Implementação e Análise de Algoritmos Genéticos para o Problema da Mochila

Murilo Henrique Baruffi[1]

[1]Universidade Tuiuti do Paraná
Curitiba – PR

{murilo.baruffi}@upt.edu.br

Resumo. *Este trabalho apresenta a implementação de um Algoritmo Genético (AG) para resolver o Problema da Mochila 0-1. Foram testadas diferentes configurações dos operadores genéticos, como tipos de crossover, taxas de mutação, métodos de inicialização e critérios de parada. O objetivo foi analisar o impacto dessas variações na qualidade das soluções e no tempo de execução do algoritmo.*

1. Introdução

O Problema da Mochila 0-1 é um problema clássico de otimização combinatória. Dado um conjunto de itens, cada um com peso e valor, o objetivo é determinar a melhor combinação de itens a serem colocados em uma mochila de capacidade limitada, de modo a maximizar o valor total. Por ser NP-difícil, métodos exatos são inviáveis para instâncias maiores, sendo comum o uso de heurísticas como Algoritmos Genéticos (AGs).

AGs são meta-heurísticas inspiradas na seleção natural. Operam sobre uma população de soluções, aplicando operadores de seleção, crossover e mutação para buscar soluções cada vez melhores ao longo das gerações. Neste trabalho, implementamos um AG para o problema da mochila e analisamos o impacto de diferentes configurações de operadores genéticos.

2. Metodologia

O algoritmo foi implementado em Python. Utilizamos uma instância do problema da mochila com cinco itens, com pesos e valores fixos, e capacidade máxima de 50 unidades. Foram testadas três taxas de mutação (0.01, 0.10, 0.30), três tipos de crossover (um ponto, dois pontos e uniforme), dois métodos de inicialização (aleatória e heurística) e dois critérios de parada (100 gerações fixas ou convergência por 5 gerações sem melhora). Ao todo, foram avaliadas 36 combinações.

A função de fitness considera o valor total dos itens, respeitando o limite de peso. A seleção foi feita por torneio e os resultados (valor final e tempo de execução) foram registrados para cada configuração.

3. Resultados

A Tabela 1 apresenta as dez melhores configurações obtidas entre as 36 combinações testadas. Todas alcançaram o valor ótimo (260), mas com pequenas variações no tempo de execução. Configurações que utilizaram inicialização heurística e critério de parada por convergência apresentaram os melhores tempos. O crossover uniforme, quando combinado com essas estratégias, gerou resultados rápidos e estáveis.

Tabela 1. Dez melhores configurações do Algoritmo Genético

Crossover	Mutação	Inicialização	Critério Parada	Valor Final	Tempo (s)
uniforme	0.01	heurística	convergência	260	0.000
uniforme	0.30	heurística	convergência	260	0.000
uniforme	0.10	heurística	convergência	260	0.000
um_ponto	0.01	aleatório	convergência	260	0.001
um_ponto	0.01	heurística	convergência	260	0.001
um_ponto	0.10	aleatório	convergência	260	0.001
um_ponto	0.10	heurística	convergência	260	0.001
um_ponto	0.30	aleatório	convergência	260	0.001
um_ponto	0.30	heurística	convergência	260	0.001
dois_pontos	0.01	aleatório	convergência	260	0.001

4. Discussão

As configurações com inicialização heurística e critério de parada por convergência apresentaram os menores tempos de execução. O crossover de um ponto demonstrou ser eficiente, aparecendo em nove das dez melhores combinações.

Taxas de mutação muito baixas (0.01) ou muito altas (0.30) não comprometeram o valor final, mas impactaram o tempo, sugerindo que uma taxa intermediária (0.10) oferece melhor equilíbrio. A inicialização heurística proporcionou uma população inicial mais promissora, acelerando a convergência.

5. Conclusão

A análise dos resultados confirma que a escolha dos operadores genéticos influencia diretamente o desempenho do algoritmo. A combinação mais eficiente foi: crossover de um ponto, mutação de 10

Como trabalhos futuros, recomenda-se aplicar o AG a instâncias maiores ou outros problemas, como o Caixeiro Viajante. Também é recomendada a execução de múltiplas rodadas para obter análises estatísticas mais robustas e investigar variações híbridas com outras meta-heurísticas.

Referências

Referências

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

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

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

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

[5] Alba, E., Troya, J. M. A survey of parallel distributed genetic algorithms. *Complexity*, 4(4), pp.31–52, 1999.