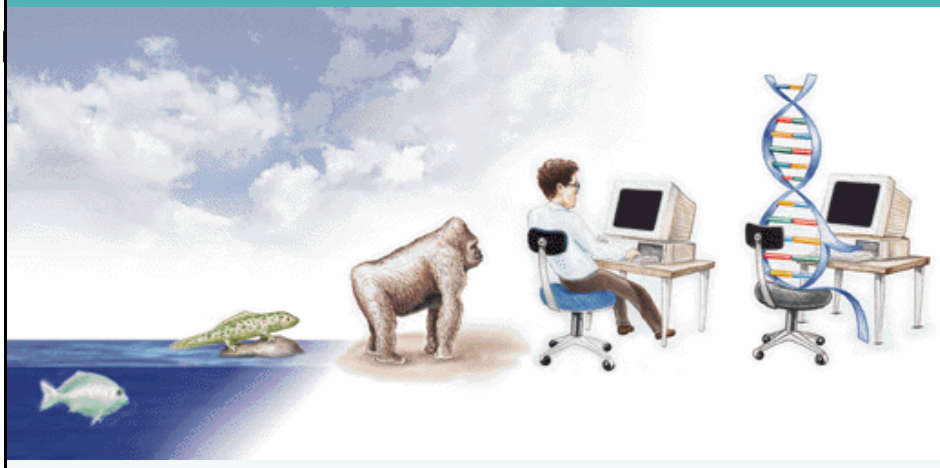
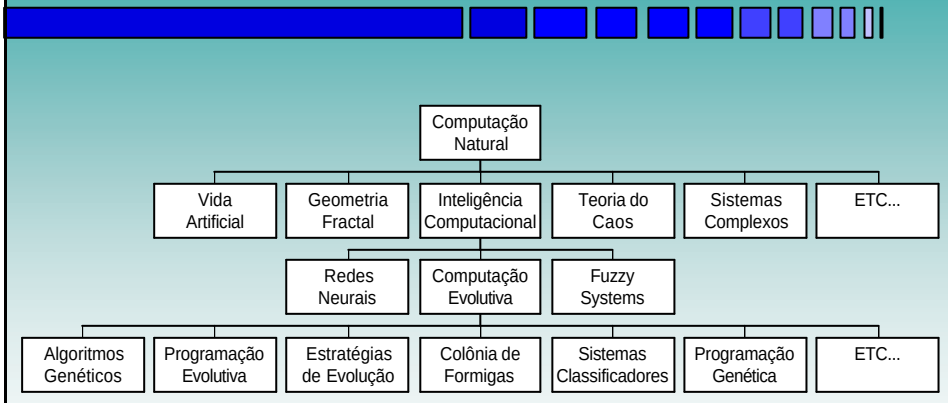
Algoritmos Genéticos



Importante: Apresentação produzida a partir de material disponível na Internet.

Computação Natural

(Sistemas computacionais baseados em princípios ou fenômenos da natureza)



- Nenhuma imposição de continuidade, convexidade e diferenciabilidade.

Visão Geral do Algoritmo Evolucionário

Algoritmos Evolucionários

- Noção darwiniana de evolução
- Elementos básicos:
  - População de indivíduos
  - Noção de desempenho (fitness)
  - Ciclo de nascimento/morte pelo desempenho
  - Noção de hereditariedade

Algoritmos Genéticos

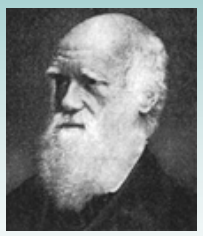
Algoritmos Genéticos

Apresentação:

- O que são ?
- Para que servem ?
- Características
- Aplicações
- Conceitos básicos
- Representação
- Função de aptidão
- Operadores fundamentais
- Parâmetros genéticos
- Exemplos

Teoria da Evolução

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



Charles Darwin

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

Teoria da Evolução

"Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes."

(DARWIN, 1859)

Algoritmos Genéticos – Breve Histórico

- Charles Darwin apresentou em 1858 sua teoria de evolução através de seleção natural;
- Por volta de 1900, a moderna teoria da evolução combina a genética e as idéias de Darwin e Wallace sobre a seleção natural, criando o princípio básico de Genética Populacional;
- Este princípio foi desenvolvido durante os anos 30 e 40, por biólogos e matemáticos;
- Nos anos 50 e 60, muitos biólogos começaram a desenvolver simulações computacionais de sistemas genéticos;
- Holland foi gradualmente refinando suas idéias e em 1975 publicou o seu livro *"Adaptation in Natural and Artificial Systems"*, hoje considerado a Bíblia de Algoritmos Genéticos;
- Nos anos 80 ***David Goldberg***, aluno de ***Holland***, consegue primeiro sucesso em aplicação industrial de Algoritmos Genéticos.

O que são?

- Os Algoritmos Genéticos são uma classe de procedimentos, com passos distintos bem definidos.
- Cada passo distinto pode ter diversas versões diferentes.

O que são?

- São algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética;
- Estudam a adaptação como ocorre na natureza;
- Evoluem a partir de uma população de cromossomos, através de operadores genéticos (seleção, crossover e mutação);
- Apesar de aleatórios, eles não são caminhadas aleatórias não direcionadas, pois exploram informações históricas para encontrar novos pontos de busca onde são esperados melhores desempenhos.

Para que servem?

- Busca e Otimização
- Amplamente utilizados, com sucesso, em problemas de difícil manipulação pelas técnicas tradicionais
- Eficiência X Flexibilidade

Características

- Utilizam uma codificação do conjunto de parâmetros (*indivíduos*) e não com os próprios parâmetros (*estados*);
- Vasculham várias regiões do espaço de busca de cada vez;
- Utilizam informações diretas de qualidade, em contraste com as derivadas utilizadas nos métodos tradicionais de otimização;
- Utilizam regras de transição probabilísticas e não regras determinísticas.

Características

- É um algoritmo estocástico (não determinístico).
- Trabalha com uma população de soluções simultaneamente.
- Utiliza apenas informações de custo e recompensa. Não requer nenhuma outra informação auxiliar (como por exemplo o gradiente).

Características

- São fáceis de serem implementados em computadores.
- São facilmente hibridizados com outras técnicas.
- Funcionam com parâmetros contínuos ou discretos.

Características

- São especialmente interessantes em problemas difíceis de otimizar de forma convencional
- Técnicas tradicionais são mais difíceis de empregar
- Se uma técnica tradicional puder ser empregada, normalmente acha melhor solução mais rápido

Características

- Existem muitos problemas práticos aos quais técnicas determinísticas tradicionais não podem ser aplicadas
- Técnicas tradicionais têm natureza serial
- Algoritmos Genéticos têm natureza paralela

Aplicações

- Em problemas díficeis de otimização, quando não existe nenhuma outra técnica especifica para resolver o problema.
- Otimização de funções numéricas em geral
- Otimização combinatória
  - Problema do caixeiro viajante
  - Problema de corte e empacotamento
  - Alocação de recursos (*job shop schedulling*)
- Robótica (Planejamento de trajetórias)

Aplicações

- Alocação de tarefas
- Configuração de sistemas complexos
- Roteirização de veículos
- Problemas de Otimização e de Aprendizagem de Máquina
- Perfuração de placas de circuitos
- Problemas de Localização (postos de saúde, telefonia celular, etc.)
- Roteamento de Telecomunicações

Aplicações

- Síntese de circuitos analógicos: para uma certa entrada e uma saída desejada, por exemplo tensão, o AG gera a topologia , o tipo e o valor dos componentes do circuito.
- Síntese de protocolos: determinação de quais funções do protocolo devem ser implementadas em hardware e quais devem ser implementadas em software para que um certo desempenho seja alcançado.
- Programação Genética: gera a listagem de um programa, numa determinada linguagem especificada, para que um determinado conjunto de dados de entrada forneça uma saída desejada.
- Gerenciamento de redes: supervisão do tráfego nos links e das filas nos "buffers" de roteadores para descobrir rotas ótimas e para reconfigurar as rotas existentes no caso de falha de algum link.
- Computação Evolutiva: gera programas que se adaptam a mudanças no sistema ao longo do tempo.

Aplicações

- Otimização evolutiva multi-critério: otimização de funções com múltiplos objetivos que sejam conflitantes.
- Problemas de otimização complexos: problemas com muitas variáveis e espaços de soluções de dimensões elevadas. Ex: problema do caixeiro viajante, gerenciamento de carteiras de fundos de investimento.
- Ciências biológicas: modela processos biológicos para o entendimento do comportamento de estruturas genéticas.
- Autômatos auto-programáveis.

Aplicações

- Na Evolução Musical
  - Foi apresentado em 1999 na *CEC99 – IEEE – International Conference on Evolutionary Computation* um ambiente interativo, utilizando Algoritmos Genéticos, para a avaliação de músicas (seqüências de acordes) tocadas em arquivos MIDI. O método emprega o formalismo difuso e é colocado como uma otimização baseada em fatores relevantes à audição de músicas.

Aplicações

- Melhorias em Telecomunicações
  - Blanchard (1994), apresenta o caso da *US West*, uma companhia regional de telecomunicações do estado do Colorado, que vem usando um sistema baseado em AGs que possibilita projetar, em duas horas, redes óticas especializadas, trabalho que levaria seis meses utilizando especialistas humanos. O sistema produz resultados ainda 10% (dez por cento) melhores que os realizados pelo homem. (estimativa de economia de 100 milhões de dólares até o final do século)

Aplicações

- Otimização de Plantão Médico Hospitalar
  - No Hospital universitário da UFSC, em Florianópolis, os Algoritmos Genéticos foram utilizados para auxiliar na elaboração de uma escala de trabalho dos médicos plantonistas neonatalogistas da maternidade. O objetivo pretendido foi o de auxiliar na solução da escala de trabalho dos médicos, em como diminuir o esforço e o desgaste humanos para a confecção do plantão. O problema resumia-se na disponibilidade de 12 (doze) médicos e na necessidade de atendimento 24 (vinte e quatro) horas por dia, tendo-se como variáveis envolvidas o número de médicos contratados e o turno com número adaptável de horas.

Aplicações

- A *AIS* (Barcelona, Espanha) utilizou um sistema apoiado em AGs e Sistemas Especialistas (SEs) para programar os Jogos Paraolímpicos de 1992 já que nas Olimpíadas os atletas são organizados em duas grandes classes, masculino e feminino, e os competidores paraolímpicos são divididos em mais de 100 (cem) classes, segundo certas restrições médicas.
- Um sistema em construção na *New Mexico State University* descreve imagens faciais de criminosos a partir de testemunhas do crime, utilizando AGs.

Aplicações

- Sponsler (1989) mostrou um sistema protótipo desenvolvido para avaliar a aplicabilidade dos Algoritmos Genéticos na otimização da programação do telescópio espacial *Hubble*. Diversos operadores genéticos foram avaliados e o melhor AG foi comparado com um otimizador baseado em Redes Neuronais (RN). Neste caso específico os AGs não se apresentaram tão eficientes quanto as RNs.
- Syswerda e Palmucci (1991) relataram a execução de um otimizador para uma aplicação prática de programação de recursos no laboratório *SITS – System Integration Test Station Laboratory* – da Marinha Americana, para o desenvolvimento do jato F-14.

Conceitos Básicos

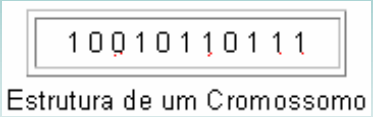
- AG manipula uma população de indivíduos.
- Individuos são possíveis soluções do problema.
- Os indivíduos são combinados (crossover) uns com os outros, produzindo filhos que podem sofrer ou não mutação.
- As populações evoluem através de sucessivas gerações até encontrar a solução ótima.

Algoritmos Genéticos - Conceitos

cromossomo (genótipo) - cadeia de bits que representa uma solução possível para o problema.
gene - representação de cada parâmetro de acordo com o alfabeto utilizado (binário, inteiro ou real).
fenótipo - cromossomo codificado
população - conjunto de pontos (indivíduos) no Espaço de Busca
geração - iteração completa do AG que gera uma nova população
aptidão - saída gerada pela função objetivo para um indivíduo da população

Algoritmos Genéticos - Representação

- Em sua forma original trabalham normalmente com uma representação binária (zero-um).



Estrutura de um Cromossomo

- Um cromossomo é formado por um conjunto de genes. Cada gene ou uma combinação de genes representa ou controla uma característica específica de um indivíduo (solução).

Algoritmos Genéticos - Representação

- A forma de representação pode ser posicional ou não posicional.

POSICIONAL	
CROMOSSOMO	FENÓTIPO
1 0 1 0 1 1 0 0 0 1 1	1379
1 0 1 0 1 1 0 0 **1** 1 1	1381

Representação Posicional

NÃO POSICIONAL	
CROMOSSOMO	FENÓTIPO
1 0 0 1 1 1 1 0 1 0 1	7
1 0 **1** 0 1 1 0 0 1 1 1	8
1 0 1 **0** 1 1 0 **1** 1 1 1	7

Representação Não Posicional

Algoritmos Genéticos - Função de avaliação

- Cada indivíduo de uma população tem associado a si um valor de aptidão, identificando seu grau de adaptabilidade ao ambiente.
- A determinação do valor de aptidão de cada indivíduo é obtido através da função de avaliação.
- A função de avaliação mostra o quanto uma solução é boa se comparada com outra solução da população.

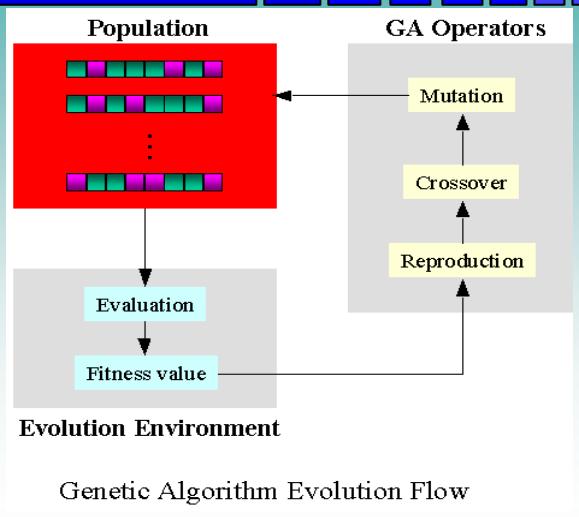
Algoritmos Genéticos - Operadores

- Para poder evoluir sucessivas populações que consigam melhorar a aptidão de seus indivíduos após gerações, são necessários os operadores genéticos [GOL89].
- Esses operadores permitem que a nova geração seja parcial ou completamente nova, possuindo as características de seus antecessores, diversificando a população e mantendo as características de adaptação adquiridas pelas gerações passadas.
- Operadores: Seleção, Crossover (Cruzamento, Reprodução) e Mutação.

Operadores Fundamentais

- Seleção Natural
- Manipulação Genética por Reprodução
- Manipulação Genética por Mutação

Algoritmos Genéticos - Animação



Genetic Algorithm Evolution Flow

Algoritmos Genéticos - Operadores

- Seleção - Responsável pelo processo de seleção dos indivíduos mais adaptados ao ambiente.

Formas de inserção de indivíduos numa nova população

A maioria dos métodos de seleção são projetados para escolher preferencialmente indivíduos com maiores notas de aptidão.

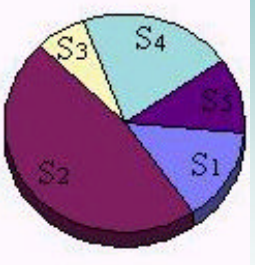
Alguns métodos

- Roleta; Torneio; Elitista.

Algoritmos Genéticos - Operadores

- Roleta

Indivíduo S_i	Aptidão $f(S_i)$	Aptidão Relativa
S_1 10110	2.23	0.14
S_2 11000	7.27	0.47
S_3 11110	1.05	0.07
S_4 01001	3.35	0.21
S_5 00110	1.69	0.11



Algoritmos Genéticos - Operadores

- Torneio

```
Inicio
    k = 0.75
    Repita N vezes
        Escolha 2 indivíduos da população aleatoriamente
        r = valor aleatório entre 0 e 1
        Se r < k
            O melhor indivíduo é escolhido
        Senão
            O pior indivíduo é escolhido
        Fim se
    Fim Repita
Fim
```

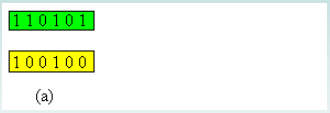
Algoritmos Genéticos - Operadores

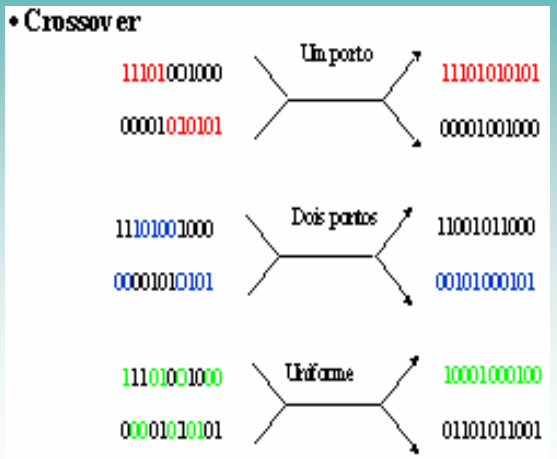
- Elitista

- previne que os melhores indivíduos não desapareçam da população pela manipulação dos operadores genéticos.
- É o método mais utilizado para melhorar a convergência dos AGs, pois através dele ocorre um aumento na velocidade de dominação da população por indivíduos com elevado valor de aptidão [SER96].

Algoritmos Genéticos - Operadores

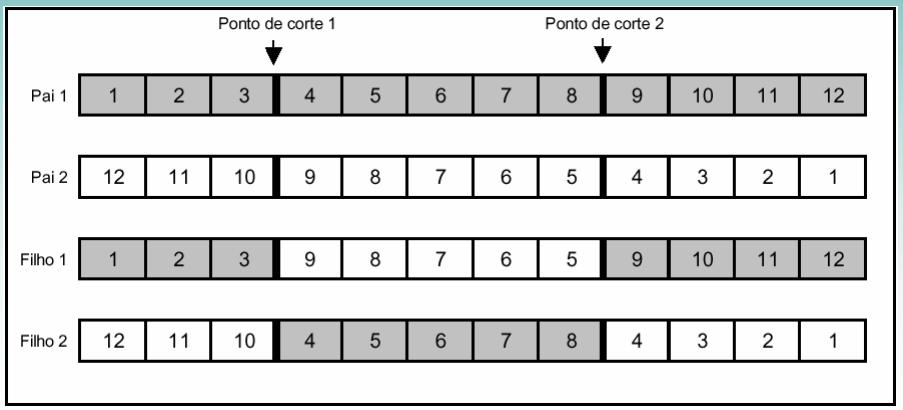
- Responsável pelo acasalamento e geração de novos descendentes.





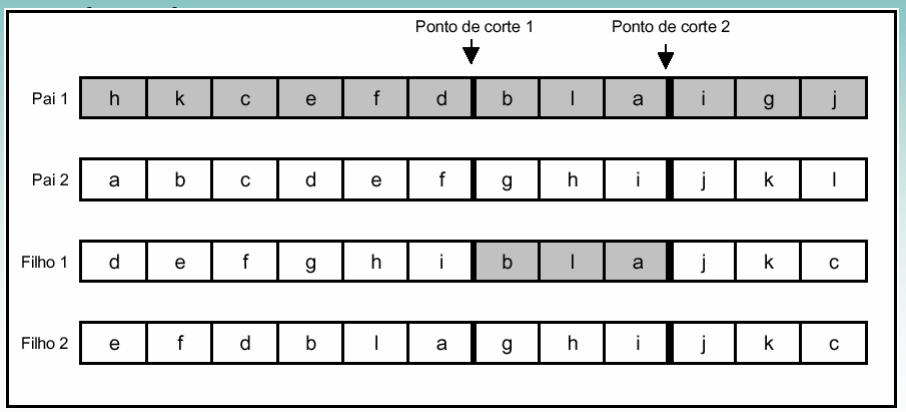
Algoritmos Genéticos - Operadores

- cruzamento aplicado a cromossomos com representação através de números inteiros.



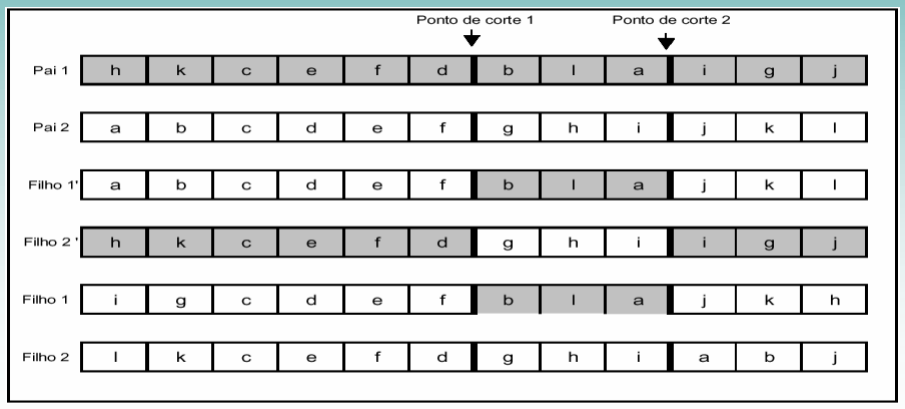
Algoritmos Genéticos - Operadores

Operador OX *(Order Crossover)*: este operador começa pela escolha aleatória de dois pontos de corte em cada um dos elementos



Algoritmos Genéticos - Operadores

Operador PMX *(Partially Mapped Crossover)*: este operador também é executado escolhendo-se aleatoriamente dois pontos de corte.

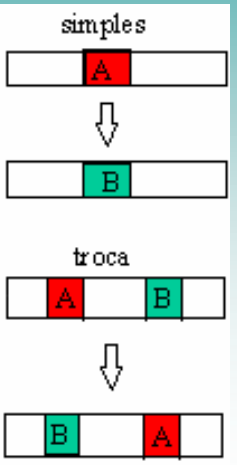


Algoritmos Genéticos - Operadores

- Responsável pela manutenção e introdução da diversidade genérica.

Antes da Mutação: 1 1 1 0 0

- Mutação



Parâmetros Genéticos

- Tamanho da população
- Taxa de cruzamento
- Taxa de mutação
- Intervalo de geração
- Critério de parada

Algoritmos Genéticos - Parâmetros

- Principais parâmetros:
  - Tamanho da população
    - pequena (queda de desempenho, pequena cobertura no espaço de busca, convergência prematura)
    - Grande (previne convergência prematura, grande esforço computacional)
  - Taxa de cruzamento
    - Elevada (a maior parte da população será substituída, pode ocorrer que indivíduos com alto valor de aptidão sejam retirados mais rapidamente da população)
    - Baixa (o algoritmo pode ficar muito lento devido a pouca geração de novos indivíduos)
    - Recomendado: taxa de cruzamento varia entre 20 a 60% ????

Algoritmos Genéticos - Parâmetros

- Principais parâmetros:
  - Taxa de mutação
    - Baixa (previne que a população fique estagnada em uma certa posição num determinado valor)
    - Elevada (a busca torna-se essencialmente aleatória)
    - Recomendado: taxa em torno de 0,1 a 5% ??????
  - Intervalo de geração (controla a percentagem de população que será substituída na próxima geração)
    - Elevado (uma grande parte da população será substituída, pode ocorrer perda de indivíduos com um alto valor de aptidão)
    - Baixo (a população será pouco substituída e o algoritmo levará muito tempo para convergir)

Problema 1

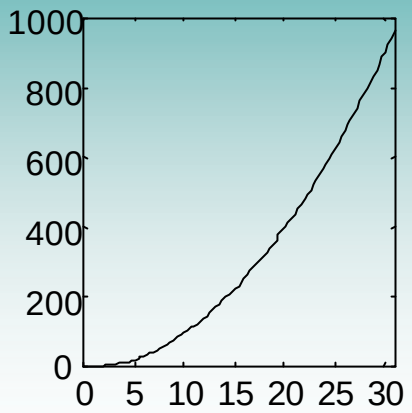
Problema: Use um AG para encontrar o ponto máximo da função:

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

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, a aptidão pode ser a própria função objetivo.
  - Exemplo:

aptidão(00011) = *f*(3) = 9

População Inicial do Problema 1

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

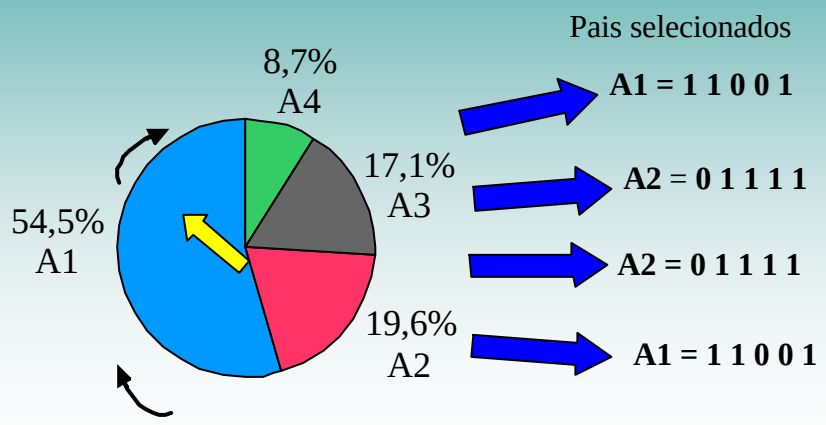
Pop. inicial

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

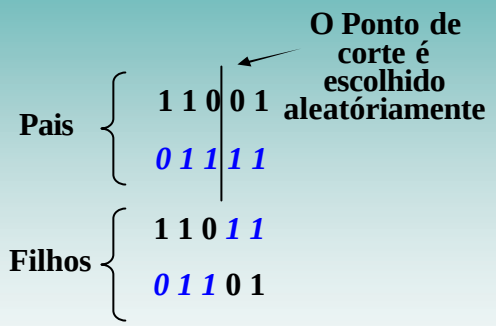
$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}$$

Seleção proporcional a aptidão (Roleta)



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.

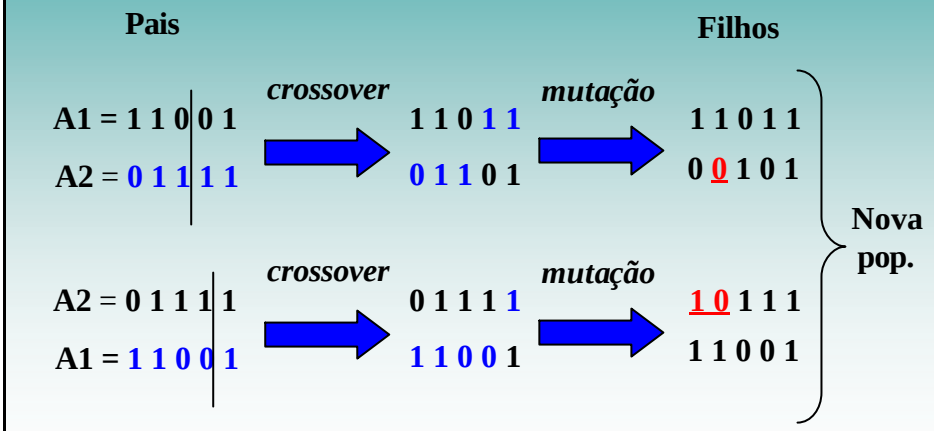
A mutação é aplicada com dada probabilidade, denominada *taxa de mutação* (~1%), em cada um dos bits do cromossomo.

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 não deve ser nem alta nem baixa, mas o suficiente para assegurar a diversidade de cromossomos na população.

A primeira geração do Problema 1



A primeira geração do Problema 1

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

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

A Simple Example (Problema 2)

The Traveling Salesman Problem:

Find a tour of a given set of cities so that

- each city is visited only once
- the total distance traveled is minimized

A Simple Example (Problema 2)

The Traveling Salesman Problem:

Suponha que um caixeiro viajante tenha de visitar **n** cidades diferentes, iniciando e encerrando sua viagem na primeira cidade. Suponha, também, que não importa a ordem com que as cidades são visitadas e que de cada uma delas pode-se ir diretamente a qualquer outra.

O problema do caixeiro viajante consiste em descobrir a rota que torna mínima a viagem total.

A Simple Example (Problema 2)

Complexidade computacional do problema do caixeiro:

O problema do caixeiro é um clássico exemplo de problema de otimização combinatória. Para acharmos o número **R(n)** de rotas para o caso de **n** cidades, basta fazer um raciocínio combinatório simples e clássico. Por exemplo, no caso de n = 4 cidades, a primeira e última posição são fixas, de modo que elas não afetam o cálculo; na segunda posição podemos colocar qualquer uma das 3 cidades restantes B, C e D, e uma vez escolhida uma delas, podemos colocar qualquer uma das 2 restantes na terceira posição; na quarta posição não teríamos nenhuma escolha, pois sobrou apenas uma cidade; consequentemente, o número de rotas é 3 x 2 x 1 = 6.

A Simple Example (Problema 2)

De modo semelhante, para o caso de n cidades, como a primeira é fixa, verifica-se que o número total de escolhas que podemos fazer é (n-1) x (n-2) x ... x 2 x 1. De modo que, usando a notação de fatorial: **R(n) = (n - 1)!**. Assim, a estratégia consiste em gerar cada uma dessas R(n) = (n - 1)! rotas, calcular o comprimento total das viagens de cada rota e ver qual delas tem o menor comprimento total. Trabalho fácil para o computador, diria alguém. Bem, talvez não.

A Simple Example (Problema 2)

Suponhamos um computador veloz, capaz de fazer 1 bilhão (10^9) de adições por segundo. Isso parece uma velocidade imensa, capaz de tudo. Com efeito, no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota e então será capaz de calcular (10^9) / 19 = 53 milhões de rotas por segundo. Contudo, essa imensa velocidade é um nada frente à imensidão do número 19! de rotas que precisará examinar. Com efeito, o valor de 19! é 121 645 100 408 832 000 (ou, aproximadamente, 1.2x10^17 em notação científica). Consequentemente, ele precisará de 1.2 x 10^17 / (53 milhões) = 2.3 x 10^9 segundos para completar sua tarefa, o que equivale a cerca de **73 anos.**

A Simple Example (Problema 2)

O problema é que a quantidade (n - 1)! cresce com uma velocidade alarmante, sendo que muito rapidamente o computador torna-se incapaz de executar o que lhe pedimos. Constate isso mais claramente na tabela a seguir:

n	rotas por segundo	(n - 1)!	cálculo total
5	250 milhoes	24	insignific
10	110 milhoes	362 880	0.003 seg
15	71 milhoes	87 bilhoes	20 min
20	53 milhoes	1.2×10^{17}	73 anos
25	42 milhoes	6.2×10^{23}	470 milhoes de anos

A Simple Example (Problema 2)

Observe que o aumento no valor do **n** provoca uma muito lenta diminuição na velocidade com que o computador calcula o tempo de cada rota (ela diminui apenas de um sexto ao **n** aumentar de 5 para 25), mas provoca um imensamente grande aumento no tempo total de cálculo. Em outras palavras: a inviabilidade computacional é devida à presença da fatorial na medida do esforço computacional do método da redução. Com efeito, se essa complexidade fosse expressa em termos de um polinômio em **n** o nosso computador seria perfeitamente capaz de suportar o aumento do **n**. Confira isso na seguinte tabela que corresponde a um esforço computacional polinomial R(n) = n^5:

A Simple Example (Problema 2)

n	rotas por segundo	n^5	cálculo total
5	250 milhoes	3 125	insignific
10	110 milhoes	100 000	insignific
15	71 milhoes	759 375	0.01 seg
20	53 milhoes	3 200 000	0.06 seg
25	42 milhoes	9 765 625	0.23 seg

Representation

Representation is an ordered list of city numbers known as an *order-based* GA.

1) London 3) Dunedin 5) Beijing 7) Tokyo
2) Venice 4) Singapore 6) Phoenix 8) Victoria

CityList1 (3 5 7 2 1 6 4 8)

CityList2 (2 5 7 6 8 1 3 4)

Crossover

Crossover combines inversion and recombination:

* *

Parent1 (3 5 **7 2 1 6** 4 8)

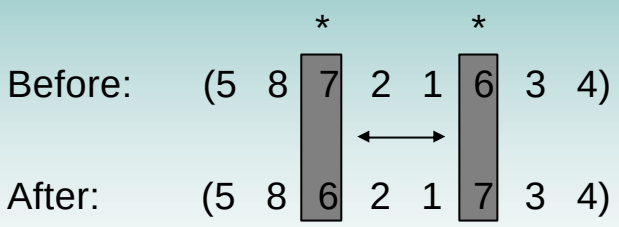
Parent2 (2 5 7 6 8 1 3 4)

Child (5 8 **7 2 1 6** 3 4)

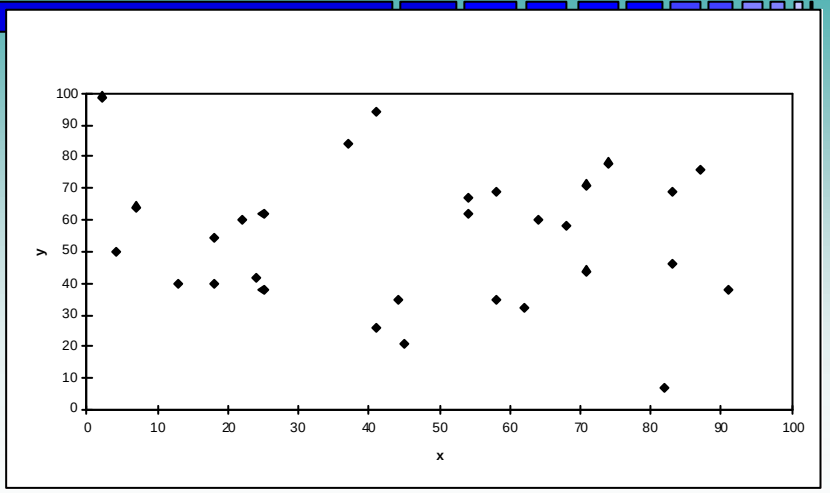
This operator is called the *Order1* crossover.

Mutation

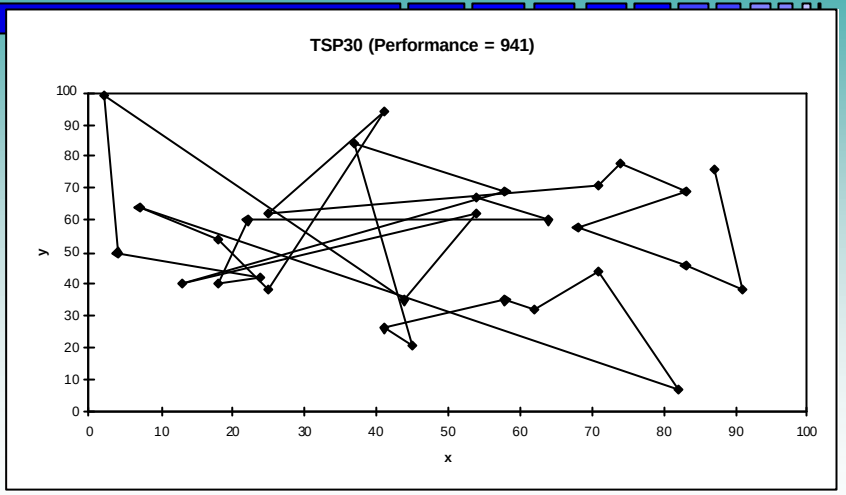
Mutation involves reordering of the list:



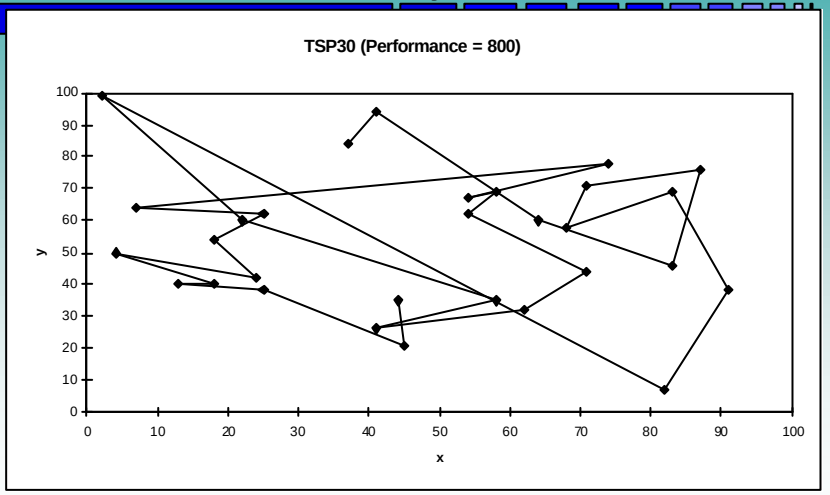
TSP Example: 30 Cities



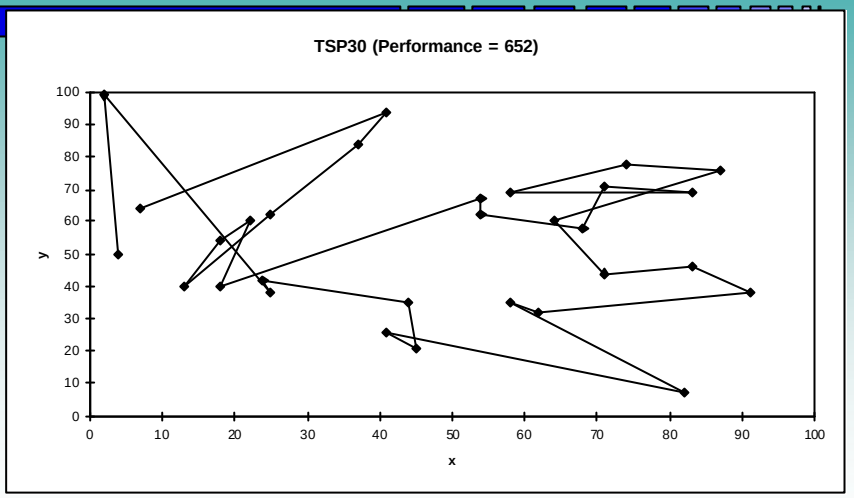
Solution $_i$ (Distance = 941)



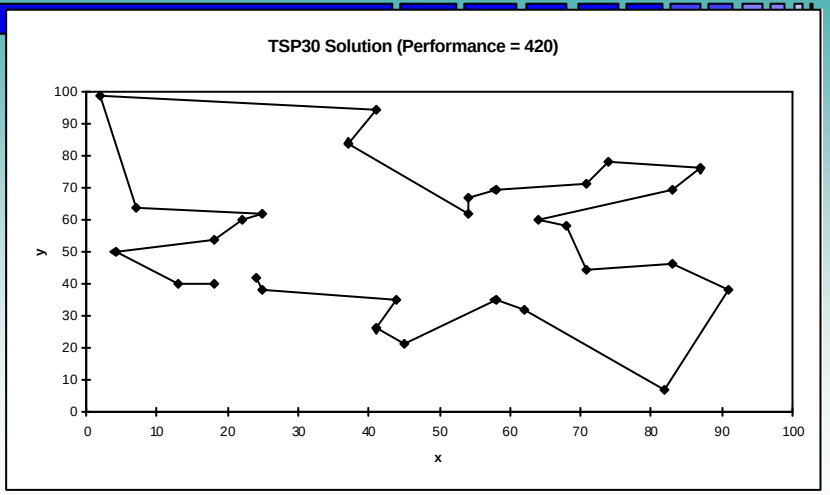
Solution $_j$(Distance = 800)



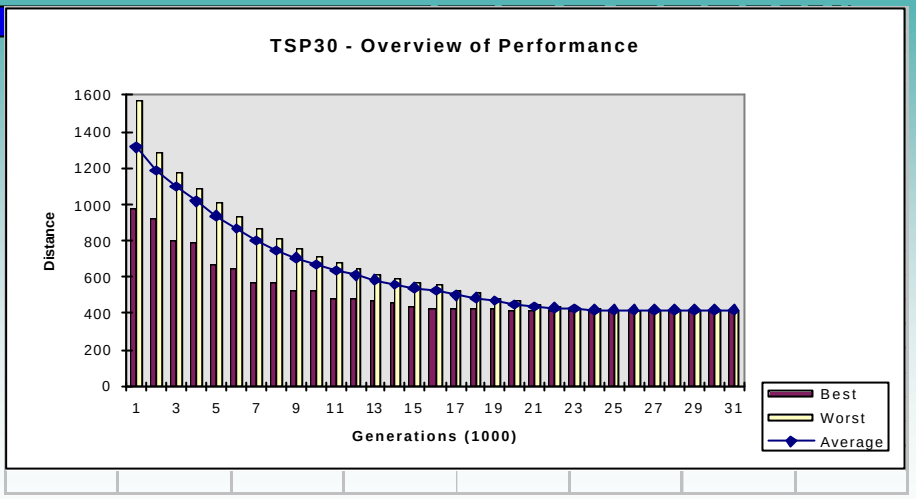
Solution $_k$(Distance = 652)



Best Solution (Distance = 420)



Overview of Performance



Problema 3

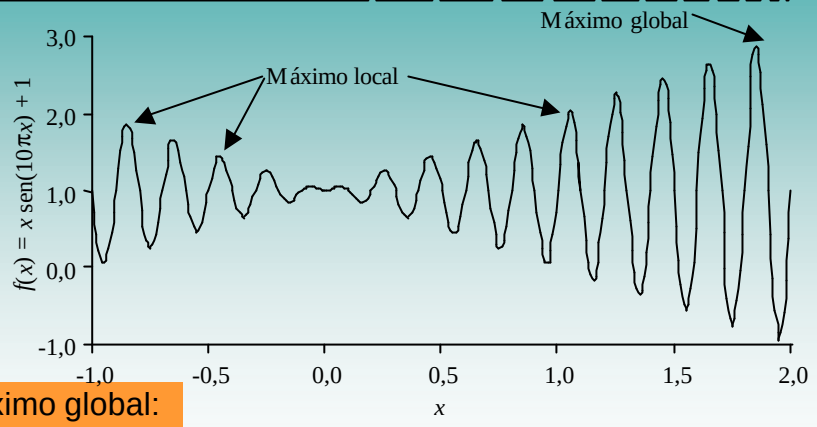
Achar o máximo da função utilizando Algoritmos Genéticos,

$$f(x) = x \operatorname{sen}(10\pi x) + 1{,}0$$

Restrita ao intervalo:

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

Problema 3



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

Problema 3

- 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.

O Cromossomo Problema 3

- Representar o único parâmetro deste problema (a 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

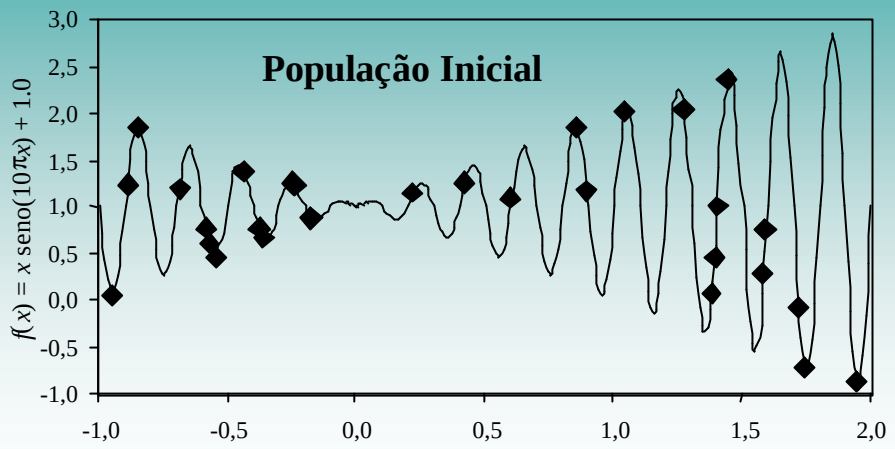
O Cromossomo Problema 3

- Decodificação
  - cromossomo = 1000101110110101000111
  - $b_{10} = (1000101110110101000111)_2 = 2288967$
  - Valor de *x* precisa estar no intervalo [-1,0; 2,0]

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

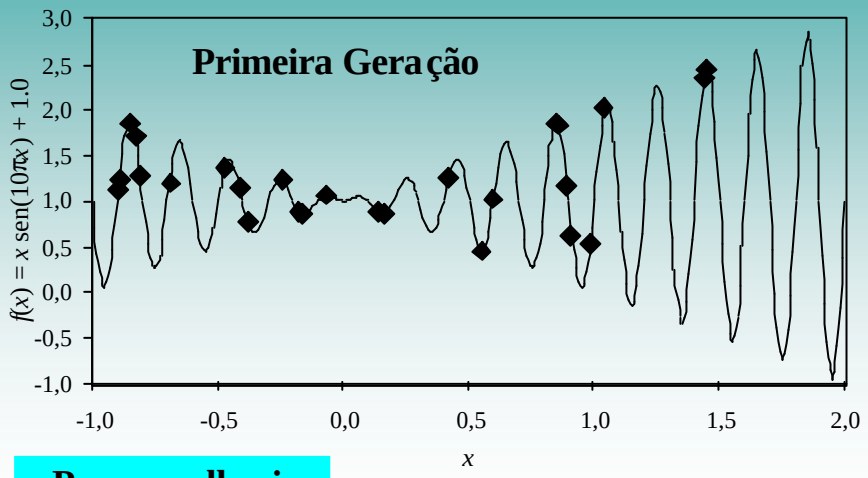
$$x = -1 + (2+1)\frac{2.288.967}{2^{22} - 1} = 0{,}637197$$

As Gerações do Problema 3



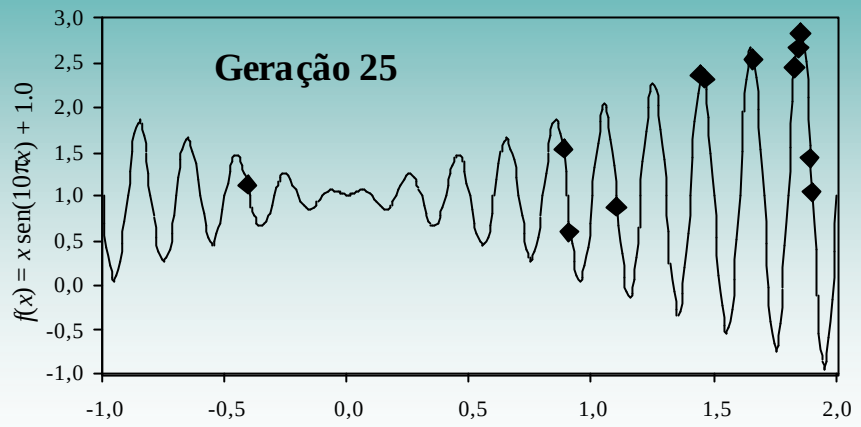
População gerada aleatoriamente

As Gerações do Problema 3



Pouca melhoria

As Gerações do Problema 3



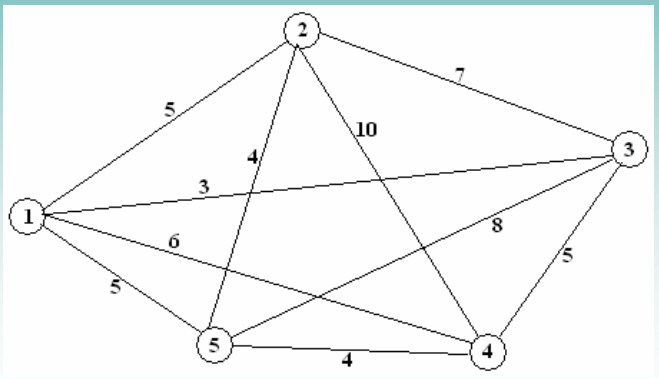
A maioria dos indivíduos encontraram o máximo global

Exercício

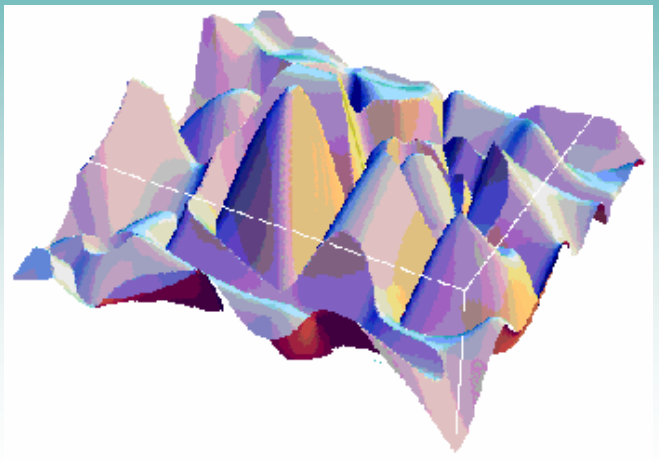
- Encontrar de x para o qual a função $f(x) = x^2 - 3x + 4$ assume o valor mínimo.
  - Assumir que $x \in [-10, +10]$
  - Codificar X como vetor binário
  - Criar uma população inicial com 10 indivíduos
  - Aplicar Mutação com taxa de 5%
  - Aplicar Crossover com taxa de 80%
  - Usar seleção por torneio.
  - Usar 10 gerações.

Exercício

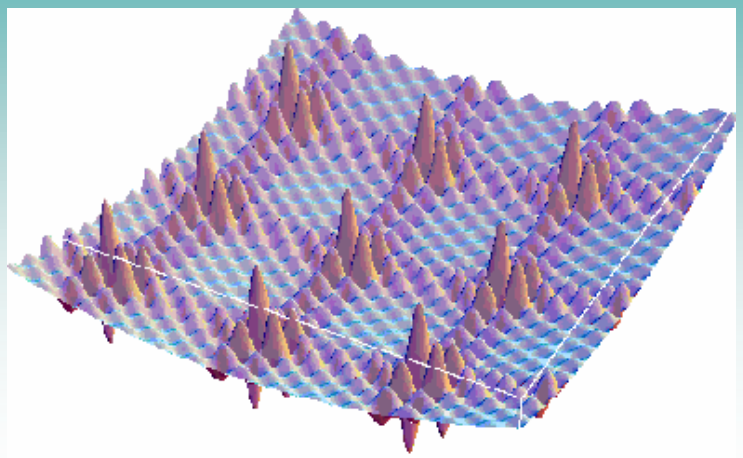
- Problema caixeiro viajante



Algumas superfícies



Algumas superfícies



Heuristica e meta-heuristica

- Heuristica: (do grego *heuriskein* = descobrir, encontrar) É qualquer método ou técnica criada, desenvolvida para resolver um determinado tipo de problema, através de um enfoque intuitivo, para se obter uma solução razoável. Técnica que procura boas (ou próximas às ótimas) soluções a um razoável custo computacional, sem ser capaz de garantir a viabilidade do ótimo.

- Meta-Heuristica: Deriva do grego *heuriskein* = descobrir, encontrar e o sufixo "*meta*" = além de, em um nível superior. São técnicas que, quando aplicadas a métodos de busca local, permitem a superação da otimalidade local com vistas à obtenção de soluções de qualidade superior. São heurísticas de busca no espaço de soluções.

Algoritmos Genéticos - Conceitos

COMPARATIVO ENTRE CONCEITOS

BIOLOGIA	ALGORITMOS GENÉTICOS
Cromossomo	Indivíduo (*string*)

locus 1 — alelo — 1º gene — valor do gene

genótipo= $1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 0.2^0 = 22$

fenótipo = valor de f (22)