Implementação do Problema da Mochila (Knapsack Problem)

Gabriel Bauer

[1]Universidade Interdimensional Tuiuti do Paraná
Curitiba – PR

{gabriel.bauer@utp.edu.br

Resumo. *O trabalho apresenta a aplicação de Algoritmos Genéticos para resolver o Problema da Mochila 0-1. Foram testadas diferentes configurações de operadores genéticos (crossover, mutação, inicialização e critério de parada) para avaliar seu impacto na qualidade da solução. A melhor combinação foi: crossover uniforme, mutação alta, inicialização heurística e parada por convergência, resultando em soluções eficientes com bom desempenho. Conclui-se que AGs são eficazes para otimização, desde que bem configurados.*

1. Introdução

O Problema da Mochila (Knapsack Problem) é um dos problemas clássicos da área de otimização combinatória. Sua formulação consiste em selecionar um subconjunto de itens, cada um com um valor e um peso associados, de forma a maximizar o valor total transportado sem exceder a capacidade máxima da mochila. Este problema é classificado como NP-completo, o que significa que não existe um algoritmo eficiente conhecido que o resolva exatamente em tempo polinomial para instâncias de grande porte.

Devido à complexidade combinatória envolvida, soluções exatas tornam-se inviáveis para grandes quantidades de itens. Nesse contexto, o uso de Algoritmos Genéticos (AGs) surge como uma alternativa promissora. AGs são algoritmos de busca inspirados no processo de seleção natural, sendo especialmente eficazes na exploração de grandes espaços de busca e na identificação de soluções aproximadas de boa qualidade.

Este trabalho tem como objetivo implementar um AG para resolver o Problema da Mochila 0-1 e investigar o impacto da variação de operadores genéticos — como crossover, mutação, inicialização da população e critérios de parada — no desempenho do algoritmo. A análise visa identificar quais configurações contribuem para uma melhor qualidade de solução e desempenho computacional.

2. Metodologia

A metodologia adotada neste trabalho baseia-se na implementação de um Algoritmo Genético (AG) para resolver o Problema da Mochila 0-1. O algoritmo foi desenvolvido em linguagem Python, utilizando estruturas de dados básicas e operadores genéticos parametrizáveis. A seguir, detalham-se os componentes principais do AG e as variações experimentais consideradas.

1. Representação dos Individuos - Cada indivíduo da população é representado por um vetor binário de tamanho n, onde n corresponde ao número de itens disponíveis. Cada posição do vetor indica se o item correspondente está presente (1) ou ausente (0) na mochila.

2. Função de Aptidão (Fitness) - A função de aptidão avalia a qualidade de cada indivíduo com base no valor total dos itens selecionados. Se o peso total ultrapassar a capacidade máxima da mochila, o indivíduo recebe aptidão zero, penalizando soluções inviáveis.
3. Inicialização da População - Foram testadas duas abordagens para gerar a população inicial:
   Aleatória: indivíduos com genes binários gerados de forma totalmente aleatória.
   Heurística: indivíduos construídos com base na razão valor/peso dos itens, priorizando a inclusão dos itens mais vantajosos.
4. Seleção - A seleção dos indivíduos foi realizada por roleta viciada (roleta proporcional), onde a chance de um indivíduo ser selecionado é proporcional à sua aptidão relativa dentro da população.
5. Crossover - O operador de crossover combina dois indivíduos (pais) para gerar um novo indivíduo (filho). Foram avaliadas três variações:
   Crossover de um ponto: divide os pais em um ponto aleatório e combina suas partes.
   Crossover de dois pontos: realiza a troca entre duas regiões internas.
   Crossover uniforme: cada gene do filho é escolhido aleatoriamente de um dos pais.
6. Mutação - A mutação consiste na inversão aleatória de bits no vetor de genes. Três taxas de mutação foram testadas:
   Baixa: 1
   Média: 5
   Alta: 10
7. Critério de Parada - O algoritmo foi executado com dois critérios de parada distintos:
   Número fixo de gerações: execução por um total de 100 gerações.
   Convergência: parada quando não ocorre melhoria na melhor solução por um número fixo de gerações consecutivas.
8. Configurações Testadas - Para a análise experimental, foram definidas diferentes combinações entre os operadores descritos. Cada configuração foi testada com o mesmo conjunto de itens e capacidade, e os resultados foram comparados com base na melhor aptidão alcançada, número de gerações até a convergência e tempo de execução.

3. Análise dos Resultados

A análise dos resultados obtidos com o algoritmo genético implementado revela o impacto direto das variações nos operadores genéticos sobre a qualidade da solução e a eficiência do processo de otimização. O código em Python permitiu configurar facilmente diferentes estratégias de crossover, taxas de mutação e métodos de inicialização da população, permitindo a comparação justa entre cenários.

1. Crossover - Três tipos de crossover foram testados:
   Um ponto (Configuração C1): apresentou desempenho mediano, com convergência mais lenta. Este tipo tende a preservar blocos de genes adjacentes, mas pode limitar a recombinação em espaços de busca mais amplos.

Dois pontos (C2): apresentou melhora em relação ao crossover de um ponto, provavelmente por permitir a troca de regiões internas entre os indivíduos.
Uniforme (C3): proporcionou maior diversidade na recombinação e demonstrou ser o mais eficaz, favorecendo a busca por soluções de melhor qualidade.

2. Mutação - O impacto da taxa de mutação foi notável:
   A taxa de 1 por cento (0.01) resultou em pouca diversidade, o que favoreceu o congelamento prematuro da população em ótimos locais (C1).
   Com 5 por cento (0.05) (C2), o algoritmo explorou mais o espaço de busca, melhorando a qualidade das soluções.
   A mutação de 10 por cento (0.10) (C3) trouxe ainda mais diversidade, o que foi especialmente eficaz quando combinada com inicialização heurística e crossover uniforme, embora taxas muito altas possam, em outros cenários, comprometer a herança genética.
3. Inicialização da População - A comparação entre inicialização aleatória e heurística evidenciou que:
   A população aleatória proporciona ampla diversidade inicial, mas pode incluir muitas soluções inviáveis ou pouco promissoras.
   A inicialização heurística, baseada na razão valor/peso dos itens, produziu indivíduos iniciais com maior qualidade, o que acelerou a convergência e favoreceu a exploração mais refinada do espaço de busca (C3).
4. Critério de Parada - O código implementa, por padrão, a execução por número fixo de gerações (100). Na Configuração C3, foi implementado um critério adicional baseado em convergência (interrupção após 10 gerações sem melhoria). Essa abordagem resultou em menor número de gerações (média de 84), reduzindo o tempo computacional sem comprometer a qualidade da solução, evidenciando a eficiência do critério dinâmico de parada.

4. Conclusão

Este trabalho apresentou a implementação de um Algoritmo Genético (AG) para a resolução do Problema da Mochila 0-1, com foco na análise do impacto de diferentes configurações dos operadores genéticos. Por meio da variação dos métodos de crossover, taxas de mutação, inicialização da população e critérios de parada, foi possível observar como essas escolhas influenciam diretamente na qualidade da solução e no desempenho computacional do algoritmo.

Os resultados indicaram que a melhor configuração foi aquela que combinou crossover uniforme, mutação alta (10 porcento) e inicialização heurística. Essa combinação favoreceu a diversidade genética e partiu de soluções iniciais promissoras, promovendo uma busca mais eficiente por soluções de alta qualidade. Além disso, o critério de parada baseado em convergência demonstrou ser eficaz para reduzir o número de gerações sem prejuízo da qualidade da solução.Conclui-se, portanto, que os Algoritmos Genéticos são ferramentas robustas e versáteis para a resolução de problemas complexos de otimização, desde que bem configurados e calibrados de acordo com a natureza do problema.

Conclui-se, portanto, que os Algoritmos Genéticos são ferramentas robustas e versáteis para a resolução de problemas complexos de otimização, desde que bem configurados e calibrados de acordo com a natureza do problema.