Problema da Mochila Binária

Eliton M. Zimmermann[1], Gabriel Filippi[1], Lucas F. da Costa[1]

[1] Universidade da Região de Joinville (Univille) Caixa Postal 246 – 89201-972 – Joinville – SC – Brazil

{eliton.zimmermann, gabriel.filippi, lucas.costa.1}@univille.br

Resumo. *Esse artigo descreve o experimento do algoritmo populacional evolucionário para resolver o problema da mochila binária..*

1. Problema da Mochila Binária

O experimento utilizando o algoritmo populacional evolucionário, foi usado para solucionar o problema da mochila binária, onde dado uma mochila, é necessário ter a melhor solução para aumentar o lucro.

Foi utilizado duas intâncias de um conjunto de benchmarks, onde para cada instância deve ser realizado testes, onde são feitos os seguintes passos: executar 30 vezes, calcular a média e o desvio padrão. Dessa forma, calculando a melhor solução.

2. Algoritmo Populacional

Os algoritmos geracacionais utilizam um conjunto de soluções para determinado problema. Onde um conjutnto de soluções é chamado de população. Uma geração é composta por um conjunto de soluções definido em um período de tempo. Esses algoritmos utilizam G gerações para modificas as soluções.

Uma solução é um vetor que é chamado de indivíduo onde possui o tamanho M, sendo que esse conjunto de soluções é uma uma matriz NxM (N é a quantidade de soluções e M é o tamanho de cada solução), onde cada linha é uma matriz é uma solução candidata.

Cada solução recebe um valor de "adequação", que define o grau em que a solução resolve o problema. O valor calculado pela função objetivo pode ser usado. Normalmente, uma matriz 1xN (matriz coluna) é usada, onde N é o número de soluções candidatas.

Então podemos usar o elitismo para gerar uma solução melhor, onde esta nova geração se inclua na próxima geração.O foco dessa tecnologia é evitar a perda da melhor solução. A mutação é outra técnica porem esta consiste em modificar um indivíduo.

E a reprodução, onde a estratégia do algoritmo é uma nova reprodução Os indivíduos podem realizar mutações e cruzar entre outros operadores. Essas operações podem envolver uma ou mais pessoas. Indivíduos recém-gerados são adicionados à nova geração e todos os indivíduos da geração anterior são descartados.

Para encerrar, quando não é mais capaz de gerar novas soluções ao longo de N gerações, e quando atinge X avaliações de indivíduos, o processo de busca é concluído.

3. Descrição dos Algoritmos

Para resolver este problema, temos ter como entrada parâmetros lucro dos objetos, peso dos objetos, tamando da mochila e a penalidade.

A penalidade serve para quando uma solução ultrapassa o tamanho da mochila. Para isso definimos que: Solução inválida - Fitness = 0, Penalizo a solução = Descontar o valor do fitness ou Corrijo a solução - Retirar algum item da mochila.

Assim, podemos desenvolveruma função onde calculanos o quão lucrativo é a mochila. Utilizadmos 1 para itens na mochila e 0 para itens que não estão na mochila. Sendo assim, multilicamos o 1 ou 0 pelo valor do item. E executamos o algoritmo de gerações. Inicialmente geramos a população com uma distribuição uniforme. Realizamos uma avaliação da população e da solução, utilizando a função objetivo.

Começamos um loop enquanto não descobrimos nosso critério de parada. Enquanto esse critério não é ãlcançado, realizamos o algoritmo elitismo e para cada item da lista que simboliza o tamanho da população realizamos a mutação naquele item. Descobrimos o fitness da próxima população e incrementamos a quantidade de avaliações realizadas. Ao terminar o loop do tamanho da população, geramos uma nova população. Ao terminar todos os loops, somos capazes de identificar a melhor solução e o fitness.

4. Resultados

Ao executar 30 vezes o algoritmo para dois casos de testes com os seguintes dados:

Parâmetro	Valor
Penalidade	25
Tamanho da População	4
Quantidade total de Avaliações	20
Percentual p/ performar uma mutação total de Avaliações	3

Os benchmarks utilizados podem ser acessados através desse link: https://people.sc.fsu.edu/ jburkardt/datasets/knapsack_01/knapsack_01.html.

um dos algoritmos, recebemos os seguintes resultados a respeito da média aritmética, desvio padrão e diagrama de caixa.

Algoritmo	Média (Distância)	Desvio Padrão (Distância)
Aleatorio	3138.97 KM	1415.68
Guloso	409.97 KM	332.13
H´ıbrido	1041.82 KM	1042.17

5. Conclusões

Com os resultados dos testes é possível concluir que um item quase se iguala aos algoritmos. Podemos comparar isso com a quantidade de dados processados, pois não inserimos os algoritmos em teste de stress, para que seja feita a validação de qual algoritmo demoraria mais para encontrar a solução em grande quantidade de dados.

Por padrão, o algoritmo aleatório busca ser o que pode achar uma solução mais rápida devido a sua lógica. Analisando cada algoritmo com os dados apresentados, podemos dizer que:

Em todas as situações o algoritmo aleatório se mostrou o pior para determinar a rota mais curta. Onde a sua média e desvio padrão ultrapassaram a soma dos respectivos valores dos outros dois algoritmos. No diagrama de caixa, é o que possui os maiores outliers e também o que possui maior dispersão dos seus pontos, sendo assim ”a maior caixa”. Podemos verificar que o algoritmo aleatório gera soluções capazes de se igualar ao algoritmo guloso e híbrido e também acaba por gerar soluções que ultrapassam os valores, tornando-se a pior solução.

O algoritmo guloso, é o que gerou as menores rotas, mesmo levando em consideração o risco dele ter problema nos ”ótimos locais”. Dentre os três foi o que gerou as melhores soluções, tanto analisando os seus valores de média e desvio, quando ao diagrama de caixa, no qual podemos visualizar que grande parte de suas soluções se encontram em valores próximos. Tendo a solução mais longa equivalendo a média das soluções do algoritmo híbrido.

O algoritmo híbrido mostrou ser o mais estabilizado dentre os três quando analisamos o diagrama de caixa. Se analisarmos a sua média e desvio padrão, ele apresenta se aproximar do algoritmo aleatório, o que nos indicaria que não compensa utiliza-lo, porém olhando para o diagrama de caixa, podemos visualizar que o que causou esses valores elevados nas médias foram seus outliers. Avaliando que 80% dos casos não entraram nesta condição, ele acaba se tornando um algoritmo com soluções bem equilibradas e capaz de contornar o problema de ótimos locais do algoritmo guloso.

Podemos concluir assim que neste experimento, levando em consideração os dados usados, o melhor algoritmo seria o guloso, porém, é necessário frisar de que, é possível que acontecem problemas de ótimos locais, o que podem gerar outliers para nós. Podemos presumir que, nesta situação o algoritmo guloso se apresentou a melhor solução pela quantidade de dados que o algoritmo precisa consumir, e o número de tentativas, pois em teoria o algoritmo hibrido que combina os outros dois tendem a ser uma boa solução quando temos uma grande quantidade de dados.