TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

Algoritmos Genéticos

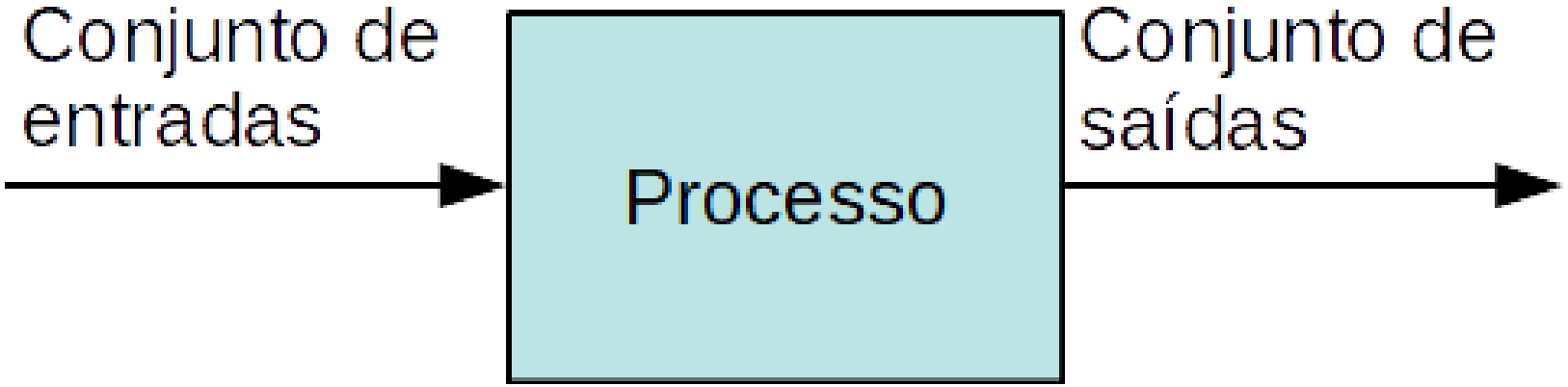
Professor Ricardo Kerschbaumer
ricardo.kerschbaumer@ifc.edu.br

http://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Algoritmos genéticos
Introdução

- Algoritmo Genético (AG) é uma técnica de otimização.
- Baseada nos princípios de Genética e Seleção Natural.
- É frequentemente usado para encontrar soluções ótimas ou quase ótimas para problemas difíceis.
- É frequentemente usado para resolver problemas de otimização, pesquisa e aprendizado automático.
- A otimização é o processo de fazer algo melhor.
- Otimização refere-se a encontrar os valores das entradas de forma a obter os "melhores" valores de saída.



Algoritmos genéticos
Introdução

- A definição de "melhor" varia, mas normalmente, refere-se a maximizar ou a minimizar uma ou mais **funções objetivo (fitness)**, variando os parâmetros de entrada.
- O conjunto de todas as soluções possíveis compõe o **espaço de busca**.
- Neste espaço de busca, é um ponto ou um conjunto de pontos que dá a solução ideal.
- O objetivo da otimização é encontrar esse ponto ou conjunto de pontos no espaço de busca.
- Algoritmos genéticos (AGs) são algoritmos de pesquisa baseados nos conceitos de seleção natural e genética.
- GAs são um subconjunto da Computação Evolutiva ou evolucionária.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Algoritmos genéticos
Introdução

- Os AGs foram desenvolvidos por John Holland, David E. Goldberg e seus alunos e colegas da Universidade de Michigan.
- Nos AGs, iniciamos com uma população de possíveis soluções para o problema dado.
- Essas soluções são submetidas a recombinação e mutação (como na genética natural), produzindo novas gerações de soluções.
- O processo é repetido por várias gerações.
- Cada indivíduo (ou solução) recebe um valor de aptidão (com base no valor da função objetiva).
- Os indivíduos mais aptos têm maior chance de se acasalar.
- Desta forma, mantemos "evoluindo" os melhores indivíduos ou soluções ao longo de gerações, até que possamos atingir um critério de parada.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Vantagens dos AGs

- Não requer nenhuma informação derivativa (que pode não estar disponível para muitos problemas do mundo real).
- É mais rápido e eficiente em comparação com os métodos tradicionais.
- Tem bons recursos para paralelização.
- Otimiza funções contínuas e discretas além de problemas multi-objetivo.
- Fornece uma lista de "boas" soluções e não apenas uma única solução.
- Sempre recebe uma resposta para o problema, que melhora ao longo do tempo.
- Útil quando o espaço de pesquisa é muito grande e há uma grande quantidade de parâmetros envolvidos.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Limitações dos AGs

- Os AGs não são adequados para todos os problemas, especialmente problemas que são simples e para os quais informações derivativas estão disponíveis.
- O valor de Fitness é calculado repetidamente, o que pode ser computacionalmente caro para alguns problemas.
- Sendo estocástico, **não há garantias** sobre a otimização ou a qualidade da solução.
- Se não for implementado corretamente, o AG pode não convergir para a solução ideal.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Solução de problemas difíceis

- Problemas NP-Difícil
- Mesmo os sistemas de computação mais poderosos, levam muito tempo.
- Em tal cenário, os AGs se revelam uma ferramenta eficiente.
- Fornecendo soluções quase ótimas úteis em um curto período de tempo.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

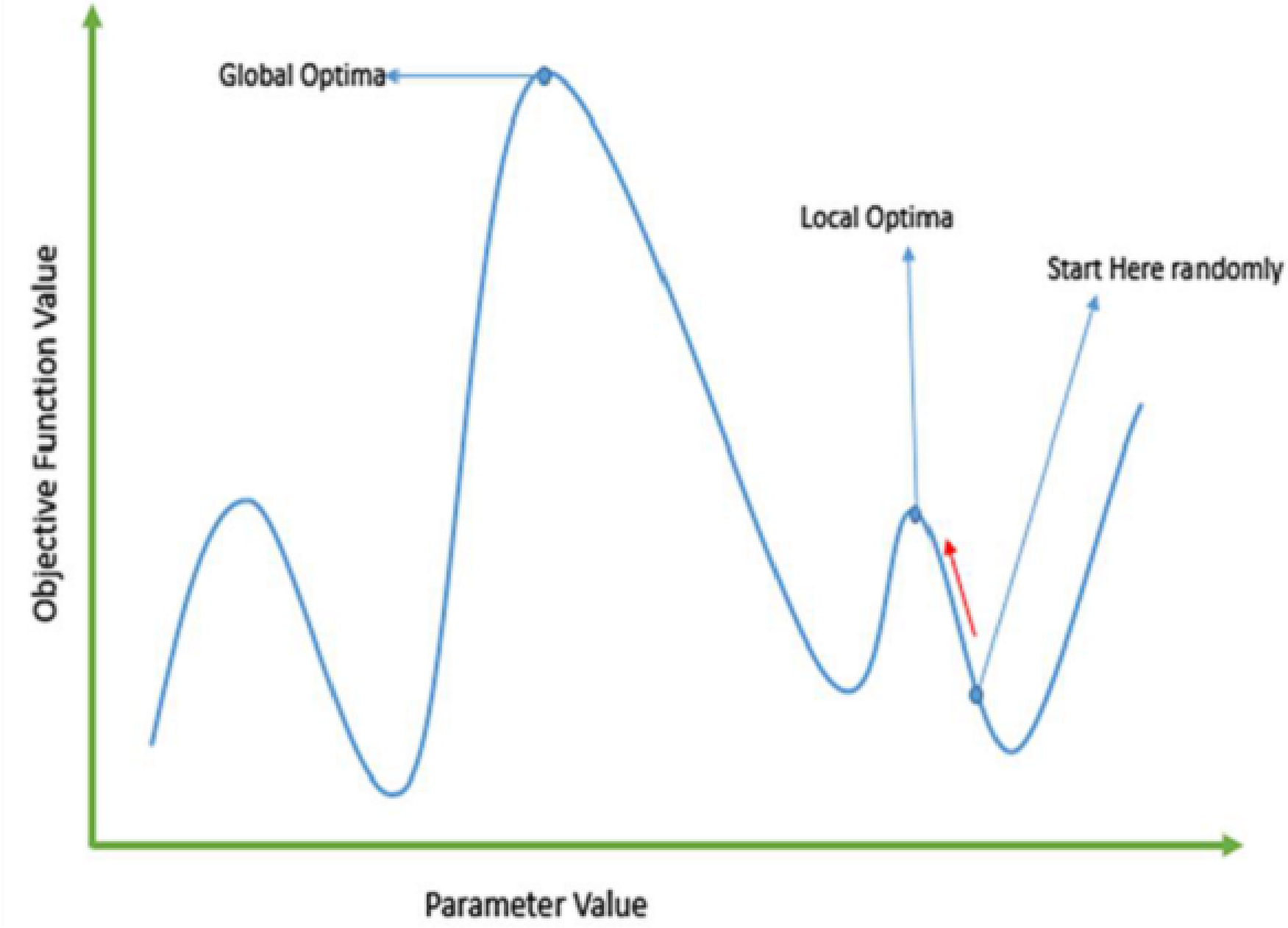
Falha em métodos baseados em gradientes

- Os métodos baseados em cálculos tradicionais funcionam começando em um ponto aleatório e movendo-se na direção do gradiente, até chegar ao ponto ótimo.

- Esta técnica funciona muito bem para funções objetivos de um único ponto como a função de custo na regressão linear.

- Nas situações do mundo real, temos funções feitas de muitos picos e vales, o que faz com que tais métodos falhem.

Falha em métodos baseados em gradientes

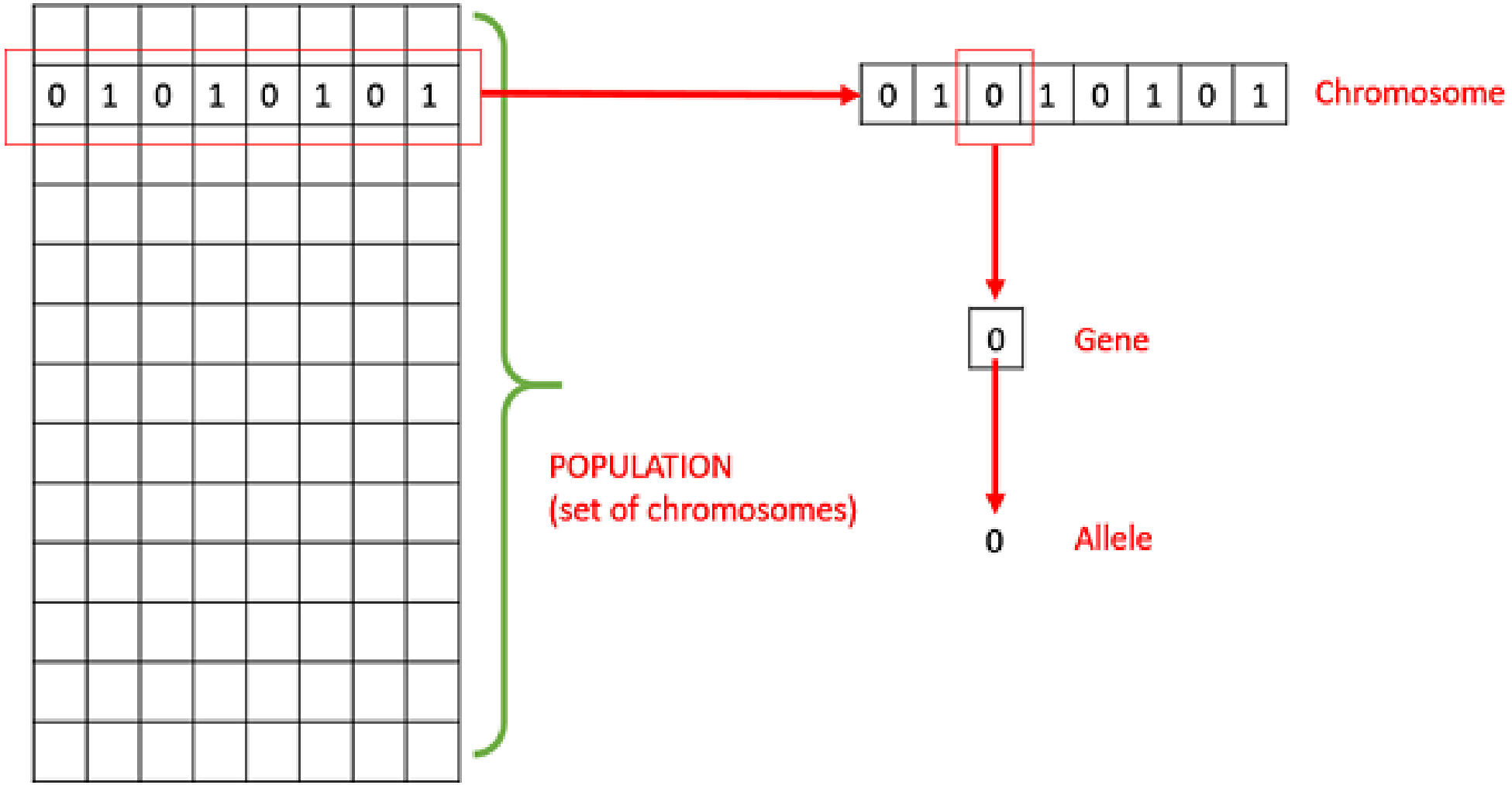


Fundamentos dos AGs

Terminologia básica dos AGs

- **População** - É um subconjunto de todas as possíveis soluções (codificadas) para o problema dado.
- **Cromossomos** - Um cromossomo é uma dessas soluções para o problema dado.
- **Gene** - Um gene é uma posição ou elemento de um cromossomo.
- **Alelo** - É o valor que um gene leva para um cromossomo em particular.

Fundamentos dos AGs



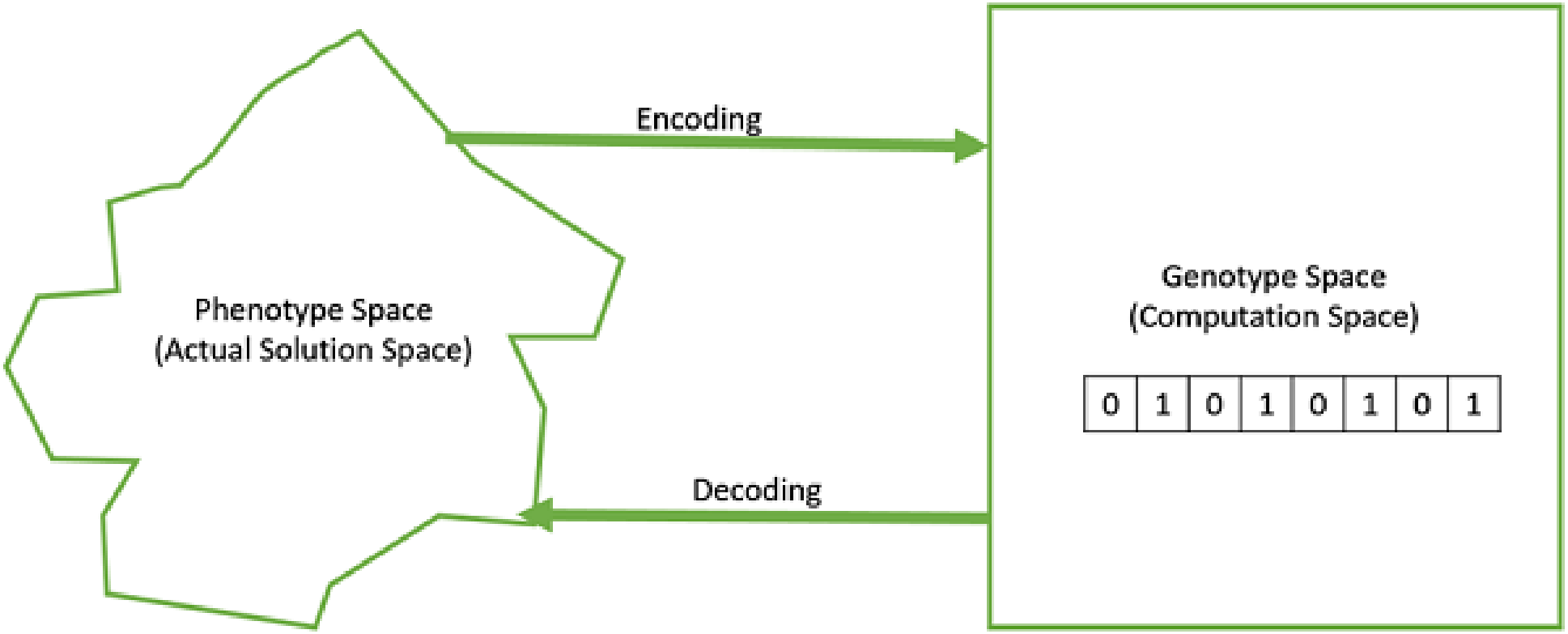
INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Fundamentos dos AGs

- **Genótipo** - é a população no espaço computacional.
- **Fenótipo** - é a população no espaço real da solução,
- **Decodificação** - é um processo de transformação de uma solução do genótipo para o espaço do fenótipo.
- **Codificação** - é um processo de transformação do fenótipo para o espaço de genótipos.

Por exemplo, considere o problema da mochila 0/1. O espaço do fenótipo consiste em soluções que apenas contêm os números dos itens. Já o genótipo pode ser representado como uma *string* binária de comprimento n onde um 0 na posição x indica que aquele x item não entra na mochila, enquanto um 1 representa o inverso. Este é um caso em que os espaços de genótipos e fenótipos são diferentes.

Fundamentos dos AGs



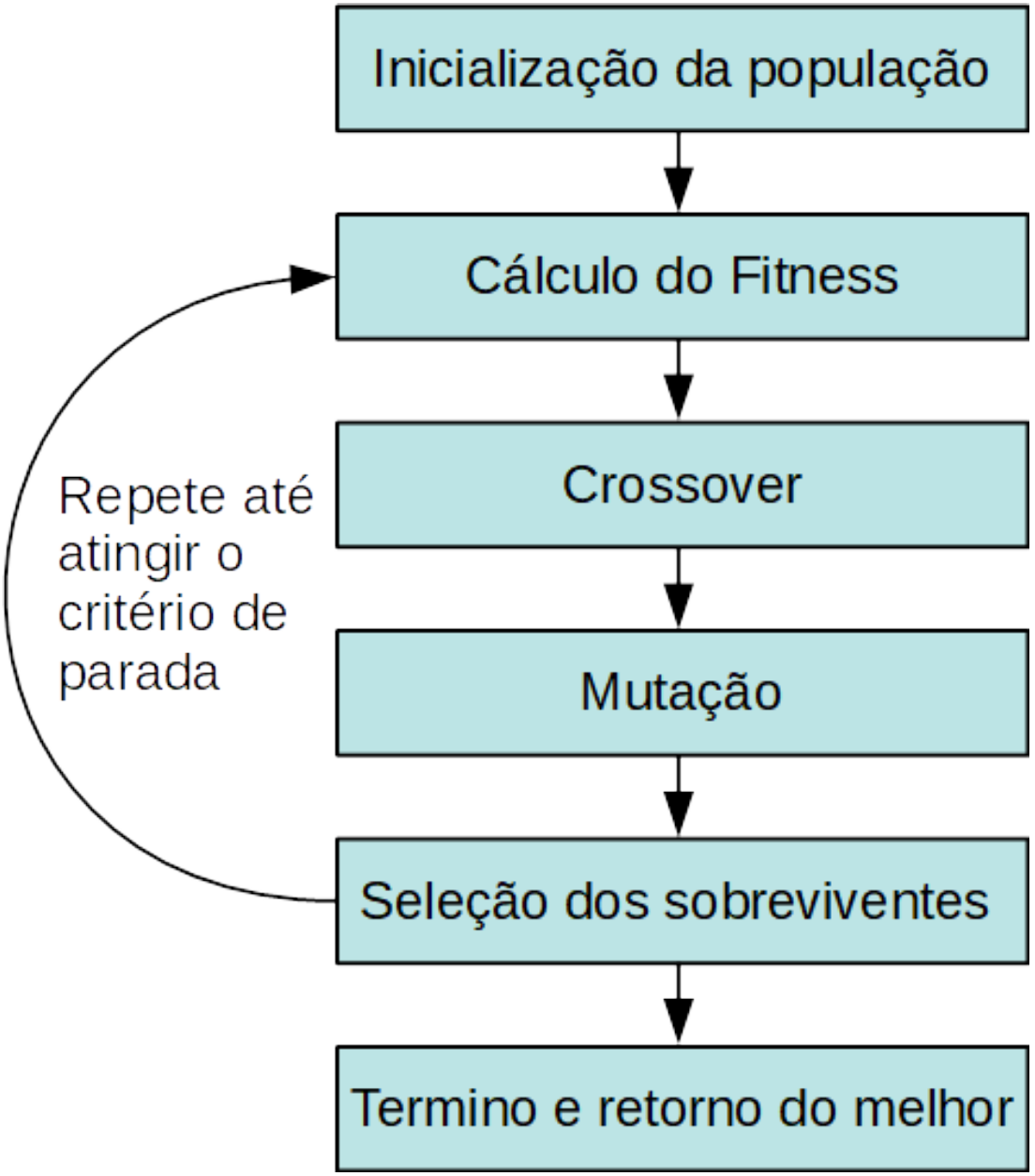
INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Fundamentos dos AGs

- **Função Objetivo ou Função de Fitness –** É uma função função que recebe as entradas e retorna a **aptidão** de um determinado indivíduo.

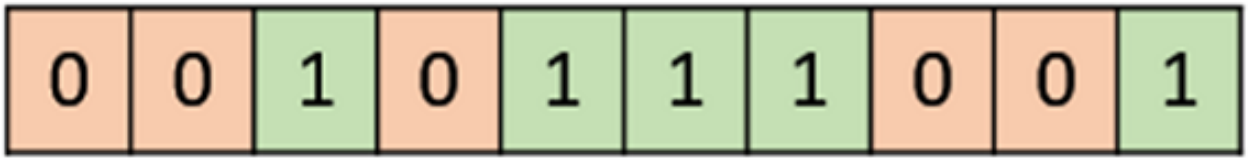
- **Operadores genéticos** – São operadores que alteram a composição genética da prole. Estes incluem cruzamento, mutação, seleção, etc.

A estrutura básica de um AG

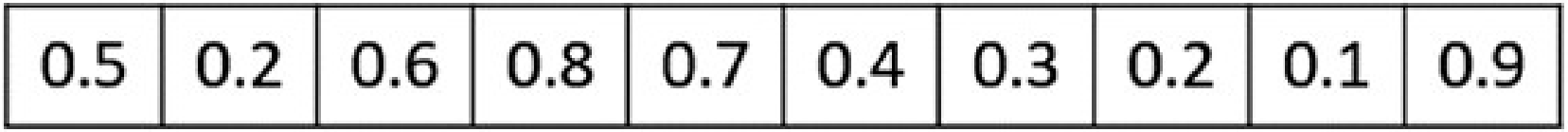


Representação do Genótipo

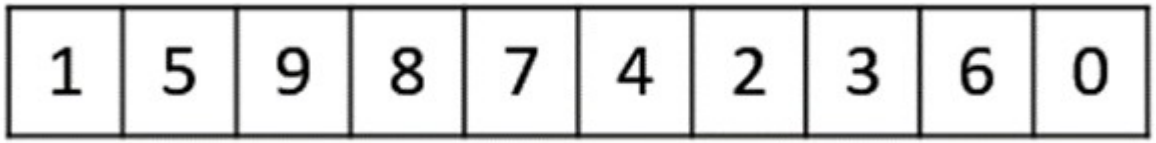
- Representação Binaria



- Representação por valores reais



- Representação por permutação



População

- A população é um subconjunto do conjunto de soluções
- A diversidade da população deve ser mantida, caso contrário poderá levar a uma convergência prematura.
- O tamanho da população não deve ser mantido muito grande, pois vai exigir muito poder computacional.
- Uma população muito pequena pode não ser suficiente para um bom conjunto de acasalamento.
- Portanto, um tamanho ideal de população precisa ser definido por tentativa e erro.
- Normalmente a população é definida como uma matriz de duas dimensões

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Inicialização da População

- **Inicialização aleatória** - Preencha a população inicial com soluções completamente aleatórias.
- **Inicialização heurística** - Preencha a população inicial usando uma heurística conhecida para o problema.
- Foi observado que a inicialização heurística, em alguns casos, afeta apenas a aptidão inicial da população, mas, no final, é a diversidade das soluções que levam à otimização.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Modelos de População

- **Estado estacionário**
  No AG de estado estacionário, geramos uma ou duas proles em cada iteração e elas substituem um ou dois indivíduos da população. Um AG de estado estacionário também é conhecido como AG Incremental.

- **Geracional**
  Em um modelo geracional, geramos 'n' indivíduos, onde n é o tamanho da população, e toda a população é substituída pela nova no final da iteração.

Função Objetivo

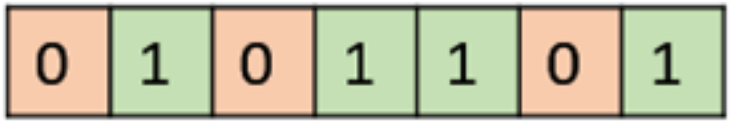
- A função objetivo é uma função que toma uma solução candidata como entrada e produz como saída um valor que representa o quão “boa” a solução é em relação ao problema em consideração.

- O cálculo do valor da aptidão é feito repetidamente em um AG e, portanto, deve ser rápido para evitar sobrecargas de processamento.

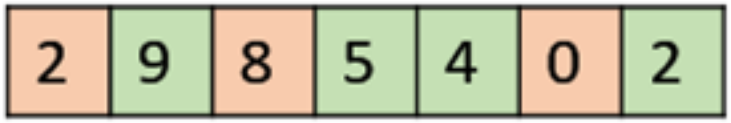
- Em alguns casos, calcular a função objetivo pode não ser possível devido a complexidades do problema. Nesses casos, faz-se uma aproximação as aptidão para atender necessidades do AG.

Função Objetivo

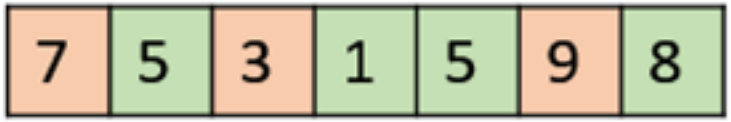
Item Number



Chromosome



Profit Values

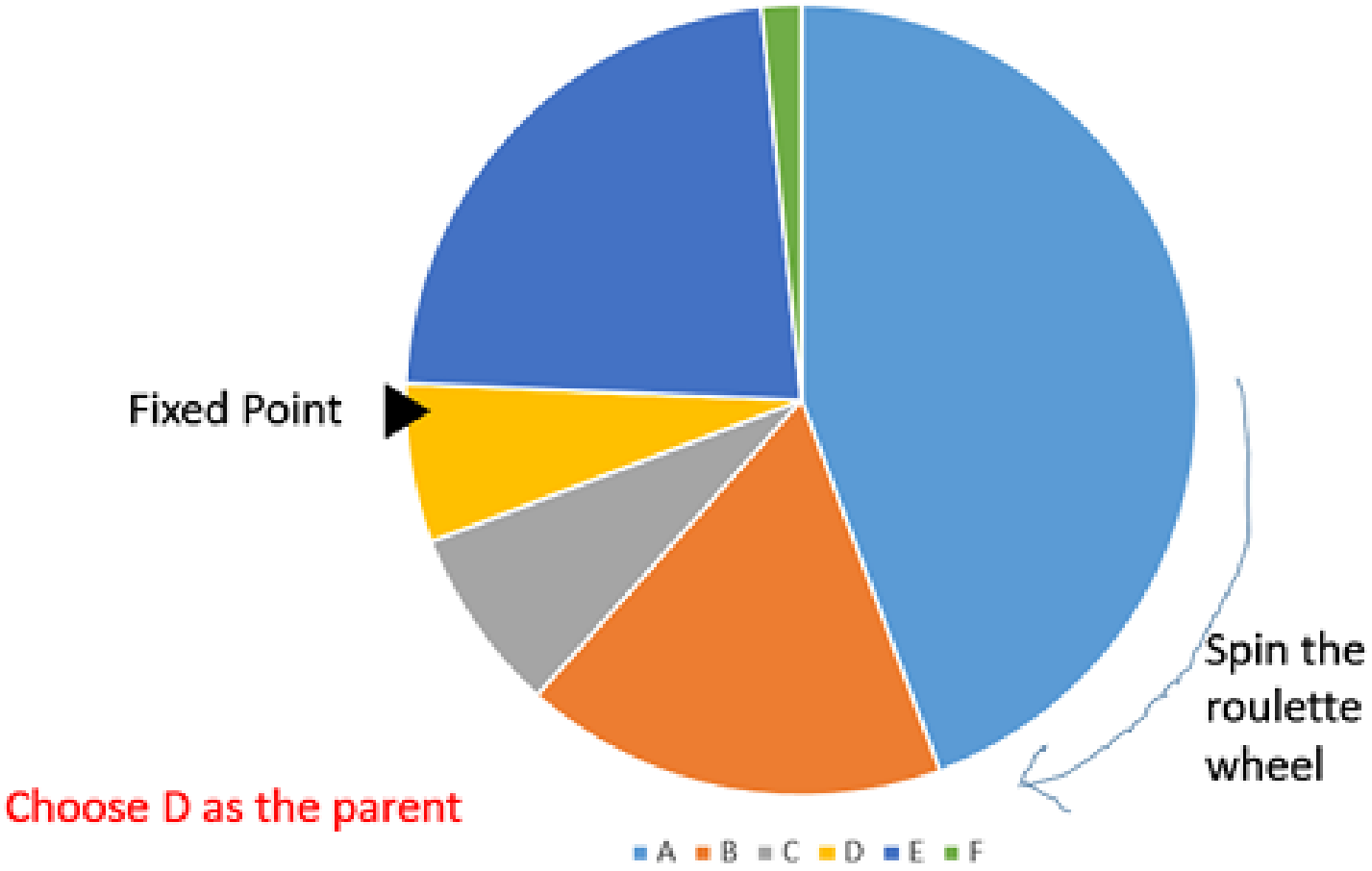


Weight Values

Knapsack capacity = 15
Total associated profit = 18
Last item not picked as it exceeds knapsack capacity

Seleção Proporcional ao Fitness

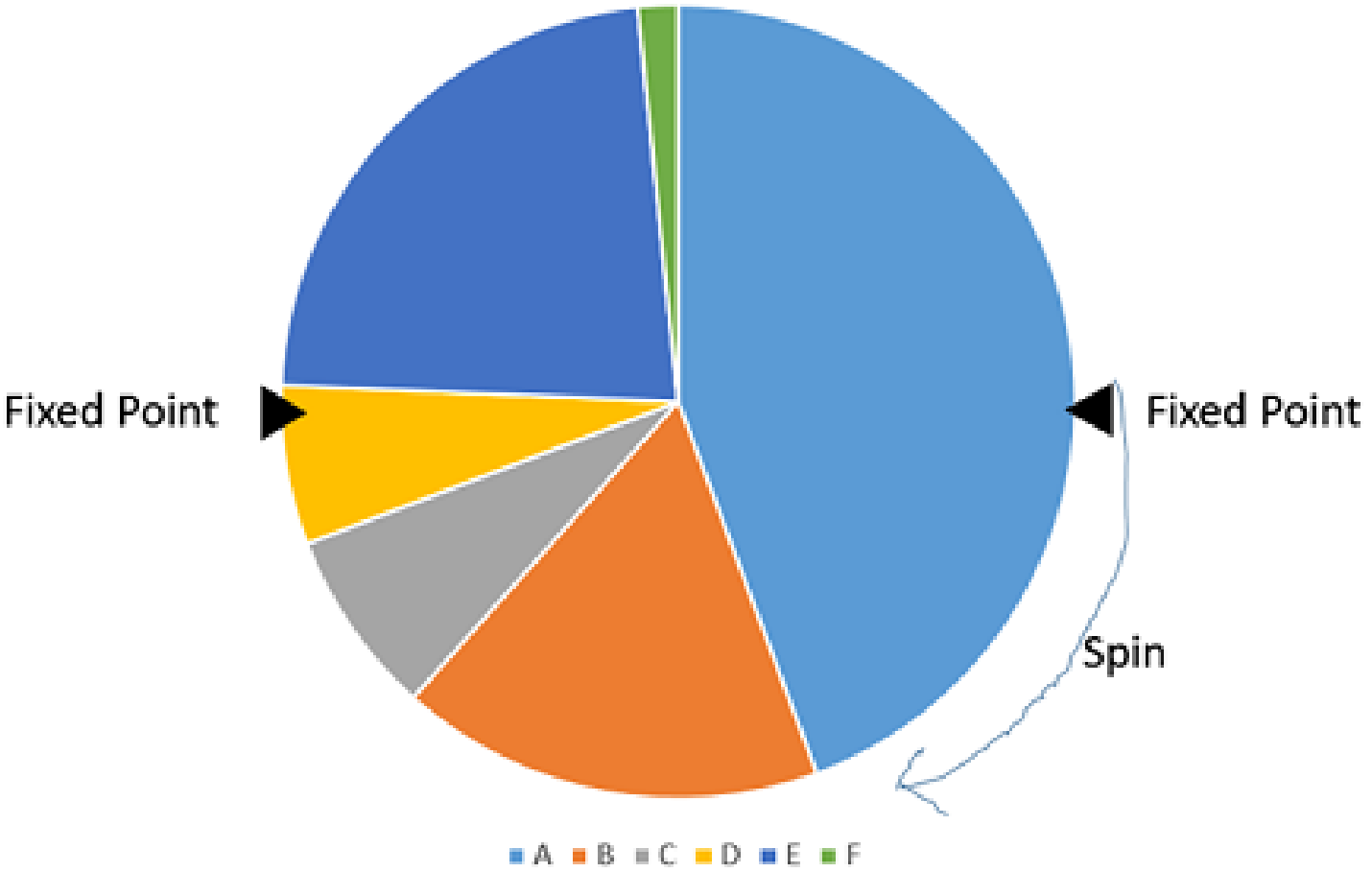
Seleção pelo método da roleta



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Seleção Proporcional ao Fitness

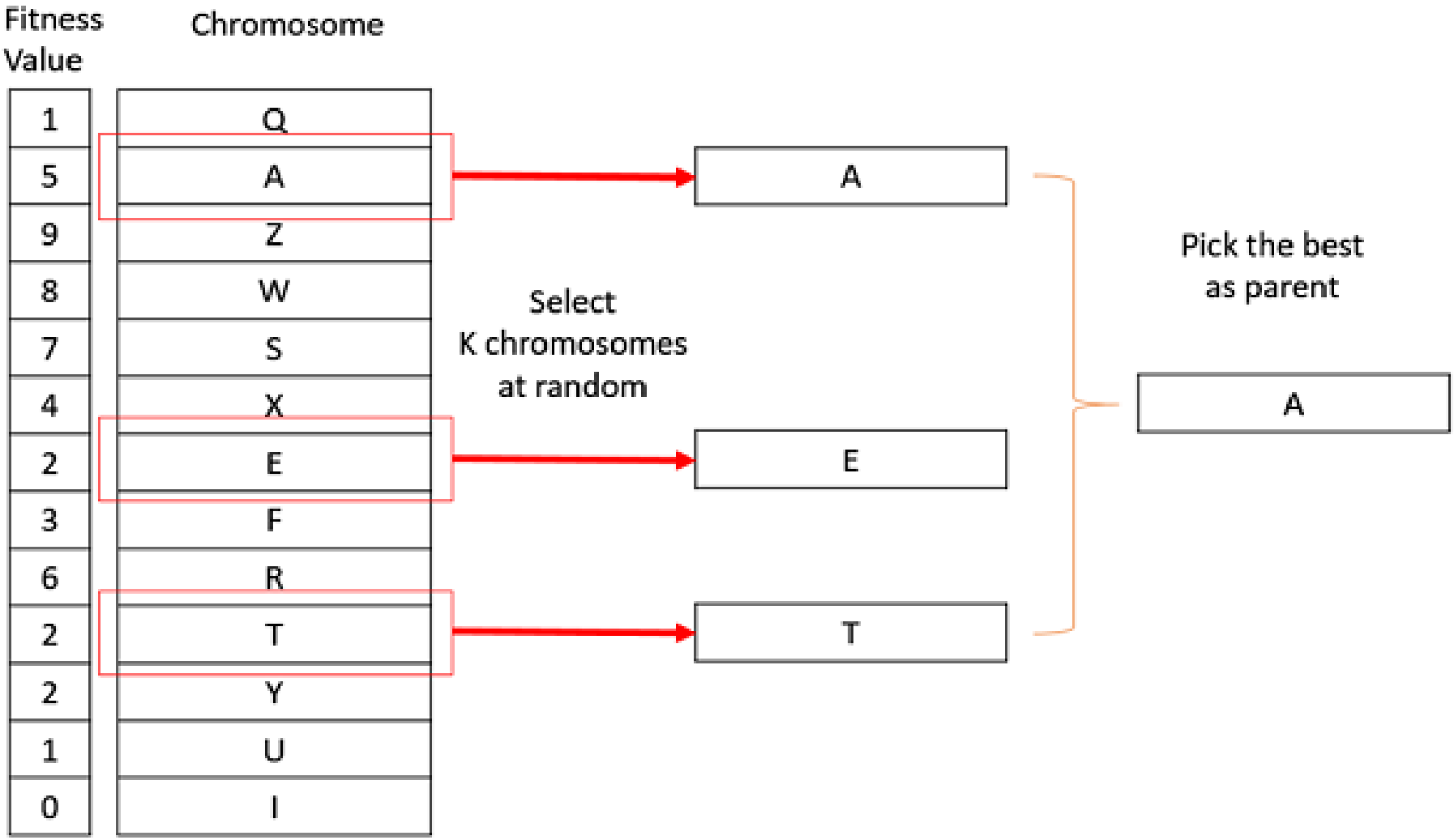
Amostragem estocástica



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Seleção Proporcional ao Fitness

Seleção por torneio



Seleção Proporcional ao Fitness

Seleção por ranking

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

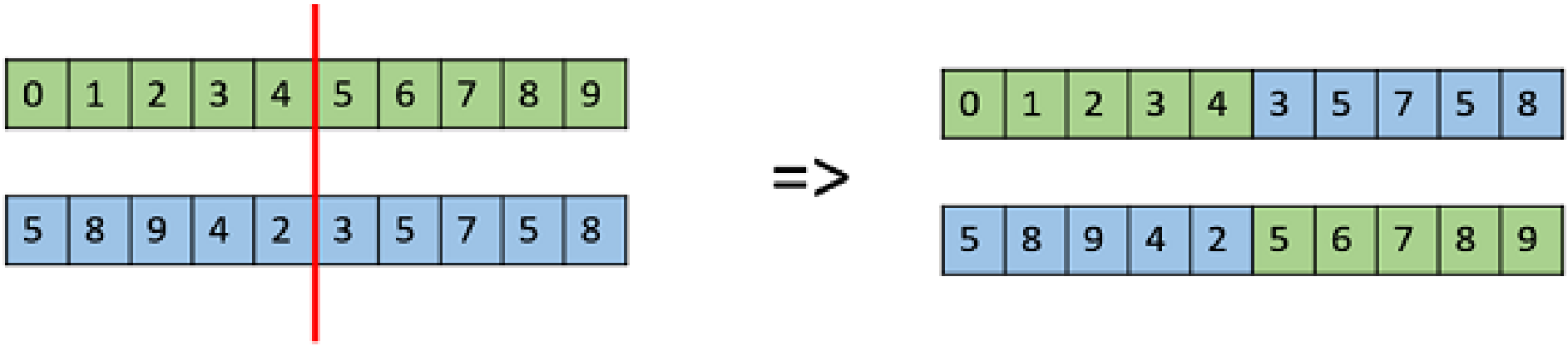
Seleção aleatória

Crossover

O operador de crossover é análogo à reprodução e ao crossover biológico. Trata-se de aproveitar o material genético dos pais para a geração da prole.

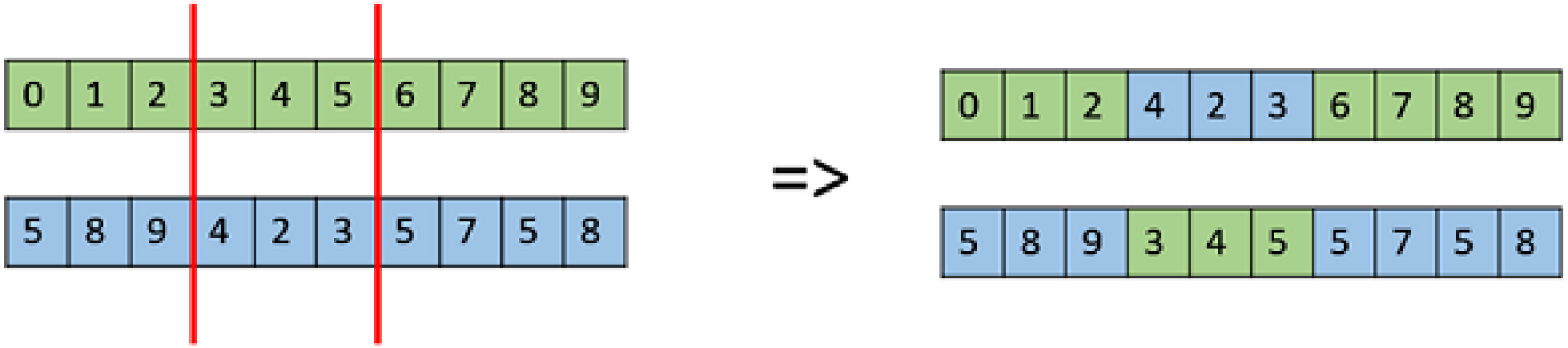
Operadores de crossover

- Crossover de 1 ponto

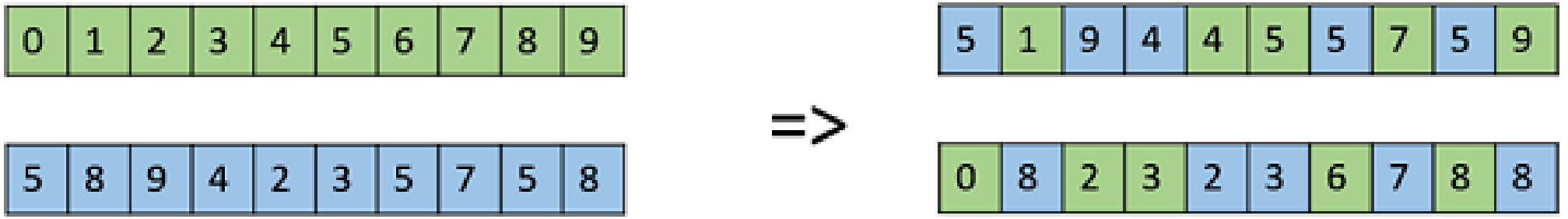


Crossover

- Crossover de múltiplos ponto



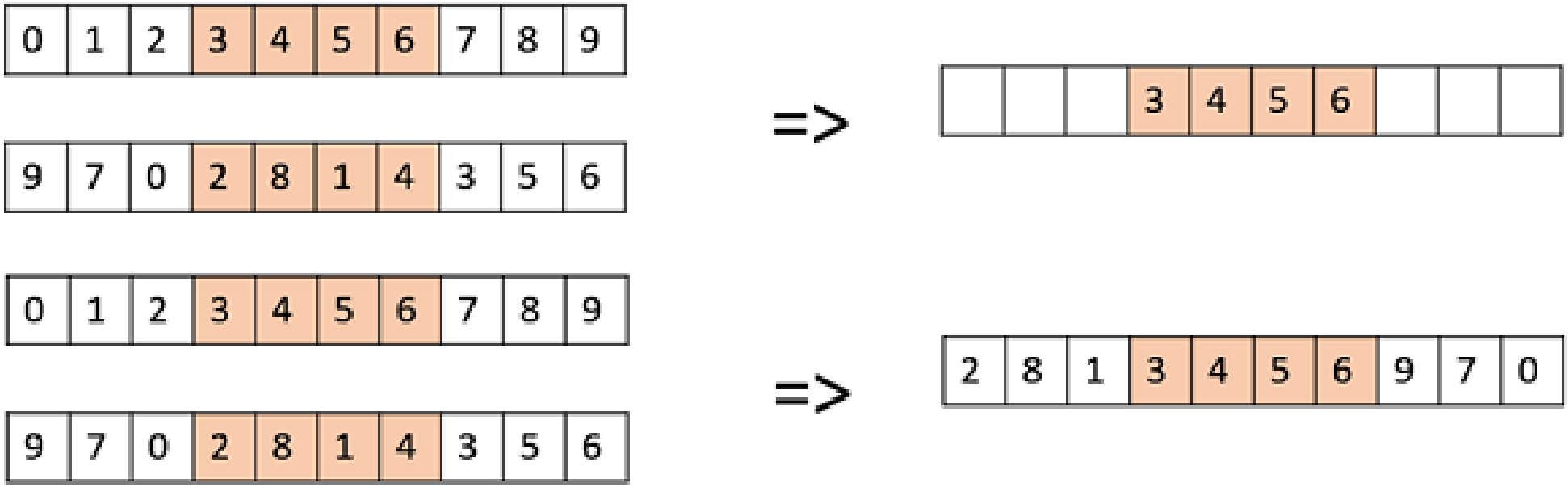
- Crossover uniforme



Crossover

- Recombinação aritmética

- Davis' Order Crossover



Repeat the same procedure to get the second child

Mutação

- Em termos simples, a mutação pode ser definida como um pequeno ajuste aleatório no cromossomo, para obter uma nova solução.

- A mutação é usada para manter e introduzir diversidade na população e é geralmente aplicado com baixa probabilidade. Se a probabilidade é muito alta, o AG é reduzido a uma pesquisa aleatória.

- Mutação é a parte do AG que está relacionada a "exploração" do espaço de busca. Foi observado que a mutação é essencial para a convergência do AG, enquanto o crossover não é.

Operadores de Mutação

Inversão de bit

Recomposição aleatória
Mutação por troca

Mutação por embaralhamento

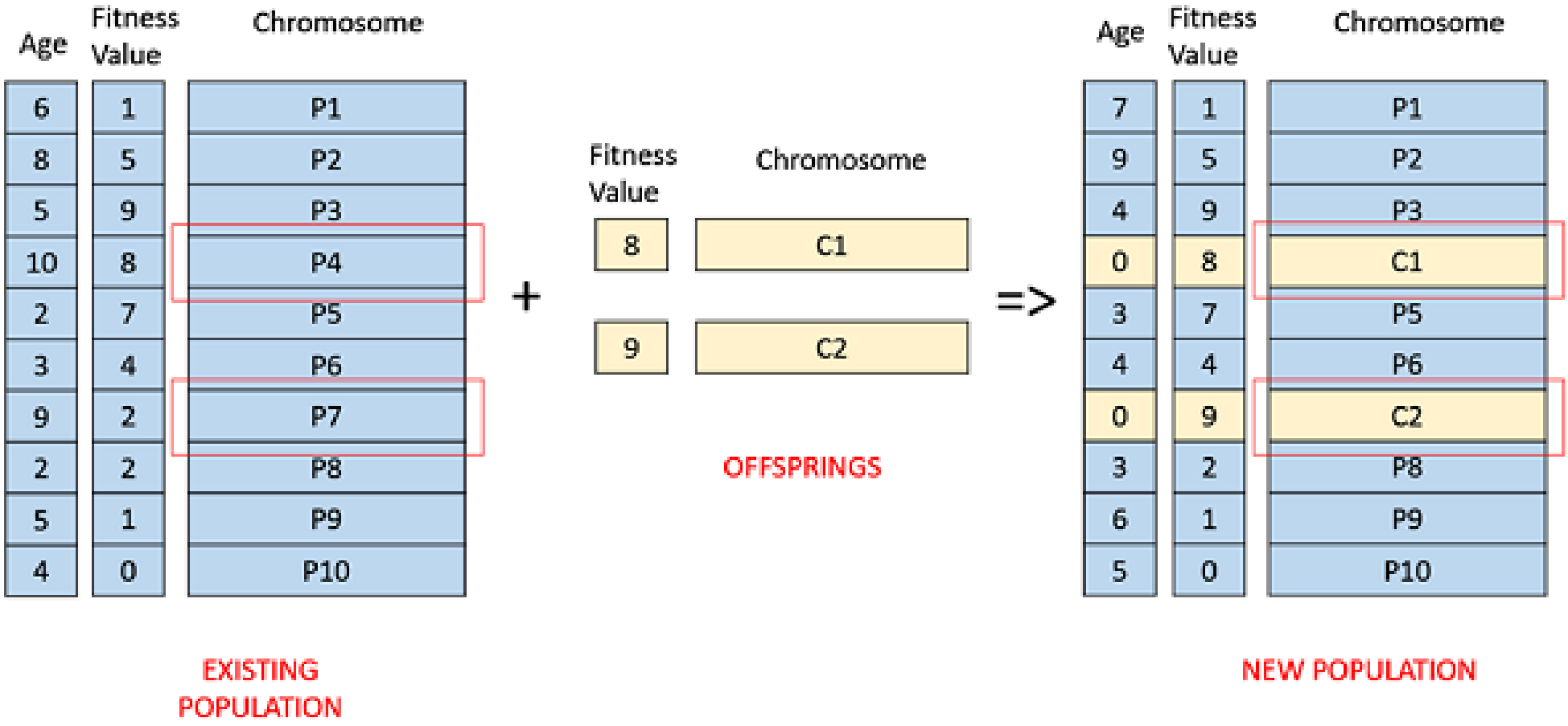
Mutação por inversão

Seleção dos sobreviventes

- A Política de Seleção de Sobreviventes determina quais indivíduos devem ser expulsos e quais devem ser mantidos na próxima geração.
- É crucial, pois deve garantir que os indivíduos mais aptos não sejam expulso da população, enquanto, ao mesmo tempo, a diversidade deve ser mantida.
- Alguns AGs empregam o **elitismo**. Em termos simples, significa que o atual membro mais apto da população é sempre propagado para a próxima geração. Portanto, sob nenhuma circunstância o membro mais apto da população atual pode ser substituído.
- A política mais fácil é expulsar membros aleatórios da população, mas essa abordagem frequentemente tem problemas de convergência, portanto, as estratégias a seguir são amplamente usadas.

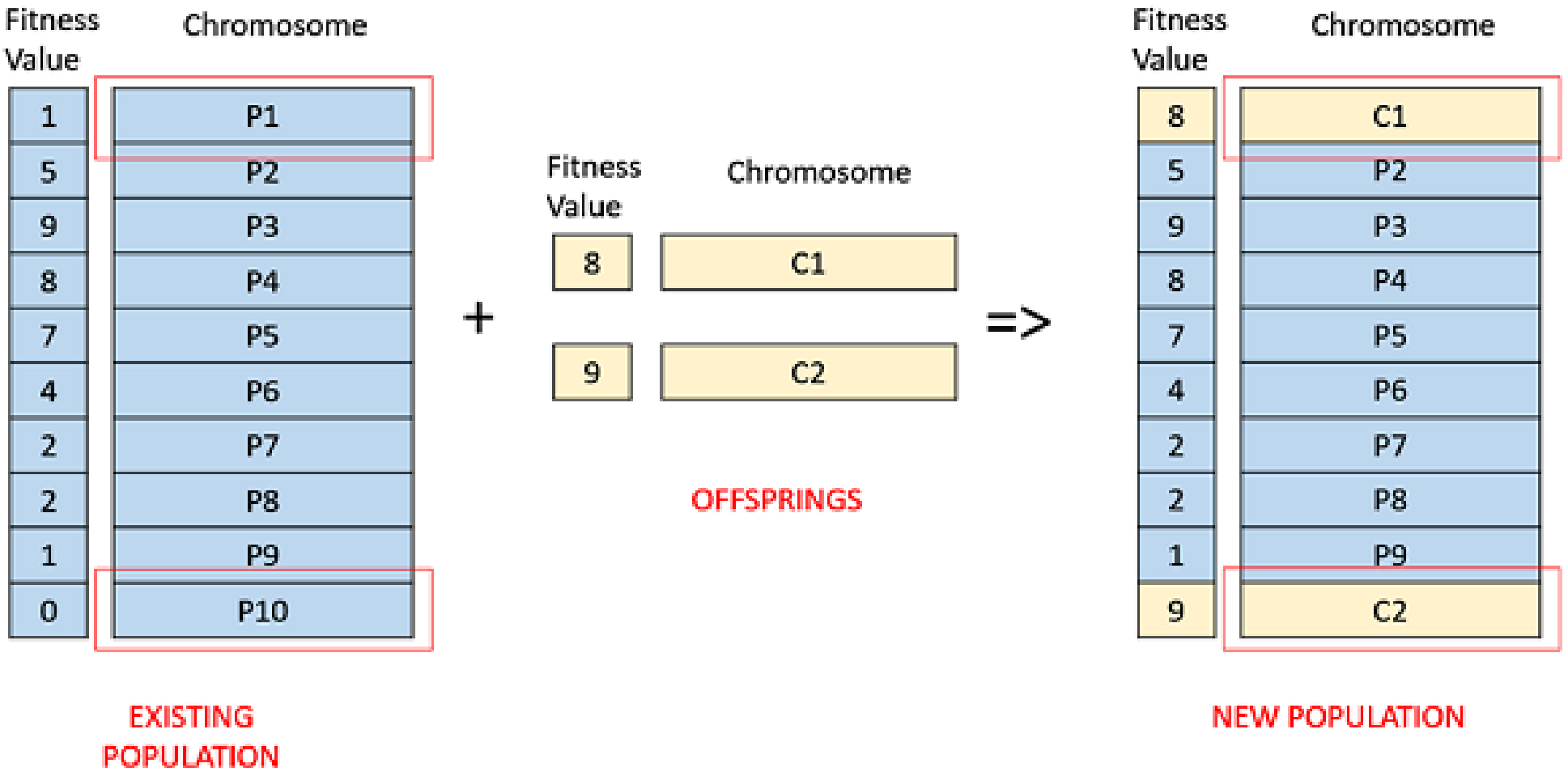
Seleção dos sobreviventes

Seleção baseada na idade



Seleção dos sobreviventes

Seleção baseada no fitness



Critérios de Parada

O critério de parada é importante para determinar quando a execução do AG terminará. Inicialmente o AG progride muito rapidamente, encontrando melhores soluções a cada poucas iterações, mas isso tende a saturar nos estágios posteriores. Assim, um critério de parada é necessário para definir o ponto final da execução.

Normalmente, mantemos um dos seguintes critérios de parada.

- Quando não houve melhora na população por X iterações.
- Quando alcançamos um número absoluto de gerações.
- Quando o valor da função objetivo atingiu um determinado valor pré-definido.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Problema da mochila

Tem-se 3 tipos de itens para colocar em uma mochila. A capacidade de mochila é de 20Kg. Os itens tem as seguintes características:

Tipo	Massa(Kg)	Valor	Disponibilidade
1	3	R$ 40,00	5 unidades
2	5	R$ 100,00	5 unidades
3	2	R$ 50,00	5 unidades

Deseja-se maximizar o valor transportado na mochila.

Fitness=40x(N1)+100x(N2)+50x(N3)

Restrições: Máximo de 20Kg e disponibilidade de cada item.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Problema da mochila

Definição dos Cromossomos:
Cada cromossomo é uma solução para o problema.

N tipo1	N tipo2	N tipo3

Probabilidades dos objetos

Total	P	P Acum
0	0,167	0,167
1	0,167	0,333
2	0,167	0,5
3	0,167	0,667
4	0,167	0,834
5	0,167	1

Problema da mochila

População inicial de 4 indivíduos, gerada aleatoriamente

				Restrição (20Kg)
1º Indivíduo	2	1	0	11 Kg
~~2º Indivíduo~~	~~3~~	~~2~~	~~3~~	~~25 Kg~~
3º Indivíduo	1	0	5	13 Kg
4º Indivíduo	2	1	1	13 Kg
5º Indivíduo	0	1	0	5 Kg

2º Indivíduo é inválido,
deve ser descartado

Problema da mochila

Cálculo do valor do Fitness

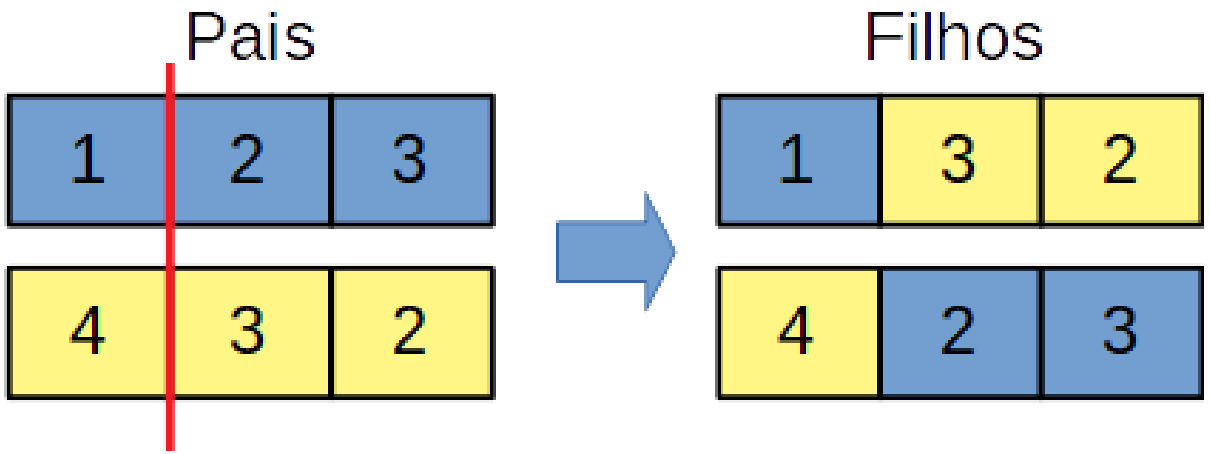
				Valor do Fitness
A	2	1	0	80+100+0=180
B	1	0	5	120+0+250=370
C	2	1	1	80+100+50=230
D	0	1	0	0+100+0=100

Indivíduo B tem o melhor Fitness (370)

Problema da mochila

Determinação dos sucessores

1º passo: Selecionar 2 indivíduos pelo método da roleta
2º passo: Gerar os filhos através do Crossover de 1 ponto



3º passo: Verificar se os filhos respeitam as restrições
4º passo: Repetir até completar a nova geração
5º passo: Substituir a geração atual pela nova geração

Problema da mochila

Mutação

1º passo: Verifica a possibilidade de um indivíduo sofrer mutação com base na taxa de mutação
2º passo: se o indivíduo vai sofrer mutação escolhe aleatoriamente em qual gene deve ser a mutação
3º passo: Encontrar um novo valor aleatório para este gene

4º passo: Verificar se o indivíduo gerado respeita as restrições
5º passo: Repetir até chegar ao final da população

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Problema da mochila

Calculo do fitness

Calcula o fitness de todos os indivíduos encontrando o valor total dos itens contidos na mochila

Elitismo

Se estiver usando elitismo escolhe um elemento da população aleatoriamente
Substitui este elemento pelo melhor elemento da população anterior

Apresentação dos resultados

A cada geração calcula e apresenta o melhor indivíduo

Critério de parada

Para após um número definido de gerações

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Gerando números aleatórios em linguagem C

A função que gera números aleatórios em C é a **rand()**.

Ela gera números entre 0 e **RAND_MAX**, onde esse **RAND_MAX** é um valor que pode variar de plataforma para plataforma.

Para utilizar o valor de **RAND_MAX**, temos que adicionar a função **stdlib.h**.

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Gerando números aleatórios em linguagem C

Exemplo com a função **rand()**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    int i;
    printf("intervalo da rand: [0,%d]\n", RAND_MAX);

    for(i=1 ; i <= 10 ; i++)
        printf("Numero %d: %d\n",i, rand());
}
```

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Gerando números aleatórios em linguagem C

Utilizando **rand()** com a **srand() : seed**

Para evitar que a sequencia de valores aleatório gerados pela função rand() seja sempre a mesma é necessário fornecer ao gerador de números aleatórios uma semente diferente a cada execução.

A função srand() serve para isso, com ela é possível fornecer uma semente para o gerador de números aleatórios.
Uma boa fonte de sementes que diferem a cada execução é o relógio, assim pode-se usar a seguinte linha de programa para este fim.

```
srand( (unsigned)time(NULL) ); //antes da chamada de rand()
```

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Gerando números aleatórios em linguagem C

Exemplo com as funções **rand()** e **srand()**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int i;
    printf("intervalo da rand: [0,%d]\n", RAND_MAX);
    srand( (unsigned)time(NULL) );

    for(i=1 ; i <= 10 ; i++)
        printf("Numero %d: %d\n",i, rand());
}
```

INSTITUTO FEDERAL
CATARINENSE
Câmpus Luzerna

Gerando números aleatórios em linguagem C

Outras faixar de valores diferente de 0 a RAND_MAX.

Para escolher a faixa de valores vamos usar operações matemáticas, principalmente o operador de módulo: %

Para fazer com que um número ‘x’ receba um valor:
Entre 0 e 9, fazemos: **x = rand() % 10**
Entre 1 e 10, fazemos: **x = 1 + (rand() % 10)**
Entre 0 e 100: **x = rand() % 101**

Para ter os valores decimais, dividimos por exemplo por 100:
x = x/100;