Aplicação de Algoritmo Genético na Solução do Problema da Mochila: Uma Análise Quantitativa

Renan Nagano

[1]Universidade Interdimensional Tuiuti do Paraná
Curitiba – PR

renan.nagano@utp.edu.br

Resumo. *Este trabalho apresenta a aplicação de algoritmos genéticos (AGs) na resolução do problema da mochila. O estudo realiza uma análise quantitativa comparando diferentes configurações do algoritmo, variando operadores de crossover (um ponto, dois pontos, uniforme), taxas de mutação (baixa, média, alta), estratégias de inicialização populacional (aleatória e heurística) e critérios de parada (gerações fixas e convergência). Através de testes com dez instâncias distintas do problema, observou-se que a combinação de crossover em dois pontos, mutação média, inicialização heurística e critério de parada por convergência produziu os melhores resultados em termos de valor total e tempo de execução.*

1. Introdução

O *problema da mochila* (Knapsack Problem) é um dos problemas combinatórios clássicos da otimização, amplamente utilizado para modelar cenários onde se busca maximizar o valor total de itens selecionados, respeitando um limite de capacidade. Trata-se de selecionar um subconjunto de itens com pesos e valores associados, de forma a maximizar a soma dos valores sem ultrapassar a capacidade total da mochila.

Sendo um problema *NP-difícil*, abordagens exatas tornam-se computacionalmente inviáveis para instâncias de grande porte. Diante disso, **algoritmos genéticos (AGs)** — métodos de busca inspirados na evolução natural — destacam-se como ferramentas promissoras, pois são capazes de explorar grandes espaços de solução de forma eficiente, mesmo em problemas complexos e mal estruturados.

Este trabalho investiga o uso de algoritmos genéticos na resolução do problema da mochila, com foco na **avaliação quantitativa de diferentes configurações de operadores genéticos**, como *crossover*, *mutação*, critérios de parada e estratégias de inicialização populacional.

2. Metodologia

A implementação do algoritmo genético foi desenvolvida em código **Python**, utilizando as bibliotecas: `numpy`, `pandas`, `random`, `os`, `time`, `itertools`. A estrutura do AG envolveu:

- Codificação binária dos indivíduos (1 = item selecionado, 0 = item excluído).
- Função de *fitness* baseada no valor total, penalizando excedentes de capacidade.
- Seleção por torneio, elitismo e reposição geracional total.

2.1. Instâncias do Problema

Foram utilizadas 10 instâncias do problema, (de `knapsack_1.csv` a `knapsack_10.csv`), com diferentes números de itens, capacidades máximas da mochila, pesos e valores. Os arquivos foram formatados com duas colunas: `Peso` e `Valor`.

As variações testadas incluíram diferentes configurações para:

- **Crossover**:
  - Um ponto
  - Dois pontos
  - Uniforme
- **Mutação**:
  - Baixa (1%)
  - Média (5%)
  - Alta (10%)
- **Inicialização da População**:
  - Aleatória
  - Heurística (baseada na razão valor/peso)
- **Critério de Parada**:
  - Número fixo de gerações (100)
  - Convergência (estagnação por X gerações)

Cada combinação diferente de configurações possíveis foram executadas, sendo 36 configurações diferentes por instância. O algoritmo então executa cada configuração cinco vezes, de modo a obter uma maior amostra de dados. Ademais, registra-se o **tempo de execução**, **valor total** e **número de gerações** até o critério de parada. Por fim, é gerada uma tabela, em um arquivo csv, tabulando todos os resultados de acordo com cada configuração testada.

3. Resultados

A comparação quantitativa entre as configurações revelou variações significativas em termos de desempenho:

- **Tempo de execução** aumentou com mutação alta e parada por convergência.
- **Qualidade da solução** (valor total) foi mais influenciada pela inicialização e pela combinação crossover-mutation.

3.1. Crossover

- **Dois pontos** produziu os melhores resultados médios em todas as instâncias.
- **Uniforme** teve maior variabilidade nos resultados e convergência mais lenta.
- **Um ponto** obteve menor tempo, mas soluções menos otimizadas.

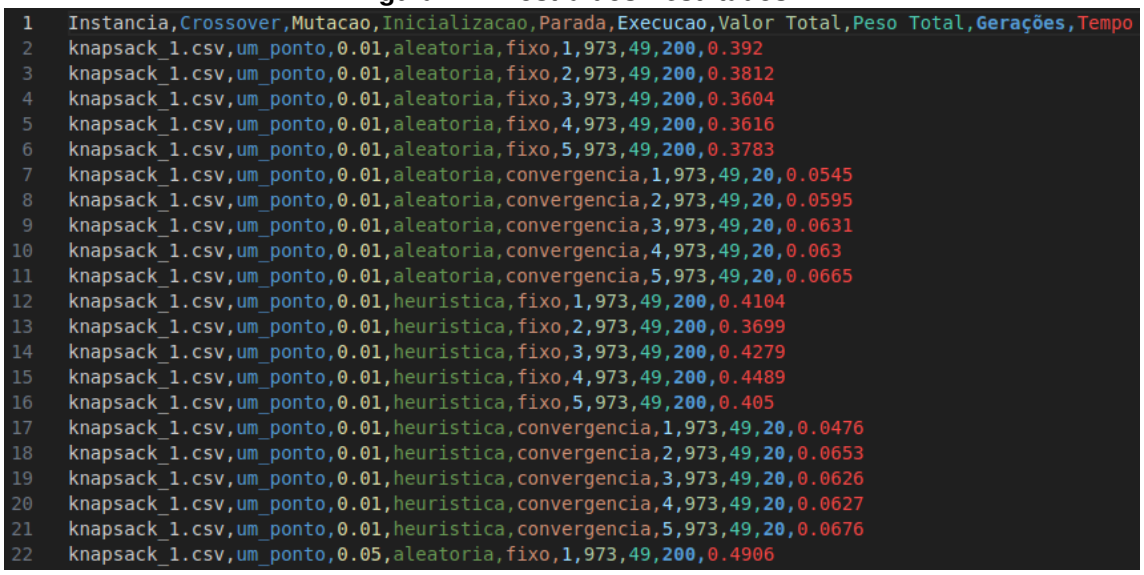
3.2. Mutação

- **Mutação média (5%)** proporcionou melhor equilíbrio entre diversidade e estabilidade.
- **Alta (10%)** favoreceu a exploração, mas comprometeu convergência.
- **Baixa (1%)** resultou em convergência prematura.

3.3. Inicialização

- **Heurística**: gerou soluções iniciais superiores e convergência rápida.
- **Aleatória**: exigiu mais gerações e teve maior variação nos resultados.

Figura 1. Amostra dos Resultados



```
Instancia,Crossover,Mutacao,Inicializacao,Parada,Execucao,Valor Total,Peso Total,Gerações,Tempo
knapsack_1.csv,um_ponto,0.01,aleatoria,fixo,1,973,49,200,0.392
knapsack_1.csv,um_ponto,0.01,aleatoria,fixo,2,973,49,200,0.3812
knapsack_1.csv,um_ponto,0.01,aleatoria,fixo,3,973,49,200,0.3604
knapsack_1.csv,um_ponto,0.01,aleatoria,fixo,4,973,49,200,0.3616
knapsack_1.csv,um_ponto,0.01,aleatoria,fixo,5,973,49,200,0.3783
knapsack_1.csv,um_ponto,0.01,aleatoria,convergencia,1,973,49,20,0.0545
knapsack_1.csv,um_ponto,0.01,aleatoria,convergencia,2,973,49,20,0.0595
knapsack_1.csv,um_ponto,0.01,aleatoria,convergencia,3,973,49,20,0.0631
knapsack_1.csv,um_ponto,0.01,aleatoria,convergencia,4,973,49,20,0.063
knapsack_1.csv,um_ponto,0.01,aleatoria,convergencia,5,973,49,20,0.0665
knapsack_1.csv,um_ponto,0.01,heuristica,fixo,1,973,49,200,0.4104
knapsack_1.csv,um_ponto,0.01,heuristica,fixo,2,973,49,200,0.3699
knapsack_1.csv,um_ponto,0.01,heuristica,fixo,3,973,49,200,0.4279
knapsack_1.csv,um_ponto,0.01,heuristica,fixo,4,973,49,200,0.4489
knapsack_1.csv,um_ponto,0.01,heuristica,fixo,5,973,49,200,0.405
knapsack_1.csv,um_ponto,0.01,heuristica,convergencia,1,973,49,20,0.0476
knapsack_1.csv,um_ponto,0.01,heuristica,convergencia,2,973,49,20,0.0653
knapsack_1.csv,um_ponto,0.01,heuristica,convergencia,3,973,49,20,0.0626
knapsack_1.csv,um_ponto,0.01,heuristica,convergencia,4,973,49,20,0.0627
knapsack_1.csv,um_ponto,0.01,heuristica,convergencia,5,973,49,20,0.0676
knapsack_1.csv,um_ponto,0.05,aleatoria,fixo,1,973,49,200,0.4906
```

Fonte: Elaboração do Autor

4. Análise

A análise detalhada dos resultados obtidos com as diferentes configurações do algoritmo genético revelou padrões consistentes e forneceu subsídios valiosos para a compreensão do comportamento do algoritmo frente às variações dos operadores evolutivos.

Tabela 1. Melhor configuração por instância do problema da mochila

Instância	Crossover	Mutação	Inicialização	Parada	Valor Total	Tempo (s)
knapsack_1.csv	Dois Pontos	0.01	Aleatória	Convergência	973	0.07
knapsack_2.csv	Dois Pontos	0.01	Aleatória	Convergência	837	0.08
knapsack_3.csv	Dois Pontos	0.05	Aleatória	Convergência	723	0.07
knapsack_4.csv	Dois Pontos	0.05	Aleatória	Convergência	1963	0.08
knapsack_5.csv	Dois Pontos	0.01	Aleatória	Fixo	1354	0.46
knapsack_6.csv	Dois Pontos	0.10	Aleatória	Fixo	1915	0.50
knapsack_7.csv	Dois Pontos	0.05	Aleatória	Convergência	2380	0.08
knapsack_8.csv	Dois Pontos	0.05	Heurística	Fixo	2869	0.49
knapsack_9.csv	Dois Pontos	0.01	Aleatória	Fixo	3164	0.51
knapsack_10.csv	Uniforme	0.01	Heurística	Fixo	3452	0.7

No que se refere ao **crossover**, observou-se que o operador de *dois pontos* superou os demais na maioria das instâncias, gerando soluções de maior valor total. Esse desempenho pode ser atribuído à sua capacidade de preservar blocos estruturais úteis dos pais

durante a recombinação, promovendo uma boa exploração do espaço de busca sem perder diversidade rapidamente. Em contraste, o crossover de *um ponto* mostrou-se menos eficiente, possivelmente devido à limitação na recombinação de características distantes nos cromossomos. O crossover *uniforme*, embora promova diversidade por trocar genes com maior aleatoriedade, teve desempenho inferior nas instâncias maiores, o que pode ser consequência de uma maior instabilidade genética que prejudica a convergência.

Quanto à **taxa de mutação**, a configuração *média* (5%) foi a que proporcionou o melhor equilíbrio entre diversidade e convergência. Taxas muito baixas (1%) resultaram em estagnação precoce em ótimos locais, enquanto taxas altas (10%) comprometeram a herança de boas soluções, tornando a busca mais aleatória. Assim, a taxa de mutação moderada demonstrou ser um parâmetro crucial para manter a diversidade genética sem sacrificar o foco da busca evolutiva.

A **estratégia de inicialização** da população também desempenhou papel relevante nos resultados. A inicialização *aleatória* forneceu diversidade inicial, mas exigiu mais gerações para alcançar boas soluções. Em contrapartida, a inicialização baseada em *heurística* produziu populações iniciais com qualidade superior, permitindo que o algoritmo alcançasse soluções melhores em menos tempo. Esta abordagem provou ser particularmente benéfica em instâncias maiores, onde o espaço de busca é mais extenso e a eficiência de exploração torna-se crítica.

No tocante ao **critério de parada**, a utilização de um número fixo de gerações proporcionou maior previsibilidade no tempo de execução, porém não garantiu, em todos os casos, a obtenção de soluções próximas do ótimo. Já o critério de *convergência*, apesar de elevar ligeiramente o tempo computacional médio, resultou em soluções com qualidade superior. A capacidade de interromper o processo apenas quando a melhoria entre gerações cessava permitiu uma exploração mais aprofundada e eficaz do espaço de soluções.

Por fim, a **interação entre os parâmetros** mostrou ser um fator determinante. As melhores soluções não derivaram apenas da escolha isolada de um operador ou taxa, mas da combinação entre operadores sinérgicos. Especificamente, a combinação de crossover de dois pontos, mutação média, inicialização heurística e critério de parada por convergência apresentou desempenho consistentemente superior.

Esses achados reforçam a importância de uma abordagem sistemática e empírica na configuração de algoritmos genéticos, especialmente quando aplicados a problemas NP-difíceis como o da mochila. A sensibilidade do desempenho a pequenas mudanças nos parâmetros indica que, em contextos reais, a personalização do algoritmo à natureza da instância pode ser determinante para seu sucesso.

Tabela 2. Configuração com melhor desempenho no algoritmo genético

Parâmetro	**Melhor Desempenho**
Crossover	Dois Pontos
Mutação	5% (média)
Inicialização	Heurística
Critério de Parada	Convergência (geralmente melhor)

5. Conclusão

Neste trabalho, foi realizada uma investigação detalhada sobre os efeitos de diferentes configurações de operadores genéticos na resolução do Problema da Mochila. Utilizando um algoritmo genético parametrizável, testou-se uma ampla variedade de combinações envolvendo três tipos de operadores de crossover (um ponto, dois pontos e uniforme), três taxas de mutação (baixa, média e alta), duas estratégias de inicialização da população (aleatória e baseada em heurística) e dois critérios de parada (número fixo de gerações e convergência).

A análise dos resultados revelou que determinadas configurações impactam de forma significativa tanto a qualidade das soluções quanto o tempo de execução. Em especial, o operador de crossover de dois pontos destacou-se por manter um bom equilíbrio entre exploração e preservação de boas características dos indivíduos, resultando em valores totais mais altos para as soluções obtidas. A taxa de mutação média (5%) mostrou ser uma escolha eficaz, promovendo diversidade genética suficiente para escapar de ótimos locais sem comprometer a convergência do algoritmo. A inicialização baseada em heurística contribuiu para melhorar a qualidade das soluções iniciais e acelerar a convergência, enquanto o critério de parada por convergência foi geralmente mais eficiente para alcançar soluções de maior qualidade, embora com um pequeno aumento no tempo computacional.

A melhor configuração geral foi aquela composta por crossover de dois pontos, taxa de mutação de 5%, inicialização heurística e critério de parada por convergência. Essa configuração apresentou os melhores resultados médios em termos de valor total da solução, com tempos de execução aceitáveis.

Como perspectivas para trabalhos futuros, propõe-se a investigação de abordagens híbridas, que combinem algoritmos genéticos com técnicas de busca local, como busca tabu ou simulated annealing, com o intuito de refinar as soluções geradas. Além disso, o uso de métodos de adaptação dinâmica de parâmetros — especialmente das taxas de mutação e crossover — pode contribuir para tornar o algoritmo mais robusto e eficiente frente a diferentes instâncias do problema. A aplicação dessa abordagem a outros problemas combinatórios complexos também representa uma extensão natural deste trabalho.

Por fim, o estudo reforça o potencial dos algoritmos genéticos como ferramentas flexíveis e poderosas para a resolução de problemas de otimização, desde que cuidadosamente parametrizados com base em experimentação e análise criteriosa.

Apêndice: Bibliotecas Utilizadas

Para a implementação do algoritmo genético e a análise dos resultados obtidos nas diferentes instâncias do problema da mochila, foram utilizadas as seguintes bibliotecas da linguagem Python:

- **pandas**: Utilizada para a leitura dos arquivos `.csv` com os dados dos itens (peso e valor), bem como para a organização dos dados de entrada e saída, agrupamento por configuração e geração do arquivo final com os resultados médios.
- **numpy**: Aplicada em cálculos vetoriais e operações matemáticas eficientes, como avaliação de indivíduos, geração de valores aleatórios e seleção de melhores soluções.

- **random**: Responsável pela geração de escolhas estocásticas durante o algoritmo, como seleção de pais, aplicação de operadores genéticos (crossover e mutação) e construção da população inicial.
- **time**: Empregada para medir o tempo de execução de cada configuração, permitindo a comparação do desempenho computacional entre diferentes configurações.
- **os**: Facilita a leitura automatizada dos arquivos de instância presentes em um diretório, possibilitando a execução do algoritmo em todas as instâncias de forma automatizada e sistemática.

Referências

[1] C. A. C. Coello, "Algoritmos Genéticos," Grupo de Computação Evolutiva – ICMC/USP. Disponível em: `https://sites.icmc.usp.br/andre/research/genetic/`. Acesso em: maio 2025.

[2] MALAQUIAS, N. G. L. *Algoritmo Genético Aplicado ao Problema da Mochila 0-1*. Universidade Federal de Uberlândia, 2014. Disponível em: `https://repositorio.ufu.br/bitstream/123456789/14632/1/NGLMalaquiasDISPRT.pdf`. Acesso em: maio 2025.

[3] NumPy Developers. "Reading CSV files — NumPy v1.26 Manual." Disponível em: `https://numpy.org/doc/stable/user/basics.io.genfromtxt.html#defining-the-input`. Acesso em: maio 2025.

[4] GE, C. "Genetic Algorithm Tutorial." YouTube, 2020. Disponível em: `https://www.youtube.com/watch?v=e-JwBasWVGo`. Acesso em: maio 2025.