Universidade Tecnológica Federal do Paraná - Campus Curitiba

INTELIGÊNCIA ARTIFICIAL

Metaheurísticas de Buscas

(Algoritmos Evolutivos – Algoritmos Genéticos)

Contexto de Busca

- Busca Cega:
  - **Largura**: estratégia fixa
  - **Profundidade**: estratégia fixa
  - **Custo Uniforme**: g(n)

- Busca Informada: h(n)
  - **A***: g(n) + h(n)
  - **Algoritmos de Busca Local**: h(n)=fitness
    - **Subida de encosta** (*Hill-climbing*)
    - **Têmpera Simulada** (*Simulated Anealing*)
    - **Feixe Local** (*Local beam search*)
    - Algoritmos **Genéticos**

Contexto de Busca

- Busca Clássica:
  - **Largura**: estratégia fixa
  - **Profundidade**: estratégia fixa
  - **Custo Uniforme**: g(n)
  - **A***: g(n) + h(n)

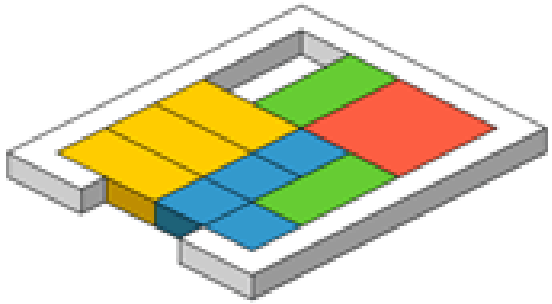
- **Busca Local**: h(n)=fitness
  - **Subida de encosta** (*Hill-climbing*)
  - **Têmpera Simulada** (*Simulated Anealing*)
  - **Feixe Local** (*Local beam search*)
  - Algoritmos **Genéticos**

Busca Clássica: espaço de estados

Ex: rotas, blocos deslizantes,etc...

teste
estado objetivo

- explícito: estar em Bucareste
- implícito: xeque-mate



Em geral a **solução** é uma sequência de ações (**caminho**) que levam do estado inicial ao objetivo; usada em problemas onde a ordem das ações é importante (*permutações nas ações produzem soluções diferentes.*)

AG no contexto de Busca em IA

CARACTERÍSTICAS

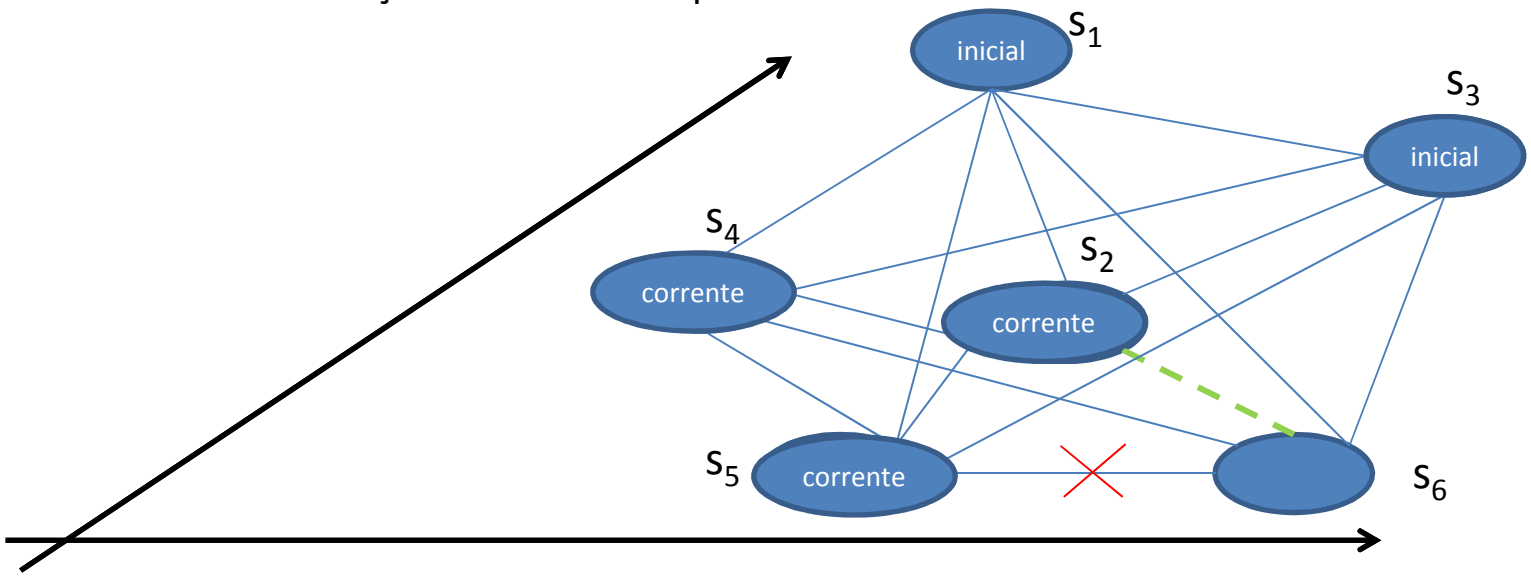
- **Busca informada**: utiliza função de *fitness (avaliação)* que guia o algoritmo
- **Busca local:** não guarda a informação do caminho para chegar à solução
- **Estados sucessores** gerados pela combinação de dois estados pais
- **Função de adequação ou fitness:** f(n) = h(n)
  f(indivíduo) = função de adequação ou fitness (indivíduo)
- **Útil quando** o espaço de estados é muito grande ou muito complexo para tratamento analítico

método de busca probabilístico de busca e otimização

Busca Local e AG: espaço de estados

Normalmente, em busca local, o espaço de estados é um grafo completo ou totalmente conexo (bidirecional).

Logicamente, a função de transição de estados – que retorna os vizinhos ao(s) estado(s) corrente(s) – pode eliminar algumas arestas (por exemplo estados associados a soluções infactíveis) e isto depende de como a tal função foi construída pelo modelador.

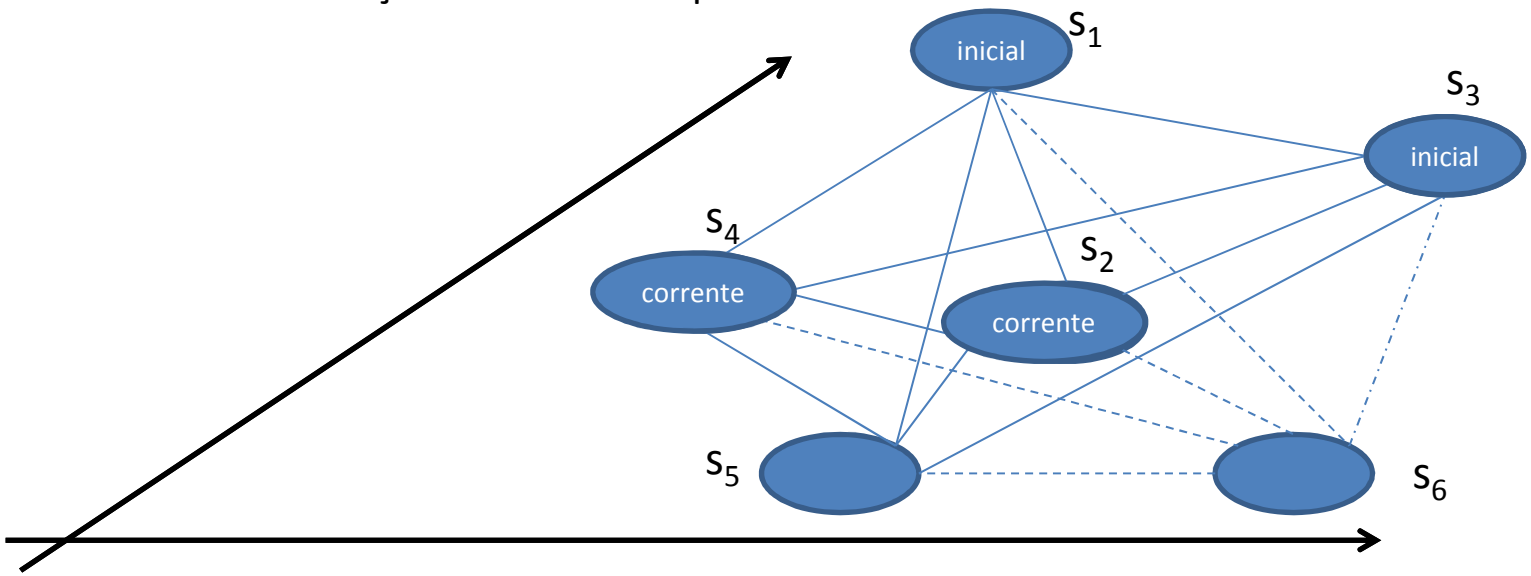


Operadores e configurações específicos do AG podem fazer com que algumas transições entre estados não sejam possíveis (ex. Reparação) ou menos prováveis (ex. Penalização).

Busca Local e AG: espaço de estados

Normalmente, em busca local, o espaço de estados é um grafo completo ou totalmente conexo (bidirecional).

Logicamente, a função de transição de estados – que retorna os vizinhos ao(s) estado(s) corrente(s) – pode eliminar algumas arestas (por exemplo estados associados a soluções infactíveis) e isto depende de como a tal função foi construída pelo modelador.

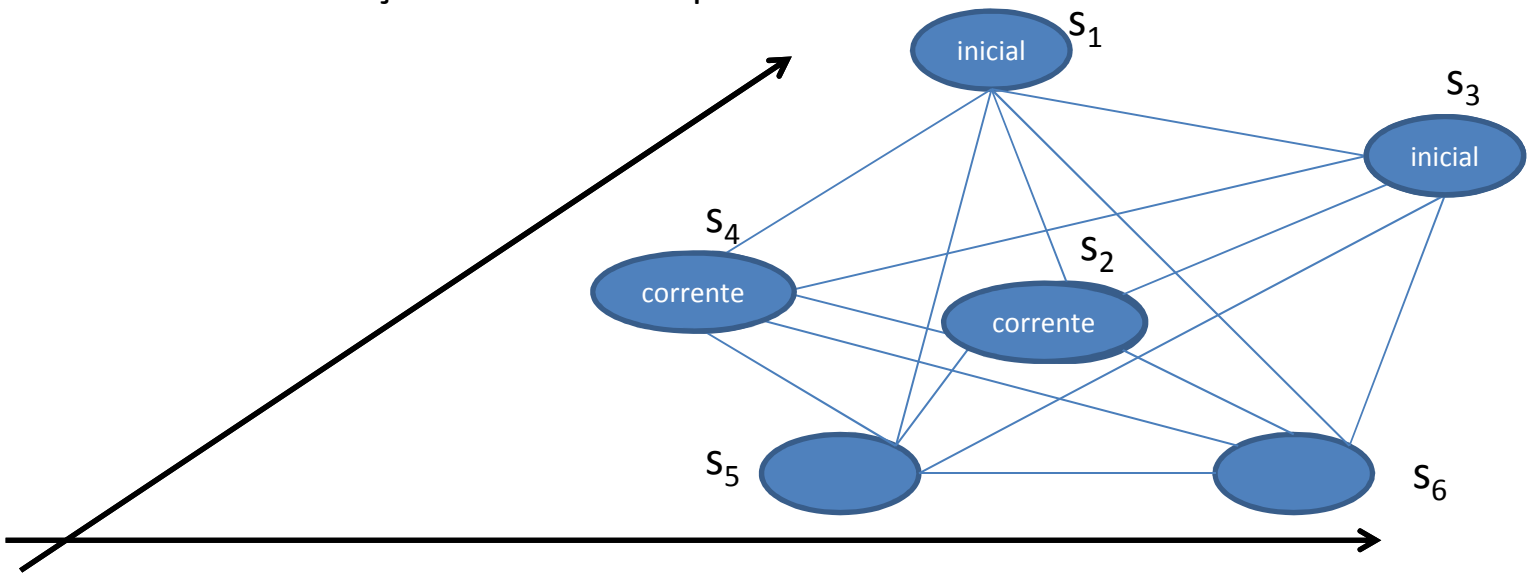


A função de seleção do próximo estado também pode deixar alguns estados de lado (ex. elitista). Eles são avaliados mas não há mudança de estado (baixo fitness – qualidade).

Busca Local e AG: espaço de estados

Normalmente, em busca local, o espaço de estados é um grafo completo ou totalmente conexo (bidirecional).

Logicamente, a função de transição de estados – que retorna os vizinhos ao(s) estado(s) corrente(s) – pode eliminar algumas arestas (por exemplo estados associados a soluções infactíveis) e isto depende de como a tal função foi construída pelo modelador.

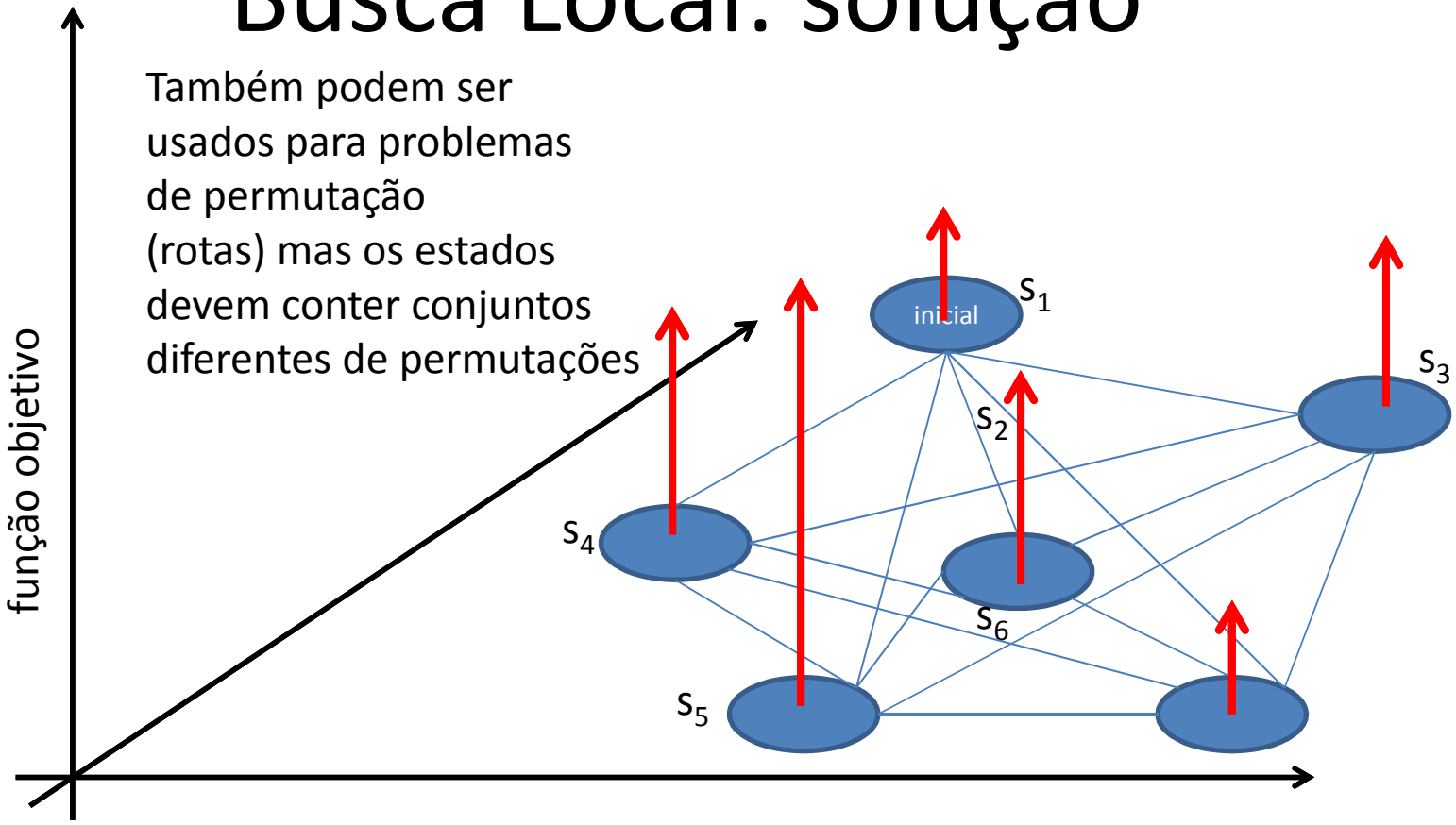


Resumindo, a combinação da função de transição, dos operadores da técnica e da seleção do próximo estado dentre os vizinhos candidatos determina como o espaço de estados (e quanto dele) vai ser explorado.

YYYY1

Busca Local: solução

Também podem ser usados para problemas de permutação (rotas) mas os estados devem conter conjuntos diferentes de permutações



Em geral, a **solução** consiste em encontrar um estado (não um caminho entre estados) cujo valor da função objetivo (seta em vermelho) seja possivelmente ótimo. Há a valoração da função objetivo (sabe-se que um estado é melhor do que outro) e esta valoração é usada para guiar a busca.

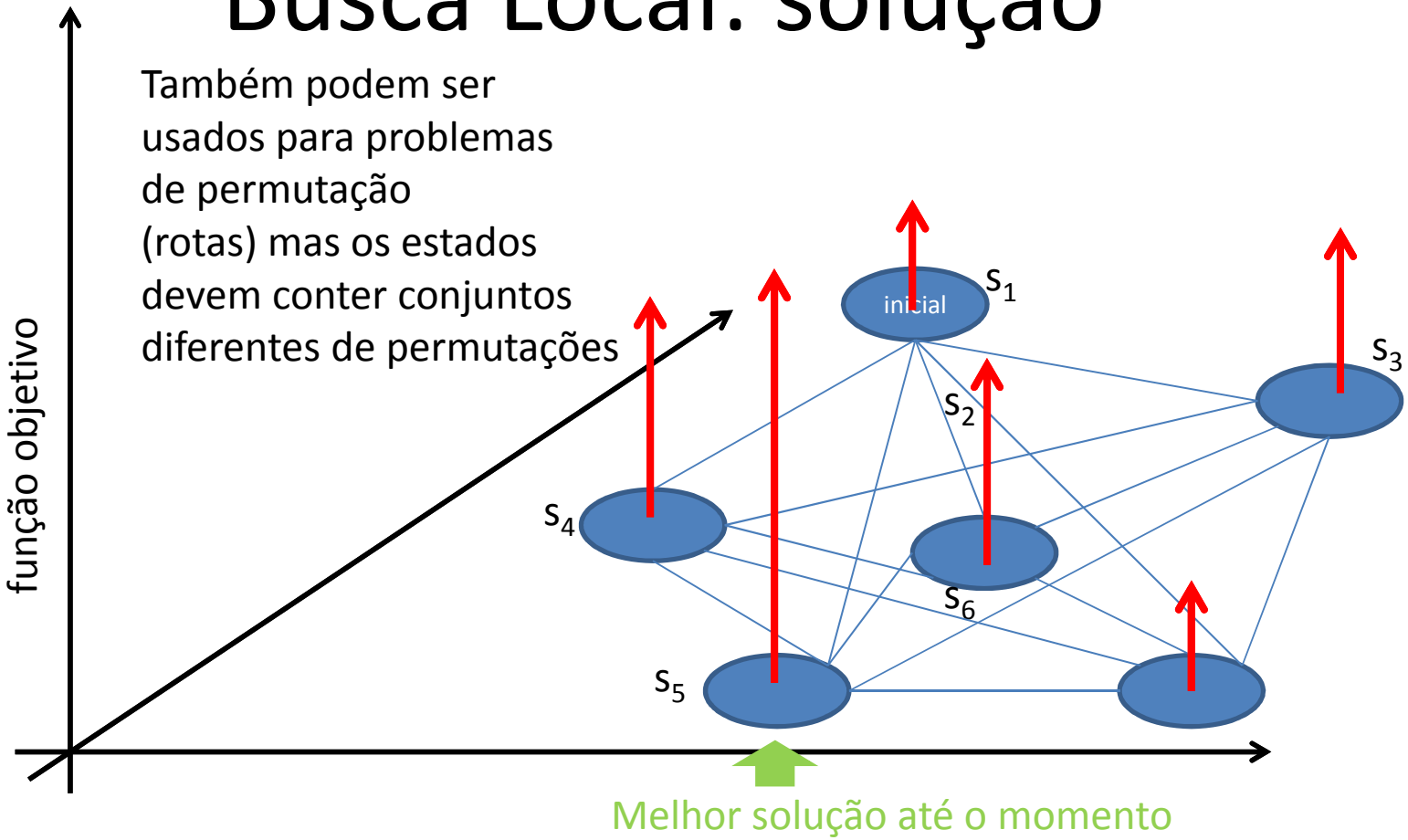
Slide 9

YYYY1 XXXX; 24/03/2013

YYYY1

Busca Local: solução

Também podem ser usados para problemas de permutação (rotas) mas os estados devem conter conjuntos diferentes de permutações



Em muitos casos não há objetivo explícito/implícito (ex. Problema da mochila) que permita usar um procedimento de decisão para saber se um estado objetivo foi atingido, então, outros critérios devem ser usados para interromper a busca (máximo de gerações ou máximo de avaliações ou estagnação no fitness).

Slide 10

YYYY1 XXXX; 24/03/2013

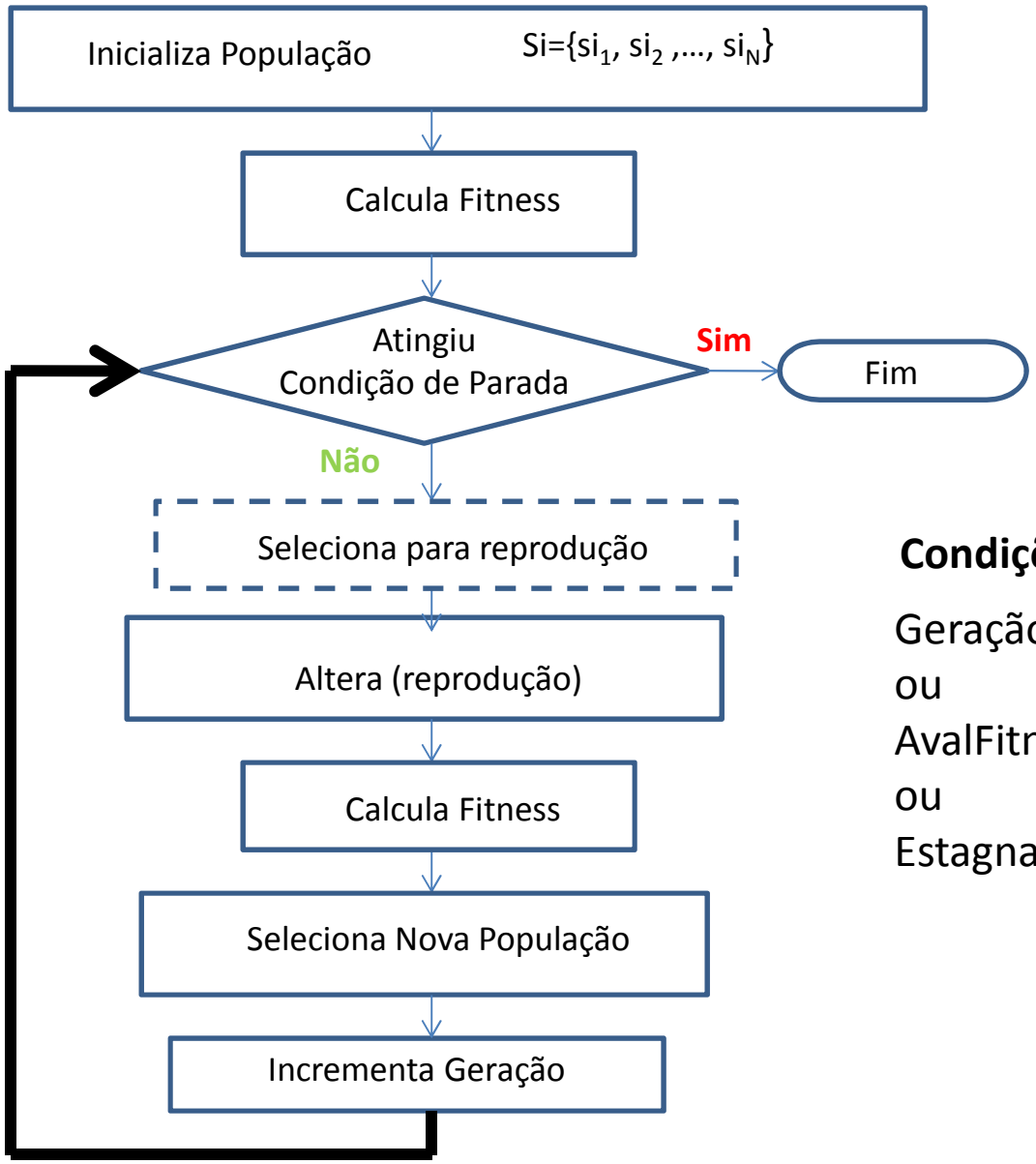
AG: Computação **Evolutiva** (CE)

- Algoritmos Genéticos (AGs)
  - Programação Genética (PG)
- Programação Evolutiva
- Estratégias Evolutivas

Algoritmos de busca global: busca ampla no espaço de busca (ao invés de restrita à vizinhança de alguma solução)

f(solução) = fitness.

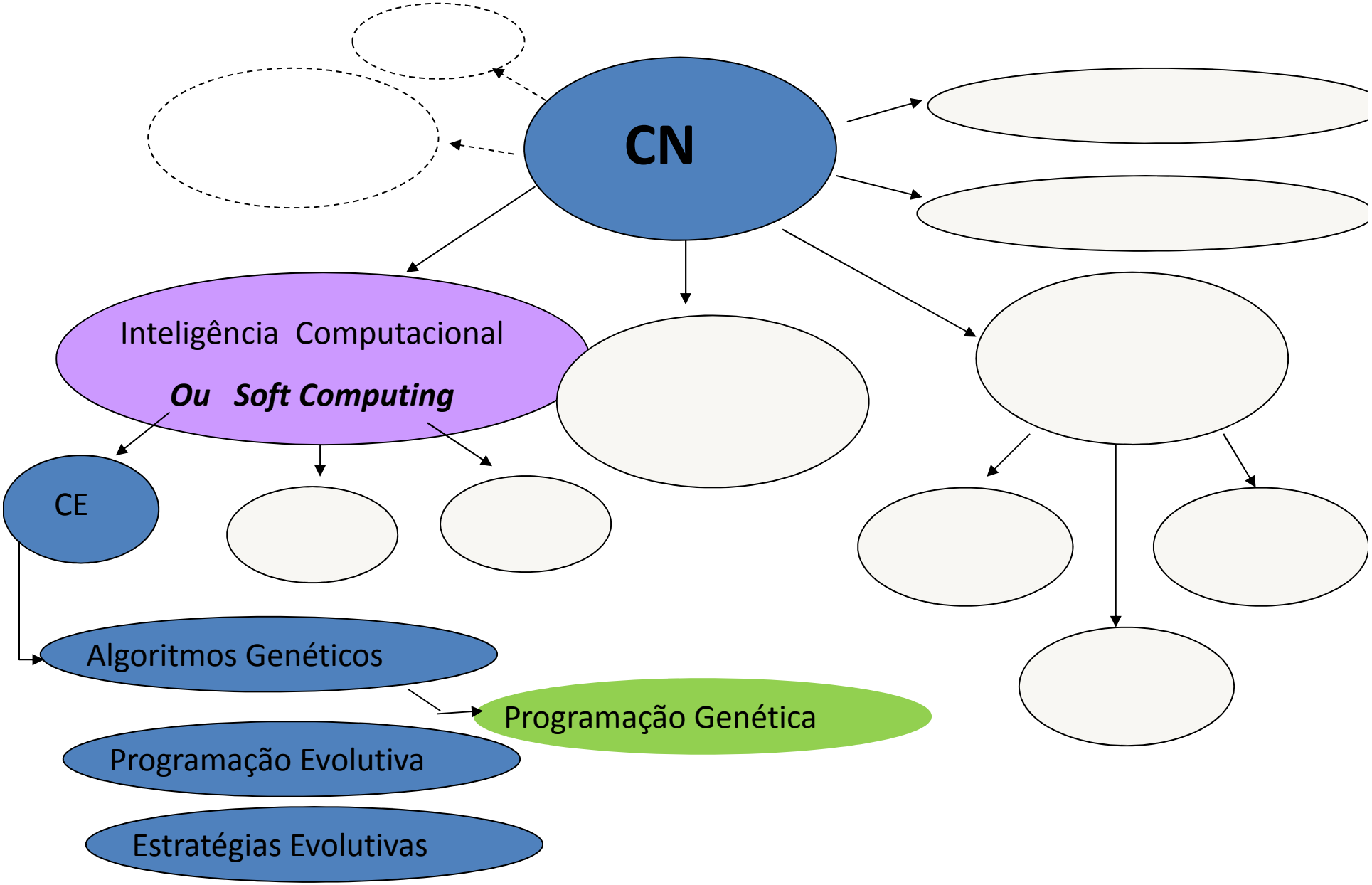
Esquema Geral de um Algoritmo Evolutivo



Condições de Parada

Geração = MaxGer
ou
AvalFitness= MaxAvalFit
ou
Estagnação do Fitness melhor

AG e Computação Natural



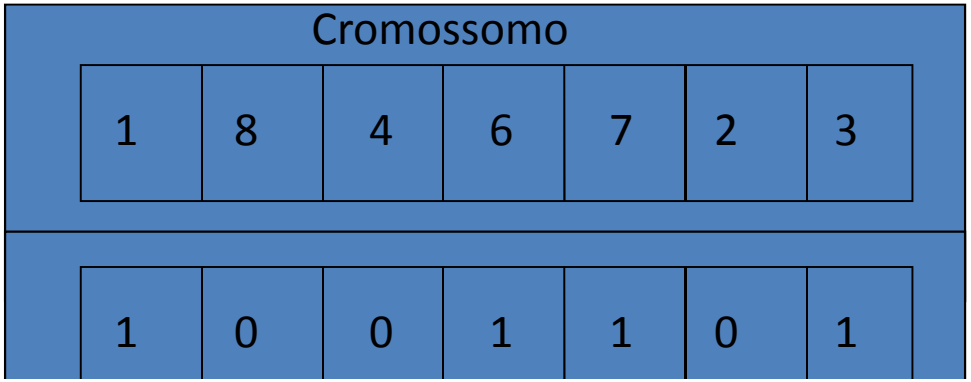
AG: Algoritmos Genéticos

- Popularizados por John Holland podem ser considerados os primeiros modelos algorítmicos desenvolvidos para simular os sistemas genéticos.

- Baseados na evolução da população e utilização de operadores de seleção e reprodução

AG: Algoritmos Genéticos

- Evolução da População: Modificação no indivíduo (cromossomo)
- Sobrevivência do melhor: manter a solução (indivíduo) de maior adequabilidade (fitness) no conjunto de soluções
- Reprodução:
  Crossover
  mutação



Exemplo: PROBLEMA DAS 8 RAINHAS

Indivíduo é representado por um cromossomo -> vetor de bits

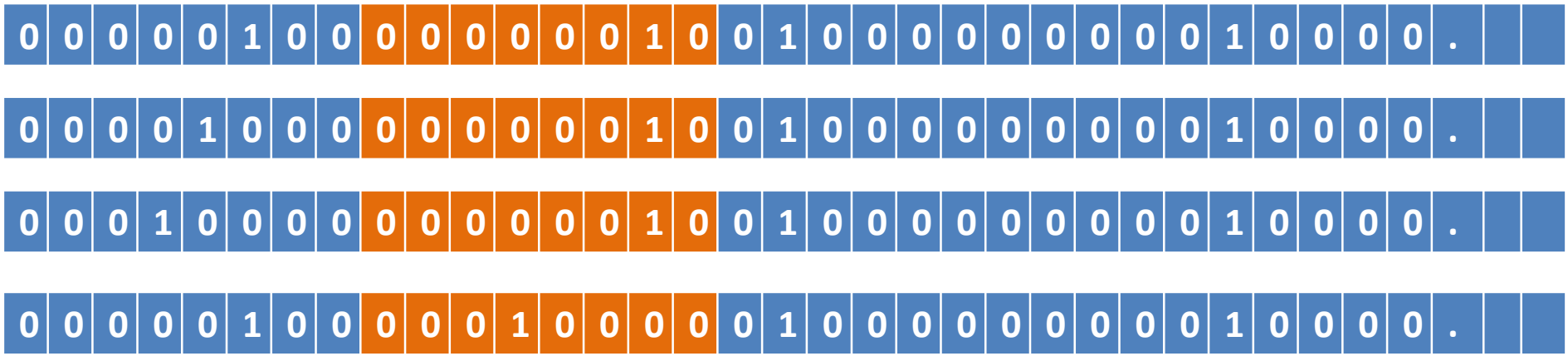
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0
0	0	0	0	0	0	1	1

1ª coluna

Exemplo: PROBLEMA DAS 8 RAINHAS

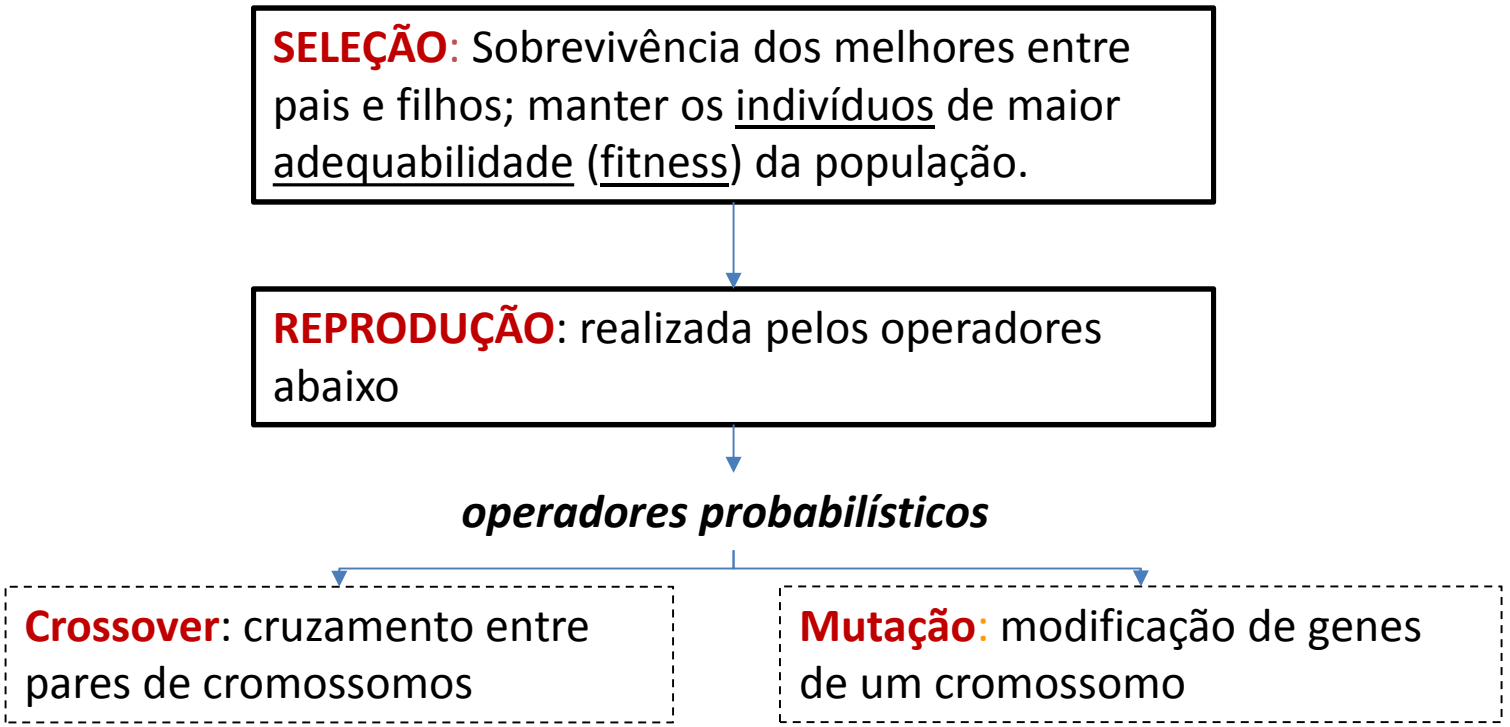
POPULAÇÃO: conjunto de cromossomos ou de indivíduos.

População de 4 indivíduos ou cromossomos para 8-Rainhas



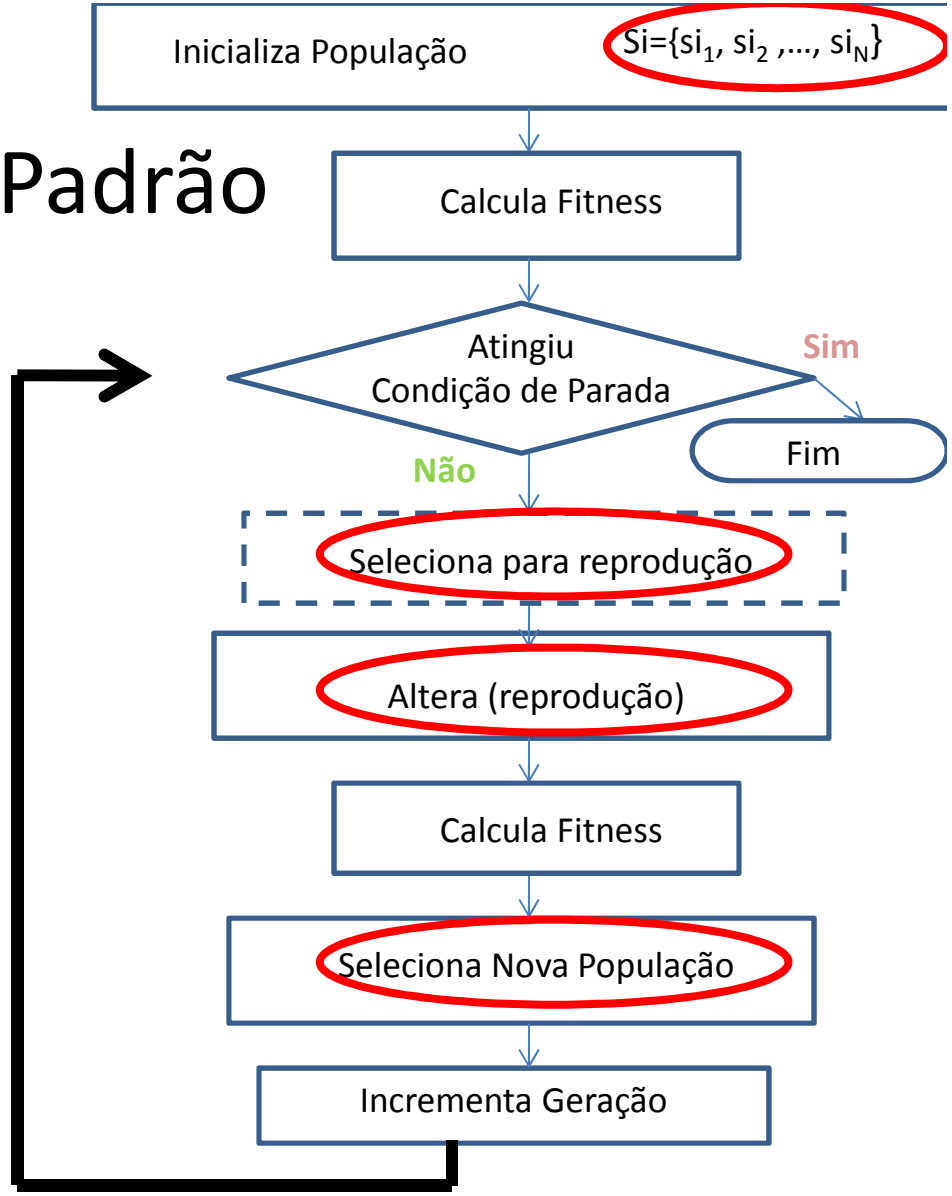
Cada indivíduo representa uma disposição diferente no tabuleiro

AG: etapas e operadores



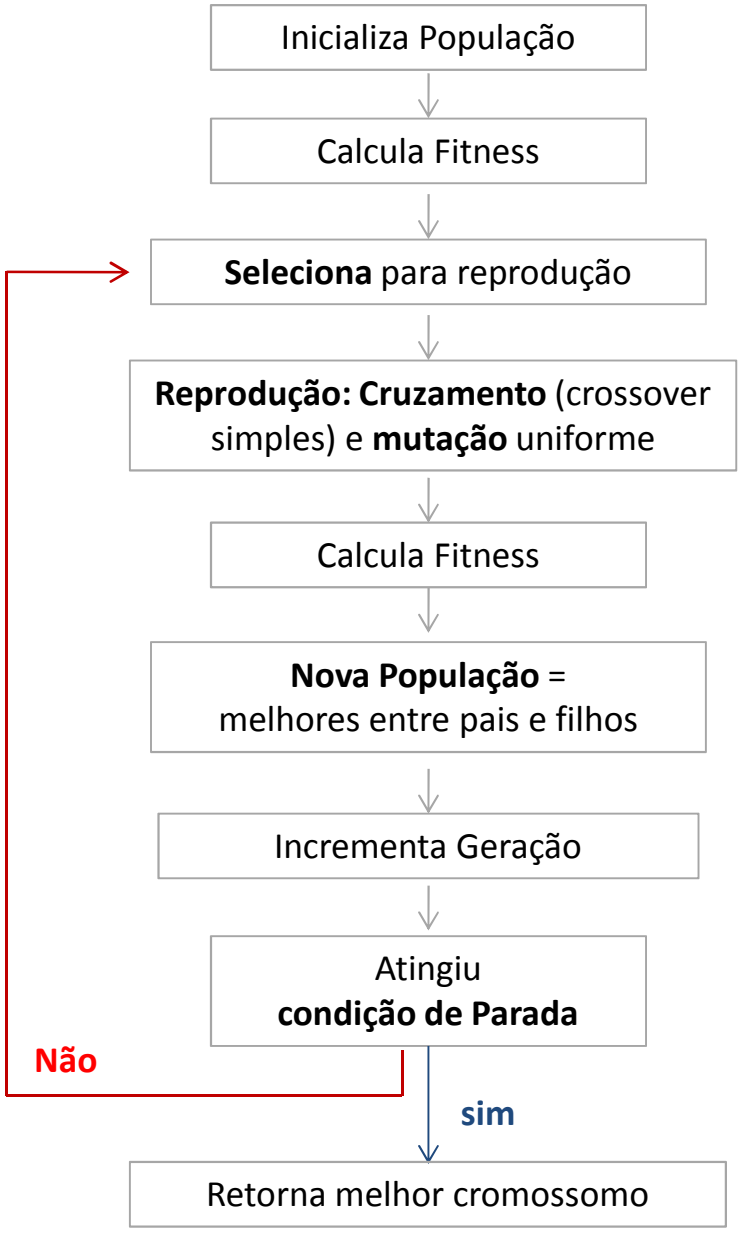
AG Canônico ou AG Padrão

- Codificação Binária
- Seleção
  - Reprodução (roleta)
- Reprodução
  - Crossover simples
  - Mutação Uniforme
- Sobrevivência
  - Melhores (entre pais e filhos) irão compor a nova população



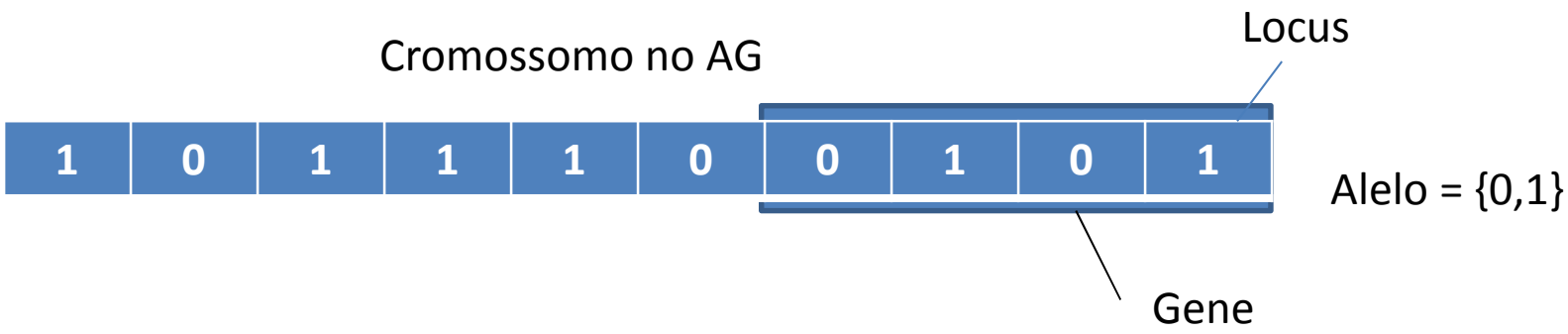
AG Canônico

- Codificação Binária
- **Seleção**
  - Reprodução (método da roleta)
- **Reprodução**
  - Crossover simples (Pc)
  - Mutação Uniforme (Pm)
- **Sobrevivência**
  - Melhores (entre pais e filhos) irão compor a nova população
- **Condição de parada**
  - Geração = MaxGer ou
  - Fitness = máximo atingido ou
  - Estagnação do melhor fitness



AG Canônico

- Codificação Binária



Cromossomo: cadeia de bits de tamanho L

Cada posição (locus) no cromossomo assume um dos dois possíveis alelos, 0 ou 1.

Exemplo: PROBLEMA DAS 8 RAINHAS

Solução por Codificação binária (matriz binária) -> vetor de bits

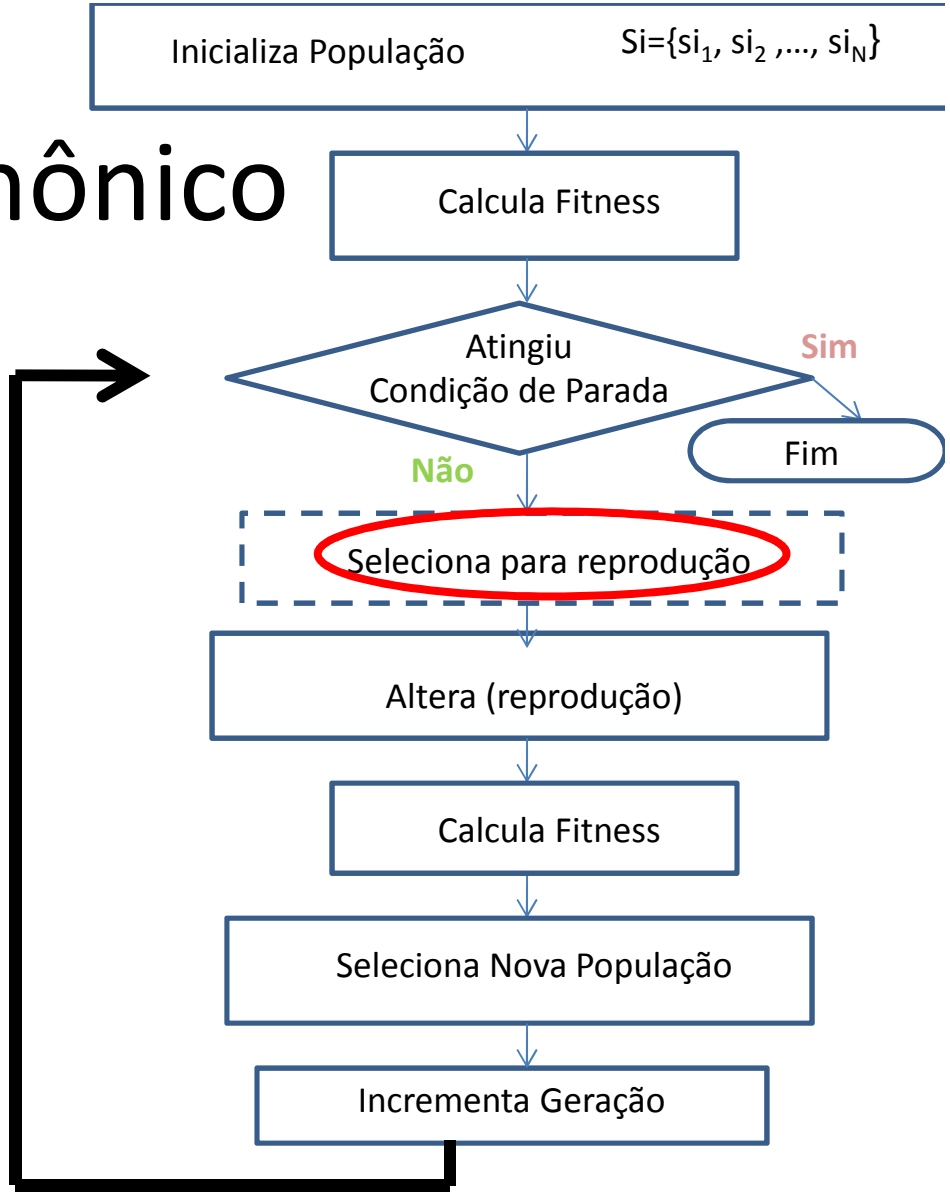
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0
0	0	0	0	0	0	1	1

1ª coluna

Gene representa uma coluna
Cada **locus** representa uma posição da coluna *(locus gênico)*
Alelo: quando vale 1 indica que a posição está ocupada

AG Canônico

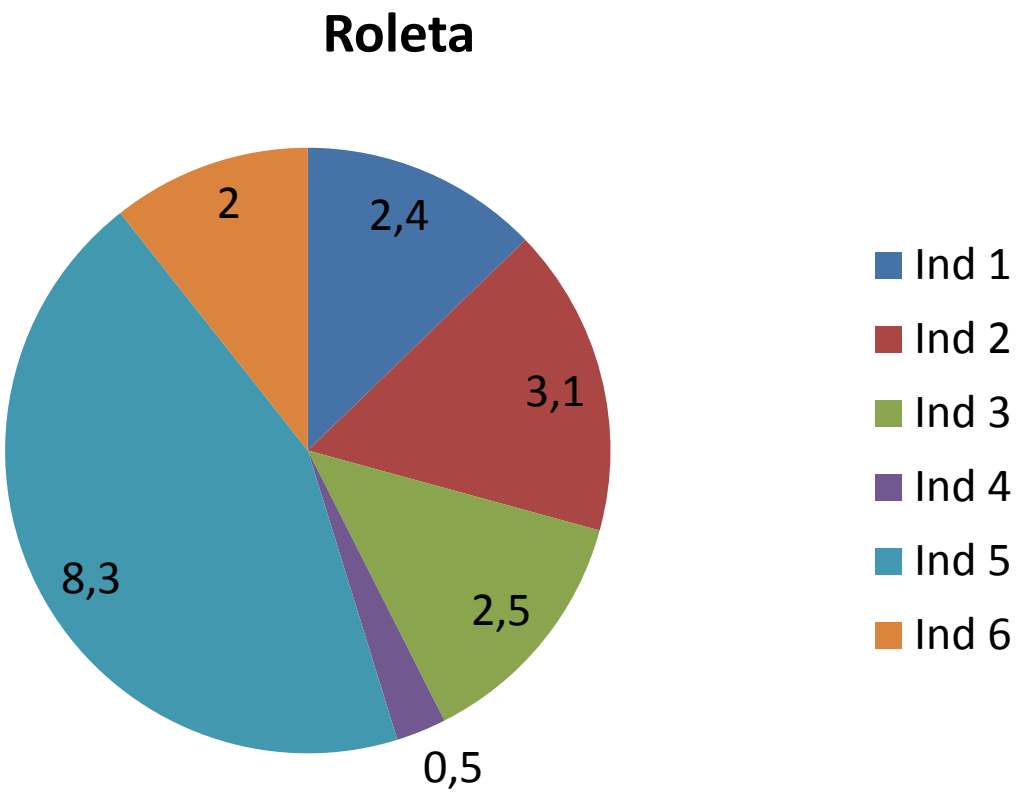
- Codificação Binária
- Seleção
  - Reprodução (roleta)
- Reprodução
  - Crossover simples
  - Mutação Uniforme
- Sobrevivência
  - Melhores (entre pais e filhos) irão compor a nova população



AG Canônico

- Seleção dos pares por roleta

Indivíduo	fitness
Ind 1	2,4
Ind 2	3,1
Ind 3	2,5
Ind 4	0,5
Ind 5	8,3
Ind 6	2,0



AG Canônico: **Seleção por roleta**

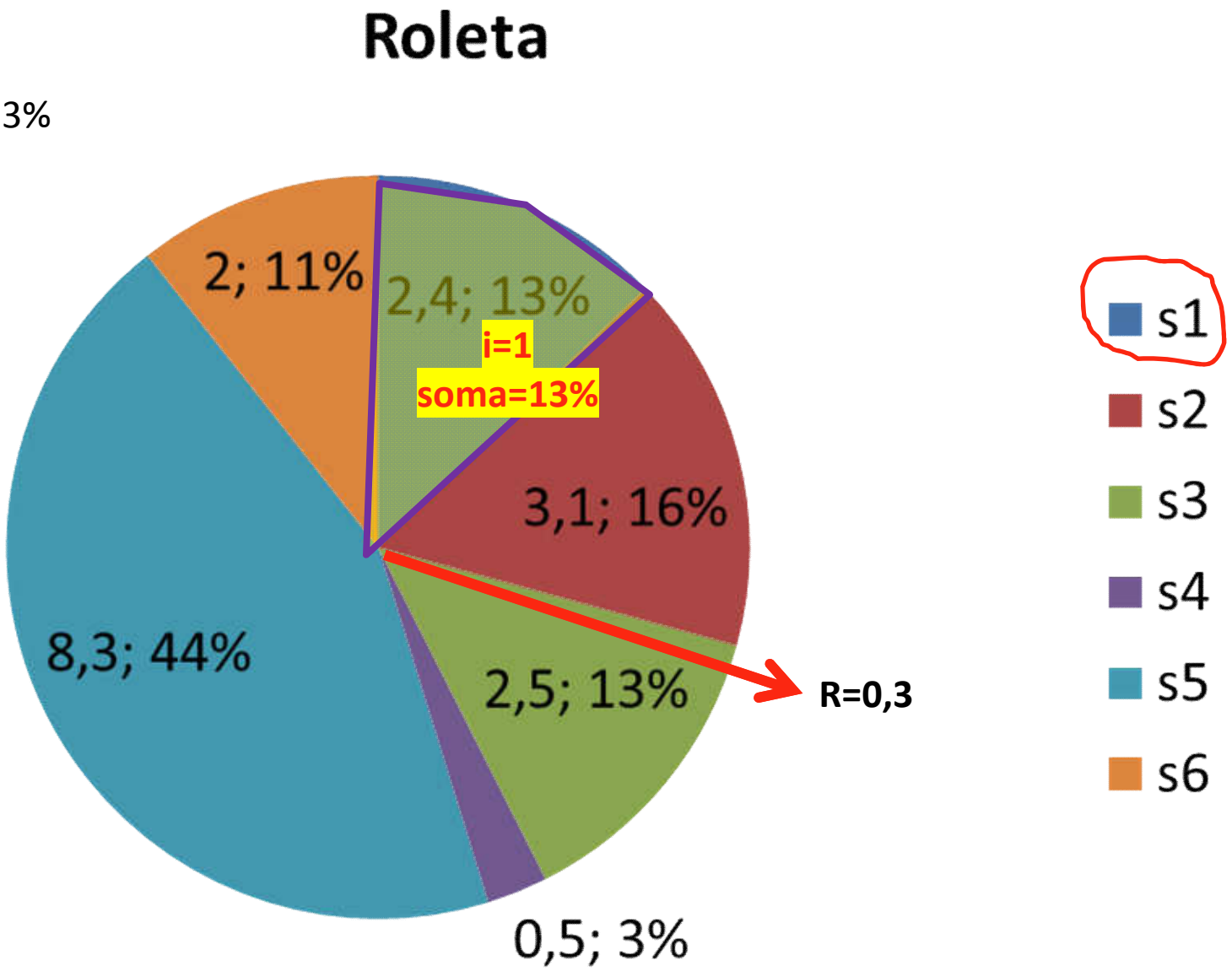
Calcula $p(s_i)$ para i=1,…,N: $$p(s_i) = \frac{fitness(s_i)}{\sum_{k=1}^{N} fitness(s_k)}$$

Algoritmo de seleção por roleta: sorteia um indivíduo por chamada

```
i=1;
soma = p(s_i)
Sorteia r ∈ [0, 1]
enquanto soma < r
    i=(i+1)
    soma = soma + p(s_i)
fim enquanto
Retorna s_i
```

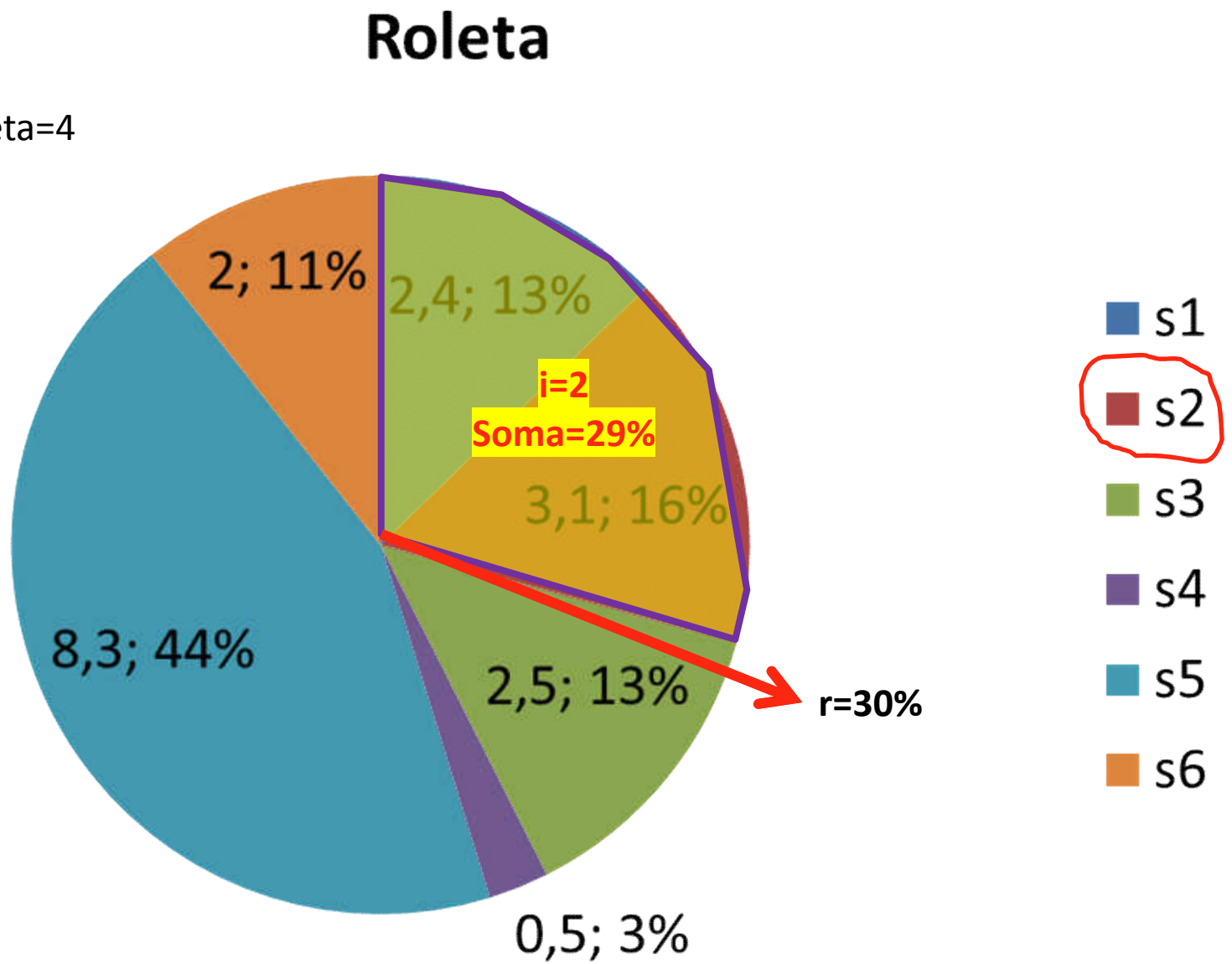
INICIALIZAÇÃO

soma := p(s1) = 13%
r=30%
13% < 30%



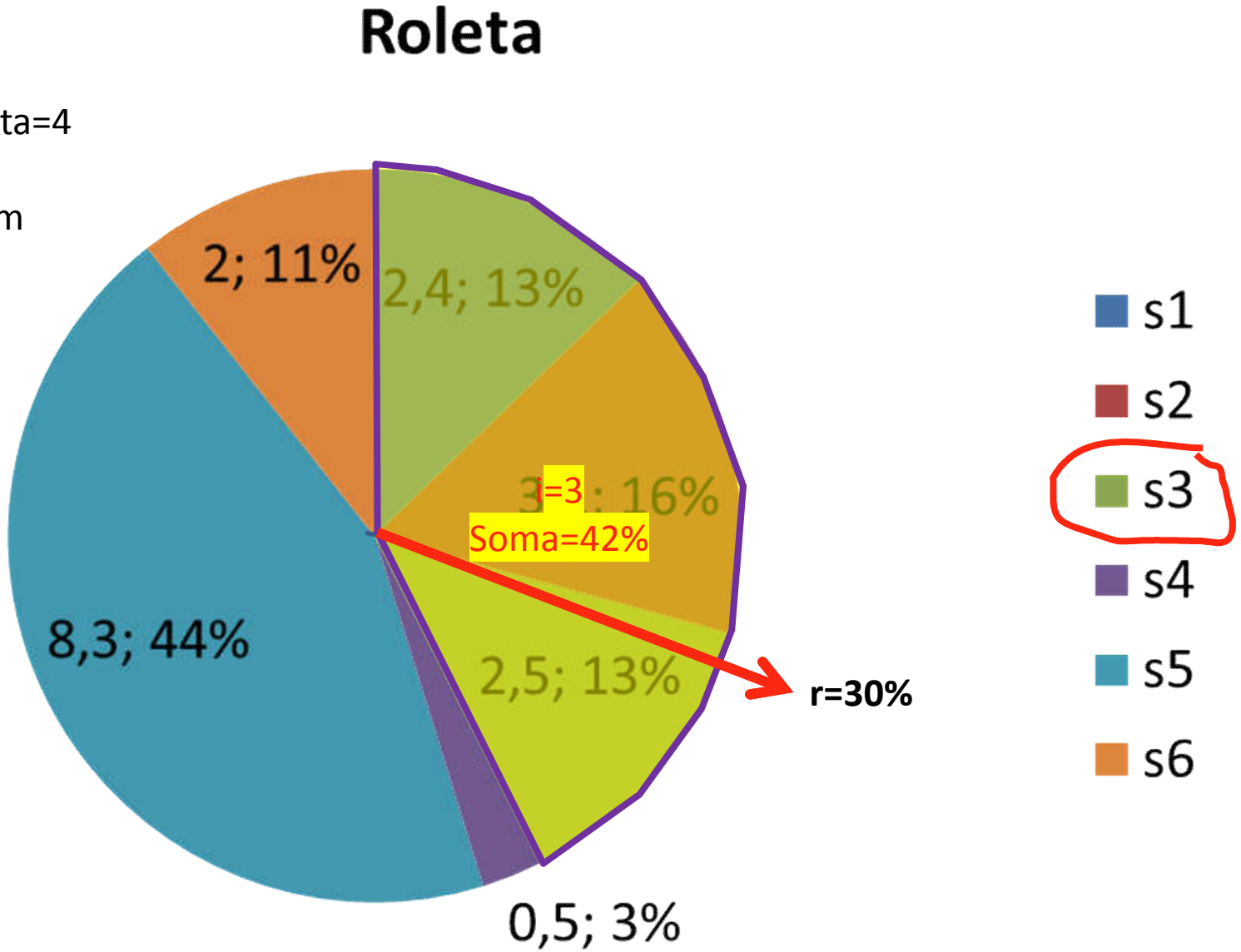
SORTEIO

MaxSorteiosRoleta=4
Soma=29%
29% < 30%



SORTEIO

MaxSorteiosRoleta=4
Soma=42%
42% > 30% => fim
Sorteado=[s3]

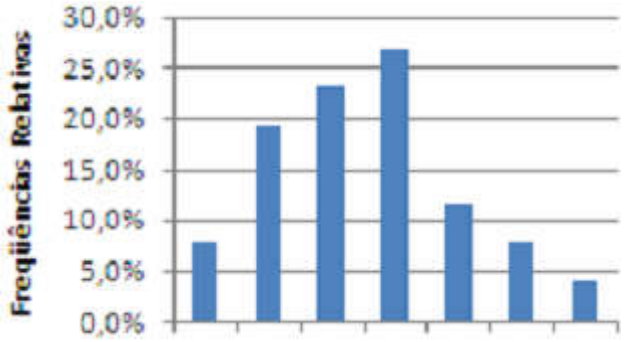


AG Canônico: Roleta

- Algoritmo

Calcula p(si) para i=1,…,N

$$p(s_i) = \frac{fitness(s_i)}{\sum_{k=1}^{N} fitness(s_k)}$$



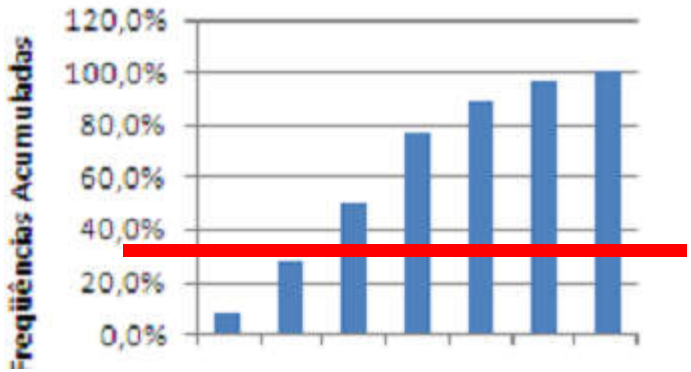
Para sorteio =1 …. MaxSorteirosDaRoleta

- i=1; soma = p(si)
- Sorteia r ~ U(0,1);
- **enquanto** soma < r

  i= i+1;

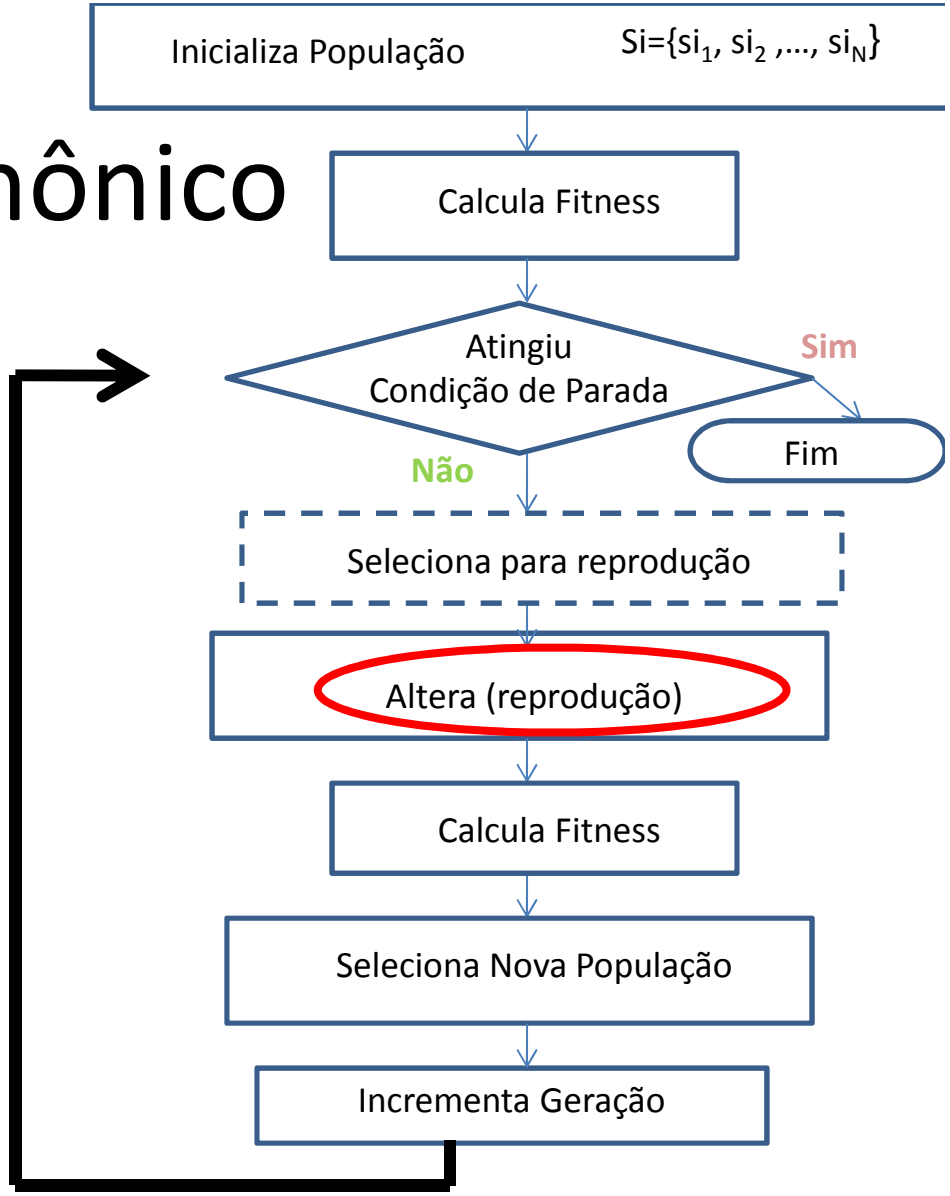
  soma = soma + p(si);
- **fim enquanto**
- Retorna(si);

Fim para



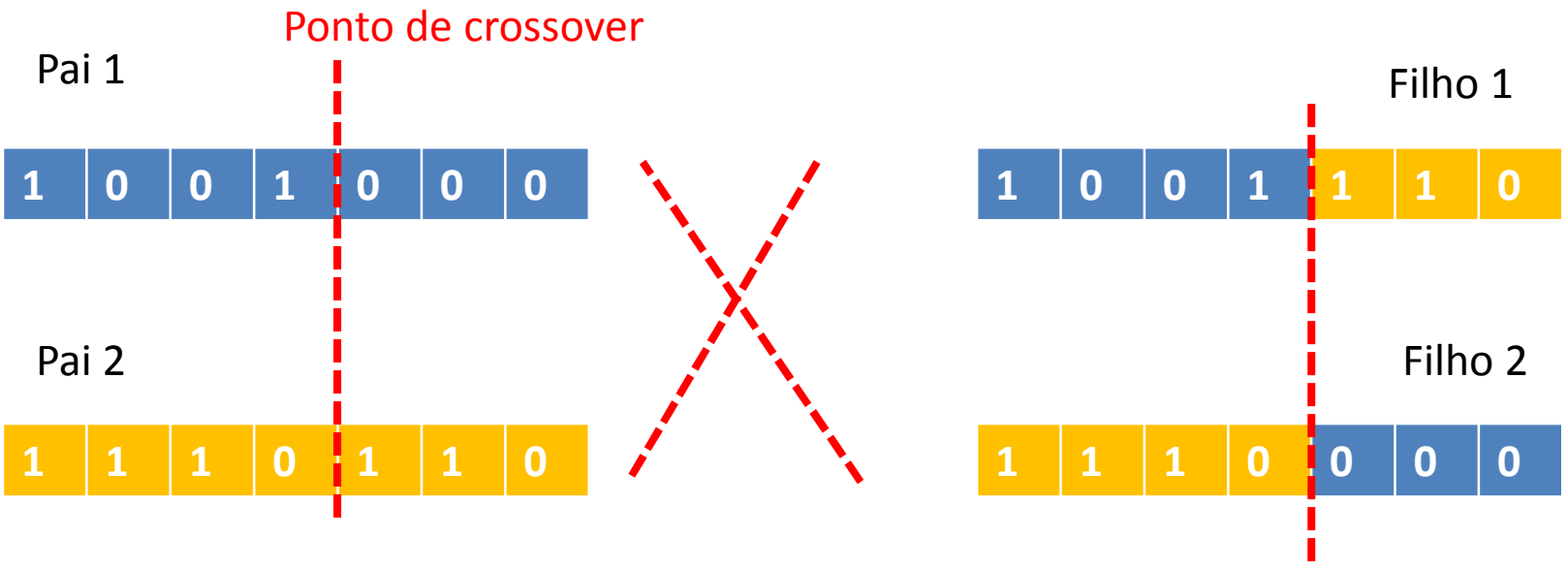
AG Canônico

- Codificação Binária
- Seleção
  - Reprodução (roleta)
- Reprodução
  - Crossover simples
  - Mutação Uniforme
- Sobrevivência
  - Melhores (entre pais e filhos) irão compor a nova população



AG Canônico: Crossover

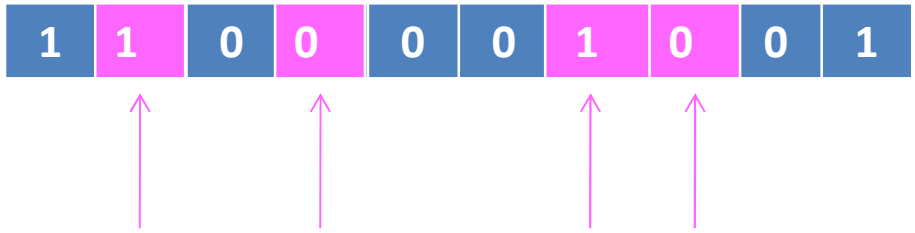
- Crossover Simples (1 ponto)



Crossover de 1 ponto
(posição do cruzamento escolhida aleatoriamente)

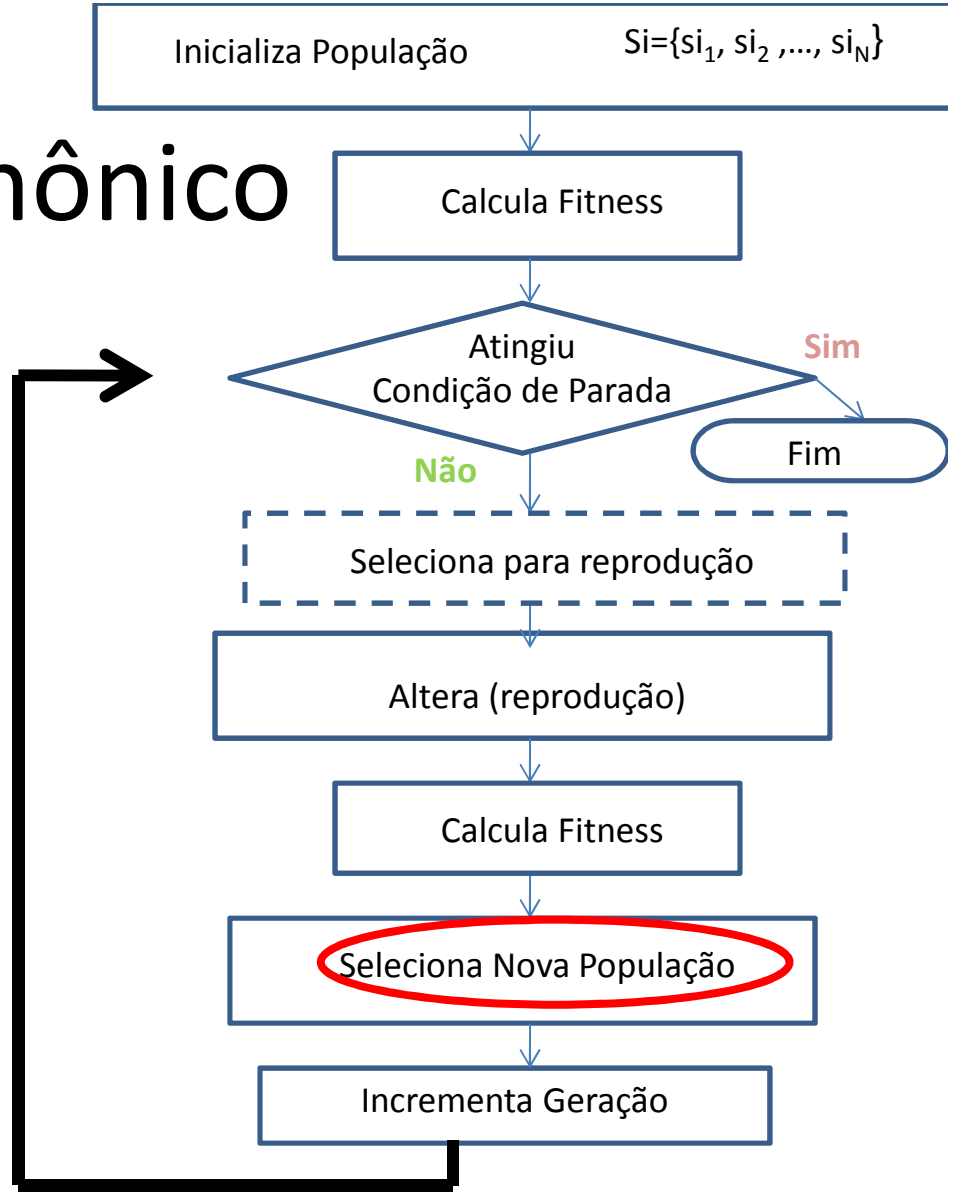
AG Canônico: Mutação

- Mutação Uniforme



AG Canônico

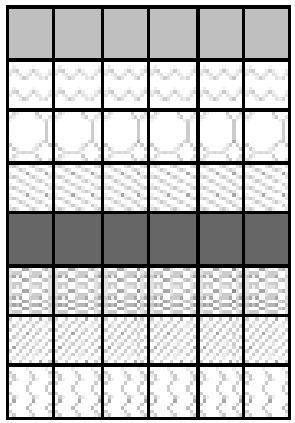
- Codificação Binária
- Seleção
  - Reprodução (roleta)
- Reprodução
  - Crossover simples
  - Mutação Uniforme
- Sobrevivência
  - Melhores irão compor a nova população



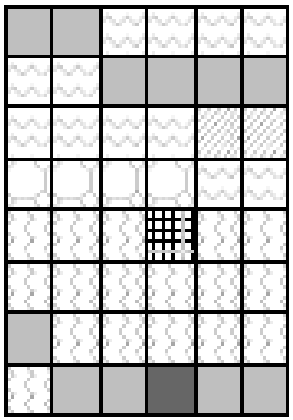
AG Canônico

- Seleção
  - Sobrevivência dos Melhores (entre originais e filhos) para compor a nova população

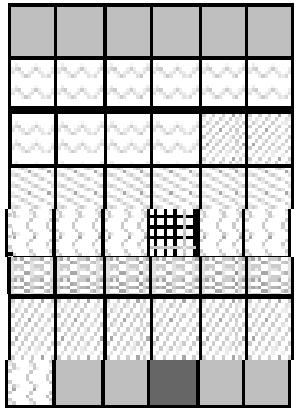
P(t)



Filhos

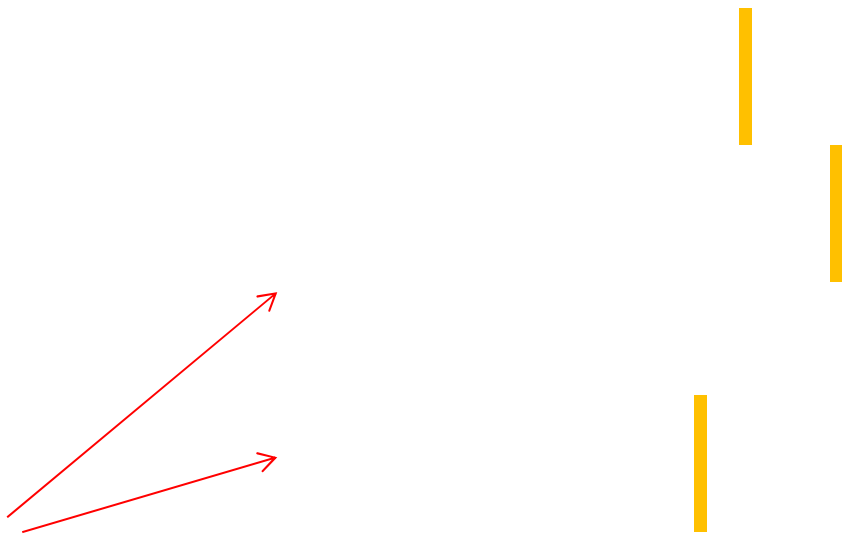


Nova População (P(t+1))

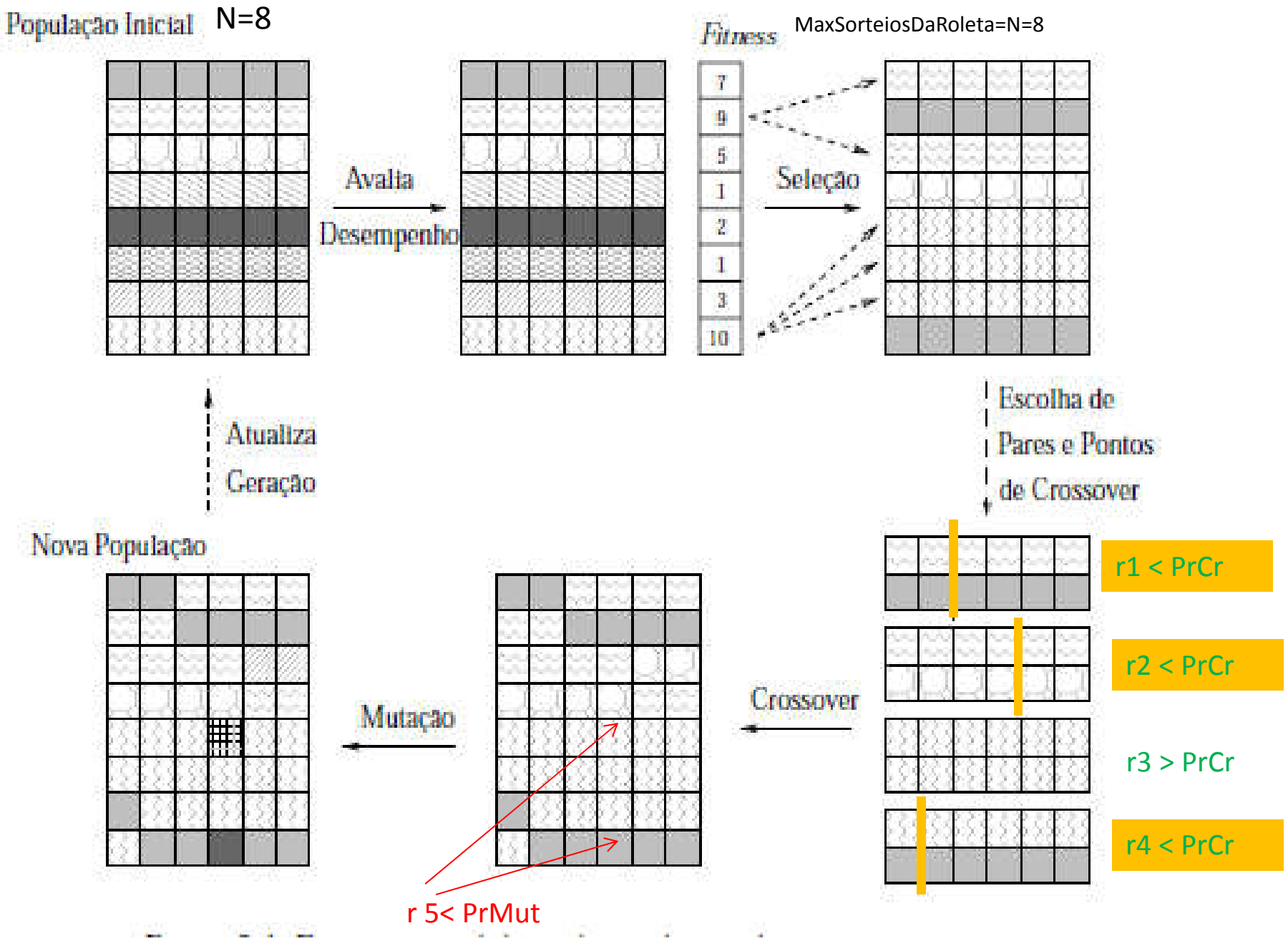


P(t+1)= Seleciona_melhores(P(t)+Filhos)

Esquema Geral da evolução de um Agoritmo Genético



Esquema Geral de um Agoritmo Genético



AG Canônico: **pseudo-código**

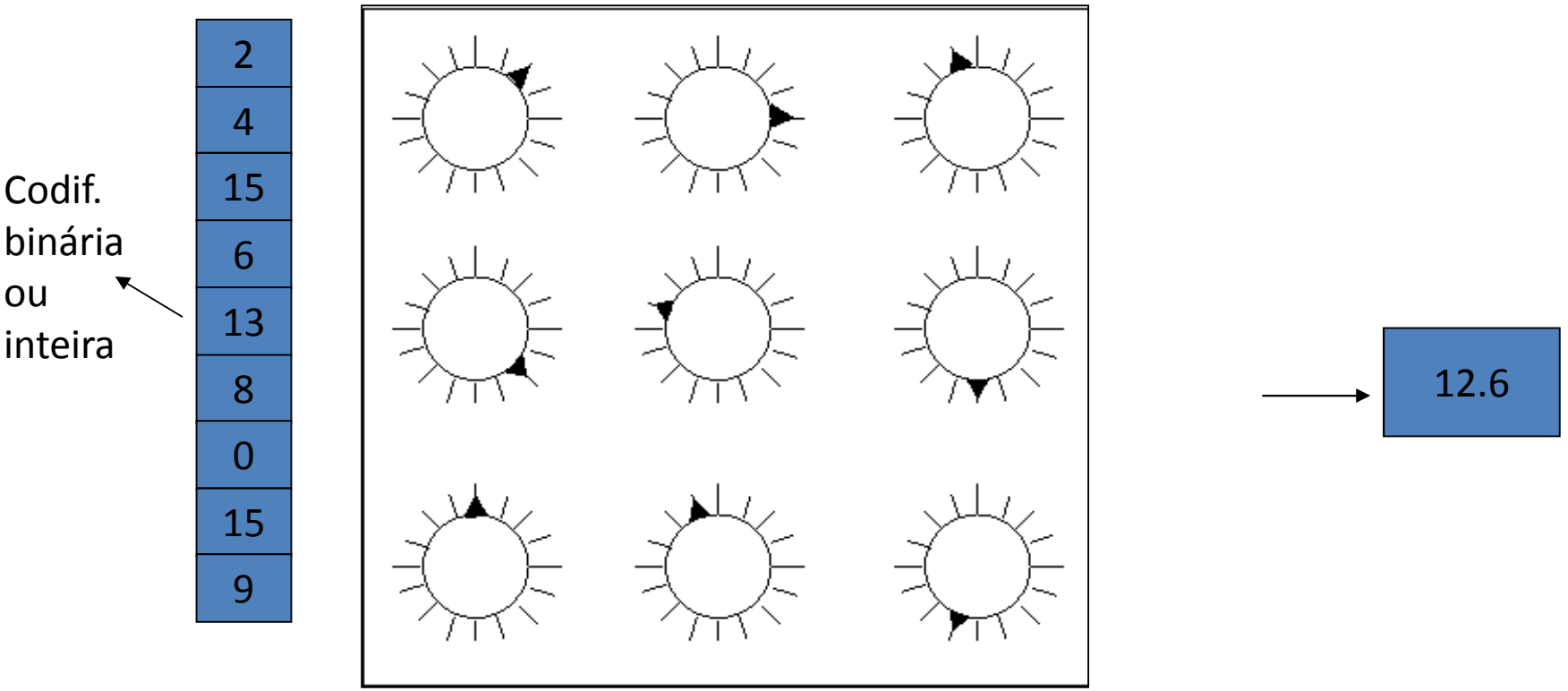
```
C:={c1, ..., cn}   população inicial de tamanho N
R tamanho da descendência ou reprodução (normalmente K = N)
Pc: probabilidade de fazer crossover (valor típico 0.8)
Pm: Probabilidade de fazer mutação nos alelos (valor típico 0.05)

AGCanônico(C, N, R, pCROSS, pMUT) {
   Para todo ci de C, calcular fitness(ci);
   geracao:=0;
   do {
      // D = Descendentes = nova geração, calculada a partir de C
      D := selecionar R cromossomos de C pelo método da roleta;
      D' := cruzamento(D); // geração de dois filhos por par (d1,d2),(d3,d4),...,(dk-1, dk)
          de D fazendo crossover com probabilidade Pc
      D" := para todo cromossomo di de D', para cada alelo aj de di, mutar aj
          com probabilidade Pm;
      Para todo di de D", calcular fitness(di);
      // Selecionar melhores entre pais e filhos
      C := selecionar n melhores cromossomos de C ⊔ D";
      geracao++;
   } while (geracao<MAX_GERACOES and !objetivo-alcançado and
            !melhor-fitness estagnado);

   retornar ci de C com melhor fitness;
}
```

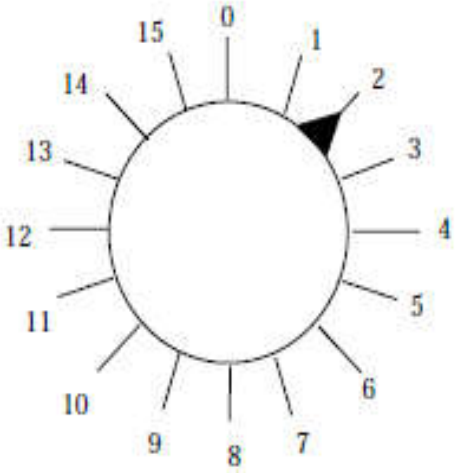
Solução de Problema por AG

Sabendo que cada botão pode ser colocado em 16 posições distintas, encontre a melhor combinação de posições para os 9 botões disponíveis na superfície da caixa preta de modo que o sinal de saída assuma o valor máximo.



AG Canônico no problema dos botões

- **Codificação**: Existem 16 posições possíveis para cada um dos 9 botões. Na codificação binária, 4 bits são suficientes para representar cada uma das 16 posições.



Posição Atual: 0010

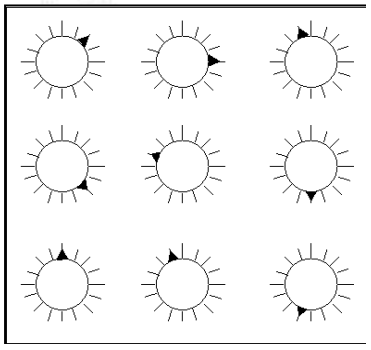
Posição	Representação	Posição	Representação
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

AG Canônico no problema dos botões

Baseado neste tipo de codificação, cada cromossomo associado à solução candidata do problema definido anteriormente é dado por uma seqüência de 36 bits $(b_1, \cdots, b_{36})$, na qual o número de possíveis configurações de botões (soluções candidatas) é $2^{36} \cong 68.72$ bilhões. Neste caso, a solução candidata, mostrada na figura seria codificada por um cromossomo na forma:

$$0010\,0100\,1111\,0110\,1101\,1000\,0000\,1111\,1001\,.$$

Função de Fitness



AG Canônico no problema dos botões

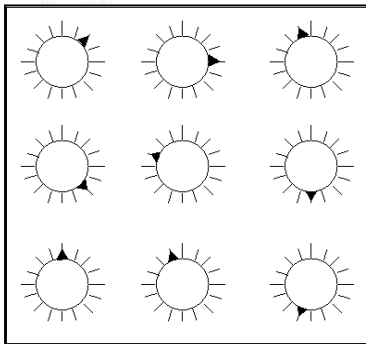
Baseado neste tipo de codificação, cada cromossomo associado à solução candidata do problema definido anteriormente é dado por uma seqüência de 36 bits $(b_1, \cdots, b_{36})$, na qual o número de possíveis configurações de botões (soluções candidatas) é $2^{36} \cong 68.72$ bilhões. Neste caso, a solução candidata, mostrada na figura seria codificada por um cromossomo na forma:

$$0010\,0100\,1111\,0110\,1101\,1000\,0000\,1111\,1001\ .$$

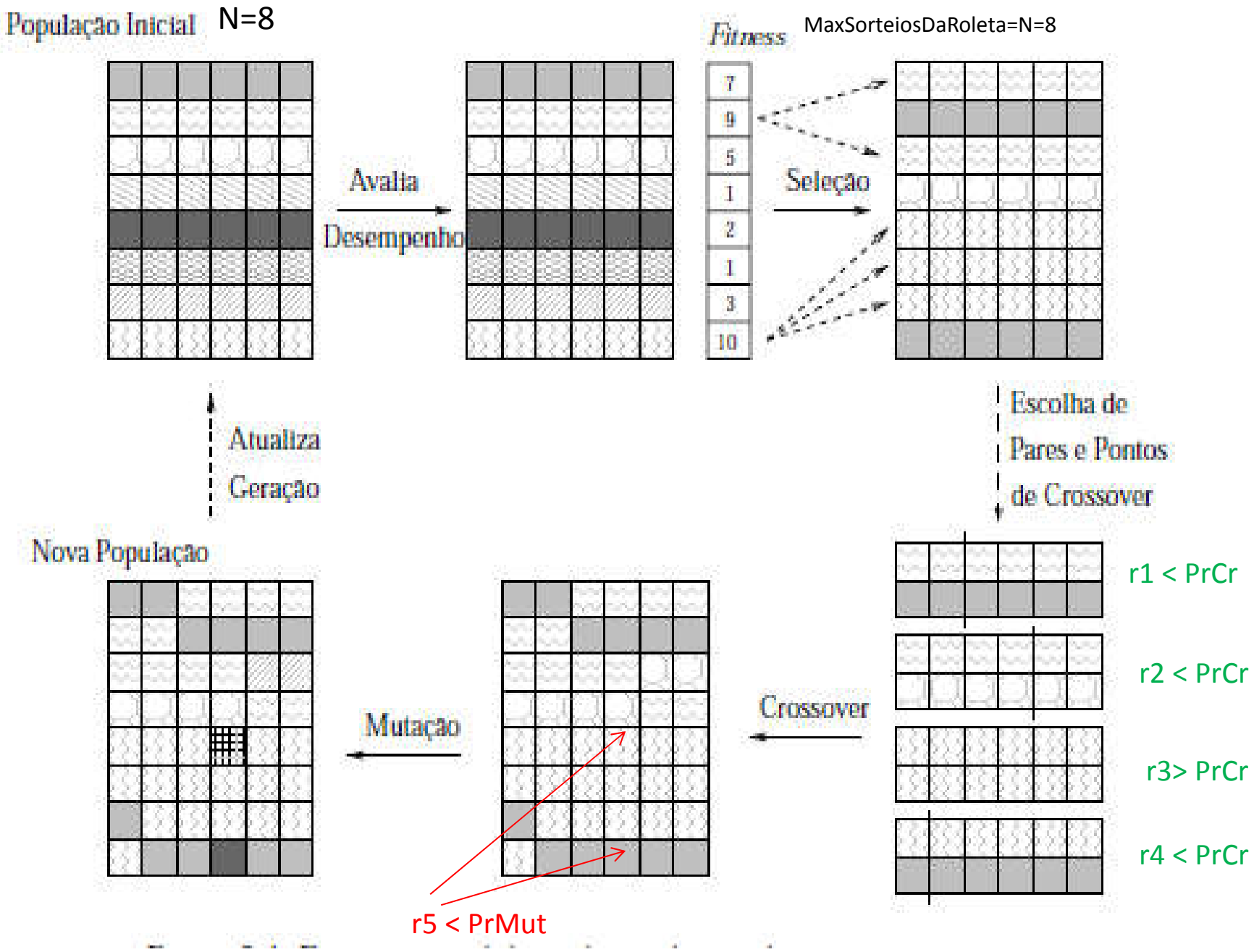
O mapeamento, suposto desconhecido, entre as 2^{36} posições possíveis dos botões e o sinal de saída é dado por:

$$\begin{aligned}\text{sinal de saída} = {} & 9 + b_2 b_5 - b_{23} b_{14} + b_{24} b_4 - b_{21} b_{10} + b_{36} b_{15} - b_{11} b_{26} + b_{16} b_{17} + b_3 b_{33} \\ & + b_{28} b_{19} + b_{12} b_{34} - b_{31} b_{32} - b_{22} b_{25} + b_{35} b_{27} - b_{29} b_7 + b_8 b_{13} - b_6 b_9 + b_{18} b_{20} - b_1 b_{30} \\ & + b_{23} b_4 + b_{21} b_{15} + b_{26} b_{16} + b_{31} b_{12} + b_{25} b_{19} + b_7 b_8 + b_9 b_{18} + b_1 b_{33}\ .\end{aligned}$$

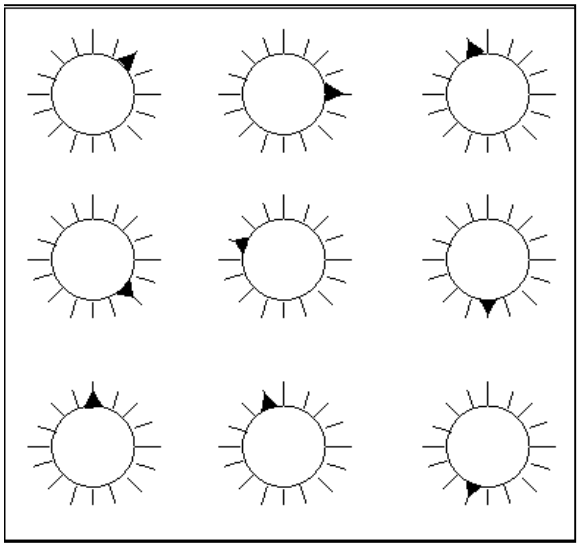
Função de Fitness

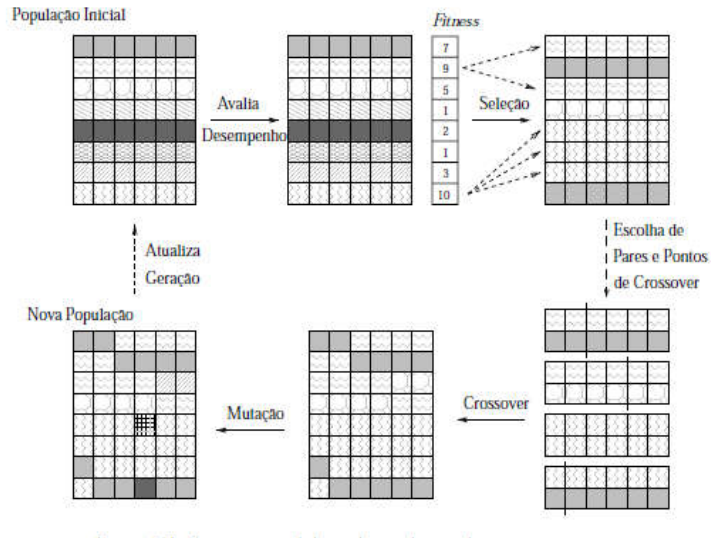


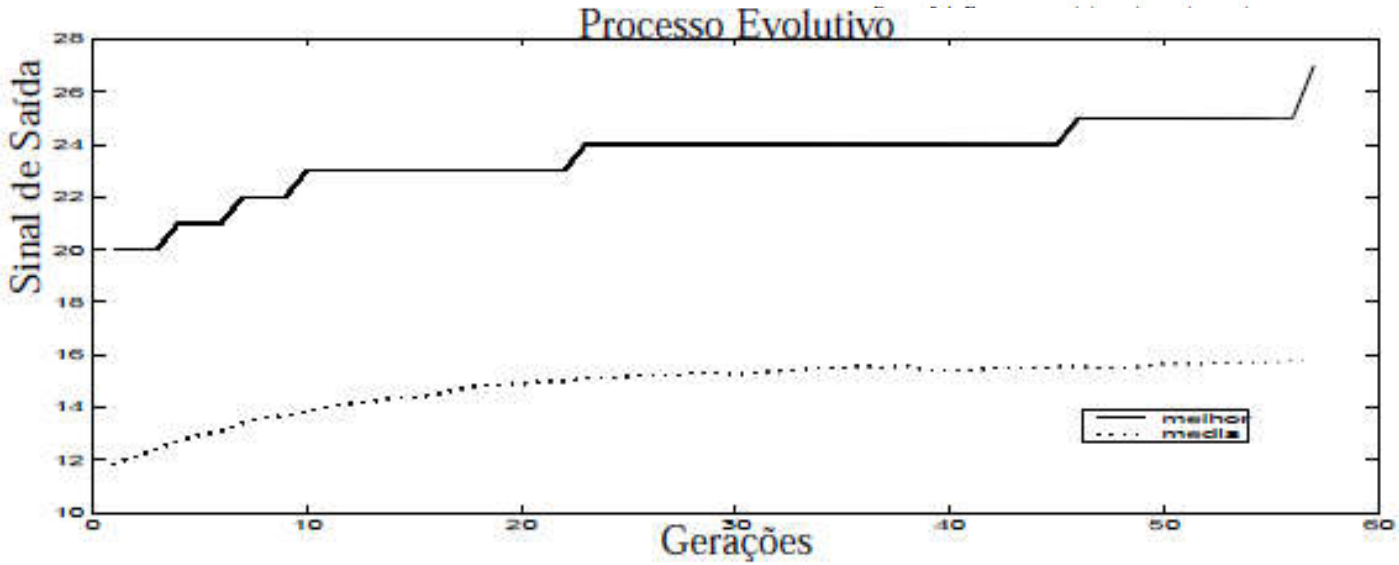
Esquema Geral de um Agoritmo Genético



AG Canônico no problema dos botões







Melhor fitness

Fitness médio da População

AG: Formulação do Problema

Problema das 8 rainhas

Solução por Codificação binária (matriz binária) -> vetor de bits

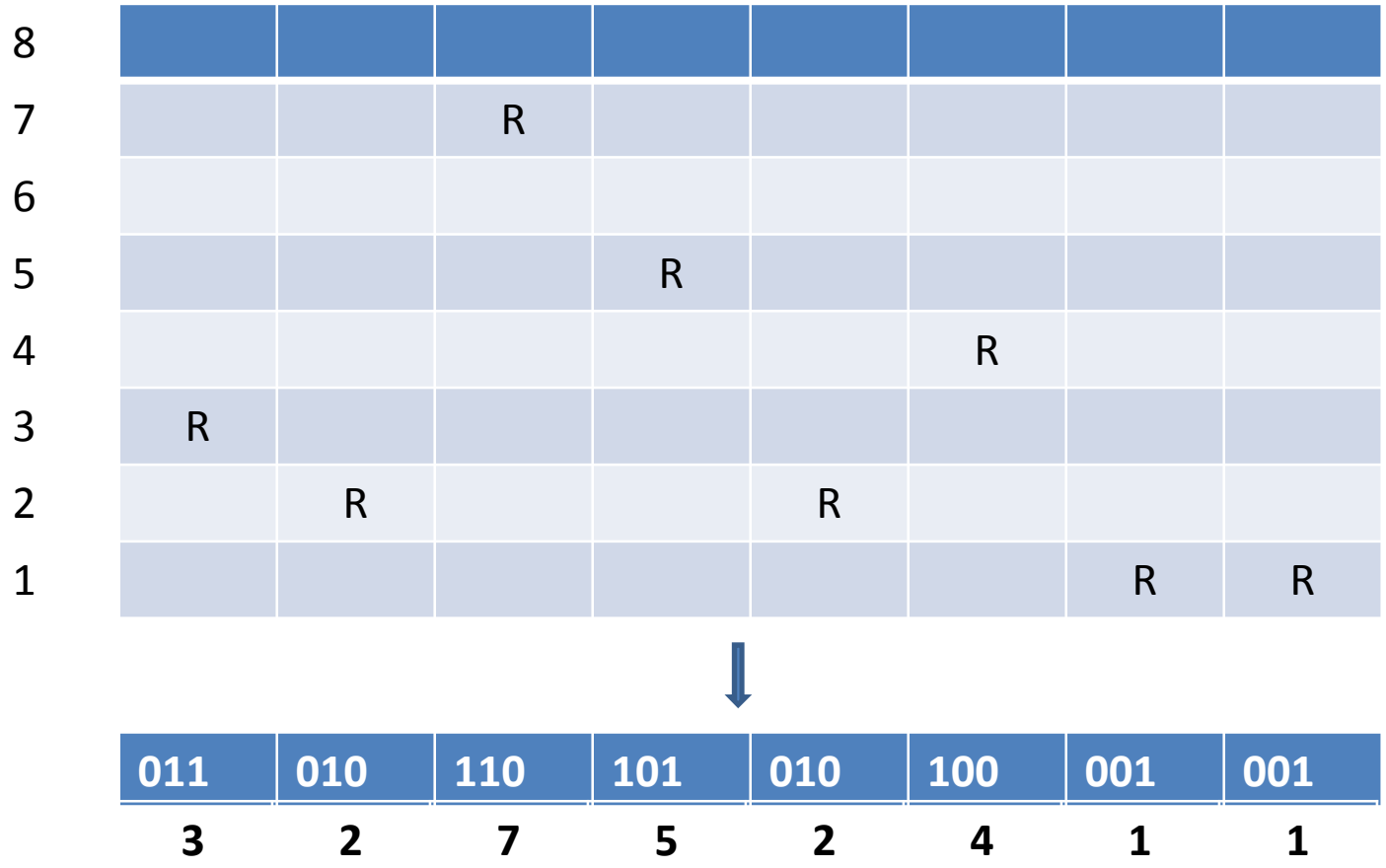
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0
0	0	0	0	0	0	1	1

0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	.		

AG: Formulação do Problema

Problema das 8 rainhas

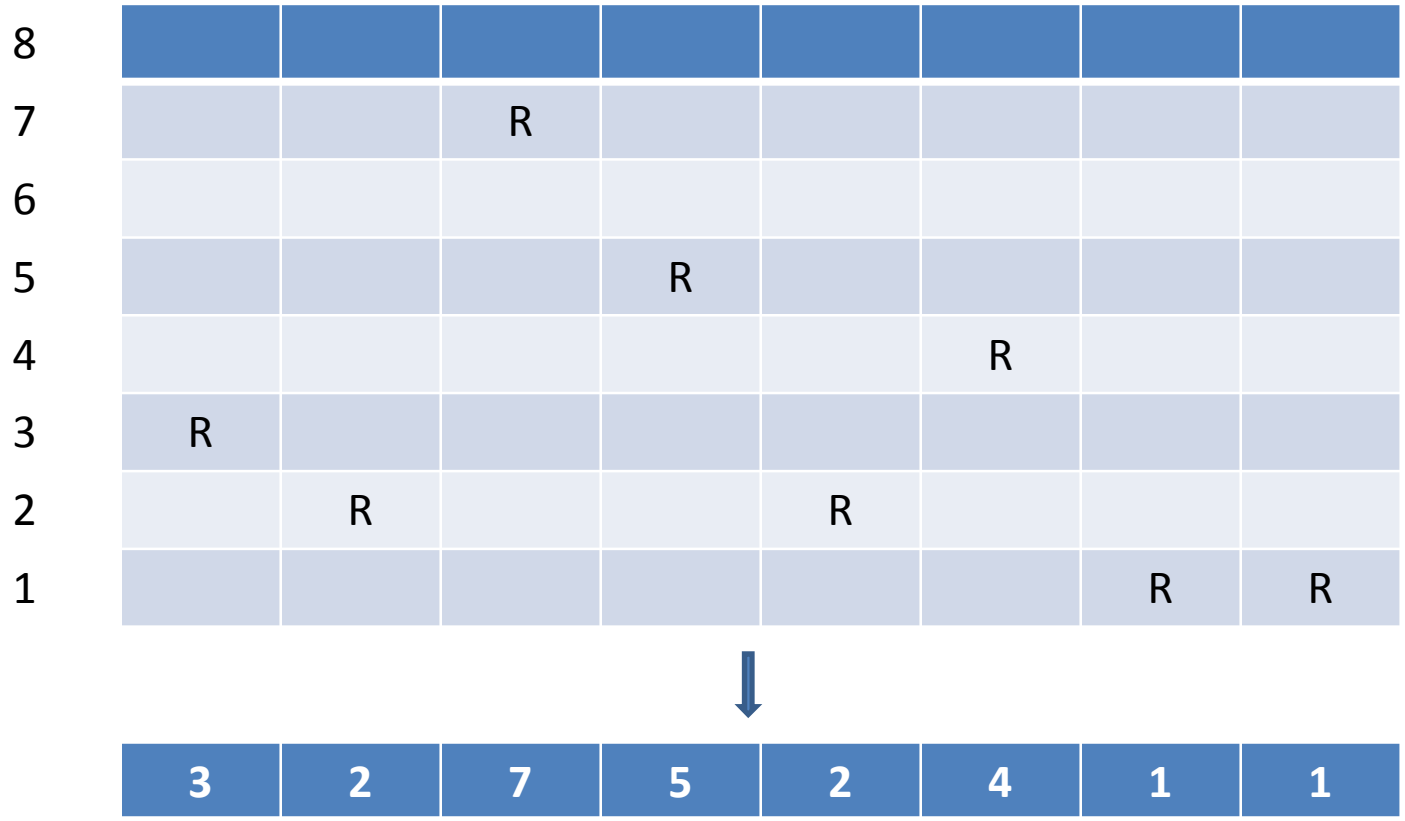
Solução por codificação binária (obtida por conversão inteiro-> binário)



AG: Formulação do Problema

Problema das 8 rainhas

Solução por Codificação inteira



AG: Formulação do Problema

Diferentes formulações podem levar o AG a solucionar diferentes problemas:

AG Canônico para Problemas com Codificação binária

- Ex. Problema das 8 rainhas com cod binária

AG para Otimização Combinatória

- Ex. Problema de otimização de rotas (permutação de cidades)
- Ex. Problema das 8 rainhas (permutação da posição das rainhas)

AG para Otimização Contínua

- Ex. Problema de otimização de funções contínuas

AG em prob. codificação binária

- Indivíduo: Codificação Binária
- Seleção
  - Reprodução (roleta)
- Reprodução
  - Crossover simples
  - Mutação Uniforme
- Sobrevivência
  - Melhores para compor a nova população

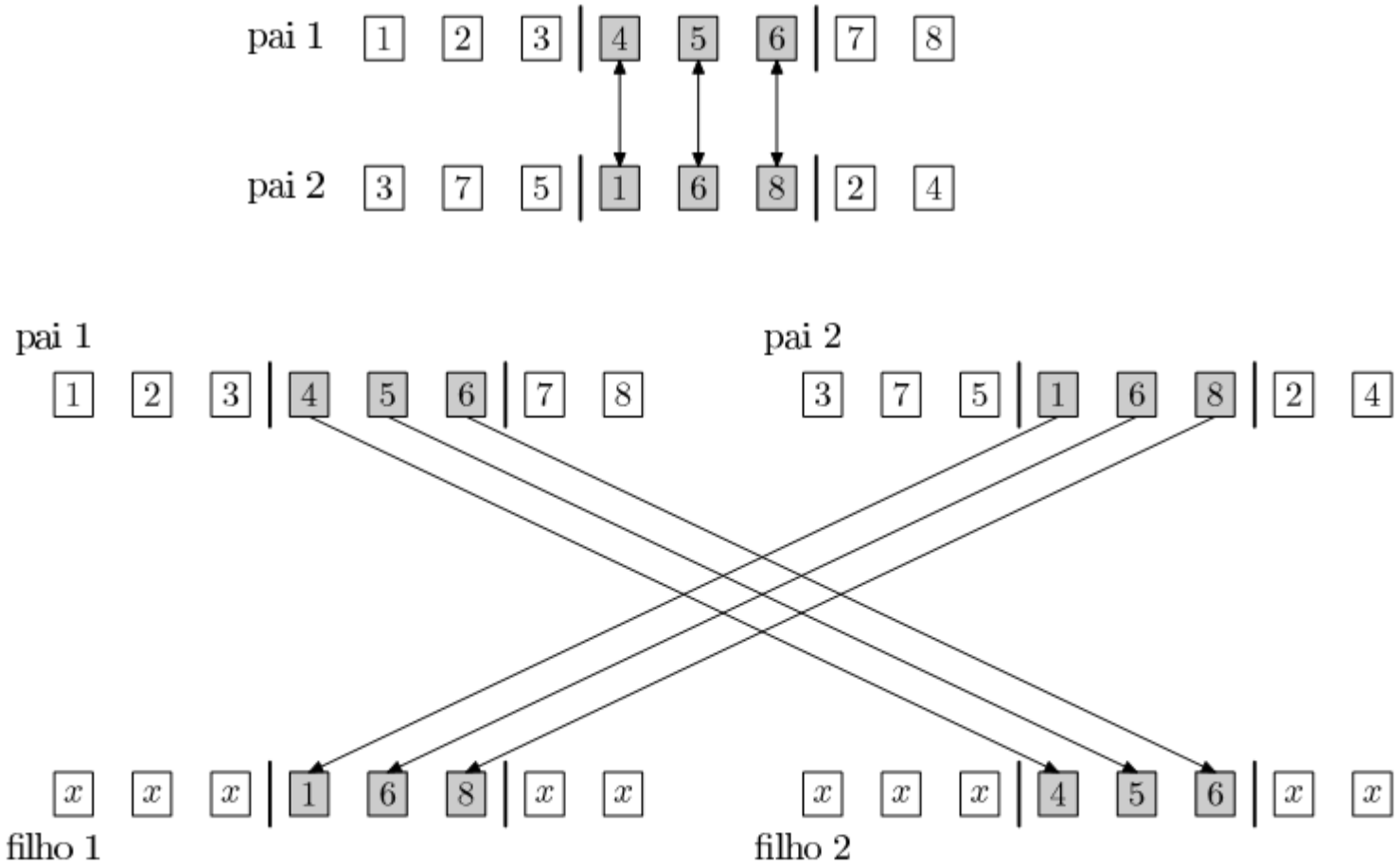
AG em Otimização Combinatória

Indivíduos que codificam uma permutação

Como realizar o crossover e a mutação?

AG em Otimização Combinatória

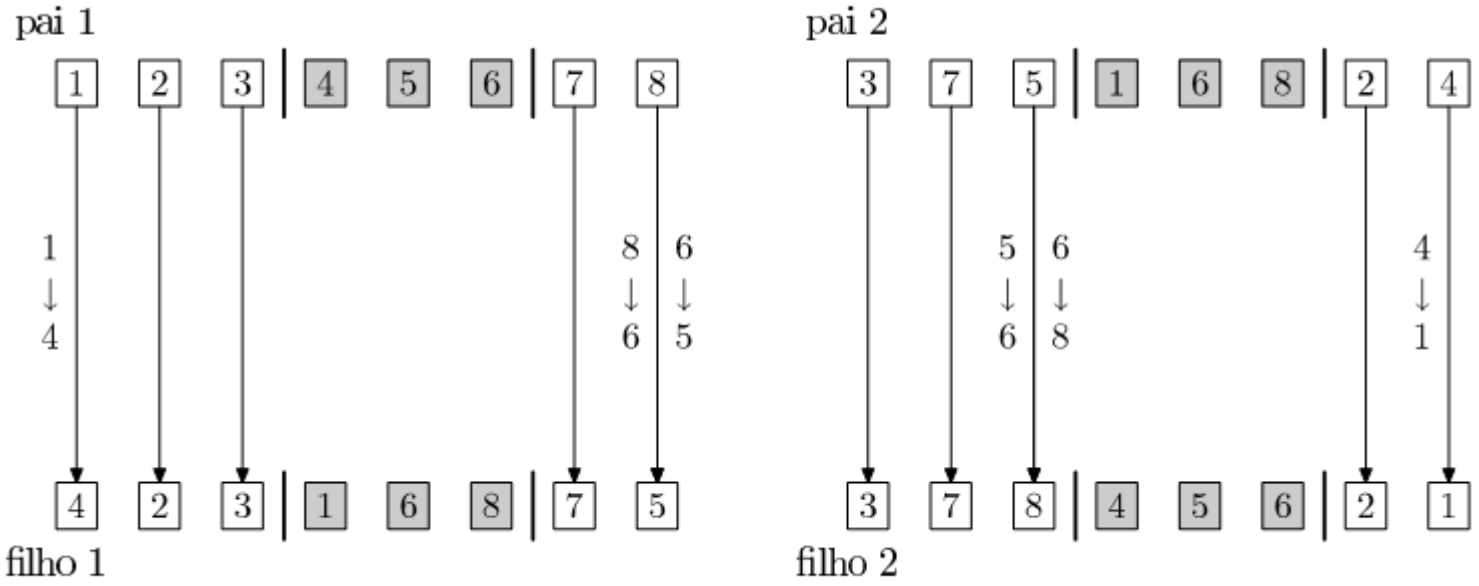
Crossover de mapeamento parcial (PMX):



AG em Otimização Combinatória

Crossover de mapeamento parcial (PMX):

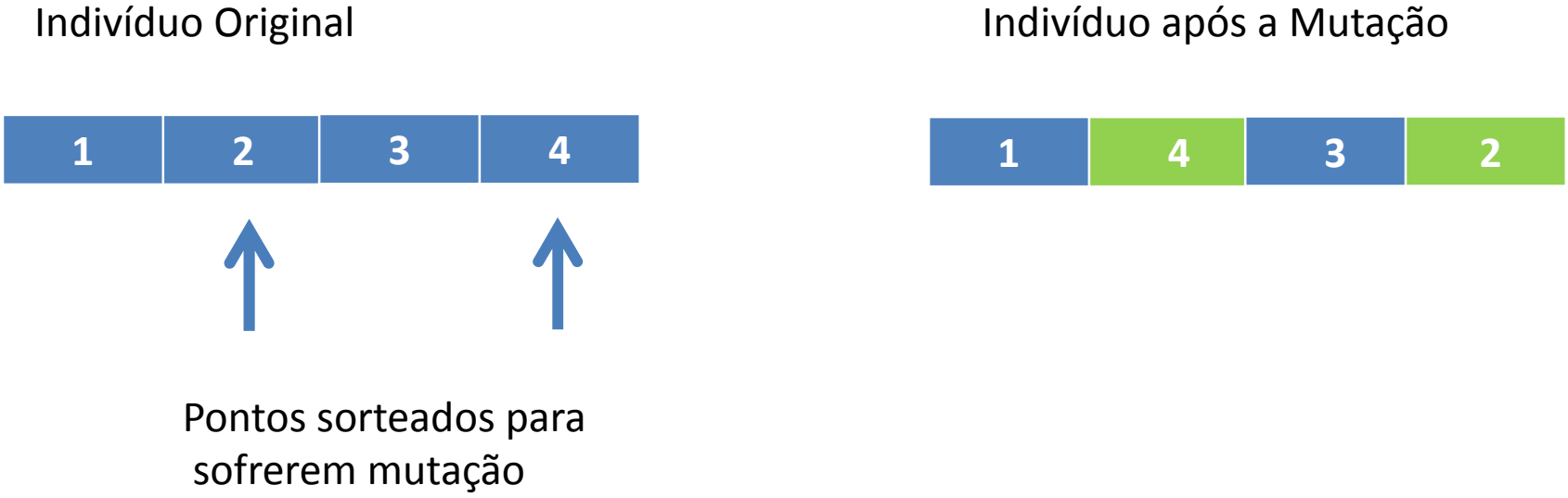
As subseqüências entre os pontos de corte são chamadas de seções de mapeamento. No exemplo eles definem os mapeamentos $4 \leftrightarrow 1$, $5 \leftrightarrow 6$ e $6 \leftrightarrow 8$.



AG em Otimização Combinatória

Mutação em indivíduos que codificam uma permutação:

Operação swap

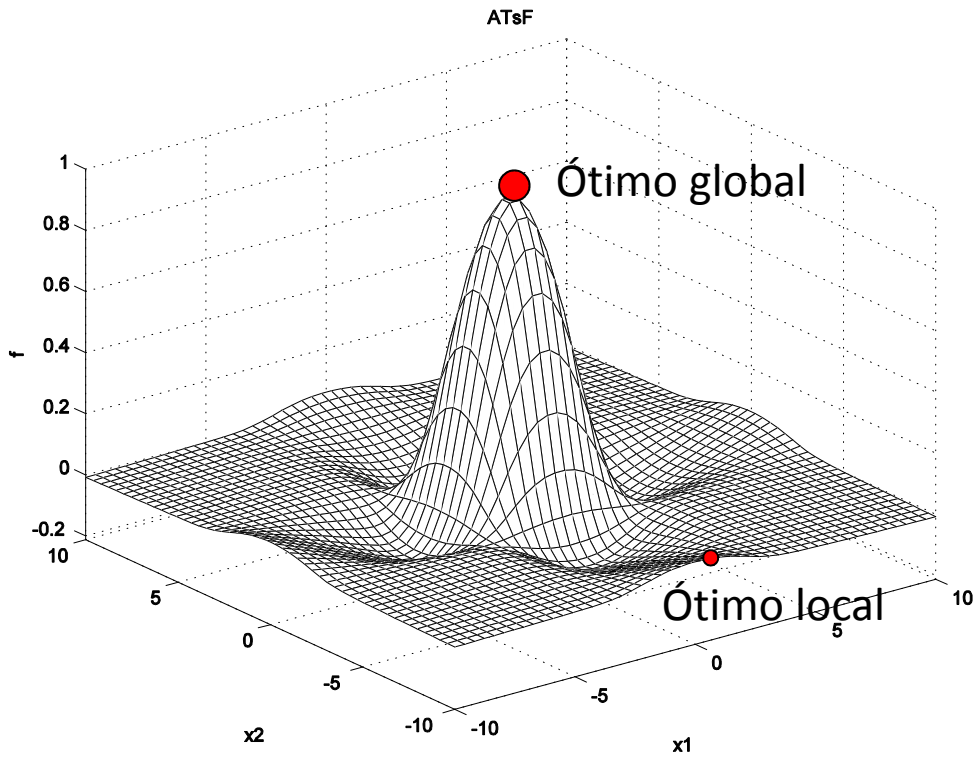


AG em Otimização Contínua

Codificação Real

S1= | -5.0 | -5.0 | X S2= | - 5.1 | -4.9 |

$$f(x_1, x_2) = \frac{\text{seno}(x_1)}{x_1} \frac{\text{seno}(x_2)}{x_2}$$



AG em Otimização Contínua

Fitness

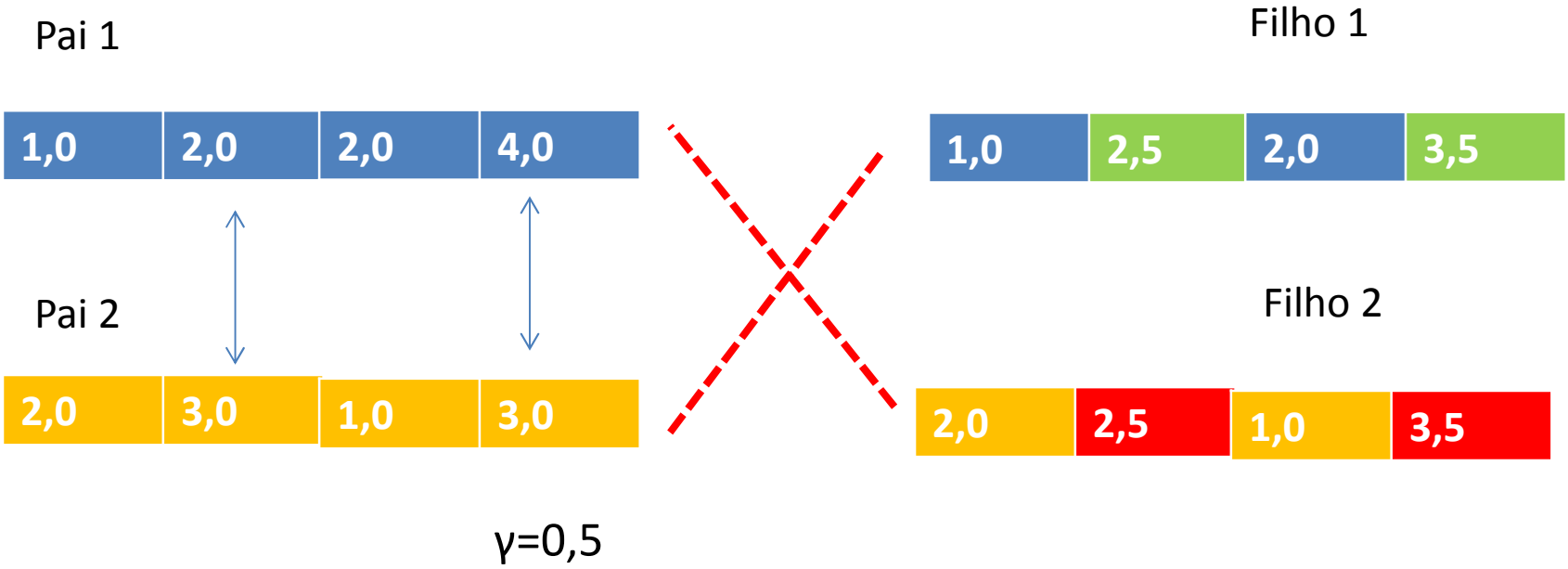
Pode ser a própria Função objetivo

$$fitness(s_i) = \frac{\text{seno}(x_1)}{x_1} \frac{\text{seno}(x_2)}{x_2}$$

AG em Otimização Contínua

Crossover aritmético (específico para codificação real)

$S'_{12j}=(1-\gamma)s_{1j} + \gamma s_{2j}$



AG em Otimização Contínua

Mutação não-uniforme

$$s'_{ij} = \begin{cases} s_{ij} + \Delta(g, \lim\_\sup - s_{ij}) & se \;\; r \leq 0{,}5 \\ s_{ij} - \Delta(g, s_{ij} - \lim\_\inf) & se \;\; r > 0{,}5 \end{cases}$$

$$onde \;\; r \sim U[0,1] \;\; \text{e} \;\; \Delta(g,x)$$

Retorna um valor
No intervalo [0,x] tal que a probabilidade do valor nulo cresce à medidade que g cresce

$$\Delta(g,x) = x\left(1 - r_2^{(1-g/MaxGer)b}\right)$$

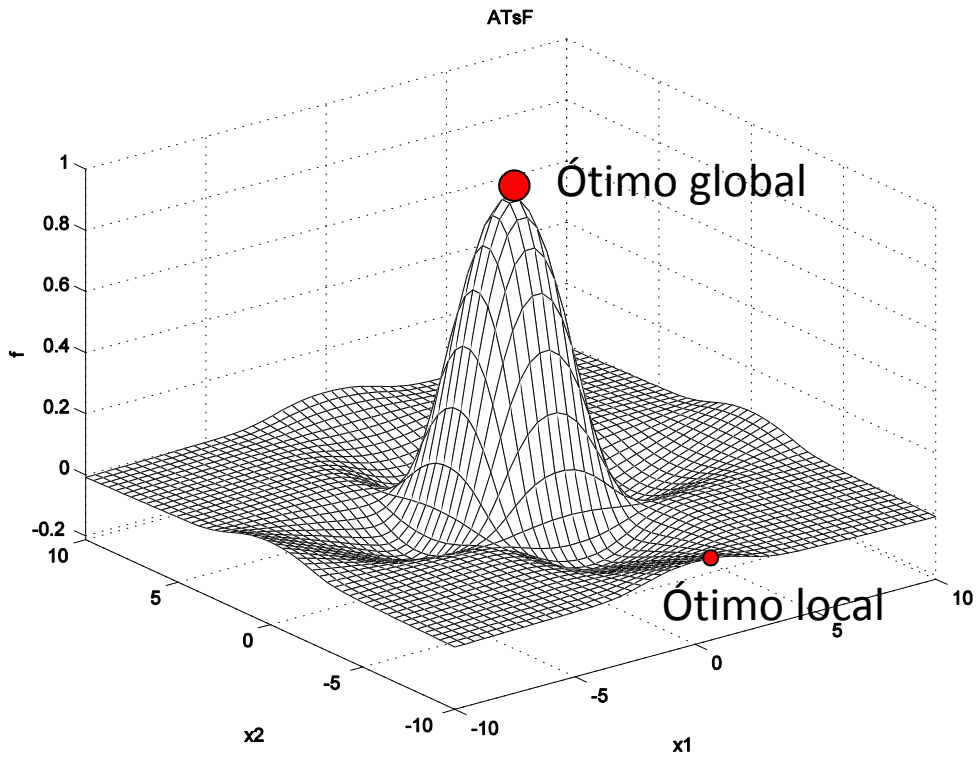
onde r_2 ~U[0,1], b parâmetros de uniformidade

AG em Otimização Contínua

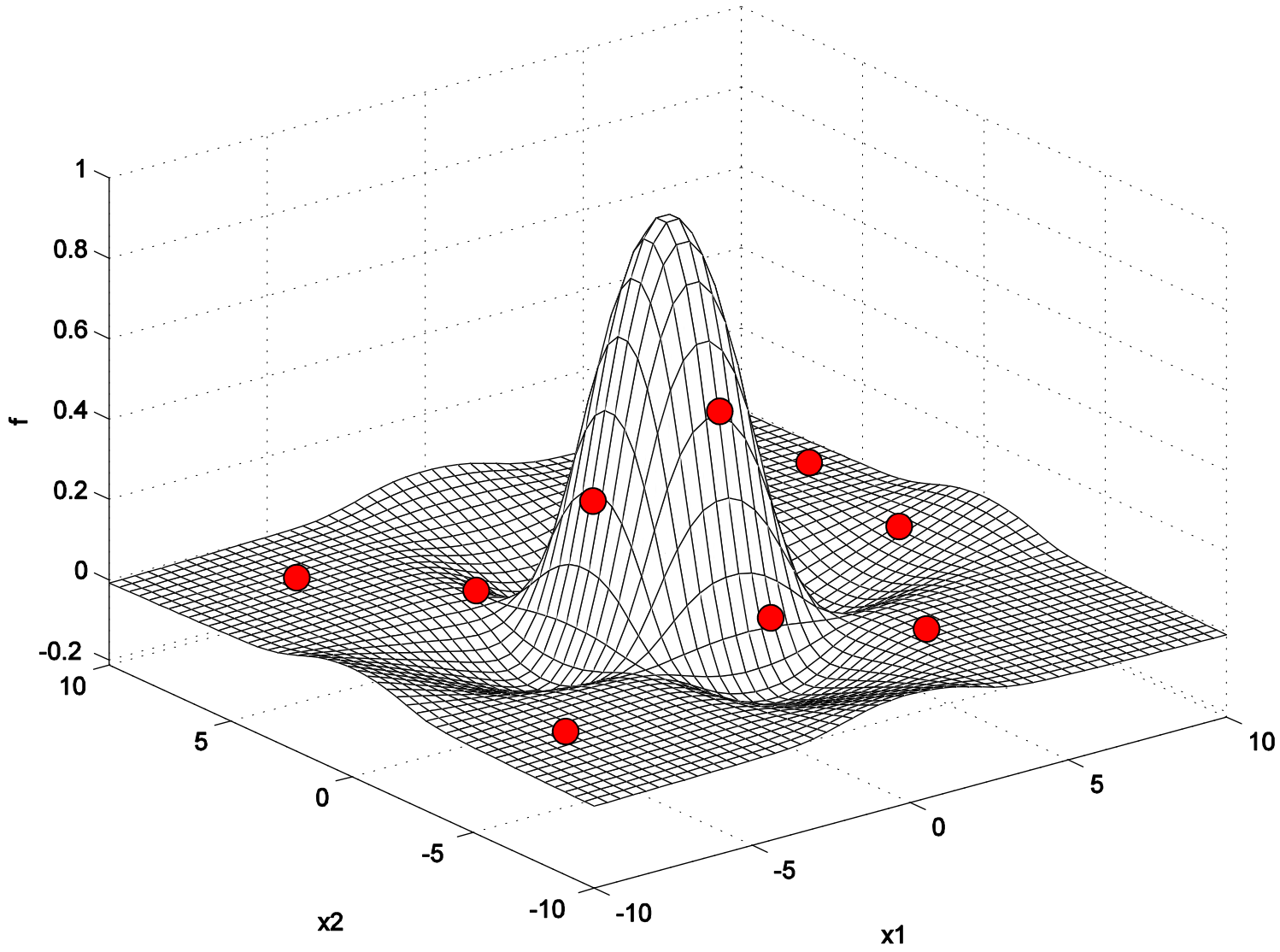
Codificação Real

S1= | -5.0 | -5.0 | X S2= | - 5.1 | -4.9 |

$$f(x_1, x_2) = \frac{\text{seno}(x_1)}{x_1} \frac{\text{seno}(x_2)}{x_2}$$



AG em Otimização Contínua



Programação Genética

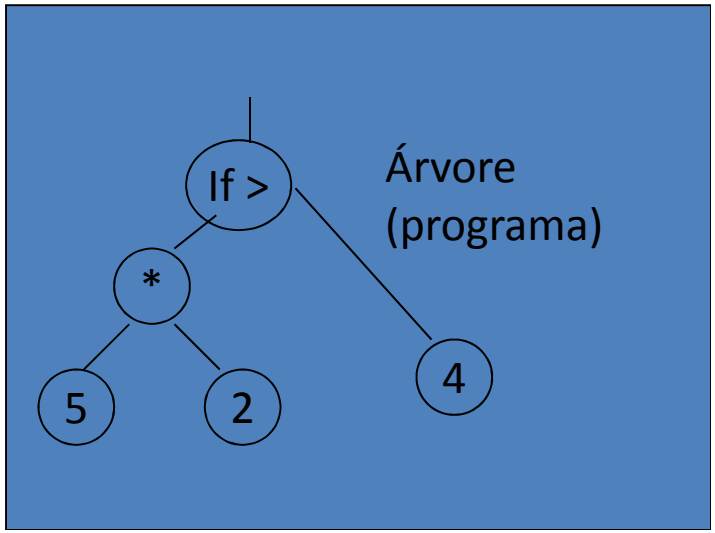
- Cada indivíduo representa um programa de computador codificado através de uma árvore

- GP: indivíduos de tamanhos, formatos e complexidade diferentes

- GP:Gramática dependende

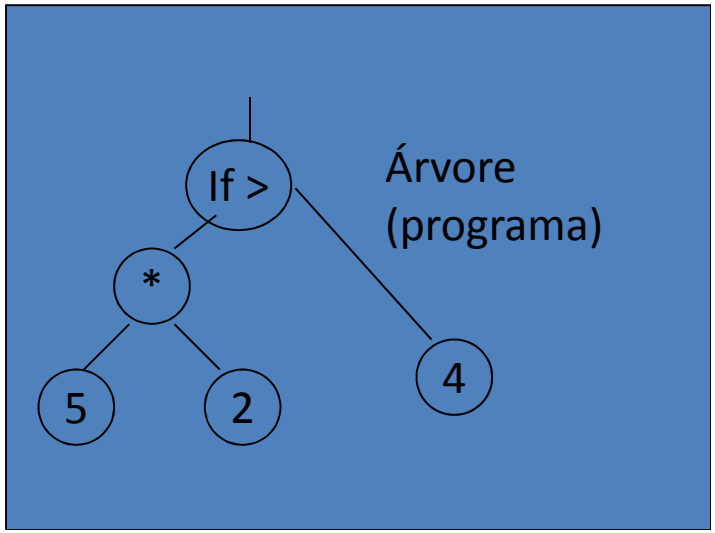
do problema (qualquer

solução deve ser possível)

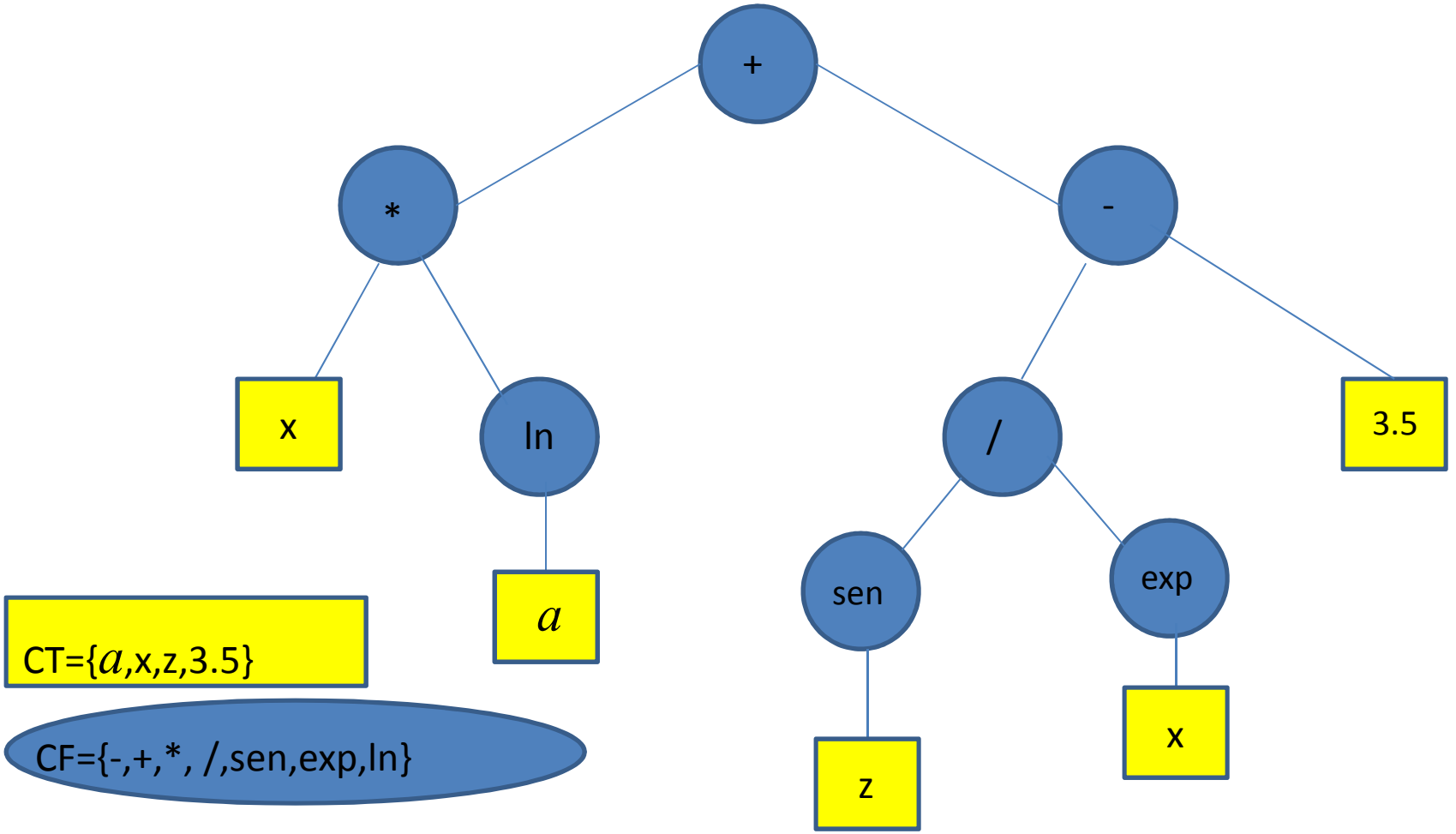


Programação Genética

- Gramática
  - Conjunto de Terminais (CT)
  - Conjunto de Funções (CF)
  - Regras semânticas (RS)

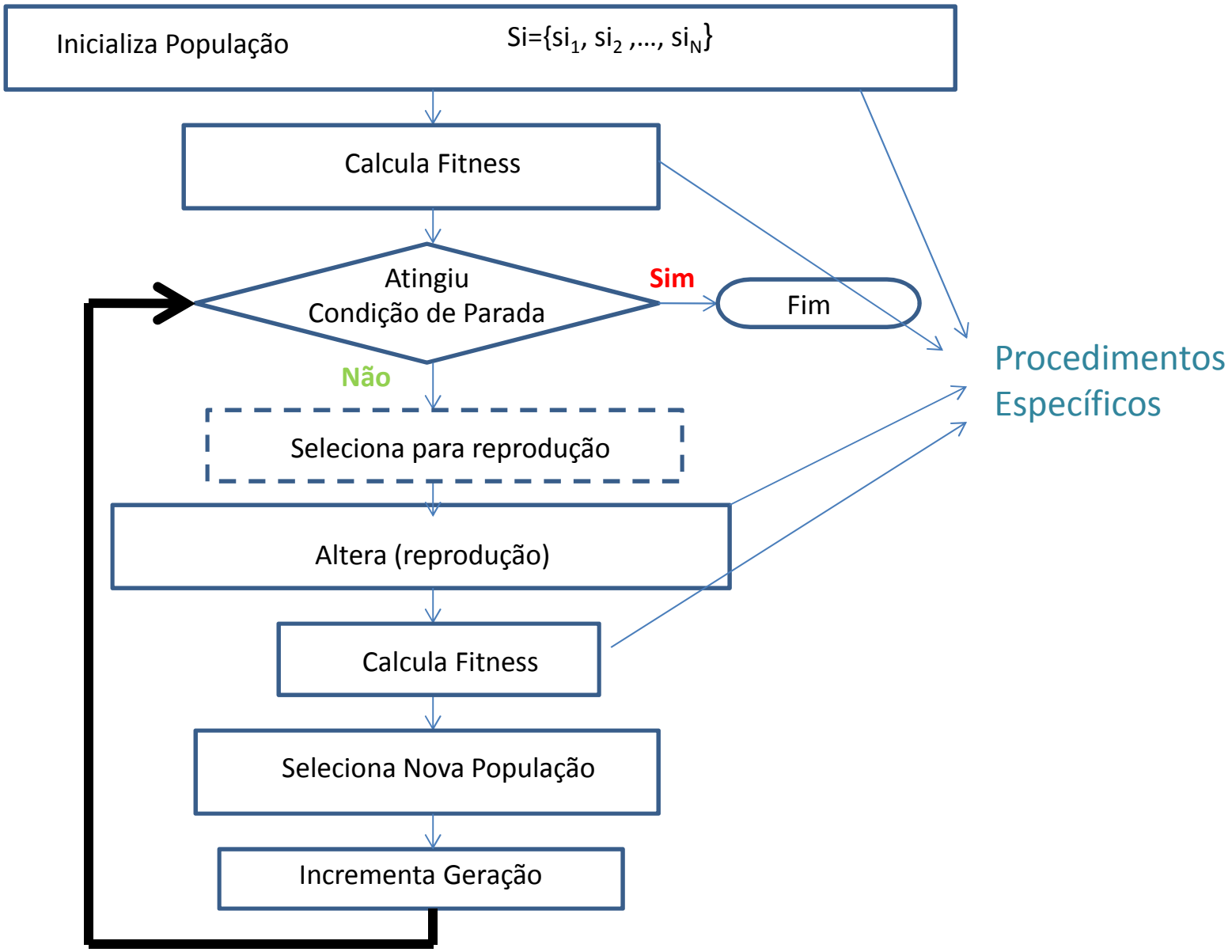


PG: Codificação em Árvore



$$y = x * \ln(a) + sen(z) / \exp(x) - 3.5$$

Esquema Geral de um Algoritmo PG



PG: Inicialização da População

Inicialização Aleatória

Para cada indivíduo um nó raiz (s_{i0}) é selecionado aleatoriamente do conjunto de funções

O número de filhos do nó raiz é determinado pela aridade da função esscolhida

Para cada nó (não raiz) o processo de inicialização define
se o elemento (s_{ij}) será escolhido do conjunto CT ou CF

- Se s_{ij} pertence a CT então o nó s_{ij} é folha e não será mais considerado para expansão

- Senão, o número de filhos de s_{ij} é determinado pela aridade da função escolhida e o nó será considerado para expansão

PG: Cálculo do Fitness

Fitness: Dependente do Problema

Expressão matemática: distância para o valor esperado

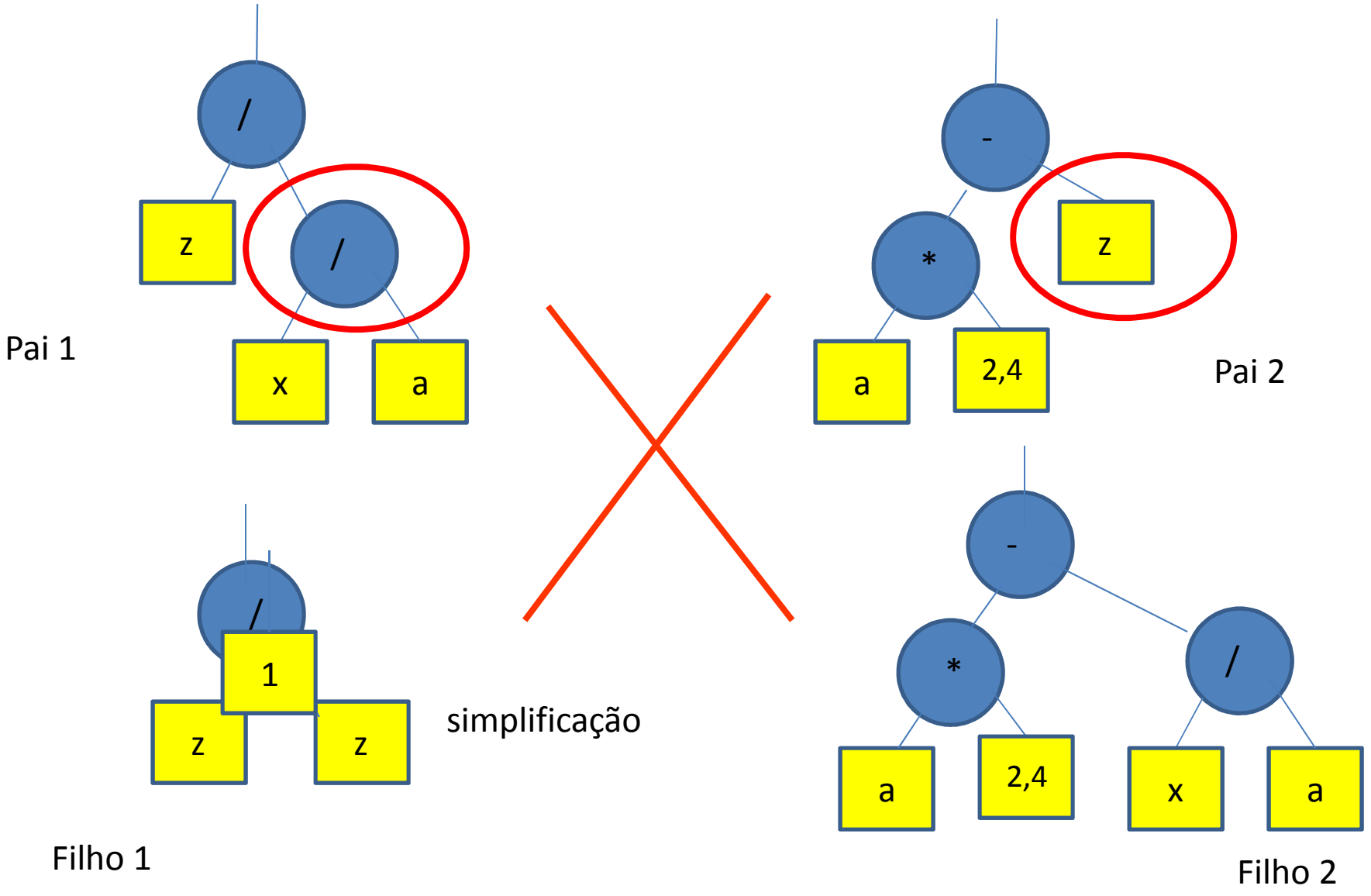
Programa: total de saídas incorretas, loops infinitos, etc…

PG: Operadores de Reprodução

Crossover: troca de material genético entre dois pares

Mutação: introdução de novo material genético

PG: Crossover



PG: Operadores de Reprodução

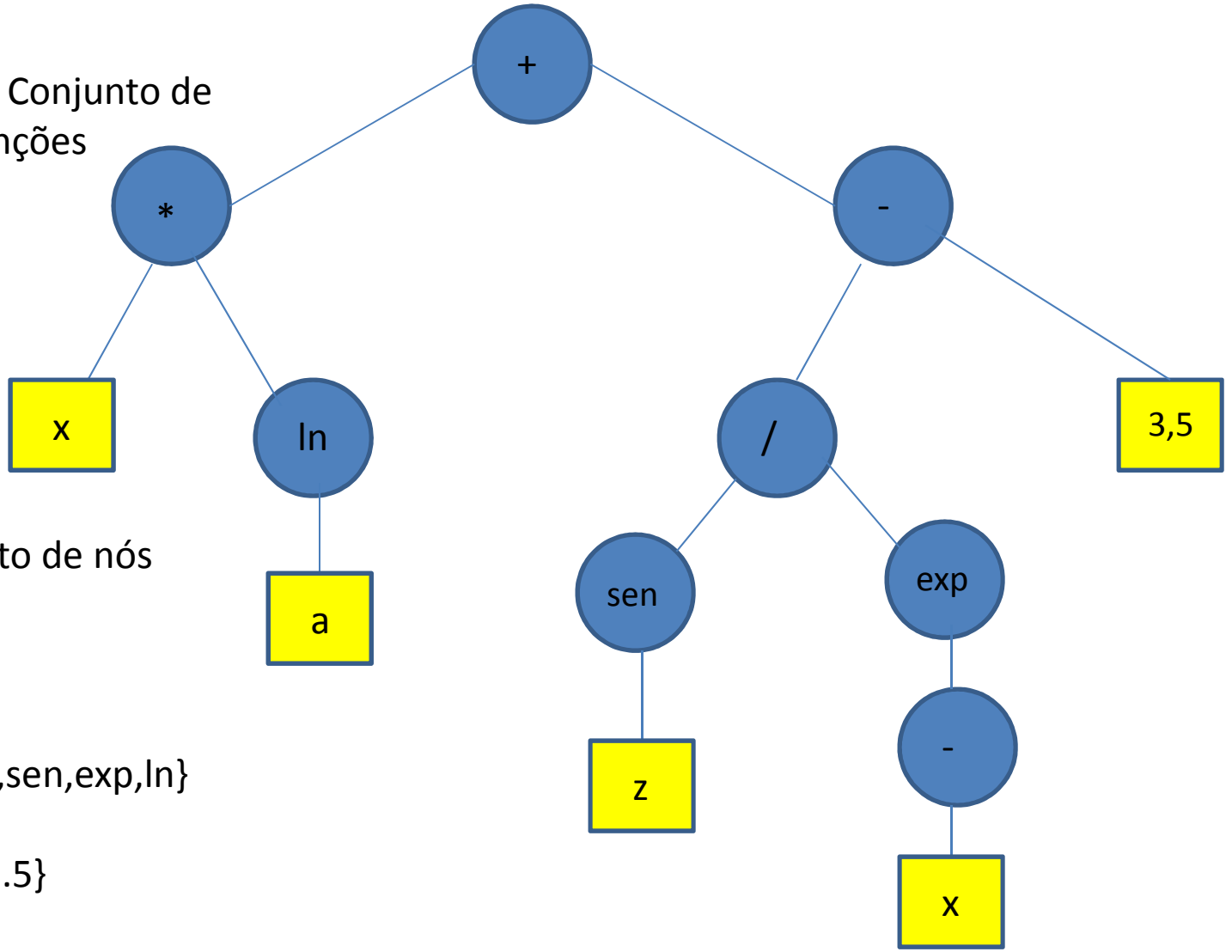
Crossover: troca de material genético entre dois pares

Mutação: introdução de novo material genético

- Nó do Conjunto de Funções
- Nó Folha
- Nó qualquer

PG: Mutação

No Conjunto de Funções

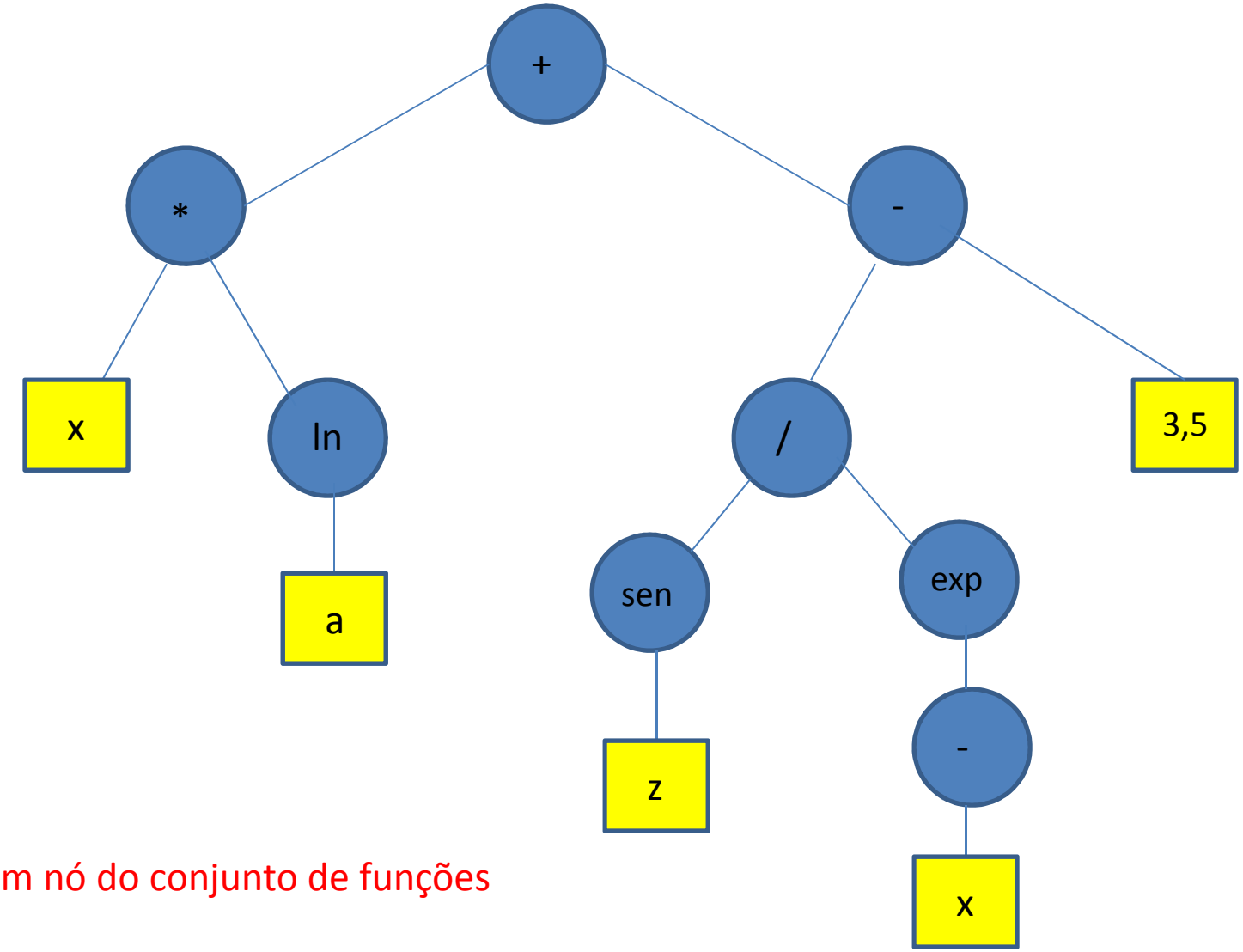


No Conjunto de nós folhas

CF={-,+,*,/,sen,exp,ln}

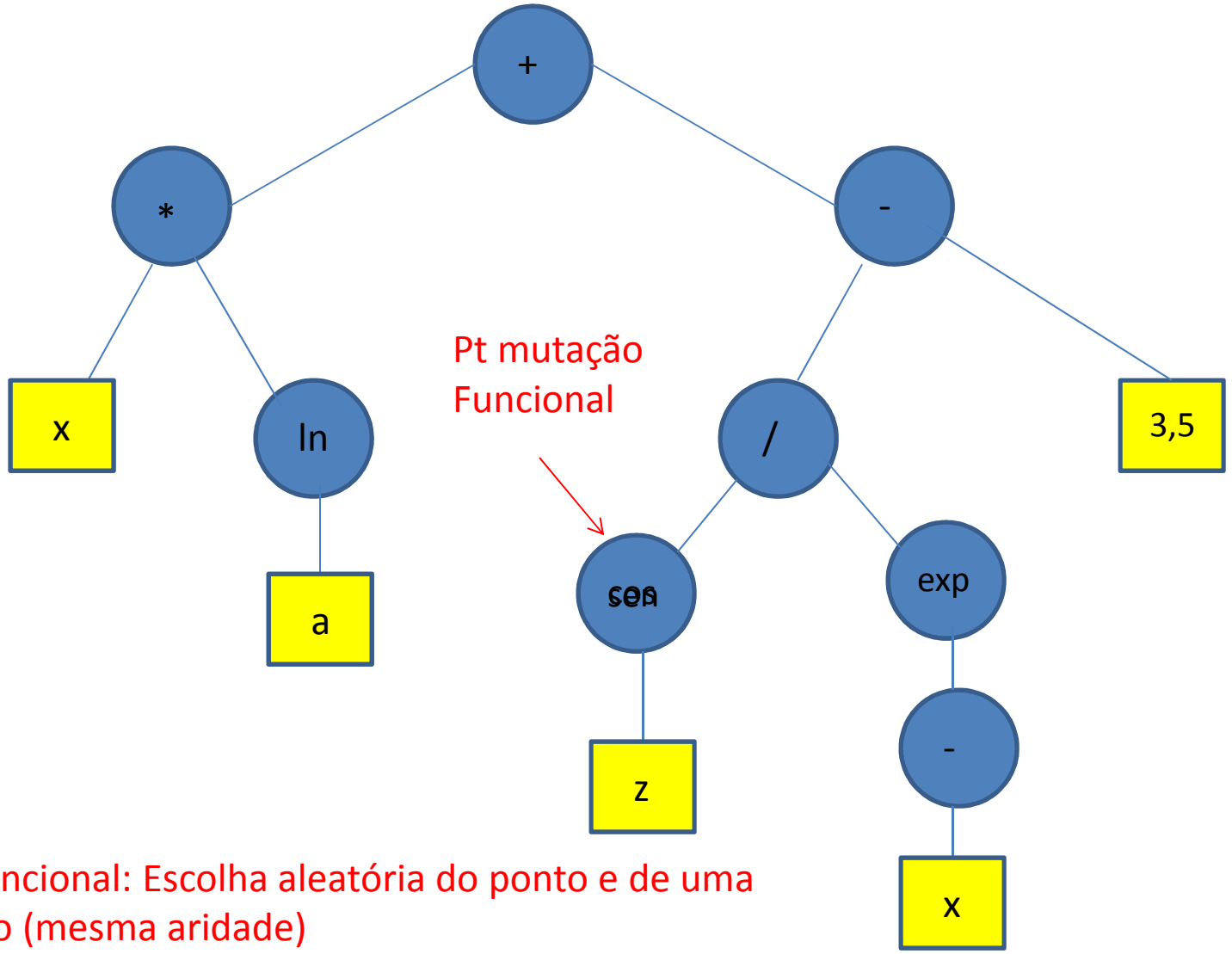
CT={a,x,z,3.5}

PG: Mutação



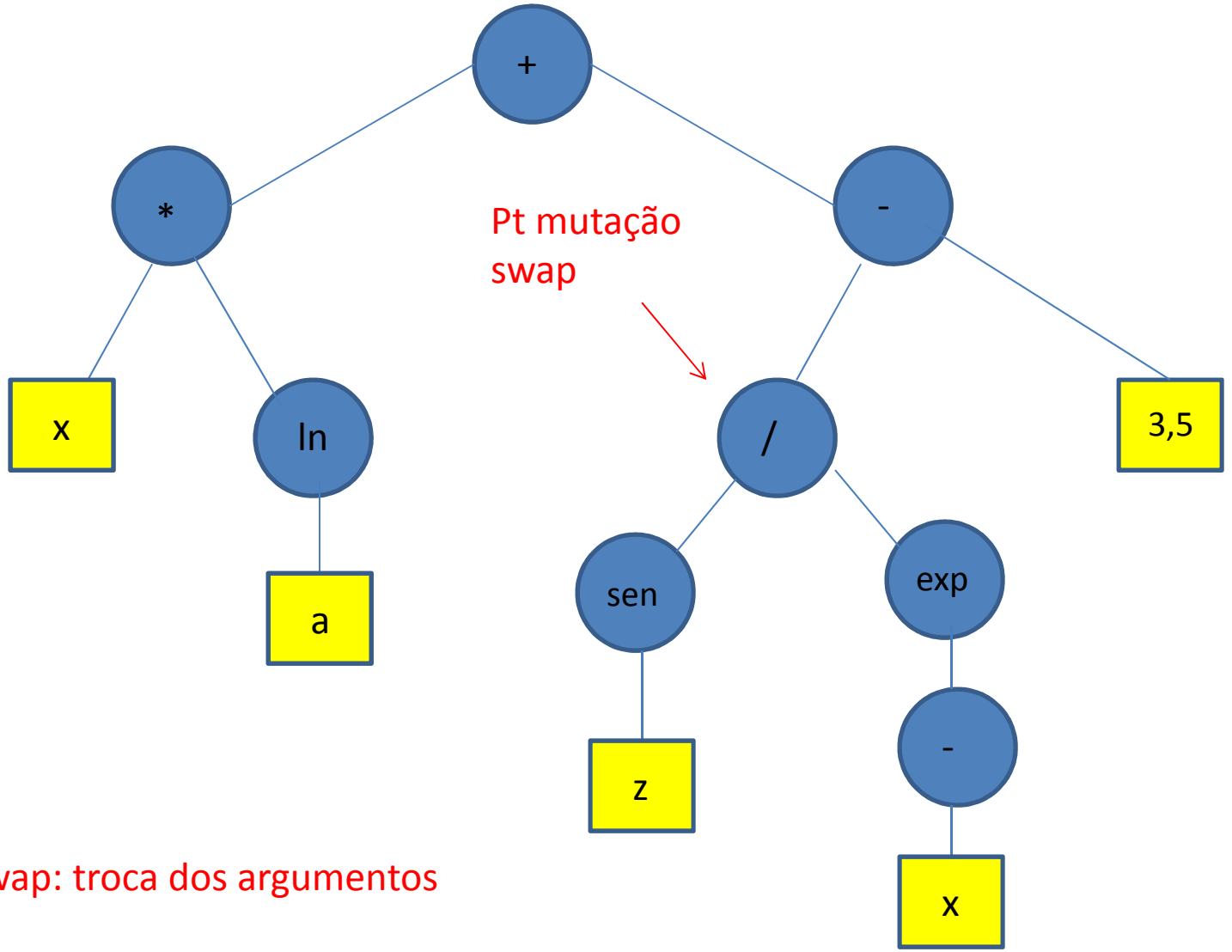
Mutação em nó do conjunto de funções

PG: Mutação em nó do CF



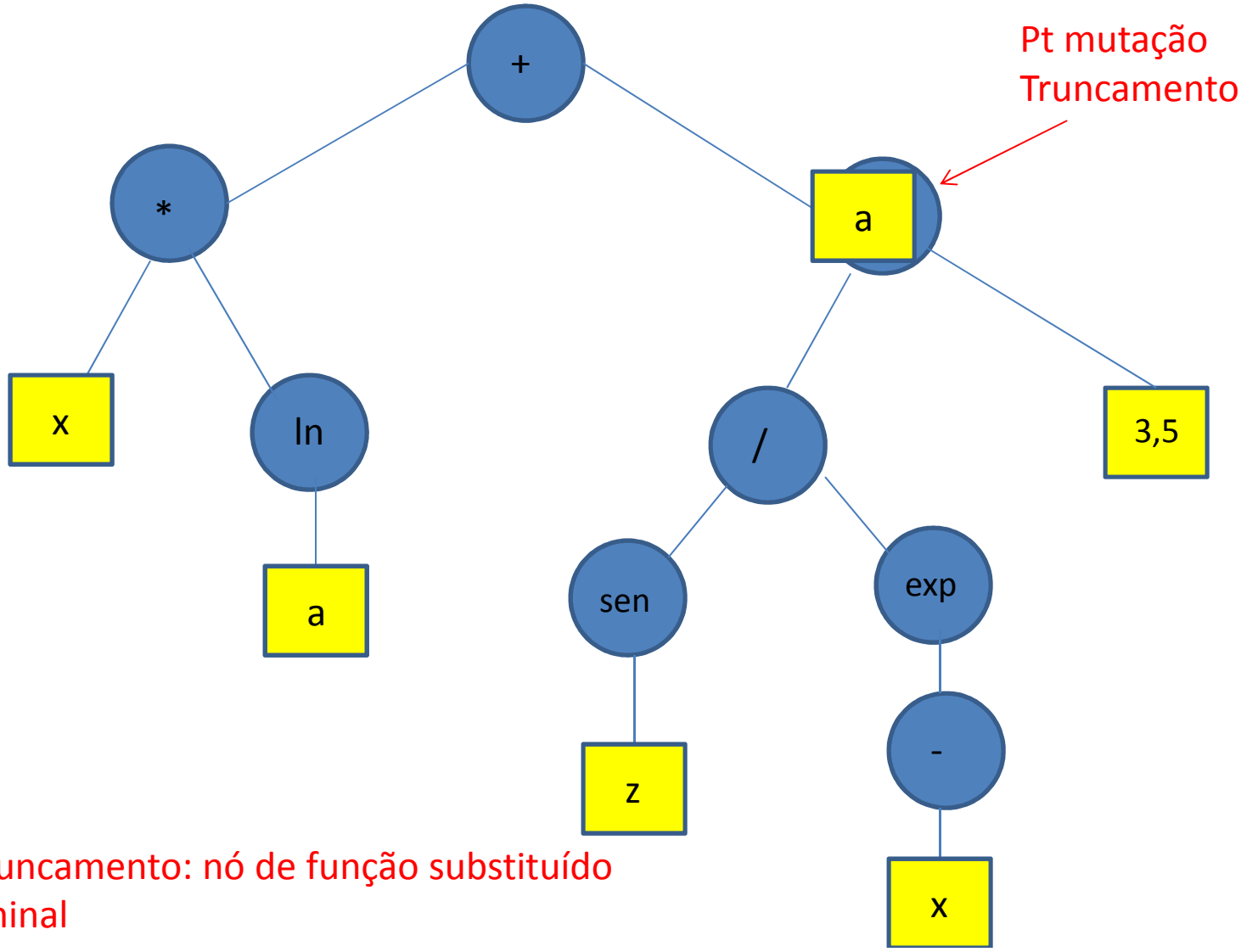
Mutação funcional: Escolha aleatória do ponto e de uma nova função (mesma aridade)

PG: Mutação em nó do CF



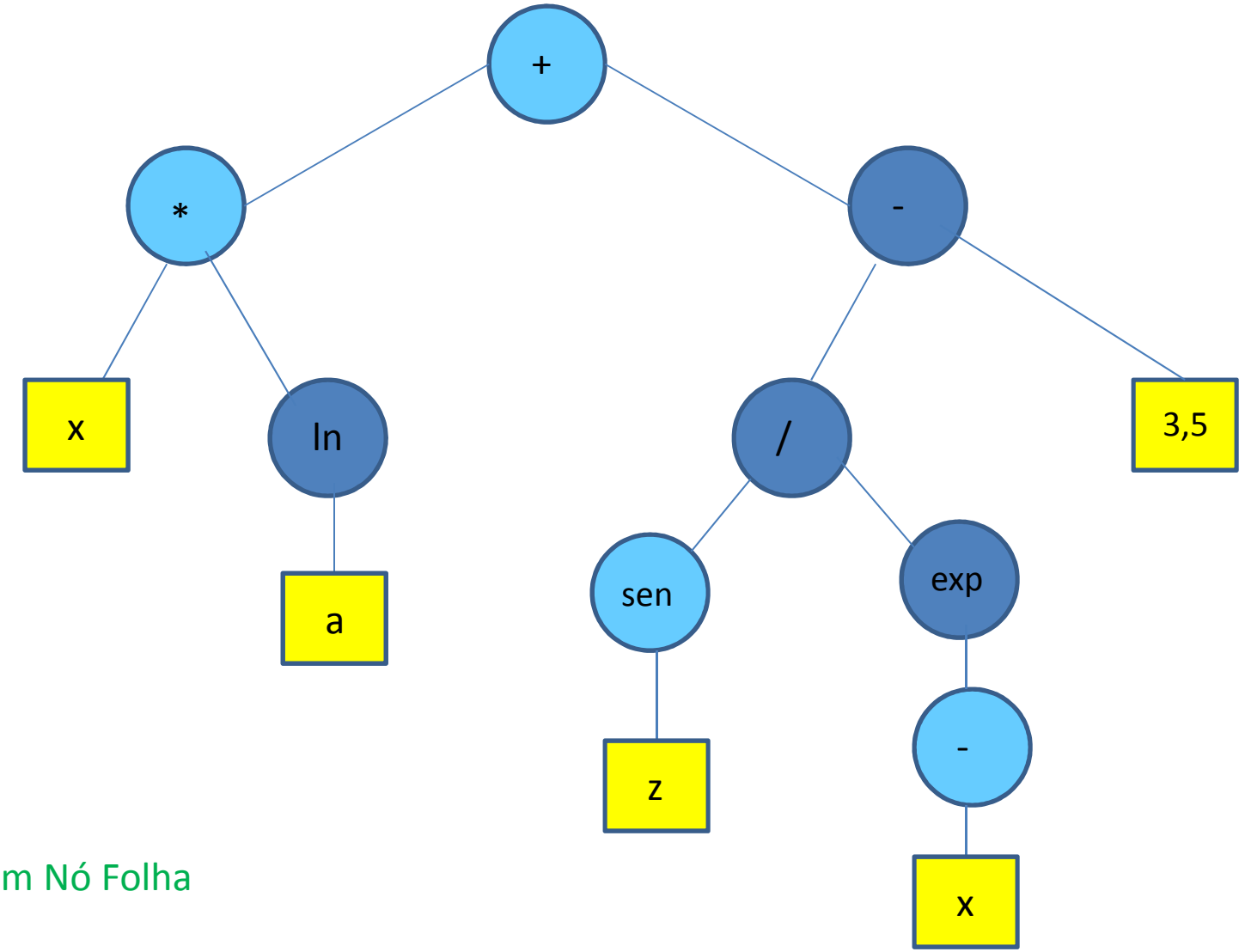
Mutação swap: troca dos argumentos

PG: Mutação em nó do CF



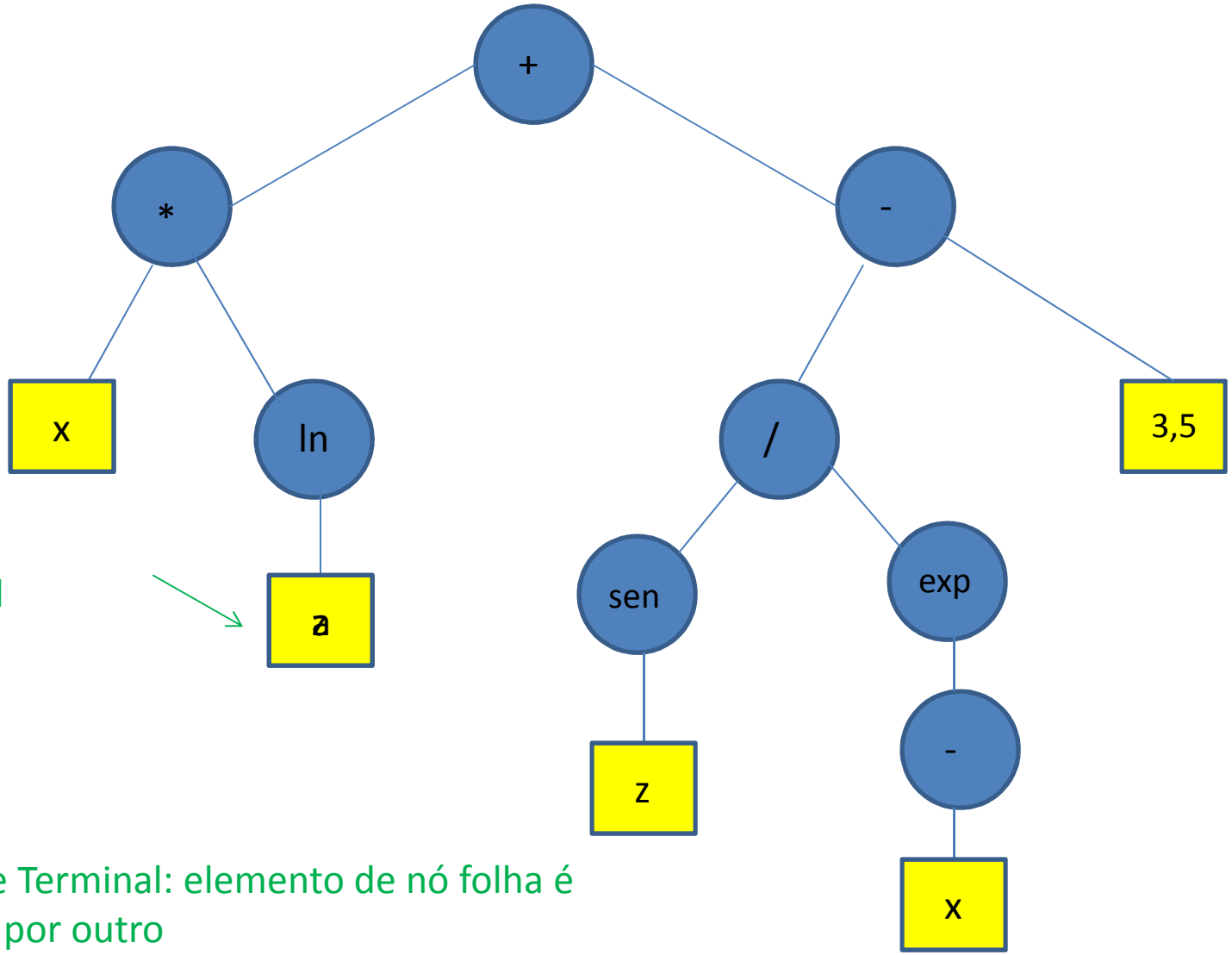
Mutação truncamento: nó de função substituído por nó terminal

PG: Mutação



Mutação em Nó Folha

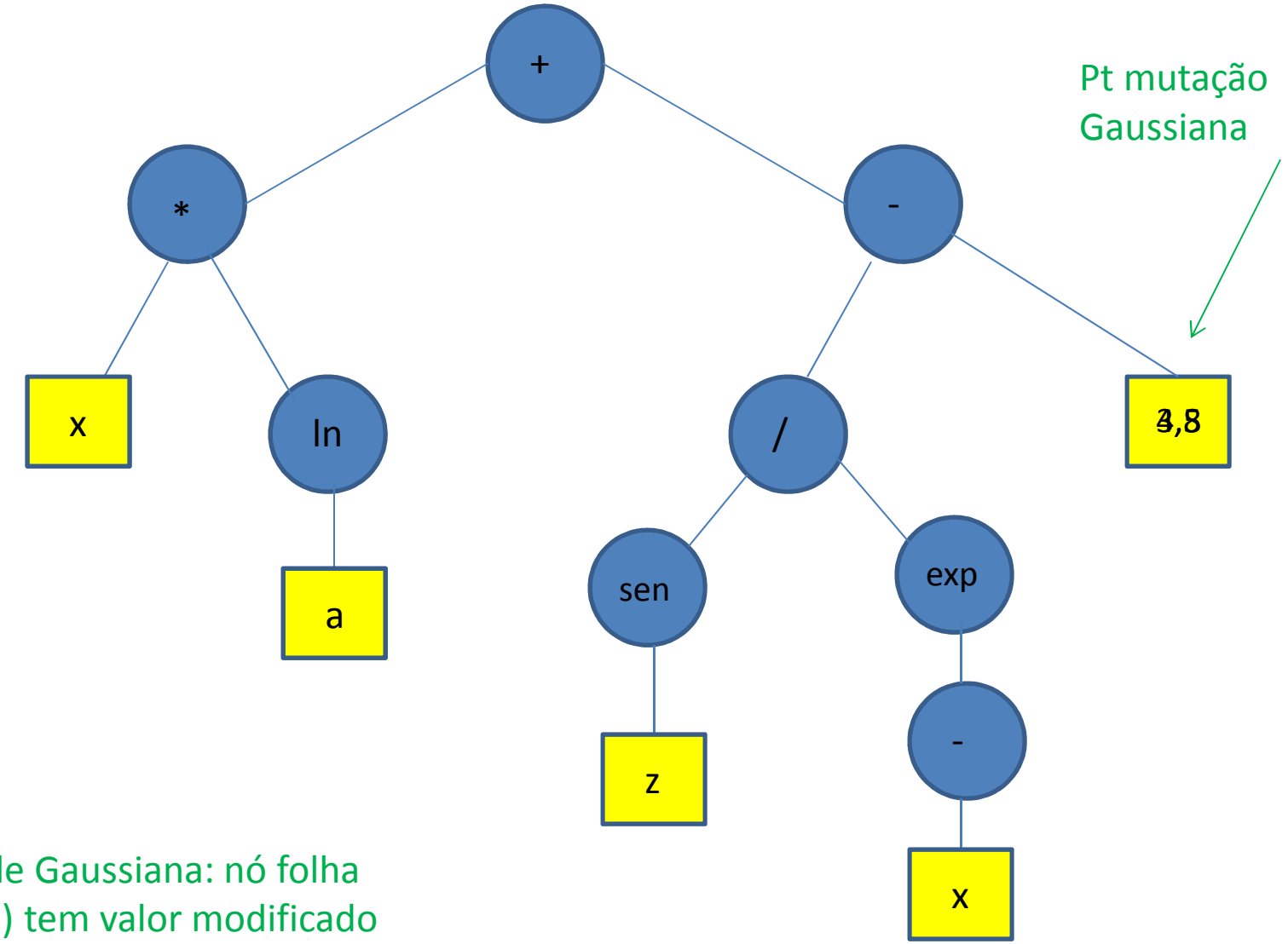
PG: Mutação em nó Folha



Pt mutação
de Terminal

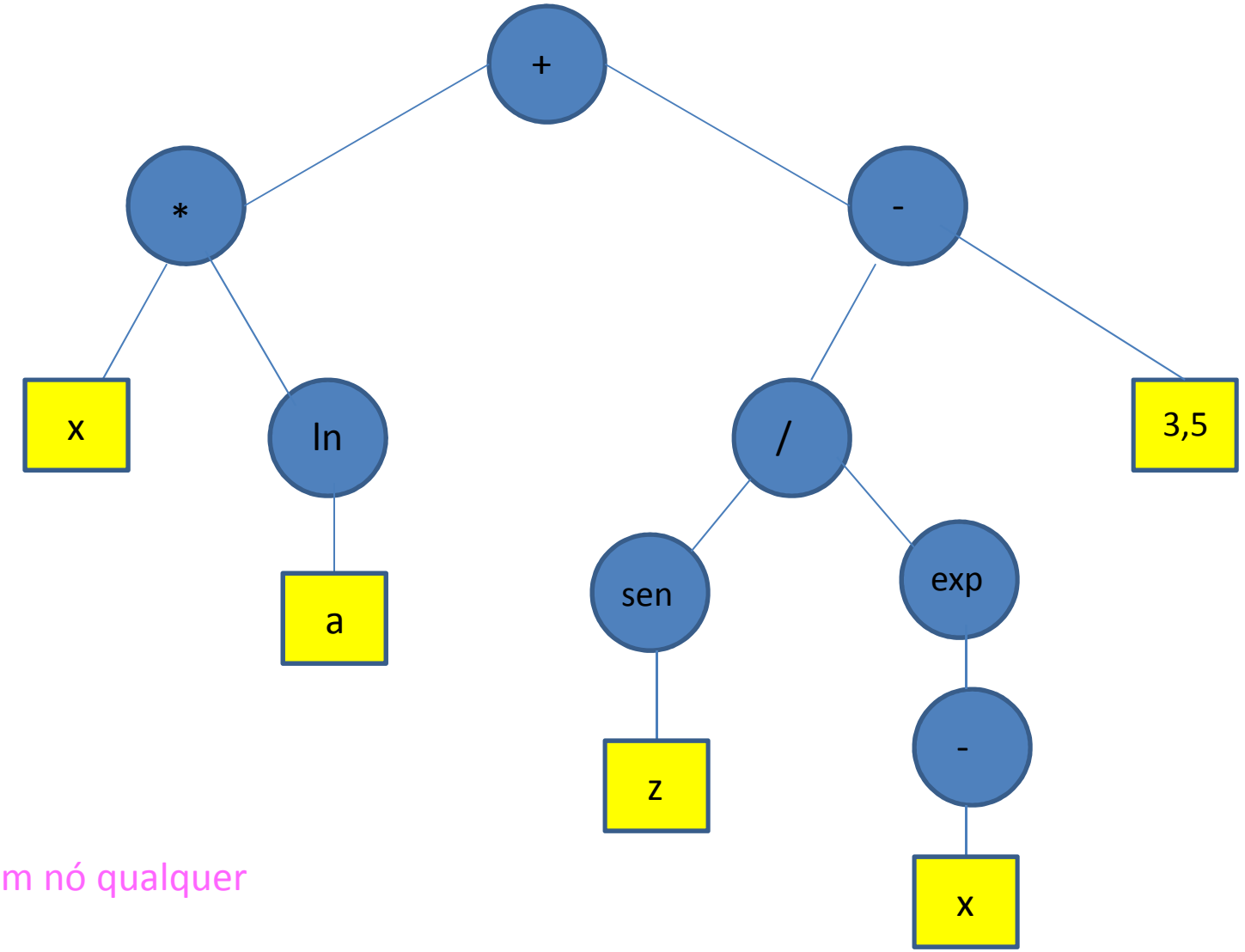
Mutação de Terminal: elemento de nó folha é substituído por outro

PG: Mutação em nó Folha



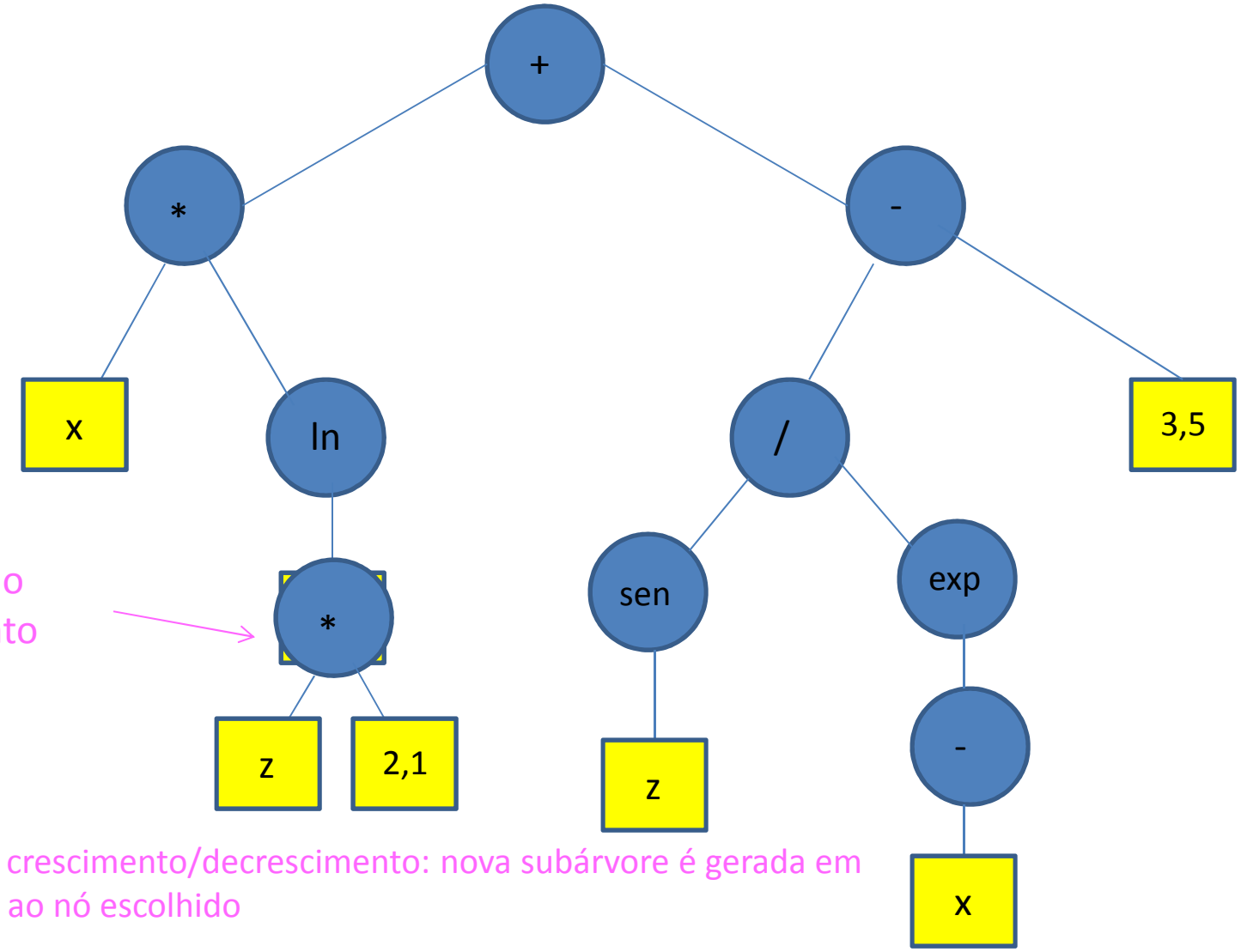
Mutação de Gaussiana: nó folha (constante) tem valor modificado

PG: Mutação



Mutação em nó qualquer

PG: Mutação em nó qualquer



Mutação de crescimento/decrescimento: nova subárvore é gerada em substituição ao nó escolhido