Universidade Federal de São João del-Rei
Algoritmo Bioinspirados
Algoritmo Genético para resolução do Problema da Mochila Binária

Felipe Francisco Rios de Melo

1. INTRODUÇÃO

Um algoritmo genético (AG) é uma técnica de busca utilizada para achar soluções aproximadas em problemas de otimização e busca. (usam técnicas inspiradas pela biologia evolutiva como hereditariedade, mutação, seleção natural e recombinação).

Este trabalho tem por objetivo a implementação e análise de um algoritmo genético para a resolução do problema da mochila binária.

O problema tem o seguinte enunciado:

Dada uma mochila de capacidade W (inteiro) e um conjunto de n itens com tamanho wi (inteiro) e valor ci associado a cada item i, queremos determinar quais itens devem ser colocados na mochila de modo a maximizar o valor total transportado, respeitando sua capacidade.

2. IMPLEMENTAÇÃO

A seguir, será mostrado um resumo das decisões tomadas para a implementação do Algoritmo Genético:

2.1. Estrutura de dados

Foi implementado uma classe nomeada AG, que é instanciada uma única vez, contendo os atributos e métodos necessários para inicialização e manipulação do algoritmo.

- Atributos: tamanho da população, número de gerações, taxa de mutação, probabilidade de vitória na seleção por torneio, número de bits de cada indivíduo, uso ou não a estratégia de elitismo;
- Métodos: gerar população, avaliar população, seleção por torneio, crossover, mutação e etc.

Além da classe AG, existe também a classe Indivíduo, usada para armazenar a solução de cada indivíduo para o problema da mochila, que é um vetor de binários, onde que cada elemento representa um objeto da instância, caso o objeto esteja

presente na mochila, terá o valor 1, caso contrário será 0. Existe também, uma variável para dizer se essa solução é viável ou não, bem como seu fitness na função objetivo.

Por fim, temos a classe Mochila, utilizada para armazenar informações sobre a instância do problema executada. As informações são: quantidade de objetos, capacidade da mochila, vetor de peso de cada objeto, vetor de utilidade de cada objeto e a solução ótima da instância.

2.2 População

A população foi gerada como uma matriz com n x m, onde n é o tamanho da população e m o número de bits. O valor de cada elemento, 0 ou 1, é obtido aleatoriamente.

2.3. Indivíduos

Obtida a população inicial, para cada indivíduo da população, instanciamos um objeto da classe *Individuo*, e setamos a solução gerada aleatoriamente pela matriz da população inicial.

2.4. Função Objetivo

Temos duas funções objetivos, a depender se o peso do conjunto de itens presentes na solução do indivíduo ultrapassou a capacidade da mochila ou não.

Se o peso da solução do indivíduo não ultrapassou a capacidade máxima aplicamos a seguinte função objetivo:

$$fitness = \sum_{\forall i \in X} v[i]$$

Caso contrário, penalizamos a solução do indivíduo por ter ultrapassado a capacidade, tendo a seguinte função objetivo:

$$fitness = \sum_{\forall i \in X} v[i] * (1 - (\sum_{\forall i \in X} p[i] - c)/c)$$

O valor resultante da função objetivo é armazenado no atributo fitness na instância da classe *Individuo,* referente ao indivíduo em questão.

2.5. Elitismo

Caso o algoritmo seja executado com o elitismo habilitado. Vai ser buscado em todos os indivíduos, àquele que obteve a melhor avaliação, e que possui uma solução viável ao problema proposto, e guarda-o para ser usado mais adiante.

Optou-se por restringir o elitismo ao indivíduo que possui solução viável e não ao indivíduo que possui a melhor solução geral, independente se é viável ou não, pois em algumas instâncias testadas, percebeu-se que mesmo com as penalizações para soluções inviáveis, esses soluções inviáveis estavam tendo um fitness maior que as soluções viáveis. Prejudicando desta forma, a convergência do algoritmo para uma solução melhor e viável. Elegendo o indivíduo com a melhor solução viável garantimos que a solução final do algoritmo será uma solução viável.

2.6. Seleção

A seleção dos indivíduos para o cruzamento é feito por torneio. No torneio, é sorteado dois indivíduos, e estes indivíduos têm seu fitness comparado. O indivíduo com melhor fitness têm uma maior chance de se consagrar vencedor do torneio. A chance de vitória é dado por um atributo *prob_vitoria* (estipulado na instanciação da classe AG). Na implementação foi usado como 90% como probabilidade de vitória ao indivíduo que contém o maior fitness.

Por questões de implementação, a partir do momento que um indivíduo vence um torneio, ele é removido da lista de indivíduos, para que nas próximas chamadas da função torneio() não corra o risco dele ser selecionado novamente.

2.7. Crossover

Tendo os pais, conseguimos realizar cruzamentos. No crossover, é necessário um ponto de corte, para que haja herança genética (herança de bits) por parte dos dois pais. Este ponto de corte é obtido pelo sorteio de um valor compreendido entre 0 e *nbits* (quantidade de itens do problema).

Tendo o ponto de corte, podemos gerar filhos. Para reposição de toda a população inicial, é gerado dois filhos por casal. O primeiro filho recebe, do *pai1*, os bits de 0 até o ponto de corte e o restante do *pai2*. Já o segundo filho recebe, do *pai2*, os bits de 0 até o ponto de corte e o restante do *pai1*.

Em seguida, adiciona-se os dois filhos à nova população, criando instâncias para cada um deles de classe *Individuo*.

2.8. Nova População

Este processo de torneios (seção 2.6) e crossovers (seção 2.7), é repetido até que a nova população tenha obtido o tamanho da população anterior.

Logo, a nova população está no tamanho fixado, porém, quando é executado com o método Elitismo, é necessário abrir um espaço na nova população para o melhor indivíduo. Então, optou-se por aleatoriamente, remover um indivíduo da nova população para inserir o indivíduo mais bem avaliado.

2.9. Mutação

Com a nova população em mãos, para cada indivíduo dela, sorteamos um inteiro entre 0 e 100, se este inteiro for menor ou igual a taxa de mutação (estipulada na instanciação da classe AG), o indivíduo sofrerá uma mutação.

A mutação provoca a inversão de um bit aleatório do código genético do indivíduo.

2.10. Gerações

O passo a passo da seção 2.4 até a 2.9 é repetido para todas as gerações. A quantidade de gerações é estipulado no momento da instanciação da classe AG.

Em cada geração, o melhor indivíduo é imprimido na tela, junto a sua solução para o problema da mochila e sua avaliação na função objetivo.

3. ANÁLISE DOS RESULTADOS

O funcionamento do algoritmo foi testado com 8 instâncias, fornecidas no seguinte site: https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/.

As instâncias de 1 à 6, tem a quantidade de objetos de 5 a 10 objetos, já as instâncias 7 e 8, são um pouco maiores, tendo uma quantidade de objetos de 15 e 24 respectivamente.

Para analisar a eficiência do algoritmo genético aplicado ao problema da mochila binária, foi realizado um experimento fatorial completo variando taxas de mutação, os valores de tamanho de população, a taxa de cruzamento, número de gerações e o uso ou não do método do elitismo, nos seguintes valores:

Taxa de mutação	1%	5%	10%
Taxa de cruzamento	60%	80%	100%
Tamanho da população	25	50	100
Número de gerações	25	50	100
Elitismo	True	False	

Para cada uma das 8 instâncias foi testada as 162 combinações possíveis dos parâmetros acima, e realizado 20 execuções do algoritmo, para obter um resultado consistente. Resultando num total de 25920 execuções, estas foram efetuadas de maneira automatizada pelo arquivo python *analysis.py*.

O único parâmetro que não foi variado foi a probabilidade de vitória do melhor indivíduo na seleção por torneio, fixado em 90%.

A identificação da combinação de parâmetros referente aos dados mostrados nesta análise estarão no padrão (instância - elitismo - *taxa de mutação - taxa de cruzamento - tamanho da população - quantidade de gerações*). Por exemplo: 03-True-10-80-100-100.

A seguinte tabela, contém as combinações, junto à média do fitness do melhor indivíduo das 20 execuções para cada instância. É, na tabela, mostrado somente as 11 melhores e piores combinações.

Instância 1 (10 objetos)	
Combinação	**Fitness Médio**
01-True-10-100-100-100	309.0
01-True-10-100-50-100	309.0
01-True-10-80-100-100	309.0
01-True-5-100-100-100	309.0
01-True-5-100-100-50	309.0
01-True-5-100-50-100	309.0
01-True-5-80-100-100	309.0
01-True-1-100-100-100	309.0
01-True-1-100-50-100	309.0
01-True-1-80-100-100	309.0
01-True-1-60-100-100	309.0
.	
.	
.	
01-False-1-100-25-25	233.1
01-False-1-60-25-100	232.7
01-False-5-80-25-25	230.05
01-False-1-100-25-100	229.1
01-False-5-60-25-100	229.05
01-False-10-100-25-50	226.75
01-False-1-80-25-100	222.7
01-False-1-80-25-50	222.1
01-False-5-100-25-25	219.95
01-False-10-100-25-25	219.7
01-False-1-60-25-25	211.35

Instância 2 (5 objetos)	
Combinação	**Fitness Médio**
02-False-10-100-100-25	51.0
02-False-10-80-100-100	51.0
02-False-5-100-100-25	51.0
02-False-5-60-100-50	51.0
02-False-5-60-100-25	51.0
02-False-1-100-100-100	51.0
02-False-1-100-100-50	51.0
02-False-1-80-100-50	51.0
02-False-1-80-100-25	51.0
02-True-10-100-100-100	51.0
02-True-10-100-100-50	51.0
.	
.	
.	
02-False-10-60-25-100	46.25
02-False-1-60-25-100	46.25
02-False-10-100-25-100	46.2
02-False-10-60-25-50	45.7
02-False-10-100-25-50	45.65
02-False-1-60-25-50	45.55
02-False-10-80-25-100	45.5
02-False-1-100-25-50	45.15
02-False-1-100-25-100	45.05
02-False-1-80-25-25	45.0
02-False-5-80-25-100	44.55

Instância 3 (6 objetos)	
Combinação	**Fitness Médio**
03-True-10-100-100-100	150.0
03-True-10-100-100-50	150.0
03-True-10-100-100-25	150.0
03-True-10-100-50-100	150.0
03-True-10-100-50-50	150.0
03-True-10-100-50-25	150.0
03-True-10-100-25-100	150.0
03-True-10-100-25-50	150.0
03-True-10-100-25-25	150.0
03-True-10-80-100-100	150.0
03-True-10-80-100-50	150.0
.	
.	
.	
03-False-5-80-25-50	129.2
03-False-10-100-25-25	129.15
03-False-1-80-25-25	128.8
03-False-5-100-25-50	128.6
03-False-1-60-25-25	128.1
03-False-10-80-25-100	127.3
03-False-10-100-25-50	126.75
03-False-10-80-25-50	125.85
03-False-5-100-25-100	125.0
03-False-1-100-25-100	121.15
03-False-1-60-25-50	121.15

Instância 4 (7 objetos)	
Combinação	**Fitness Médio**
04-True-10-100-100-100	107.0
04-True-10-100-100-50	107.0
04-True-10-100-100-25	107.0
04-True-10-100-50-100	107.0
04-True-10-100-50-50	107.0
04-True-10-80-100-100	107.0
04-True-10-80-100-50	107.0
04-True-10-80-100-25	107.0
04-True-10-80-50-100	107.0
04-True-10-80-50-50	107.0
04-True-10-60-100-100	107.0
.	
.	
.	
04-False-1-60-25-50	93.9
04-False-5-80-25-50	93.85
04-False-1-80-25-50	93.55
04-False-5-60-25-100	93.4
04-False-10-80-25-100	93.15
04-False-1-80-25-25	93.1
04-False-1-60-25-100	92.5
04-False-5-100-25-50	91.6
04-False-1-100-25-100	90.7
04-False-5-80-25-100	89.05
04-False-1-80-25-100	84.8

Instância 5 (8 objetos)		Instância 6 (7 objetos)		Instância 7 (15 objetos)		Instância 8 (24 objetos)	
Combinação	**Fitness Médio**	**Combinação**	**Fitness Médio**	**Combinação**	**Fitness Médio**	**Combinação**	**Fitness Médio**
05-True-10-100-100-100	900.0	06-True-10-100-100-100	1735.0	07-True-10-100-100-100	1456.75	08-True-1-80-100-100	13446099.5
05-True-10-100-100-50	900.0	06-True-10-100-100-50	1735.0	07-True-1-100-100-100	1456.7	08-True-1-100-100-100	13438050.65
05-True-10-100-100-25	900.0	06-True-10-100-100-25	1735.0	07-True-5-100-100-100	1456.6	08-True-10-100-50-100	13435254.45
05-True-10-100-50-100	900.0	06-True-10-100-50-100	1735.0	07-True-1-80-100-100	1456.5	08-True-5-100-100-100	13434231.6
05-True-10-100-50-50	900.0	06-True-10-100-50-50	1735.0	07-True-5-80-100-100	1455.35	08-True-10-100-100-100	13433002.05
05-True-10-100-50-25	900.0	06-True-10-100-50-25	1735.0	07-True-1-60-100-100	1455.25	08-True-5-80-100-100	13432004.3
05-True-10-100-25-100	900.0	06-True-10-100-25-100	1735.0	07-True-10-80-100-100	1455.05	08-True-5-100-50-100	13418372.55
05-True-10-80-100-100	900.0	06-True-10-100-25-50	1735.0	07-True-1-100-50-100	1454.55	08-True-10-80-50-100	13415658.15
05-True-10-80-100-50	900.0	06-True-10-100-25-25	1735.0	07-True-5-100-100-50	1454.5	08-True-1-100-50-100	13413326.4
05-True-10-80-50-100	900.0	06-True-10-80-100-100	1735.0	07-True-10-100-50-100	1454.45	08-True-5-80-50-100	13412097.7
05-True-10-80-50-50	900.0	06-True-10-80-100-50	1735.0	07-True-5-60-100-100	1454.25	08-True-1-60-100-100	13404321.0
.		.		.		.	
.		.		.		.	
.		.		.		.	
05-False-5-100-25-100	881.05	06-False-1-100-25-50	1666.95	07-False-1-100-25-100	1387.8	08-False-1-60-25-100	12573872.2
05-False-10-80-25-50	880.45	06-False-5-100-25-100	1666.8	07-False-10-60-25-100	1387.0	08-False-10-80-25-100	12560593.75
05-False-1-60-25-25	879.75	06-False-5-80-25-50	1665.7	07-False-1-100-25-25	1386.85	08-False-1-60-25-50	12558435.4
05-False-5-60-25-100	879.65	06-False-5-60-25-100	1664.7	07-False-10-100-25-100	1385.9	08-False-10-100-25-25	12539190.9
05-False-5-60-25-50	879.3	06-False-10-60-25-100	1663.1	07-False-10-100-25-50	1385.8	08-False-10-60-25-50	12533775.05
05-False-10-100-25-25	878.25	06-False-1-60-25-50	1659.95	07-False-10-60-25-50	1380.55	08-False-5-100-25-100	12528853.4
05-False-5-100-25-50	877.9	06-False-5-100-25-50	1657.85	07-False-1-60-25-25	1378.1	08-False-5-100-25-50	12520107.55
05-False-1-80-25-100	877.45	06-False-1-60-25-100	1655.1	07-False-1-80-25-100	1376.85	08-False-10-100-25-100	12509369.1
05-False-10-80-25-100	875.8	06-False-1-80-25-50	1652.65	07-False-1-100-25-50	1374.5	08-False-5-80-25-50	12480639.55
05-False-1-100-25-100	874.35	06-False-1-80-25-100	1634.0	07-False-1-60-25-50	1366.35	08-False-5-60-25-100	12479938.4
05-False-1-80-25-50	870.05	06-False-1-100-25-100	1591.45	07-False-5-100-25-100	1362.85	08-False-1-80-25-50	12437648.15

Tabela 1. Instâncias e suas respectivas médias de fitness do melhor indivíduo, ordenado pela combinação com melhor fitness médio.

Logo de início, observando o topo e o fim da tabela, vemos que as piores combinações são as não elitistas e as melhores são elitistas. As execuções com o método elitista convergem muito mais rápido, isso devido ao fato de que o melhor indivíduo de determinada geração vai perseverar pelas gerações até que exista um melhor do que ele. Logo, se o algoritmo ser executado com 20 gerações, e um indivíduo da geração 4 consegue o melhor fitness possível, caso não houvesse o elitismo, o indivíduo geraria dois filhos compartilhando apenas uma parte de seus genes (a chance dos filhos não serem tão bons quanto o pai é alta), e por fim seria removido. Esta condição descrita, desperdiçaria uma solução que tinha o potencial de ser a melhor. Assim, vemos a importância do elitismo na implementação do algoritmo genético.

A exceção em relação às combinações do topo da lista serem elitistas é a instância 2, porém tem uma explicação para isso. A instância 2, é composta por apenas 5 objetos, facilitando a convergência da solução tanto para soluções com elitismo quanto para sem elitismo por existir menor possibilidades de combinações de objetos dentro da mochila. E, como só foi mostrado os 11 melhores fitness na tabela, e os 9 primeiros são não elitistas, dá uma falsa impressão de que as combinações sem elitismos se sobressaíram, porém a grande maioria das combinações com elitismo e apenas estas 9 combinações sem elitismos conseguiram alcançar a solução ótima. As combinações sem elitismo só ficaram no topo da lista devido a uma mera aleatoriedade onde houve muitas combinações que encontraram o ótimo global.

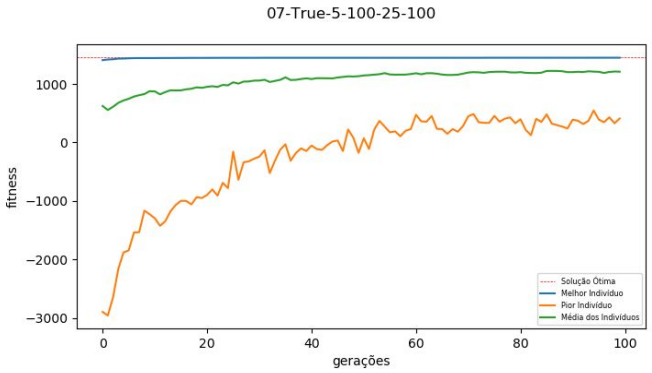

Gráfico 1. combinação com elitismo.

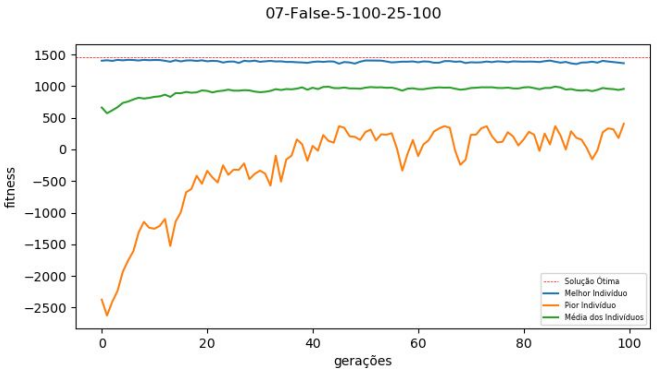

Gráfico 2. combinação sem elitismo.

Nos gráficos acima, conseguimos visualizar de maneira palpável o efeito do elitismo na busca pela solução ótima. A partir do momento que encontra um indivíduo com uma solução interessante, ele não sairá da população enquanto não houver outro melhor que ele, fazendo com que a linha do “Melhor indivíduo” no gráfico 1, nunca sofra uma queda. Enquanto que, no gráfico 2, notamos que a linha do “Melhor indivíduo” sofre várias perturbações, por não manter a melhor solução para a próxima geração.

Em relação à AG conseguir encontrar a otimalidade, primeiramente vamos listar qual é o valor da solução ótima de cada instância na função objetivo:

- instância 1: 309;
- instância 2: 51;
- instância 3: 150;
- instância 4: 107;
- instância 5: 900;
- instância 6: 1735;
- instância 7: 1458;
- instância 8: 13549094.

Observando a tabela 1, constatamos que apenas nas instâncias 7 e 8, nenhuma combinação de parâmetros obtiveram um fitness médio equivalente ao fitness da solução ótima, isso pode ser justificado pelo fato de que quanto maior o conjunto de entrada da instância, maior o número de combinações possíveis de objetos dentro e fora da mochila, o que teoricamente faz com que o algoritmo leve mais tempo para convergir para uma solução melhorada. Tanto é que, as combinações que obtiveram os melhores fitness são as com as com 100 gerações, e com a população relativamente grande, 50-100 indivíduos, pois quanto maior a quantidade de geração, mais iterações o algoritmo tem para aperfeiçoar a solução e quanto maior o tamanho da população, desde que seja uma população diversificada, mais soluções para serem avaliadas.

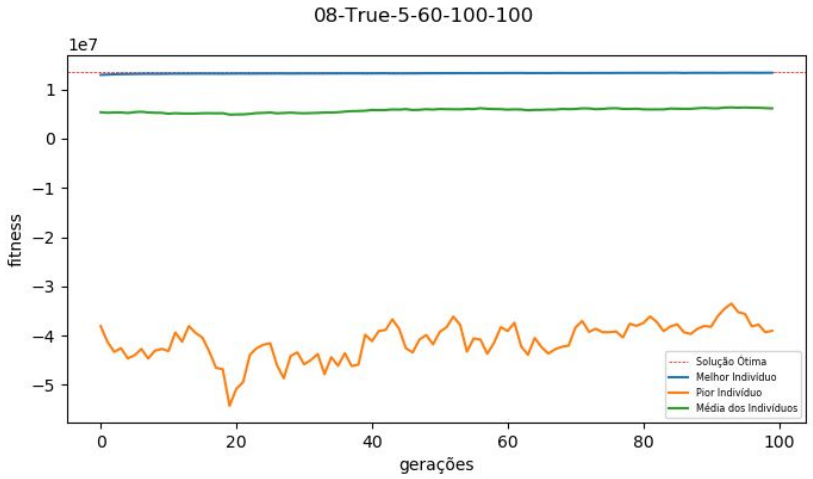

Gráfico 3. combinação da instância 8 com 100 gerações e 100 indivíduos

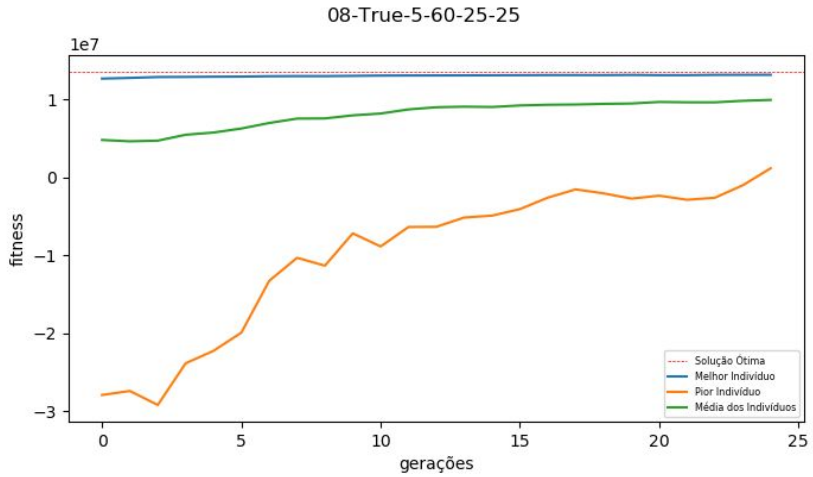

Gráfico 4. combinação da instância 8 com 25 gerações e 25 indivíduos

Nos gráficos acima, com a maior das instâncias, notamos que a linha do “Melhor indivíduo” se aproxima mais rápido e fica mais próximo da solução ótima na combinação com mais gerações e mais indivíduos, em comparação com a combinação com menos gerações e menos indivíduos.

Quanto às taxas dos operadores genéticos, cruzamento e mutação, não foi observado nenhuma mudança contrastante para ser analisada. Como pode ser visto nos gráficos abaixo.

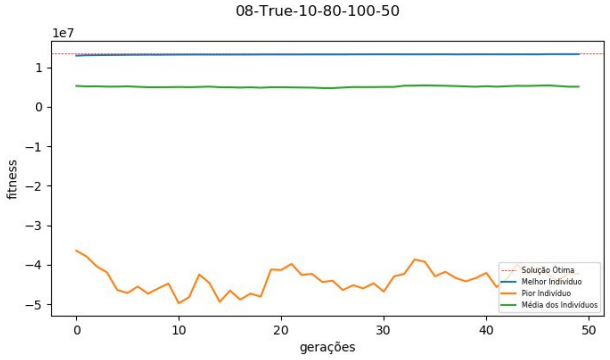

Gráfico 5. Combinação com taxa de mutação de 10%

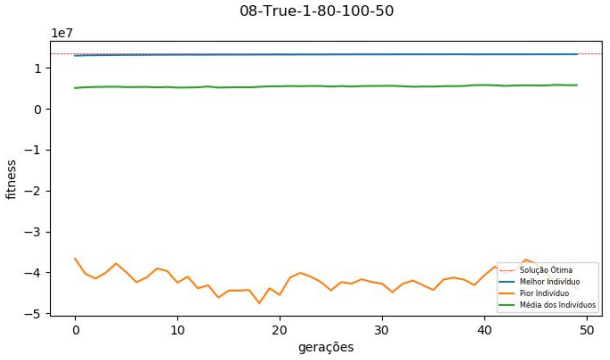

Gráfico 6. Combinação com taxa de mutação de 1%.

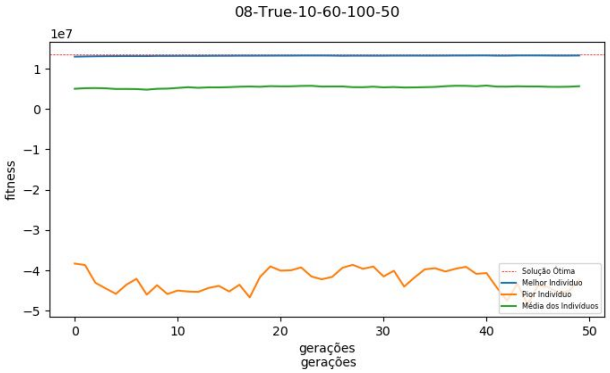

Gráfico 7. Combinação com taxa de cruzamento de 60%

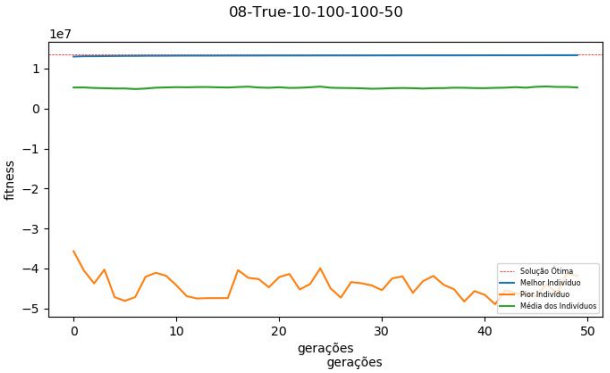

Gráfico 8. Combinação com taxa de cruzamento de 100%.

O que podemos tirar de conclusão das análises, é que os grandes responsáveis por encontrar uma solução ótima ou muito próxima da ótima, é o uso do método elitismo e a execução do algoritmo com um maior número de gerações e uma quantidade maior de indivíduos por população, para garantir maior quantidade de iterações para o algoritmo convergir para uma solução aceitável e maior combinações de possíveis soluções, respectivamente.

Porém quanto maior o número de gerações e a quantidade de indivíduos por população, maior o custo computacional para encontrar uma solução final.

4. CONCLUSÃO

Com este exercício de implementação foi possível aplicar o conceito de um algoritmo genético com em um problema conhecido na ciência da computação que é o problema da mochila binária.

E em relação à análise, algumas conclusões e sugestões em relação aos parâmetros pode ser feitas:

Em instâncias com uma menor quantidade de objetos, o algoritmo tem menor dificuldade de encontrar a solução ótima, não se fazendo necessário uma grande população e um grande número de gerações.

Já em instâncias maiores, talvez se tivermos conhecimento da solução ótima, podemos estabelecer um número maior de gerações e interromper o algoritmo quando encontrado a solução ótima, desta forma poupando custo computacional, ou então estabelecer um limite de tempo computacional gasto. Desta forma aumentando as chances do algoritmo encontrar a solução ótima ou uma tão boa quanto.

5. REFERÊNCIAS BIBLIOGŔAFICAS

O que são algoritmos genéticos? - StackOverflow - Disponível em: <https://pt.stackoverflow.com/questions/185996/o-que-s%C3%A3o-algoritmos-gen%C3%A9ticos>