Colônia de Formigas Para o Problema Do Caixeiro Viajante

Algoritmos Bioinspirados

Heitor Lourenço Werneck
heitorwerneck@hotmail.com

1 Introdução

Colônia de formigas é uma meta-heurística que toma como inspiração o comportamento forrageiro de algumas especies de formigas. Essas formigas depositam feromônio no chão para marcar um bom caminho que deve ser seguido por outros membros da colônia [5]. Então o colônia de formigas é aplicada para problemas de otimização com uma abordagem similar a o que acontece na prática com uma colônia de formigas.

Na Otimização por Colônia de Formigas (ACO - Ant Colony Optimization) um número de formigas artificiais constroem soluções para o problema de otimização considerado e trocam informações da qualidade das soluções através da comunicação que também e feita por formigas reais [5].

A ACO está sendo abordada por diversos métodos de otimização de problemas discretos que também são NP-difíceis, que são problemas que nenhum algoritmo que soluciona o problema em tempo polinomial é conhecido, nesses problemas a ACO pode ser utilizada para obter soluções de alta qualidade em tempo viável. Alguns problemas que são tratados utilizando colônia de formigas: caixeiro viajante [3, 6, 4]; roteamento de veículos [10, 1]; coloração de grafos [2, 9] e outros [5].

Esse trabalho consiste na implementação e análise de um algoritmo de colônia de formigas para a solução do problema do caixeiro viajante.

2 Problema

O problema do caixeiro viajante (PCV) consiste na busca por um circuito que possua a menor distância, começando em qualquer cidade e visitando todas outras cidades, cada uma exatamente uma vez, e então voltando para a cidade de origem.

Este problema pode ser definido formalmente como: dado um conjunto $V = \{v_1, ..., v_n\}$ de n cidades/-vértices v_i e uma matriz de distâncias $d_{n\times n}$ (d_{ij} é a distância da cidade/vértice v_i até a cidade/vértice v_j), tal que $d_{ij} = d_{ji}$ e $d_{ii} = 0$, o objetivo é encontrar uma permutação s de V que minimize a função da equação a seguir:

$$f(s) = \sum_{i=1}^{n-1} d_{s_i s_{i+1}} + d_{s_n s_1} \tag{1}$$

3 Solução

Para tratar o problema do caxeiro viajante será utilizado o modelo de colônia de formigas, como declarado anteriormente. O Ant System é o primeiro algoritmo de ACO proposto na literatura [7], ele foi utilizado devido a simplicidade e eficiência.

O algoritmo implementado seguirá os seguintes passos: a cada iteração os valores de feromônio são atualizados por todas m formigas que construíram uma solução na iteração corrente. Os valores de feromônio ficam em uma matriz $\tau_{n\times n}$, tal que τ_{ij} denota o feromônio entre o vertíce v_i e o vertíce v_j, τ_{ij} é atualizado após cada iteração de acordo com a regra:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{2}$$

tal que ρ é a taxa de evaporação, $\Delta\tau_{ij}^k$ é a quantidade de feromônio deixado pela formiga k, sendo este:

$$\Delta\tau_{ij}^k = \begin{cases} Q/L_k & \text{se a formiga } k \text{ possui a aresta } (i,j) \text{ no seu circuito} \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

tal que Q é uma constante e L_k é o comprimento do circuito da formiga k.

Na construção de uma solução a formiga vai selecionar a cidade a partir de uma estrátegia probabilistica. Dado uma formiga k na cidade i e com um caminho s' construido até então, a probabilidade de ir para uma outra cidade j será dada por:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{e_{il} \in N(s')} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{se } e_{ij} \in N(s') \\ 0 & \text{caso contrário} \end{cases} \tag{4}$$

$N(s')$ é o conjunto de arestas que são viáveis para a formiga no estado atual dela, isso é, as arestas para os vertíces/cidades ainda não exploradas pela formiga k, a partir da cidade atual dela. Os parâmetros α e β controlam, em conjunto, o trade-off da importância que será dada para o feromônio τ_{ij} e a informação da heurística η_{ij} (dado por $\eta_{ij} = 1/d_{ij}$, ou seja, a heurística irá priorizar as arestas com menor distância)$.

O ponto de partida para cada formiga é um vértice aleátorio do conjunto de vértices, a partir desse ponto a solução é construida; o críterio de parada do algoritmo é um número de iterações máximo e a melhor solução de todas sempre é preservada.

Uma visão geral, simplificada, do algoritmo feito é dada a seguir (o algoritmo real possui algumas diferenças esetruturais para otimização, porém a lógica é a mesma):

Algoritmo 1 Algoritmo de ACO

```
Inicia ⌊n · Taxa de formigas⌋ (taxa de formigas que serão utilizadas relativo ao tamanho do problema) formigas, todas com soluções vazias
L* ← +∞
s*
Inicializa a matriz de feromônios τ_{n×n} com τ_0
for i = 1 to #Iterações do
    Seleciona um ponto de partida aleátorio para cada formiga
    for k = 1 to m do
        Constroi uma solução s_k para a formiga k a usando a regra de probabilidade (equação 4) até esgotar as arestas viáveis
        Calcula o comprimento L_k do circuito criado pela formiga k
        if L_k < L* then
            s* ← s_k, L* ← L_k
    Atualiza os feromônios a partir da regra dada na equação 2
```

Todos parâmetros utilizados no algoritmo são descritos a seguir:

Tabela 1: Parâmetros

Parâmetro	Descrição
#Iterações	Críterio de parada do algoritmo, terminará após #Iterações
Taxa de formigas	Taxa de formigas relativo ao tamanho do problema
τ_0	Valor para inicializar a matriz de feromônios
ρ	Taxa de evaporação
Q	Constante que define uma taxa de feromônio a ser adicionada, relativa a um comprimento
α	Define um nível de importância para o feromônio na escolha de um caminho
β	Define um nível de importância para a heurística na escolha de um caminho

4 Resultados

Para a análise de resultados foi utilizado 10 execuções para cada conjunto de parâmetro, devido a natureza probabilística do problema é importante sempre sumarizar os resultados de varias execuções para ter uma ideia do resultado esperado.

Foram utilizadas as instâncias descritas na tabela a seguir, obtidas de `https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html`.

Tabela 2: Características das instâncias.

Instância	#Cidades	Solução ótima
lau15	15	291
sgb128	128	-

4.1 Calibragem de parâmetros

Para o começo da análise um experimento fatorial completo será realizado em cada instância. Os valores para busca estão indicados na tabela a seguir com o domínio de busca para cada parâmetro (tabela 3).

Tabela 3: Domínio de busca de cada parâmetro.

#Iterações	25
Taxa de formigas	1.0
τ_0	10^{-16}
ρ	{0.3,0.5,0.7}
Q	{75,100,125}
α	1
β	{3,5,7}

É possível observar que somente 3 parâmetros foram escolhidos para variação, ja que em instâncias grandes um tempo grande é necessário para realizar cada execução. O número de iterações foi escolhido como 25, pois é no geral um número no qual o algoritmo ja se estabiliza em uma solução. A taxa de formigas foi de 1.0 para que o número de formigas sempre seja igual ao número de cidades, e assim uma busca mais completa seja feita. τ_0 foi escolhido um valor baixo arbitrário. O α foi escolhido um valor fixo, 1, e somente com o β o trade-off será ajustado.

A taxa de evaporação ρ tem um grande impacto na qualidade das soluções então foi escolhida ser variada, assim como o Q e o β. Pelo custo computacional grande somente 3 parâmetros foram escolhidos para variação, porém vai ser possível observar que somente com esses parâmetros conseguimos ajustar e dar diferentes comportamentos para o algoritmo, para a busca de melhores soluções.

Depois da execução o top-15 parâmetros, de cada instância, são descritos nas tabelas 5 e 4, onde μ denota a média e σ denota o desvio padrão das 10 execuções. Nessa tabela os dados são obtidos pela ultima iteração de cada execução.

A coluna "Melhor aptidão global"descreve o melhor valor obtido na ultima iteração, de todas soluções obtidas durante toda execução da meta-heurística.

"Melhor aptidão"descreve a melhor valor obtido nas formigas da última iteração, as outras colunas seguem a mesma lógica.

4.2 Experimento fatorial

De acordo com a tabela 4 foi possível ver que na instância mais simples o valor ótimo foi facilmente obtido por diversos conjuntos de parâmetros e execuções, assim como um desvio padrão de 0 que indica que todas execuções conseguiram chegar na solução ótima. Isso mostra que o algoritmo implementado consegue até mesmo obter a solução ótima no caso de uma instância simples.

Outros valores da tabela indicam que as formigas convergiram para a melhor solução. As formigas também não tiveram muitos outliers, o que mostra uma busca mais inteligente, proxima da solução de mais qualidade. Poucos valores foram distantes da melhor solução, porém até mesmo esses não são mais que

duas vezes a melhor solução. A variação no geral neste algoritmo é baixa, tanto para as piores soluções, media, mediana e melhor solução entre execuções. Isso pode ser dado por diversas fatores: o modelo apresentado é faz a construção da solução direcionado a boas soluções, dificilmente uma opção local muito ruim será incluída em uma solução; o problema não apresenta fortes distinções entre soluções ou pequenas variações em soluções, já que em outros problemas pequenas modificações podem gerar grandes diferenças na aptidão de uma solução (e.g., tornar a solução inviável).

No geral todos parâmetros conseguiram chegar na solução ótima, devido a instância ser muito simples não há muito o que analisar, então vamos focar mais na análise da próxima instância que é mais complexa.

Na tabela 5 que mostra os resultados da instância sgb128 é possível observar que nessa instância muitas questões da instância anterior se repetiram, como por exemplo: a variação entre execuções não é tão grande; formigas bem estáveis em relação a soluções, os outliers não são tão extremos, porém isso também é um bom indicativo da continuação de procura por boas soluções não obvias que podem ser boas.

Nessa instância fica bem claro uma carácteristica do algoritmo implementado, que a cada iteração um novo conjunto de formigas é gerado e essas muitas vezes não vão chegar em uma melhor solução já obtida, porém podem continuar buscando outras próximas da melhor solução, o que mostra uma forte característica de diversificação. Com uma boa solução encontrada então é realmente interessantem gastar recursos computacionais para procura de outras soluções ainda não exploradas, com a busca sendo guiada com informações de boas soluções atráves do feromônio.

Essa instância é bem complexa então diversas melhorias foram obtidas variando os parâmetros e cada parâmetro possui pequenas melhorias em relação a outros. A solução ótima para essa instância não é conhecida então as comparações serão feitas entre as soluções obtidas.

É possível observar que os melhores parâmetros, de acordo com espaço de busca, estão tentando priorizar mais a informação de distância das cidades do que o feromônio (com esse maior balanço uma convergência mais lenta é realizada), porém para os 2 melhores parâmetros mais evaporação foi escolhida, o que mostra uma maior necessidade por diversificação das soluções que ajudou na obtenção da solução melhor [8]. Com esses melhores parâmetros um Q mais elevado foi priorizado, o conjunto de parâmetros com Q de 75 obteve uma grande diferença de qualidade de soluções, ficando 13 posições abaixo da melhor solução somente com essa pequena diferença, demonstrando a importância desse parâmetro no ajuste junto com os outros parâmetros. O terceiro melhor conjunto de parâmetros utilizou menos diversificação pela evaporação, porém com um Q menor conseguiu resultados proximos do melhor encontrado.

È possível ver que o melhor conjunto de parâmetros realmente diversifica pelo desvio padrão mais elevado comparado a outros conjuntos de parâmetros, em média, mediana e pior aptidão.

	ρ	Q	β	Melhor aptidão global		Melhor aptidão		Aptidão média		Aptidão mediana		Pior aptidão	
				μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1	0.3	75.0	3.0	291.0	0.0	291.0	0.000	308.940	13.107	300.2	13.990	380.3	35.577
2	0.7	75.0	7.0	291.0	0.0	291.0	0.000	295.246	3.029	293.4	3.373	300.7	5.375
3	0.7	125.0	5.0	291.0	0.0	291.0	0.000	294.153	2.862	292.6	3.373	299.9	11.892
4	0.7	100.0	5.0	291.0	0.0	291.8	2.529	296.380	6.277	293.0	2.828	309.9	29.053
5	0.7	75.0	5.0	291.0	0.0	291.4	1.264	297.560	8.870	293.8	3.794	317.5	39.172
6	0.7	125.0	3.0	291.0	0.0	293.0	3.399	304.013	8.703	297.1	8.089	373.5	58.705
7	0.7	100.0	3.0	291.0	0.0	291.0	0.000	297.126	7.370	294.2	4.541	325.7	47.131
8	0.5	125.0	7.0	291.0	0.0	291.0	0.000	294.746	3.700	293.0	3.399	309.1	26.349
9	0.5	100.0	7.0	291.0	0.0	291.0	0.000	294.906	2.034	292.6	3.373	307.2	25.930
10	0.5	75.0	7.0	291.0	0.0	291.0	0.000	294.226	1.069	293.4	3.864	299.0	0.000
11	0.5	125.0	5.0	291.0	0.0	291.0	0.000	294.793	2.835	293.8	3.794	308.7	23.228
12	0.7	100.0	7.0	291.0	0.0	291.4	1.264	294.813	1.433	295.0	4.216	299.0	0.000
13	0.5	100.0	5.0	291.0	0.0	291.0	0.000	299.246	7.246	294.2	4.131	347.3	53.851
14	0.5	100.0	3.0	291.0	0.0	291.4	1.264	308.486	15.490	296.1	7.030	373.1	59.039
15	0.5	75.0	3.0	291.0	0.0	291.8	2.529	301.633	8.760	295.0	3.771	350.2	54.448

Tabela 4: Resultados da instância lau15.

	ρ	Q	β	Melhor aptidão global		Melhor aptidão		Aptidão média		Aptidão mediana		Pior aptidão	
				μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1	0.7	100.0	3.0	22139.0	347.09	23466.5	505.575	27785.802	176.164	27689.40	233.642	33603.3	1217.9388
2	0.7	125.0	3.0	22204.7	479.38	23670.0	652.263	27914.913	123.489	27771.45	114.871	33800.9	810.693
3	0.3	75.0	3.0	22206.7	403.79	23550.7	866.141	27449.961	148.227	27224.00	155.854	32633.4	755.983
4	0.3	100.0	3.0	22334.1	260.56	23162.8	574.870	27469.114	87.5869	27335.35	197.449	32393.8	633.228
5	0.3	125.0	3.0	22429.5	278.26	23164.6	493.491	27437.459	93.9285	27367.20	159.708	32837.1	931.587
6	0.5	125.0	3.0	22456.0	263.15	23488.7	537.761	27762.289	210.157	27606.75	209.268	33938.3	1559.489
7	0.5	75.0	3.0	22475.7	233.72	23605.8	649.736	27703.592	128.875	27496.15	185.101	33921.1	850.237
8	0.5	125.0	5.0	22500.0	191.69	23603.0	602.683	27301.614	172.121	27128.50	289.213	32264.0	648.857
9	0.5	100.0	3.0	22507.8	329.60	23592.4	473.375	27833.778	213.384	27698.90	291.338	33576.4	1088.355
10	0.5	75.0	5.0	22520.2	339.68	23715.8	527.737	27365.262	172.132	27267.45	162.588	32046.2	536.163
11	0.3	100.0	5.0	22572.2	343.50	23329.2	509.407	27221.716	147.836	27097.20	163.689	32141.1	868.438
12	0.3	125.0	5.0	22593.7	185.06	23655.4	428.113	27074.442	119.454	27018.20	213.714	31371.8	757.668
13	0.7	75.0	3.0	22608.6	358.73	23608.9	444.962	27938.792	190.031	27771.50	200.921	33714.0	1139.154
14	0.7	75.0	5.0	22630.3	280.32	23860.3	399.902	27416.364	165.145	27281.85	144.866	32969.2	1059.333
15	0.3	75.0	5.0	22671.3	230.48	23847.4	499.791	27209.414	245.608	27081.75	249.146	31887.3	1125.048

Tabela 5: Resultados da instância sgb128.

4.3 Análise das melhores soluções

Também é importante analisar os parâmetros indivídualmente (suas execuções) para entender possíveis situações e seus comportamentos. Serão apresentados dois gráficos, um da média das execuções durante as iterações e outro de cada execução individualmente.

4.3.1 Instância lau15

Na figura 1 é possível ver que a média, mediana e pior aptidão vão convergindo para o melhor caminho durante o tempo, o que significa que as formigas com o tempo vão tender mais para um certo caminho o que é o esperado. Como a melhor aptidão global chega no ótimo rapidamente então esse comportamento de convergência ao melhor é esperado com o tempo, por isso não vemos muitas diferenças entre melhor aptidão e melhor aptidão global após a quarta iteração.

Um outro ponto a se notar, que com a facilidade dessa instância uma execução consegue chegar em um valor ótimo logo na primeira iteração.

Pela simplicidade e facilidade na busca pela solução ótima nessa instância há poucas análises a se fazer, porém os gráficos validam o método criado para encontrar boas soluções para o problema do caxeiro viajante.

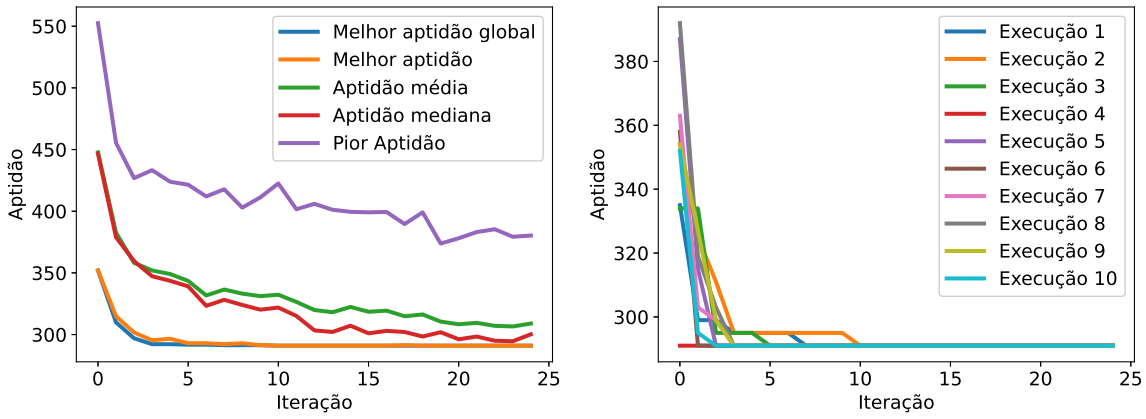

Figura 1: Execuções do melhor conjunto de parâmetros da instância lau15.

4.3.2 Instância sgb128

A instância sgb128 já é bem mais complexa que a análisada anteriormente, primeiramente o gráfico 2 mostra a execução do melhor conjunto de parâmetros nessa instância. É possível ver que a aptidão média e mediana não seguem muito a melhor aptidão global, isso pode ser devido a complexidade grande do problema, já que as formigas vão ter que explorar mais soluções, irão existir mais caminhos para se decidir qual incluir na solução, e também devido a diversificação aplicada pelos parâmetros, ja dito anteriormente. A melhor aptidão sempre fica próxima da melhor aptidão global nas diversas execuções. É importante notar que o gráfico apresenta a média das execuções das várias execuções, logo a melhora na melhor aptidão global nem sempre segue a melhor aptidão já que é a média que está sendo apresentado e não uma execução.

Então todas execuções com esse parâmetro seguem um bom ritmo de melhoramento de soluções. Comparando com a figura 3 e 4, elas tem uma diferença menor entre a melhor aptidão e a melhor aptidão global. Isso pode indicar que uma diversificação maior é feita no melhor conjunto de parâmetros, que reforça suposições anteriores. No geral todos parâmetros apresentaram um comportamento semelhante, porém o melhor parâmetro se distoa na questão da diferença entre melhor aptidão e melhor aptidão global.

Também é possível ver que na figura 2, as linhas de execuções são mais diversas comparada aos outros parâmetros.

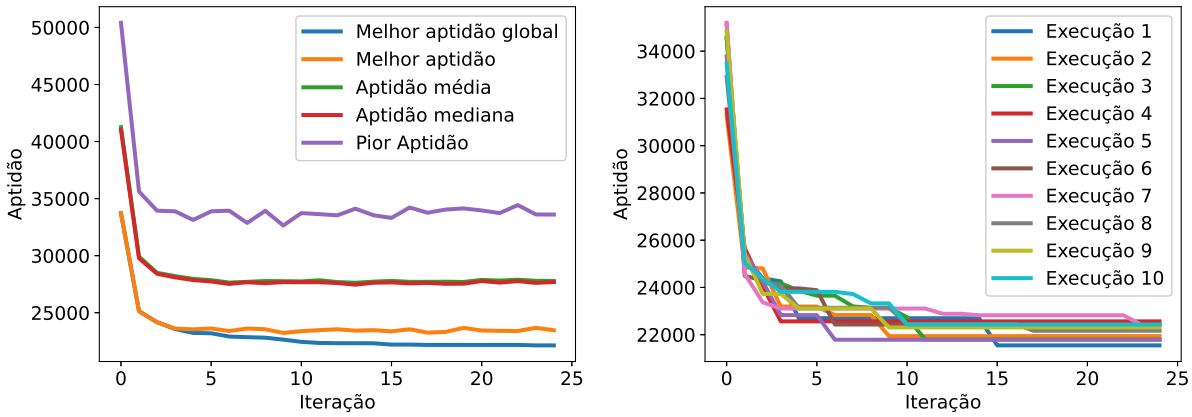

Figura 2: Execuções do melhor conjunto de parâmetros da instância sgb128.

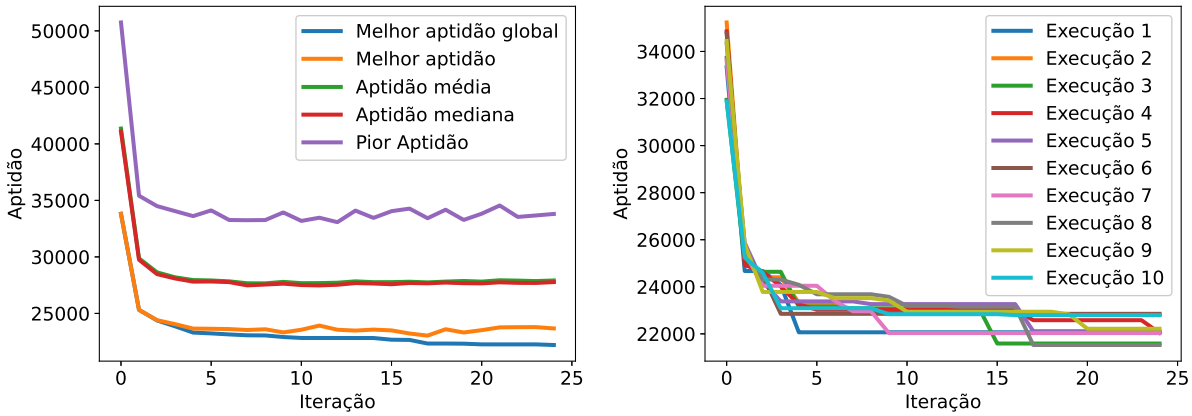

Figura 3: Execuções do segundo melhor conjunto de parâmetros da instância sgb128.

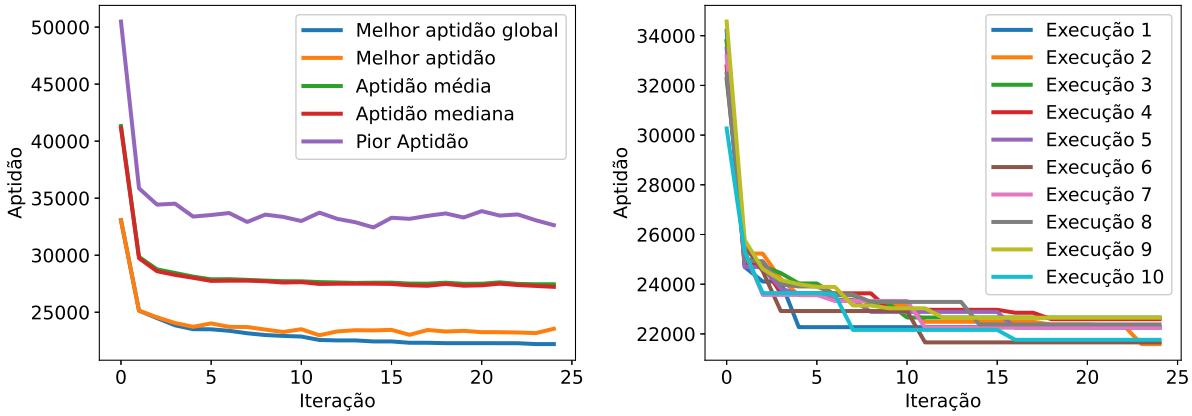

Figura 4: Execuções do terceiro melhor conjunto de parâmetros da instância sgb128.

5 Conclusão

Com esse trabalho foi possível ver na prática a efetividade de um dos primeiros algoritmos de otimização por colônia de formigas, o Ant System, e o problema do caxeiro viajante foi tratado, testando em 2 instâncias, uma com a qual foi possível obter a solução ótima, e uma outra que foi possível obter boas soluções.

Referências

[1] John E Bell and Patrick R McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48, 2004.

[2] Malika Bessedik, Rafik Laib, Aissa Boulmerka, and Habiba Drias. Ant colony system for graph coloring problem. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 786–791. IEEE, 2005.

[3] Ivan Brezina Jr and Zuzana Čičková. Solving the travelling salesman problem using the ant colony optimization. *Management Information Systems*, 6(4):10–14, 2011.

[4] Chi-Bin Cheng and Chun-Pin Mao. A modified ant colony system for solving the travelling salesman problem with time windows. *Mathematical and Computer Modelling*, 46(9-10):1225–1235, 2007.

[5] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[6] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81, 1997.

[7] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[8] Prasanna Kumar and G Raghavendra. A note on the parameter of evaporation in the ant colony optimization algorithm. In *International mathematical forum*, volume 6, pages 1655–1659, 2011.

[9] Ehsan Salari and Kourosh Eshghi. An aco algorithm for graph coloring problem. In *2005 ICSC Congress on computational intelligence methods and applications*, pages 5–pp. IEEE, 2005.

[10] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. An improved ant colony optimization for vehicle routing problem. *European journal of operational research*, 196(1):171–176, 2009.