Aprendizado de Máquina

Inteligência Artificial

Prof. Leandro C. Fernandes

Adaptado a partir dos materiais de:
Grupo de Inteligencia Computacional (UFPE) e do VisionLab da PUC-Rio

A IA usa sempre algumas metáforas...

- Cérebro e sistema nervoso

  $\Rightarrow$ conexionismo
- Linguagem + processos cognitivos

  $\Rightarrow$ IA simbólica
- Teoria da evolução

  $\Rightarrow$ computação evolutiva (algoritmos genéticos)

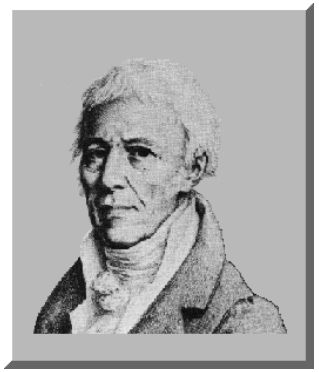
Indagações de alguns séculos atrás...



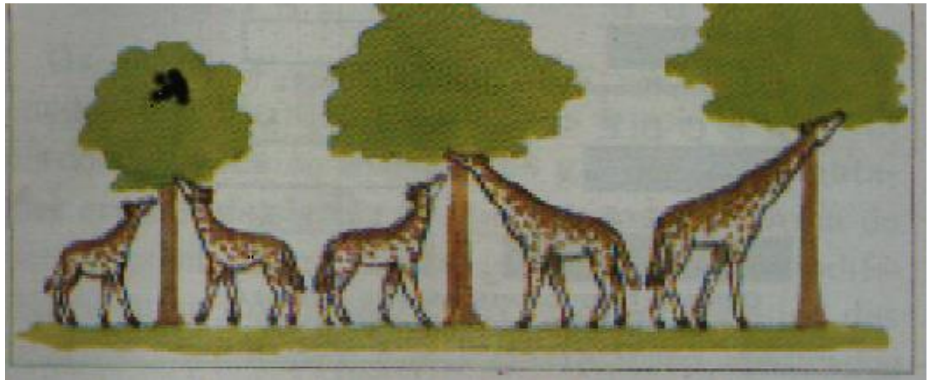
- Como explicar a diversidade de animais?
- Como explicar sua evolução?
  - Qual é a influência do dos antepassados?
  - Qual é a influência do meio ambiente?

História da Teoria da Evolução

- 1809: Jean-Baptiste Lamarck

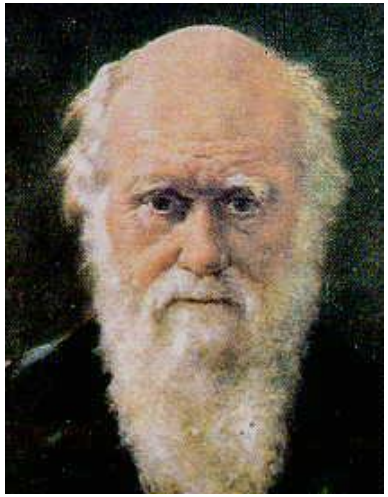


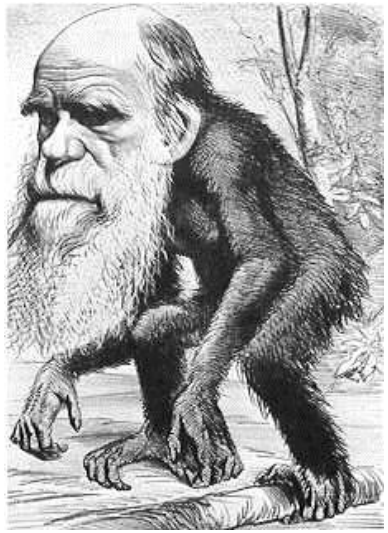
  - Lei do uso e do desuso
    - pelo uso e desuso de suas aptidões, a natureza força os seres a se adaptarem para sobreviverem.
  - Lei dos caracteres adquiridos.
    - Os serem mais fortes são mais capazes de “transmitir” suas aptidões às novas gerações



História da Teoria da Evolução

- 1859: Charles Darwin
  - Existe uma diversidade de seres devido aos contingentes da natureza (comida, clima, ...) e é pela lei da **Seleção Natural** que os seres mais adaptados ao seus ambientes sobrevivem
    - contra lei do uso de desuso
  - Os caracteres adquiridos são herdados pelas gerações seguintes
    - o homem vem do macaco...
- Na época, que isto tudo foi polêmico...





História da Teoria da Evolução

- 1865: Gregor Mendel
  - Formalizou a “herança de características”, com a teoria do DNA (ervilhas)
- 1901: Hugo De Vries
  - Só a seleção natural não é responsável pela produção de novas (mais adaptadas) espécies. Tem de haver uma mudança genética!
  - Formalizou o processo de geração de diversidade: **Teoria da Mutação**

Computação evolutiva

- 1975: Jonh Holland: Idealizou os algoritmos genéticos
  - Adaptation in Natural & Artificial Systems
    MIT Press, 1975 (2nd ed. 1992)

- Porque a evolução é uma boa metáfora?
  - Muitos problemas computacionais
    - envolvem busca através de um grande número de possíveis soluções
    - requerem que o programa seja adaptativo, apto a agir em um ambiente dinâmico
  - A evolução biológica é
    - é uma busca massivamente paralela em um enorme espaço de problema
    - soluções desejadas = organismos mais adaptados

Computação Evolutiva

ALGORITMOS GENÉTICOS

Roteiro

- Conceitos básicos
- Funcionamento dos algoritmos genéticos
  - seleção
  - mutação
  - reprodução
  - substituição
- Ferramentas de desenvolvimento e exemplos

Computação Evolutiva: introdução

- Computação evolutiva
  - Método probabilista de busca para resolução de problemas (otimização) “inspirado” na teoria da evolução
  - Tem várias variantes: algoritmos genéticos, programação genética, estratégia evolutiva e programação evolutiva

- Ideia:
  - indivíduo = solução
  - provoca mudança nos indivíduos por intermédio de *mutação* e *reprodução*
  - *seleciona* indivíduos mais adaptados através de sucessivas gerações
  - A aptidão de cada indivíduo é medida pela “função de aptidão” (*fitness function*) f(i): R ->[0,1]

Computação Evolutiva: introdução

- Os algoritmos evolucionários funcionam mantendo uma **população de estruturas** que **evoluem** de forma semelhante à evolução das espécies.

- Nestas estruturas são aplicados **operadores genéticos**, como a **recombinação** e **mutação**.

Computação Evolutiva: introdução

- Cada indivíduo recebe uma **avaliação (fitness)** que é uma quantificação da sua **qualidade** como solução do problema em questão.

- Baseados nesta avaliação são aplicados operadores genéticos de forma a simular a **sobrevivência do mais apto**.

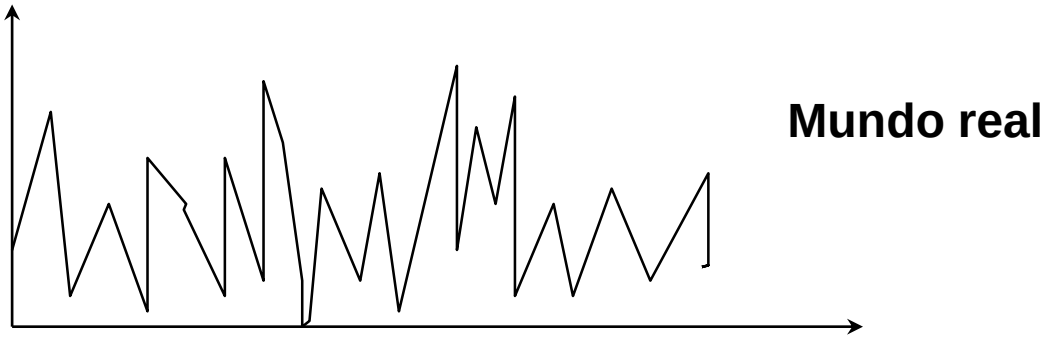
Computação Evolutiva: introdução

- Algoritmos evolucionários **buscam** (dentro da atual população) aquelas **soluções** que possuem as **melhores características** e tenta combiná-las de forma a gerar soluções ainda melhores.

- O processo é repetido até que tenha se passado tempo suficiente ou que tenhamos obtido uma solução satisfatória para nosso problema.

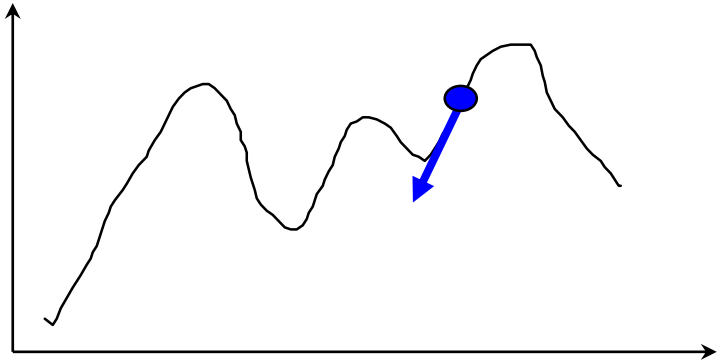
Aplicação: classes de problemas

- Aproximação de funções
  - não-lineares/lineares, multi-modais
  - Mono-modais e discretas/contínuas
- Otimização combinatória (NP hard)
- Aprendizagem
  - por isto não pode ser contida na estatística!!

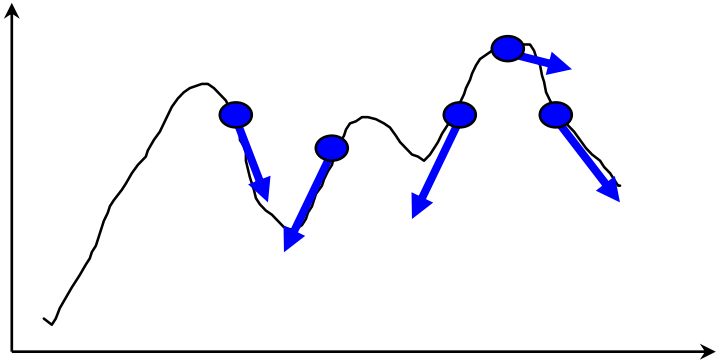


Métodos de busca: otimização

- Matemáticos de gradiente: *Hill-Climbing*
  - problema: mínimos locais

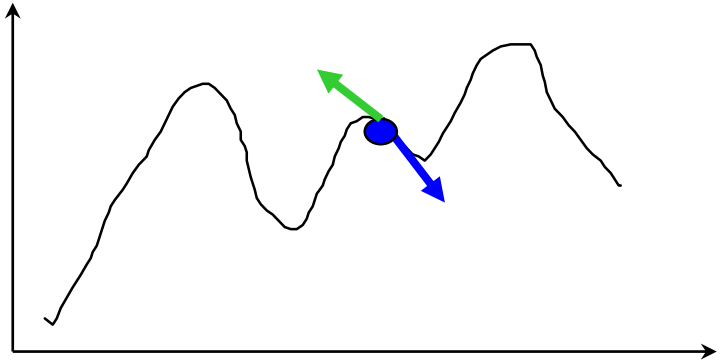


- Enumerativos -> cada ponto...
  - Problema: custa caro



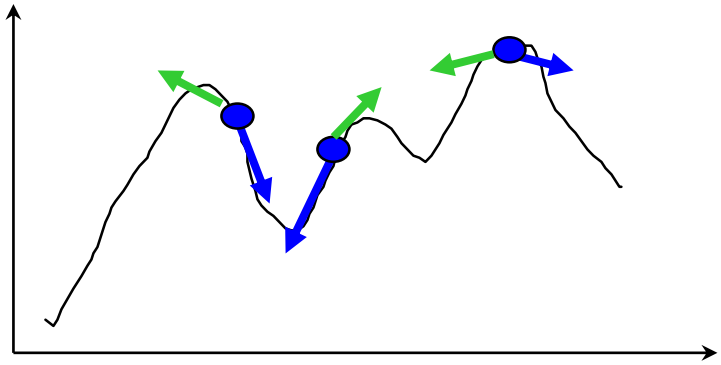
Métodos de busca: otimização

- Aleatórios: Recozimento Simulado (ou *Simulated Annealing*)



- Aleatórios: computação evolutiva
  - indivíduo = solução

Processo adaptativo e paralelo!



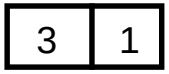
Algoritmos genéticos

- Definição de um problema em algoritmos genéticos:
  - É necessário definir uma maneira de codificar os **indivíduos**.
  - Definir uma **função de avaliação** para medir a capacidade de sobrevivência de cada indivíduo.
  - Definir um método de seleção dos pais.
  - Definir os **operadores genéticos** que serão utilizados.
    - Recombinação.
    - Mutação.

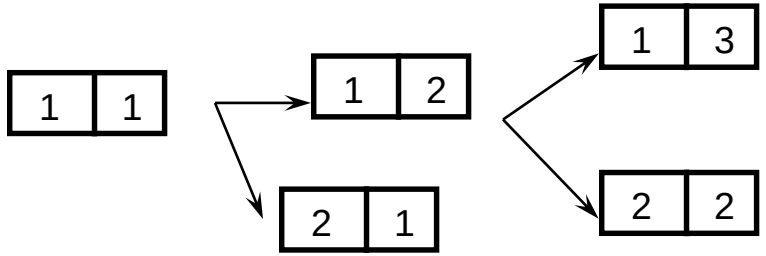
Algoritmos genéticos

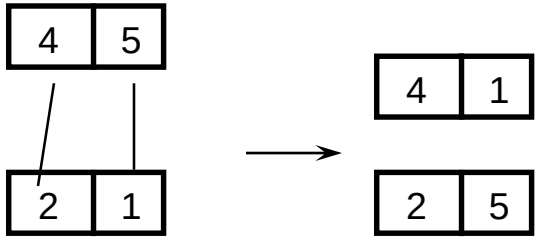
```
t := 0                                          // tempo inicial
P := população inicial de indivíduos            // conjunto de soluções
Avalia aptidão de cada indivíduo de P           // função objetivo
Enquanto critérioDeParada(MaxGerações, fitness(P)),
    não é satisfeito faça
    t := t + 1                                  // incrementa tempo
    P´ := seleciona(P)                          // população mais adequada
    P´´ := reproduz(P´)                         // gera descendentes
    P´´´ := muta(P´´)                           // diversifica-os
    Avalia aptidão de P´´´
    P := substitui(P,P´´´)                      // escolhe os sobreviventes
```

Exemplo: Quanto de açúcar e farinha de trigo para fazer um bom biscoito?

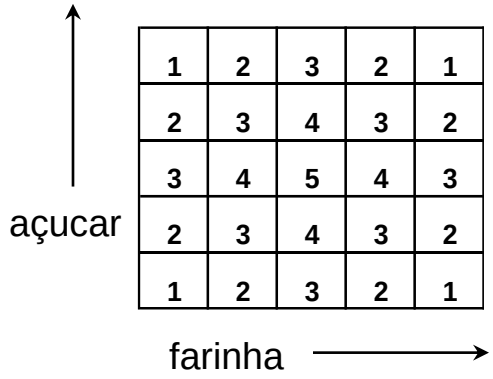
indivíduo: 

3 g de açucar e 1g de farinha de trigo

Mutação: +/- 1 num intervalo de 1 a 5 

Reprodução: 

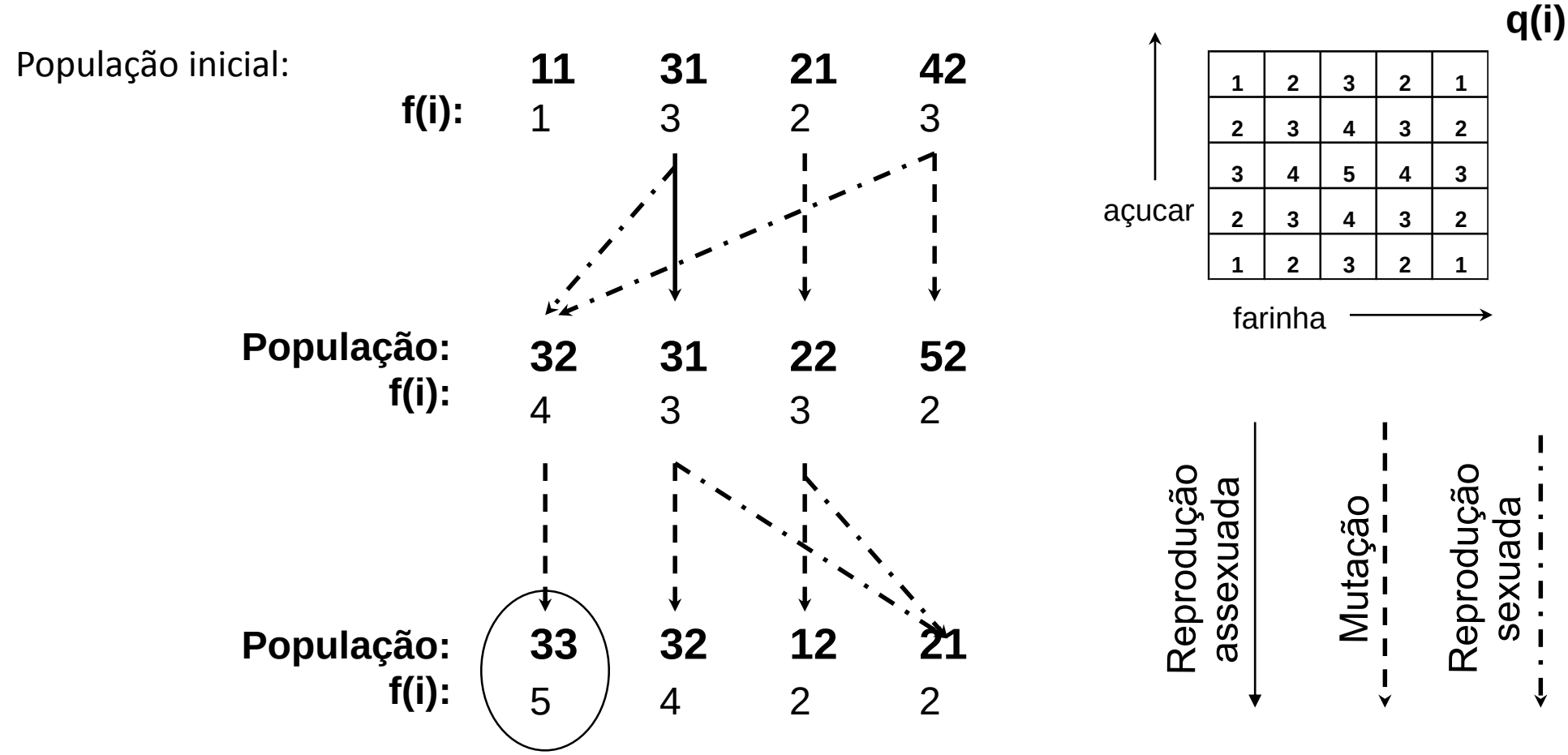
q(i)



Função objetivo: f(i) = q(i) / ∑j q(j)

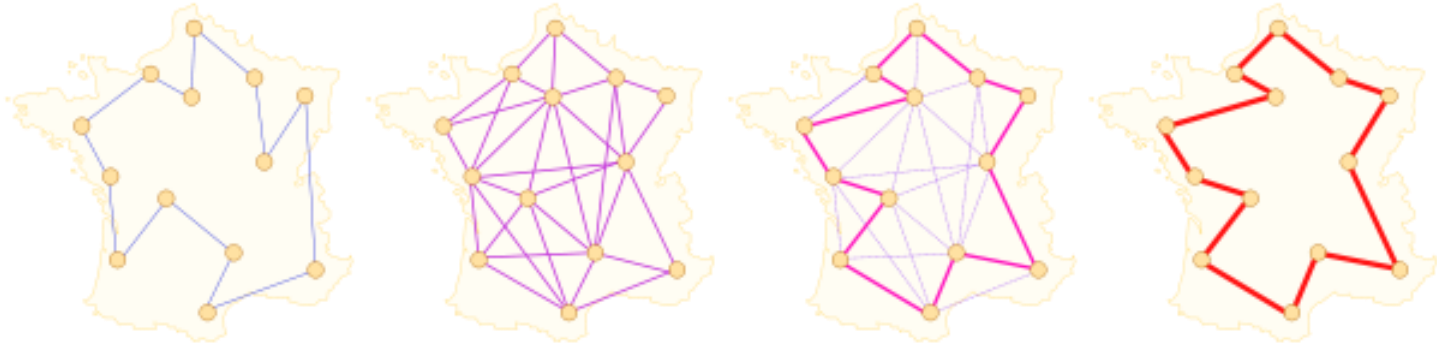
Seleção/substituição: nova geração substitui a antiga (max. 4 indivíduos)

Exemplo (cont.)



Ex#2: Problema do Caixeiro Viajante (TSP)

- Um caixeiro viajante deve visitar N cidades em sua área de vendas
- O caixeiro começa de uma base, visita cada cidade uma única vez e retorna à sua cidade no final
- A cada viagem esta associado um custo
  - O caixeiro deve percorrer a rota mais curta



TSP: Implementação

- Cromossomo - Enumerado

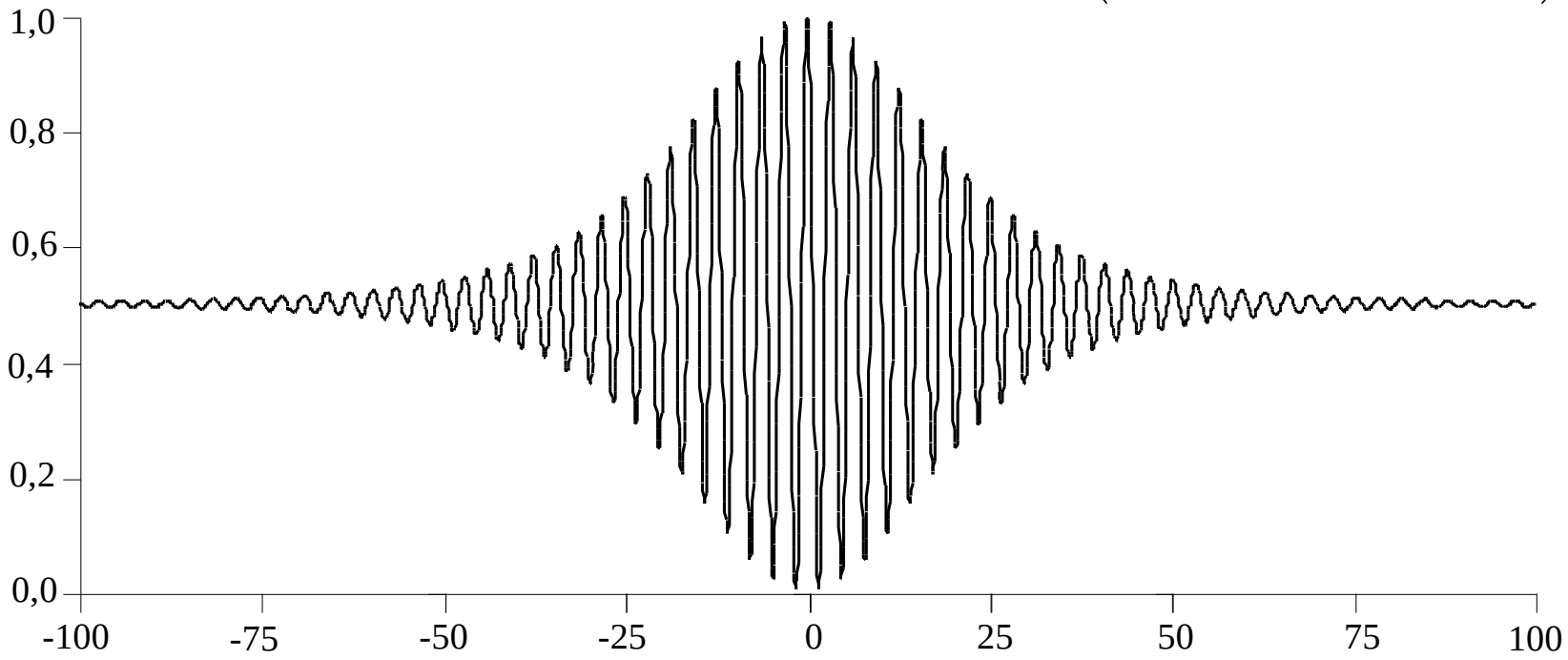
4	5	2	1	6	3

- Objetivo - minimizar o caminho total (tour)
  - soma de cada elemento do vetor

- Applet exemplo animado
  - http://www.cs.washington.edu/education/courses/cse473/06sp/GeneticAlgDemo/tspexample.html
  - http://www.professor.webizu.org/ga/tspexample.html (pt-BR)

Ex#3: Otimização de Função

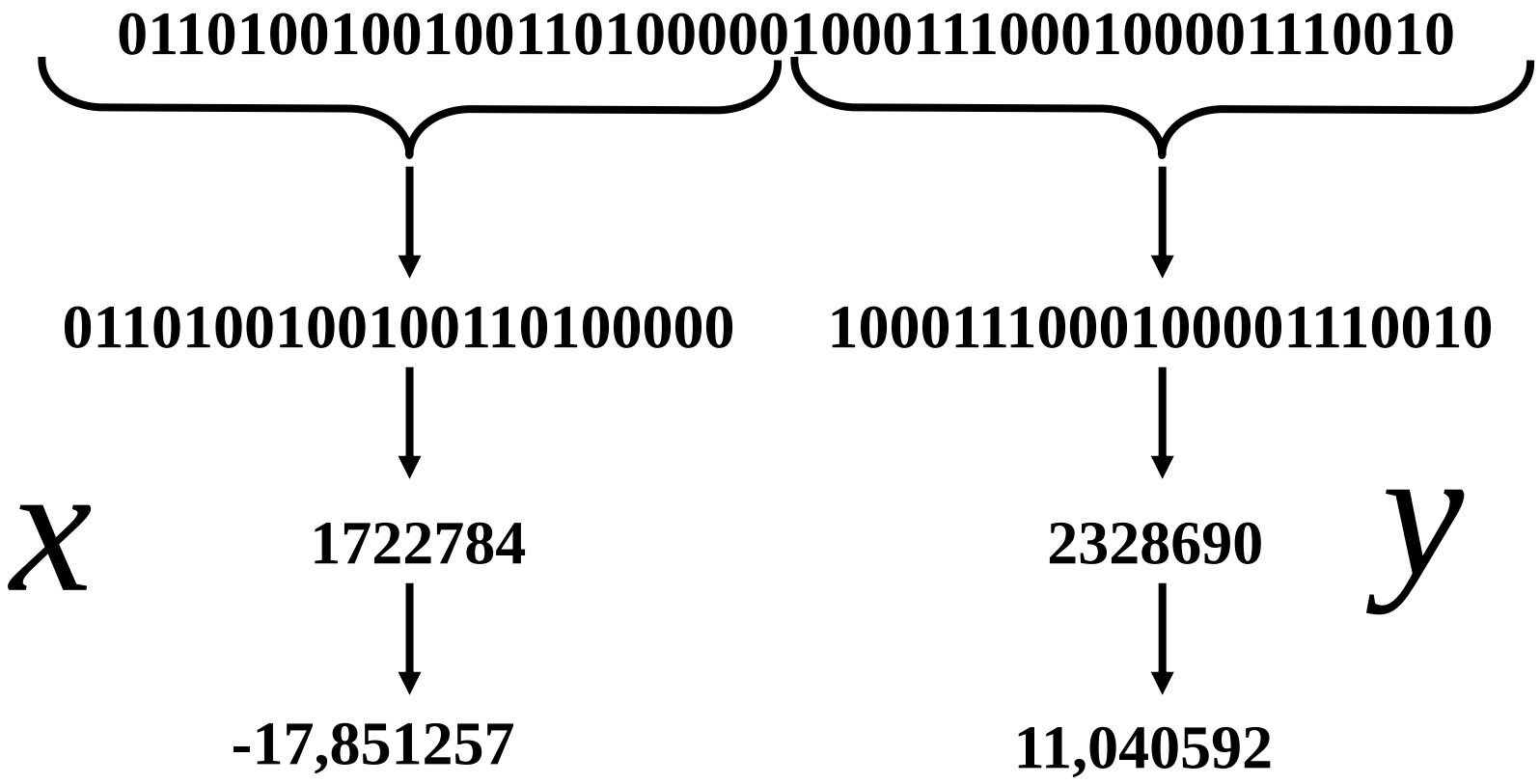
FUNÇÃO OBJETIVO : $$f(x,y) = 0{,}5 - \frac{\left(\text{sen}\sqrt{x^2+y^2}\right)^2 - 0{,}5}{\left(1{,}0 + 0{,}001\left(x^2+y^2\right)\right)^2}$$



Aptidão = Função Objetivo

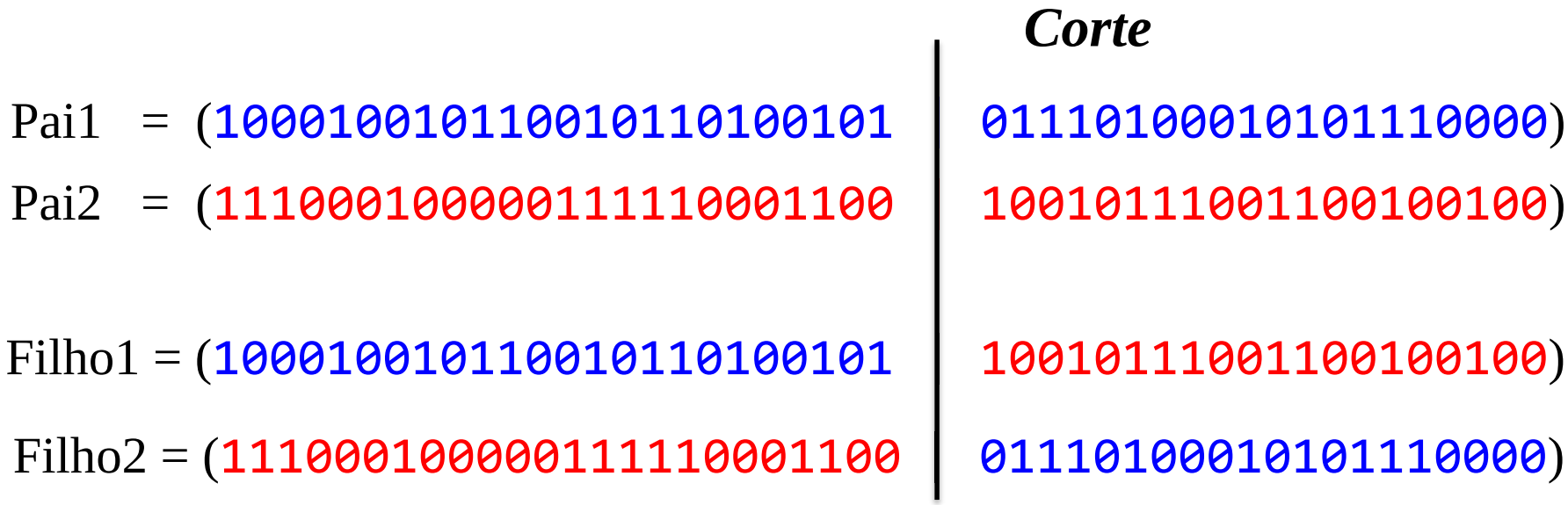
Exemplo de cromossomo

- Decodificar a cadeia de bits para o intervalo [-100,100]



Exemplo de Crossover

- Ponto de corte é aleatório
- Taxa de crossover (prob.) ~ 0,6 a 0,9



Exemplo de mutação

- Cada bit sofre mutação com taxa de mutação (prob.) ~ 0,001 a 0,01

Pai2 = (11001100**0**0110000000010001000000000000**1**001)

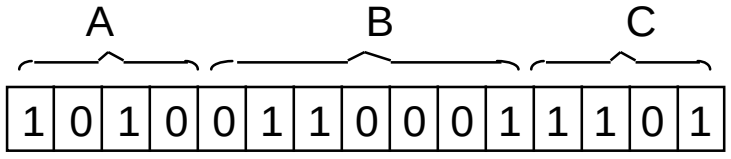
Filho2 = (11001100**1**0110000000010001000000000000**0**001)

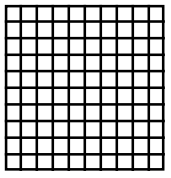
Questões centrais

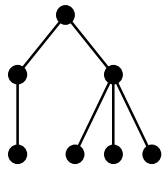
- Como representar os indivíduos?
- Quem é a população inicial?
- Como definir a função objetivo?
- Quais são os critérios de seleção?
- Como aplicar/definir o operador de reprodução?
- Como aplicar/definir o operador de mutação?
- Como garantir a convergência e ao mesmo tempo a solução ótima?

Representação

- Única restrição: determinar de modo não ambíguo uma solução
- Exemplos comuns:
  - cadeia
  - vetor de bits
  - matrizes
  - árvores, ...







Representação

- Representações mais gerais:
  - conjunto de elementos que podem ser bits, números reais, símbolos, **regras**, outros conjunto de elementos, ...
  - Indivíduo = regra e população = base de regras
  - indivíduo = base de regras e população = agentes
- Exemplos
  - ex. SAGACE: jogo
    - indivíduo = (eu, adv-min, jeton, joga)
  - ex. SAMUEL: agentes reativos
    - indivíduo1 = (r1, r7, r10, r15, r21)
    - indivíduo2 = (r5, r7, r11, r13, r22),

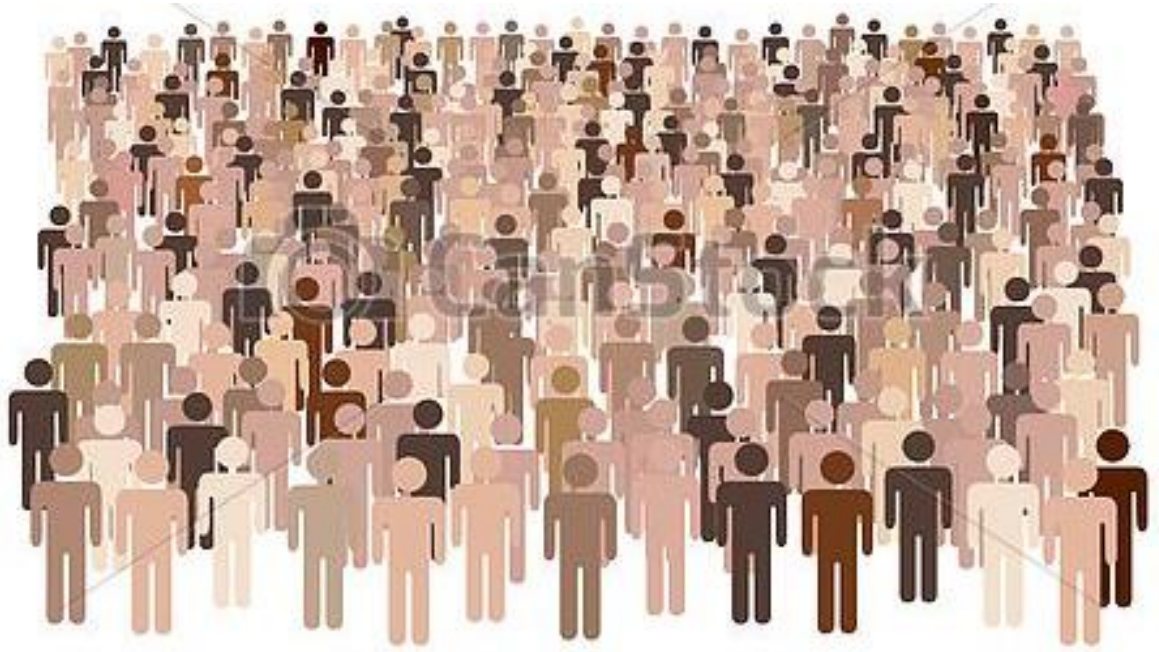
Representação: conhecimento do domínio

- Representações mais estruturadas
  - implica em redefinição dos operadores genéticos (maior complexidade)
  - ex. cadeia x matriz de matrizes

- Representações mais ricas
  - não necessariamente evidentes de se definir
  - ex. Em SAGACE o indivíduo final tem 16 parâmetros
    - (eu-min, eu-max, adv-min,..., jeton,..., nb-utilizada, fitness..., joga)

População inicial

- Aleatoriamente escolhida
- Trade-off: velocidade de convergência x variedade
- Na prática, 100 indivíduos (100^3 *patterns*)

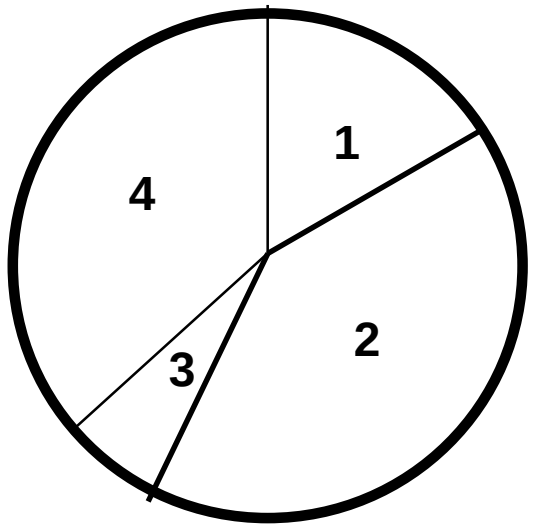


Seleção

- Objetivo:
  - propagar material genético dos indivíduos mais adaptados
- Problemática da convergência prematura (trade-off rapidez x diversidade):
  - Um indivíduo super adaptado no começo não deve ser valorizado demais
  - indivíduos ruins no começo não podem ser desprezados
- Tipos:
  - Ranking (os n mais adaptados)
  - Roda da roleta (ranking probabilístico): **pselect(i) = f(i) / $\sum$ f(j)**
  - Torneio (eliminatórias 2 a 2)
  - Outros: stochatic reminder sampling, ...

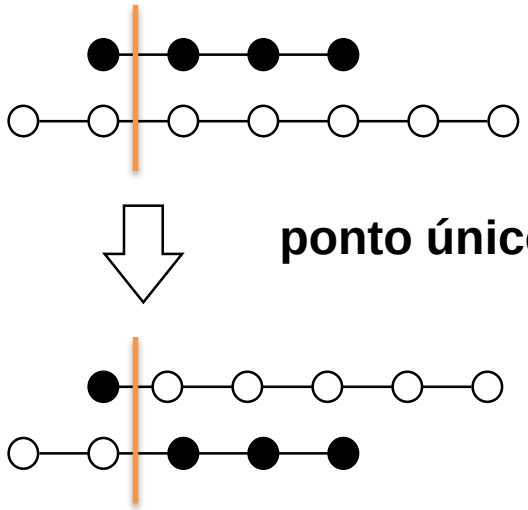
Roda da roleta

n.	cadeia	aptidão	% do total
1	01101	169	14,4
2	11000	576	49,2
3	01000	64	5,5
4	10011	361	30,9
Total		1170	100,0

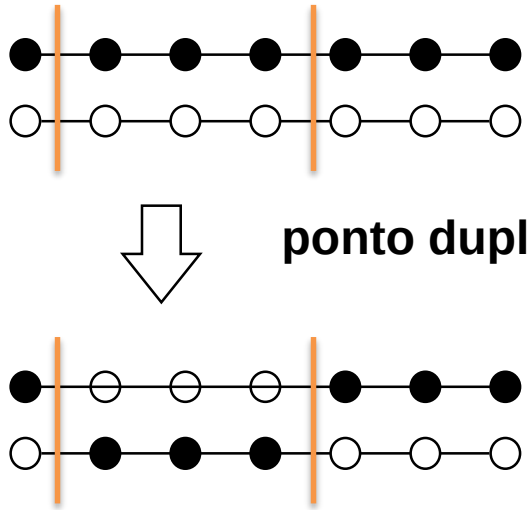


Reprodução/recombinação

- Função:
  - combinar e/ou perpetuar material genético dos indivíduos mais adaptados
- Tipos:
  - assexuada (=duplicação)
  - sexuada (crossover)



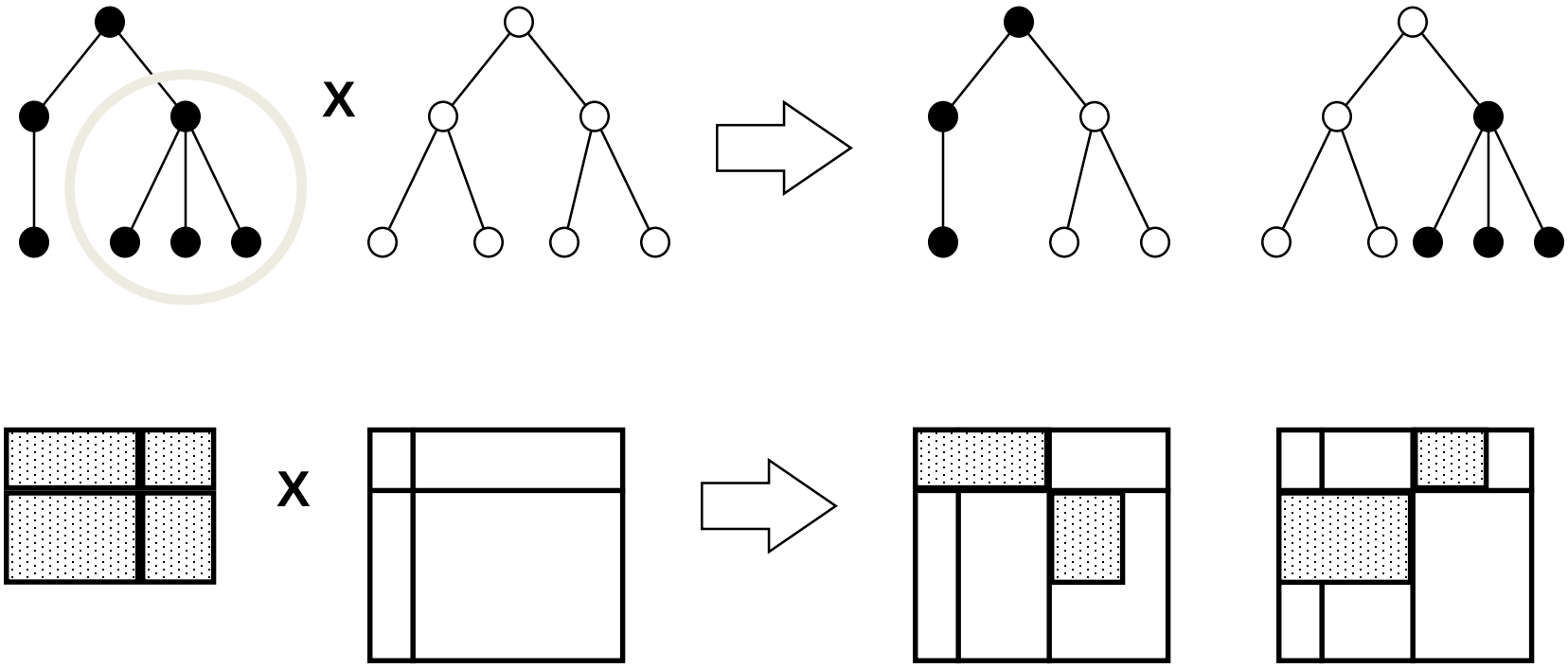
ponto único



ponto duplo

Reprodução (2)

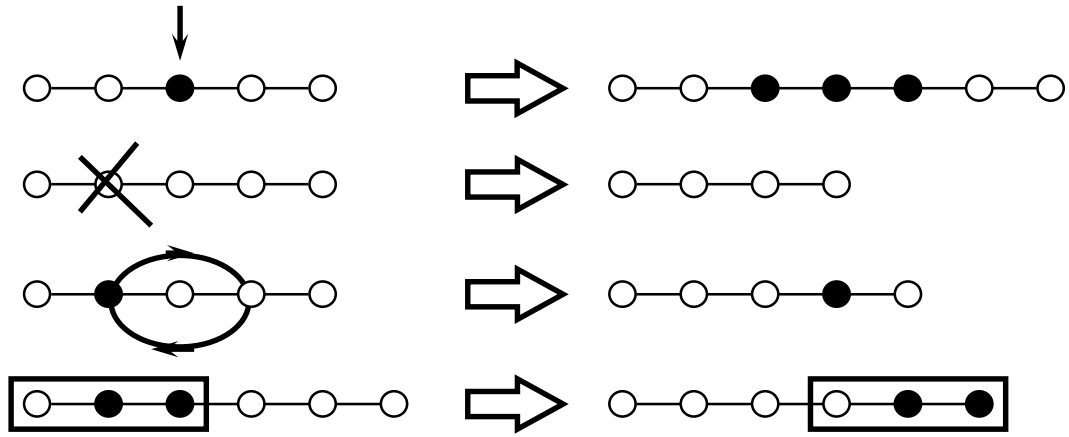
- Quanto mais “estruturada” a representação mais difícil de definir o cruzamento



Mutação

- Objetivo:
  - gerar diversidade (p/ escapar de ótimos locais)

- Tipos:
  - generativa
  - destrutiva
  - swap
  - swap de sequência



- Observação:
  - Existe uma “taxa de mutação” (ex. % da população selecionada) que diminui com o tempo para garantir convergência

Substituição

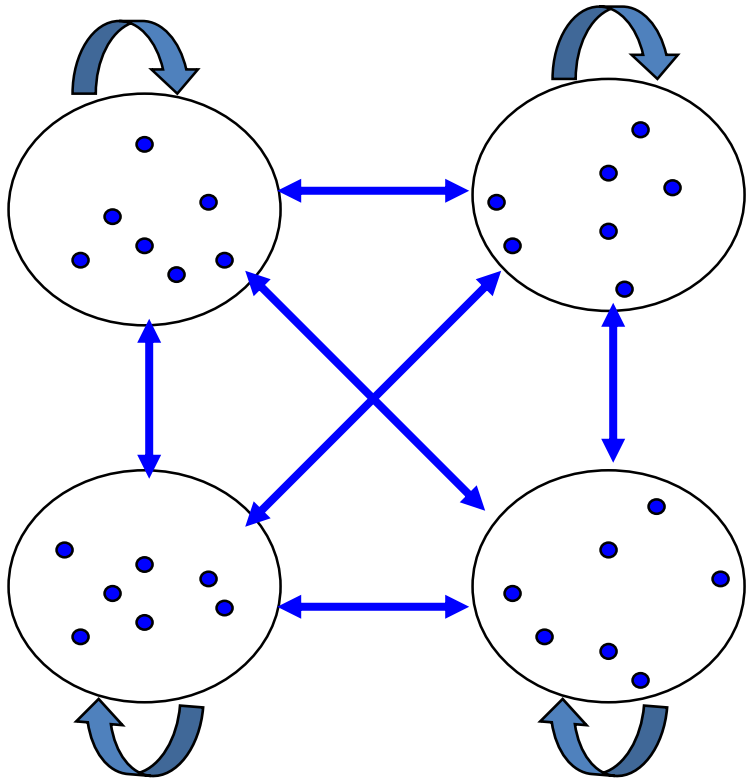
- Objetivo:
  - garantir uma convergência adequada
- Tipos:
  - simples (m,l) : a nova geração SUBSTITUI a antiga
  - elitista ou steady-state (m + l): a nova geração se MISTURA com a antiga.
- Critérios de substituição no caso elitista:
  - os piores
  - os mais semelhantes
    - para evitar convergência prematura
  - os melhores
  - os pais
  - aleatoriamente, ...

Evitando a convergência prematura

- Crowding
  - substitui os mais semelhantes
- Escalonamento
  - Linear fitness scaling: normaliza a função de avaliação "bruta"
- Sharing
  - diminui score dos ind. + semelhantes
- Janelamento:
  - escore mínimo para todos
- Algoritmos genéticos paralelos

Algoritmos genéticos paralelos

- K Populações são iniciadas evoluem paralelamente



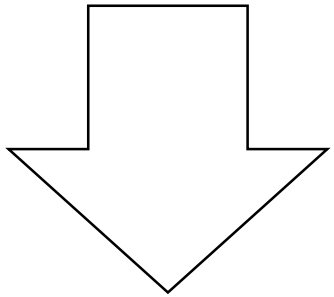
- A cada *n*-ésima geração, as populações trocam indivíduos

Porque converge?

- Esquemas
  - “sub-partes” comuns recorrentes
- Teorema dos esquemas
  - o número de esquemas bem adaptados cresce exponencialmente
- Building-blocks hypothesis:
  - a otimalidade é obtida por justaposição de pequenos esquemas altamente adaptados
- Paralelismo implícito:
  - Tomando o alfabeto {0,1,*}, existem 2^k cadeias de k bits mas 3^k possíveis esquemas
  - n indivíduos @ n^3 esquemas úteis

Paralelismo implícito: exemplo

n.	cadeia	aptidão	% do total
1	01101	169	14,4
2	11000	576	49,2
3	01000	64	5,5
4	10011	361	30,9



esquema 1****, melhor que 0****

Computação Evolutiva

- Técnicas
  - Algoritmos genéticos
  - Programação genética
  - Estratégia evolutiva
  - Programação evolutiva
- O que varia
  - Critérios de escolha dos sobreviventes
  - Operadores de “transformação” dos indivíduos
  - Representação dos indivíduos

Algoritmos evolutivos

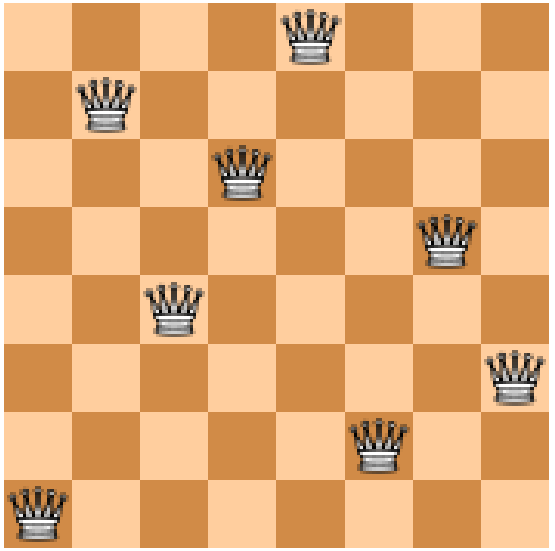
	Indivíduos	*operadores*	*seleção/subst..*
Algoritmos Genéticos	soluções	mut. + repr.	
Programação genética	programas	mut. + repr.	
Estratégia evolutiva.	soluções	mut.	(1,1)
Programação evolutiva	soluções	mut.	overlap, 1/2

Exemplos de aplicações

- Roteamento de Telecomunicações
- Planejamento dos Jogos Olímpicos
- Avaliação de Crédito e Análise de Risco
- Particionamento de circuitos
- Jogos

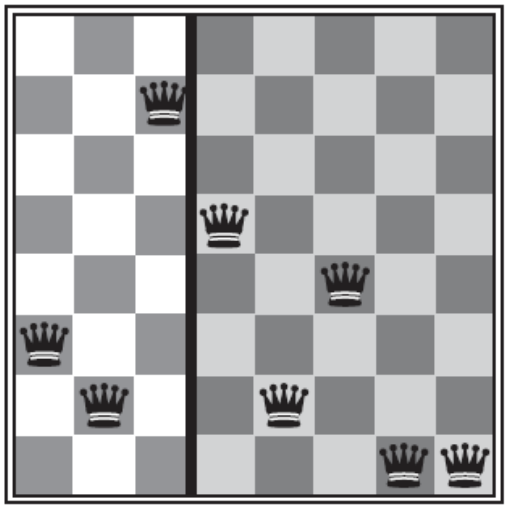
Algoritmos Genéticos - Exemplo 1

- **Problema:** 8 Rainhas

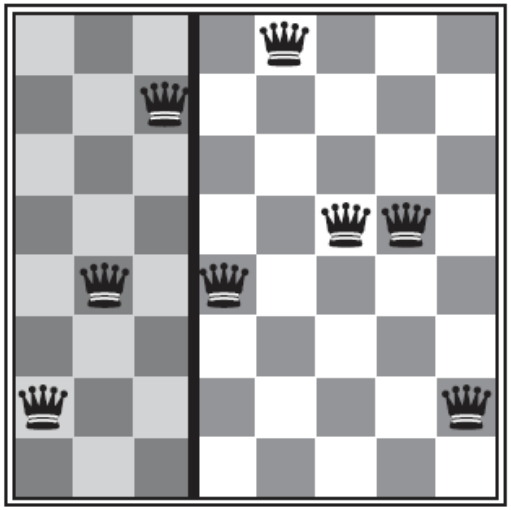


- **Como representar os indivíduos?**
  - 8 dígitos – cada um representado a posição da rainha em sua coluna.
  - **Exemplo:** (1, 7, 4, 6, 8, 2, 5, 3)
- **Qual a função de avaliação?**
  - Número de pares de rainhas não sedo atacadas.

Algoritmos Genéticos - Exemplo 1

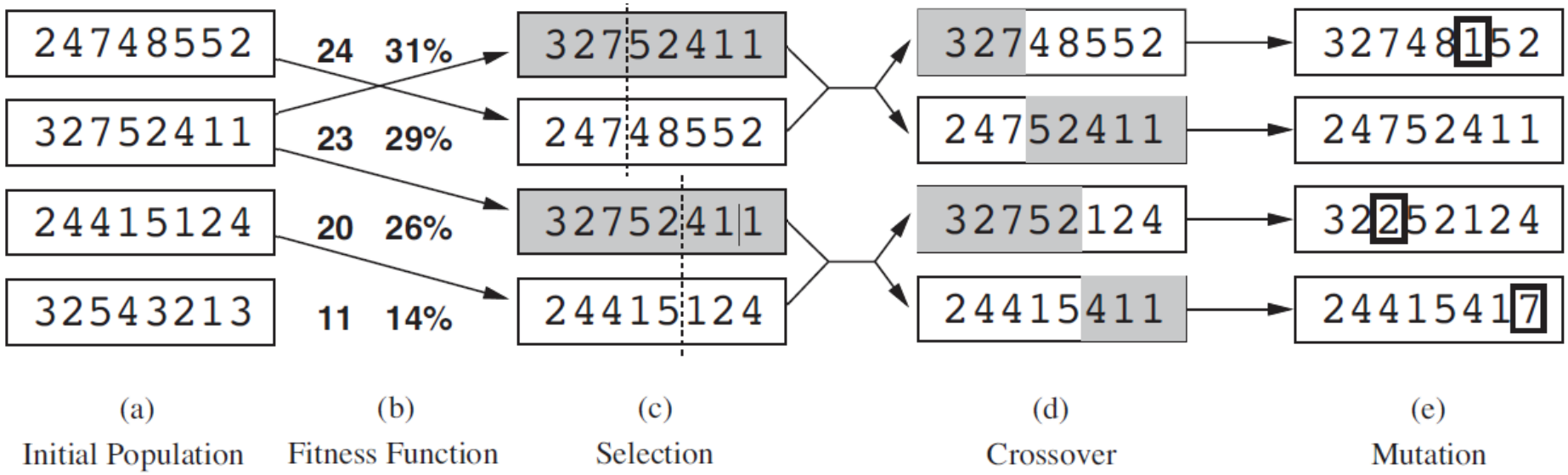


(3, 2, 7, 5, 2, 4, 1, 1) = 23



(2, 4, 7, 4, 8, 5, 5, 2) = 24

Algoritmos Genéticos - Exemplo 1



Algoritmos Genéticos – Exemplo 2

- **Problema do caixeiro viajante:** Deve-se encontrar o caminho mais curto para percorrer *n* cidades sem repetição.

- **Como representar os indivíduos?**
  - Cada indivíduo pode ser representador por uma lista ordenada de cidades, que indica a ordem em que cada uma será visitada.
  - **Exemplo:** (3 5 7 2 1 6 4 8)

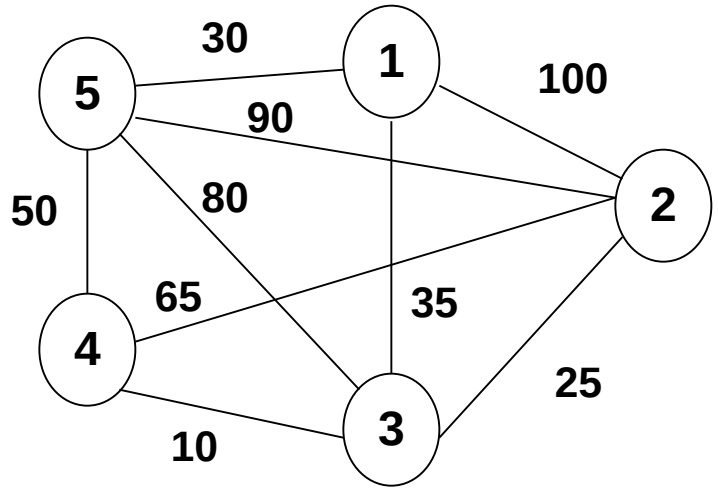
Algoritmos Genéticos – Exemplo 2

- Cada cromossomo tem que conter **todas as cidades** do percurso, **apenas uma vez**.

- **Considerando 8 cidades:**
  - **Cromossomos válidos:** (1 2 3 4 5 6 7 8), (8 7 6 5 4 3 2 1), (1 3 5 7 2 4 6 8)...
  - **Cromossomos inválidos:** (1 5 7 8 2 3 6) - Falta a cidade 4, (1 5 7 8 2 3 6 5) - Falta a cidade 4 e a cidade 5 está representada 2 vezes...

Algoritmos Genéticos – Exemplo 2

- Qual a função de avaliação?
  - A **função de avaliação** consiste em somar todas as distâncias entre cidades consecutivas.
  - **Exemplo:**



  - O cromossomo (1 3 5 4 2) tem avaliação igual a 35+ 80 + 50 + 65 = 230

Algoritmos Genéticos – Exemplo 2

- **Recombinação (uniforme):**
  - Para cada gene é sorteado um número zero ou um.
    - Se o valor sorteado for 1, um filho recebe o gene do primeiro pai e o segundo filho o gene do segundo pai. Se for 0, ocorre o inverso.
  - Pai1 (3 5 7 2 1 6 4 8)
  - Pai2 (2 5 7 6 8 4 3 1)

1) Gera-se uma string de bits aleatória do mesmo tamanho que os pais: 1 0 0 1 0 1 0 1
2) Copia-se para o filho 1 os elementos do pai 1 referentes àquelas posições onde a string de bits possui um 1: 3 _ _ 2 _ 6 _ 8
3) Elementos não copiados do pai1: _ 5 7 _ 1 _ 4 _
4) Permuta-se essa lista de forma que os elementos apareçam na mesma ordem que no pai 2 e copia-se eles para dentro do Filho1: 3 5 7 2 1 6 4 8

Algoritmos Genéticos – Exemplo 2

- **Mutação:**

Individuo (3 5 7 2 1 6 4 8)

- Escolhem-se dois elementos aleatórios dentro do cromossomo e trocam-se as suas posições:

  (3 5 7 2 1 6 4 8)

- Novo individuo mutante:

  (3 1 7 2 5 6 4 8)

Balanço

Vantagens

- Simples (várias representações, 1 algoritmo) e pouco sensível a pequenas variações no set-up
- Vasto campo de aplicações (inclusive em NN)
- Ainda custa caro mas pode ser paralelizado facilmente

Desvantagens

- Como o método é basicamente numérico nem sempre é fácil introduzir conhecimento do domínio.

Links

- *Course on Genetic Algorithms*
  - http://gal4.ge.uiuc.edu/ge493/ge493.top.html
- Intro to GAs (slides)
  - http://lancet.mit.edu/~mbwall/presentations/IntroToGAs/
- GA faq
  - http://www.cis.ohio-state.edu/hypertext/faq/usenet/ai-faq/genetic/top.html
- Links para *Genetic Algorithms*
  - http://www.ics.hawaii.edu/~sugihara/research/link-dga.html