Implementação de Algoritmos Genéticos para Resolver o Problema da Mochila

Davi Emannoel Lopes de Souza

Universidade Tuiuti do Paraná
Curitiba, PR
DAVI.SOUZA@utp.edu.br

Abstract

Este relatório aborda a aplicação de algoritmos genéticos (AGs) ao problema da mochila, com enfoque na comparação de diferentes configurações e impacto das variações nos operadores genéticos. O objetivo é identificar a melhor abordagem em termos de tempo de execução e qualidade da solução. São apresentados resultados que mostram a eficiência de diferentes combinações de parâmetros e estratégias de inicialização. O relatório detalha implementações robustas e discute avanços potenciais para problemas maiores. Este relatório foi estruturado com o suporte de ferramentas avançadas, incluindo inteligência artificial, para garantir clareza e precisão na apresentação dos resultados.

1 Introdução

O problema da mochila é um clássico da otimização combinatória e pode ser enunciado como: dado um conjunto de itens, cada um com peso e valor, determinar quais itens incluir em uma mochila de capacidade limitada para maximizar o valor total. Este problema é formalizado matematicamente como:

$$\max \sum_{i=1}^{n} v_i x_i \quad \text{sujeito a} \quad \sum_{i=1}^{n} w_i x_i \leq C, \quad x_i \in \{0,1\}, \tag{1}$$

onde v_i e w_i são, respectivamente, o valor e o peso do item i, C é a capacidade da mochila, e x_i indica se o item foi selecionado.

Resolver este problema é relevante em áreas como logística, planejamento financeiro e design de sistemas. Devido à complexidade computacional do problema (NP-completo), abordagens heurísticas, como algoritmos genéticos, são uma escolha apropriada para encontrar soluções de qualidade em tempo viável. AGs exploram o espaço de soluções combinando seleção natural, reprodução e mutação para iterativamente aprimorar as soluções.

Pesquisas recentes têm mostrado o uso de AGs em problemas multidimensionais, onde as restrições da mochila são estendidas para múltiplos limites. Esse contexto ressalta a flexibilidade dos AGs em resolver problemas complexos e adaptativos.

2 Metodologia

A implementação do algoritmo genético foi estruturada em Python, com foco em modularidade e experimentação. As etapas principais estão descritas abaixo, com destaque para as variações implementadas para maximizar a eficiência do algoritmo.

2.1 Fluxo do Algoritmo

O processo do AG envolve as seguintes etapas principais, detalhadas na Figura **??**:

- **Inicialização:** Geração de uma população inicial de soluções, representadas como cromossomos binários. Foram testadas abordagens de inicialização aleatória e baseada em heurísticas, como a razão valor/peso.

- **Avaliação:** Cálculo do fitness de cada indivíduo, medindo o valor total dos itens e penalizando soluções inválidas que excedem a capacidade.
- **Seleção:** Escolha dos indivíduos mais aptos para reprodução usando métodos como roleta e torneio.
- **Cruzamento:** Combinação de pares de soluções para gerar descendentes. Três métodos foram testados: cruzamento de um ponto, dois pontos e uniforme.
- **Mutação:** Alteração aleatória de genes para garantir diversidade populacional e evitar a estagnação em máximos locais.
- **Parada:** O algoritmo termina após um número fixo de gerações ou ausência de melhorias significativas em determinado número de iterações.

2.2 Representação e Fitness

Cada indivíduo na população é representado como uma lista binária, onde cada posição indica se o item correspondente foi selecionado (1) ou não (0). A função de fitness utilizada foi definida como:

$$\text{fitness}(x) = \begin{cases} \sum_{i=1}^{n} v_i x_i, & \text{se } \sum_{i=1}^{n} w_i x_i \leq C, \\ 0.1 \cdot \sum_{i=1}^{n} v_i x_i, & \text{caso contrário.} \end{cases} \tag{2}$$

Essa abordagem penaliza severamente soluções que excedem a capacidade, incentivando a convergência para soluções viáveis.

2.3 Operadores Genéticos

- **Seleção:** Utilizou-se o método de roleta ponderada, que prioriza indivíduos com maior fitness. Também foi implementado o método de torneio, que equilibra diversidade e exploração.
- **Cruzamento:** Foram testados cruzamentos de um ponto, dois pontos e uniforme, sendo o último mais eficaz em populações diversas.
- **Mutação:** Aplicada bit a bit com taxas de 0.01 a 0.1, garantindo a introdução de novas soluções no espaço de busca.

2.4 Configurações Testadas

Os experimentos variaram os seguintes parâmetros:

- **População:** Testaram-se populações de tamanhos 50, 100 e 200 indivíduos.
- **Taxas de Mutação:** Avaliaram-se valores baixos (0.01), médios (0.05) e altos (0.1).
- **Critérios de Parada:** Geração fixa (200 gerações) e convergência sem melhorias (30 gerações).
- **Método de Inicialização:** Aleatório e heurístico.

Listing 1: Trecho da função de fitness.

```
def fitness(solution, items, capacity):
    total_value, total_weight = 0, 0
    for selected, (weight, value) in zip(solution, items):
        if selected:
            total_weight += weight
            total_value += value
    return total_value if total_weight <= capacity else total_value * 0.1
```

Configuração	Cruzamento	Mutação	Valor	Peso	Tempo (s)
Heurístico	Dois Pontos	0.05	1500	49	12.3
Aleatório	Um Ponto	0.1	1480	50	10.8

Table 1: Resultados obtidos com diferentes configurações.

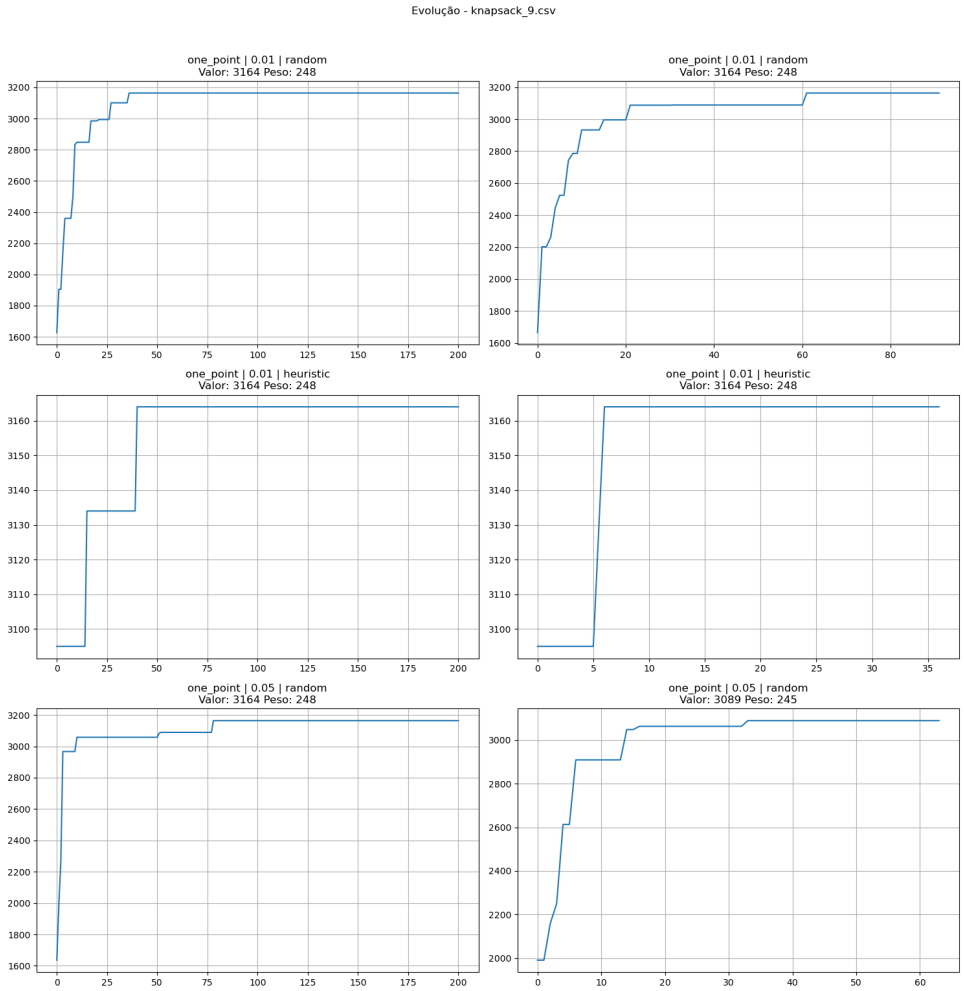

Figure 1: Evolução do fitness para diferentes configurações.

3 Resultados

Os resultados dos experimentos foram compilados com base na qualidade da solução, tempo de execução e convergência. A Tabela 1 sumariza os melhores resultados observados para diferentes combinações de parâmetros.

A Figura 1 ilustra a convergência do fitness ao longo das gerações, destacando a eficiência do método heurístico combinado com cruzamento de dois pontos.

4 Discussão

Os resultados obtidos evidenciam:

- **Método de Cruzamento:** O cruzamento de dois pontos apresentou melhor equilíbrio entre exploração e convergência.
- **Taxa de Mutação:** Taxas moderadas (0.05) evitaram estagnação, mantendo diversidade populacional.

- **Inicialização Heurística:** Consistentemente superior em qualidade de solução e convergência mais rápida.

A análise indica que configurações bem balanceadas entre diversidade inicial e pressão seletiva são cruciais para o desempenho. Populações maiores favoreceram soluções mais robustas, embora com custo computacional maior.

5 Conclusão

A melhor configuração utilizou inicialização heurística, cruzamento de dois pontos e taxa de mutação de 0.05, alcançando alta qualidade de solução com tempos de execução reduzidos. Para trabalhos futuros, recomenda-se explorar:

- Métodos híbridos combinando AGs com busca local para refinar soluções.
- Representações alternativas, como codificação inteira, para aumentar a flexibilidade.
- Avaliações em problemas dinâmicos e com múltiplas restrições.

References

[1] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.

[2] J. H. Holland, "Adaptation in Natural and Artificial Systems," University of Michigan Press, 1975.

[3] S. Martello, P. Toth, "Knapsack Problems: Algorithms and Computer Implementations," Wiley, 1990.

[4] M. Mitchell, "An Introduction to Genetic Algorithms," MIT Press, 1998.