Universidade Federal
de São João del-Rei

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

Problemas do Caixeiro Viajante com o Scatter Serach

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 da UFSJ.

1.1 Objetivo

Este trabalho tem como objetivo aprender a construir e testar o algoritmo *Scatter Search*, buscando resolver o problema do caixeiro-viajante.

Além disso, o foco está na análise e calibragem dos parâmetros, visando melhorar a eficiência do algoritmo no problema.

2 O PROBLEMA

Este trabalho é baseado na solução de um problema muito famoso na Computação, um exemplo clássico para o teste de heurísticas e metaheurísticas, que é o problema do caixeiro-viajante.

2.1 Problema do Caixeiro-Viajante

O problema do caixeiro-viajante é um dos exemplos mais famosos de problemas de otimização combinatória na Computação. Por ser um problema difícil de resolver, pertencente à classe NP-Difícil, mas fácil de explicar e ilustrar, ele se tornou um problema comum para testar diferentes algoritmos de aproximação, nos quais se busca encontrar uma boa solução 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 um certo peso, que representa a distância entre elas, o objetivo é encontrar um circuito de menor distância possível, que comece em uma cidade, passe por todas as outras apenas uma vez, e então retorne à cidade inicial.

3 O ALGORITMO

Nesta seção, é introduzido o algoritmo utilizado no trabalho.

3.1 SCATTER SEARCH

O *Scatter Search* (SS), ou Busca Dispersa, é uma meta-heurística populacional que trabalha com a recombinação de soluções em um conjunto de referência, com o objetivo de gerar novas soluções.

Originado no artigo de F. Glover, em 1977, e posteriormente popularizado por Manuel Laguna e Rafael Martí, em 2003, o SS apresentou bons resultados em diversos problemas de otimização combinatória.

A ideia básica do algoritmo é a construção e atualização de um conjunto de referência que contém as melhores soluções geradas por um método de geração de soluções diversificadas.

Através da construção e aprimoramento de um grande conjunto de soluções, é formado um conjunto de referência menor, que será responsável por armazenar as melhores soluções encontradas na busca. Com o conjunto de referência, são gerados vários subconjuntos que passam por um processo de combinação e melhoria. Com essas novas soluções, o conjunto de referência é atualizado, repetindo-se o ciclo até que nenhuma nova solução seja adicionada.

Neste trabalho, é utilizado o método de construção gulosa aleatória, baseado no GRASP, para o método de geração de soluções diversificadas. Para a montagem do conjunto de referência, ele é dividido em duas partes: uma contendo as melhores soluções geradas e a outra contendo soluções diversificadas, adicionadas ao conjunto com base no número de vértices distintos em relação a qualquer solução já presente no conjunto.

O conjunto de referência será mantido com as soluções geradas pelas combinações de seus subconjuntos, de forma a evitar soluções repetidas.

4 ANÁLISE DE DESEMPENHO

Para a análise do desempenho do algoritmo, devem ser ajustados os parâmetros e anotados os valores das soluções encontradas, bem como o tempo gasto para realizar a execução. Utilizando um *script* para facilitar múltiplas execuções, foram elaboradas tabelas e gráficos para visualizar os efeitos dos parâmetros utilizados na solução.

Os parâmetros a serem testados são:

- P: número de soluções a serem geradas no método de geração diversificada;

- b_1: número de soluções boas escolhidas para compor b;
- b_2: número de soluções diversas escolhidas para compor b;
- α: taxa utilizada para o método de construção gulosa aleatória.

Inicialmente, foram realizadas 10 execuções com diferentes parâmetros, utilizando diferentes entradas e elaborando uma tabela para comparar os resultados.

A Figura 1 mostra a tabela com testes realizados ao executar o programa dez vezes para cada conjunto de parâmetros em uma entrada com 15 cidades, cujo melhor global conhecido é 291. É evidente a eficiência do algoritmo nessa entrada pequena, conseguindo alcançar o melhor resultado em todas as execuções, independentemente dos parâmetros. Considerando que o método de geração utilizado é baseado no GRASP, isso era esperado, pois as soluções construídas já possuem boa qualidade.

S	B1	B2	α	Media	Desvio
5	1	1	0.03	291.00	0.00
10	2	2	0.03	291.00	0.00
20	3	4	0.03	291.00	0.00
30	4	4	0.03	291.00	0.00
40	5	5	0.03	291.00	0.00

Figura 1: Tabela com testes em uma entrada com 15 cidades.

Realizando a mesma rotina de testes, foram elaboradas as tabelas nas Figuras 2 e 3 com entradas de 42 cidades, com ótimo global conhecido de 699, e 48 cidades, com ótimo global conhecido de 33551, respectivamente. Assim como na entrada de 15 cidades, em todas as execuções foram obtidos bons resultados, embora em alguns casos o ótimo global não tenha sido alcançado.

S	B1	B2	α	Media	Desvio
30	5	5	0.1	699.00	0.00
40	5	5	0.03	699.00	0.00
30	5	5	0.03	699.12	0.33
40	10	0	0.1	699.25	0.66
10	1	2	0.05	699.62	1.65

Figura 2: Tabela com testes em uma entrada com 42 cidades.

S	B1	B2	α	Media	Desvio
40	5	5	0.03	33551.00	0.00
40	5	5	0.2	33554.75	9.92
20	5	0	0.03	33622.25	106.89
20	2	3	0.03	33746.62	286.15
30	2	3	0.03	33792.62	224.16

Figura 3: Tabela com testes em uma entrada com 48 cidades.

Observa-se que valores baixos de α apresentaram melhores resultados. Isso se deve ao fato de que um α pequeno faz com que o método de construção gulosa aleatória tenda mais à parte gulosa do que à aleatória.

Outro ponto importante é o pequeno número de soluções em ambos os conjuntos inicial e de referência. Esse é um fator essencial para o funcionamento do algoritmo, pois, ao utilizar diferentes técnicas, ele se torna complexo e lento, e um número reduzido de indivíduos nos conjuntos melhora o desempenho. Como os resultados obtidos foram satisfatórios, não há necessidade de sacrificar o tempo de computação para buscar uma solução melhor.

Utilizando os dados obtidos na tabela, foi realizada uma execução para uma entrada com 128 cidades, sem melhor global conhecido. Com $S = 40$, $B_1 = 5$, $B_2 = 5$ e $\alpha = 0.03$, em dez execuções foi obtida uma média de distância de 20279.12, com um desvio de 227.68, sendo que o melhor valor encontrado foi 19812. Comparativamente, o algoritmo de colônia de formigas, implementado em outra atividade da disciplina, obteve o melhor valor de 20877 nas mesmas dez execuções. O pior valor encontrado pelo SS foi 20839, evidenciando seu melhor desempenho na resolução do problema.

Para analisar o progresso do algoritmo em direção à melhor solução, foram gerados gráficos mostrando o valor assumido pela melhor solução em dez execuções.

A Figura 4 mostra as dez execuções do algoritmo com os parâmetros $S = 40$, $B_1 = 5$, $B_2 = 5$ e $\alpha = 0.03$. Em todas as execuções, o ótimo global foi alcançado. Observe que isso ocorre em poucas iterações, embora cada iteração do algoritmo seja relativamente demorada, pois envolve a geração e combinação de vários subconjuntos, o aprimoramento e o ordenamento das soluções, além da remoção de soluções repetidas no conjunto. Todos esses métodos geram um custo computacional considerável. No entanto, o SS ainda consegue ser eficiente, pois converge rapidamente.

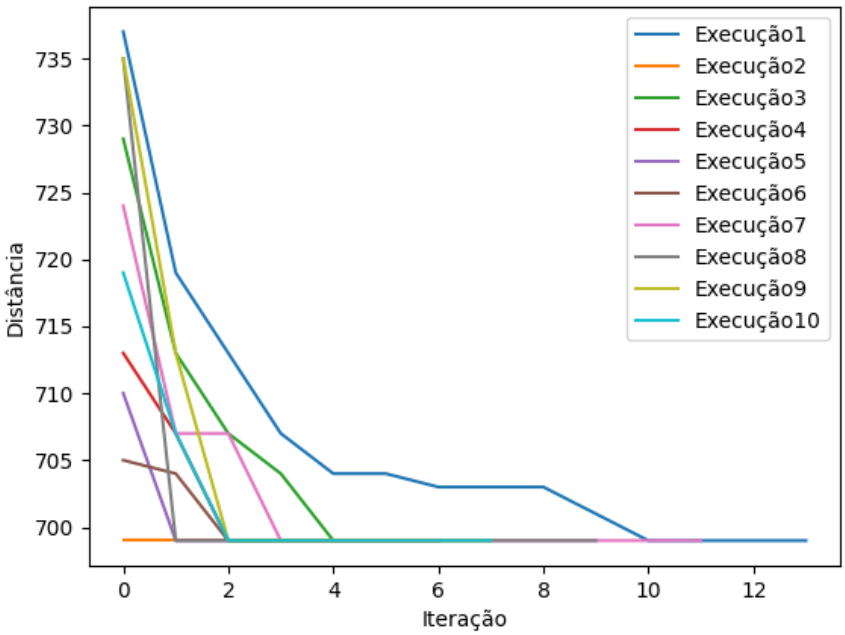

Figura 4: Gráfico com testes em uma entrada com 42 cidades.

Com a entrada de 48 cidades, foi gerado o gráfico da Figura 5, utilizando os mesmos parâmetros da Figura 4. O comportamento das execuções é semelhante, com o ótimo global sendo rapidamente alcançado.

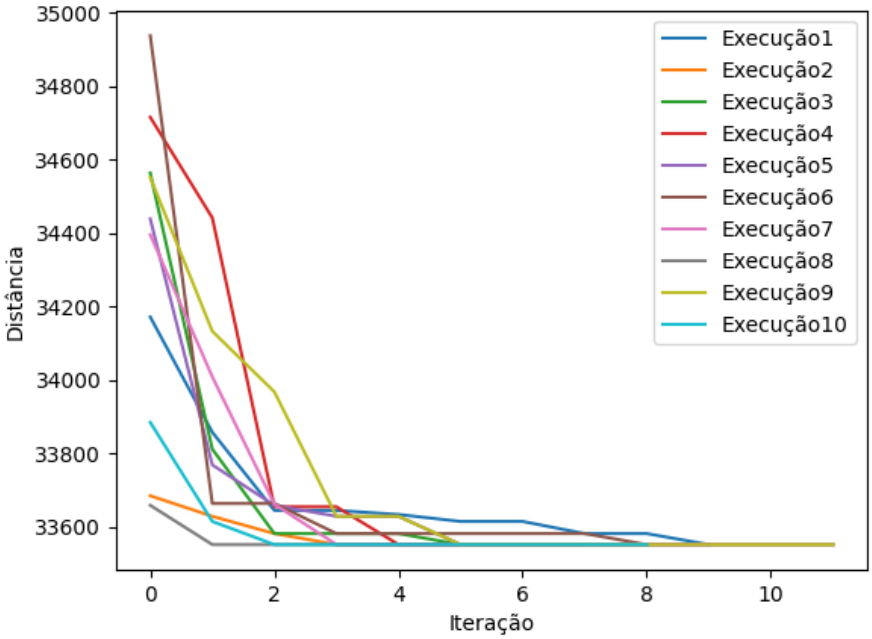

Figura 5: Gráfico com testes em uma entrada com 48 cidades.

Para entender melhor o comportamento do conjunto de referência, foram gerados gráficos, mostrando, em uma única execução, o melhor e o pior valor do conjunto, além de sua média e mediana, para entender a distribuição da qualidade das soluções.

As Figuras 6 e 7 mostram a análise das entradas com 42 e 48 cidades, respectivamente.

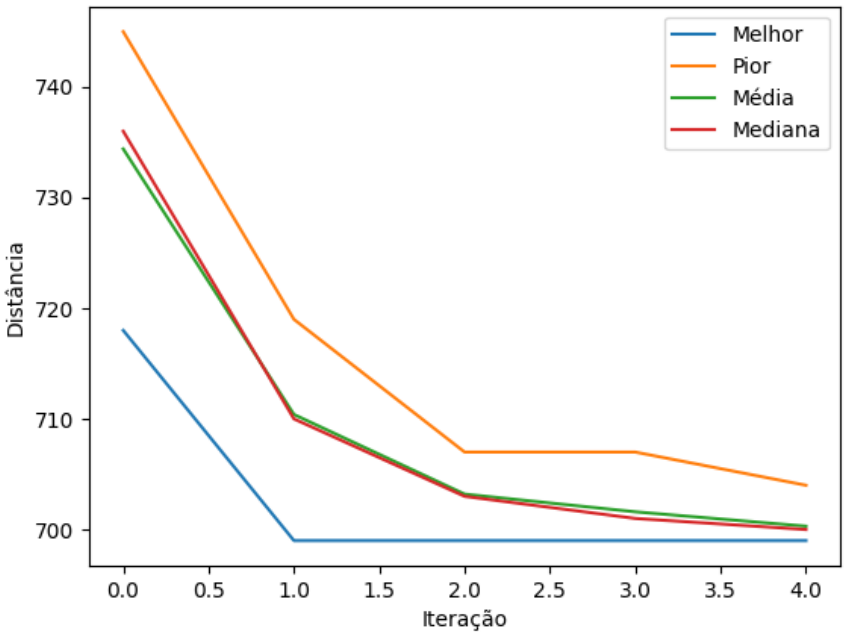

Figura 6: Gráfico com análise de execução em uma entrada com 42 cidades.

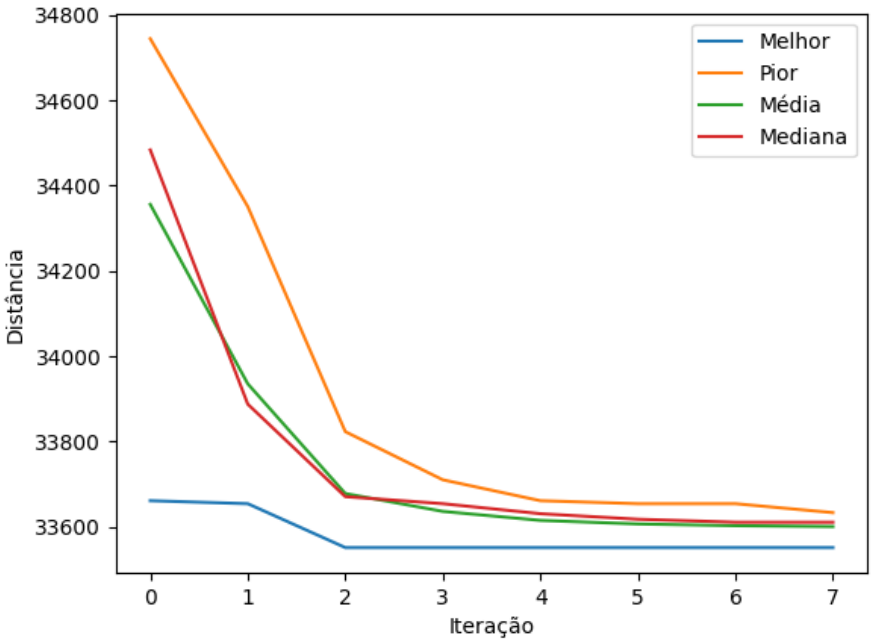

Figura 7: Gráfico com análise de execução em uma entrada com 48 cidades.

Para uma entrada maior, a Figura 8 mostra os resultados com a entrada de 128 cidades, utilizando os mesmos parâmetros anteriores. Com um espaço de busca maior,

fica ainda mais evidente a importância de manter o conjunto de referência diversificado. Para melhorar esse aspecto do algoritmo, uma possível solução seria implementar uma operação de mutação, semelhante à de um algoritmo genético.

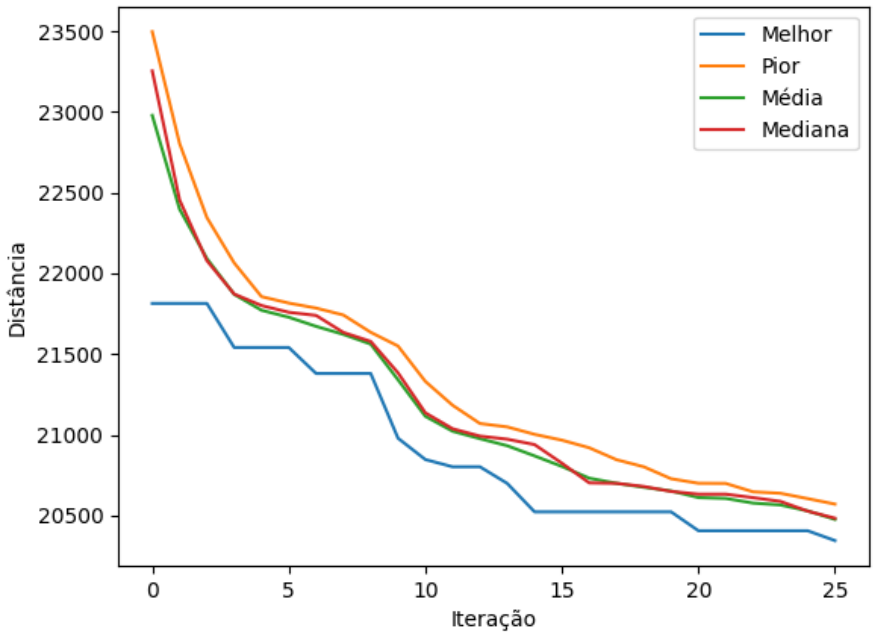

Figura 8: Gráfico com análise de execução em uma entrada com 128 cidades.

5 CONCLUSÃO

Ao observar os resultados obtidos pelo SS e compará-los com as soluções ótimas globais conhecidas, é possível perceber um bom desempenho, no qual o algoritmo consegue se aproximar bastante do ótimo global ou alcançá-lo em muitos casos.

Entretanto, é importante destacar o tempo elevado de execução do algoritmo, já que ele é bastante pesado, o que sugere a necessidade de otimizações para melhorar seu desempenho. Uma das possibilidades seria a paralelização do código, pois alguns de seus métodos são passíveis de paralelização, o que apresenta um grande potencial para melhorar seu desempenho.

Por fim, o algoritmo mostrou-se eficaz para o problema, mas é sempre importante verificar o funcionamento de cada uma de suas etapas no momento de implementá-lo em um novo problema, pois seu desempenho depende muito da geração de soluções que sejam boas tanto em termos de qualidade quanto de diversidade.