Implementação e Análise de um Algoritmo Genético para o Problema da Mochila

Rony Soares de Freitas
Universidade do Estado de Minas Gerais
Pará de Minas, Brasil
ronyfreitas98@gmail.com

Vitor Soares Silva
Universidade do Estado de Minas Gerais
Divinópolis, Brasil
vitrola.vitor@gmail.com

***Resumo*—Este trabalho prático tem como objetivo a implementação e análise de um algoritmo genético para solucionar um problema de Inteligência Artificial, o Problema da Mochila.**

***Index Terms*—Sistemas Inteligentes, Algoritmo Genético, Algoritmos, Inteligência Artificial, Problema da Mochila.**

I. Introdução

A otimização vem ganhando muita relevância nas últimas décadas, sendo alvo de muitas pesquisas devido à sua importância operacional e, sobretudo, econômica.

Dentre os problemas de otimização vale ressaltar o Problema da Mochila, que se trata de é um problema de otimização combinatória.

Também conhecido como "knapsack problem", o Problema da Mochila está relacionado com um grande número de outros modelos de programação. Sua importância está associada exatamente a esse fato.

Neste trabalho implementamos e analisamos o Problema da Mochila tendo como base um Algoritmo Genético.

II. Algoritmo Genético

Algoritmo Genético (AG) é uma técnica de busca utilizada na computação para encontrar soluções aproximadas de um determinado problema de busca e otimização. O principal nome por trás dos AG's é o americano John Henry Holland.

Segundo (LACERDA, 1999), Algoritmos Genéticos são métodos de otimização e busca inspirados nos mecanismos de evolução de populações de seres vivos.

Estes algoritmos são uma simulação de computador baseada na biologia evolutiva de modo a utilizar algumas de suas técnicas como hereditariedade, mutação, seleção natural e recombinação.

A. Mutação

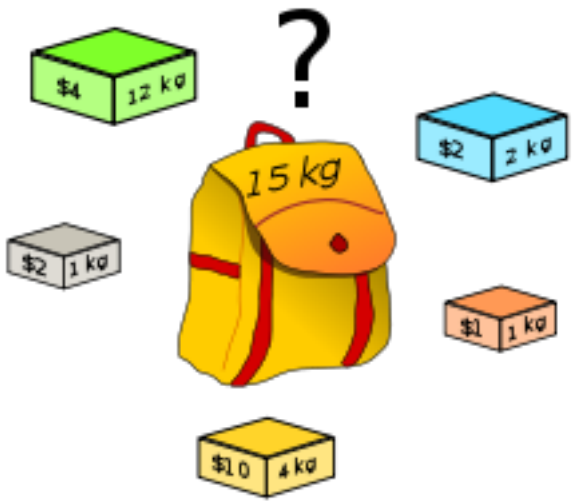
O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida fornecendo assim meios para introdução de novos elementos na população.

B. Crossover

É um operador genético usado para variar a programação de um cromossomo de uma geração para a próxima. O crossover é um processo de se obter mais de uma solução progenitora e produzir uma solução descendente a partir deles.

III. Problema da Mochila

O problema da mochila é um problema de otimização combinatória. O nome dá-se devido ao modelo de uma situação em que é necessário preencher uma mochila com objetos de diferentes pesos e valores.



Este é um problema estudado e aplicados a séculos (desde 1897) pelas aplicações práticas que pode ser implementado, como: investimento de capital, corte e empacotamento, carregamento de veículos, orçamento.

A. Definindo o Problema

Segundo (MARQUES, 2000), o Problema da Mochila é definido do seguinte modo: Suponha que um alpinista deva carregar sua mochila selecionando itens, dentre vários disponíveis, e considerando a capacidade da mochila. A cada item é atribuído um valor de utilidade e o alpinista deve selecioná-los buscando maximizar o valor de utilidade total.

Metaforicamente podemos entendê-lo como o desafio de encher uma mochila sem ultrapassar um determinado limite de peso, otimizando o valor do produto (benefício) carregado.

IV. Implementação do Problema

Os parâmetros padrões para o teste do algoritmo genético são os seguintes:

- Peso máximo da mochila: 20kg

- Quantidade de itens disponíveis: 29 itens
- Tamanho da população: 20 indivíduos
- Taxa de Crossover: 100%
- Taxa de Mutação: 2%
- Parâmetro de parada: 100 interações

Para analisar o funcionamento e performance do algoritmo genético, foi alterado, ao longo dos testes, os valores: tamanho da população, taxa de crossover, taxa de mutação e parâmetro de parada.

A. Ambiente e Recursos

A resolução para o problema foi desenvolvida, com base em (VALMIROZUNO, 2018), utilizando a linguagem de programação C++ pelo IDE Visual Studio Code e foi executado para testes e obtenção de resultados em um notebook Dell, S.O. Ubuntu 18.04, processador Intel CORE i7, 8GB de memória RAM.

V. Resultados e Análise

Foram elaborados e processados sete testes com diferentes parâmetros, executados cinco vezes cada:

A. 1º Teste

O primeiro teste foi executado com os seguintes parâmetros: 100 interações, taxa de crossover 100%, taxa de mutação 2%, população de 20 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	19,48	127	3,086
2	16,70	119	3,913
3	19,60	117	2,845
4	18,88	123	3,506
5	19,78	124	2,028

Figura 1. Teste 1. Tempo em milissegundos.

Podemos ver que o teste gera um peso e benefício um pouco instável, com o benefício em torno de 117 e 127.

B. 2º Teste

O segundo teste foi executado com os seguintes parâmetros: 100 interações, taxa de crossover 100%, taxa de mutação 2%, população de 5 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	19,15	119	0,404
2	19,18	98	0,658
3	18	108	1,335
4	18,2	109	0,948
5	19,28	117	1,38

Figura 2. Teste 2. Tempo em milissegundos.

Analisamos que o tempo gasto em relação ao teste um foi menor, mas o benefício ficou abaixo.

C. 3º Teste

O terceiro teste foi executado com os seguintes parâmetros: 500 interações, taxa de crossover 100%, taxa de mutação 2%, população de 20 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	19,7	132	10,723
2	19,4	138	14,579
3	19,63	130	11,364
4	17,5	135	9,345
5	19,28	135	11,992

Figura 3. Teste 3. Tempo em milissegundos.

Aumentando as interações do algoritmo vemos que o benefício melhora, contundo o tempo gasto aumenta significativamente.

D. 4º Teste

O quarto teste foi executado com os seguintes parâmetros: 500 interações, taxa de crossover 100%, taxa de mutação 2%, população de 5 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	19,73	121	3,412
2	15,68	121	3,952
3	18,05	119	3,316
4	18,8	114	2,555
5	17,8	131	5,446

Figura 4. Teste 4. Tempo em milissegundos.

Com o teste quatro, analisamos que o tamanho da população influência diretamente no benefício, sendo melhor quanto maior a população, e no tempo gasto, maior ao aumentar a população.

E. 5º Teste

O quinto teste foi executado apenas para confirmar a influência da quantidade de interações do algoritmo na resolução, com os seguintes parâmetros: 100.000 interações, taxa de crossover 100%, taxa de mutação 2%, população de 20 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	19,68	139	1,52609
2	19,98	140	1,51534
3	19,90	138	1,52103
4	19,90	141	1,50904
5	19,90	141	1,51312

Figura 5. Teste 5. Tempo em segundos.

Confirmamos agora que aumentando as interações do algoritmo o tempo gasto aumenta, agora em segundos, mas obtém um benefício melhor.

Observação: 141 foi o melhor benefício encontrado em todos os testes.

F. 6º Teste

Os testes seis e sete foram realizados para analisar a influência da taxa de mutação e taxa de crossover, respectivamente. Foi usado como base para comparação o teste um.

O sexto teste foi executado com os seguintes parâmetros: 100 interações, taxa de crossover 100%, taxa de mutação 50%, população de 20 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	18,73	122	4,139
2	18,28	117	3,942
3	20,00	113	4,033
4	19,43	116	3,752
5	17,98	113	2,684

Figura 6. Teste 6. Tempo em milissegundos.

Apesar de pior, o resultado foi melhor do que o esperado. Isso pois uma baixa taxa de mutação previne que uma dada posição fique estagnada em um valor, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Com uma taxa muito alta a busca se torna essencialmente aleatória.

G. 7º Teste

Conforme dito, o sétimo teste foi realizado para analisar a influência da taxa de crossover e foi executado com os seguintes parâmetros: 100 interações, taxa de crossover 10%, taxa de mutação 2%, população de 20 indivíduos.

Interação	Peso	Benefício	Tempo Gasto
1	16,33	102	1,926
2	19,48	104	2,666
3	19,78	107	2,593
4	18,43	104	2,25
5	19,75	101	2,75

Figura 7. Teste 7. Tempo em milissegundos.

Percebemos uma piora nos resultados, isso acontece pois o algoritmo genético é baseado na teoria da evolução, e nela para que haja variabilidade genética é necessário gerar filhos e para que isso aconteça no algoritmo genético é preciso crossover para gerar novos indivíduos.

VI. Considerações Finais

Esse trabalho abordou o Problema da Mochila, um problema de importantes aplicações práticas.

Nesse estudo, propusemos o problema sobre um Algoritmo Genético, implementado em C++, que nos proporcionou uma dificuldade média mas que por fim representou o problema de maneira objetiva.

Além disso, tratamos os parâmetros do Algoritmo Genético, como taxa de mutação e taxa de crossover, afim de analisar suas respectivas funcionalidades e influência nos resultados.

O trabalho alinhou nosso aprendizado teórico, visto em sala de aula, com desenvolvimento práticos de programação e análise.

O código implementado pode ser encontrado nas contas do GitHub de cada autor.

Referências

LACERDA, A. Estéfane G. M. de. Introdução aos algoritmos genêticos. *Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais*, v. 1, p. 99–148, 1999.

MARQUES, F. d. P. *O problema da mochila compartimentada*. Tese (Doutorado) — Universidade de São Paulo, 2000.

VALMIROZUNO. *Problema da mochila resolvido utilizando a estratégia de algoritmo genético na linguagem C++*. 2018. <https://github.com/valmirozuno/Mochila_Genetico>. [17 jun. 2019].