Inteligência Computacional

Algoritmos Evolutivos

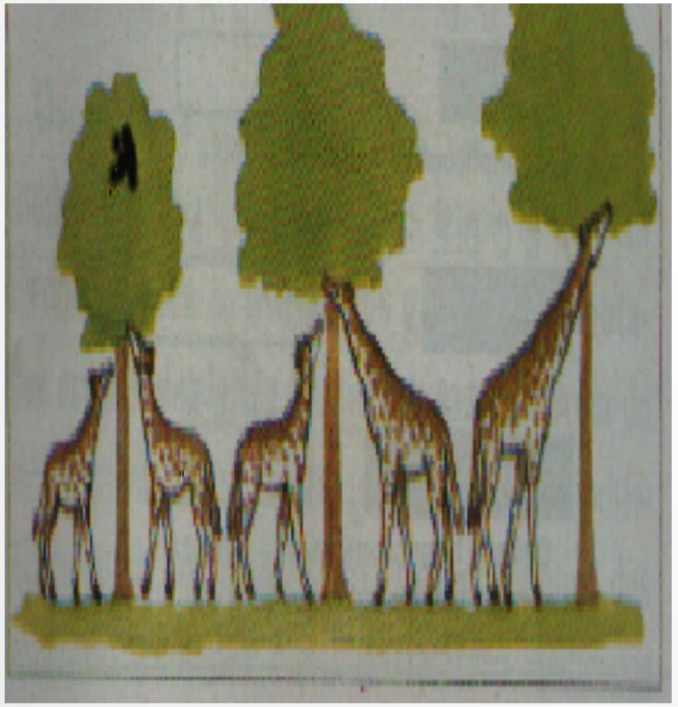
Profa Ana Carolina Lorena

2º semestre 2019

Algoritmos Evolutivos

Engloba métodos e técnicas computacionais inspirados:

- na teoria da evolução das espécies, de seleção natural (Darwin)
- na Genética iniciada por Mendel



Bases da evolução:

- diversidade é gerada por cruzamento e mutações
- os seres mais adaptados ao seus ambientes sobrevivem
- as características genéticas de tais seres são herdadas pelas próximas gerações

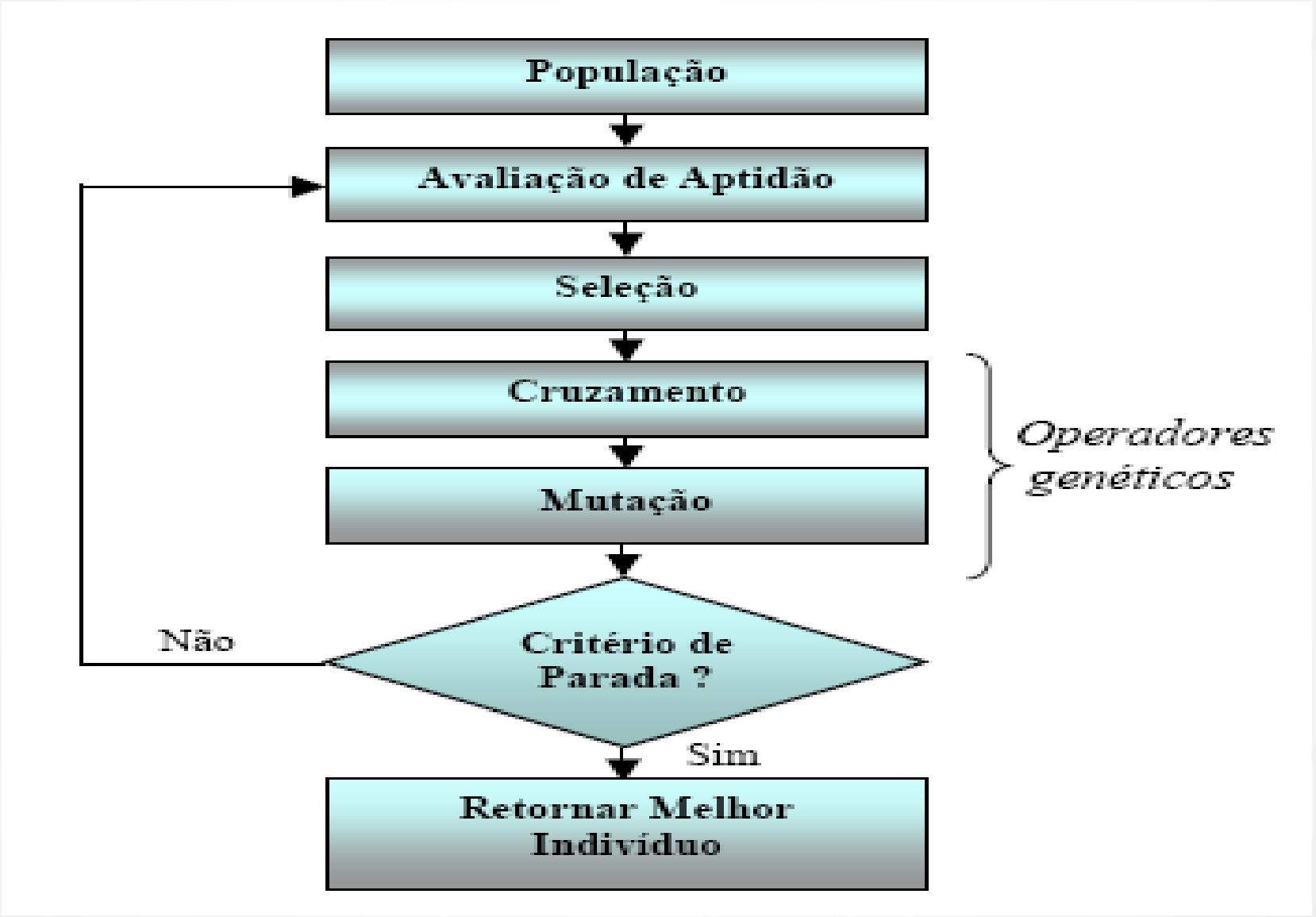
Algoritmos Genéticos

Os algoritmos genéticos usam como base, e procuram combinar:

- *A teoria da evolução das espécies* - a sobrevivência das estruturas/soluções mais adaptadas a um ambiente/problema

- *Estruturas genéticas* - utiliza a conceitos de hereditariedade e variabilidade genética para troca de informações entre as estruturas, visando a sua melhoria

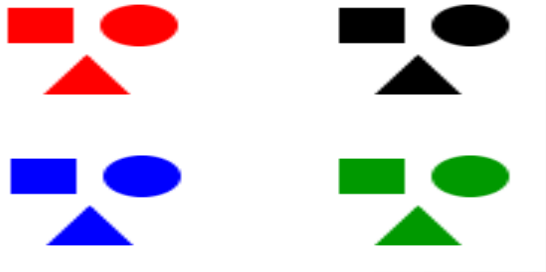
Algoritmo Genético

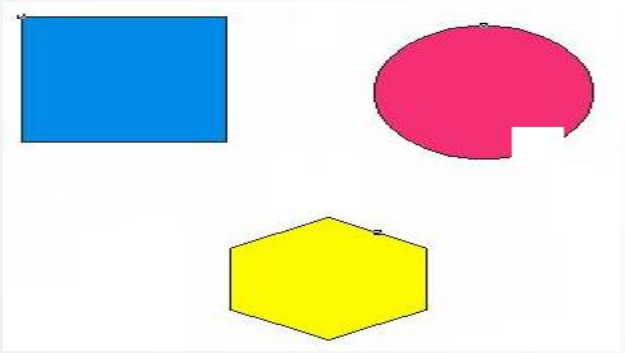


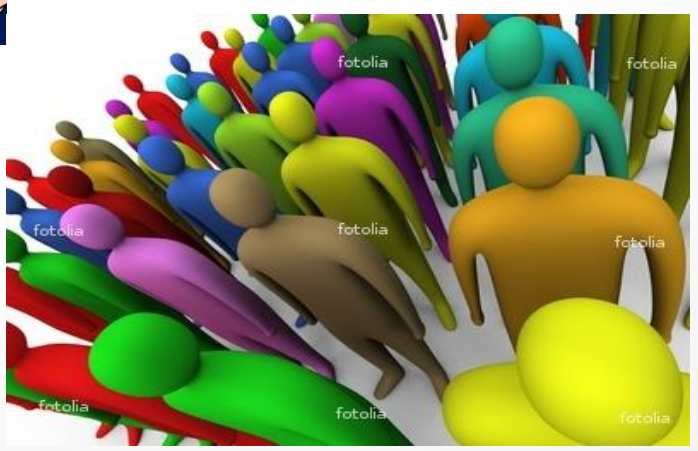
População

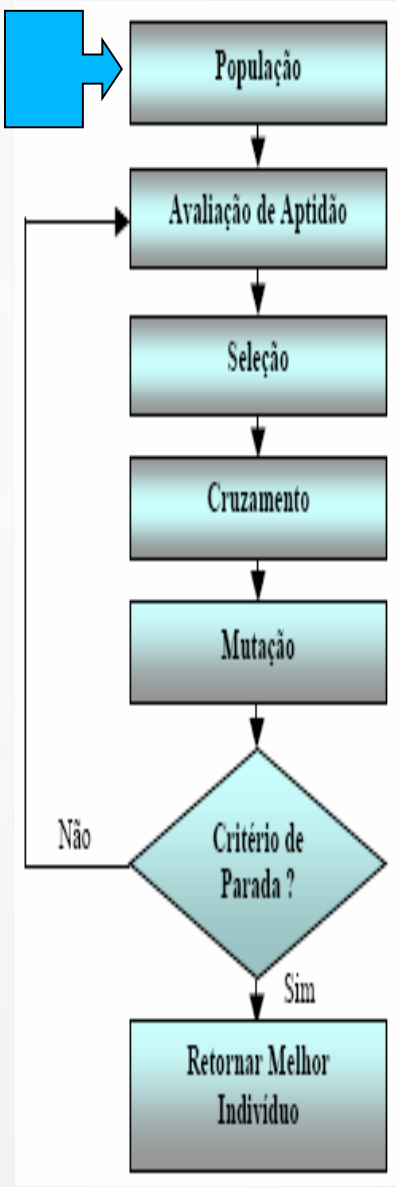
A população de um algoritmo genético é o conjunto de indivíduos que estão sendo cogitados como solução

Cada indivíduo é uma possível solução do problema









Indivíduo

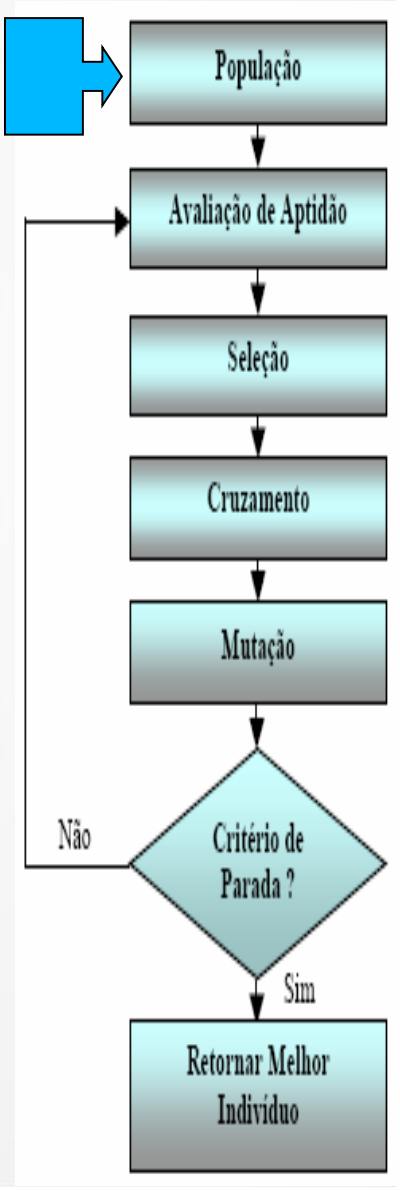
Um indivíduo no AG é um cromossomo

Ou seja, um indivíduo é um conjunto de atributos da solução

Geralmente é uma cadeia de bits que representa uma solução possível para o problema

Outras representações são possíveis

Boa representação depende do problema



Exemplo: população de tamanho N=5
Geração de indivíduos, com seus cromossomos
Cada elemento do vetor é um gene, um atributo da solução

```
Indivíduo 1 = [1 1 1 0 1]
Indivíduo 2 = [0 1 1 0 1]
Indivíduo 3 = [0 0 1 1 0]
Indivíduo 4 = [1 0 0 1 1]
```

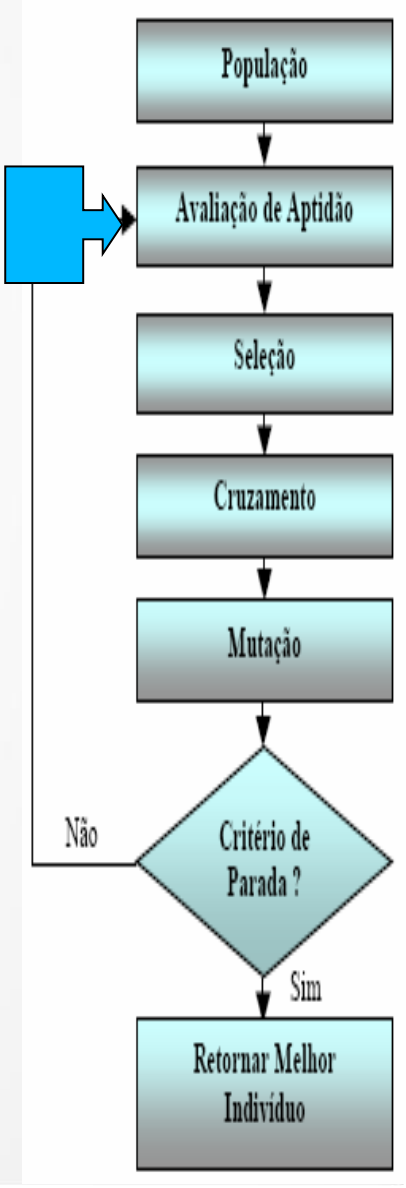
Função de aptidão

A função de avaliação, função de *fitness*, determina uma nota a cada indivíduo

Esta nota avalia quão boa é a solução que este indivíduo representa

Por exemplo, o objetivo de um AG pode ser maximizar o número de 1s

Indivíduos	Função de aptidão (*fitness*)
[11101]	4
[01101]	3
[00110]	2
[10011]	3
Aptidão média	3



Seleção

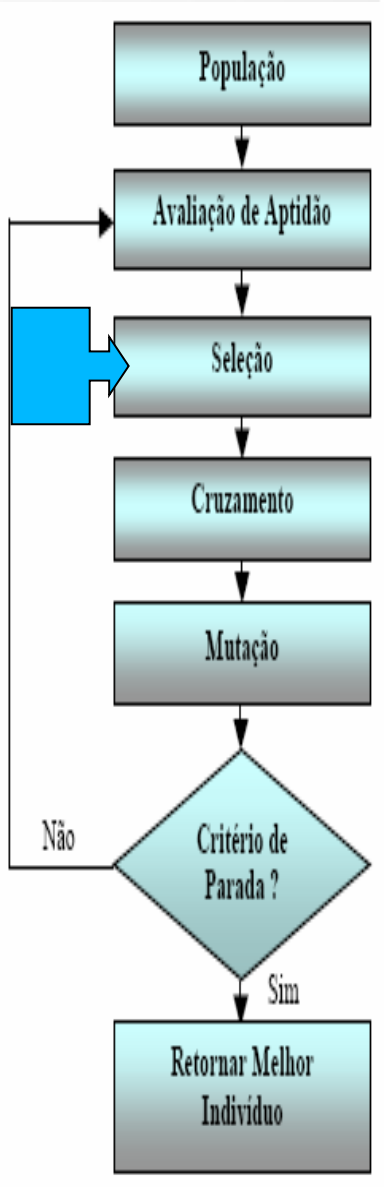
De acordo com a teoria de Darwin, o melhor sobrevivente para criar a descendência é selecionado

Há muitos métodos para selecionar o melhor cromossomo

Dentre eles:

- Seleção por roleta
- Seleção por torneio

A seleção dirige o AG para as melhores regiões do espaço de busca



Seleção

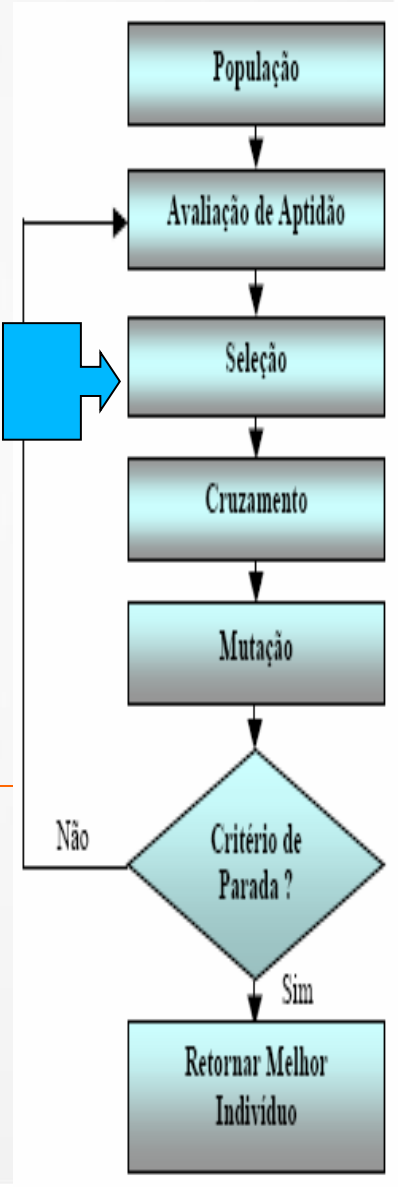
No método da roleta cada indivíduo tem uma probabilidade de ser selecionado, proporcional à sua aptidão

cromossomos	x	$f(x^2)$	
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%

$$1*2^4+1*2^3+0*2^2+0*2^1+1*2^0 = 16+8+0+0+1 = 25$$

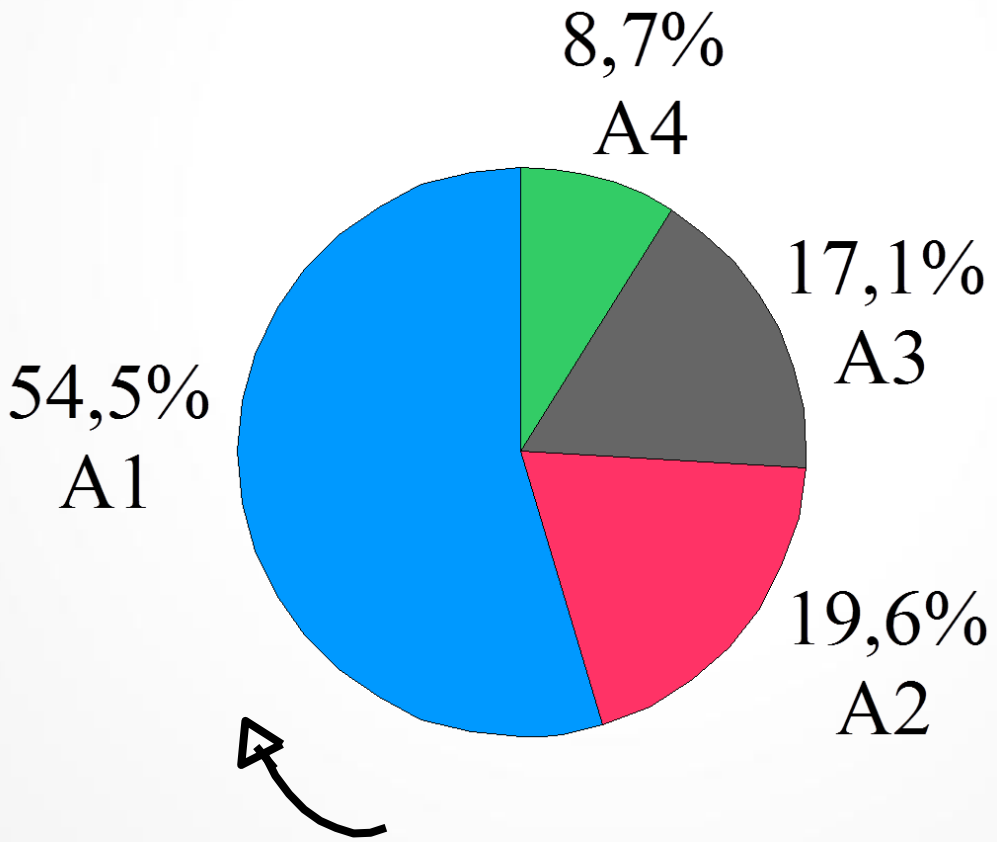
Probabilidade de seleção:

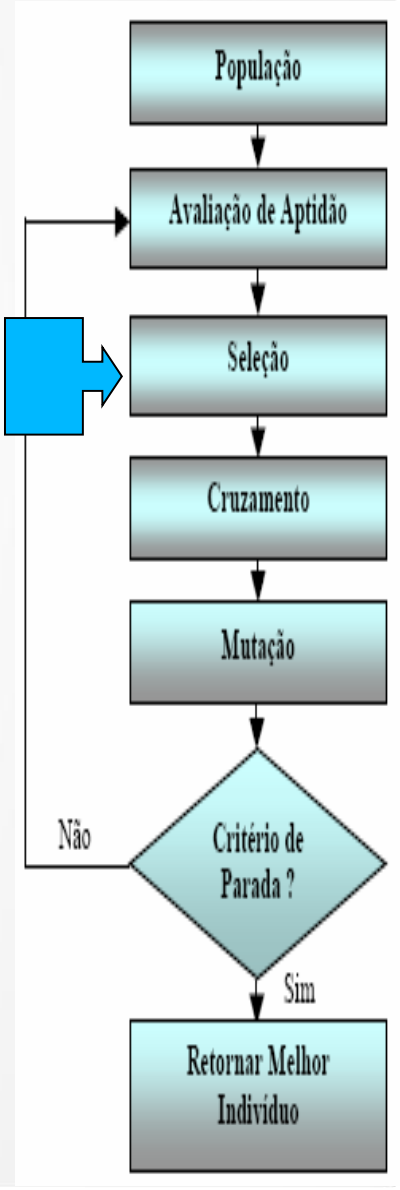
$$p_i = \frac{f(x_i)}{\sum_{k=1}^{N} f(x_k)}$$



Seleção por roleta

Para visualizar este método considere um círculo dividido em N regiões (tamanho da população), onde a área de cada região é proporcional à aptidão do indivíduo



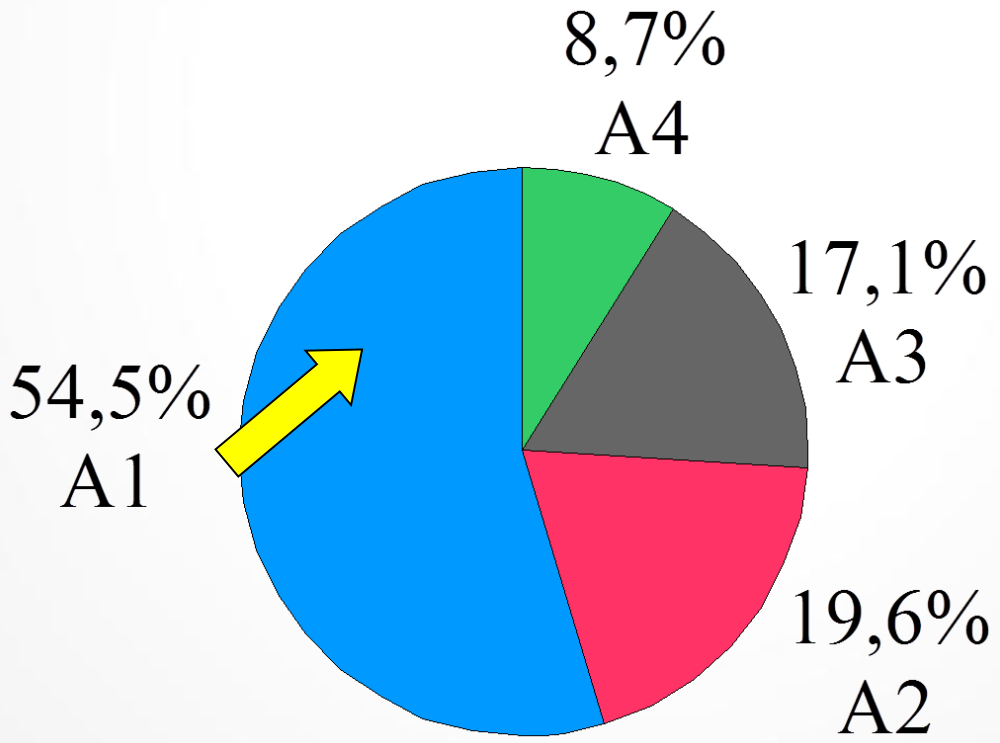


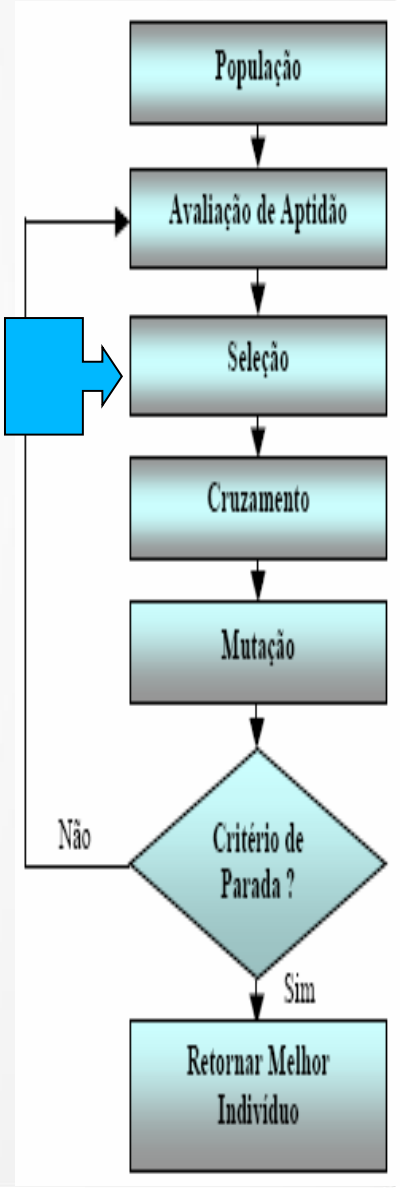
Seleção por roleta

Coloca-se sobre este círculo uma “roleta”

A roleta é girada um determinado número de vezes, dependendo do tamanho da população

São escolhidos como indivíduos que participarão da próxima geração, aqueles sorteados na roleta





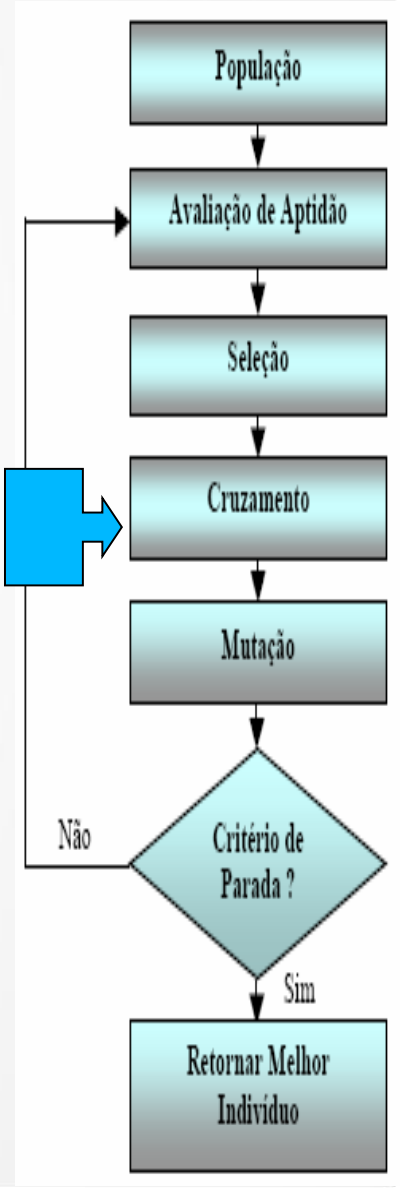
Operadores genéticos

Um conjunto de operações é necessário para que, dada uma população, se consiga gerar populações sucessivas que (espera-se) melhorem sua aptidão com o tempo

Estas operações são os **operadores genéticos**. São eles:

- Cruzamento
- Mutação

Os operadores genéticos permitem explorar áreas desconhecidas do espaço de busca



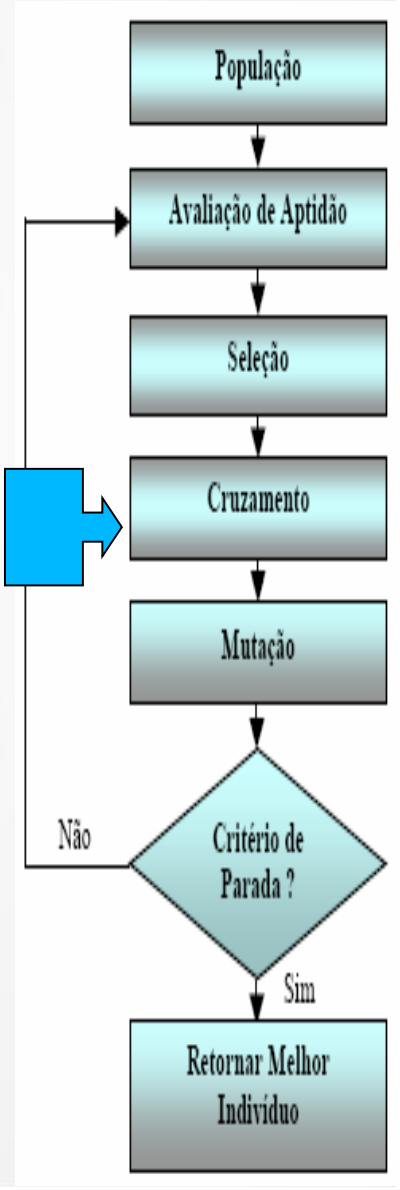
Cruzamento

O operador *crossover* (cruzamento) cria novos indivíduos, misturando características de dois indivíduos pais

O resultado desta operação é um indivíduo que potencialmente combine as melhores características dos indivíduos usados como base

Alguns tipos de cruzamento são:

- Cruzamento em um ponto
- Cruzamento em dois pontos

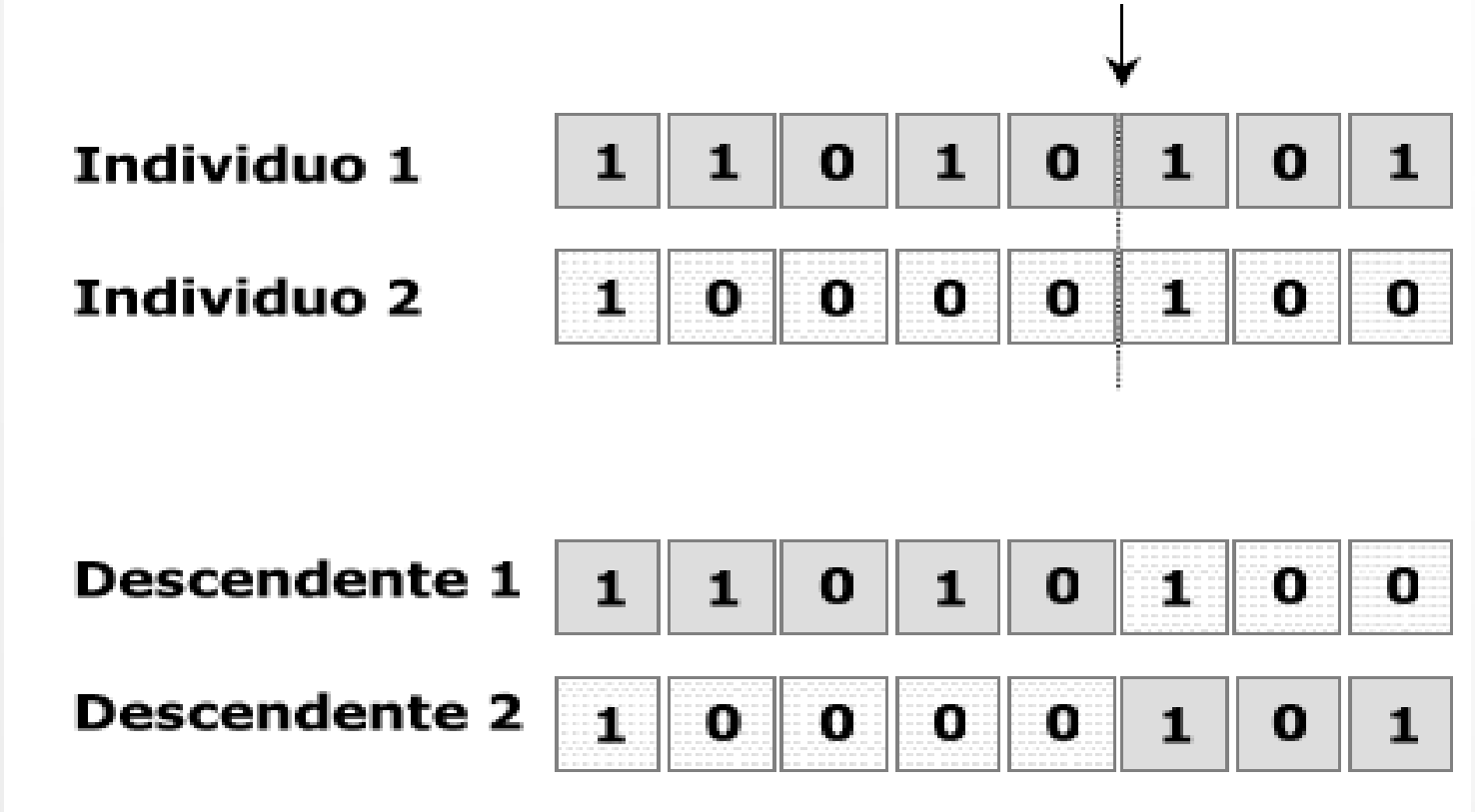


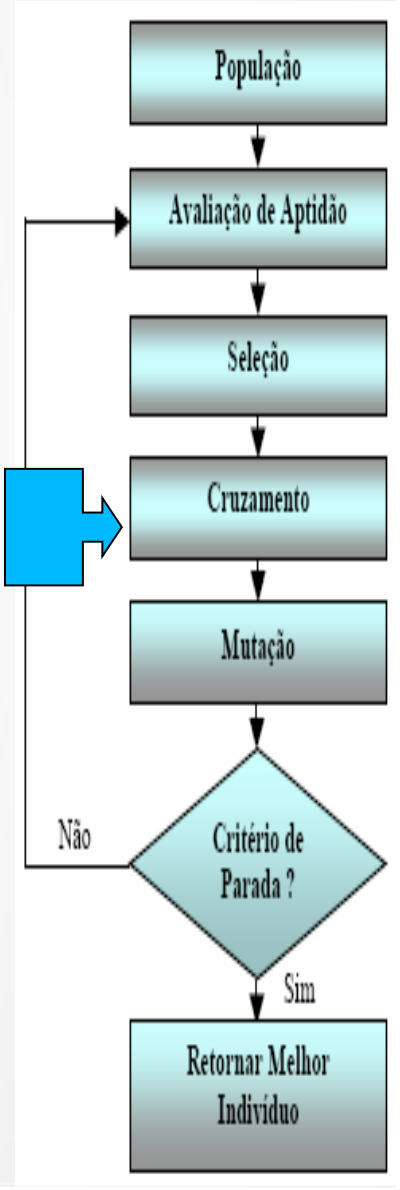
Cruzamento de um ponto

No cruzamento de um ponto divide-se cada progenitor em duas partes, em uma localidade `k` (escolhida aleatoriamente)

O descendente 1 consiste em genes 1 a `k-1` do progenitor 1, e genes `k` a `n` do progenitor 2

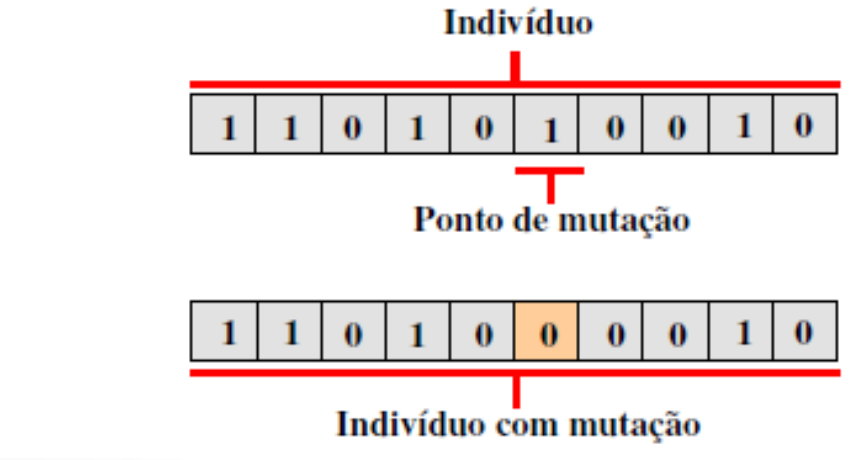
O descendente 2 é “reverso”





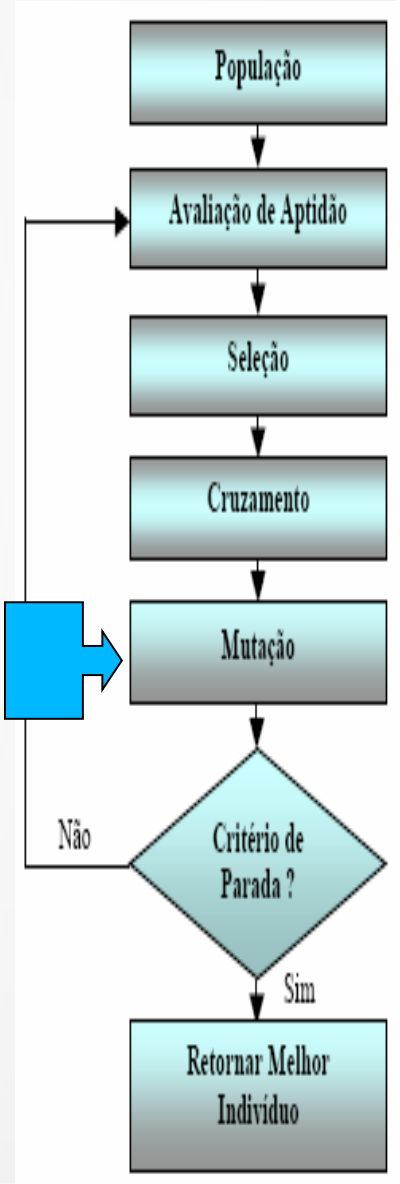
Mutação

A mutação modifica aleatoriamente alguma característica do indivíduo, sobre o qual é aplicada



O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população

Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca, possivelmente, não será zero



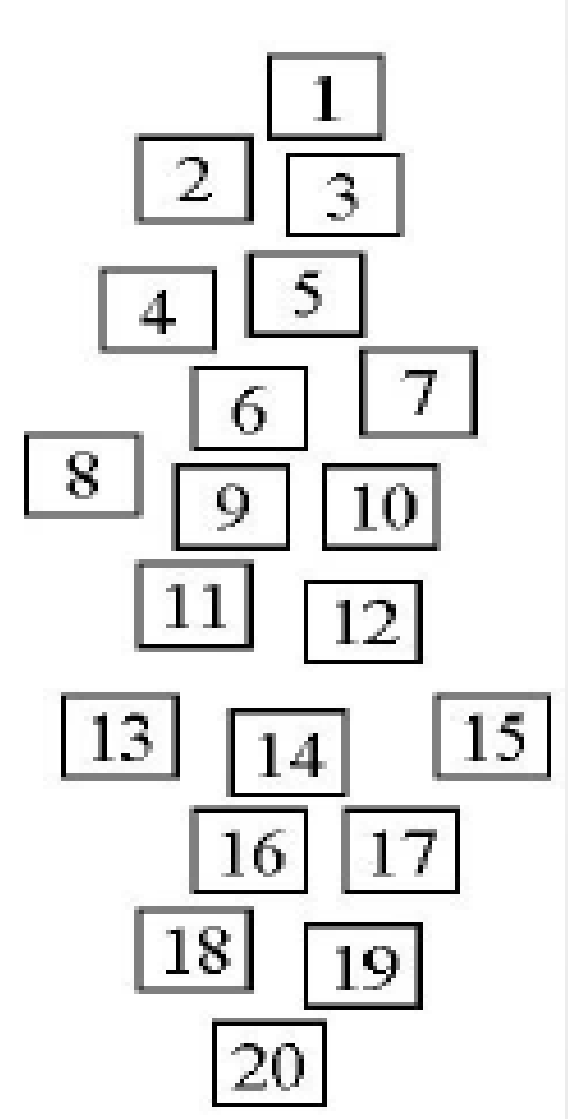
Gerações

Algoritmo é iterado até algum critério de parada

A cada passo, um novo conjunto de indivíduos é gerado a partir da população anterior

A este novo conjunto dá-se o nome de **geração**

Com a criação de uma grande quantidade de gerações que é possível obter resultados dos AGs



Algoritmo

Algoritmo_genético

-p = tamanho da população

-r = taxa de cruzamento

-m = taxa de mutação

Codificação e avaliação de aptidão são pontos chave

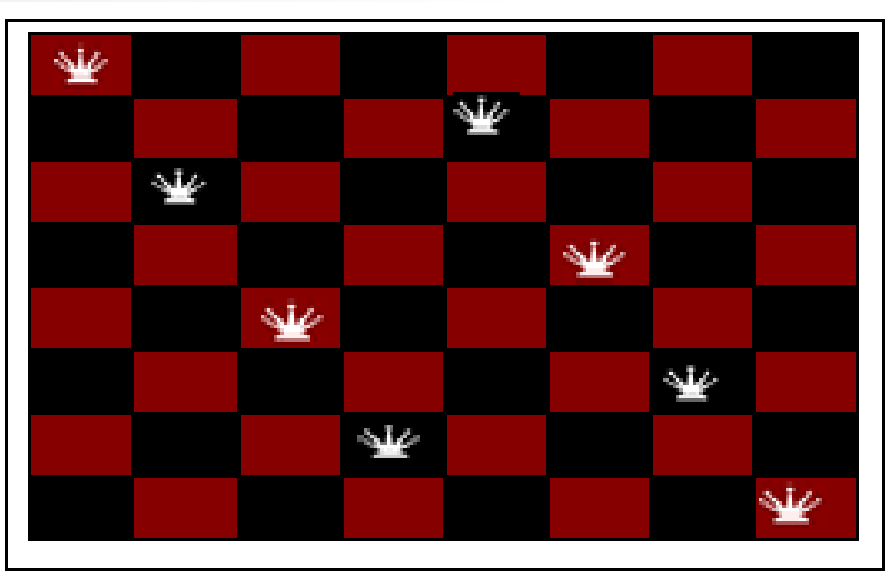
1. P← gerar aletoriamente *p* indivíduos
2. **Para cada** *i* em *P*, computar *Aptidão(i)*
3. **Enquanto** critério_parada não é atingido
    - 3.1 Selecionar *p* membros de *P* para reprodução
    - 3.2 Aplicar cruzamento a pares de indivíduos selecionados segundo taxa *r*, adicionando filhos em *PS*
    - 3.3 Realizar mutação em membros *PS*, segundo taxa *m*
    - *3.5* *P* ← *PS*
    - 3.6 **Para cada** *i* em *P*, computar *Aptidão(i)*
4. **Retornar** o indivíduo de *P* com maior aptidão

Exemplo

Codificando o problema

Ex.: problema 8 rainhas

- Cada estado deve especificar posição de 8 rainhas, em coluna com 8 quadrados
  - $8 \times \log_2 8 = 24$ bits se codificação binária



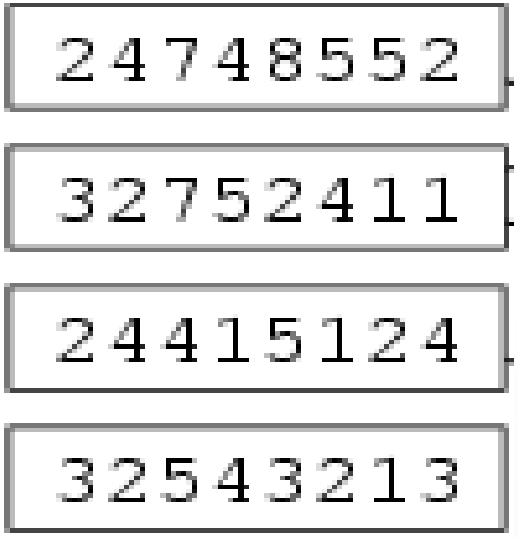
Binário = 111 101 011 001 110 100 010 000
Inteiro = 8 6 4 2 7 5 3 1

Exemplo

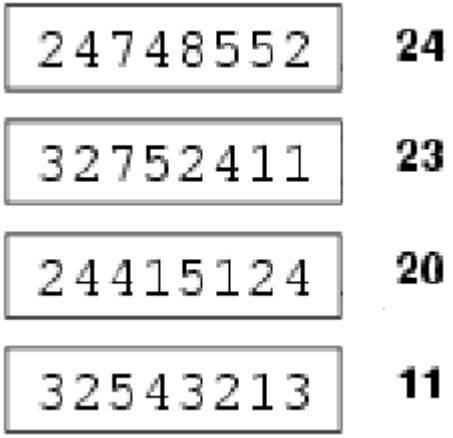
(a) Gerando população inicial

8 dígitos, com valores de 1 a 8

- Exemplo: população com 4 cadeias de 8 dígitos
  - Representam estados de 8 rainhas



Exemplo

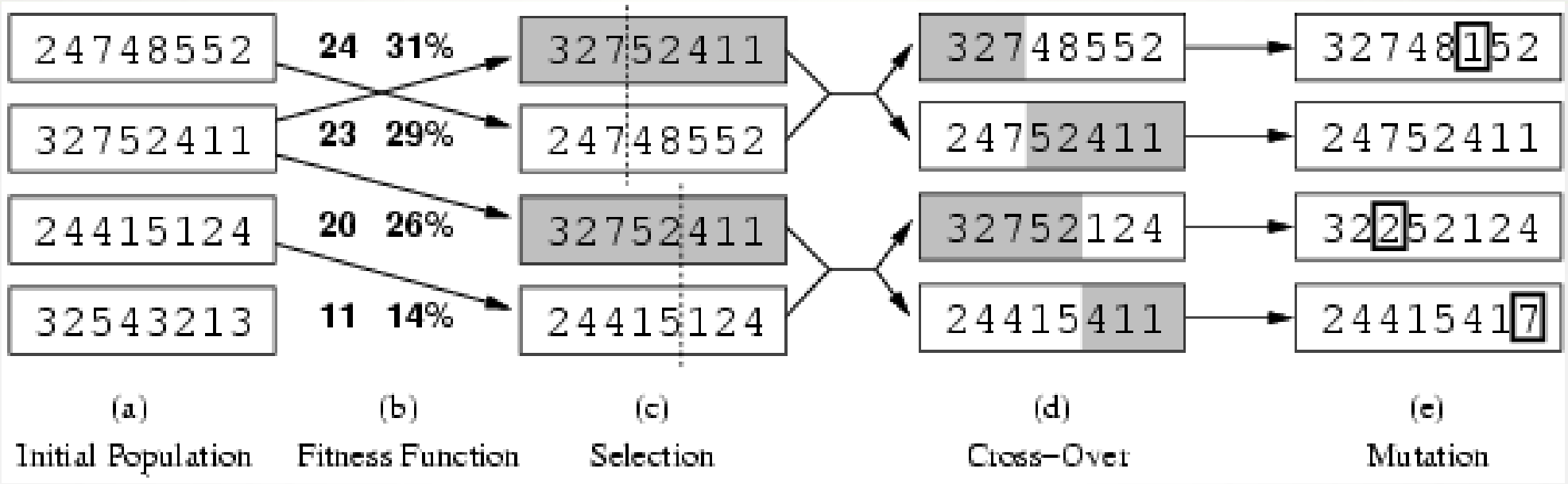


(b) *Avaliação*

Função de avaliação (aptidão)

- Deve retornar valores maiores para estados melhores
- Ex.: 8 rainhas: número de pares de rainhas não-atacantes
  - Valor 28 para uma solução
  - (min = 0, max = 8 × 7/2 = 28)

Exemplo

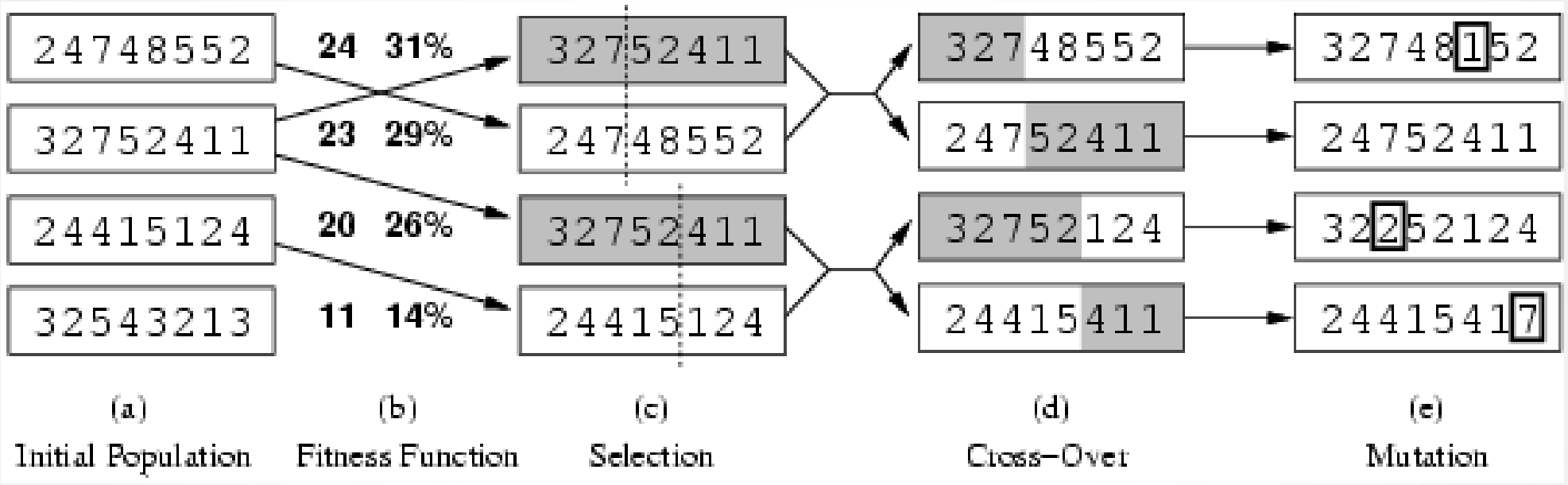


(c) *Seleção*

Proporcional à aptidão do indivíduo

- Vários métodos
- Todos tendem a privilegiar indivíduos mais aptos
  - 24/(24+23+20+11) = 31% no exemplo
  - 23/(24+23+20+11) = 29% etc

Cruzamento



(d) *Cruzamento*

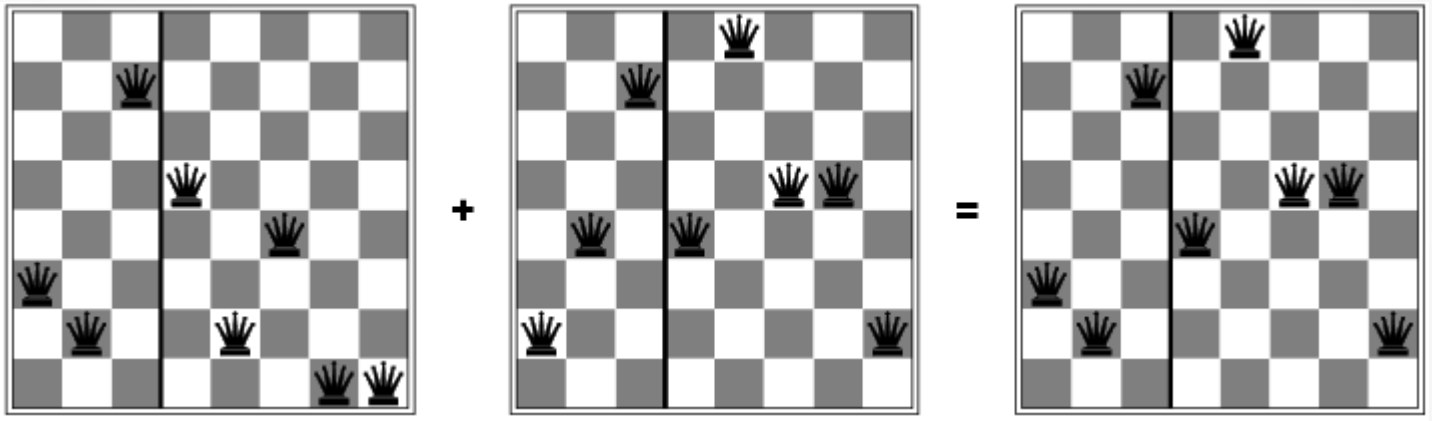
Indivíduos selecionados formam pares

Operador de cruzamento combina pares

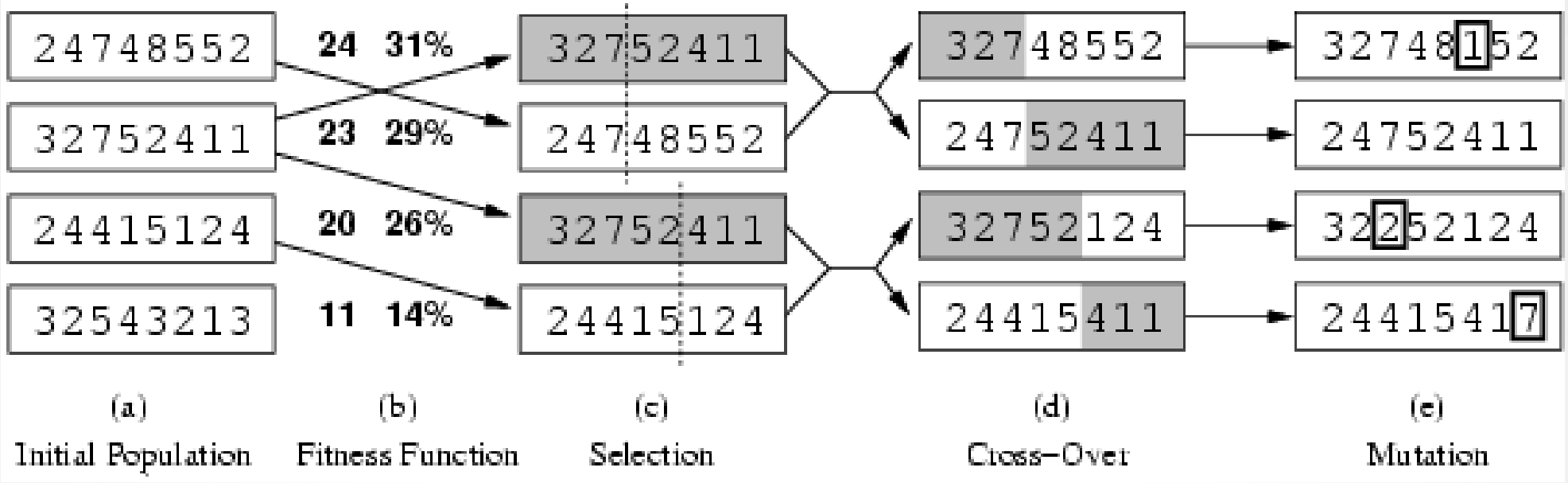
- Ponto de cruzamento gerado ao acaso

Algoritmos Genéticos

Cruzamento



Algoritmos Genéticos



(e) *Mutação*

Mudança aleatória do valor de um gene

- Com pequena probabilidade

Introduz variações aleatórias

- Permitindo soluções pularem para diferentes partes do espaço de busca

Observações

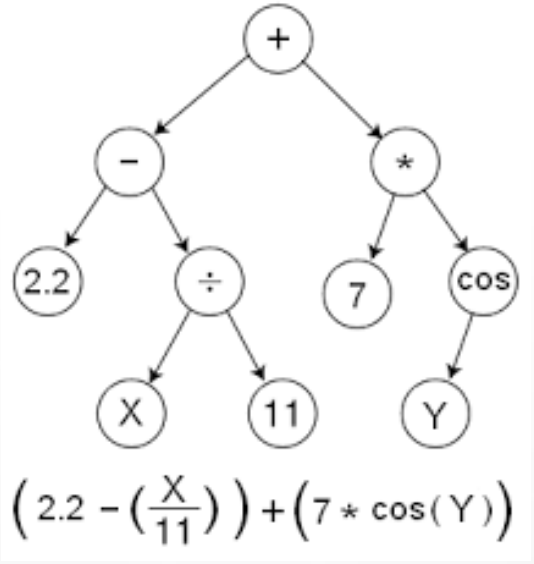
- Se o AG estiver corretamente implementado, a população evolui em gerações sucessivas

- Aptidão do melhor indivíduo e do indivíduo médio aumentam em direção a um ótimo global

Programação genética

•Evolução de programas computacionais usando analogias mecanismos utilizados pela evolução biológica natural (Von Zuben, UNICAMP)

• Representação de indivíduos: árvores

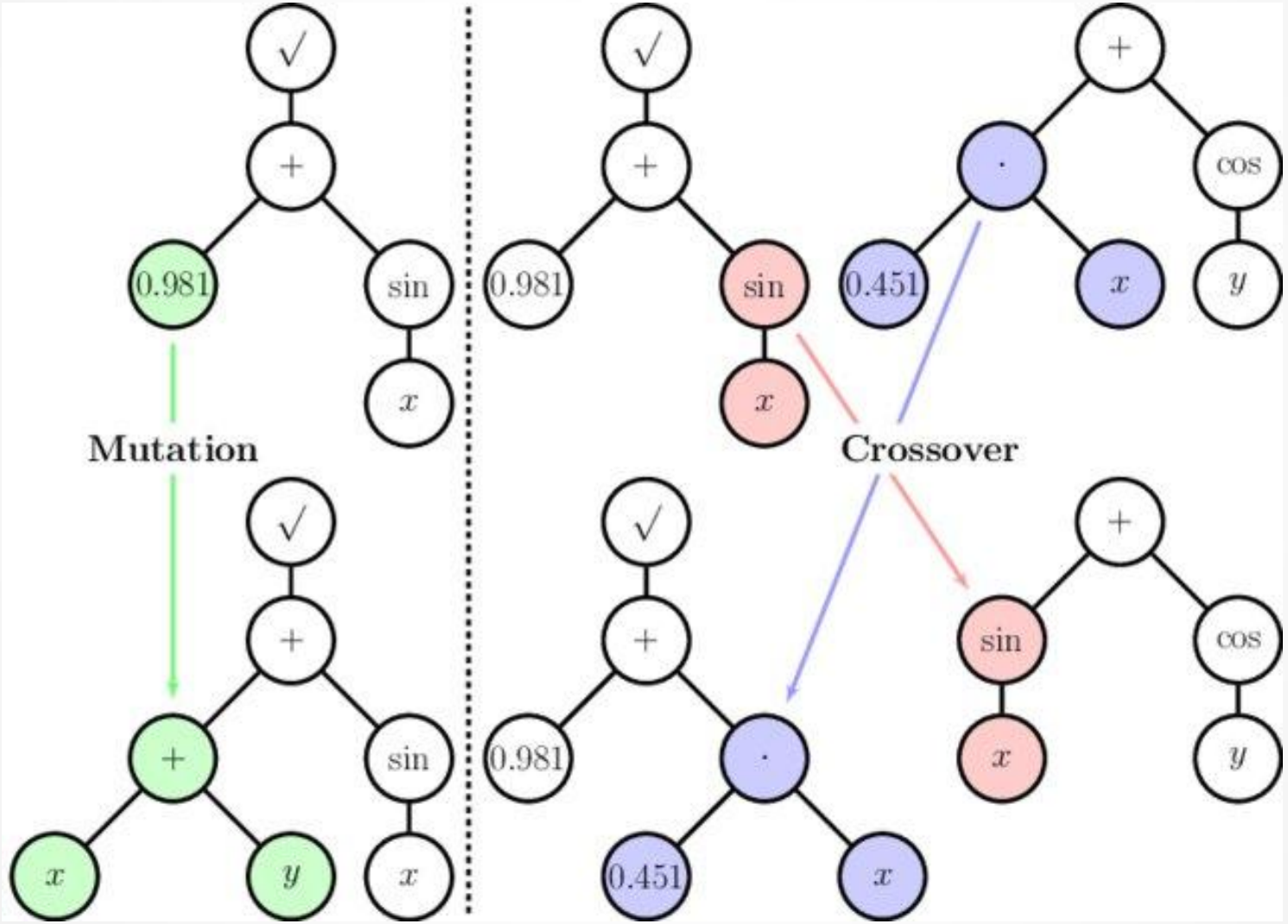


Programação genética

- Passos (Von Zuben, UNICAMP):

1. Gerar uma população inicial de composições aleatórias de funções e terminais (programas computacionais);
2. Executar os seguintes passos até que o critério de parada seja satisfeito:
    1. Execute cada programa da população e atribua um valor de fitness;
    2. Crie uma nova população pela aplicação dos operadores de recombinação, possivelmente mutação (ausente nas versões originais), e seleção junto à população atual, baseado nos valores de fitness de cada indivíduo;
    3. Avalie o resultado obtido nesta nova geração

Programação genética

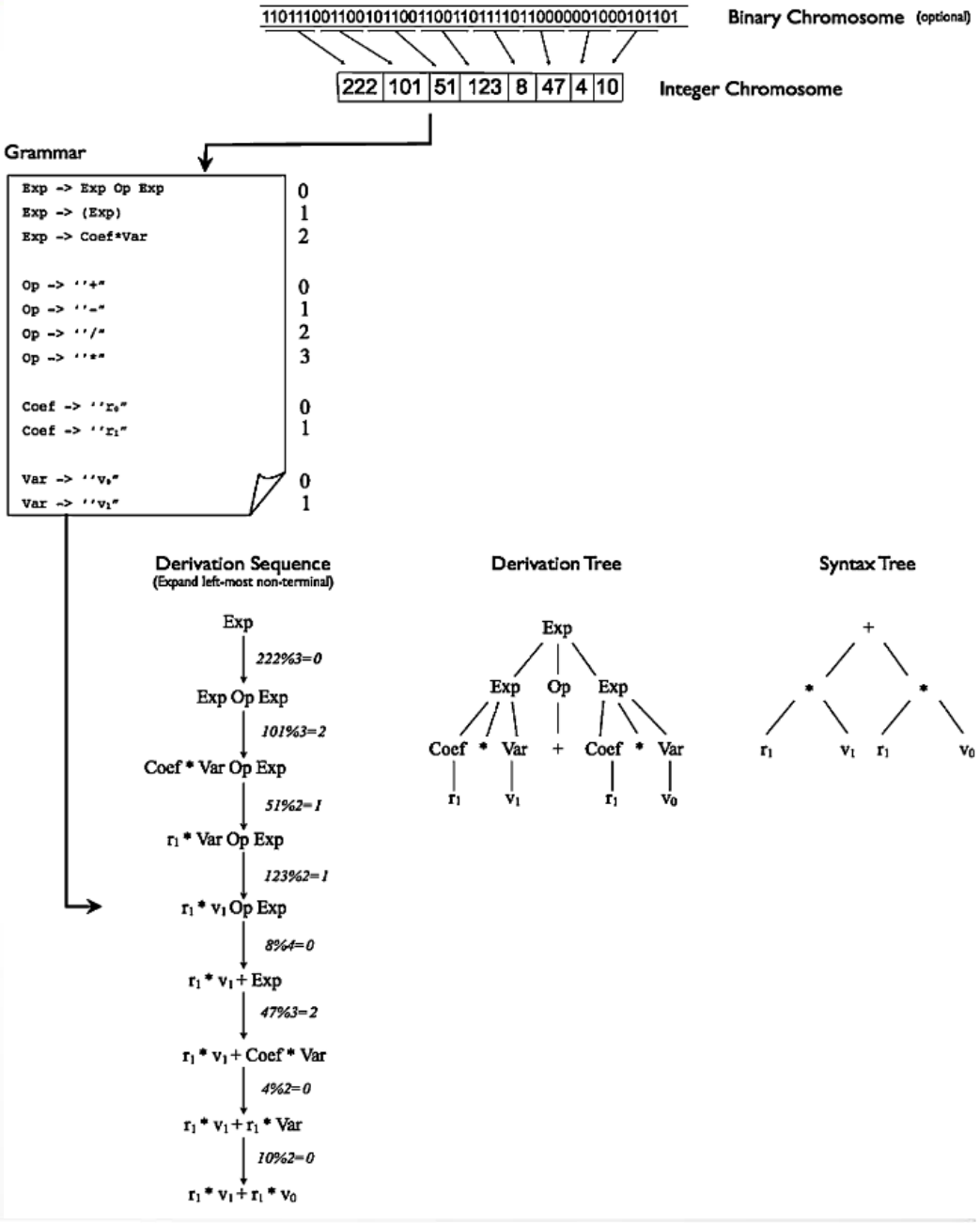


PG baseada em gramática

- Gramática pode ser usada para especificar e gerar:
  - Árvores válidas (indivíduos)
  - Impor e descrever restrições

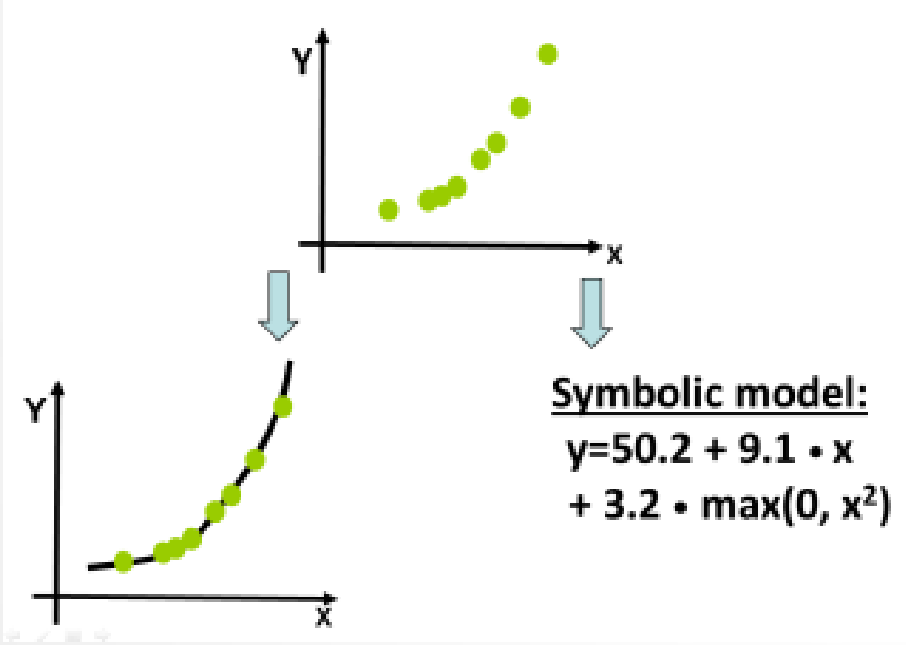
Evolução gramatical

- Usa uma representação linear
- Genes podem ter valor entre 0 e 255 (1 byte)
  - Começa pelo símbolo inicial da gramática
  - Divide valor do gene pelo número de regras envolvendo o não terminal correspondente
  - O resto da divisão fornece a regra a ser aplicada
  - Faz derivações à esquerda até que chegue a uma sentença de terminais
  - Caso não chegue a sentença no final do indivíduo, volta à esquerda no vetor e continua aplicando



Evolução gramatical

- Vamos usar em regressão simbólica:
  - Achar a função de regressão que melhor descreve um conjunto de pontos



- Trabalho em dupla
- Cada dupla vai definir a gramática que julgar mais conveniente para seu algoritmo e para solucionar o problema de regressão proposto
- Problema de regressão será disponibilizado na plataforma Kaggle

Evolução gramatical

- Entregar:
  - Relatório descrevendo o que fez e os resultados alcançados
  - Código fonte implementado

Referências

- Livros:
  - Russel e Norvig: Inteligência Artificial, cap 4
  - R. Linden: Algoritmos Genéticos
  - Z. Michelewicz: Genetic Algorithms + Data Structures = Evolution Programs
  - Koza, J. R. (1994). *Genetic programming II* (Vol. 17). Cambridge: MIT press.
  - O'Neill, M., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, *5*(4), 349-358.
- Slides de:
  - UNICAMP
  - UFPE
  - Cornell University
  - Maria das Graças B. Marietto, UFABC