Universidade Federal
de São João del-Rei

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

Problema do Caxeiro Viajante com Colônia de Formigas

Gustavo Henriques da Cunha

São João del-Rei
2024

Lista de Figuras

1 tabela com comparação entre os testes lau15-dist 3
2 tabela com comparação entre os testes dantizig42-d 4
3 tabela com comparação entre os testes att48-d 4
4 gráfico com comparação entre as execuções lau15-dist 5
5 tabela com comparação entre as execuções dantizig42-d 5
6 tabela com comparação entre as execuções att48-d 6
7 tabela com comparação entre os testes lau15-dist 7
8 tabela com comparação entre os testes dantizig42-d 7
9 tabela com comparação entre os testes att48-d 8

Sumário

1 INTRODUÇÃO **1**

1.1 Objetivo 1

2 O PROBLEMA **1**

3 ABORDAGEM DO PROBLEMA **1**

4 ANÁLISE DE DESEMPENHO **3**

5 CONCLUSÃO **8**

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 o algoritmo Colônia de Formigas, buscando resolver o problema do caxeiro viajante.

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 uma variação do algoritmo de Colônia de Fromigas, o *Rank-Based Ant System*, onde começamos colocando uma formiga em cada

cidade e, em cada itteração, criaremos uma solução para cada formiga baseando-se na distância da cidade atual da formiga para as cidades vizinhas e no conhecimento providenciado pela colônia.

O *Ant System* mantém, além das formigas, uma matriz de feromônios, que representam o conhecimento da colônia. Esses feromônios mostram os caminhos que as fromigas passaram, no caso do *Rank-Based Ant System*, o conhecimento das melhores, com enfoque no caminho da formiga com distância mais curta.

Na construção do caminho da formiga, temremos uma função probabilística que irá retornar a próxima cidade a ser vizitada, com base no conhecimento da matriz de feromônios e da matriz de distância do grafo. Na primeira execução, a matriz de feromônios estará nivelada, assim a decisão será feita com base na distância da próxima cidades, o que se aproxima de uma heurística gulosa. Com isso, logo na primeira execução já teremos uma solução que poderá ser melhor que uma solução aleatória qualquer.

A função probabilística é dada por:

$$p_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \in \mathcal{N}_i^k}(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}$$

Onde $\tau_i j$ é o feromônio associado a aresta (i, j), $\mathcal{N}_i^k$ representa a vizinhaça da cidade atual, nas quais contamos apenas as cidades que a formiga ainda não visitou, e $|eta_i j$ é um fator inversamente proporcional ao custo da aresta (i, j). Os fatores α e β são constantes ajustáveis, que dizem, respectivamente, o quanto os feromônios e o quanto a distância influenciam no cálculo da probabilidade.

Quando todas as formigas construírem seus caminhos, devemos avaliar suas solução com base na distância percorrida, e então atualizar a matriz de feromônios.

No *Rank-Based Ant System* escolhemos um número w das melhores soluções encontradas e atualizamos a matriz de feromônios com base na formula:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \sum_{r=1}^{w-1}(w - r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs}$$

Onde τ_{ij} representa a aresta (i, j) na matriz de feromônios, $\Delta\tau_{ij}^{r} = 0$ se a aresta(i, j) não pertença a solução e $\Delta\tau_{ij}^{r} = \frac{Q}{L_k}$ caso pertença, onde Q é uma constante definida pelo usuário e L_k representa a distância da solução.

Assim, pelo somatório passaremos pelas melhores soluções e colocaremos uma quan-

tidade de feromônios proporcional ao seu ranking, e no final, adicionaremos o feromônio referente a melhor solução em maior quantidade.

4 ANÁLISE DE DESEMPENHO

O algoritmo *Rank-Based Ant System* é probabilistico e depende da qualidade de seus parâmetros para gerar soluções melhores. 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, podemos variar os parâmetros como número de iterações, as constantes α e β, a taxa de evaporação do feromônio ρ, a constante Q e o número de indivíduos a deixarem feromônio W. Neste trabalho, escolhemos variar os três parâmetros α, β e ρ, ajustando os outros parâmetros com base na entrada escolhida. Para cada conjunto de parâmetros, é executado 10 vezes o algoritmo, 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 objetivo de achar quem têm a maior média e que 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.

Iterações	α	β	ρ	Q	W	Média	Desvio
100	1.0	5.0	0.5	100	5	291.00	0.00
100	1.0	5.0	0.2	100	5	291.80	1.60
100	1.0	3.0	0.3	100	5	293.80	4.75
100	3.0	5.0	0.3	100	5	294.60	5.20
100	1.0	5.0	0.8	100	5	295.00	6.20
100	3.0	5.0	0.5	100	5	295.20	6.72
100	3.0	5.0	0.8	100	5	296.00	8.45
100	1.0	3.0	0.8	100	5	297.00	8.99
100	1.0	3.0	0.5	100	5	300.10	11.65
100	3.0	3.0	0.3	100	5	309.20	18.53

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

Podemos ver que o algoritmo conseguiu constantemente a solução ótima com um β maior que o α e com uma taxa de evaporação relativamente alta. Vemos que com poucas iterações o algoritmo consegue com facilidade achar uma solução boa.

Entretanto, a entrada é pequena, e conseguiu bons resultados com vários conjuntos, o que mostra que a análise dos parâmetros ainda não é suficiente.

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

Iterações	α	β	ρ	Q	W	Média	Desvio
100	1.0	5.0	0.1	200	14	706.70	7.13
150	1.0	5.0	0.1	200	14	707.40	6.90
150	1.0	4.0	0.1	200	14	709.90	9.17
150	1.0	3.0	0.1	200	14	711.80	9.30
150	1.0	8.0	0.1	200	14	716.10	12.38
100	1.0	5.0	0.3	200	14	716.20	13.60
150	1.0	3.0	0.2	200	14	716.20	6.95
100	1.0	5.0	0.5	200	14	720.70	10.65
150	1.0	3.0	0.5	200	14	723.20	9.10
100	1.0	5.0	0.8	200	14	738.10	17.96
150	3.0	5.0	0.5	200	14	751.00	17.75
150	3.0	5.0	0.1	200	14	768.90	22.16

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

Observamos resultados bons com parâmetros de β maior que o α semelhantes aos usados na tabela da figura 1, mas uma grande mudança a ser observada é a mudança de ρ, que com um valor pequeno de 0.1 gera as melhores soluções.

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.

Iterações	α	β	ρ	Q	W	Média	Desvio
150	1.0	4.0	0.1	6000	16	34616.10	413.70
150	1.0	5.0	0.1	6000	16	34664.20	406.04
150	1.0	5.0	0.2	6000	16	34688.90	428.08
150	1.0	6.0	0.1	6000	16	34761.50	404.07
150	1.0	5.0	0.5	6000	16	35369.20	780.10

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

Note que para essa entrada, a mudança do β para 4 gerou resultados ligeiramente melhores que com β igual a 5, mas com maior desvio, o que mostra que ambos os casos conseguem gerar bons resultados, e seria necessário um teste mais extendo para resolver qual é melhor.

Depois disso, pegamos a melhor combinação de parâmetros de cada uma de suas respectivas tabelas e traçamos dois gráficos cada.

Os gráficos das imagens 4, 5 e 6 mostram dez execuções usando esses parâmetros onde foram traçados retas que mostram o progresso da *qualidade* da melhor solução por geração.

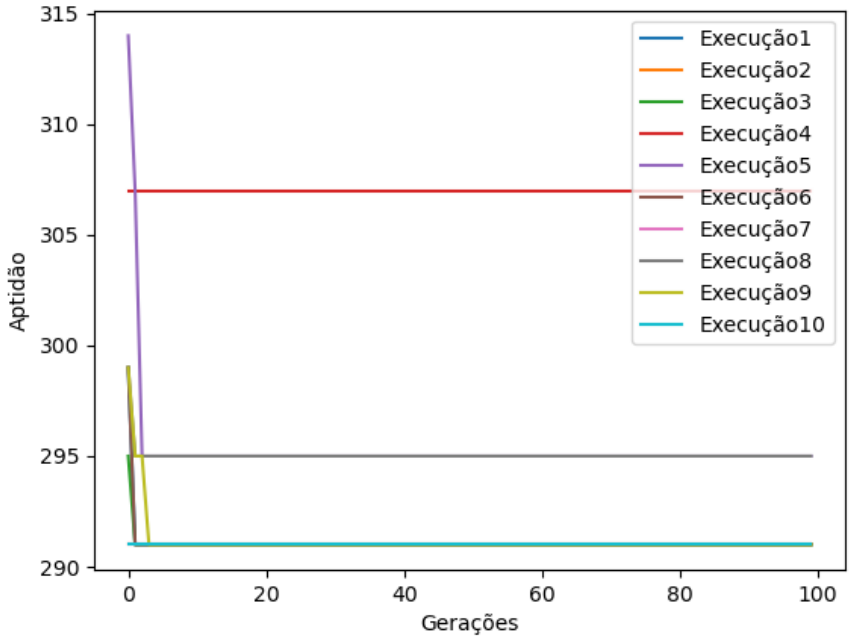

Figura 4: gráfico com comparação entre as execuções lau15-dist

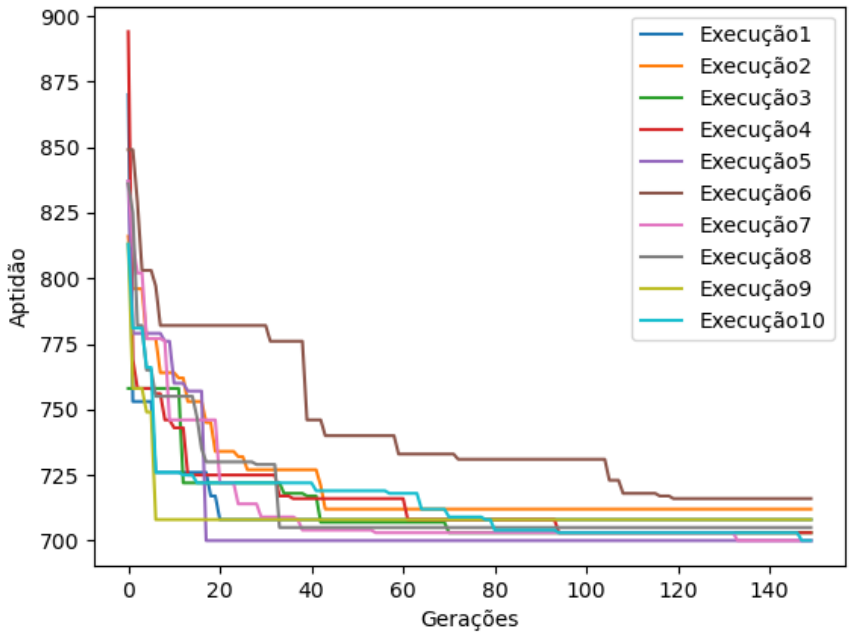

Figura 5: tabela com comparação entre as execuções dantizig42-d

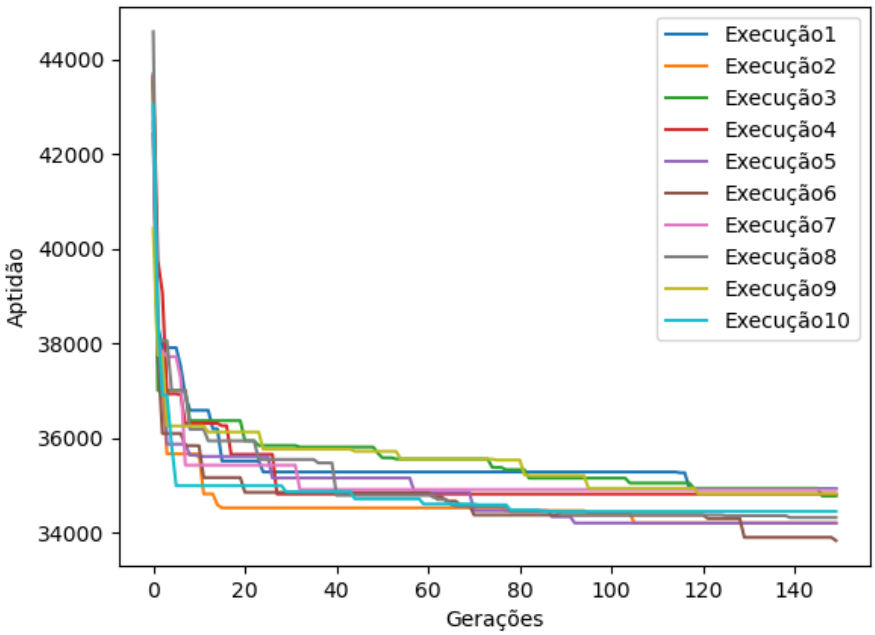

Figura 6: tabela com comparação entre as execuções att48-d

Podemos notar que em algumas das execuções tivemos uma convergência rápida, encontrando o ótimo local ou global logo nas primeiras gerações. Mas podemos ver também que em alguns casos, essa convergência é mais demorada, o que mostra que poderíamos até mesmo aumentar o número de iterações ou utilizar um critério de parada por um número de iterações sem melhoria.

No caso dos gráficos nas imagens 7, 8 e 9 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 na *qualidade* da população na pior solução, na média e na mediana, mas que ao longo das execuções elas tendem a se aproximar da melhor solução.

No caso da melhor solução, como salvamos a melhor solução achada pelo algoritmo em uma variável diferente a das soluções das formigas, ela não é retratada no gráfico, mas é mostrado a solução da formiga com melhor aptidão em cada iteração. Mesmo com a melhor solução não sendo mostrada no gráfico, podemos perceber que as soluções das formigas, no geral, melhoram em cada geração, apesar de alguns picos de piora. Entretanto esses picos não são tão ruins, pois eles podem permitir que a solução fuja de

um ótimo local.

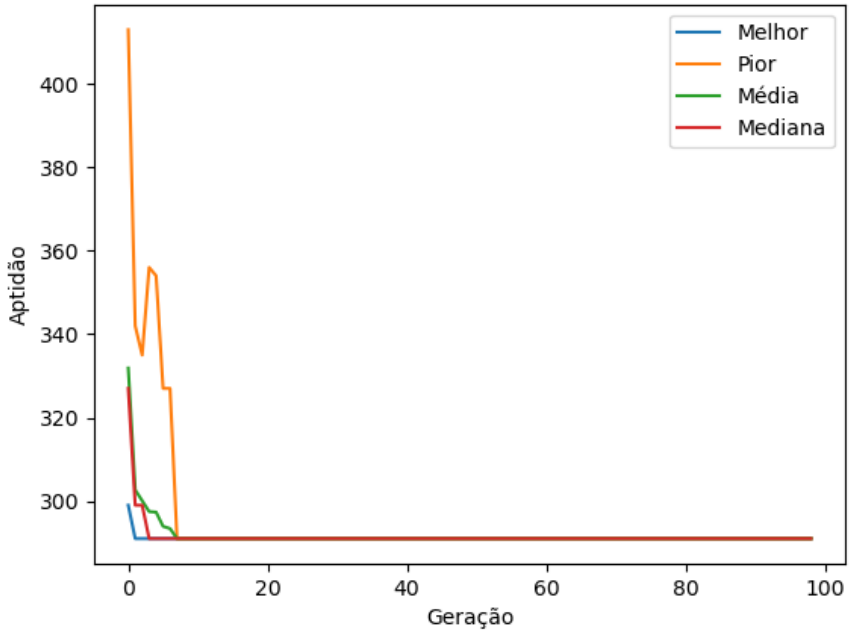

Figura 7: tabela com comparação entre os testes lau15-dist

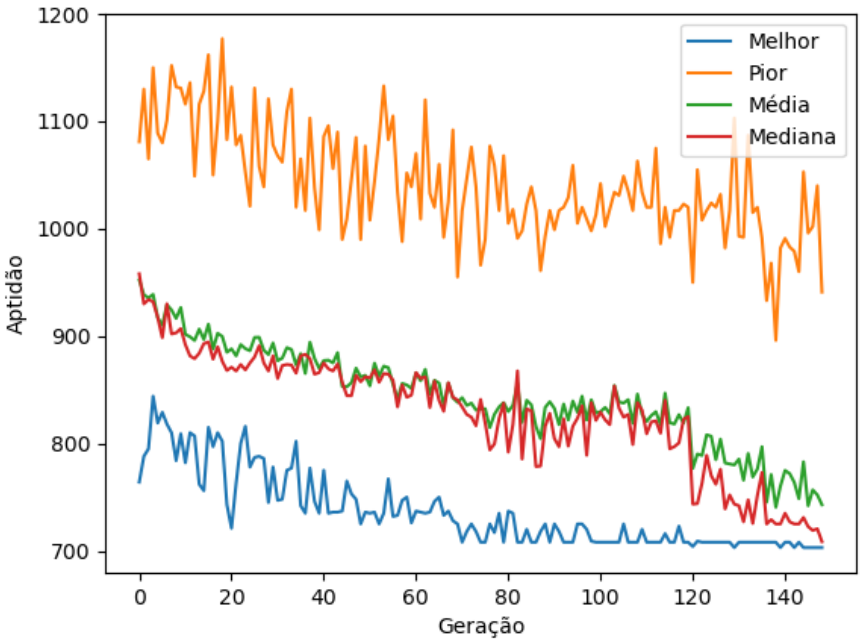

Figura 8: tabela com comparação entre os testes dantizig42-d

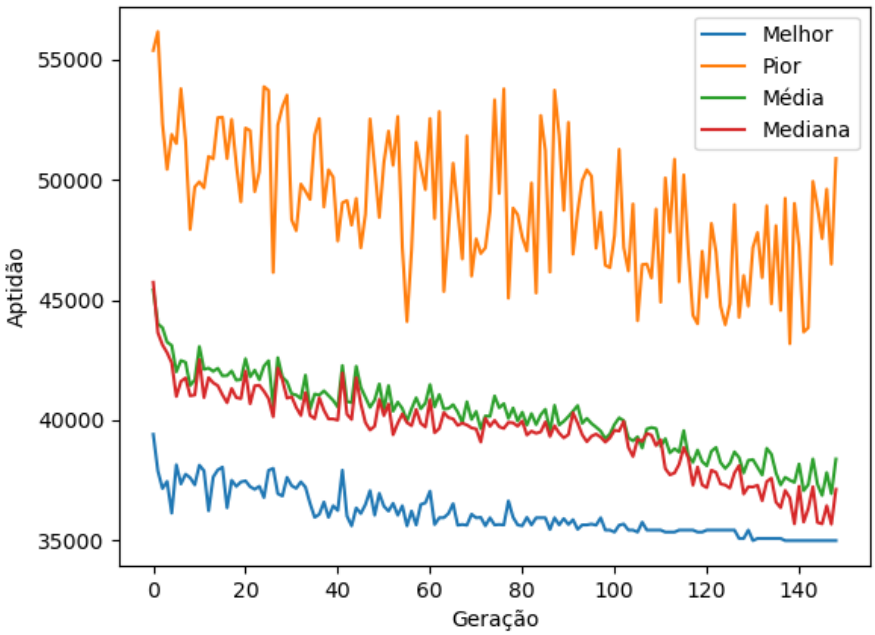

Figura 9: tabela com comparação entre os testes att48-d

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 deu resultados muito bons, onde vimos que o algoritmo *Rank-Based Ant System* consegue de forma bastante eficiente encontrar soluções boas para um problema custoso como o PCV.