

UNIVERSIDADE FEDERAL DE MINAS GERAIS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Ant Colony Optimization for the Longest Path Problem

Caio Felipe Zanatelli.
caio.zanatelli@idcc.ufmg.br

8 de Novembro de 2018

Resumo

Este trabalho possui como objetivo a criação de um modelo computacional, utilizando uma abordagem de inteligência coletiva via colônias de formigas (ACO – *Ant Colony Optimization*), para resolver o problema do caminho mais longo em um grafo ponderado. Os conceitos e detalhes do modelo criado serão abordados nas seções seguintes e, ademais, serão apresentados também os resultados obtidos através de testes em ambiente real e sua respectiva análise de sensibilidade.

1 Introdução

Como definido por [1], o algoritmo de Colônia de Formigas é um método dos campos de Inteligência de Enxames, Metaheurísticas e Inteligência Computacional. Trata-se, portanto, da simulação do comportamento de formigas na natureza para a resolução de problemas reais. Neste contexto, explora-se, principalmente, as formas de comunicação, baseadas em feromônios, utilizadas pelas formigas objetivando encontrar bons caminhos na busca por alimento em um determinado ambiente, sendo essa a inspiração para encontrar potenciais soluções para um problema.

Com isso, é necessário primeiro avaliar o comportamento de formigas na busca por alimento. Inicialmente, elas se espalham aleatoriamente em seu ambiente, dado que nenhum caminho é conhecido de antemão. Eventualmente, a fonte de comida será localizada e, nesse ponto, as formigas que a encontraram depositam feromônio nesse ambiente, de forma a sinalizar que o caminho em questão as levaram até a fonte. Dado que as formigas seguem as trilhas com maiores índices de feromônio, com o passar do tempo os caminhos que levam até a fonte de comida serão privilegiados, uma vez que as quantidades de feromônio neles depositados serão maiores. Em contrapartida, naturalmente, caminhos poucos percorridos perdem feromônio à medida que o tempo passa, devido à sua evaporação. Dessa forma, o ponto fundamental do algoritmo é baseado na comunicação das formigas, a qual ocorre através do feromônio depositado, uma vez que a tendência é que elas sigam caminhos cujo nível de feromônio é maior.

A partir dessa inspiração, surge a abordagem algorítmica intitulada *Colônias de Formigas*, cuja estratégia geral é construir soluções candidatas, de forma estocástica, para um determinado problema. Para tanto, tais soluções são geradas de maneira probabilística e têm suas qualidades avaliadas, gerando assim as quantidades de feromônio que serão depositadas em cada caminho, sendo às soluções melhores atribuída uma quantidade maior de feromônio, de forma que novas soluções criadas tenham a tendência de seguir pelos mesmos caminhos já avaliados como bons.

Por fim, neste trabalho apresentamos uma solução, utilizando ACO, para o problema do caminho mais longo em um grafo ponderado¹, o qual é enunciado a seguir. Dado um grafo $G(E, V)$, onde E e V são os conjuntos de arestas e vértices, respectivamente, uma função $w : E \rightarrow \mathbb{R}$ que atribui pesos a cada aresta e dois

¹Para mais detalhes: https://en.wikipedia.org/wiki/Longest_path_problem

vértices $u, v \in V$, denota-se como \mathcal{P} o conjunto de caminhos **simples**² partindo de u e chegando em v . O problema consiste em encontrar $P^* = e_1^*, e_2^*, \dots, e_k^*$ tal que

$$P^* = \operatorname{argmax}_{P \in \mathcal{P}} \sum_{e_i \in P} w(e_i),$$

ou seja, queremos encontrar o caminho simples de u a v que maximize o peso total do caminho. Trata-se de um problema NP-Difícil e, portanto, utilizaremos uma abordagem estocástica para resolvê-lo.

2 Modelagem do Problema

A modelagem proposta é baseada em um grafo ponderado nas arestas e a cada aresta foi atribuído também a quantidade de feromônio deixado naquele trajeto. Para a representação interna, foi escolhida a estrutura dicionário, onde o mapeamento é realizado da seguinte forma: a chave de entrada é o vértice origem, que mapeia para um conjunto de dicionários que representam os vértices nos quais o nó origem se liga, isto é, as arestas. Cada dicionário que representa uma aresta possui também um dicionário com duas entradas, mais especificamente o peso e a quantidade de feromônio, possibilitando, assim, o armazenamento de todas as informações necessárias para a aplicação do algoritmo ACO, além de fornecer um mecanismo simples de pesquisa. De forma a fornecer uma melhor visualização acerca da representação interna do grafo, abaixo é apresentado o dicionário resultante do exemplo ilustrado pela Figura 1, onde cada aresta possui um peso p e uma quantidade f de feromônio, indicado por p/f .

```
{
  1: { 2: { 'pheromone': 1, 'weight': 5 },
       { 3: { 'pheromone': 1, 'weight': 10 } }
     },
  3: { 2: { 'pheromone': 1, 'weight': 8 } },
  4: { 1: { 'pheromone': 1, 'weight': 1 },
       { 3: { 'pheromone': 1, 'weight': 2 } }
     }
}
```

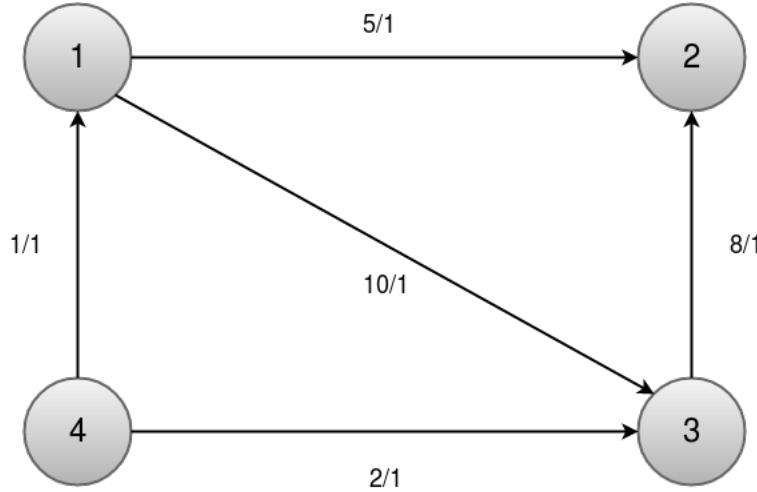


Figura 1: Exemplo de representação de um grafo.

²Um caminho simples é tal que não passa pelo mesmo vértice mais de uma vez.

3 Implementação

Como já abordado, o algoritmo ACO se baseia no comportamento de formigas na busca por alimento. Em suma, o algoritmo em questão é dado abaixo, onde $G = (E, V)$, A , s , t , *iterations* e *decay_rate* representam, respectivamente, o grafo, o conjunto de formigas, o vértice de origem, o vértice de destino, o número de iterações e a taxa de evaporação do feromônio para o ACO.

Algorithm 1 ACOLongestPath($G = (E, V)$, A , s , t , *iterations*, *decay_rate*)

```

1: InitPheromone()
2: for  $i$  in range(iterations) do
3:   GenerateSolutions( $G$ ,  $A$ ,  $s$ ,  $t$ )
4:   EvaluateSolutions( $G$ ,  $s$ ,  $t$ )
5:   UpdatePheromone( $G$ , decay_rate)
6: end for
```

O primeiro passo do algoritmo é inicializar os feromônios no grafo. Nesse trabalho, essa inicialização foi realizada simplesmente atribuindo um feromônio inicial igual a um (1) para todas as arestas, sendo esses valores atualizados com o decorrer do tempo na medida em que as formigas são distribuídas e encontram caminhos viáveis.

Em seguida, um processo iterativo se inicia, o qual é executado i vezes, onde i é o número de iterações definido pelo usuário. A primeira função executada é *GenerateSolutions()*, a qual é responsável por construir soluções probabilísticas para o problema baseado em um modelo de feromônio a cada iteração. A função *EvaluateSolutions()*, em seguida, avalia a qualidade das soluções construídas, de forma a atualizar a quantidade de feromônio nos caminhos de acordo com a qualidade obtida, sendo a última etapa realizada através da função *UpdatePheromone()*.

3.1 Geração de Soluções

Como supracitado, a construção de soluções é realizada através da função *GenerateSolutions()*, utilizada no pseudocódigo apresentado anteriormente. Em suma, essa função gera n soluções válidas para o problema, isto é, n caminhos simples do vértice s ao vértice t , de forma a maximizar o seu peso, onde n é o número de formigas. Vale ressaltar, portanto, que é gerado um caminho para cada formiga, de forma a distribuí-las no grafo. A cada iteração do algoritmo, essa função atualiza o grafo com os feromônios apropriados, que são calculados de acordo com a qualidade das soluções geradas, e retorna a melhor solução obtida. Os detalhes de implementação foram omitidos dos pseudocódigos que serão apresentados na sequência, mas estão devidamente comentados no código-fonte. Além disso, a comparação da qualidade das soluções é realizada através de uma função para o cálculo da *fitness* de um caminho em um grafo $G = (E, V)$. Por fim, é necessário salientar que o algoritmo aqui apresentado gera apenas soluções válidas – são realizadas várias tentativas para a geração do caminho e, caso não exista nenhum, é retornada uma lista vazia, a qual possui *fitness* zero associada e, com isso, se o problema não possuir solução, a saída será nula. O pseudocódigo em questão é apresentado na sequência.

Algorithm 2 GenerateSolutions($G = (E, V)$, A , s , t)

```

1: paths =  $\emptyset$ 
2: for all  $a_i \in A$  do
3:    $p = G.generatePath(s, t)$ 
4:   paths = paths  $\cup p$ 
5: end for
6: best_path =  $\emptyset$ 
7: for all  $p_i \in paths$  do
8:    $G.calculateFitness(p_i)$ 
9:   if  $p_i$  is better than best_path then
10:    best_path =  $p_i$ 
11:   end if
12:   source =  $p_i.pop(0)$ 
13:   for  $d_i \in p_i$  do
14:     increment_pheromone( $G$ , source,  $d_i$ , INCREMENT)
15:     source =  $d_i$ 
16:   end for
17:    $tmp = best\_path$ 
18:   source =  $tmp.pop(0)$ 
19:   for  $d_i \in tmp$  do
20:     increment_pheromone( $G$ , source,  $d_i$ , INCREMENT)
21:     source =  $d_i$ 
22:   end for
23: end for
24: return best_path

```

3.2 Avaliação de Soluções

Em seguida, tem-se a função *evaluateSolution()*, que recebe como parâmetro o melhor caminho encontrado pela função *generateSolution()*. O pseudocódigo em questão é apresentado abaixo.

Algorithm 3 EvaluateSolutions($G = (E, V)$, A , s , t , *best_path*)

```

1: path = GenerateSolutions( $G$ ,  $A$ ,  $s$ ,  $t$ )
2:  $G.calculate\_fitness(path)$ 
3: if path is better than best_path then
4:   best_path = path
5: end if

```

Nota-se, através dos trechos de código apresentados, que a avaliação da qualidade de soluções é baseada em uma função de cálculo de *fitness*. Essa função, inerente à estrutura de grafos criada, recebe como parâmetro um caminho, teoricamente simples e válido, e retorna sua *fitness*, que é dada por

$$fitness(P) = \sum_{e_i \in P} w(e_i),$$

onde P é um caminho no grafo $G = (E, V)$, isto é, $P \in E$, e w uma função que computa o peso de arestas $e_i \in P$. O pseudocódigo para a função em questão é apresentado abaixo.

Algorithm 4 CalculateFitness($G = (E, V)$, $path$)

```

1: if  $path == \emptyset$  then
2:   return 0
3: end if
4: if  $path$  is not valid then
5:   return NONE
6: end if
7:  $tmp = path$ 
8:  $sum\_weight = 0$ 
9:  $source = tmp.pop(0)$ 
10: for  $dest \in path$  do
11:    $weight = G.get\_weight(source, dest)$ 
12:    $sum\_weight = sum\_weight + weight$ 
13:    $source = dest$ 
14: end for
15: return  $sum\_weight$ 

```

3.3 Atualização de Feromônios

Por fim, a última rotina executada trata da atualização dos feromônios depositados pelas formigas nas trilhas, isto é, nas arestas dos grafos. Isso é realizado de forma igual para todas as arestas, mesmo o feromônio de cada caminho sendo proporcional à solução associada a ele. A evaporação do feromônio é dada pela seguinte fórmula:

$$pheromone'(e_i) = pheromone(e_i) \cdot (1 - decay_rate),$$

onde $pheromone'$ é o novo valor para o feromônio depositado na aresta e_i , indicado pelo peso $pheromone$, cuja atualização depende da taxa de evaporação $decay_rate$, a qual é configurada pelo usuário. A seguir apresentamos o pseudocódigo dessa função.

Algorithm 5 UpdatePheromone($G = (E, V)$, $decay_rate$)

```

1: for all  $e_i \in E$  do
2:    $G.set\_pheromone(e_i \cdot (1 - decay\_rate))$ 
3: end for

```

3.4 Estrutura do Projeto

O código foi modularizado de forma a proporcionar futuras extensões, tornando o algoritmo ACO algo genérico para outros problemas, sendo necessário alterar apenas a modelagem do problema e o cálculo da *fitness*, além da taxa de decaimento do feromônio, caso seja conveniente. A estrutura adotada é a seguinte:

- **main.py**: programa principal. Trata-se do módulo executado pelo usuário, sendo responsável por realizar o *parsing* dos parâmetros e por efetuar as chamadas para dar início à execução do ACO para a resolução do problema proposto.
- **graph.py**: define a classe *Graph*, responsável pela representação de um grafo através de dicionários. Fornece também métodos para manipulação da estrutura, tais como: adicionar uma aresta, verificar se um caminho é simples e/ou válido, gerar um caminho simples de um vértice s a um vértice t , calcular a *fitness* de um caminho, evaporar o feromônio nas arestas, etc.
- **aco.py**: define a classe *ACO*, responsável por executar o algoritmo de otimização por colônia de formigas sobre o grafo de entrada. As iterações e construções das soluções se encontram nesse módulo.
- **utils.py**: define a classe *Utils*, responsável por prover funções genéricas, como a escolha de um nó aleatório em um grafo e o cálculo de probabilidades.

- **ioutils.py**: define a classe *IOUtils*, responsável por fornecer funções relativas à manipulação de arquivos, como entrada e saída e manipulação de arquivos em formato CSV, por exemplo.

3.5 Execução e Parâmetros

De forma a tornar o projeto adaptável, optou-se por utilizar parâmetros por linha de comando para definir os pontos chave do modelo utilizado. A Tabela 1 apresenta a lista de parâmetros permitidos.

Tabela 1: Lista de parâmetros para a execução do algoritmo.

Parâmetro	Descrição	Valor Padrão
<code>--h, --help</code>	Exibe um menu de ajuda	—
<code>-input, --input</code>	Arquivo de entrada	—
<code>-output, --output</code>	Arquivo de saída	—
<code>-i, --iterations</code>	Número de iterações	10
<code>-n, --ants</code>	Número de formigas	100
<code>-e, --evaporation-rate</code>	Taxa de evaporação do feromônio	0.1
<code>-s, --seed</code>	Semente para a geração de números aleatórios	1

Como é possível observar na Tabela 1, alguns parâmetros são opcionais. Portanto, o código para a execução do programa é o seguinte, onde parâmetros entre colchetes indicam que são opcionais:

```
$ python main.py [-h] -input INPUT -output OUTPUT [-n ANTS] [-i ITERATIONS]
                  [-e EVAPORATION_RATE] [-s SEED]
```

3.6 Entrada e Saída

A entrada é dada por um arquivo que descreve um grafo direcionado e ponderado nas arestas. Cada linha do arquivo representa, portanto, uma aresta que liga um vértice u a um vértice v com peso p . O formato utilizado na entrada é dado por: $< u > < v > < p >$, ou seja, o vértice origem, o vértice destino e o peso da aresta, separados por um único espaço.

Em relação à saída, o código produz resultados relativos a cada iteração (geração) do algoritmo, os quais são direcionados para o arquivo de saída definido por parâmetro. Além disso, de forma a facilitar a análise experimental, a saída é dada em formato CSV, uma vez que é um meio razoável para a aplicação dos *scripts* utilizados para a geração de gráficos. O resultado produzido para cada instância do problema, portanto, é dado por três dados, separados por vírgula, são eles: identificador da iteração atual, melhor solução até a i -ésima geração e média das soluções encontradas até a i -ésima geração. A seguir apresentamos um exemplo de saída gerada pelo algoritmo.

```
1,711,711
2,711,678
3,735,697
4,735,689
5,735,673
6,735,639
7,735,632
8,735,624
9,735,624
10,735,620
15,735,598
16,735,592
17,735,591
18,735,594
```

4 Análise Experimental

Tendo em vista que o algoritmo de otimização por Colônia de Formigas, assim como qualquer algoritmo bioinspirado, depende fortemente de seus parâmetros, esta seção tem como objetivo apresentar os experimentos realizados para a validação de seu funcionamento, bem como a análise de sensibilidade dos parâmetros associados. Os experimentos introduzidos a seguir foram executados de forma distribuída em máquinas e servidores do Laboratório de Redes sem Fio (WINET/DCC/UFMG – *Wireless Networking Laboratory*).

A metodologia utilizada foi baseada em testes exaustivos de forma a avaliar todas as possíveis combinações de parâmetros para todas as bases, com o intuito de encontrar a melhor configuração possível, além de avaliar o impacto de cada um dos parâmetros separadamente. Em suma, os seguintes parâmetros foram variados: número de formigas, número de iterações e taxa de decaimento de feromônio, fixando dois deles e variando o terceiro. Ainda, por se tratar de um processo estocástico, cada experimento foi executado em média 30 vezes, mantendo o controle sobre a semente do algoritmo de geração de números aleatórios, permitindo assim a reprodução dos resultados.

4.1 Base *graph1*

Para a base *graph1*, foram variados todos os possíveis parâmetros: número de formigas, número de iterações e taxa de decaimento do feromônio. Os respectivos gráficos e análises são apresentados a seguir.

4.1.1 Variação do Número de Iterações

Primeiramente, fixou-se a taxa de decaimento do feromônio com valor padrão igual a 0.1. Em seguida, variaramos o número de iterações de 50 a 500, de forma a mensurar o impacto desse parâmetro no processo de otimização via ACO. Ainda, optou-se por representar a curva da qualidade das soluções geradas a cada iteração e, com isso, é possível observar a melhor solução obtida até a *i*-ésima iteração. A quantidade de formigas escolhida foi 50, 100 e 500, sendo as três curvas representadas no mesmo gráfico para fins comparativos.

O gráfico da Figura 2 ilustra a variação da quantidade de iterações, mantendo a taxa de decaimento como 0.1. Como é possível observar, a partir da geração 20 as melhores soluções começam a convergir, sem alterações muito significativas em suas qualidades, indicando que outro parâmetro também deve ser alterado para que haja uma melhora na solução obtida. Nesse caso, o parâmetro em questão seria a taxa de evaporação do feromônio, que faz com que a curva dos gráficos se altere, uma vez que as trilhas formadas pelas formigas se perdem mais rapidamente, proporcionando assim uma exploração maior do espaço de busca. Nota-se, ainda, que a quantidade de formigas utilizadas na amostragem influencia diretamente na qualidade das soluções encontradas: os experimentos mostraram que o aumento do número de formigas faz com que as soluções melhorem, entretanto a partir de 500 formigas as soluções convergem para valores muito próximos e, portanto, o esforço computacional para melhora-las não vale a pena.

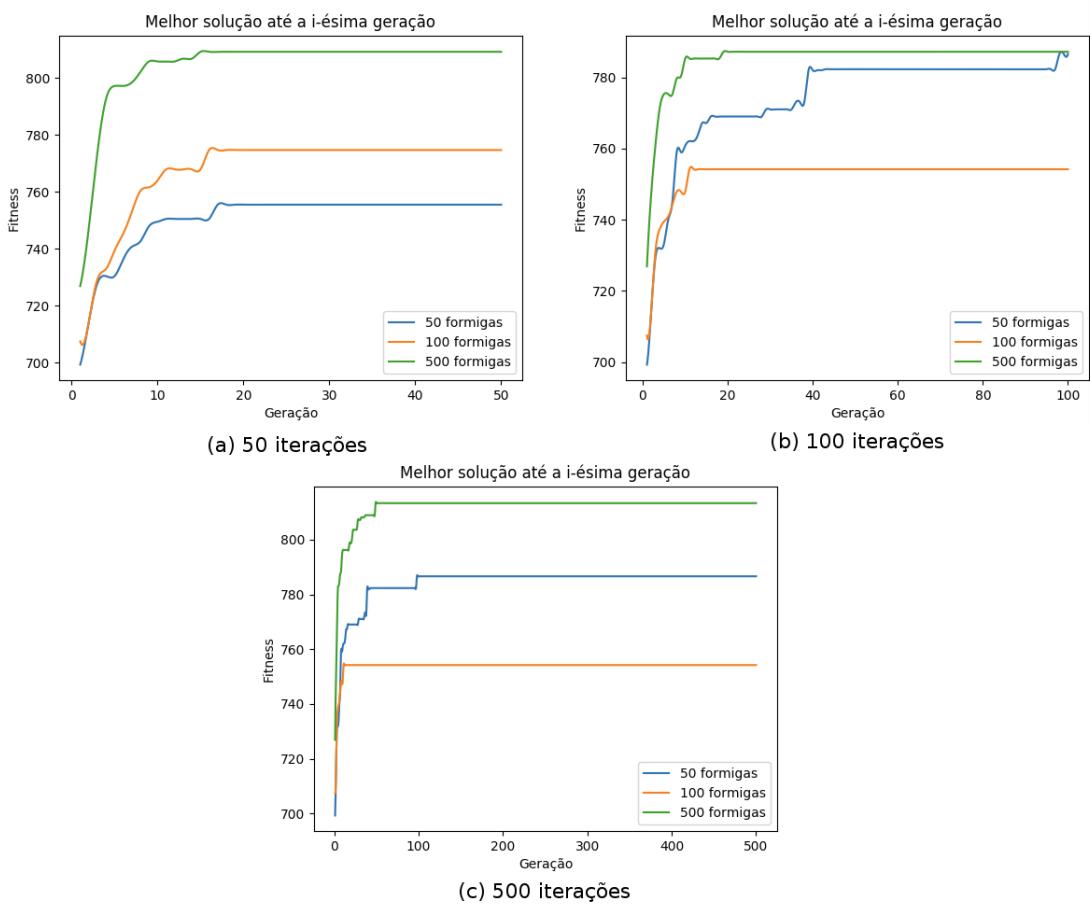


Figura 2: Experimento 1 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500 e taxa de evaporação 0.1 (base *graph1*).

4.1.2 Variação da Taxa de Decaimento do Feromônio

Avaliando os resultados obtidos na subseção anterior, observamos que o número de iterações influencia na qualidade das soluções, entretanto faz-se necessária a alteração do parâmetro referente à taxa de decaimento do feromônio para que possamos explorar melhor o espaço de busca. Ainda, a partir da geração 20 as soluções convergiam e, portanto, 100 iterações é um bom limiar para essa base de dados. Nesse contexto, abaixo são apresentados os resultados obtidos da variação da taxa de evaporação do feromônio (0.1–0.5) e avaliamos o comportamento das soluções nesse cenário. Vale ressaltar, porém, que todos os parâmetros foram combinados, de forma a encontrar a melhor configuração possível. Os resultados obtidos para cada configuração, bem como os melhores resultados, são apresentados na seção seguinte.

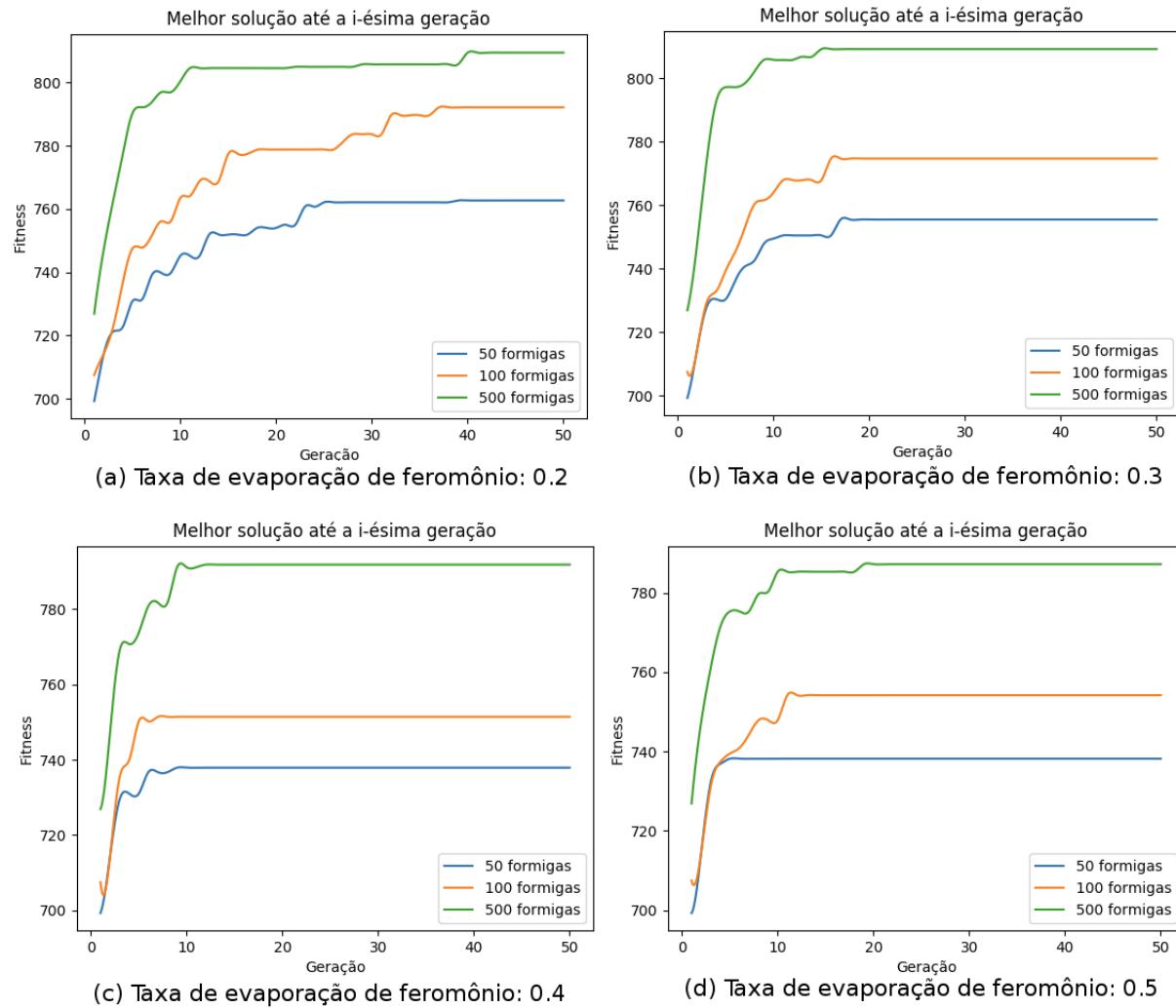


Figura 3: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 50 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph1*).

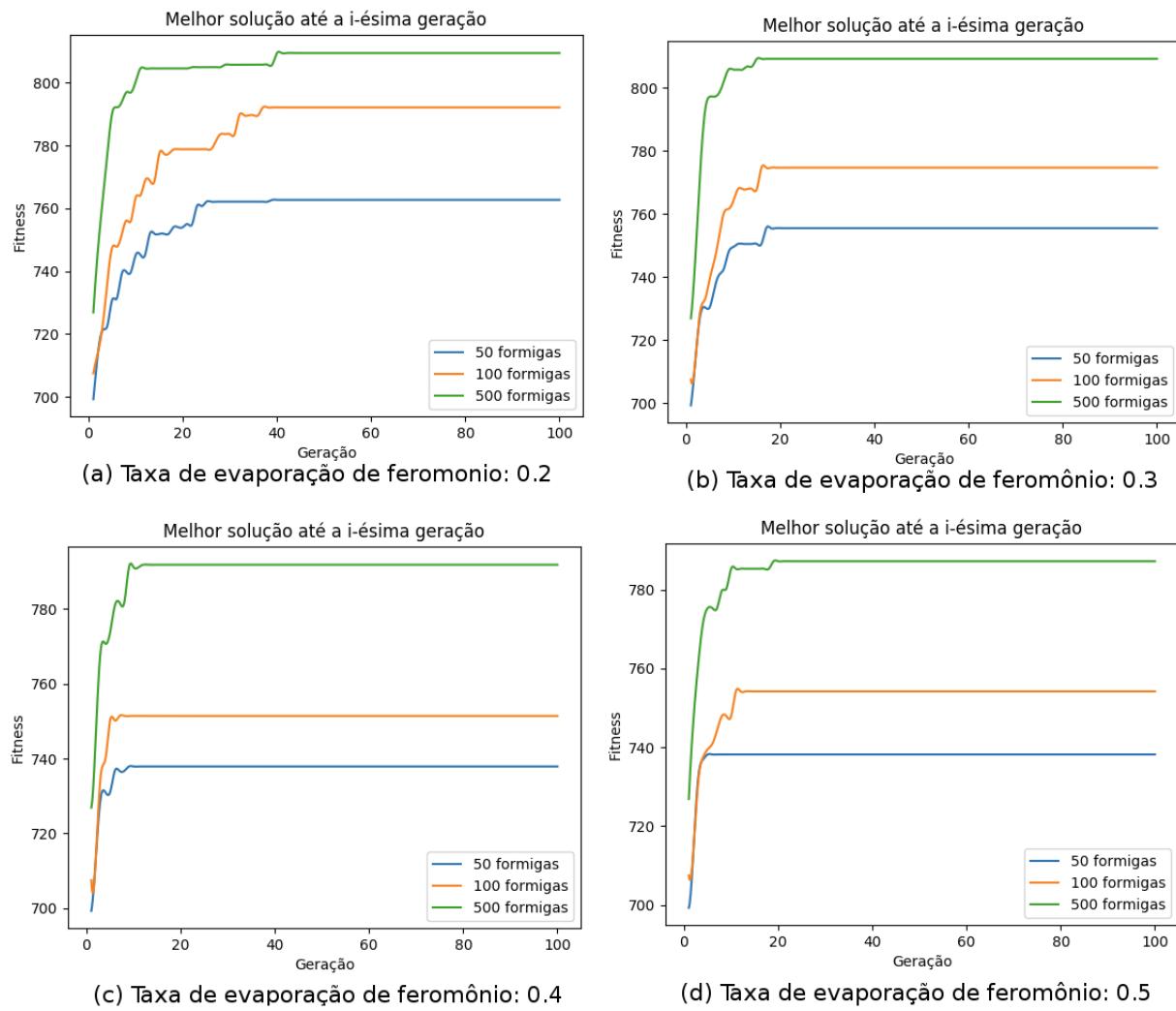


Figura 4: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 100 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph1*).

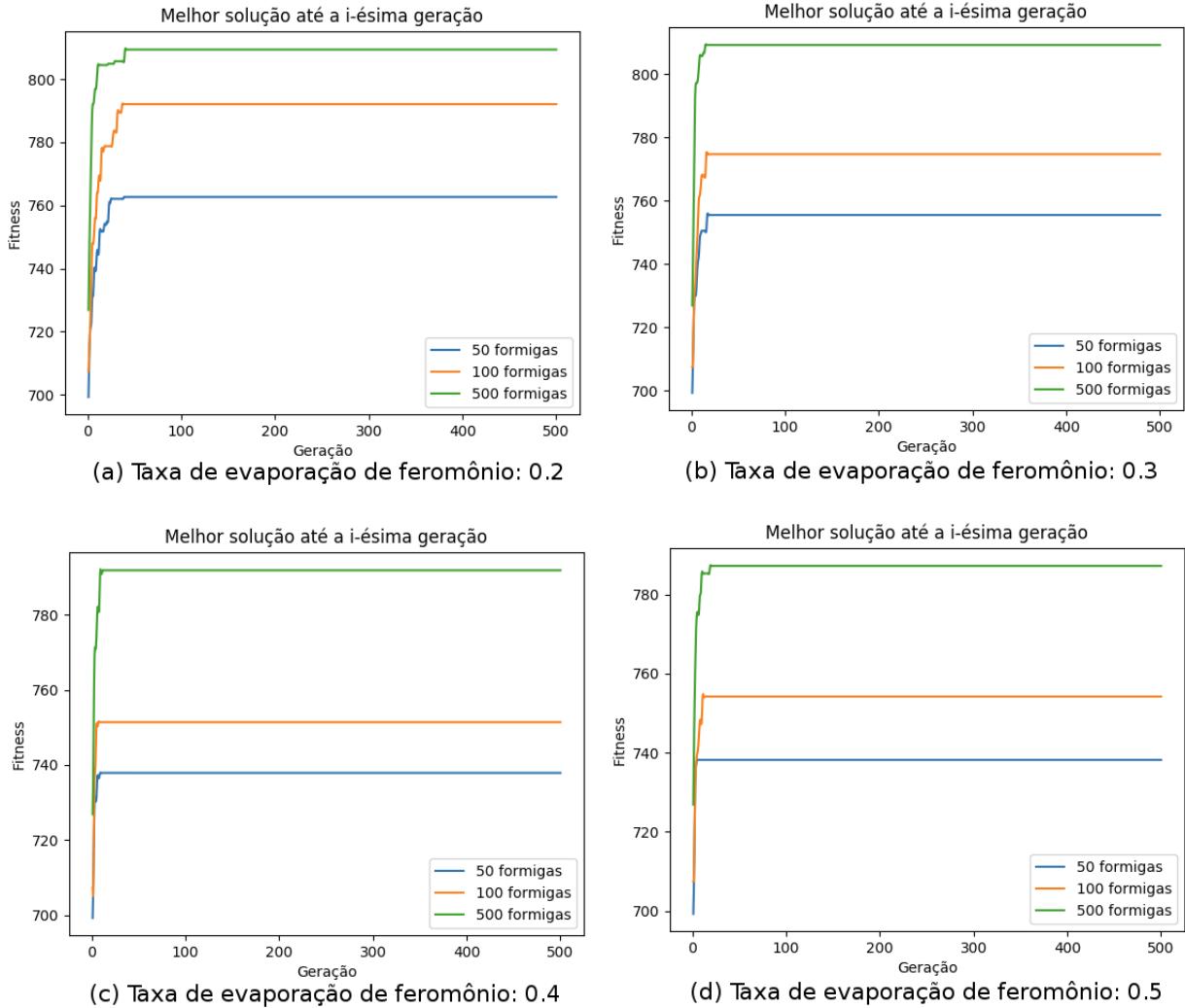


Figura 5: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 500 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph1*).

Através dos gráficos das Figuras 3, 4 e 5, em contraste com a Figura 2, é possível observar que uma taxa de evaporação de feromônio baixa, como 0.1, faz com que a convergência da melhor solução seja mais lenta, estagnando em um melhor local. Por outro lado, uma taxa muito alta, como 0.5, não melhora tanto as soluções, uma vez que as trilhas formadas pelas formigas se desfazem muito rapidamente. Avaliando os valores aplicados a essa taxa, nota-se que 0.3, aliado a 100 iterações, se mostrou um valor adequado, uma vez que a solução encontrada se aproxima bem da esperada para essa base de dados.

4.2 Base *graph2*

Para a base *graph2*, assim como para *graph1*, todos os possíveis parâmetros também foram variados: número de formigas, número de iterações e taxa de decaimento do feromônio. Os respectivos gráficos e análises são apresentados a seguir.

4.2.1 Variação do Número de Iterações

Inicialmente, fixou-se a taxa de evaporação do feromônio com valor padrão igual a 0.1. Em seguida, variou-se o número de iterações de 50 a 500, objetivando mensurar o impacto desse parâmetro na qualidade das soluções obtidas pelo ACO. Ainda, como já salientado, os gráficos apresentam as curvas para 50, 100 e 500 formigas, de forma a contrastar o impacto da quantidade de formigas no grafo.

Os resultados obtidos estão ilustrados na Figura 6. Nela, é possível observar que, para essa base de dados, a convergência das soluções é mais lenta, uma vez que o crescimento da curva que representa a *fitness* da melhor solução até a i -ésima geração possui um crescimento mais suave se comparado àquela da base *graph1*. Ainda, como esperado, a quantidade de formigas utilizadas influenciou na qualidade da solução, pois, com isso, a quantidade de feromônio depositado nas trilhas é maior e, portanto, a probabilidade de que as formigas encontrem soluções melhores e optem por esses caminhos também é maior. Avaliando a Figura 6c, notamos também que as soluções convergem a partir da geração 100 e, assim, o esforço computacional para avaliar mais iterações não é interessante. Entretanto, avaliamos na próxima subseção o comportamento da curva ao variar a taxa de evaporação do feromônio para toda a faixa de iterações sugerida.

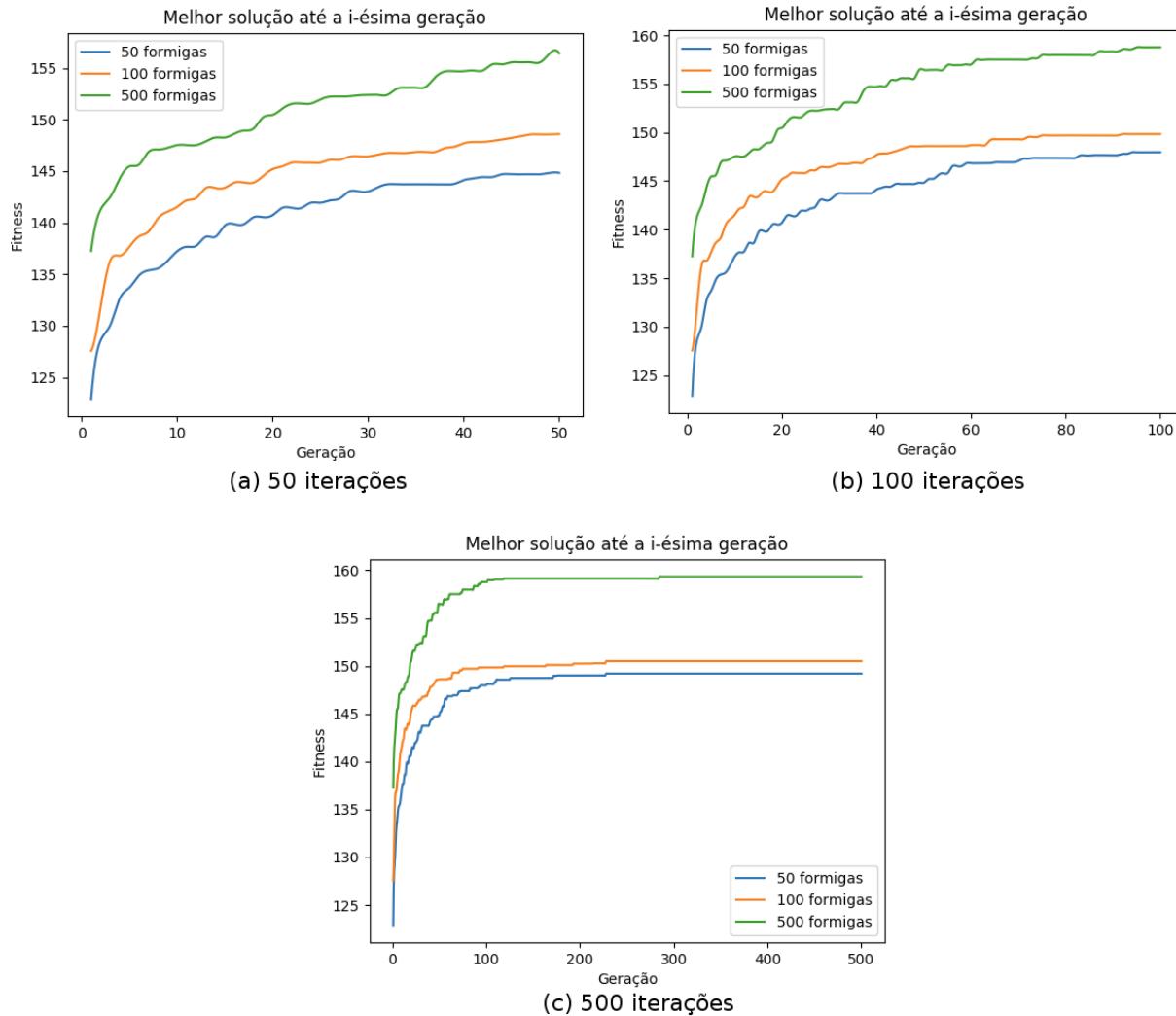


Figura 6: Experimento 1 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500 e taxa de evaporação 0.1 (base *graph2*).

4.2.2 Variação da Taxa de Decaimento do Feromônio

Avaliando os resultados obtidos na subseção anterior, observamos que o número de iterações influencia na qualidade das soluções, entretanto faz-se necessária a alteração do parâmetro referente à taxa de decaimento do feromônio para que possamos explorar melhor o espaço de busca. Ainda, a partir da geração 20 as soluções convergiam e, portanto, 100 iterações é um bom limiar para essa base de dados. Nesse contexto, abaixo são apresentados os resultados obtidos da variação da taxa de evaporação do feromônio (0.1–0.5) e avaliamos o comportamento das soluções nesse cenário. Vale ressaltar, porém, que todos os parâmetros foram combinados, de forma a encontrar a melhor configuração possível. Os resultados obtidos para cada configuração, bem como os melhores resultados, são apresentados na seção seguinte.

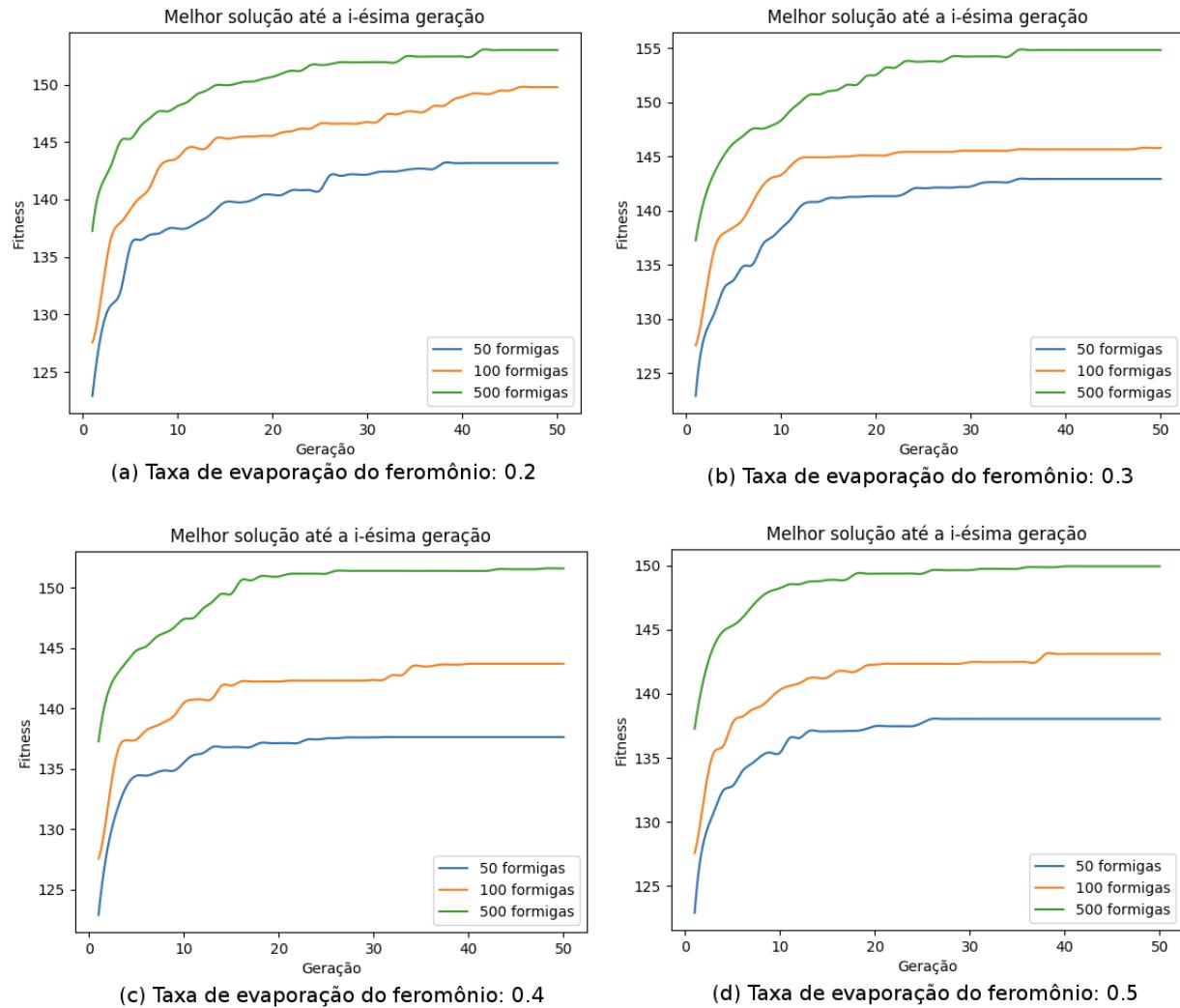


Figura 7: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 50 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph2*).

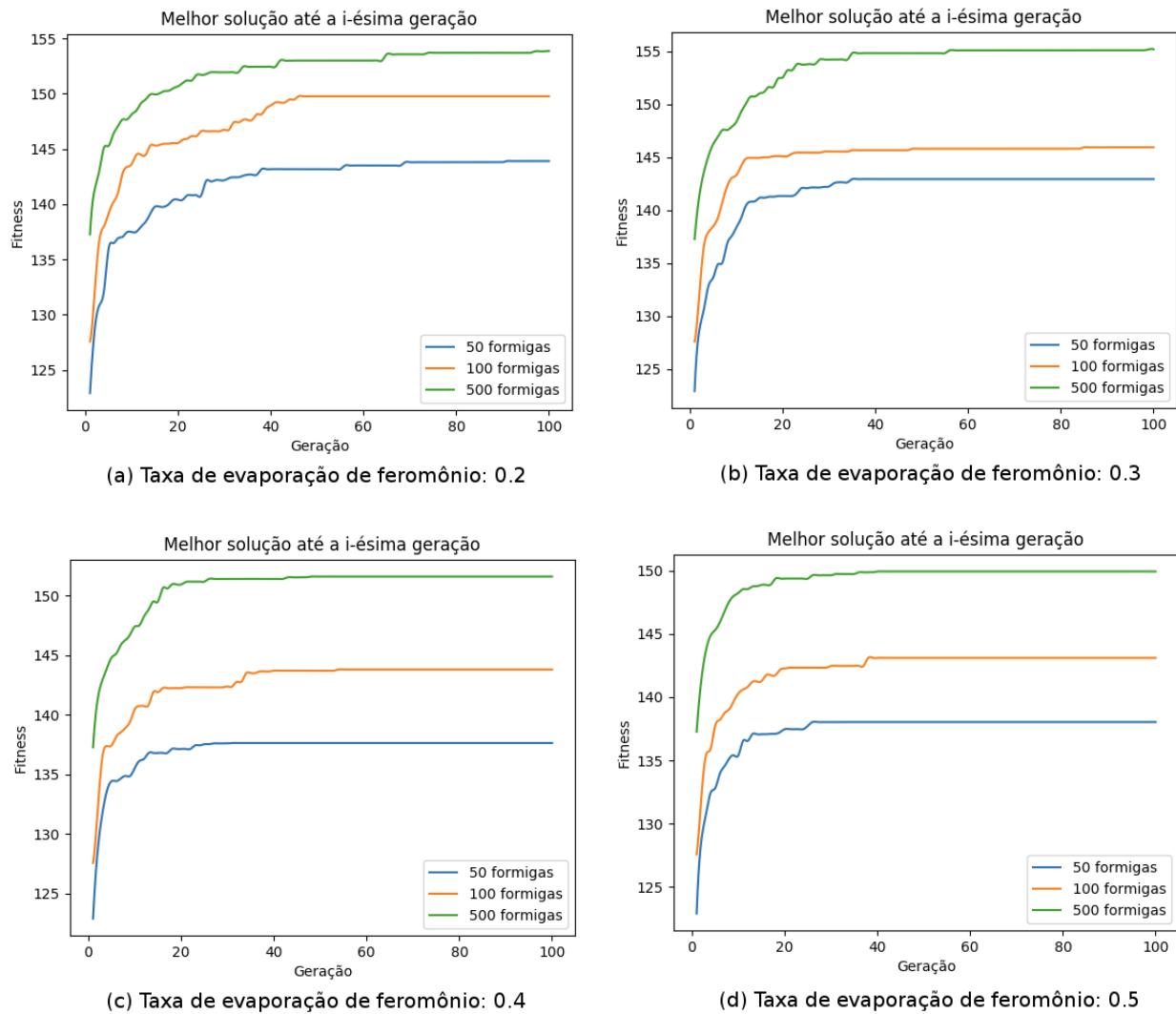


Figura 8: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 100 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph2*).

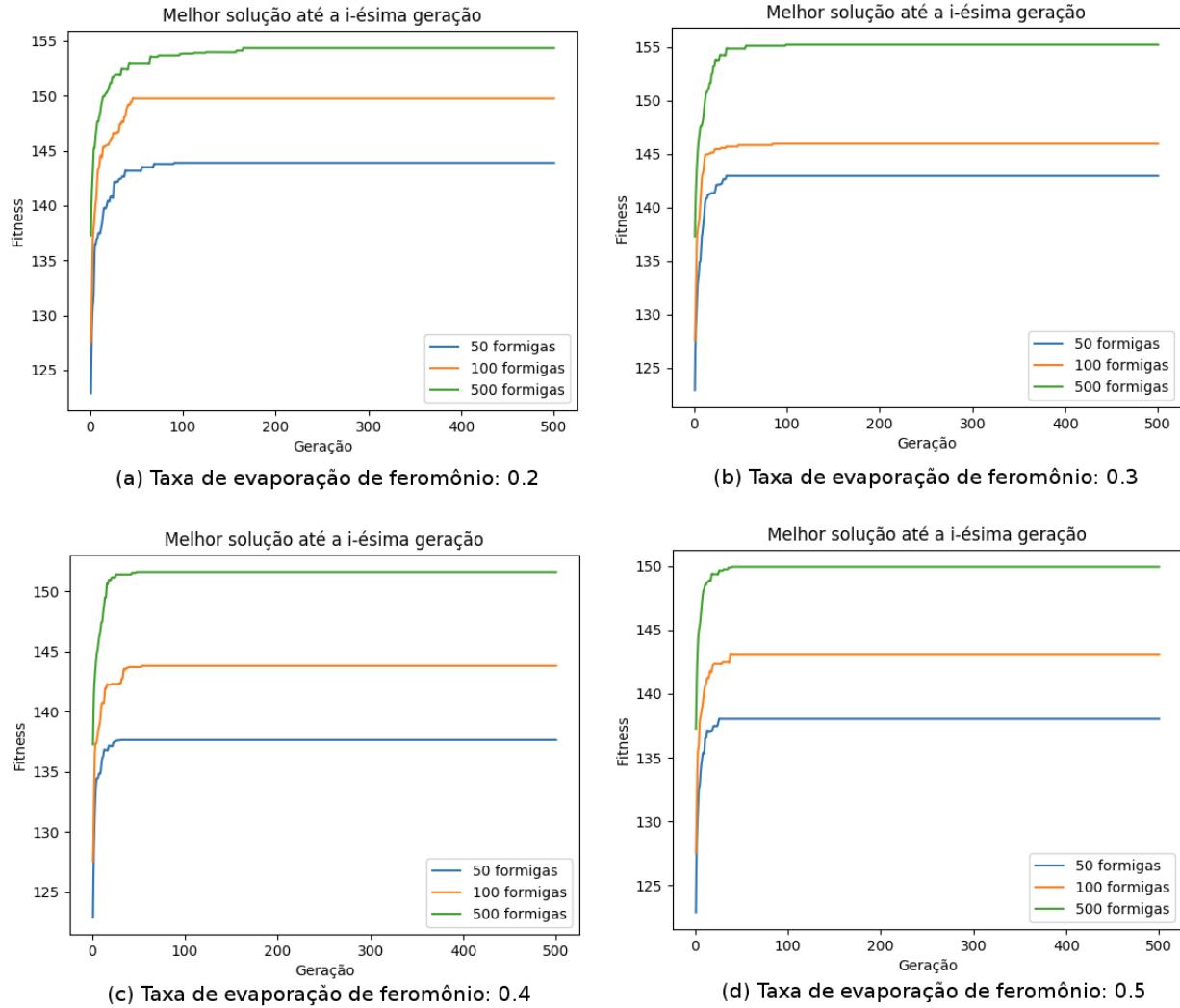


Figura 9: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 500 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph2*).

Através dos gráficos das Figuras 7, 8 e 9, em contraste com a Figura 6, nota-se que, assim como ocorreu na base *graph1*, uma baixa taxa de evaporação de feromônio torna a convergência da solução mais lenta, uma vez que a probabilidade de estagnação em um ótimo local é maior. Por outro lado, uma alta taxa de evaporação faz com que as trilhas se percam mais rapidamente, dificultando a localização daquelas que levam à melhor solução. Dessa forma, avaliando os valores aplicados a essa taxa, pode-se afirmar que, mais uma vez, o valor 0.3 se mostrou o mais adequado para a base em questão, resultando em soluções melhores quando aplicada com 100 iterações.

4.3 Base *graph3*

Por fim, foram avaliados também os parâmetros para a base *graph3*, entretanto com testes menos exaustivos, uma vez que o tempo computacional é bastante elevado, dada a quantidade de arestas presentes no grafo associado. Os resultados encontrados são apresentados na sequência.

4.3.1 Variação do Número de Iterações

Assim como nos experimentos anteriores, para mensurar o impacto da variação do número de iterações do ACO, fixou-se a taxa de evaporação do feromônio com valor padrão igual a 0.1. Em seguida, variou-se o número de iterações, apresentando a curva de soluções para 50, 100 e 500 formigas, de forma a mensurar o impacto desse parâmetro para a base em questão.

Os resultados mostram que a solução converge a partir da geração 40, porém com uma curva um pouco caótica. Isso pode ser justificado pelo tamanho da entrada, de forma que várias possíveis soluções são encontradas e, dado uma taxa menor de evaporação de feromônio, as soluções começam divergindo bastante mas seguem um mesmo padrão a partir de uma determinada geração.

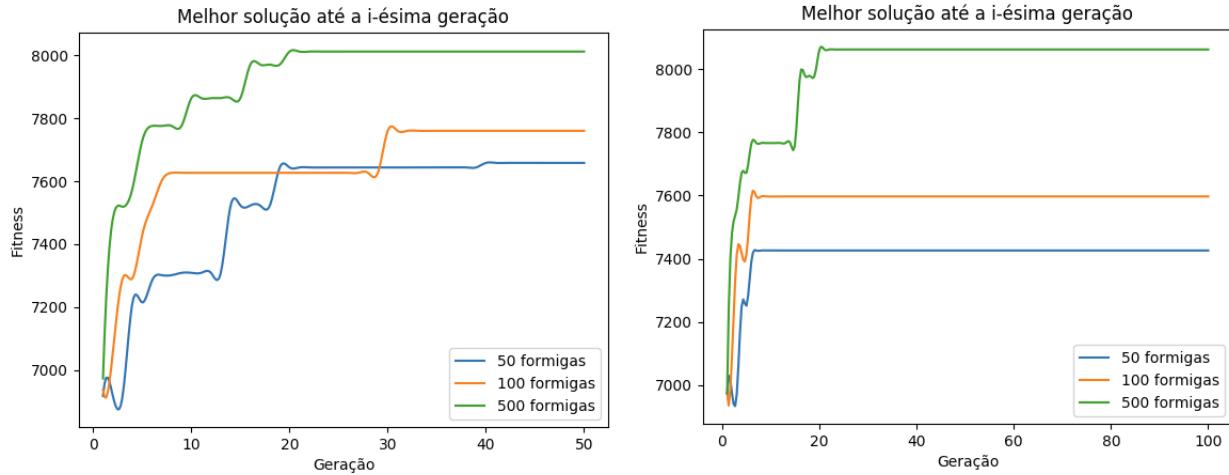


Figura 10: Experimento 1 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500 e taxa de evaporação 0.1 (base *graph3*).

4.3.2 Variação da Taxa de Decaimento do Feromônio

Observando os resultados obtidos na subseção anterior, nota-se que a solução não melhora significativamente a partir da geração 40 e, com isso, é inviável executar testes com mais iterações, dada a complexidade do grafo de entrada e o tempo computacional para isso. Utilizamos, para fins de análise, portanto, 50 e 100 iterações e variamos a taxa de evaporação do feromônio em 0.2, 0.3, 0.4 e 0.5.

As figuras 11 e 12 ilustram, respectivamente, a curva das melhores soluções para 50 e 100 iterações, variando a taxa de decaimento de feromônio em 0.2, 0.3, 0.4 e 0.5, para 50, 100 e 500 formigas. Curiosamente, as taxas de evaporação de 0.1 e 0.2 foram as que produziram melhores resultados, como é possível observar nos gráficos da Figura 10, Figura 11(a) e 12(a). Isso pode ter ocorrido por se tratar de um grafo razoavelmente grande e, com isso, o espaço de busca inicial é bem maior se comparado às demais bases. Dessa forma, com uma exploração melhor do espaço de soluções, aliada a uma taxa pequena de decaimento de feromônio, é possível testar vários caminhos por formigas distintas e, por isso, um número elevado de formigas é necessário para essa base. A partir da geração 40, então, as soluções convergem e não há melhorias significativas a partir deste ponto.

5 Resultados Obtidos

Uma vez realizada a análise de sensibilidade dos parâmetros pertinentes ao ACO para as três bases de dados, esta seção tem como objetivo apresentar os melhores resultados obtidos na variação de todos os parâmetros em cada base.

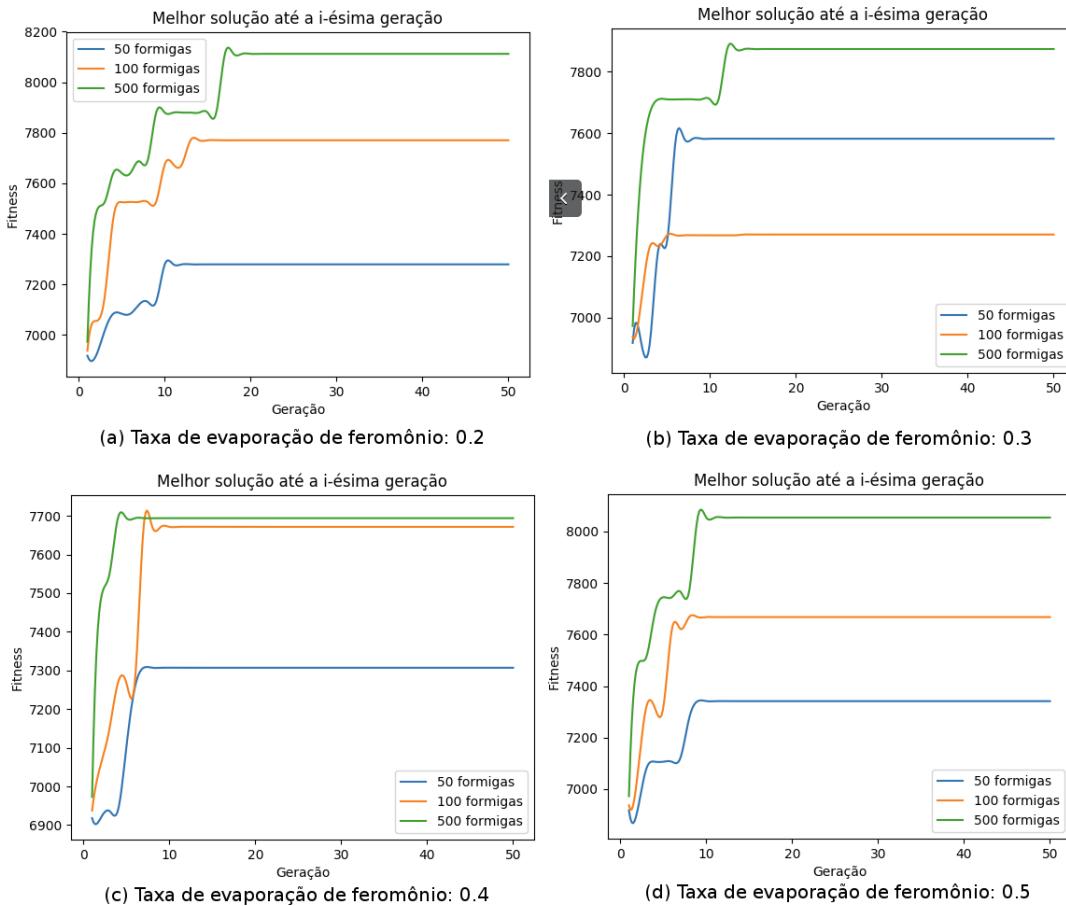


Figura 11: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 50 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph3*).

5.0.1 Base *graph1*

Como abordado na seção anterior, para a base *graph1* todos os parâmetros foram variados, isto é, o número de formigas, o número de iterações e a taxa de evaporação do feromônio, realizando todas as combinações possíveis para os valores estipulados. As Tabelas 2–6 apresentam os melhores resultados obtidos para cada configuração.

Tabela 2: Base *graph1* – Melhores resultados com taxa de evaporação igual a 0.1.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.1	823
50	100	0.1	835
50	500	0.1	835
100	50	0.1	858
100	100	0.1	858
100	500	0.1	858
500	50	0.1	856
500	100	0.1	856
500	500	0.1	856

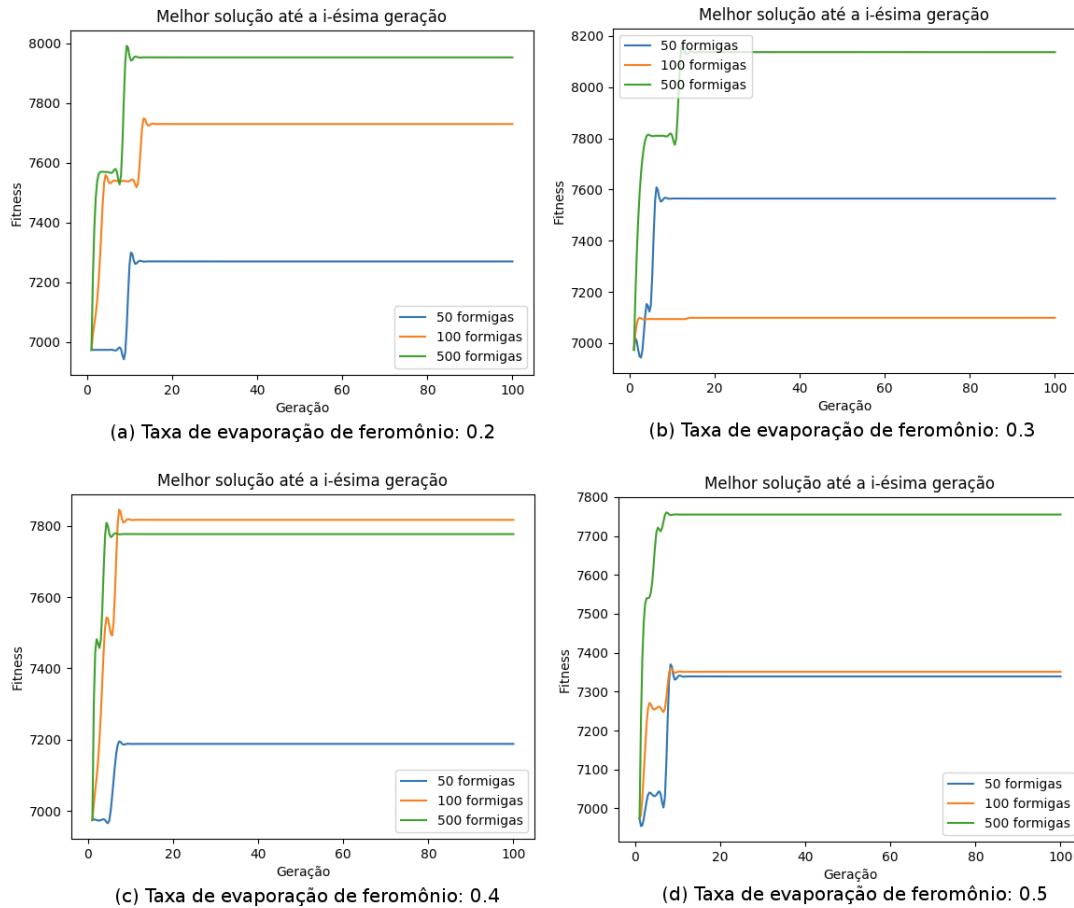


Figura 12: Experimento 2 – Geração x Melhor solução até a i -ésima geração. Configuração: quantidade de formigas 50, 100 e 500; número de iterações 100 e taxa de evaporação $\in \{0.2, 0.3, 0.4, 0.5\}$ (base *graph3*).

Tabela 3: Base *graph1* – Melhores resultados com taxa de evaporação igual a 0.2.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.2	833
50	100	0.2	833
50	500	0.2	833
100	50	0.2	821
100	100	0.2	821
100	500	0.2	821
500	50	0.2	855
500	100	0.2	855
500	500	0.2	855

Tabela 4: Base *graph1* – Melhores resultados com taxa de evaporação igual a 0.3.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.3	806
50	100	0.3	806
50	500	0.3	806
100	50	0.3	845
100	100	0.3	845
100	500	0.3	845
500	50	0.3	863
500	100	0.3	863
500	500	0.3	863

Tabela 5: Base *graph1* – Melhores resultados com taxa de evaporação igual a 0.4.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.4	807
50	100	0.4	807
50	500	0.4	807
100	50	0.4	824
100	100	0.4	824
100	500	0.4	824
500	50	0.4	845
500	100	0.4	845
500	500	0.4	845

Tabela 6: Base *graph1* – Melhores resultados com taxa de evaporação igual a 0.5.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.4	792
50	100	0.4	792
50	500	0.4	792
100	50	0.4	838
100	100	0.4	838
100	500	0.4	838
500	50	0.4	828
500	100	0.4	828
500	500	0.4	828

5.0.2 Base *graph2*

Avaliamos também, a seguir, os melhores resultados obtidos em cada configuração para a base de dados *graph2*. As soluções são ilustradas pelas Tabelas 7–11.

Tabela 7: Base *graph2* – Melhores resultados com taxa de evaporação igual a 0.1.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.1	168
50	100	0.1	168
50	500	0.1	168
100	50	0.1	167
100	100	0.1	167
100	500	0.1	167
500	50	0.1	167
500	100	0.1	167
500	500	0.1	167

Tabela 8: Base *graph2* – Melhores resultados com taxa de evaporação igual a 0.2.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.2	159
50	100	0.2	159
50	500	0.2	159
100	50	0.2	167
100	100	0.2	167
100	500	0.2	167
500	50	0.2	167
500	100	0.2	167
500	500	0.2	167

Tabela 9: Base *graph2* – Melhores resultados com taxa de evaporação igual a 0.3.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.3	163
50	100	0.3	163
50	500	0.3	163
100	50	0.3	164
100	100	0.3	164
100	500	0.3	164
500	50	0.3	167
500	100	0.3	167
500	500	0.3	167

Tabela 10: Base *graph2* – Melhores resultados com taxa de evaporação igual a 0.4.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.4	159
50	100	0.4	159
50	500	0.4	159
100	50	0.4	156
100	100	0.4	156
100	500	0.4	156
500	50	0.4	167
500	100	0.4	167
500	500	0.4	167

Tabela 11: Base *graph2* – Melhores resultados com taxa de evaporação igual a 0.5.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.5	158
50	100	0.5	158
50	500	0.5	158
100	50	0.5	167
100	100	0.5	167
100	500	0.5	167
500	50	0.5	167
500	100	0.5	167
500	500	0.5	167

5.0.3 Base *graph3*

Por fim, avaliamos também as melhores soluções obtidas na variação de todos os parâmetros para a base *graph3*. Os resultados encontrados estão ilustrados nas Tabelas 12–16.

Tabela 12: Base *graph3* – Melhores resultados com taxa de evaporação igual a 0.1.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.1	8042
50	100	0.1	8042
50	500	0.1	8042
100	50	0.1	8332
100	100	0.1	8332
100	500	0.1	8332
500	50	0.1	8235
500	100	0.1	8235
500	500	0.1	7492

Tabela 13: Base *graph3* – Melhores resultados com taxa de evaporação igual a 0.2.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.2	8039
50	100	0.2	8039
50	500	0.2	8039
100	50	0.2	8002
100	100	0.2	8002
100	500	0.2	8002
500	50	0.2	8272
500	100	0.2	8272
500	500	0.2	8272

Tabela 14: Base *graph3* – Melhores resultados com taxa de evaporação igual a 0.3.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.3	7970
50	100	0.3	7970
50	500	0.3	7970
100	50	0.3	8141
100	100	0.3	8141
100	500	0.3	8141
500	50	0.3	8326
500	100	0.3	8326
500	500	0.3	7953

Tabela 15: Base *graph3* – Melhores resultados com taxa de evaporação igual a 0.4.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.4	8004
50	100	0.4	8004
50	500	0.4	8004
100	50	0.4	8198
100	100	0.4	8198
100	500	0.4	8198
500	50	0.4	8064
500	100	0.4	8064
500	500	0.4	8160

Tabela 16: Base *graph3* – Melhores resultados com taxa de evaporação igual a 0.5.

Num. Formigas	Num. Gerações	Taxa de Evaporação de Feromônio	Melhor Solução
50	50	0.4	7339
50	100	0.4	7339
50	500	0.4	—
100	50	0.4	7984
100	100	0.4	7351
100	500	0.4	—
500	50	0.4	—
500	100	0.4	7755
500	500	0.4	—

5.1 Melhores Resultados

Na subseção anterior foram apresentados todos os resultados obtidos ao variar todos os parâmetros para as três bases de dados. Filtrando as configurações de parâmetros e as respectivas soluções encontradas pelo ACO, ilustrados pelas Tabelas 2–16, a Tabela 17 ilustra os melhores resultados obtidos nesses experimentos.

Para a base *graph1*, obtemos uma solução de peso 863, sendo 990 o valor esperado. Já para a base *graph2*, obtemos o valor 168, igual ao ótimo esperado. Por fim, para a base *graph3*, encontramos uma solução de peso 8332, porém não sabemos de antemão o valor ótimo esperado para esse teste. Entretanto, com base nos resultados gerais obtidos, pode-se afirmar que foi possível obter uma boa aproximação para as bases de teste, o que indica que a convergência do algoritmo está correta, isto é, ele aproxima bem a solução para um problema de natureza NP-Difícil.

Tabela 17: Base *graph3* – Melhores resultados encontrados nos experimentos (bases *graph1*, *graph2* e *graph3*).

Base	Num. Formigas	Num. Gerações	Taxa de Evaporação	Melhor Solução
<i>graph1</i>	500	50	0.3	863
<i>graph2</i>	50	50	0.1	168
<i>graph3</i>	100	500	0.1	8332

6 Conclusão

Este trabalho apresentou uma modelagem para o problema *Longest Path* (caminho mais longo) em um grafo ponderado nas arestas através de um algoritmo bioinspirado baseado no comportamento de Colônias de Formigas (ACO – *Ant Colony Optimization*). Assim, o projeto foi de grande importância para o esclarecimento e fixação dos conceitos vistos em sala de aula. Além disso, muitos problemas difíceis de otimização, cuja modelagem se baseia em grafos, podem ser resolvidos através de ACO e, com isso, certamente o algoritmo desenvolvido poderá ser reutilizado para outros fins, uma vez que todos os módulos foram desenvolvidos de forma a reduzir o acoplamento do projeto como um todo.

Ainda, através desse trabalho foi possível praticar a modelagem de problemas, algo que muitas vezes é abstrato demais e, portanto, foi de grande relevância compreender melhor o funcionamento de um algoritmo baseado em Colônias de Formigas. Entretanto, a maior dificuldade encontrada refere-se à execução dos testes, uma vez que o algoritmo requer um custo computacional elevado, dependendo fortemente do tamanho da entrada e dos parâmetros configurados. Portanto, realizar a implementação e todos os testes para o tunelamento adequado dos parâmetros foi um desafio nesse trabalho, porém permitiu a visualização da importância e o impacto de cada um deles.

Referências

- [1] Browlee, J. *Clever Algorithms*. LuLu, 1st edition, 2011.
- [2] de França, F. O., Zuben F. J. V., de Castro, L. N.. *Max min ant system and capacitated p-medians: Extensions and improved solutions*. *Informatica (Slovenia)*, 29(2): 163–172.