Implementação de um algoritmo genético para o Problema da Mochila

Murielly Oliveira Nascimento

11 de setembro, 2023

O Problema da Mochila é definido da seguinte forma:

Dado um conjunto de itens, cada item com um peso e valor associados a ele. O problema da mochila consiste em encontrar o conjunto de itens tal que o peso total seja menor ou igual a um determinado limite (tamanho da mochila) e o valor total obtido seja o maior possível. Como restrição, os itens não podem ser "quebrados", ou seja, a decisão consiste apenas em inserir ou não inserir um item na mochila.

Existem diversas maneiras de solucionar este problema, neste trabalho aquela por Algoritmos Genéticos é discutida. Ela faz parte da área de pesquisa Computação Bioinspirada e foi desenvolvida por (HOLLAND, 1975) e seus alunos na Universidade de Michigan em Ann Arbor.

Algoritmos Genéticos são inspirados no princípio Darwiniano da evolução das espécies e na genética (GOLDBERG, 1989). São aplicados, principalmente, na busca de soluções ótimas para problemas combinatórios, cujas técnicas tradicionais são ineficientes. Seu funcionamento básico é ilustrado no Algoritmo 1.

Algoritmo 1 Pseudocódigo do Algoritmo Genético.

Gerar população inicial.

Avaliar população segundo função objetivo.

Enquanto número de gerações não atingido.

    Selecionar indivíduos para reprodução.

    Realizar cruzamento entre os indivíduos.

    Realizar mutação em alguns indivíduos.

    Inserir filhos na população.

    Avaliar população segundo função objetivo.

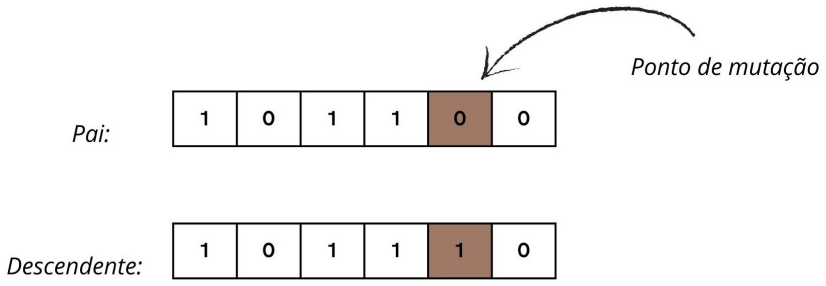
1 Implementação

Para a solução do Problema da Mochila, o indivíduo no Algoritmo Genético é definido como uma classe contendo os atributos vetor binário, para representar os itens adicionados (1) ou não (0) a mochila; *fitness*, para avaliar quão adaptado o indivíduo é; e peso atual da mochila. Além dele, as classes, Mochila com os atributos tamanho e capacidade; e Item com os atributos id, valor e peso; são usadas.

A população de soluções é inicializada aleatoriamente, com a condição que uma vez atingido o peso máximo nenhum item é adicionado a mochila. O programa lê as entradas (tamanho da mochila, capacidade e sequência de itens) de um arquivo texto. O *fitness* é calculado como a somatória do valor de cada item adicionado a mochila, sendo que estes são ordenados do mais pesado para o menos.

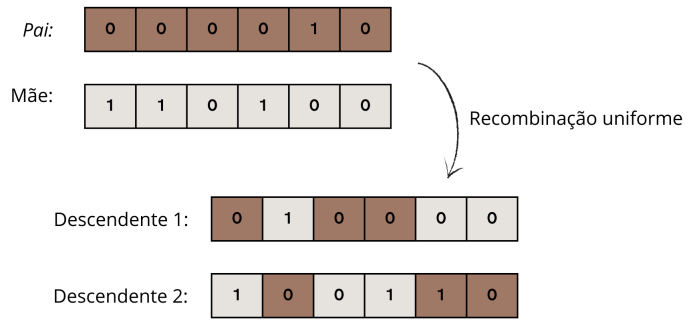
A mutação segue a implementação tradicional: o operador de mutação padrão simplesmente troca o valor de um gene em um cromossomo(GOLDBERG, 1989), como a representação dos cromossomos, nesse caso, é um vetor de 0s e 1s, o operador troca esses valores, como ilustrado na Figura 1. Com a diferença que, para este problema, é avaliado o peso atual do indivíduo (solução), caso seja maior do que a capacidade da mochila um item é retirado, do contrário outro é adicionado.

Figura 1 – Operador de Mutação.



A recombinação usada é a uniforme, na qual é calculado a probabilidade de cada gene ser trocado entre os pais. A Figura 2 descreve o funcionamento desse operador. Quanto a forma de seleção, o método do torneio foi usado. Nele N indivíduos são escolhidos aleatoriamente da população e aquele com melhor *fitness* é selecionado.

Figura 2 – Recombinação Uniforme.



2 Experimentos

A Tabela 1 ilustra os resultados do Algoritmo Genético, com os parâmetros: população (50), gerações (25), mutação (15) e torneio (3); para 16 experimentos, o tempo gasto para a execução do programa foi de 178,06 segundos. Por se tratar de um Algoritmo empírico, os seus resultados estão fortemente atrelados aos parâmetros e como foi implementado, podendo apresentar melhoras ou pioras caso um desses fatores seja alterado. Os valores de entrada usados para os testes são descritos na tabela.

Tabela 1 – População (50); número de gerações (25); taxa de mutação(15%); torneio (3).

Resultados dos Testes com o Algoritmo Evolutivo.		
Arquivo de Entrada	Tamanho da Mochila	*Fitness*
1º	20	26107
2º	40	56743
3º	50	90512
4º	50	21347
5º	100	6079
6º	200	47103
7º	300	424
8º	400	314
9º	500	1713
10º	100	2638
11º	100	15728
12º	10000	49
13º	5000	233
14º	5000	0
15º	1000	0
16º	1000	14310

A Tabela 2, por sua vez, mostra os resultados obtidos com um Algoritmo de Programação Dinâmica.

Tabela 2 – Algoritmo de Programação Dinâmica.

Resultados dos Testes com o Algoritmo de Programação Dinâmica.		
Arquivo de Entrada	Tamanho da Mochila	*Fitness*
1º	20	31621
2º	40	67829
3º	50	143449
4º	50	28840
5º	100	15785
6º	200	99861
7º	300	1940
8º	400	741
9º	500	10281
10º	100	20149
11º	100	30001

Os arquivos de entrada são os mesmos usados anteriormente. Contudo, a partir da entrada 11º o algoritmo atinge o limite de recursões permitidas pela linguagem *python*. Este limite pode ser verificado com a função *sys.getrecursionlimit()* e, o mesmo, visa impedir que uma função execute indefinidamente (PYTHON SOFTWARE FOUNDATION, 2001).

Embora, o Algoritmo de Programação Dinâmica apresente essa desvantagem, ele traz resultados relativamente melhores do que o Algoritmo Genético. Este, para a 1ª entrada, obteve um *fitness* 26107, enquanto aquele, de 31621. O mesmo cenário se repete para outras entradas.

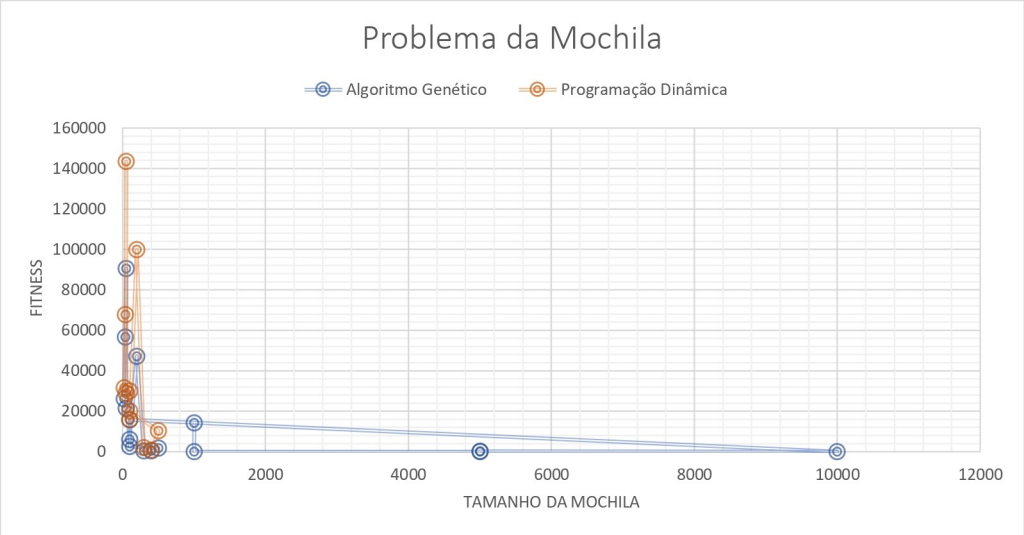
A Tabela 3 mostra as medidas de avaliação (média ponderada e desvio padrão) dos Algoritmos Genético e Programação Dinâmica. O primeiro apresentou uma média ponderada de 1.564 para o *fitness* enquanto o segundo 23.948,73. Em compensação, o desvio padrão do Algoritmo Genético é relativamente menor ao da Programação Dinâmica.

Tabela 3 – Medidas de Avaliação.

Medidas de avaliação		
Algoritmo	Média Ponderada	Desvio Padrão
Algoritmo Genético	1.564,301	26.142,3
Programação Dinâmica	23.948,73	44.941,83

A Figura 3 ilustra o desempenho de ambos os algoritmos considerando o tamanho da mochila.

Figura 3 – Comparação do Algoritmo Genético e Programação Dinâmica para o Problema da Mochila.



A partir do gráfico é possível observar, com mais clareza, as diferenças de desempenho entre os métodos. Enquanto a Programação Dinâmica mostra bons resultados para entradas menores, rapidamente ela se torna ineficiente com mochilas de tamanho acima de 1000.

Logo, o algoritmo de Programação Dinâmica é útil em cenários cujas entradas são menores, enquanto o Algoritmo Genético é melhor para entradas maiores. Sendo passível de melhoras dado ao impacto dos parâmetros (mutação, recombinação, taxa de elitismo) nos seus resultados.

Referências

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning.* New York, NY: Addison-Wesley, 1989. Citado 2 vezes nas páginas 1 e 2.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems.* MA, USA: MIT Press, 1975. Citado na página 1.

PYTHON SOFTWARE FOUNDATION. *Python Doccumentation.* [S.l.], 2001. Disponível em: <https://docs.python.org/3/about.html>. Citado na página 4.