Universidade Federal
de São João del-Rei

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ALGORITMOS BIOINSPIRADOS

Problema da Mochila Binária com Algoritmos Genéticos Combinatórios

Gustavo Henriques da Cunha

São João del-Rei
2024

Lista de Figuras

Sumário

1 INTRODUÇÃO

Este é um trabalho prático da disciplina de Algoritmos Bioinspirados no curso de Ciência da Computação na UFSJ.

1.1 Objetivo

Neste trabalho, temos como objetivo aprender a construir e testar algoritmos genéticos, buscando resolver o problema do caxeiro viajante com o uso de algoritmos genéticos combinatórios.

Além disso, focamos na análise da calibragem dos parâmetros, buscando melhorar a eficiência do algoritmo.

2 O PROBLEMA

O problema do caixeiro-viajante é um dos exemplos de problemas de otimização combinatória mais famosos que encontramos na computação. Por ser, em sua natureza, um problema difícil de resolver, pertencente à classe NP-Difícil, mas fácil de explicar e ilustrar, ele se tornou um problema comum para testarmos diferentes algoritmos de aproximação, nos quais buscamos encontrar uma solução boa em tempo polinomial.

Dado um grafo $G = (V, A)$ onde temos n vértices V que representam as cidades e arestas A que ligam essas cidades com certo peso que representa a distância entre elas, precisamos encontrar um circuito de menor distância possível que comece em uma cidade, passe por todas as outras uma vez e retorne para a cidade de origem.

Como já dito anteriormente, o PCV é NP-difícil; sendo assim, não conhecemos um algoritmo em tempo polinomial que encontre uma resposta ótima para o problema, e é por isso que abordagens aproximadas são populares alternativas para resolvê-lo.

3 ABORDAGEM DO PROBLEMA

Em nosso problema, armazenaremos o grafo em uma matriz, lendo os dados de um arquivo contendo a matriz com as distâncias das cidades. Note que, para esse problema, só consideramos grafos completos.

Para resolver o problema, utilizaremos de um algoritmo genético combinatório, onde a partir de uma população inicial, constituida de um vetor de cidades aleatório que forme um ciclo que passe por todas as cidades apenas uma vea, usaremos a distância percorrida no caminho como função de avaliação para calcularmos seu fitness, onde temos que quanto menor o fitness, melhor é o indivíduo.

Depois disso, passamos para a parte de seleção de pais, onde selecionamos indivíduos da população, preferencialmente os melhores, e os utilizamos para gerarem novos indivíduos. O método para seleção de pais utilizado foi o método do torneio e o da roleta, podendo o usuário escolher qual deles usar.

Com os pais escolhidos, vamos para a etapa de cruzamento, onde são gerados os indivíduos para a próxima geração. Nessa etapa, foi implementado o *ox-crossover* que guarda informações presentes em seus dois pais. Temos uma taxa de cruzamento que dita a probabilidade de um pai continuar para a próxima geraçãp.

Seguimos então para as etapas de mutação e elitismo. Para fazer a mutação, foi utilizado um método onde trocamos a posição de duas cidades no vetor da solução. A mutação tem uma taxa que diz se a cidade vai ser trocada ou não. Caso seja, a outra cidade será sorteada. Já no elitismo, simplesmente escolhemos os melhores indivíduos da população, onde nos testes, podemos ajustar esse número de selecionados.

Para o critério de parada foram usados tanto um número máximo de gerações como o número de gerações sem melhora da melhor solução. Esse número que de gerações sem melhora é calculado como dez porcento do número máximo de iterações.

4 ANÁLISE DE DESEMPENHO

O algoritmo genético gera resultados diversos conforme especificamos seus parâmetros. Assim, para verificar qual o melhor conjunto de parâmetros para conseguirmos o melhor resultado, devemos realizar diferentes testes com diferentes conjuntos de parâmetros e fazer uma análise estatística em cima dos resultados obtidos.

Assim, calibrando os parâmetros de taxa de mutação, taxa de cruzamento, tamanho da população, número de gerações, tipo de cruzamento e elitismo, foram feitos testes, principalmente com diferentes entradas, cada um sendo executado 10 vezes, sendo coletado o melhor indivíduo de cada execução, para podermos calcular a média e o desvio padrão

dos melhores resultados por conjunto de parâmetros, com o intuito de que quem tiver a maior média está consistentemente produzindo os melhores resultados.

Com isso, temos a tabela da figura 1, que mostra ordenadamente os resultados desses testes com o arquivo "lau15_dist", que possui solução ótima conhecida de 291 unidades.

Gerações	População	P. de Cruzamento	P. de Mutação	N Elite	T. Cruzamento	Media	Desvio
15000	40	1.0	0.001	4	Roulette	291.0	0.00
10000	30	1.0	0.001	2	Roulette	292.6	4.80
15000	30	1.0	0.001	2	Roulette	292.6	4.80
10000	40	1.0	0.01	0	Tournament	294.0	9.00
15000	40	1.0	0.01	4	Roulette	294.2	6.40
10000	40	0.8	0.01	0	Tournament	294.9	11.70
10000	30	1.0	0.01	4	Roulette	295.6	9.72
15000	40	0.8	0.01	4	Tournament	295.6	9.72
15000	40	1.0	0.001	2	Roulette	296.1	11.05
15000	30	0.6	0.01	0	Tournament	296.4	10.66
15000	30	0.8	0.01	2	Tournament	296.4	10.88
5000	20	1.0	0.01	4	Roulette	297.2	10.14
15000	40	1.0	0.01	2	Tournament	297.2	10.14
5000	40	0.8	0.01	2	Tournament	297.2	10.14
5000	40	1.0	0.01	2	Tournament	297.2	10.14
15000	40	1.0	0.001	2	Tournament	297.5	19.50
10000	20	0.8	0.01	0	Tournament	297.6	13.27
15000	30	1.0	0.01	4	Tournament	297.7	11.35
15000	40	1.0	0.01	2	Roulette	298.2	11.74
10000	20	1.0	0.001	2	Roulette	298.6	12.16
15000	30	1.0	0.01	2	Tournament	298.6	12.16
15000	30	1.0	0.01	4	Roulette	298.6	12.16
10000	40	1.0	0.0001	2	Roulette	298.6	13.68
15000	40	0.6	0.001	4	Roulette	298.6	12.16
15000	30	0.8	0.001	4	Roulette	298.8	10.29
10000	40	1.0	0.01	4	Roulette	298.8	10.29
10000	40	0.6	0.01	0	Tournament	299.0	8.00
15000	30	0.6	0.001	0	Tournament	299.1	13.13
5000	30	0.8	0.001	2	Tournament	299.6	17.21
10000	20	1.0	0.01	2	Tournament	300.0	13.12

Figura 1: tabela com comparação entre os testes lau15-dist

Podemos ver que o algoritmo conseguiu bons resultados com uma população pequena, mas com muitas iterações. Foram feitos testes com entradas maiores, mas os resultados não pareciam bons, e como o algoritmo não é muito rápido, o número de indivíduos na população faz enorme diferença em seu desempenho, então ao usar uma população pequena e um número grande de gerações, o algoritmo consegue explorar várias soluções com um desempenho aceitável.

Outro fator que podemos notar é o número da taxa de mutação, que funcionou bem com valores bem pequenos, que foram o serviram para manter as soluções melhorando, e o sufucuente para fugir de ótimos locais. O número alto de gerações também combina com a mutação baixa, pois uma mutação pode demorar a ocorrer.

Também temos que o tipo de seleção de torneio, apesar de nesse conjunto de testes não performar melhor que a seleção de roleta, ainda assim consegui bons resultados, e podemos notar que nesse tipo de seleção uma mutação um pouco maior que na roleta é

preferido.

Com os resultados dessa entrada menor, utilizamos como base para testar a entrada "dantizig42_d"que possui 42 cidades e solução ótima conhecida de 699 unidades, e montamos a tabela da figura 2.

Gerações	População	P. de Cruzamento	P. de Mutação	N Elite	T. Cruzamento	Media	Desvio
100000	40	1.0	0.0001	4	Roulette	784.3	50.09
100000	40	1.0	0.001	4	Roulette	788.4	54.23
100000	50	1.0	0.0001	4	Roulette	807.4	62.70
100000	40	1.0	0.0001	2	Roulette	810.5	68.67
100000	40	1.0	0.01	4	Tournament	821.2	38.24
100000	20	1.0	0.001	4	Roulette	829.2	64.65
100000	40	1.0	0.001	4	Tournament	834.7	38.77
100000	40	0.8	0.001	0	Tournament	853.8	42.44
100000	30	1.0	0.0001	2	Roulette	866.9	64.83
100000	40	1.0	0.001	0	Tournament	867.4	59.39
100000	20	1.0	0.001	2	Roulette	867.5	77.15
100000	20	1.0	0.0001	2	Roulette	871.4	74.76
100000	40	1.0	0.02	4	Tournament	871.5	86.48
70000	20	1.0	0.001	2	Roulette	875.2	77.39
70000	20	1.0	0.0001	2	Roulette	882.8	78.16
100000	20	0.8	0.0001	2	Roulette	884.1	70.08
100000	20	0.8	0.0001	4	Roulette	896.6	51.08
100000	20	1.0	0.01	4	Roulette	916.2	89.71
100000	20	1.0	0.0001	4	Roulette	920.2	51.32
100000	20	0.8	0.0001	4	Roulette	959.5	173.13
100000	20	0.6	0.0001	4	Roulette	1007.4	255.36
100000	20	0.8	0.001	4	Roulette	1025.6	223.53
100000	40	1.0	0.01	0	Tournament	1308.0	94.94
100000	40	0.8	0.01	0	Tournament	1361.4	119.04
100000	20	0.6	0.001	4	Roulette	1412.4	256.45
100000	20	0.6	0.001	2	Roulette	1759.8	151.44

Figura 2: tabela com comparação entre os testes dantizig42-d

Observamos resultados bons com parâmetros semelhantes aos usados na tabela da figura 1. Uma mudança óbvia é o número de gerações, mudado para acomodar a entrada maior. Outra coisa importante é que a diminuir ainda mais a taxa de mutação para a seleção de torneio, conseguimos os melhores resultados.

Com esses parâmetros, fizemos o teste de ainda outra entrada, agora a "att48_d", que possui número de cidades semelhante a entrada anterior, mas com distâncias bem maiores, com a solução ótima conhecida de 33523 unidades. Com isso, temos a tabela na figura 3 a seguir.

Gerações	População	P. de Cruzamento	P. de Mutação	N Elite	T. Cruzamento	Media	Desvio
100000	40	1.0	0.01	4	Tournament	36972.5	1791.20
100000	40	1.0	0.0001	4	Roulette	37214.5	2089.96
100000	50	1.0	0.0001	4	Roulette	37921.7	2269.88
100000	40	1.0	0.001	4	Roulette	37977.5	2768.60
100000	40	1.0	0.0001	4	Tournament	41598.5	3959.01

Figura 3: tabela com comparação entre os testes att48-d

Note que para essa entrada, o tipo de seleção torneio, com sua taxa de mutação

levemente maior, deu os melhores resultados em média, apesar de ter pouca diferença para os resultados com a roleta. Para determinar com maior certeza qual tipo de seleção se sobressai, seria necessário a execução do código mais vezes, além de testes mais rigorosos.

Depois disso, para as entradas "dantizig42_d"e "dantizig42_d", pegamos a melhor combinação de parâmetros de cada uma de suas respectivas tabelas e traçamos dois gráficos cada.

Os gráficos das imagens **??** e **??** mostram dez execuções usando esses parâmetros onde foram traçados retas que mostram o progresso do *fitness* da melhor solução por geração.

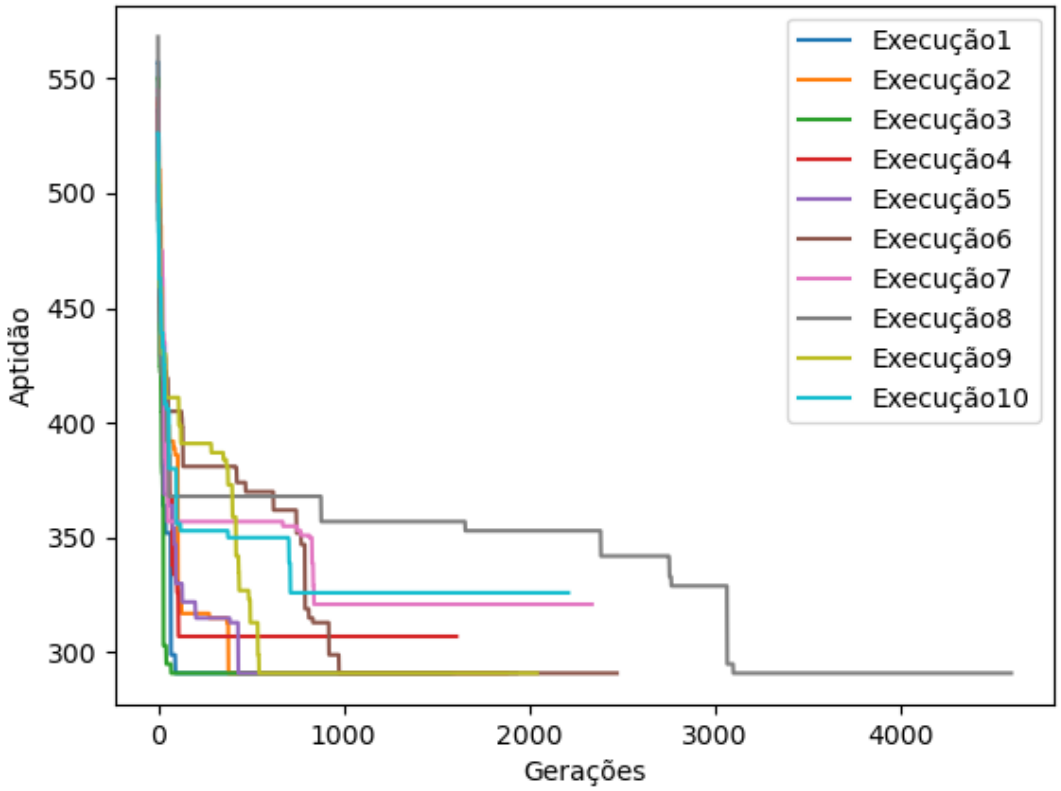

Figura 4: tabela com comparação entre os testes att48-d

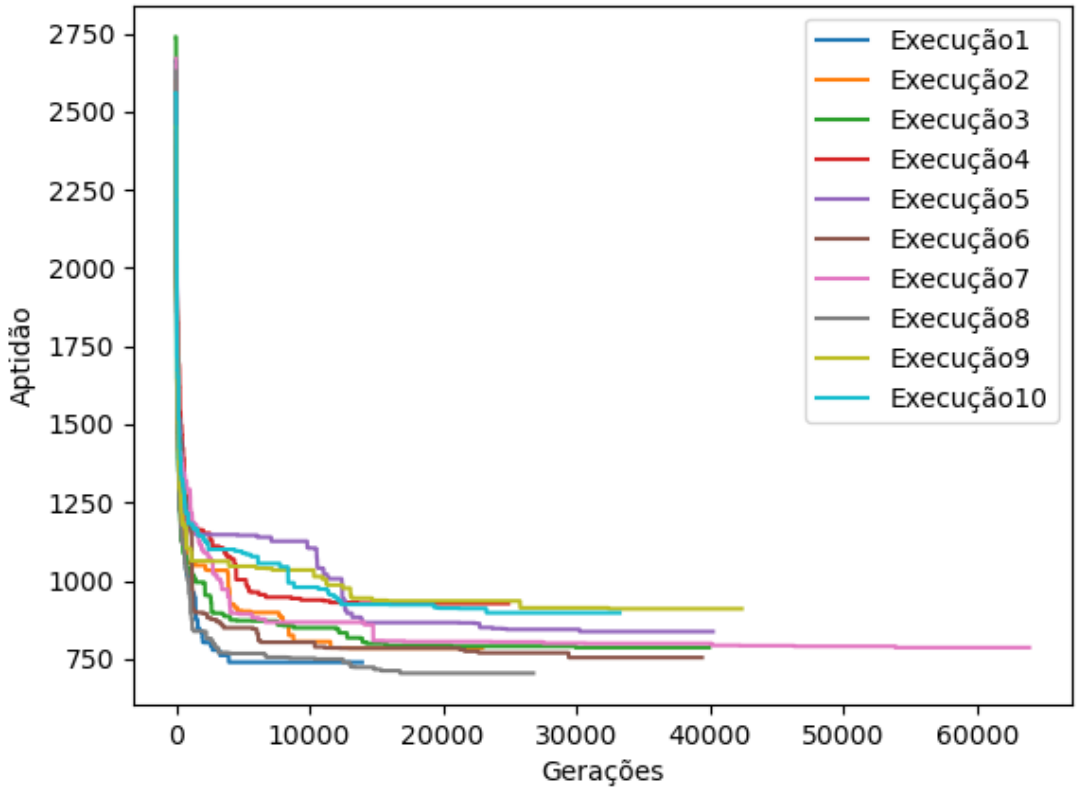

Figura 5: tabela com comparação entre os testes att48-d

Podemos notar que na maioria das execuções tivemos uma convergência rápida, encontrando o ótimo local ou global logo nas primeiras gerações e por conta do critério de parada de gerações sem melhoria, finalizando a execução antes de completar todas as gerações definidas. Mas podemos ver também que em alguns casos, essa convergência é mais demorada, o que mostra a importância do número de gerações ser alto.

No caso dos gráficos nas imagens 6 e 7 abaixo, temos a análise, em uma execução, do melhor e pior de cada geração, além da média e mediana da população durante a execução. Podemos verificar uma grande variação nos *fitness* da população na média e na mediana, que se encontram quase uma em cima da outra no gráfico. Isso reflete também na pior solução, que continua variando, muitas vezes por causa do elitismo, que previne toda a população de convergir em um ótimo local. Com a linha da melhor solução vemos o elitismo em ação, pois ele salva a melhor solução encontrada para que ela não seja perdida por uma mutação ao um cruzamento que gere um indivíduo pior.

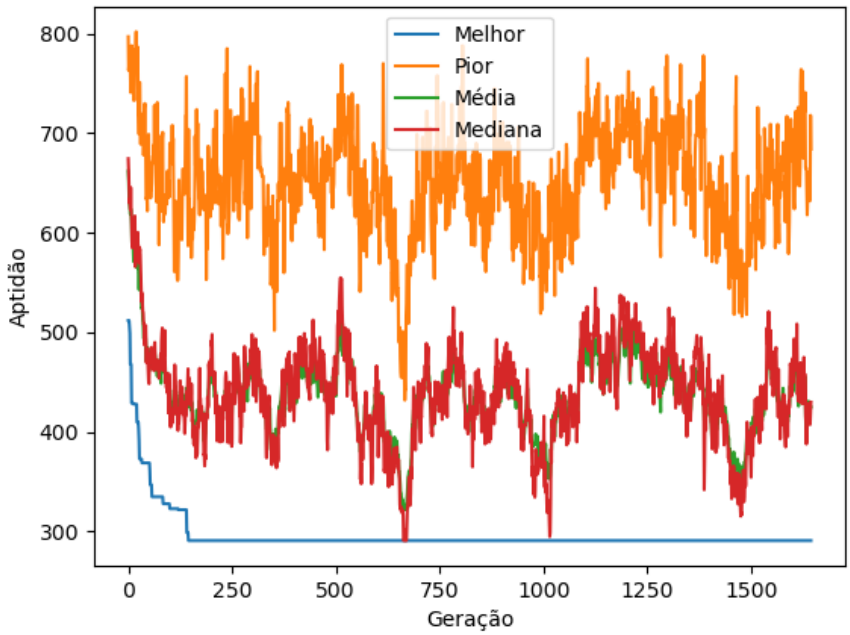

Figura 6: tabela com comparação entre os testes att48-d

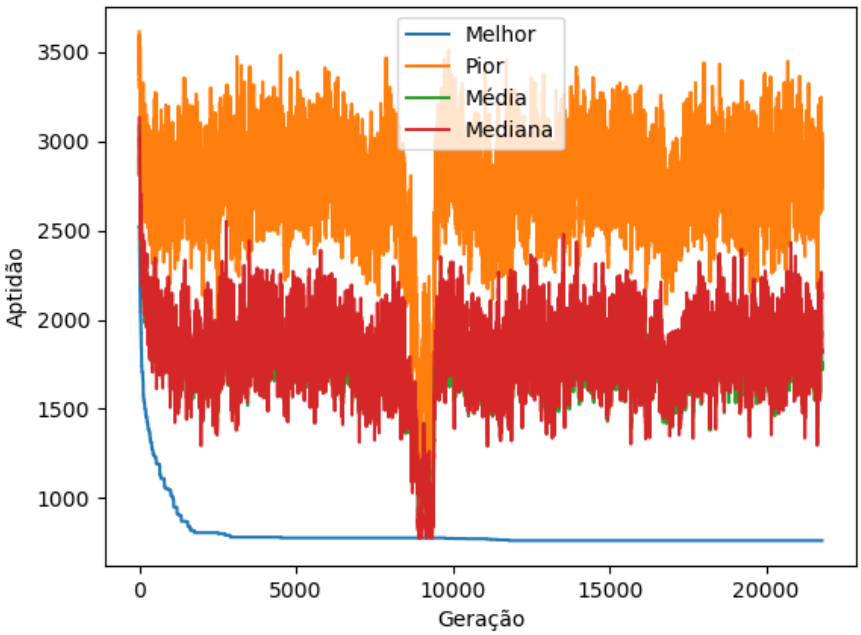

Figura 7: tabela com comparação entre os testes att48-d

5 CONCLUSÃO

Em todos os testes realizados, podemos ver que com os parâmetros certos conseguimos achar várias solução que se aproximam ou chegam na solução ótima conhecida das entradas. Isso nos mostra a qualidade do algoritmo genético para resolver o problema, conseguindo em poucos minutos ou segundos uma soluções que melhora muito em relação a soluções geradas aleatóriamente, sendo além, de relativamente simples implementação.