possível solução
- Cromossomo e Genoma
  - Coleção de genes
  - Estrutura de dados que codifica a solução
- Genótipo
  - Em biologia, a composição genética no genoma
  - Nos AGs, a informação no cromossomo ou genoma

Conceitos de AG

- Fenótipo:
  - Objeto ou estrutura construída a partir das informações do genótipo
  - É o cromossomo decodificado
    - Exemplo: Se o cromossomo codifica as dimensões de um edifício, então o fenótipo é o edifício construído
- Gene:
  - Codifica um simples parâmetro do problema

Otimização

- Busca da melhor solução para um problema
  - Tentar várias soluções e usar a informação para conseguir soluções cada vez melhores
- Exemplo de otimização:
  - Telespectador ajusta a antena da televisão para otimizar a imagem buscando várias soluções até alcançar a melhor
  - Empacotador coloca os objetos em uma caixa de forma a otimizar a ocupação do espaço

Otimização

- As técnicas de otimização geralmente apresentam:
  - **Espaço de busca**: onde estão todas as possíveis soluções do problema
  - **Função objetivo**: para avaliar as soluções produzidas associando uma nota

Características dos Algoritmos Genéticos

- É um algoritmo estocástico (é não determinístico)
- Trabalha com uma população de soluções simultaneamente
- Utiliza informações de custo e recompensa

Características dos Algoritmos Genéticos (II)

- Fáceis de implementação em computadores
- Adaptam-se bem a computadores paralelos
- Facilmente hibridizados com outras técnicas
- Funcionam com parâmetros contínuos ou discretos

Algoritmos Genéticos (Conceitos Básicos)

- AG manipula uma população de indivíduos
- Individuos são possíveis soluções do problema
- Indivíduos são combinados (crossover) produzindo filhos que podem sofrer ou não mutação
- Populações evoluem em sucessivas gerações até encontrar a solução ótima

Aplicações

- Otimização de funções numéricas em geral
- Otimização combinatória
  - Problema do caixeiro viajante
  - Problema de empacotamento
  - Alocação de recursos (*job shop schedulling*)
- Aprendizado de máquina
- Problemas difíceis de otimização quando não existe técnica exata para resolver o problema ou o custo é elevado
- Busca de soluções em previsão de séries temporais

Algoritmo Genético Tradicional

- Algoritmo básico:

  Gerar população inicial

  `while` critério-de-parada **`do`**

  Escolher cromossomos reprodutores

  Fazer o *crossover* dos reprodutores

  Gerar mutações

  Avaliar aptidões e atualizar a população

  `end-while`

- Parâmetros importantes:
  - tamanho da população
  - critério de seleção
  - função de aptidão
  - critério de cruzamento (crossover)
  - taxa de mutação
  - critério de sobrevivência dos cromossomos
  - critério de parada (estabilização da população, impossibilidade de melhorar a melhor solução, número de gerações)

Algoritmo Genético Tradicional

- Função de aptidão:
  - utilizada para quantificar a qualidade genética dos cromossomos, correspondendo à função de custo em problemas de otimização combinatória.
  - utilizada para selecionar os indivíduos reprodutores.
  - utilizada para decidir se um cromossomo gerado através de um *crossover* substitui ou não um cromossomo reprodutor.

- *Crossover*: operação probabilística (originalmente), onde os indivíduos mais adaptados têm maior chance de participar

- *Crossover* uniforme: cada *bit* de um filho é gerado escolhendo-se aleatoriamente um dos pais e repetindo-se o *bit* do pai escolhido

Algoritmo Genético Tradicional

- *Crossover* de um ponto:

  reprodutores: 2 cromossomos de n bits

  $a = (a_1,\ldots,a_k,\ldots,a_n)$ $b = (b_1,\ldots,b_k,\ldots,b_n)$

  operação: $k \in \{1,\ldots,n\}$ aleatório

  filhos:

  $(a_1,\ldots,a_k,b_{k+1},\ldots,b_n)$ $(b_1,\ldots,b_k,a_{k+1},\ldots,a_n)$

- *Crossover* de dois (ou mais) pontos

- *Crossover* por fusão: como o uniforme, mas a probabilidade de escolha de cada pai é proporcional à sua aptidão

- Utilizar consenso no *crossover*: repetir *bits* comuns aos dois reprodutores

Algoritmo Genético Tradicional

- Mutação: normalmente implementada com a complementação de bits da população.
- Seleção aleatória dos *bits* a serem modificados: percentual muito baixo do total de bits na população de cromossomos.
- Mutação é o mecanismo para introduzir diversidade na população.
- Critérios de atualização da população podem também permitir que pais e filhos permaneçam na população, removendo-se sempre os menos aptos, sendo possível a utilização de cromossomos pais e filhos como reprodutores.

Indivíduo

- Cromossomo
  - Estrutura de dados que representa uma possível solução para o problema
  - Os parâmetros do problema de otimização são representados por cadeias de valores
  - Exemplos:
    - Vetores de reais, (2.345, 4.3454, 5.1, 3.4)
    - Cadeias de bits, (111011011)
    - Vetores de inteiros, (1,4,2,5,2,8)
    - ou outra estrutura de dados

Individuo (II)

- Aptidão
  - Nota associada que avalia a solução representada pelo individuo
- Aptidão pode ser:
  - Igual à função objetivo
  - Resultado do **escalonamento** da função objetivo

Problema 1

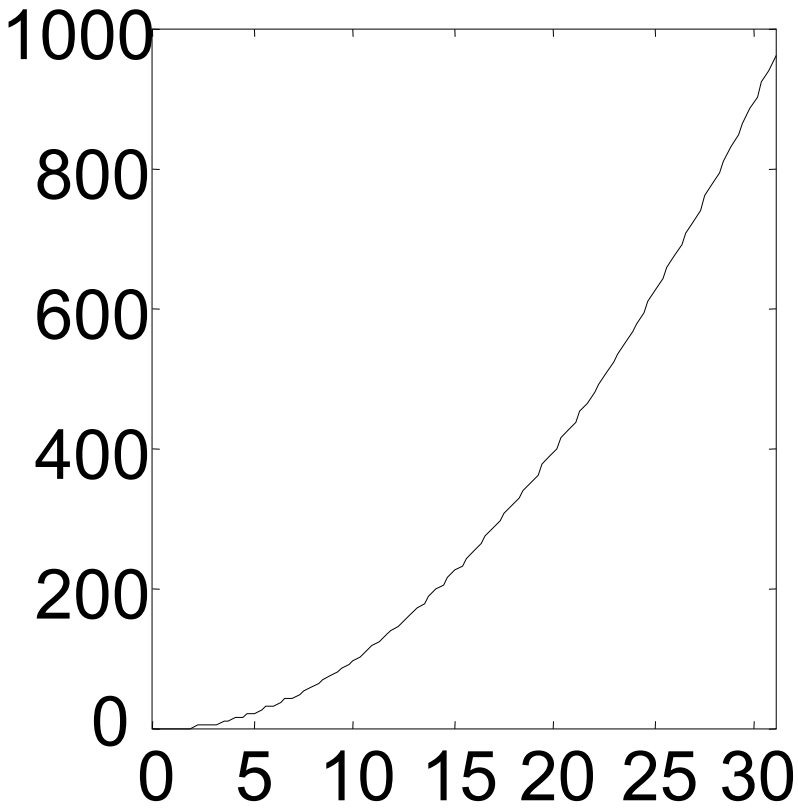
Problema: Use um AG para encontrar o ponto máximo da função:

$$f(x) = x^2$$

com x sujeito as seguintes restrições:

$$0 \leq x \leq 31$$

x é inteiro



Cromossomo do Problema 1

- Cromossomos binários com 5 bits:
  - 0 = 00000
  - 31 = 11111
- Aptidão
  - Neste problema, pode ser a própria função objetivo
  - Exemplo:

$$\text{aptidão}(00011) = f(3) = 9$$

Seleção

- Seleção
  - Imitação da seleção natural
  - Os melhores indivíduos (maior aptidão) são selecionados para gerar filhos através de crossover e mutação
  - Dirige o AG para as melhores regiões do espaço de busca
- Tipos mais comuns de seleção
  - Proporcional à aptidão (Roleta)
  - Torneio

População Inicial do Problema 1

É aleatória (mas quando possível, o conhecimento da aplicação pode ser utilizado para definir população inicial)

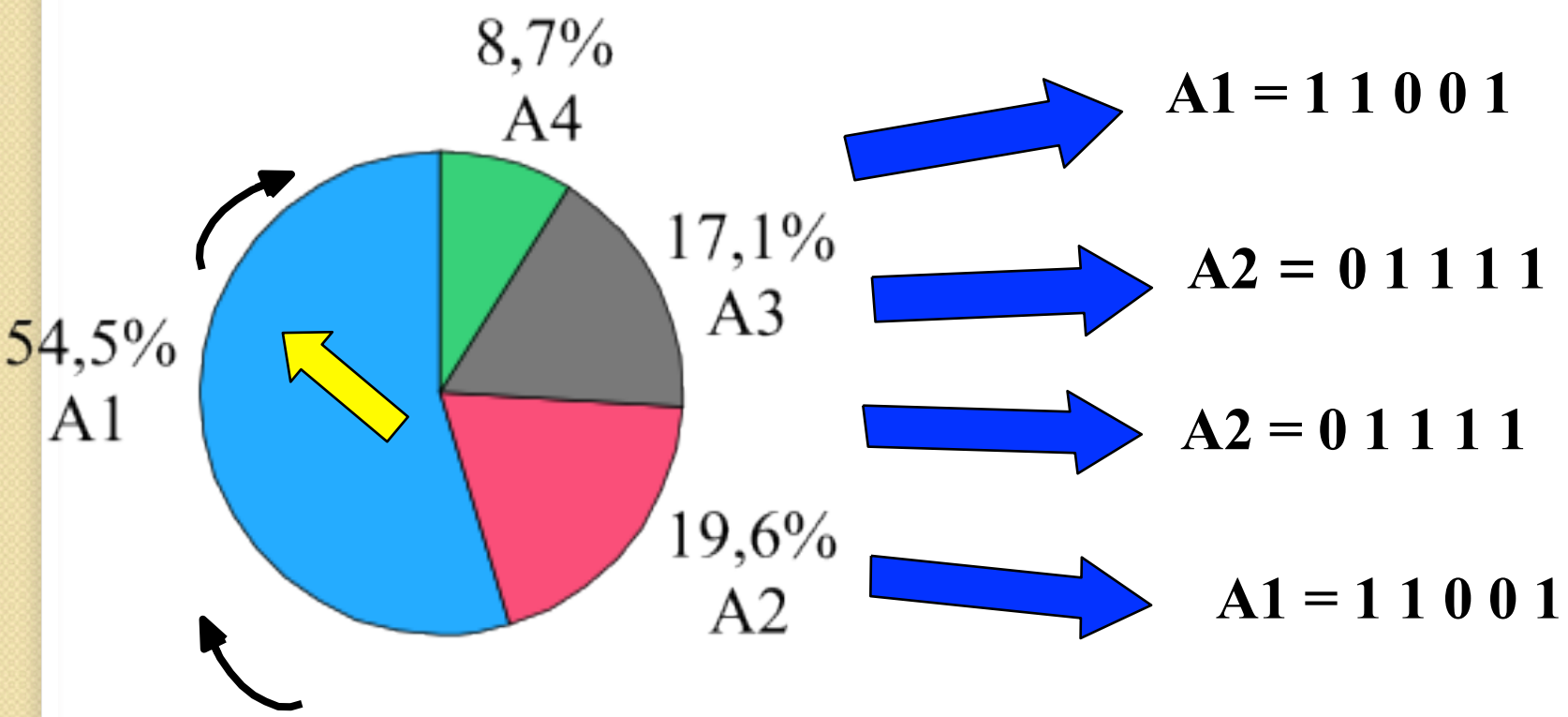
Pop. inicial

cromossomos	x	f(x)	Prob. de seleção
$A_1 = 1\ 1\ 0\ 0\ 1$	25	625	54,5%
$A_2 = 0\ 1\ 1\ 1\ 1$	15	225	19,6%
$A_3 = 0\ 1\ 1\ 1\ 0$	14	196	17,1%
$A_4 = 0\ 1\ 0\ 1\ 0$	10	100	8,7%

Probabilidade de seleção proporcional a aptidão

$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}$$

Seleção Proporcional à Aptidão (Roleta)



Seleção por Torneio

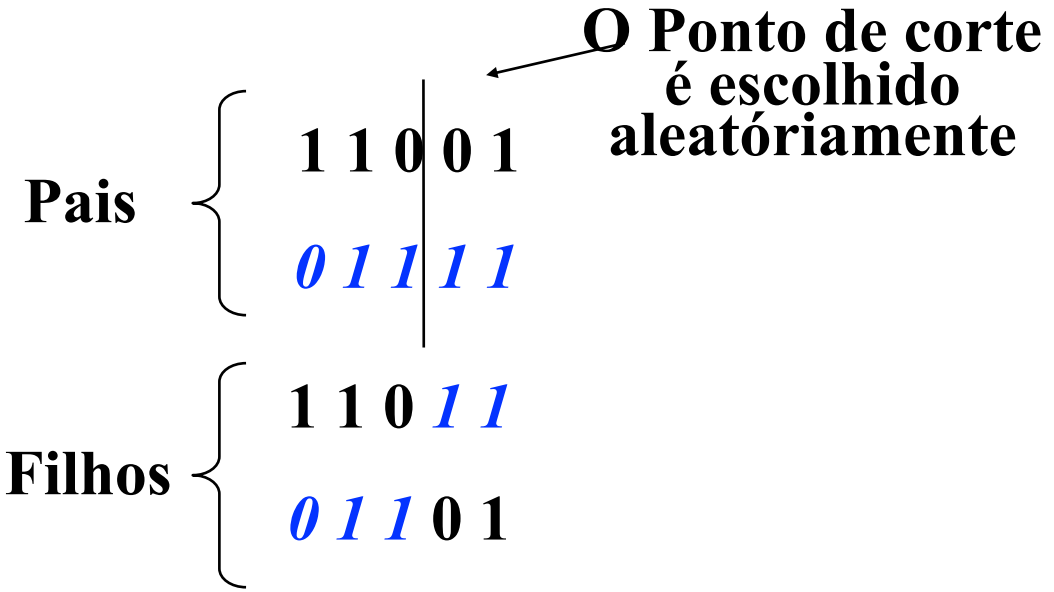
- Escolhe-se *n* (de 4, escolhe-se tipicamente 2, por exemplo) indivíduos aleatoriamente da população e o melhor é selecionado

Crossover e Mutação

- Combinam ou modificam pais selecionados para produção de filhos
- Principais mecanismos de busca do AG
- Permite explorar áreas desconhecidas do espaço de busca

Crossover de 1 Ponto

O crossover é aplicado com uma dada probabilidade denominada *taxa de crossover* (60% a 90%)



Se o crossover é aplicado os pais trocam suas caldas gerando dois filhos, caso contrário os dois filhos serão cópias exatas dos pais.

Mutação

Mutação inverte os valores dos bits

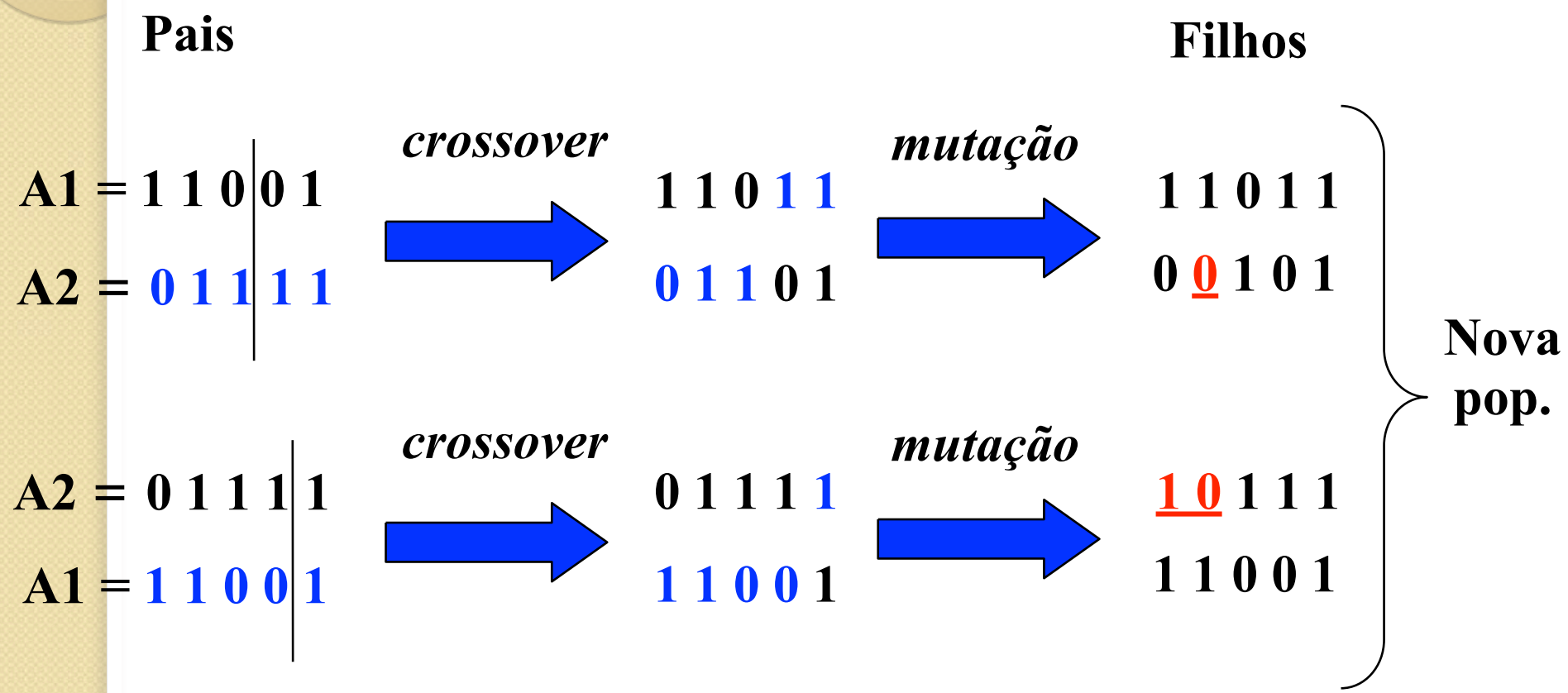
A mutação é aplicada com dada probabilidade, denominada *taxa de mutação* (~1%), em cada um dos bits do cromossomo

Antes da mutação	**0 1 1 0 1**
Depois	**0 0 1 0 1**

Aqui, apenas o 2o.bit passou no teste de probabilidade

A taxa de mutação em geral deve ser baixa, o suficiente para assegurar a diversidade de cromossomos na população

A primeira geração do Problema 1



A primeira geração do Problema 1 (II)

	cromossomos	x	$f(x)$	prob. de seleção
1	1 1 0 1 1	27	729	29,1%
2	1 1 0 0 1	25	625	24,9%
3	1 1 0 0 1	25	625	24,9%
4	1 0 1 1 1	23	529	21,1%

As demais gerações do Problema 1

Segunda Geração

		x	$f(x)$
1	1 1 0 1 1	27	729
2	1 1 0 0 0	24	576
3	1 0 1 1 1	23	529
4	1 0 1 0 1	21	441

Terceira Geração

		x	$f(x)$
1	1 1 0 1 1	27	729
2	1 0 1 1 1	23	529
3	0 1 1 1 1	15	225
4	0 0 1 1 1	7	49

As demais gerações do Problema 1 (II)

Quarta Geração

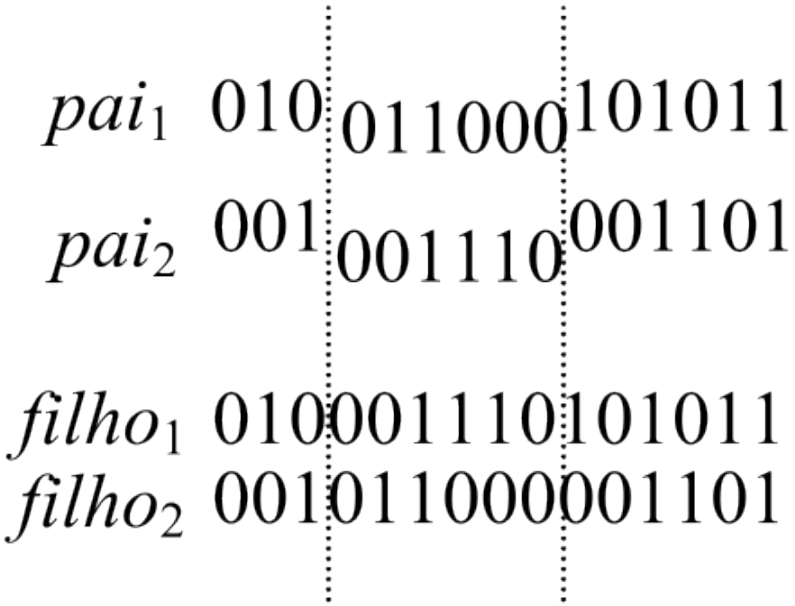
		x	$f(x)$
1	1 1 1 1 1	31	961
2	1 1 0 1 1	27	729
3	1 0 1 1 1	23	529
4	1 0 1 1 1	23	529

Quinta Geração

		x	$f(x)$
1	1 1 1 1 1	31	961
2	1 1 1 1 1	31	961
3	1 1 1 1 1	31	961
4	1 0 1 1 1	23	529

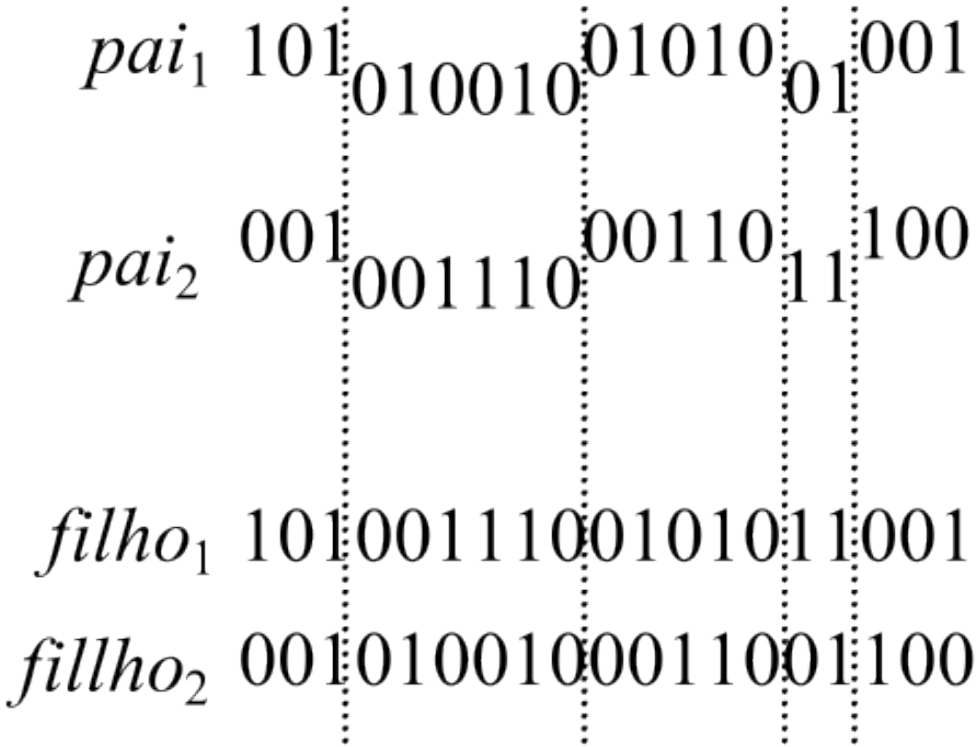
Outros Crossover´s

- Crossover de 2-pontos



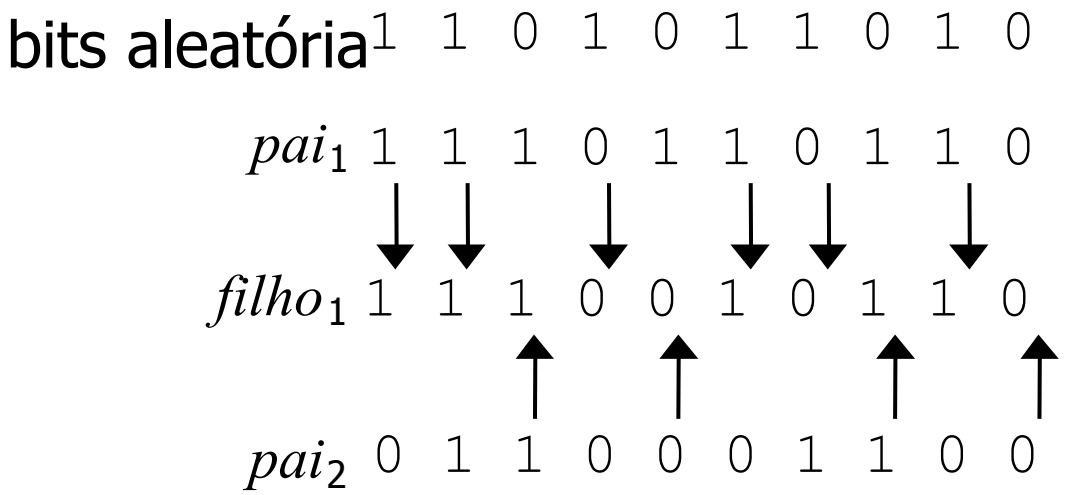
Considerado melhor que o crossover de 1 ponto.

Crossover de n-Pontos



Crossover de 4-pontos

Crossover Uniforme



O *filho*1 tem 50% de chance de levar um bit do *pai*1 e 50% de chance de levar um bit de *pai*2

O *filho*2 leva o que sobra de *pai*1 e *pai*2

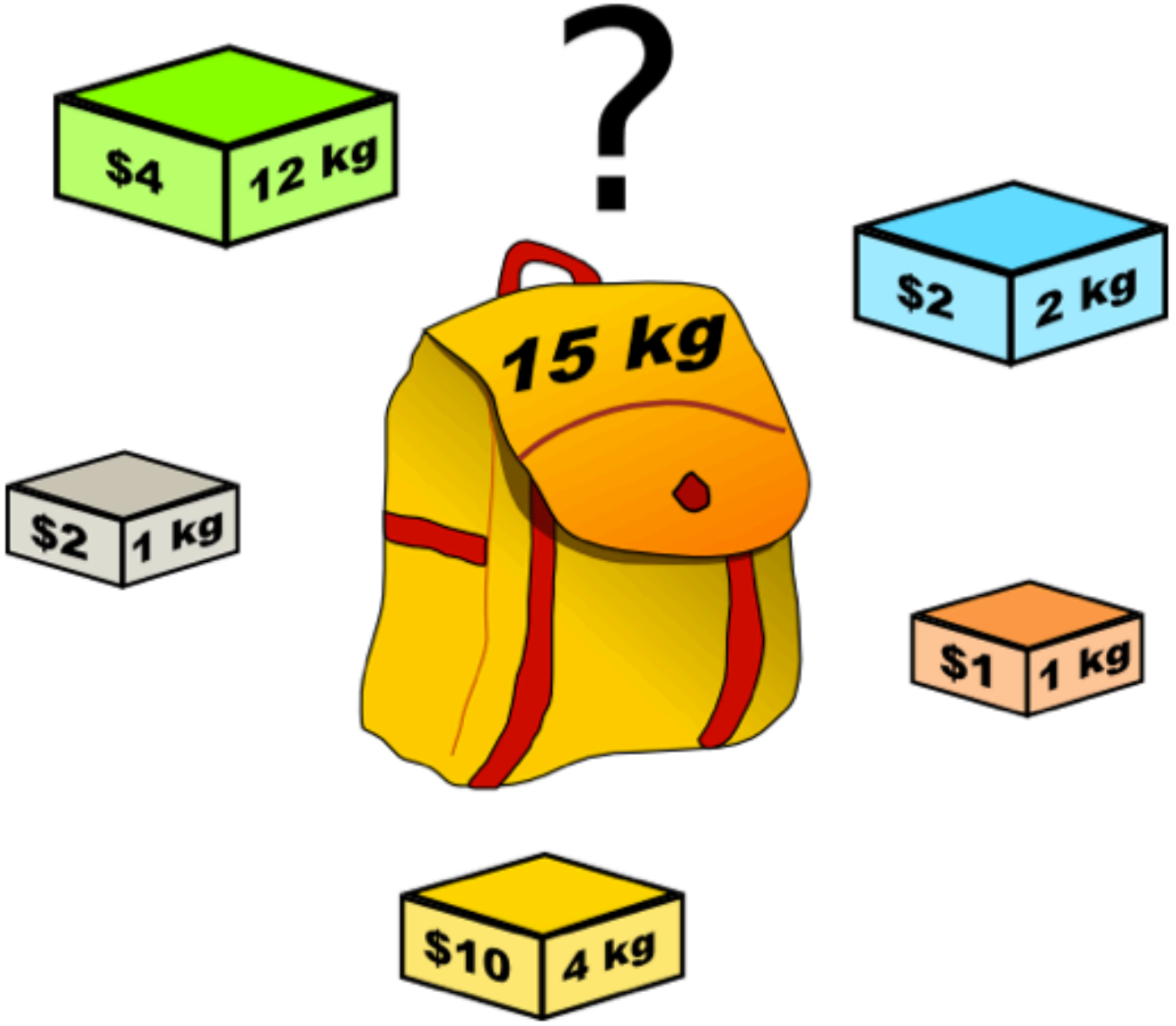
Elitismo

- O crossover ou mutação podem destruir o melhor indivíduo
- Por que perder a melhor solução encontrada?
- Elitismo transfere a cópia do melhor indíviduo para a geração seguinte

Critérios de Parada

- Número de gerações
- Encontrou a solução (quando esta é conhecida)
- Perda de diversidade
- Convergência
  - nas últimas k gerações não houve melhoria na aptidão

O Problema da Mochila



Problema da Mochila

- Dados um conjunto de *n* objetos e uma mochila com:
  - c_j = benefício do objeto *j*
  - w_j = peso do objeto *j*
  - *b* = capacidade da mochila
- Determinar quais objetos devem ser colocados na mochila para maximizar o benefício total de tal forma que o peso da mochila não ultrapasse sua capacidade.

Problema da Mochila

(do inglês, 0-1 knapsack problem)

Maximizar $z = \sum_{j=1}^{n} c_j s_j$

Sujeita a $\sum_{j=1}^{n} w_j s_j \leqslant b$

$s_j \in \{0,1\}$

Uma solução s é um vetor de uns e zeros.
Se o objeto j está mochila então $s_j = 1$, caso contrário $s_j = 0$.

Problema da Mochila

- **Cromossomo**
  - A solução s (um vetor de uns e zeros) é naturalmente representada por um cromossomo binário.
- **Operadores binários padrão**
  - Crossover de 1-ponto (ou 2-pontos, etc)
  - Mutação (invertendo os bits)

Problema da Mochila

Objeto (j)	1	2	3	4	5	6	7	8
Benefício (c_j)	3	3	2	4	2	3	5	2
Peso (w_j)	5	4	7	8	4	4	6	8

Capacidade da mochila: $b = 25$

11001110 (cromossomo válido)

peso = 5+4+4+4+6 = 23 ≤ 25
função objetivo = 3+3+2+3+5 = 16

11111001 (inválido)

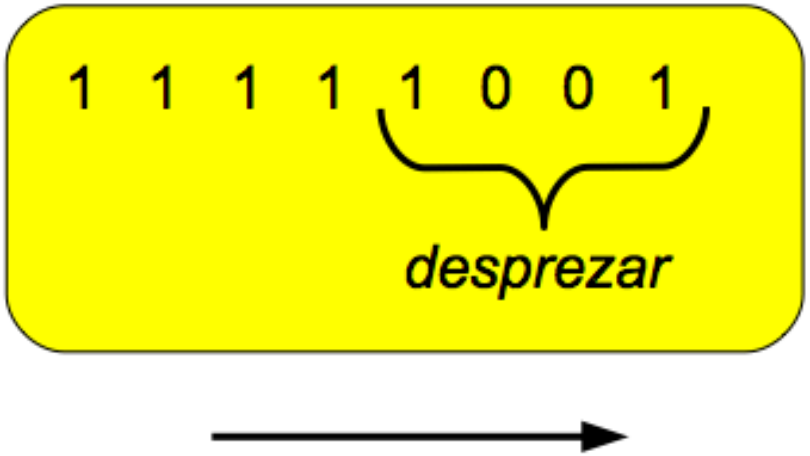
peso = 36 > 25
Função objetivo = 16

Problema da Mochila – Cromossomos Inválidos

- Solução 1 – reparar o indivíduo
- Solução 2 – penalizar a função objetivo

Problema da Mochila – Reparação

- **Indivíduo inválido**
  - 1 1 1 1 1 0 0 1
  - peso = 36 > 25
  - Função objetivo = 16



- **Indivíduo "reparado"**
  - 1 1 1 1 0 0 0 0
  - Peso = 24 (ok!)
  - Função objetivo = 12

visitar cada bit da esquerda para a direita e desprezar os bits que invalidam a solução.

Problema da Mochila – Reparação

- Por qual ordem dos bits devem ser visitados?
  - Da esquerda para direita?
  - No sentido oposto?
  - Aleatoriamente?
- Algoritmo Guloso
  - Visitar primeiro os bits com a maior razão benefício/peso;
  - Pode produzir melhores resultados.

Problema da Mochila – Penalização

Um exemplo de penalidade é:

$$f(s)=\sum_{j=1}^{n} c_j s_j - \alpha \times max\left(0, \sum_{j=1}^{n} s_j w_j - b\right)$$

Onde α é um coeficiente de penalidade igual a:

$$\alpha=\sum_{j=1}^{n} c_j = 14$$

Objetos que ultrapassam a capacidade da mochila são penalizados.

Problema da Mochila – Penalização

- **Exemplo**
  - 1 1 1 1 1 0 0 1
  - peso = 36 > 25
  - Função original = 16
  - Função com penalidade = 16 – 14 x (36-25) = -138

Exercício – Mochila

	Item	Preço	Volume
1	Notebook	2.500,00	40
2	Máquina fotografica	1.200,00	15
3	Netbook	1.500,00	20
4	Smart phone	850,00	5
5	Impressora	350,00	10
6	Video game	2.000,00	35

- Objetivo: Maximizar Z=somatorio Preco_j*s_j (j = 1 a 6) (s_j = [0 ou 1]) sujeita a somatorio Volume_j*s_j (j = 1 a 6) <= 50
- a) Proponha uma função de aptidão, para verificar o custo de cada cromossomo
  - R:
- b) Faça a codificação de 4 cromossomos e calcule a função de aptidão
  - R:
  - R:
- c) Faça o crossover entre os dois pares de cromossomos codificados no item anterior (ponto de corte na segunda casa)
  - R:
  - R:
- d) Gere os descendentes (4 descendentes)
  - R:
  - R:
- e) Aplique mutação nos descendentes em dois genes (3ª e 5ª casa)
  - R:
- f) Aplique a função de aptidão nos filhos gerados mostrando a melhor solução
  - R:
  - R:

Exercício – Mochila

	Item	Preço	Volume
1	Notebook	2.500,00	40
2	Máquina fotografica	1.200,00	15
3	Netbook	1.500,00	20
4	Smart phone	850,00	5
5	Impressora	350,00	10
6	Video game	2.000,00	35

- Objetivo: Maximizar Z=somatorio Preco_j*s_j (j = 1 a 6) (s_j = [0 ou 1]) sujeita a somatorio Volume_j*s_j (j = 1 a 6) <= 50
- a) Proponha uma função de aptidão, para verificar o custo de cada cromossomo
  - R: Z=somatorio Preco_j*s_j (j = 1 a 6) (s_j = [0 ou 1])
- b) Faça a codificação de 4 cromossomos e calcule a função de aptidão
  - R: [1 1 0 0 0 0] , [0 1 1 1 0 0] , [0 1 0 1 1 0] , [1 0 0 0 1 0]
  - R: R$3.700 , R$3.550 , R$2.400 , R$2.850
- c) Faça o crossover entre os dois pares de cromossomos codificados no item anterior (ponto de corte na segunda casa)
  - R: Seleção por roleta = [1 1 0 0 0 0], [1 0 0 0 1 0] , [1 1 0 0 0 0] , [1 1 0 0 0 0]
  - R: Crossover Pais: [1 1 | 0 0 0 0] , [1 0 | 0 0 1 0] , [1 1 | 0 0 0 0] , [1 1 | 0 0 0 0]
- d) Gere os descendentes (4 descendentes)
  - R: Filhos: [1 1 | 0 0 1 0] , [1 0 | 0 0 0 0] , [1 1 | 0 0 0 0] , [1 1 | 0 0 0 0]
  - R: Filhos (ajustados): [1 0 0 0 1 0] , [1 0 0 0 0 0] , [1 0 0 0 0 0] , [1 0 0 0 0 0] (da esquerda para direita, para cada bit=1, inclua o item na mochila se somado ao volume dos já inclusos não excede o volume limite)
- e) Aplique mutação nos descendentes em dois genes (3ª e 5ª casa)
  - R: [1 0 0 0 1 0] , [1 0 0 0 1 0] , [1 0 0 0 0 0] , [1 0 0 0 0 0]
- f) Aplique a função de aptidão nos filhos gerados mostrando a melhor solução
  - R: R$2.850 , R$2.850 , R$2.500,00 , R$2.500,00
  - R: [1 0 0 0 1 0]

Exercício 3

- Encontrar valor de x para o qual a função f(x) = x2 – 3x + 4 assume o valor mínimo
  - Assumir que $x \in [-10, +10]$
  - Codificar X como vetor binário
  - Criar uma população inicial com 4 indivíduos
  - Aplicar Mutação com taxa de 1%
  - Aplicar Crossover com taxa de 60%
  - Usar seleção por torneio
  - Usar 5 gerações.

Referências 1

- Obitko, Marek, 1998, Ermelindo Pinheiro Manoel (Versão em Português)
- http://www.obitko.com/tutorials/genetic-algorithms/portuguese/resources.php

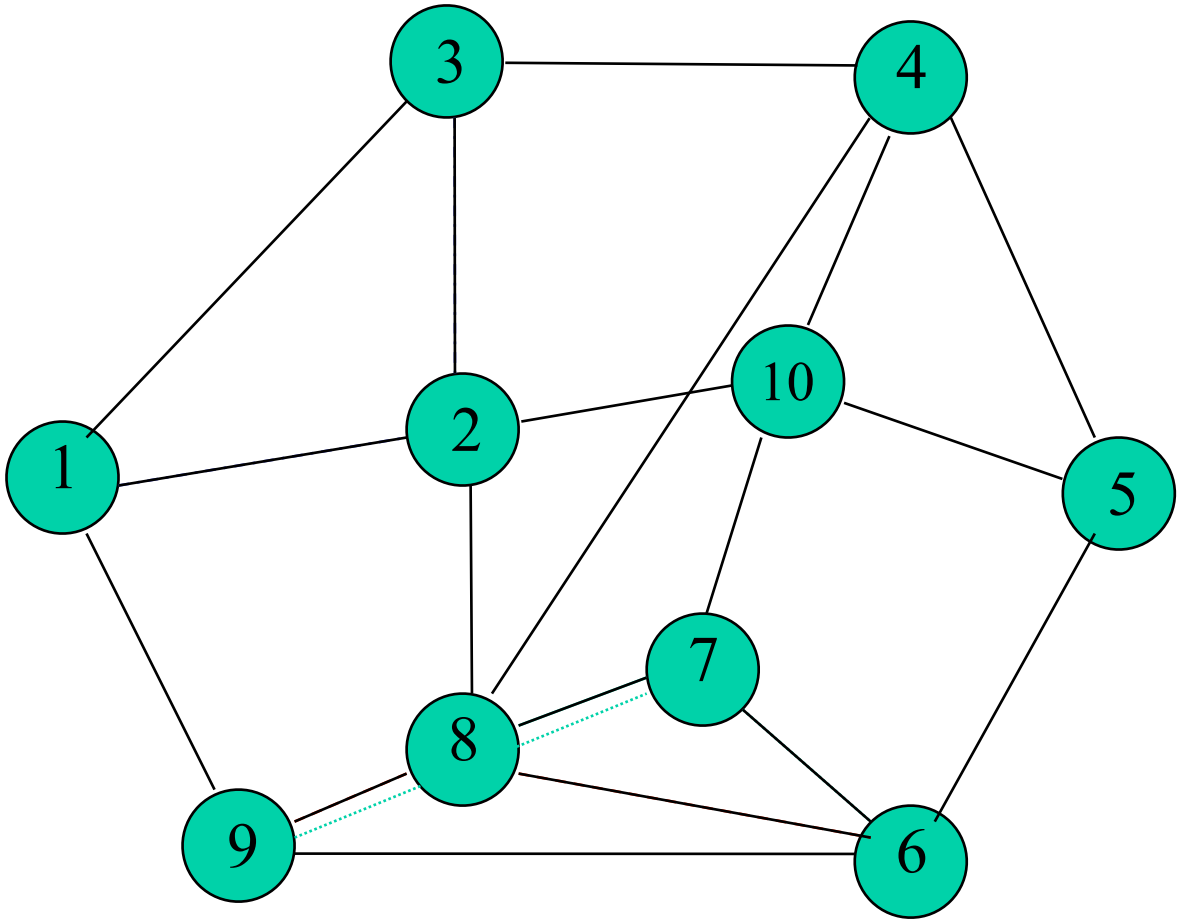
1. ↑ Goldberg, David E.. *Genetic Algorithms in Search, Optimization, and Machine Learning*. EUA: Addison-Wesley, 1989.
2. ↑ Para uma discussão sobre as formas de representação do espaço de busca, veja: Goldberg, David E.. *Genetic Algorithms in Search, Optimization, and Machine Learning*. EUA: Addison-Wesley, 1989. página 80
3. ↑ Veja em Goldberg, David E.. *Genetic Algorithms in Search, Optimization, and Machine Learning*. EUA: Addison-Wesley, 1989. página 121 uma comparação sobre diversas formas de seleção.
4. ↑ Veja em Goldberg, David E.. *Genetic Algorithms in Search, Optimization, and Machine Learning*. EUA: Addison-Wesley, 1989. página 147 para ver outras orperações que podem ser aplicadas nos indivíduos para a reprodução.
5. ↑ KOZA, J.R.. *Genetic Programming*: On the Programming of Computers by Means of Natural Selection. [S.l.]: MIT Press, 1992.
6. ↑ Poli, R., Langdon, W. B., McPhee, N. F.. *A Field Guide to Genetic Programming*. [S.l.]: freely available via Lulu.com, 2008

Referências 2

- ENCORE, the EvolutioNary COmputation REpository network
  Rede Repositório de Computação Evolucionária
  ftp://alife.santafe.edu/pub/USER-AREA/EC/ (tem também alguns outros nós)
- FAQ – The Hitch-Hiker's Guide to Evolutionary Computation
  O guia Hitch-Hiker sobre Computação Evolucionária
  ftp://alife.santafe.edu/pub/USER-AREA/EC/FAQ/www/index.html
- FAQ – Genetic programming
  Perguntas mais freqüentes sobre Programação Genética
  http://www-dept.cs.ucl.ac.uk/research/genprog/gp2faq/gp2faq.html
- The Genetic Algorithms Archive – many links, information about mailing list, some fun stuff
  Arquivo de Algoritmos Genéticos – vários links, informações sobre listas, e algum material divertido
  http://www.aic.nrl.navy.mil:80/galist/
- Artificial Life Online – Vida Artificial Online – links.
  Se você está procurando por material introdutório, veja aqui
  http://alife.santafe.edu/

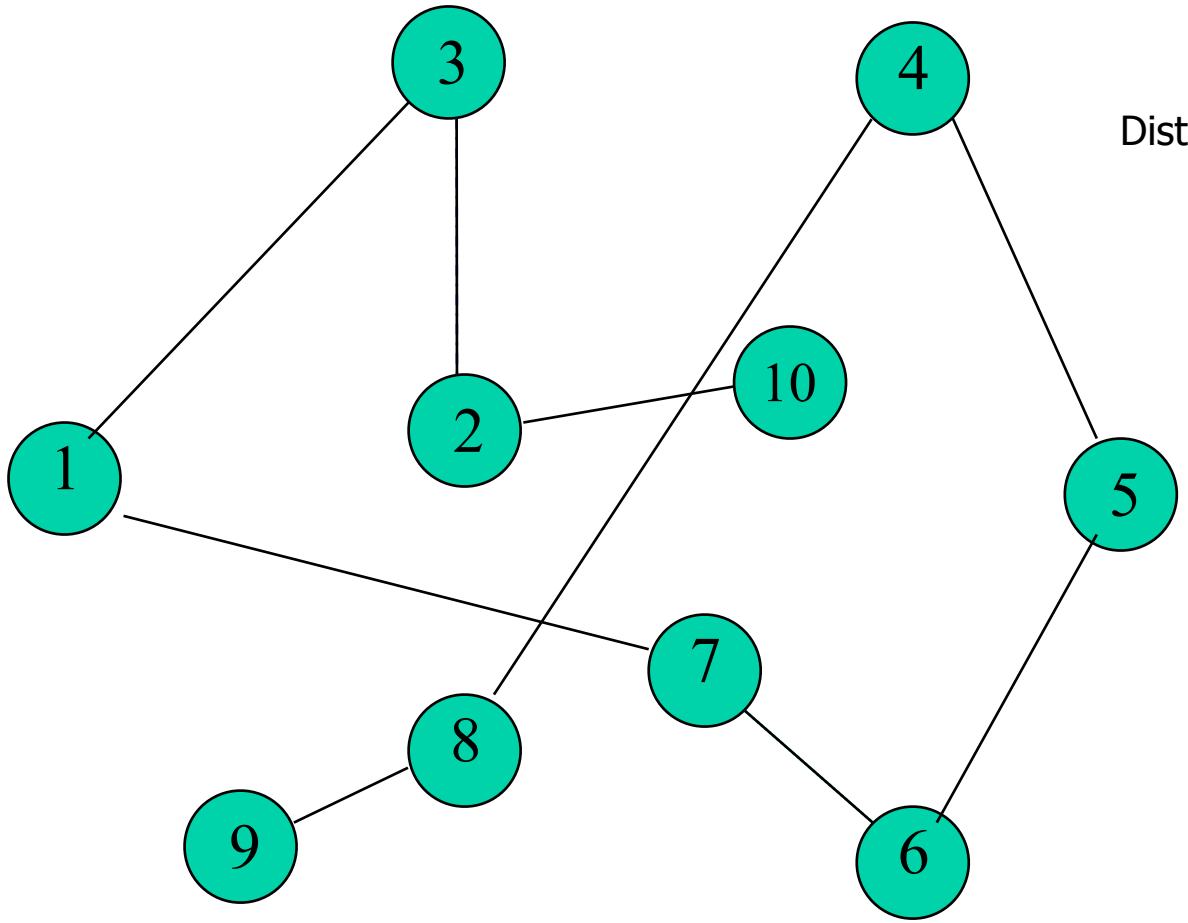
Caixeiro Viajante

Dado um conjunto de cidades, encontrar uma rota de menor custo, que passe por todas as cidades apenas uma vez



Caixeiro Viajante

- Cromossoma
- 9 8 4 5 6 7 1 3 2 10 — Representa uma trajetória



Função Objetivo:

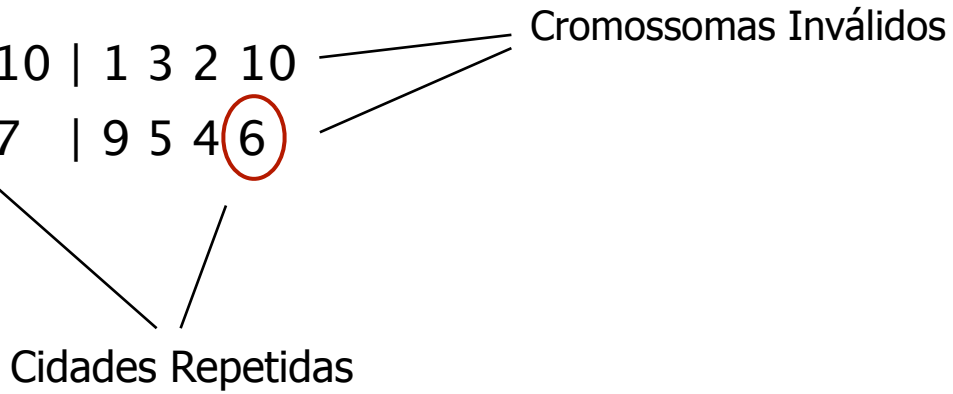
Distância Total Percorrida

Caixeiro Viajante

- Crossover de 2 pontos

- Pais:
- 9 8 4 | 5 6 7 | 1 3 2 10
- 8 7 1 | 2 3 10 | 9 5 4 6

- Filhos:
- 9 8 4 | 2 3 10 | 1 3 2 10
- 8 7 1 | 5 6 7 | 9 5 4 6

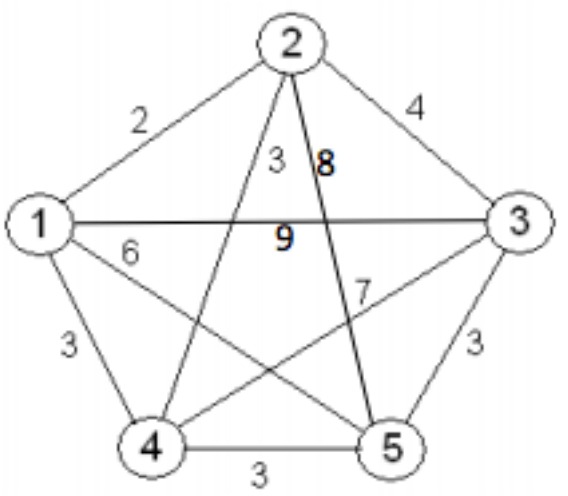


Caixeiro Viajante – Uma Solução

- Fazer o crossover normalmente entre os dois pontos
- Trocado um vértice do pai 1 por um vértice do pai 2 para cada posição entre os pontos, cada uma destas trocas define um mapeamento. Se o vértice i estava na posição k no pai 1, o vértice j estava na posição k no pai 2, e a posição k está entre os dois pontos de corte, i está mapeado em j e viceversa;
- Após crossover normal, provavelmente serão geradas rotas inválidas;
- Para corrigir substitui-se todos os vértices repetidos que estão fora dos pontos de corte pelos vértices aos quais eles estão mapeados;
- Pais:
- 9 8 4 | 5 6 7 | 1 3 2 10
- 8 7 1 | 2 3 10 | 9 5 4 6
- Operador troca a cadeia de números entre os dois pontos de corte, e repete as cidades dos pais que não são repetidas:
- 9 8 4 | 2 3 10 | 1 x x x
- 8 x 1 | 5 6 7 | 9 x 4 x
- A troca entre os pontos de crossover define alguns mapeamentos:
- 2 ↔ 5, 3 ↔ 6, 10 ↔ 7
- As cidades repetidas dos cromossomas originais são substituídas pelas cidades mapeadas:
- 9 8 4 | 2 3 10 | 1 6 5 7
- 8 10 1 | 5 6 7 | 9 2 4 3

Exercício- Caixeiro Viajante

Dado um conjunto de cidades, encontrar uma rota de menor custo, que passe por todas as cidades apenas uma vez



- O grafo mostra a ligação entre 5 cidades e as respectivas distâncias em quilômetros:
- O objetivo é encontrar uma rota de menor custo usando um algoritmo genético
- a) Proponha uma maneira de codificar os cromossomos.
- b) Defina uma função de aptidão para avaliar a qualidade dos cromossomos
- c) Gere dois cromossomos e avalie a aptidão deles
- d) Realize o cruzamento entre os cromossomos
- e) Aplique uma mutação em um gene dos cromossomos
- f) Aplique a função de aptidão nos descendentes gerados verificando se a solução encontrada é melhor ou não

Problema 2

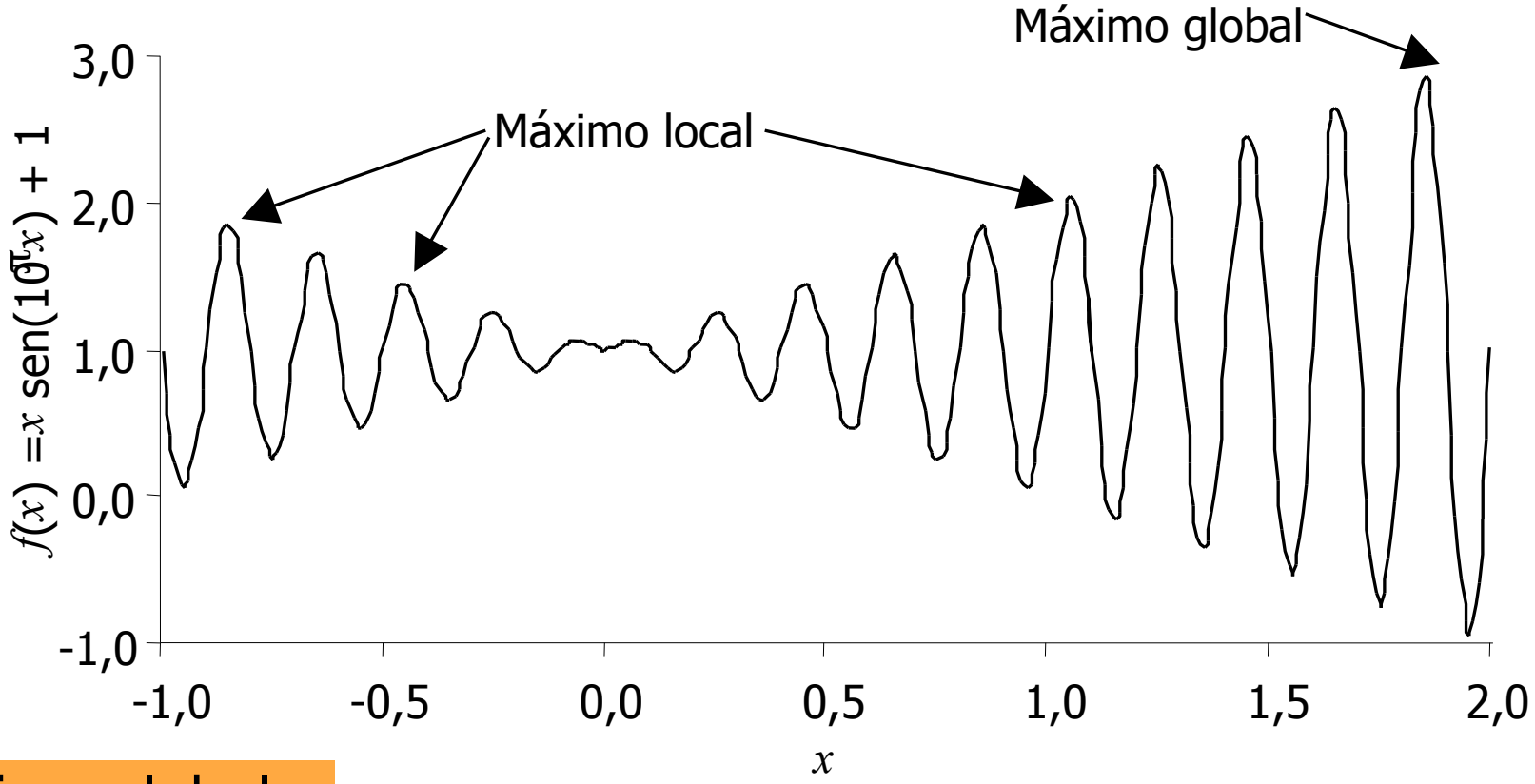
Achar o máximo da função utilizando Algoritmos Genéticos,

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

Restrita ao intervalo:

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

Problema 2 (II)



Máximo global:

x = 1,85055

f(x) = 2,85027

Problema 2 (III)

- Função multimodal com vários pontos de máximo.
- É um problema de otimização global (encontrar o máximo global)
- Não pode ser resolvido pela grande maioria dos métodos de otimização convencional.
- Há muitos métodos de otimização local, mas para otimização global são poucos.

Cromossomo do Problema 2

- Representar o único parâmetro (variável *x*) na forma de um cromossomo:
  - Quantos bits deverá ter o cromossomo?
  - Quanto mais bits melhor precisão númerica
  - Longos cromossomos são difíceis de manipular
  - Para cada decimal é necessário 3,3 bits
  - Cromossomo com 22 bits

1000101110110101000111

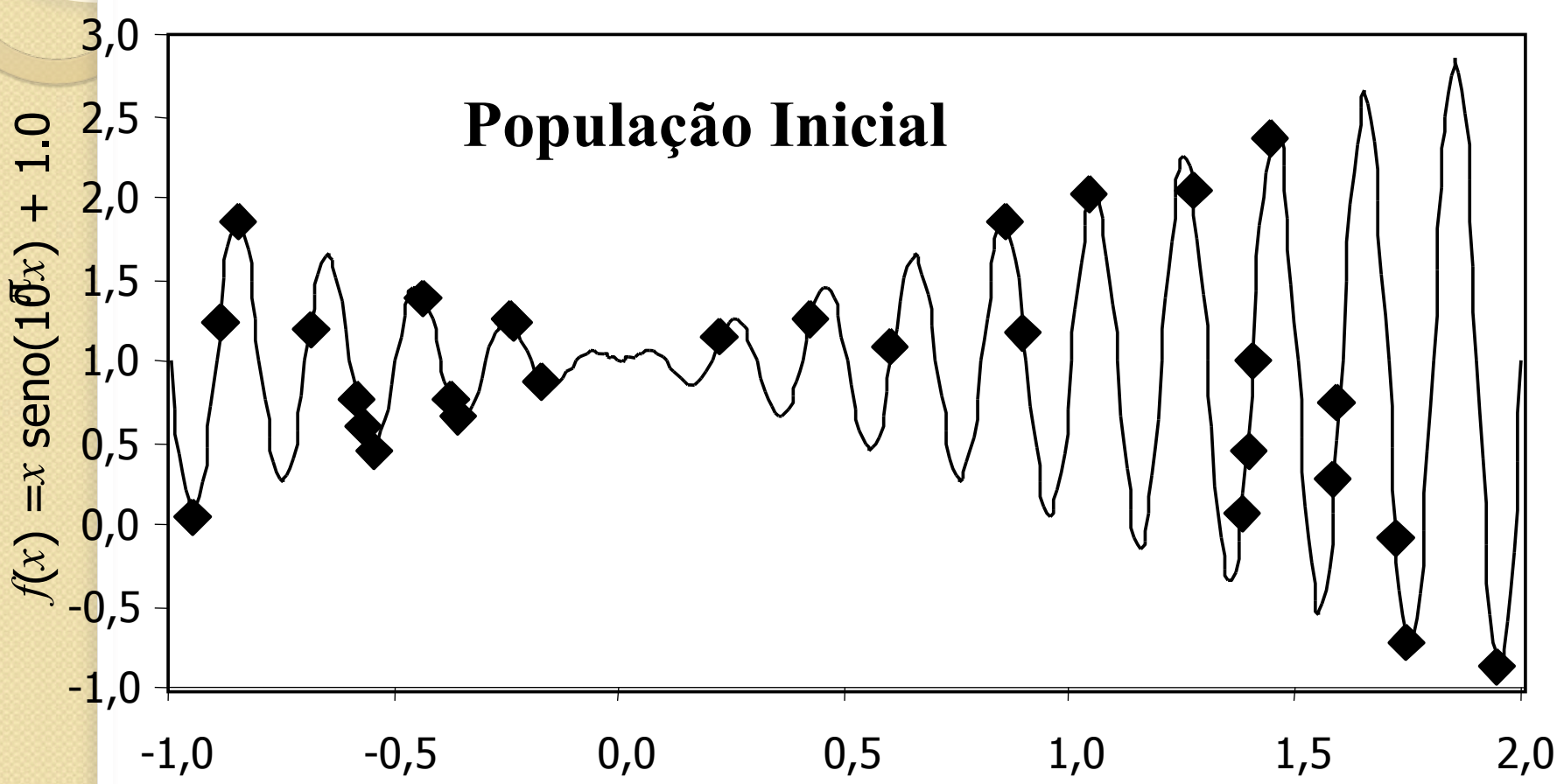
O Cromossomo Problema 2 (II)

- Decodificação
  - cromossomo = 10001011101101010000111
  - $b_{10} = (10001011101101010000111)_2 = 2288967$
  - Valor de x precisa estar no intervalo [-1,0; 2,0]

$$x = \min + (\max - \min)\frac{b_{10}}{2^l - 1}$$

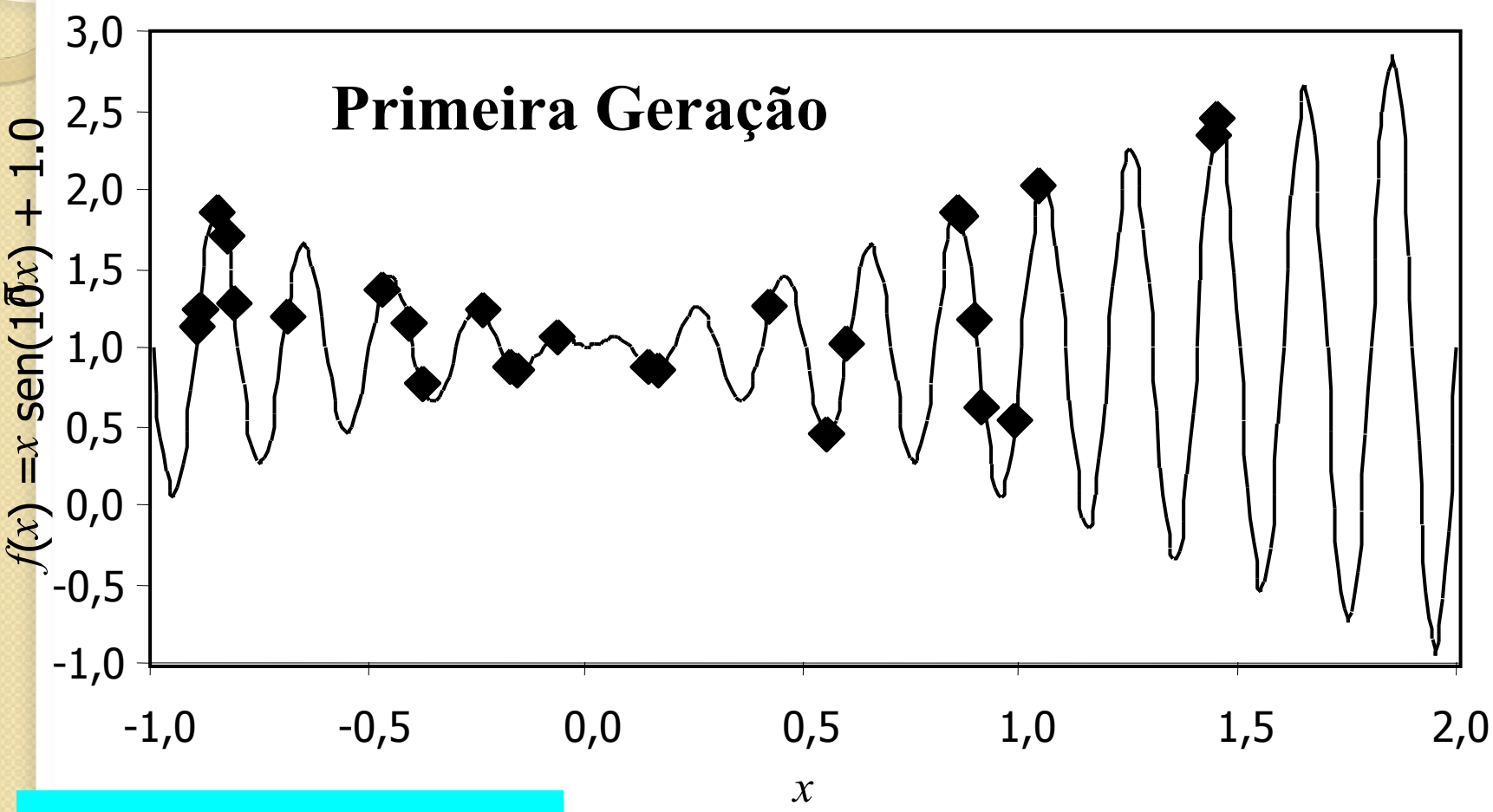
$$x = -1 + (2+1)\frac{2.288.967}{2^{22} - 1} = 0{,}637197$$

As Gerações do Problema 2



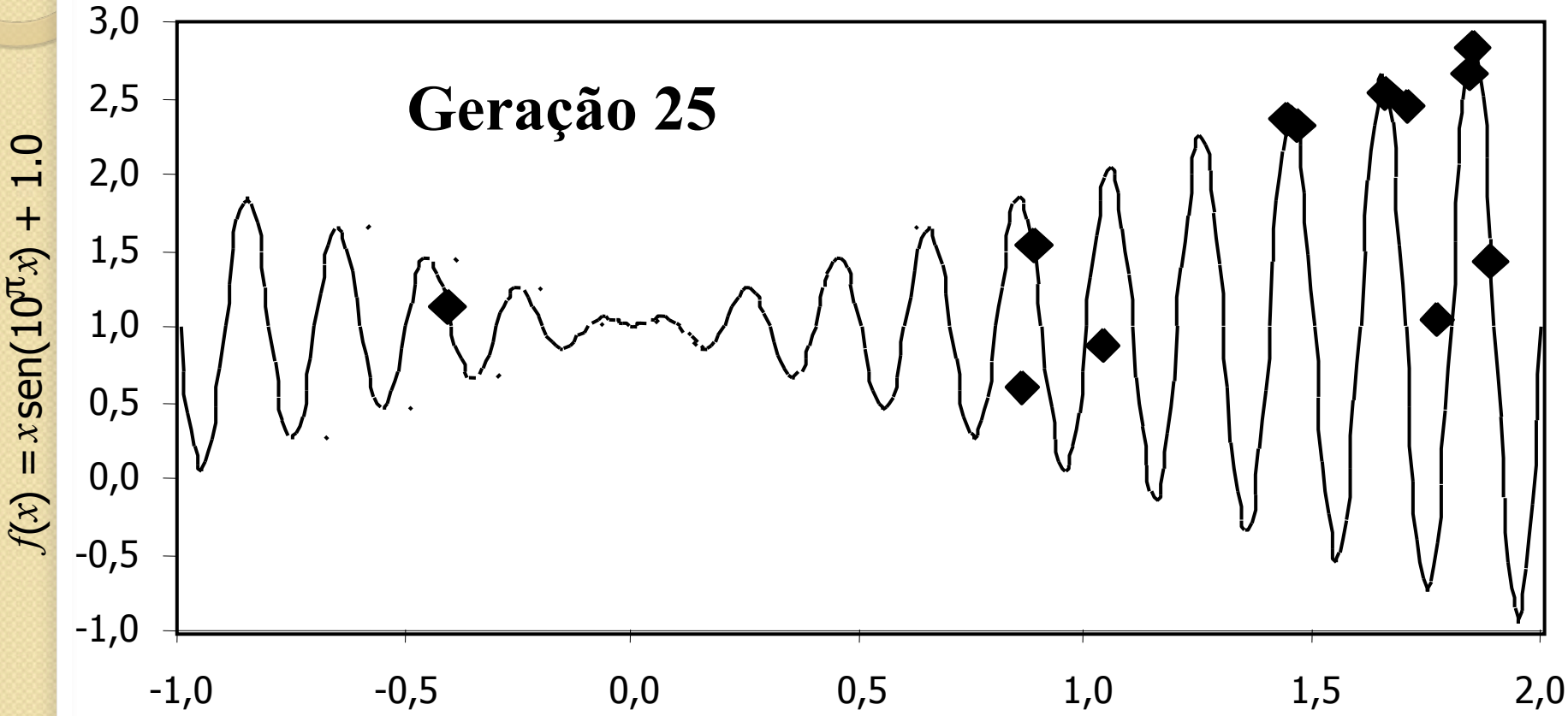
População gerada aleatóriamente

As Gerações do Problema 2 (II)



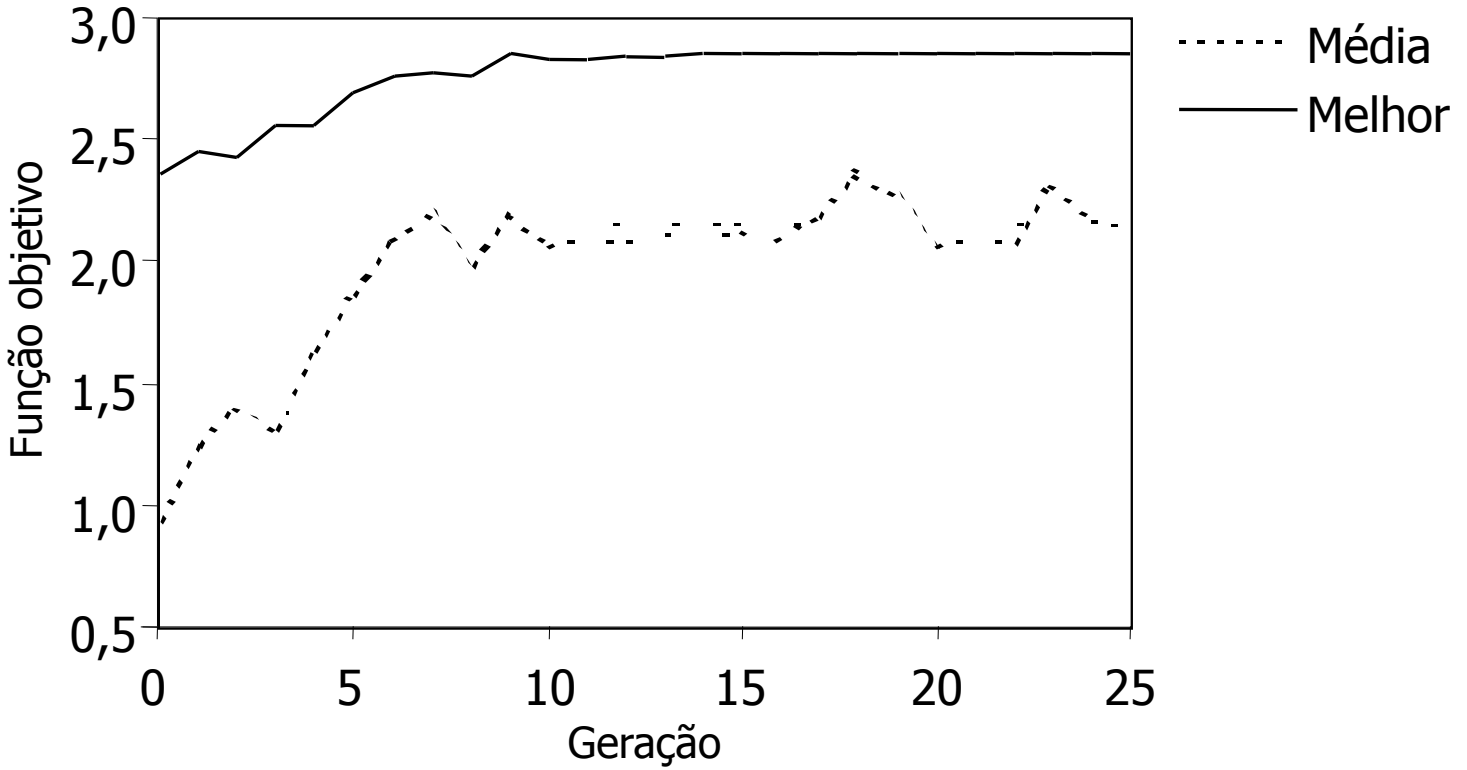
Pouca melhoria

As Gerações do Problema 2 (III)



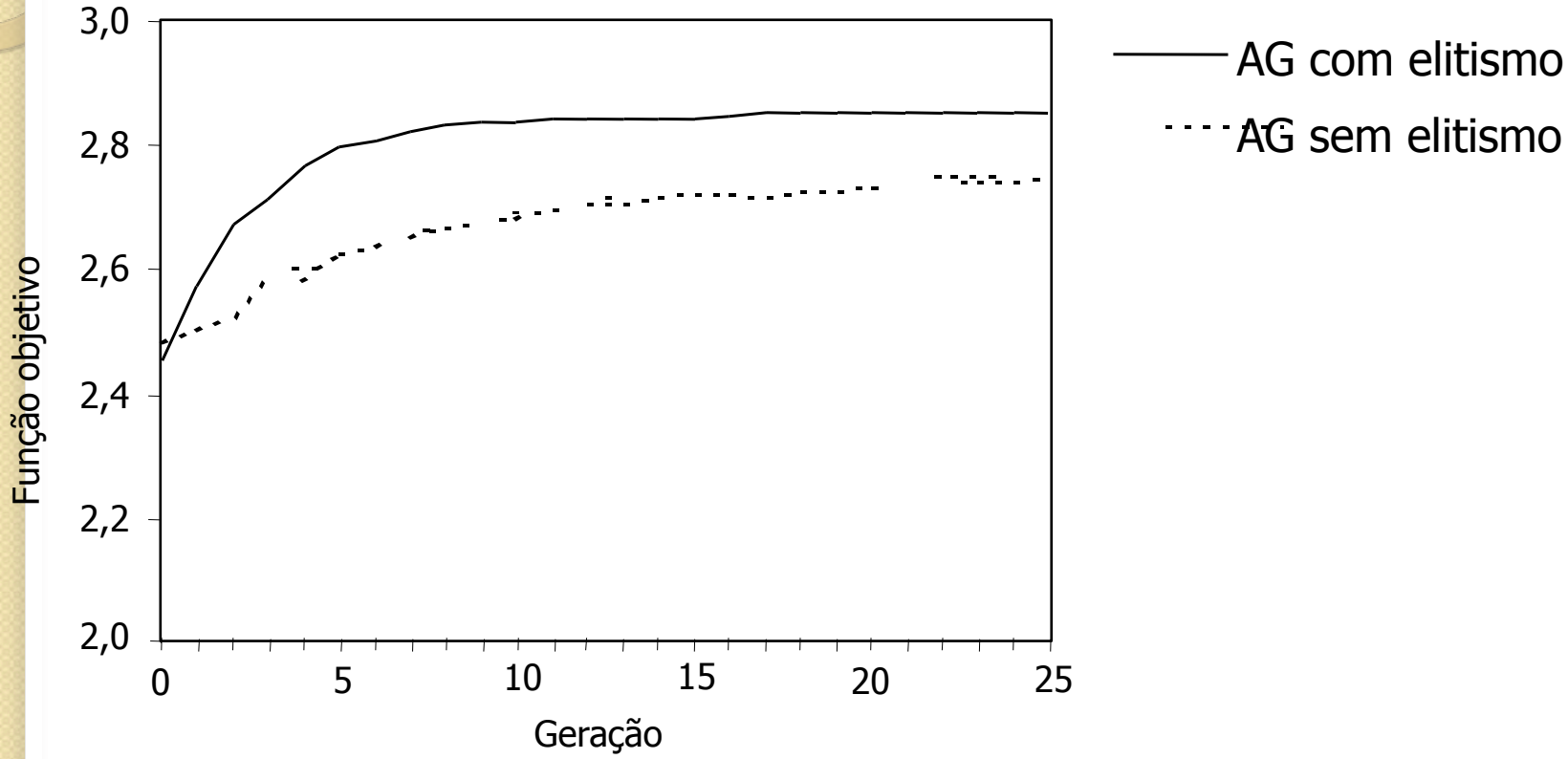
A maioria dos indivíduos encontraram o máximo global

As Gerações do Problema 2 (IV)



Na geração 15 o AG já encontrou o ponto máximo

Elitismo no Problema 2



AG com elitismo é melhor ?