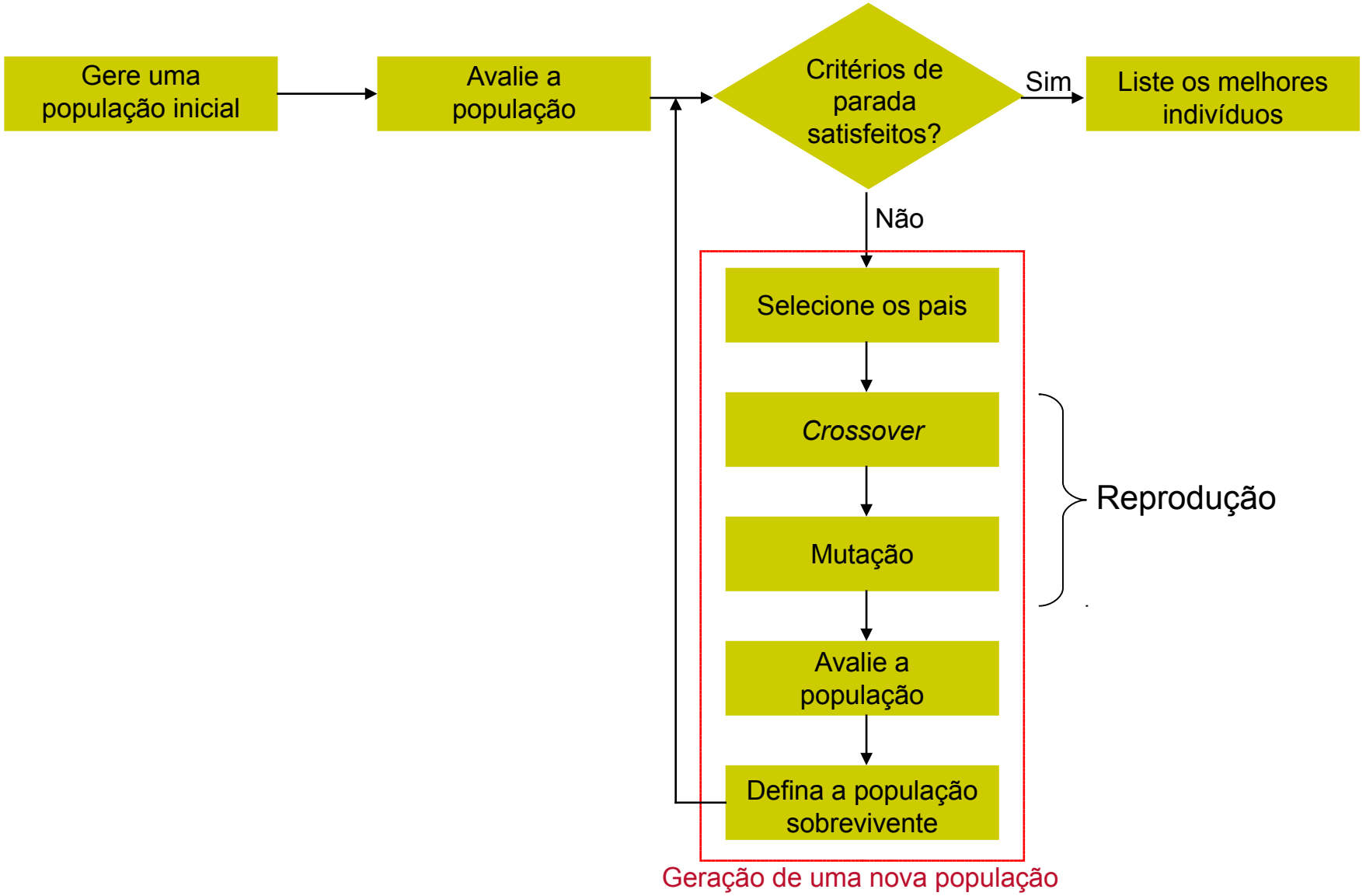
Aplicação de algoritmos genéticos

Problema da Mochila
(knapsack problem)

Algoritmos genéticos

- Passos inspirados no processo biológico de evolução
- Ideia de sobrevivência dos mais adaptados
- Soluções cada vez melhores, a partir da evolução das gerações anteriores, até que uma solução próxima do ótimo seja obtida
- Tentativa de melhorar o desempenho de outros métodos de IA
- O método produz novas gerações melhores com base na medida de uma função de *fitness*

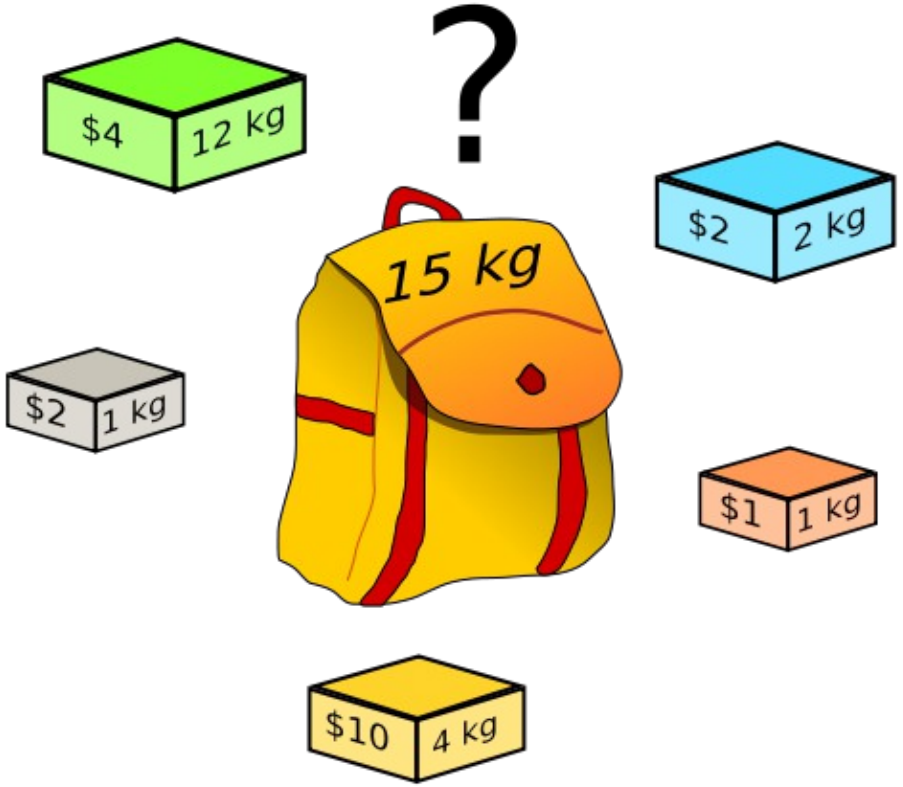
Estrutura de um AG



Algoritmo de um AG

1. **[Início]**
   - ✓ Codificação: representação do indivíduo
   - ✓ Geração de uma população aleatória de *n* cromossomos (soluções úteis)
2. **[Fitness]** Avaliação das aptidões de cada cromossomo
3. **[População]** Definição de uma nova população
4. **[Seleção]** Seleção dos melhores pais
5. **[Crossover]** Crossover dos pais para formar novos filhos
6. **[Mutação]** Com uma probabilidade de mutação
7. **[Aceitação]** Colocação de um novo filho em uma nova população
8. **[Substituição]** Utilização da nova população gerada para a próxima rodada do algoritmo
9. **[Teste]** Se a condição é satisfeita, então finaliza
10. **[Loop]** Retorno ao passo 2

Problema da mochila (knapsack)



- Vários itens que gostaria de levar em uma mochila
- Cada item com um peso e um benefício (valor)
- Há uma capacidade limite de peso
- Deve-se carregar itens com o máximo valor total sem superar o limite de peso

Problema da mochila (knapsack)

- Problema de otimização combinatória (NP-Completo)
- Estudado por mais de um século (desde ~1897)
- Resolvido por variados algoritmos
- Aplicações
  - Gravação de arquivos desperdiçando o mínimo espaço em cada mídia
  - Corte e empacotamento
  - Carregamento de veículos
  - Alocação de recursos em geral
    - Naves espaciais

Problema da mochila (knapsack)

Exemplo:

- Item: 1 2 3 4 5 6 7
- Benefício: 5 8 3 2 7 9 4
- Peso: 7 8 4 10 4 6 4
- Mochila suporta, no máximo, 22 quilos
- Adicione itens de modo a ter o máximo benefício

Problema da mochila (knapsack)

- Codificação: 0 = não existe, 1 = existe na mochila

  Cromossomo: 1010110

Item	1	2	3	4	5	6	7
Cromossomo	1	0	1	0	1	1	0
Existe?	sim	não	sim	não	sim	sim	não

  ➔ Itens pegos: 1, 3 , 5, 6

- Geração aleatória de população com *n* cromossomos:
  a) 0101010
  b) 1100100
  c) 0100011

Problema da mochila (knapsack)

- Fitness & seleção

a) 0101010: Benefício = 19, Peso = 24 ✘

Item	1	2	3	4	5	6	7
Cromossomo	0	1	0	1	0	1	0
Benefício	5	8	3	2	7	9	4
Peso	7	8	4	10	4	6	4

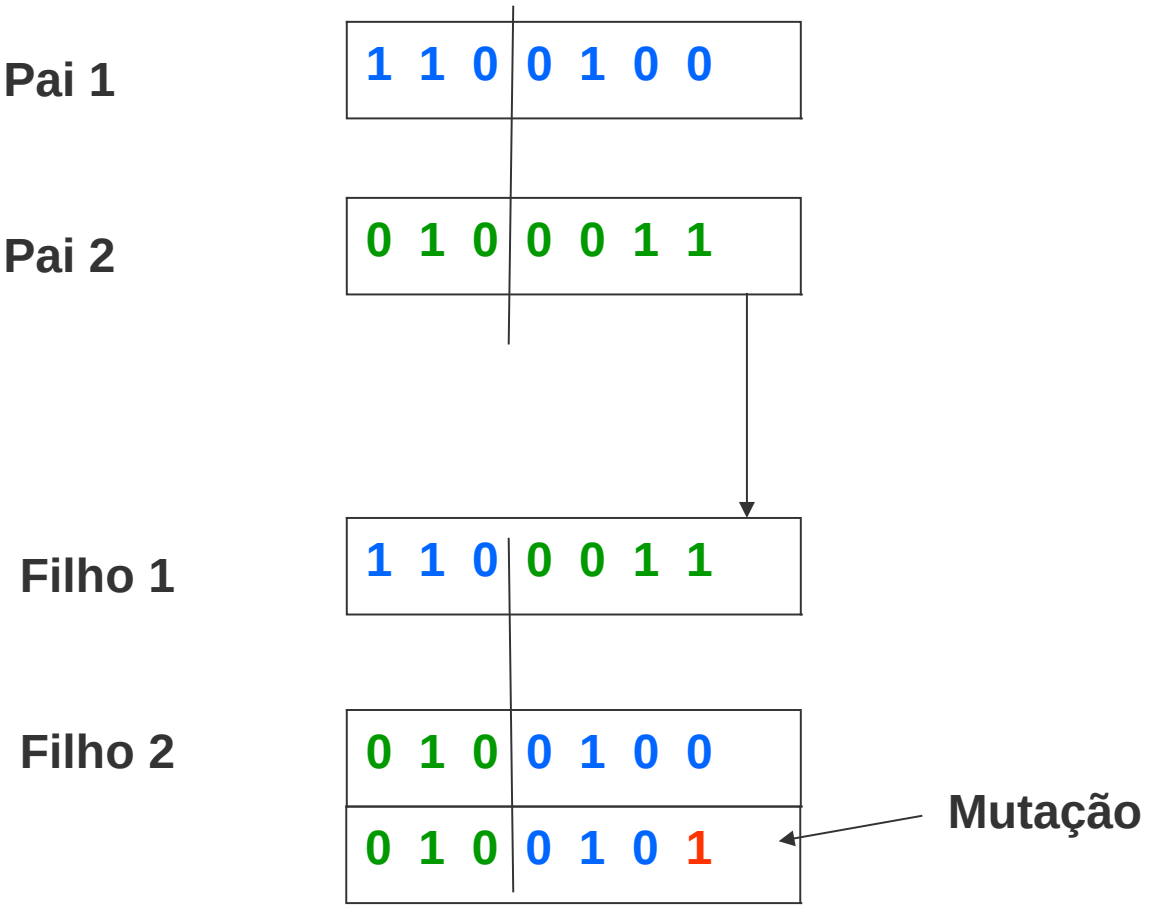
b) 1100100: Benefício = 20, Peso = 19 ✓

c) 0100011: Benefício = 21, Peso = 18 ✓

Serão selecionados os cromossomos b & c.

Problema da mochila (knapsack)

- Crossover & mutação



Problema da mochila (knapsack)

- Aceitação, substituição & teste
  - Definição de nova prole em uma nova população
  - Uso da nova população gerada para um próxima rodada do algoritmo
  - Se a condição final é satisfeita, então **finaliza**. Condições de finalização:
    - Número de gerações
    - Melhoramento da melhor solução
  - Caso contrário, retorne ao passo de **fitness**

Programação

- Modelo em Java do problema da mochila
  - https://www.cs.bgu.ac.il/~orlovm/teaching/assignments/intro-2009a-evo-knapsack.pdf

- Bibliotecas e Frameworks para Algoritmos Genéticos de modo generalista
- Pyevolve (Framework open-source para Algoritmos Genéticos e Programação Genética – Python) - http://pyevolve.sourceforge.net/
- GAUL (Biblioteca open-source para Algoritmos Genéticos e metaheurísticas - Linguagem C) - http://gaul.sourceforge.net/
- JGAP (Pacote open-source para Algoritmos Genéticos – Java) - http://jgap.sourceforge.net/
- JAGA (Pacote open-source para Algoritmos Genéticos e Programação Genética – Java) - http://www.jaga.org/
- GAlib (Framework open-source para Algoritmos Genéticos – C++) - http://lancet.mit.edu/ga/
- EvolveDotNet (Framework open-source para Algoritmos Genéticos – C#) - http://code.google.com/p/evolvedotnet/
- jMetal (Framework open-source para otimização multiobjetivo que contém Algoritmos Genéticos – Java) - http://jmetal.sourceforge.net/