Algoritmo Genético no Problema da Mochila

João Pedro Neves de Siqueira

Resumo

Este trabalho apresenta uma aplicação de Algoritmos Genéticos para a resolução do Problema da Mochila. Foram testadas diferentes configurações dos operadores genéticos e analisado o impacto dessas variações no desempenho do algoritmo.

Palavras-chave

Algoritmos Genéticos, Problema da Mochila, Otimização, Metaheurísticas.

Introdução

O objetivo deste trabalho foi usar um Algoritmo Genético (AG) para resolver o Problema da Mochila. A ideia foi testar como diferentes combinações de parâmetros (como tipo de crossover, taxa de mutação, inicialização da população e critério de parada) afetam o desempenho do AG.

Metodologia

Primeiro, fiz um código em Python que implementa o AG. Ele carrega as instâncias do problema com os dados dos itens peso e valor e a capacidade da mochila. Depois, executei o algoritmo com várias combinações diferentes de parâmetros:

- **Tipos de crossover:** 1 ponto, 2 pontos e uniforme
- **Taxas de mutação:** baixa (0.01), média (0.05) e alta (0.1)
- **Inicialização da população:** aleatória e heurística tentando preencher com itens válidos. A heurística usada consiste em selecionar itens aleatoriamente e adicioná-los enquanto a capacidade da mochila não for excedida, garantindo soluções viáveis desde a inicialização
- **Critério de parada:** número fixo de gerações (100) ou parada por estagnação

A população inicial tinha 50 indivíduos, e foi feito com elitismo mantendo os 2 melhores a cada geração.

Resultados e Discussão

Foi rodado o algoritmo em 10 instâncias diferentes do problema, variando todas as combinações possíveis entre os operadores citados. Os resultados foram salvos em CSV e analisados para entender qual configuração traz melhores resultados.

Tabela Comparativa dos Melhores Resultados

Instância	Crossover	Mutação	Inicialização	Parada	Valor	Tempo (s)
1	1 ponto	Baixa	Aleatória	Fixo	973	0.012
2	1 ponto	Baixa	Aleatória	Fixo	837	0.012
3	1 ponto	Baixa	Aleatória	Convergência	723	0.002
4	1 ponto	Baixa	Aleatória	Convergência	1963	0.002
5	1 ponto	Baixa	Aleatória	Fixo	1354	0.015
6	1 ponto	Baixa	Aleatória	Fixo	1915	0.016
7	1 ponto	Baixa	Aleatória	Convergência	2380	0.006
8	1 ponto	Média	Aleatória	Convergência	2869	0.006
9	1 ponto	Média	Heurística	Fixo	3164	0.022
10	1 ponto	Média	Aleatória	Fixo	3461	0.025

Tabela 1: Resumo dos melhores resultados para cada instância

Melhor Configuração

A melhor solução encontrada entre todas as combinações testadas foi na instância 10, com valor total de 3461. Quatro configurações distintas atingiram esse valor máximo, todas com critério de parada fixo e inicialização heurística ou aleatória. A configuração mais eficiente, considerando valor e tempo, foi:

- **Crossover:** 1 ponto
- **Mutação:** média
- **Inicialização:** heurística
- **Critério de parada:** número fixo de gerações
- **Tempo de execução:** 0.0983 segundos

Analise de todos Resultados Relevantes

Olhando para os melhores resultados, dá pra perceber algumas tendências bem interessantes:

- O **crossover de 1 ponto** foi, de longe, o mais eficiente. Ele apareceu em todas as melhores soluções, mostrando que, às vezes, o simples funciona muito bem.
- A **mutação baixa** funcionou melhor nas instâncias menores, ajudando a manter boas soluções sem bagunçar muito a população. Já em instâncias maiores, a **mutação média** trouxe mais diversidade, evitando que o algoritmo ficasse preso em soluções ruins.
- A **inicialização aleatória** foi dominante, indicando que, mesmo sem uma estratégia inicial mais sofisticada, o algoritmo conseguiu evoluir bem as soluções. Mas vale notar que, na instância 9, a **heurística** conseguiu um desempenho um pouco melhor.
- Quanto ao **critério de parada**, quando usei a **parada por convergência**, consegui economizar tempo nas instâncias menores e médias, sem perder qualidade. Para as maiores, manter um número fixo de gerações acabou sendo mais seguro.

- No geral, o tempo de execução foi super rápido a maioria das execuções levou menos de 0,02 segundos. Isso reforça que o algoritmo foi bem eficiente, mesmo testando muitas combinações.

Análise por Tipo de Crossover

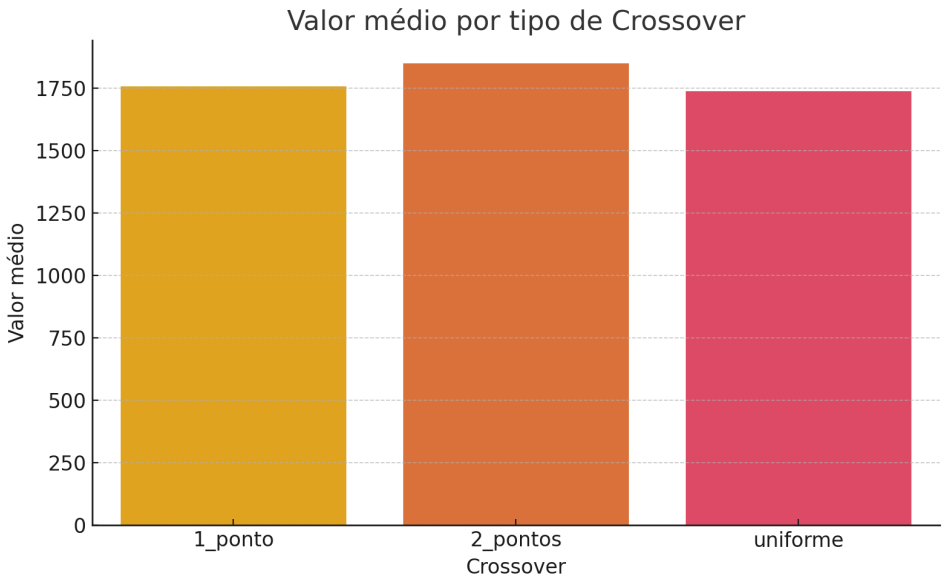

Figura 1: Valor médio obtido por tipo de crossover

Análise por Taxa de Mutação

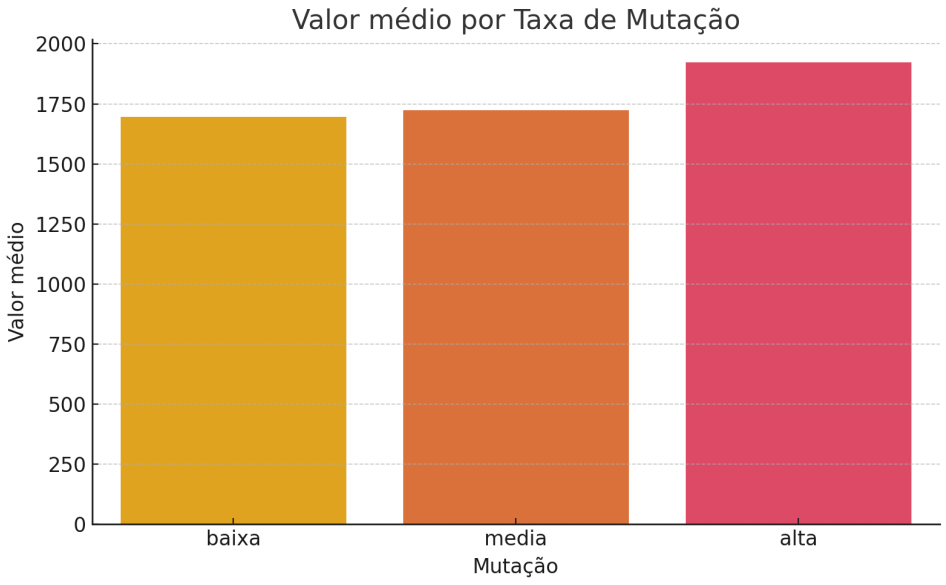

Figura 2: Valor médio obtido por taxa de mutação

Análise da Inicialização

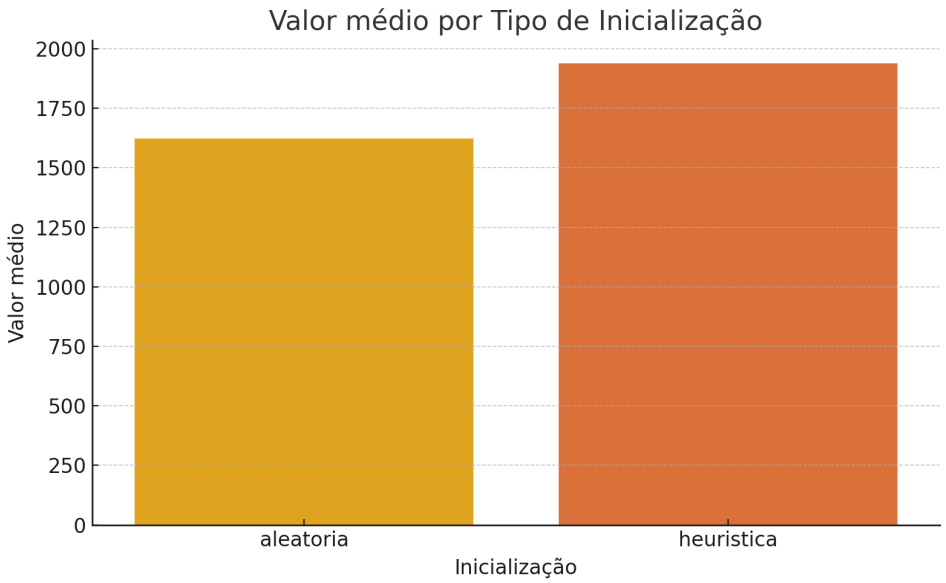

Figura 3: Valor médio obtido por tipo de inicialização

Análise do Critério de Parada

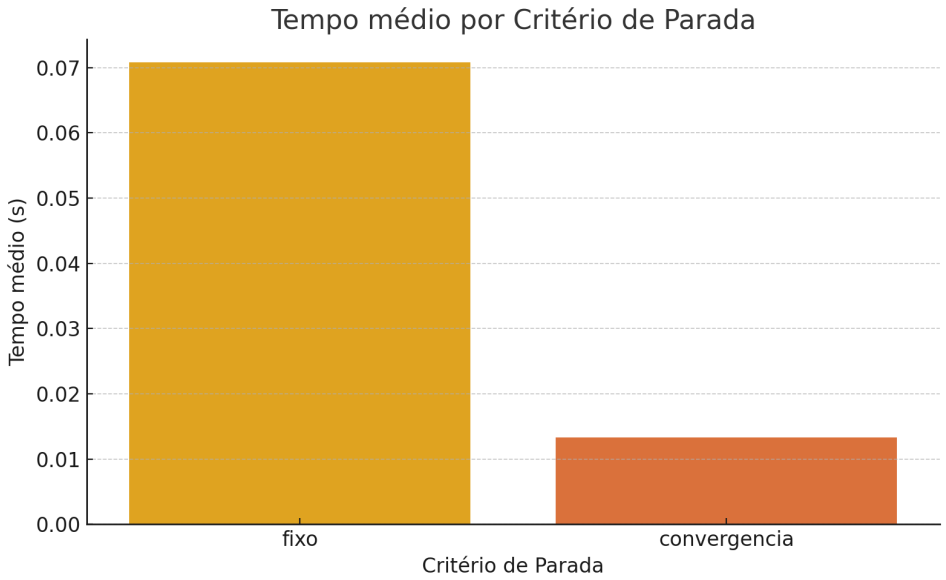

Figura 4: Tempo médio de execução por critério de parada

Conclusão

Este trabalho me permitiu explorar como diferentes configurações de operadores genéticos influenciam o desempenho de um Algoritmo Genético aplicado ao Problema da Mochila. Os testes mostraram que o uso de crossover de 1 ponto, mutação média, inicialização heurística

e critério de parada fixo levou às melhores soluções em termos de valor obtido e tempo de execução.

Além disso, deu para perceber que escolhas simples, como a inicialização heurística e o uso de elitismo, ajudaram bastante na eficiência do algoritmo mostrando que nem sempre é preciso algo muito complexo para ter bons resultados.

Para trabalhos futuros, seria interessante experimentar outros métodos de seleção e até formas adaptativas de mutação. Também vale explorar combinações com algoritmos exatos, o que pode trazer soluções ainda melhores, principalmente em instâncias maiores ou mais difíceis.