Population Algorithm - Problema da Mochila Binária

Ana C. Knop[1], **Carlos S. Mondo**[1], **Leandro M. da Silva**[1]

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

{ana.knop, carlosm, leandrosilva.1}@univille.br

Resumo. *Experimento com o algoritmo populacional evolucionário para resolver o problema da mochila binária.*

1. Problema da Mochila Binária

Este experimento com o algoritmo populacional evolucionário foi utilizado para resolver o problema da mochila binária, no qual dado uma mochila, qual a melhor solução a fim de maximizar o lucro.

Utilizando duas instâncias diferentes de um conjunto de benchmarks, devem ser realizados testes para cada instância no qual consistem em: executar 30 vezes, calcular a média e o desvio padrão. Com isso, calcular a melhor solução.

2. Algoritmo Populacional

São algoritmos geracionais que utilizam um conjunto de soluções candidatas para o determinado problema. No qual seu conjunto de soluções é chamado de população.

Uma geração representa um conjunto de soluções em um determinado período de tempo. Os algoritmos utilizam G gerações para manipular as soluções.

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

Para cada solução é atribuído um valor de "fitness", este valor de fitness ou aptidão define o qual bem a solução resolve o problema. Pode ser utilizado o valor calculado por uma função objetivo. Tipicamente é utilizado uma matriz 1xN (matriz coluna), onde N é a quantidade de soluções candidatas.

Assim podemos utilizar do elitismo para gerar uma solução melhor, no qual esta nova geração é incluída na próxima geração, o foco desta técnica é evitar perder a melhor solução encontrada durante o processo.

A mutação é outra técnica porem esta consiste em modificar um indivíduo.

E a reprodução, na qual a estratégia desse algoritmo é a reprodução de novos indivíduos pode ser realizada através de mutações, cruzamento entre outros operadores. Estas operações podem envolver 1 ou mais indivíduos. Novos indivíduos gerados são adicionados na nova geração, e todos os indivíduos da geração anterior são descartados.

Para finalizar o processo de busca quando não for mais capaz de gerar novas soluções por N gerações, quando atingir X avaliações de indivíduos entre outro.

3. Descrição dos Algoritmos

Para solucionar este problema, devemos ter como entrada de parâmetros lucro dos objetos, peso dos objetos, tamanho 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.

Com isso, podemos desenvolver uma função objetiva no qual calculamos o quão lucrativo é a mochila. Sendo 1 para itens na mochila e 0 para itens que não estão na mochila. Sendo assim multiplicamos o 1 ou 0 pelo valor do item.

E executamos o algoritmo de gerações. Inicialmente geramos a população com uma distribuição uniforme.

Fazemos uma avaliação da população e da solução usando a função objetivo.

Iniciamos um loop enquanto não atingimos nosso critério de parada. Enquanto este critério não é atingido, realizamos o algoritmo de elitismo e para cada item numa lista que representa 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

Executando 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 encontrados neste link: https://people.sc.fsu.edu/ jburkardt/datasets/knapsack_01/knapsack_01.html.

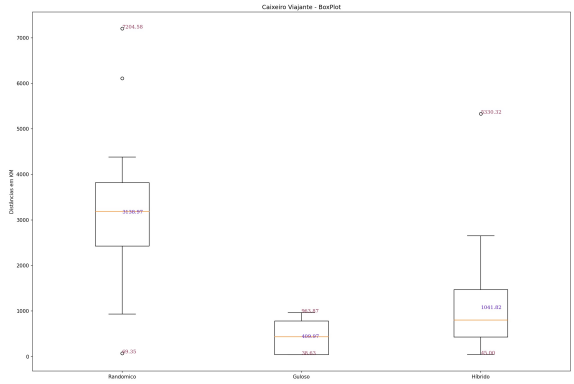
um dos algoritmos, obtemos os seguintes resultados a respeito da média aritmética, desvio padrão e diagrama de caixa.

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

5. Conclusões

Analisando os testes realizados, é possível concluir um item que quase se equivalente aos algoritmos: seu tempo de execução. Podemos relacionar isso com a quantidade de dados processados, pois não colocamos os algoritmos em testes de estresse, para validar qual algoritmo demoraria mais para encontrar sua solução com grandes quantidades de dados.

Figura 1. Diagrama de Caixa - Algoritmos



Por via de regra, o algoritmo aleatório tende a ser o que pode achar uma solução mais rápido devido a sua lógica, se pegar qualquer caminho indiferente da distância, porem isso não analisa o ponto mais relevante deste experimento que se trata de encontrar a melhor solução. Analisando cada algoritmo com os dados levantados, podemos constar que:

Em todas as situações o algoritmo aleatório se mostrou o pior para determinar a rota mais curta. Sua média e desvio padrão ultrapassa a soma dos respectivos valores dos outros dois algoritmos. Em relação ao 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 visualizar que o algoritmo aleatório gera soluções capazes de se igualar ao algoritmo guloso e híbrido porem também acaba por gerar soluções que extrapolam os valores, tornando-se a pior solução.

O algoritmo guloso, surpreendentemente é o que gerou as menores rotas, mesmo levando em consideração o risco dele ter problema nos ”ótimos locais”. Dos três algoritmos 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 aparentou ser o mais equilibrado dentre os três quando analisamos o diagrama de caixa. Se analisarmos a sua média e desvio padrão, ele aparenta se aproximar do algoritmo aleatório, o que nos indicaria que não compensa utiliza-lo, porem olhando para o diagrama de caixa, podemos visualizar que o que causou esses valores elevados nas médias foram seus outliers. Se considerarmos que mais de 80 porcento dos casos não entram nesta condição, ele acaba se tornando um algoritmo com soluções bem equilibradas e capaz de contornar o problema de ótimos locais do algoritmo guloso.

Como conclusão, neste experimento, levando em consideração os dados usados, o melhor algoritmo seria o guloso, porem, é necessário levantar a ressalva de que, é possível que acontecem problemas de ótimos locais, o que podem gerar outliers para nós. Podemos especular que, nesta situação o algoritmo guloso se apresentou a melhor solução pela quantidade de dados que o algoritmo precisa consumir, e o numero 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.