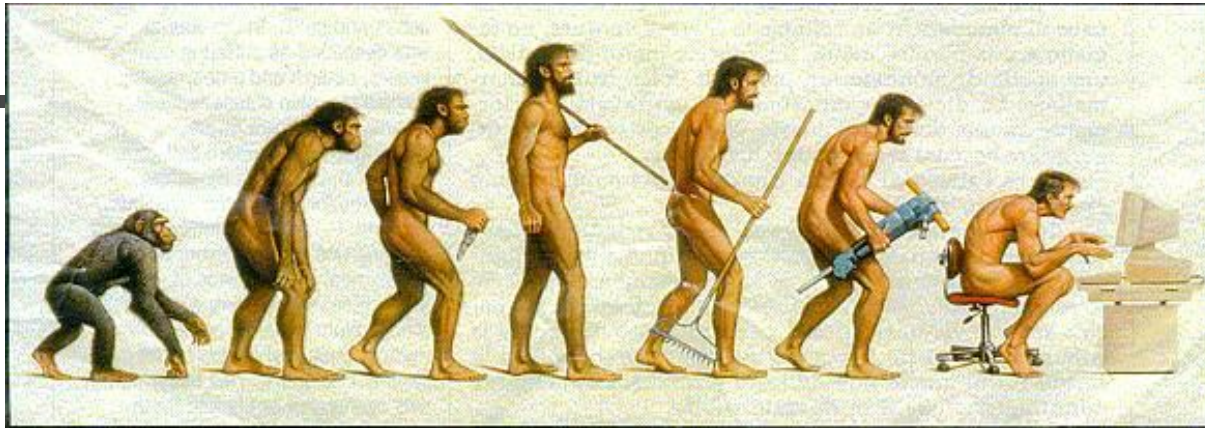


Computação Evolutiva



Prof. Eduardo R. Hruschka

(Slides baseados nos originais do Prof. André C. P. L. F. de Carvalho)



Principais Tópicos

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



Introdução

- Computação Evolutiva
 - Técnicas computacionais para resolução de problemas baseados em:
 - Genética
 - Teoria da evolução natural
 - Pesquisas tiveram início na década de 50



Algoritmos Evolutivos

Estratégias
de Evolução

Algoritmos
Genéticos

Programação
Evolutiva



Programação
Genética



Sistemas de
Classificação



Algoritmos Evolutivos

- Aspectos de cada área estão sendo assimilados por outras
 - Difícil definir fronteiras
- Programação Evolutiva (PE)
 - Utilizada principalmente para otimização de funções contínuas, sem usar *crossover*
- Estratégias Evolutivas (EEs)
 - Utilizadas principalmente para otimização de funções contínuas, utiliza *crossover*



Algoritmos Evolutivos

- Algoritmos Genéticos (AGs)
 - Usados em geral para problemas de otimização combinatória
 - Programação Genética (PG)
 - Evolui programas
 - Sistemas de classificação
 - Evolui regras



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 (*Charles Darwin, A Origem das Espécies*)



Algoritmos Genéticos

- Desenvolvido por John Holland e sua equipe (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

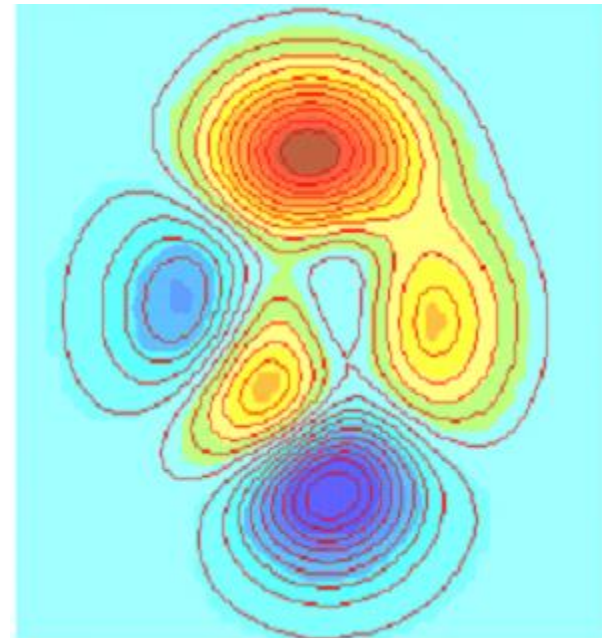
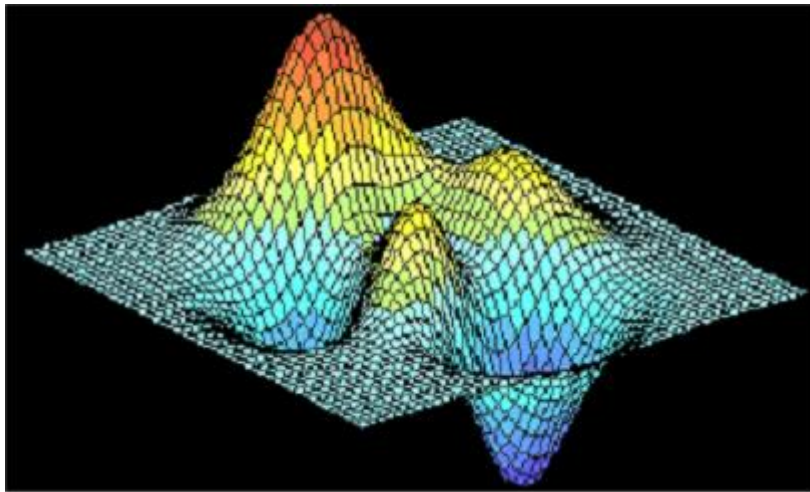
- Utilizam uma população de soluções candidatas (indivíduos)
- Otimização ocorre em várias gerações
 - A cada geração
 - Mecanismos de seleção selecionam os indivíduos mais aptos
 - Operadores de reprodução geram novos indivíduos



Algoritmos Genéticos

- Cada indivíduo representa uma possível solução para um dado problema
- A cada indivíduo é associado um escore de aptidão - mede o quão boa é a solução que ele representa
- Indivíduos mais aptos têm mais oportunidades de serem reproduzidos
 - Produzindo descendentes cada vez mais aptos

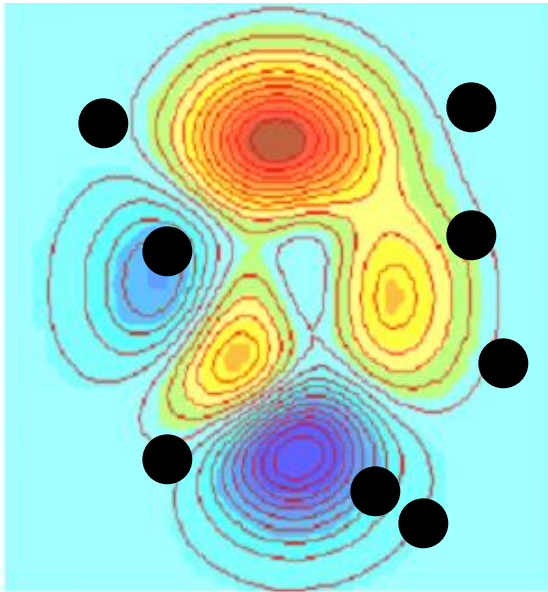
Processo Evolutivo



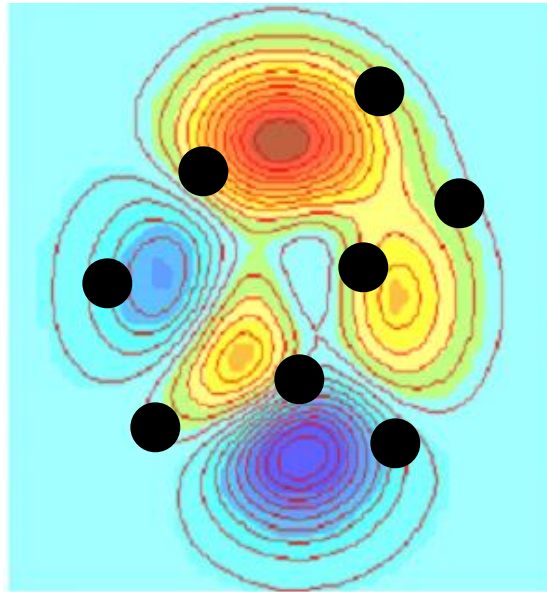
www.cs.bham.ac.uk/~axk



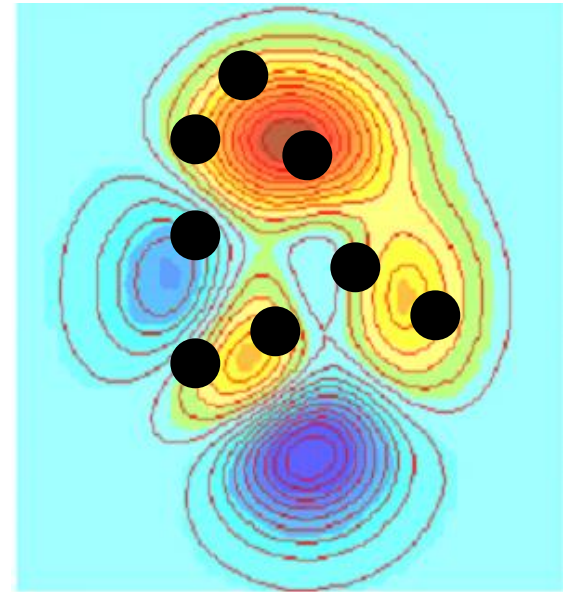
Processo Evolutivo



Geração 1

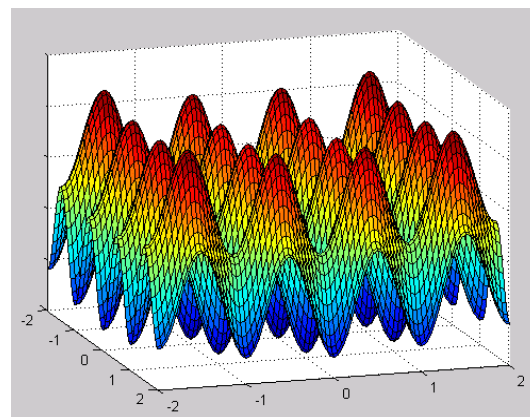
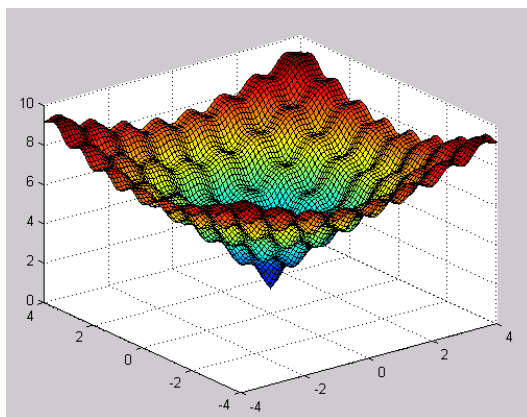
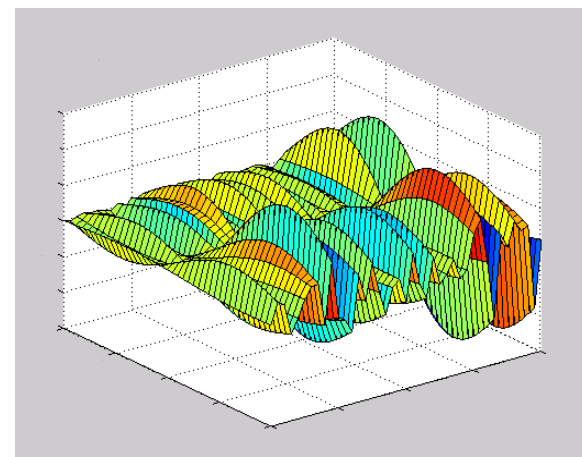
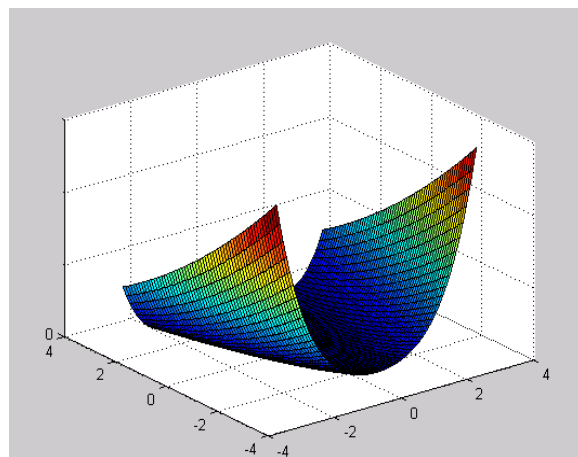
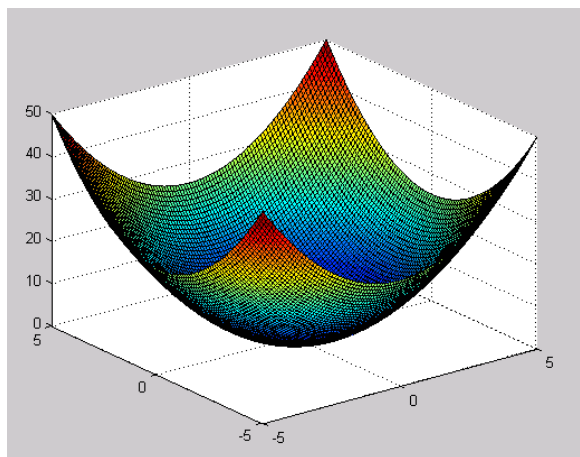


Geração 10

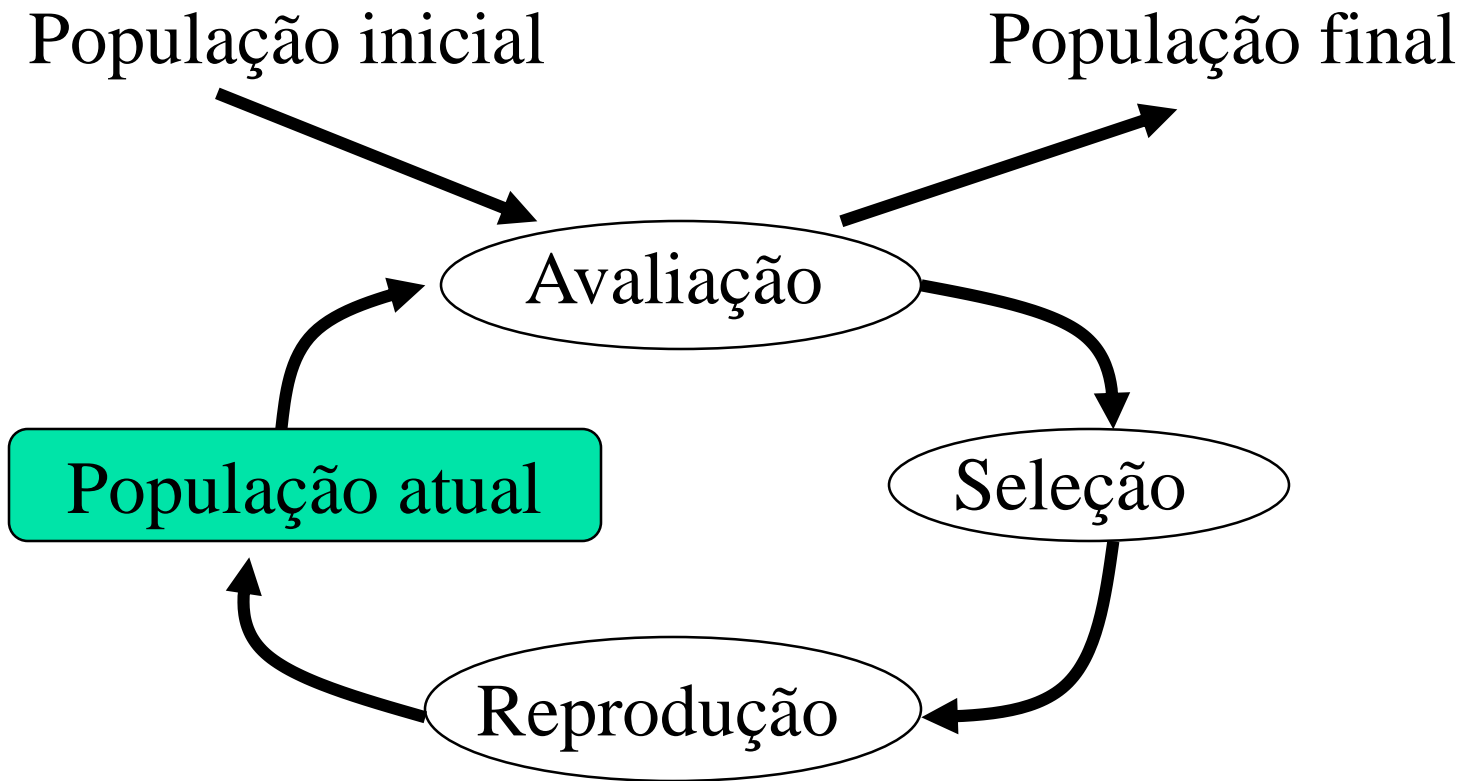


Geração 20

Espaço de Busca



Algoritmos Genéticos





Algoritmos Genéticos

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



Princípios básicos

- Indivíduo
- Codificação
- Função de aptidão
- Reprodução
- Convergência



Indivíduo

- Possível solução para um dado problema
 - Também chamado de cromossomo ou *string*
- Codificado como vetor de características
- Cada indivíduo possui um valor de aptidão
 - Mede qualidade da solução que ele representa
- Conjunto de indivíduos forma uma população



Codificação

- Cada indivíduo é codificado por um conjunto de parâmetros (genes)
 - Genes podem assumir valores:
 - Binários
 - Inteiros
 - Reais
- Parâmetros são combinados para formar *strings* ou vetores (cromossomos)



Codificação

- **Genótipo**
 - Conjunto de parâmetros representado por um cromossomo
 - Contém informação necessária para construir um organismo (fenótipo)
 - Fenótipo codificado
- **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

- Tradicionalmente, os indivíduos são representados por vetores binários
 - 1 = presença
 - 0 = ausência
 - Representação independente do problema
 - Permite utilizar operadores de reprodução padrão



Codificação

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



Codificação

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



Codificação de Inteiros

- Representar um valor $g \in I$ por um vetor de binários b_1, b_2, \dots, b_N
 - Valor de g restrito ao intervalo $[g_{\min}, \dots, g_{\max}]$ de valores inteiros
 - Definir função $f[g] \rightarrow \{0,1\}^N$
 - Possibilidades
 - Valor de g varia entre 0 a 2^N-1
 - Valor de g varia entre M a $M + 2^N-1$
 - Valor de g varia entre 0 a $K-1$, onde $K \neq 2^N-1$



Codificação de Inteiros

- Valor de g varia entre 0 a 2^N-1
 - Converter valor de g para valor binário correspondente
- Existem várias outras alternativas dependentes de contexto...



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 à utilidade ou habilidade do indivíduo
 - Aptidão também captura a probabilidade do indivíduo sobreviver para a próxima geração
 - Cada problema tem sua própria função de aptidão



Função de aptidão

- 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

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



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



Aptidão baseada em ranking

- Aptidão padrão: escolha da escala do índice de aptidão pode *prejudicar* reprodução
 - Ex.: 1, 10, 100
- Usar *ranking* de cromossomos por aptidão
 - Seleção baseada apenas nos *ranks*
 - Pode-se converter *ranks* em probabilidades



Aptidão ranking-espaco

- Medidas de aptidão vistas ignoram diversidade
 - mede o quão diferente são os cromossomos de uma população
- Princípio da diversidade: ser diferente pode ser tão bom quanto ser apto
 - *exploration* (busca global) x *exploitation* (busca local)



Seleção

- Escolhe preferencialmente indivíduos com maiores valores de aptidão
 - Probabilisticamente
 - Procura manter a diversidade da população
- Indivíduos mais aptos têm maiores probabilidades de gerar descendentes
 - Os quais serão provavelmente cada vez mais aptos



Seleção

- Selecciona uma população intermediária
 - Para aplicar os operadores de reprodução
- Existem vários métodos de seleção
 - Por roleta
 - Por torneio
 - Amostragem Universal Estocástica



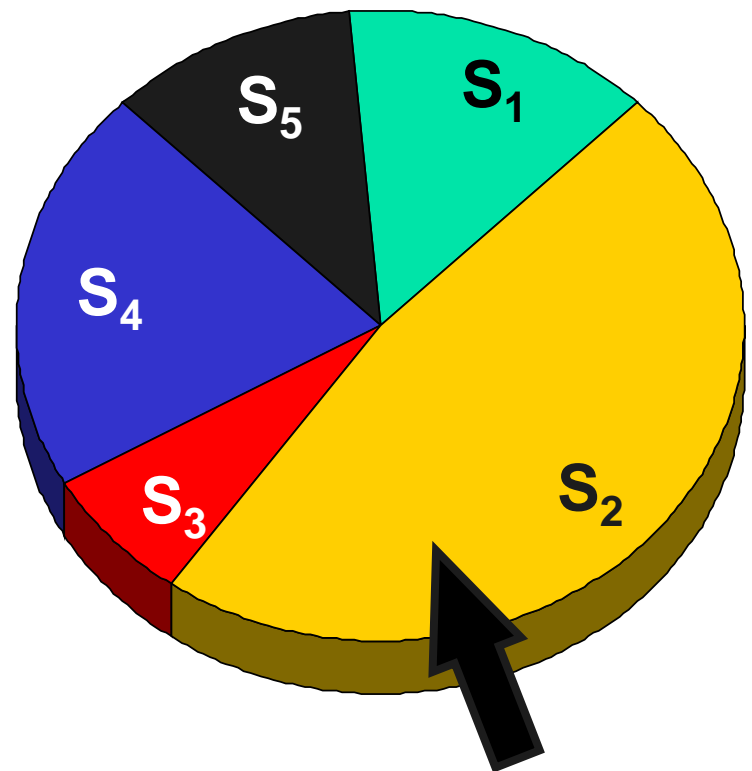
Seleção pela roleta

- Método simples e muito utilizado
- Escolhe indivíduos para fazer parte da próxima geração por meio de uma *roleta polarizada*
- Cada indivíduo da população é representado na roleta por uma fatia proporcional ao seu índice de aptidão

Seleção pela roleta

Método da Roleta baseado em Aptidão Relativa

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





Seleção por torneio

- Escolhe n indivíduos da população aleatoriamente, com a mesma probabilidade
 - Geralmente $n = 3$
- Cromossomo com maior aptidão dentre estes n cromossomos é selecionado para a população intermediária
- Processo se repete até que a população intermediária seja preenchida

Seleção por torneio

Método da Torneio baseado em Aptidão Relativa

Indivíduo	Aptidão	Aptidão	
S_i	$f(S_i)$	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

Supondo $n = 3$

Candidatos \Rightarrow vencedor

$S_1, S_2, S_5 \Rightarrow S_2$

$S_2, S_4, S_5 \Rightarrow S_2$

$S_5, S_1, S_3 \Rightarrow S_1$

$S_4, S_5, S_3 \Rightarrow S_4$

$S_3, S_1, S_5 \Rightarrow S_1$



Seleção por torneio

- Possui um parâmetro que permite definir explicitamente a pressão seletiva durante a evolução
 - Tamanho do torneio
 - Quanto maior o número de indivíduos que participam do torneio, maior a pressão seletiva
 - Indivíduo tem que ser melhor do 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

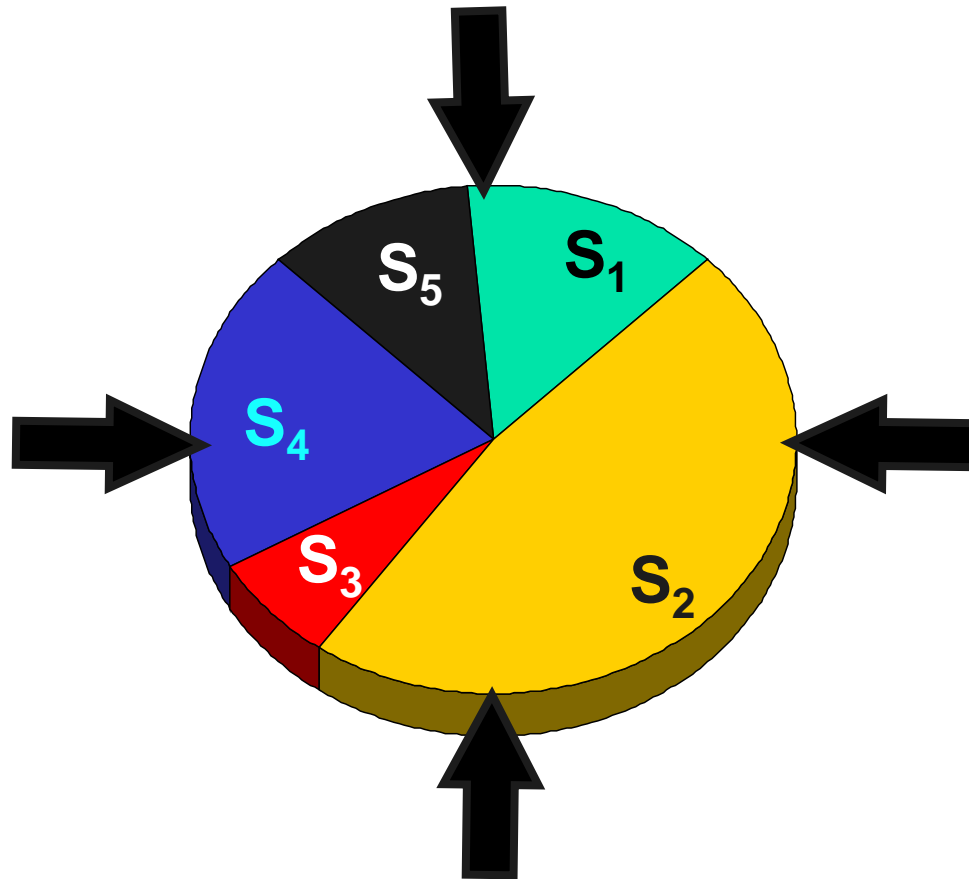
- Deve haver equilíbrio entre pressão seletiva e diversidade
- Formas de prevenir convergência prematura
 - Controlar número de oportunidades de reprodução de cada indivíduo
- Formas de promover diversidade
 - Aumento do tamanho da população
 - Aumento da taxa de mutação



Seleção por Amostragem Universal Estocástica

- Conhecido como SUS (do inglês, *Stochastic Universal Sampling*)
- Variação do método da roleta
 - Utiliza n agulhas igualmente espaçadas ao invés de 1
 - Onde n é o número de indivíduos a serem selecionados para a próxima geração
 - Ao invés de n vezes, a roleta é girada uma única vez
 - Exibe menos variância do que as repetidas chamadas do método da roleta

Seleção por Amostragem Universal Estocástica





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 pais que serão substituídos 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 os combina, gerando novos indivíduos



Crossover

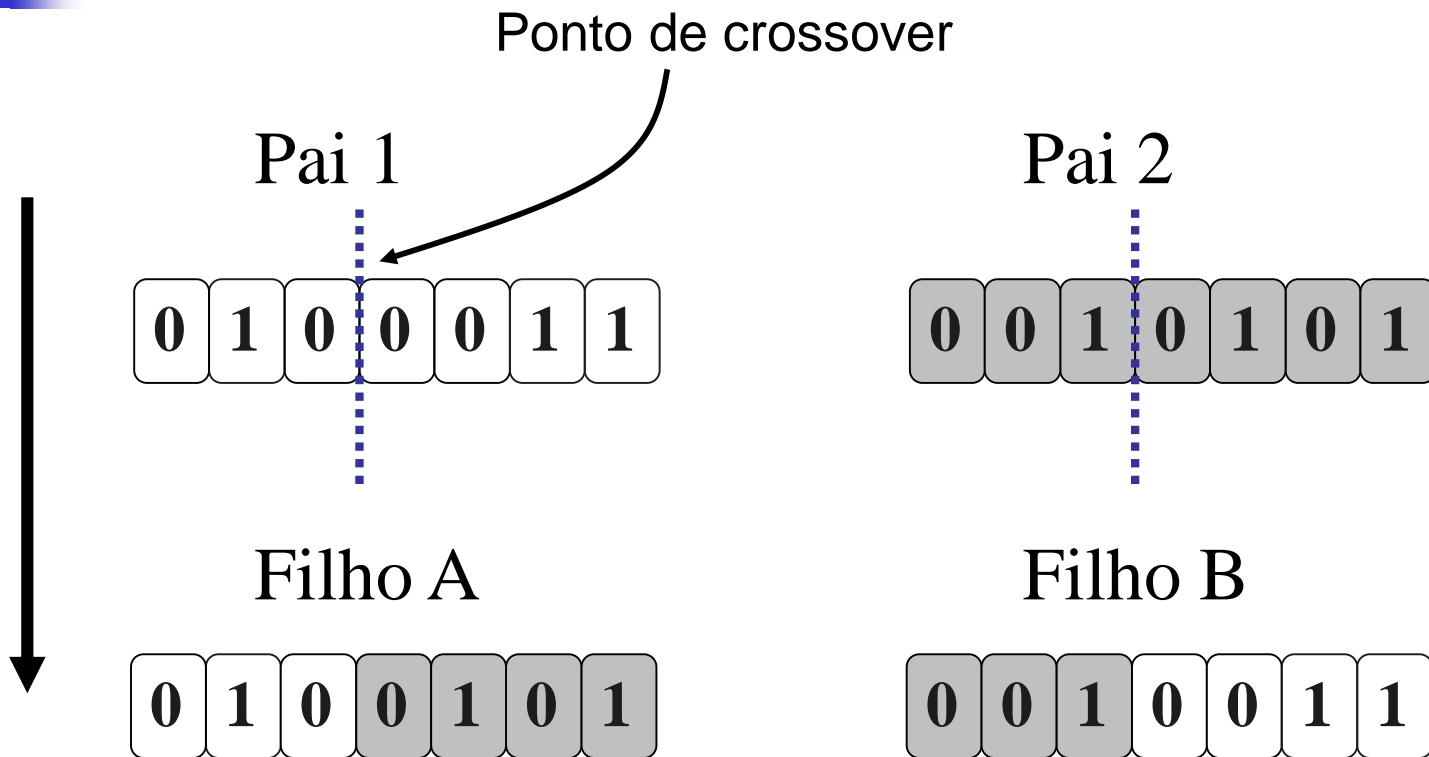
- É o operador genético predominante
 - A taxa de *crossover* usualmente deve ser maior que a taxa de mutação
 - Taxa de *crossover*: $0.6 < P_c < 1.0$



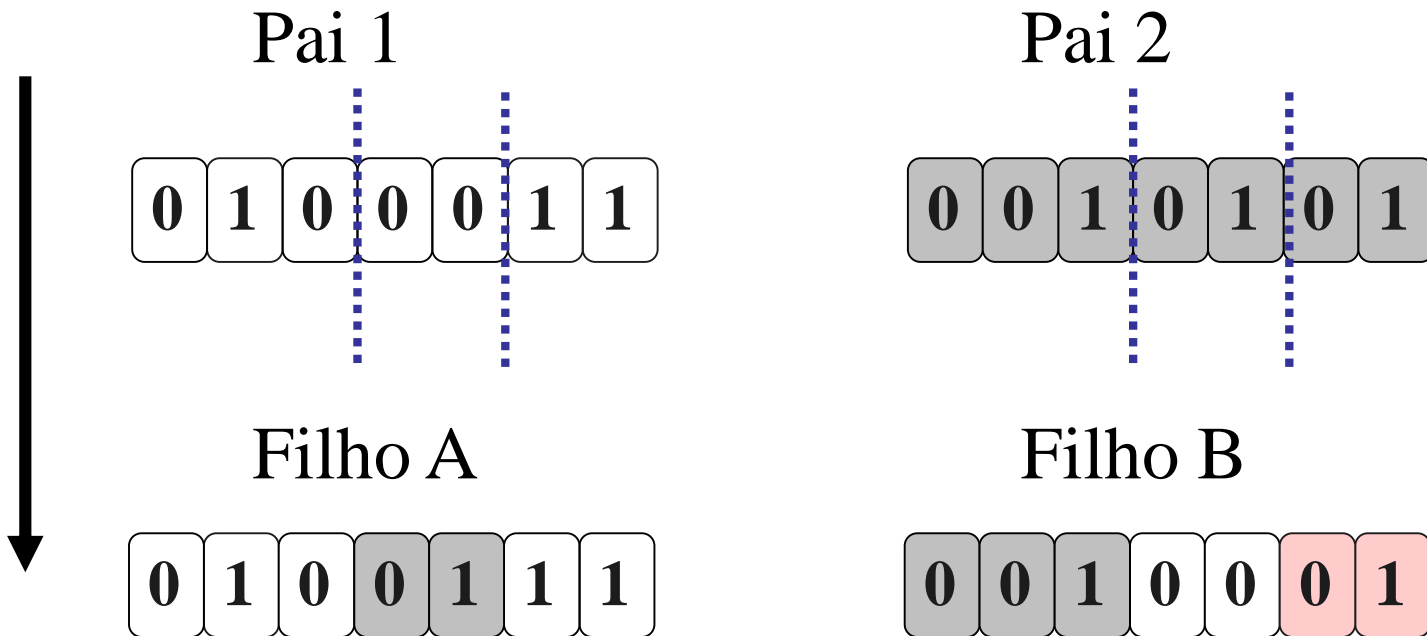
Crossover

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

Crossover 1 ponto



Crossover de 2 pontos



Crossover Uniforme

Máscara: 0 1 0 1 0 0 0

Pai 1

0 1 0 0 0 1 1

Pai 2

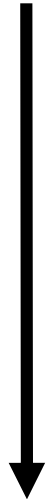
0 0 1 0 1 0 1

Filho A

0 1 1 0 1 0 1

Filho B

0 0 0 0 0 1 1





Mutação

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



Mutação

- Assegura que a probabilidade de atingir qualquer ponto do espaço de busca nunca será zero
 - Reduz chance de ficar preso em Mínimos Locais
 - Taxa de mutação pequena $P_m \cong 0.001$



Mutação

Antes da mutação

0 1 0 0 0 1 1

Após a mutação

0 1 1 0 0 1 1



Elitismo

- Conjunto de indivíduos melhores adaptados é deterministicamente selecionado
- Evita modificações deste(s) indivíduo(s) pelos operadores genéticos
 - Utilizado para que os melhores indivíduos não desapareçam da população



Observações

- Se o AG estiver corretamente implementado, a população usualmente evolui (na média) ou não piora em gerações sucessivas
- Aptidões tendem a aumentar na direção de um ponto de ótimo local ou global



Critérios de Parada

- Tempo de execução
- Número de gerações
- Valores de aptidão mínimo, médio e/ou máximo
- Convergência relativa ou estrita (pouco usada na prática)



Um Algoritmo Genético

- 1. Gerar uma população inicial de cromossomos*
- 2. Avaliar cada cromossomo da população*
- 3. Enquanto critério de parada não for atingido fazer:*
 - 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 :

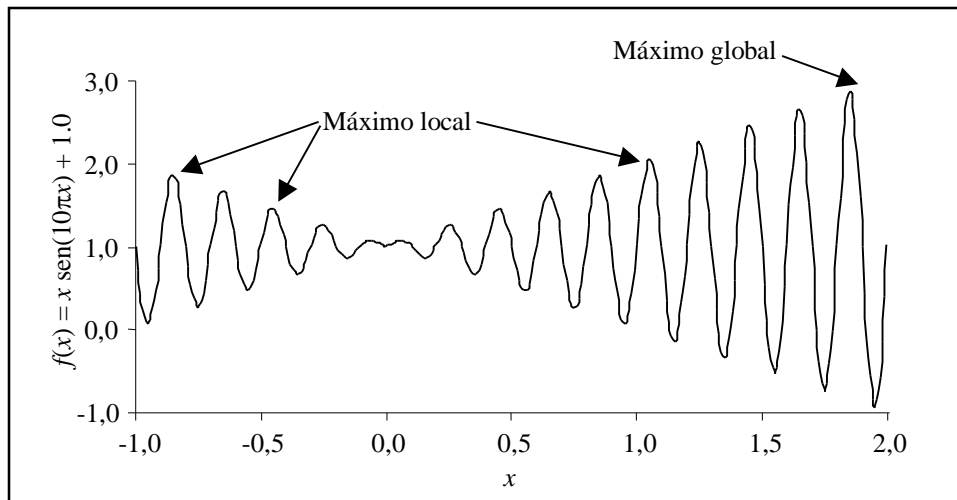
$$f(x) = x \operatorname{seno}(10\pi x) + 1,0$$

Restrita no intervalo:

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

Exemplo

Gráfico da função $f(x) = x \operatorname{seno}(10\pi x) + 1.0$



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

→ Existem vários pontos de máximo local



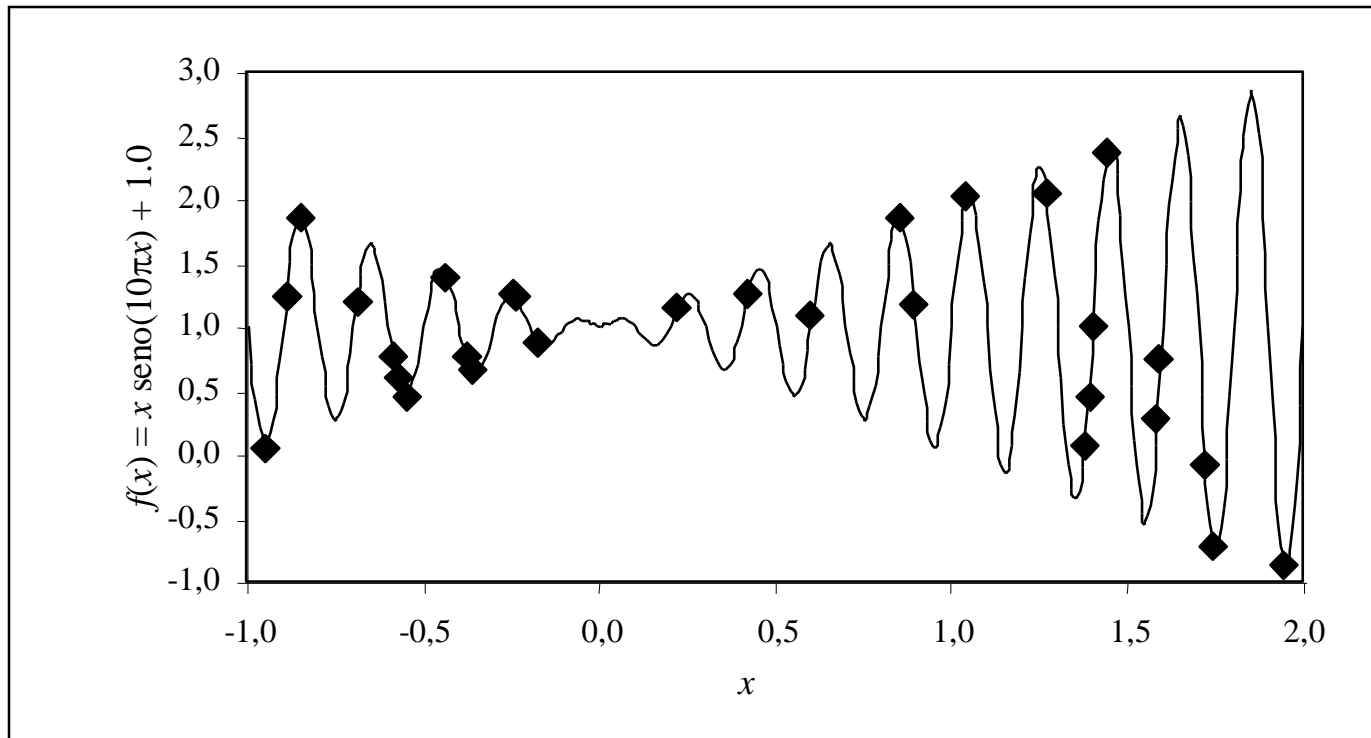
Representação dos Cromossomos

- Representar o único parâmetro deste problema (a variável x) na forma de um cromossomo
 - Binário = 1000101110110101000111
 - Decimal = $(1000101110110101000111)_2 = 2.288.967$
 - Valor de x precisa estar no intervalo $[-1,0; 2,0]$

Geração da População Inicial

Rank i	Cromossomo s_i	x_i	Função objetivo $f(x_i)$	Aptidão f_i	Aptidão acumulada $\sum_{k=1}^i f_k$
1	1101000000011110110111	1,43891	2,35251	2,00000	2,00000
2	1100000110100100011111	1,26925	2,04416	1,93103	3,93103
3	1010111001010110010000	1,04301	2,01797	1,86207	5,79310
4	1001111000011001000101	0,85271	1,84962	1,79310	7,58621
5	1001110110111000011100	0,84829	1,84706	1,72414	9,31035
6	0000110011111010010110	-0,84792	1,84610	1,65517	10,96552
7	0011000000100111010010	-0,43570	1,39248	1,58621	12,55172
8	0111100101000001101100	0,42098	1,25777	1,51724	14,06897
9	010000000110011101000	-0,24764	1,24695	1,44828	15,51724
10	010000010001111011110	-0,24343	1,23827	1,37931	16,89655
11	0000100101000000111010	-0,89156	1,23364	1,31035	18,20690
12	0001101001100010101111	-0,69079	1,19704	1,24138	19,44828
13	1010000110011000011011	0,89370	1,17582	1,17241	20,62069
14	0110100001011011000100	0,22292	1,14699	1,10345	21,72414
15	1000100011110001000011	0,60479	1,09057	1,03448	22,75862
16	1100110011001010001110	1,39988	0,99483	0,96552	23,72414
17	0100011001000100011101	-0,17655	0,88140	0,89655	24,62069
18	0011010011110100101000	-0,37943	0,77149	0,82759	25,44828
19	0010001101001100101100	-0,58633	0,75592	0,75862	26,20690
20	1101110101101111111111	1,59497	0,74904	0,68966	26,89655
21	0011011011001101110110	-0,35777	0,65283	0,62069	27,51724
22	0010010001001111100111	-0,57448	0,58721	0,55172	28,06897
23	1100101110110011111000	1,38714	0,45474	0,48276	28,55172
24	0010011001100110100111	-0,54999	0,45001	0,41379	28,96552
25	1101110010010100100001	1,58492	0,27710	0,34483	29,31035
26	1100101011000111010011	1,37631	0,06770	0,27586	29,58621
27	0000010000100100110001	-0,95144	0,04953	0,20690	29,79310
28	1110100001000000010001	1,72169	-0,08458	0,13793	29,93103
29	1110101000111100000000	1,74494	-0,72289	0,06897	30,00000
30	1111101100000001010111	1,94147	-0,87216	0,00000	30,00000

Geração da População Inicial





Exemplo 2

- Problema: encontrar valor inteiro de x que minimiza $f(x) = x^2$, $x \in [-7, +7]$
 - Representação \rightarrow 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 \rightarrow aleatória
 - População de tamanho 4
 - Função de aptidão $\rightarrow f(x)$



Exemplo 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
- 2 filhos substituirão indivíduos com menor aptidão



Exemplo 2

	x	f(x)
Pai ₁ = A ₁ → 1 1 0 1	-5	25
Pai ₂ = A ₃ → 0 0 1 0	2	4
Filho ₁ → 1 1 1 0		
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 valor de x para o qual a função $f(x) = x^2 - 4x + 4$ assume o valor mínimo
 - Assumir que $x \in [-15, +15]$
 - 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



Aplicações

- Otimização contínua
- Otimização combinatorial
 - Problema do caixeiro viajante
 - Problema de empacotamento
 - Alocação de recursos (*job shop schedulling*)
- Projetos
 - Projeto de pontes
- Aprendizado de Máquina
 - Jogos
 - Classificação, Agrupamento

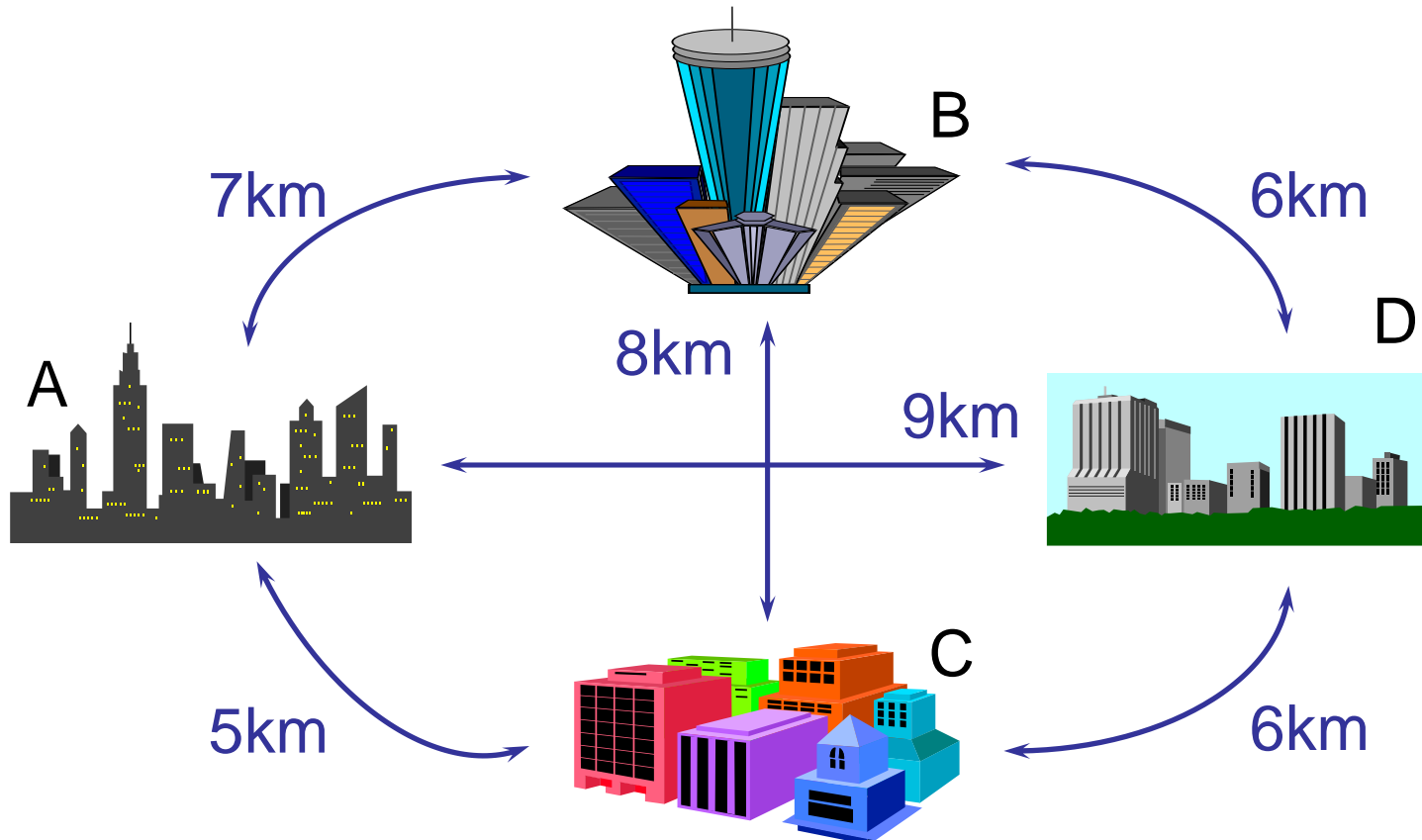


Trabalho 1

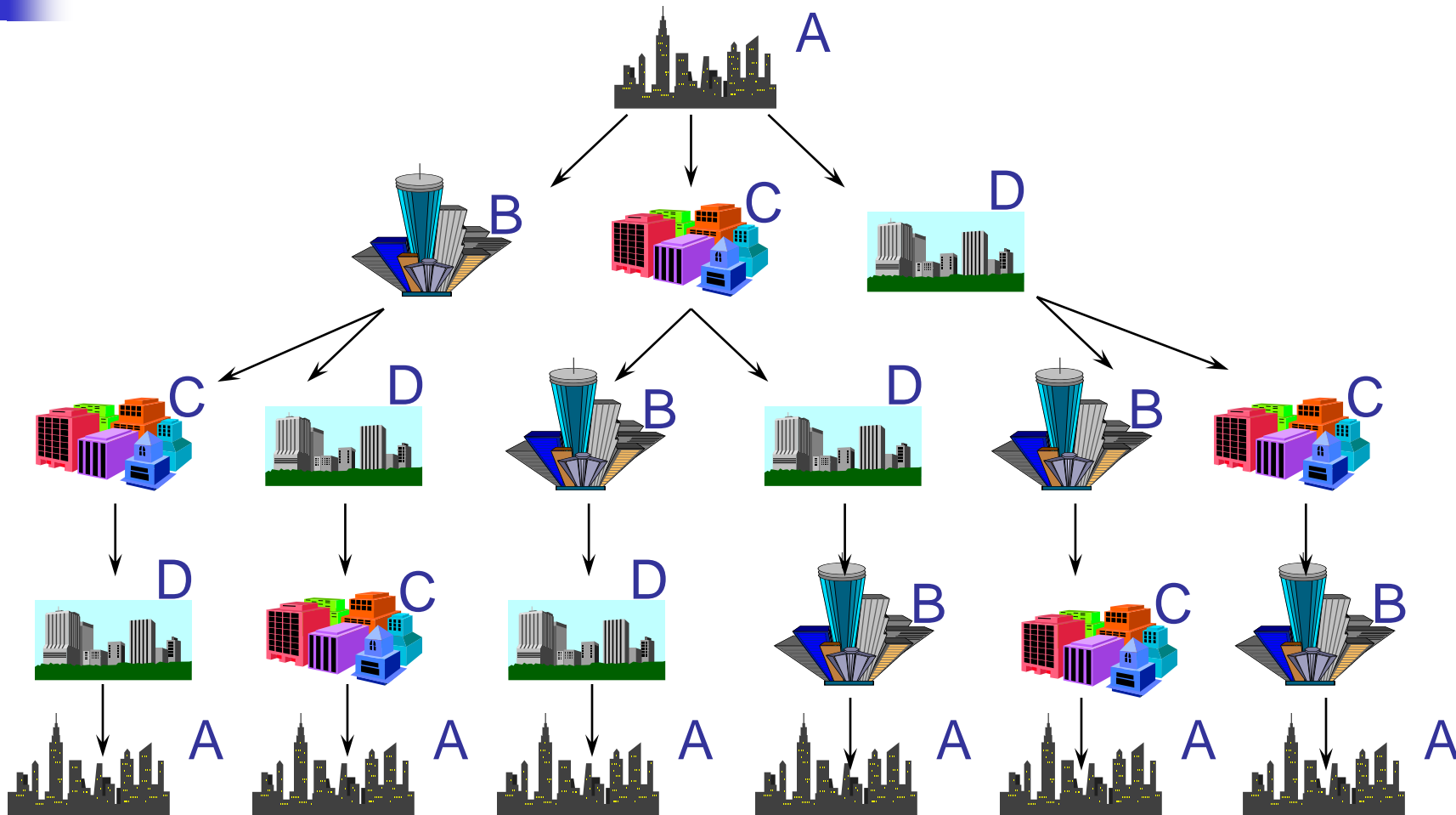
- Encontrar de valor x para o qual a função $f(x) = 2x^4 - 3x^3 + 7x - 5$ assume o valor mínimo
 - Assumir que $x \in [-31, +32]$
 - Codificar x como vetor binário
 - Operadores diferentes (por grupo)
 - Fixar número de gerações
 - Comparar soluções
 - Custo computacional?

TSP

Rotas entre as cidades A, B, C e D:



TSP como Árvore de Busca





TSP – Desafio Computacional

- Explosão Combinatória
 - Para 4 cidades: 6 caminhos possíveis
 - Para 10 cidades: 362.880 caminhos possíveis
 - Estime o custo computacional de avaliar uma rota, multiplique pelo número de caminhos possíveis, e decida se há tempo viável para tomar a decisão.

TSP

- Histórico
 - Problemas matemáticos relacionados ao caixeiro viajante foram tratados por volta de 1800
 - Pelos matemáticos Sir William Rowan Hamilton e Thomas Penyngton Kirkman
 - TSP geral foi primeiro tratado por volta de 1930 por Karl Menger





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

Várias referências mais recentes – ver Google.



Exemplo de Matriz de Distâncias (*toy problem*)

	A	B	C	D	E	F	G	H
A	-							
B	42	-						
C	61	14	-					
D	30	87	20	-				
E	17	28	81	34	-			
F	82	70	21	33	41	-		
G	31	19	8	91	34	19	-	
H	11	33	29	10	82	32	59	-



Trabalho 2

- Utilizar AG para resolver problemas do caixeiro viajante (TSP)
 - Caixeiro viajante deve visitar N cidades em sua área de vendas
 - Caixeiro começa numa cidade-base, visita cada cidade uma única vez e retorna à sua origem
 - A cada viagem está associado um custo que é proporcional à distância percorrida
 - O caixeiro deve percorrer a rota mais curta, minimizando o custo global