Implementação de algoritmo de AG para sesolver o problema da mochila (Knapsack Problem) usando diferentes configurações e variações de parâmetros

Rafael Grabowski de Lima[1]

[1]Universidade Interdimensional Tuiuti do Paraná
Curitiba – PR

{RAFAEL.LIMA2}@utp.edu.br

Resumo. *Este trabalho apresenta um estudo experimental da aplicação de Algoritmos Genéticos (AGs) na resolução do Problema da Mochila (Knapsack Problem), com foco na análise quantitativa do impacto de diferentes configurações de parâmetros sobre o desempenho do algoritmo. Foram investigadas três estratégias de crossover (um ponto, dois pontos e uniforme), combinadas com diferentes taxas de mutação (baixas, médias e altas), e dois métodos distintos de inicialização da população (aleatória e baseada em heurísticas). Além disso, foram avaliados dois critérios de parada: número fixo de gerações e convergência para uma solução estável.*

Os experimentos foram conduzidos sobre instâncias com complexidade crescente, e os resultados foram analisados com base em tempo de execução, qualidade da solução (valor total alcançado na mochila) e estabilidade (desvio padrão das soluções obtidas). Os resultados indicam que determinadas combinações de operadores e parâmetros promovem melhorias significativas na eficiência e na qualidade das soluções, destacando a importância da calibragem adequada do AG para problemas de otimização combinatória.

1. Introdução

Problemas de otimização combinatória são frequentes em diversos campos, como logística, alocação de recursos, engenharia e ciência da computação. Um dos problemas mais estudados nesta categoria é o Problema da Mochila (Knapsack Problem), que consiste em selecionar um subconjunto de itens com diferentes pesos e valores, de forma a maximizar o valor total carregado, sem ultrapassar a capacidade da mochila. Trata-se de um problema NP-difícil, para o qual métodos exatos tornam-se inviáveis à medida que o número de itens aumenta.

Para lidar com essas limitações, técnicas de meta-heurísticas têm sido amplamente exploradas. Entre elas, os Algoritmos Genéticos (AGs) se destacam por sua simplicidade, flexibilidade e eficiência na busca por soluções aproximadas de alta qualidade. Inspirados nos mecanismos de seleção natural, AGs evoluem uma população de soluções ao longo de várias gerações, utilizando operadores como seleção, crossover e mutação.

Este trabalho tem como objetivo principal implementar um Algoritmo Genético para resolver o Problema da Mochila e realizar uma análise comparativa entre diferentes configurações dos seus parâmetros. Foram exploradas variações nos seguintes aspectos:

Estratégias de seleção (roleta, torneio, etc.);

Taxas de crossover (Uniforme, de um e dois pontos);

Tamanhos populacionais;

Critérios de parada (por número de gerações e convergência).

Inicialização aleatória e por Heurísticas

Cada configuração foi testada em instâncias do problema com complexidade crescente, e os resultados foram analisados com base em métricas quantitativas como tempo de execução, qualidade da solução (valor total da mochila), e estabilidade dos resultados (desvio padrão entre execuções).

A proposta visa não apenas resolver o problema da mochila de forma eficiente, mas também avaliar o impacto de cada configuração no desempenho do AG, oferecendo subsídios para futuras aplicações práticas em cenários similares de otimização.

2. Metodologia

Neste trabalho, implementou-se um Algoritmo Genético (AG) para resolver o Problema da Mochila 0/1 (*0/1 Knapsack Problem*). O objetivo é selecionar um subconjunto de itens com o maior valor total possível, respeitando a restrição de capacidade da mochila. As soluções são representadas por cromossomos binários, onde cada gene indica a presença (1) ou ausência (0) de um item.

2.1. Representação e Avaliação

Cada indivíduo é codificado como uma cadeia binária de comprimento igual ao número de itens. A função de avaliação (*fitness*) calcula o valor total dos itens incluídos, aplicando uma penalidade proporcional caso o peso exceda a capacidade da mochila.

2.2. Configurações de Crossover

Três estratégias de recombinação foram testadas:

- **Crossover de um ponto:** um ponto de corte é escolhido aleatoriamente, e os genes subsequentes são trocados entre os pais.
- **Crossover de dois pontos:** dois pontos de corte são definidos, e os genes entre eles são trocados.
- **Crossover uniforme:** cada gene é escolhido aleatoriamente de um dos pais com probabilidade igual.

2.3. Configurações de Mutação

A mutação inverte bits aleatórios com diferentes taxas:

- **Taxa baixa:** 1%
- **Taxa média:** 5%
- **Taxa alta:** 10%

2.4. Inicialização da População

Duas estratégias de inicialização da população foram comparadas:

- **Aleatória:** todos os genes de cada indivíduo são definidos de forma aleatória.
- **Heurística:** parte da população é gerada com base na razão valor/peso, priorizando itens mais valiosos por unidade de peso.

2.5. Critérios de Parada

Os critérios de término utilizados foram:

- **Número fixo de gerações:** a execução é interrompida após um número predefinido de gerações.
- **Convergência:** o algoritmo para quando não há melhora significativa na melhor solução durante um número consecutivo de gerações.

2.6. Parâmetros Fixos

Os parâmetros abaixo foram mantidos constantes em todos os testes:

- Tamanho da população: 50 indivíduos
- Probabilidade de crossover: 0,8
- Elitismo: 2 indivíduos mantidos por geração
- Método de seleção: torneio binário

2.7. Instâncias e Repetições

As configurações foram testadas em instâncias com 50, 100 e 200 itens, com capacidade da mochila proporcional ao peso médio dos itens. Cada experimento foi repetido 10 vezes para reduzir o impacto da aleatoriedade.

As métricas utilizadas para análise dos resultados foram:

- Valor médio das soluções obtidas
- Tempo médio de execução
- Desvio padrão da melhor solução

3. Resultados e Discussão

Nesta seção, comparam-se os resultados obtidos pelas diferentes configurações do Algoritmo Genético aplicadas ao Problema da Mochila 0/1. As métricas analisadas foram:

- **Qualidade da solução:** valor médio da melhor solução encontrada.
- **Tempo de execução:** tempo médio de processamento por execução.

3.1. Variações de Crossover

A Tabela 1 apresenta a comparação entre os três métodos de crossover utilizados.

Tabela 1. Comparação entre tipos de crossover

Crossover	Valor médio da solução	Tempo médio (s)
Um ponto	1055,8	1,23
Dois pontos	1083,2	1,28
Uniforme	1062,6	1,35

Observa-se que o crossover de dois pontos apresentou o melhor desempenho em termos de qualidade da solução, embora tenha um leve aumento no tempo de execução.

3.2. Variações na Taxa de Mutação

A Tabela 2 mostra o impacto de diferentes taxas de mutação sobre os resultados.

Tabela 2. Comparação entre taxas de mutação

Taxa de Mutação	Valor médio da solução	Tempo médio (s)
Baixa (1%)	1018,5	1,19
Média (5%)	1067,4	1,27
Alta (10%)	1035,2	1,30

A taxa de mutação média obteve os melhores resultados, equilibrando a exploração e evitando a estagnação prematura.

3.3. Inicialização da População

A Tabela 3 compara a inicialização aleatória e heurística da população.

Tabela 3. Comparação entre estratégias de inicialização

Inicialização	Valor médio da solução	Tempo médio (s)
Aleatória	1024,7	1,15
Heurística	1095,3	1,42

A abordagem heurística mostrou-se mais eficaz na obtenção de soluções de maior qualidade, com um custo de tempo ligeiramente maior.

3.4. Critérios de Parada

A Tabela 4 apresenta os resultados para diferentes critérios de parada.

Tabela 4. Comparação entre critérios de parada

Critério de Parada	Valor médio da solução	Tempo médio (s)
Gerações fixas (100)	1032,8	1,20
Convergência (10 sem melhora)	1087,1	1,45

O critério baseado em convergência superou o método de gerações fixas em qualidade da solução, apesar de requerer mais tempo de execução.

3.5. Análise individual de alguns arquivos contendo diferentes valores e pesos da mochila

[h!]

Arquivo	Melhor Valor Encontrado	Observações
1	973	Alcançado por praticamente **todas as configurações**, exceto uma com `uniforme + aleatória + mutação 0.3`, que obteve 944.
2	837	Todas as configurações chegaram a esse valor. A inicialização heurística e cruzamentos diversos mostraram **ótima estabilidade**.
3	**723** (predominante) **678** (pontual) **0** (diversas vezes)	Resultados **altamente instáveis** com `mutação=0.3`. Algumas execuções retornaram **valor zero**, indicando **colapso da população**.
4	1963	Todas as execuções (até onde fornecido) atingiram esse valor. Comportamento **estável e ótimo**.
5	1354	A maior parte dos testes ficou entre **1327 e 1354**. Configurações com mutação 0.02 e 0.1 se saíram melhor.

Figura 1. Análise induvidual dos arquivos

3.6. Análise Geral

De forma geral, as melhores configurações observadas foram:

- **Crossover:** dois pontos
- **Mutação:** taxa média (5%)
- **Inicialização:** heurística
- **Critério de parada:** convergência

Essa combinação demonstrou a melhor relação custo-benefício entre tempo de execução e qualidade da solução, indicando um bom equilíbrio entre exploração do espaço de busca e eficiência computacional.

4. Conclusão e Melhorias futuras

Este trabalho implementou e avaliou um Algoritmo Genético (AG) para resolver o Problema da Mochila 0/1, com o objetivo de identificar as configurações de parâmetros que otimizam a qualidade da solução e o tempo de execução. Através de uma análise sistemática de diferentes estratégias de crossover, taxas de mutação, métodos de inicialização da população e critérios de parada, foi possível determinar a melhor configuração para o cenário estudado.

4.1. Melhor configuração encontrada

A combinação de parâmetros que demonstrou a melhor relação custo-benefício, equilibrando a exploração do espaço de busca com a eficiência computacional, foi:

Crossover de dois pontos: Este método superou as abordagens de um ponto e uniforme, gerando soluções de maior valor, mesmo com um ligeiro aumento no tempo de processamento. Sua capacidade de recombinação mais diversificada provavelmente contribui para uma melhor exploração do espaço de soluções. Mutação com taxa média (5Inicialização heurística: A estratégia de inicializar a população com base na razão valor/peso dos itens provou ser significativamente mais eficaz. Ao partir de soluções iniciais com maior qualidade intrínseca, o AG consegue convergir para resultados superiores de forma mais eficiente. Critério de parada por convergência: Embora exigindo um tempo

de execução um pouco maior, a parada por convergência garantiu que o algoritmo continuasse a buscar melhorias até que a solução não apresentasse mais ganhos significativos. Isso resultou em soluções de maior qualidade em comparação com um número fixo de gerações, que pode parar o processo antes que a melhor solução seja encontrada. Essa configuração específica destacou-se por consistentemente gerar soluções de maior valor médio, consolidando-se como a abordagem mais robusta e eficaz para o Problema da Mochila 0/1 no contexto desta pesquisa.

4.2. Sugestões de melhorias futuras

Para aprimorar ainda mais o desempenho do Algoritmo Genético e explorar outras vertentes do Problema da Mochila, sugerem-se as seguintes melhorias e linhas de pesquisa futuras:

Otimização dos Parâmetros via Meta-heurísticas: Em vez de testar configurações de forma manual, a aplicação de técnicas como Algoritmos Genéticos de segunda ordem (para otimizar os próprios parâmetros do AG), Recozimento Simulado ou Otimização por Enxame de Partículas poderia automatizar e refinar a busca pelos melhores parâmetros.

Exploração de Outros Operadores Genéticos: Investigar operadores de seleção (e.g., seleção por ranking, seleção elitista mais agressiva), crossover (e.g., multiponto adaptativo) e mutação (e.g., mutação adaptativa, mutação por troca) pode revelar combinações ainda mais poderosas. Problema da Mochila Multi-objetivo: Estender o algoritmo para lidar com versões multi-objetivo do Problema da Mochila, onde há múltiplos critérios a serem otimizados simultaneamente (e.g., maximizar valor e minimizar peso, ou considerar múltiplas capacidades), seria um avanço significativo.

Referências

1.Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press.

2.Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley.

3.Fogel, D. B. (1995). Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press.