UNIVERSIDADE FEDERAL DE MINAS GERAIS
Documentação
Trabalho Prático 2 de Programação Natural

Marco Túlio Motta de Jesus
marcotmotta@gmail.com

Resumo. *Este trabalho tem como objetivo desenvolver um estudo detalhado do algoritmo de Colônia de Formigas para resolução de problemas em grafos, em específico, o de caminho simples mais longo. A análise consiste em diversas execuções considerando variações de parâmetros buscando identificar os maiores impactos gerados por eles de acordo com as bases de dados propostas.*

Palavras Chave. *Colônia de Formigas, Caminho Mais Longo, Grafo*

1. Introdução

Os fundamentos do **algoritmo de otimização da colônia de formigas** quando relacionados com diversos outros aspectos da área de conhecimento da Computação Natural são fortemente impactantes no universo da Ciência da Computação como um todo.

Esse trabalho trata de um problema típico de **teoria de grafos** onde precisamos encontrar um caminho simples de comprimento máximo num dado grafo. Para isso, simularemos o comportamento de uma colônia de formigas e, através dessa representação, buscaremos obter caminhos ótimos ou quase ótimos.

2. Metodologia

Como é de extrema importância nesse tipo de problema se realizar uma boa representação do **genótipo** dos **indivíduos**, foi escolhida neste trabalho uma representação em forma de dicionário para os grafos, dado que a linguagem de programação usada para desenvolver o algoritmo foi Python e, para as formigas, armazenamos em forma de vetor o caminho percorrido por cada uma delas.

Além dos **vértices**, armazenamos no dicionário do grafo o **peso das arestas** correspondentes e, também, o valor do **feromônio** deixado pelas formigas como forma de identificação presente em cada transição. Cada chave do dicionário representa um vértice. Nos valores dessas chaves estão contidas as outras informações citadas acima para cada

outro vértice que possui ligação com o vértice da chave atual. Dessa forma, temos rápido acesso a qualquer informação presente no grafo e podemos manipulá-lo de uma maneira eficiente.

Como dito, a representação das soluções é através de vetores que armazenam os vértices visitados pela formiga correspondente. Como temos acesso aos dados do grafo apenas indicando um vértice, o ato de calcular o peso das arestas contidas nesse caminho, por exemplo, é muito simples. Com isso, podemos ranquear as soluções obtidas pelas formigas (***fitness***) e, ao final da execução, descobrir qual foi o melhor resultado.

Outro aspecto importante da implementação é a **função de probabilidade** usada para indicar quais arestas serão escolhidas durante os movimentos das formigas e, também, a **função de atualização do feromônio**. Para isso, usaremos os seguintes termos:

Para a atualização do feromônio:

- F: Quantidade de feromônio atual na aresta;
- E: Taxa de evaporação do feromônio;
- Custo_Caminho: Soma dos pesos das arestas do caminho da formiga;
- Custo_Total: Soma do peso de todas as arestas do grafo.

Para a função de probabilidade:

- F: Quantidade de feromônio atual na aresta;
- P: Peso da Aresta;
- P_Total: Peso das arestas possíveis de serem visitadas;
- F_Total: Feromônio presente nas arestas possíveis de serem visitadas.

Com isso, definimos:

1. Atualização do Feromônio em determinada aresta:

$$F = F * (1 - E) + Custo\_Caminho / Custo\_Total$$

2. Função de Probabilidade para escolher a aresta *a*:

$$Prob_a = F * P / P\_Total * F\_Total$$

Para lidar com soluções inválidas, não são consideradas soluções que não atendam os requisitos de vértice (começar no 1 e terminar no N) apesar do algoritmo ainda gerá-las. Não são gerados caminhos que passam pelo mesmo vértice duas vezes, visto que esse caso já é tratado no próprio processo de escolha de arestas durante o movimento da formiga.

3. Testes

Utilizando o método de otimização de colônia de formigas, teremos majoritariamente 3 parâmetros para variar com o objetivo de testar o nosso algoritmo, sendo eles:

1. Número de Formigas;
2. Número de Gerações;
3. Taxa de Evaporação do Feromônio.

Começaremos os testes utilizando a segunda base de dados disponibilizada visto que esta é consideravelmente menor e, por isso, torna-se apropriada para os primeiros testes. Todos as configurações serão rodadas um número considerável de vezes para evitar resultados discrepantes ou não coerentes.

1. Variando Número de Formigas (N)

Valor fixo para número de gerações: 50
Valor fixo para taxa de evaporação do feromônio: 0.1

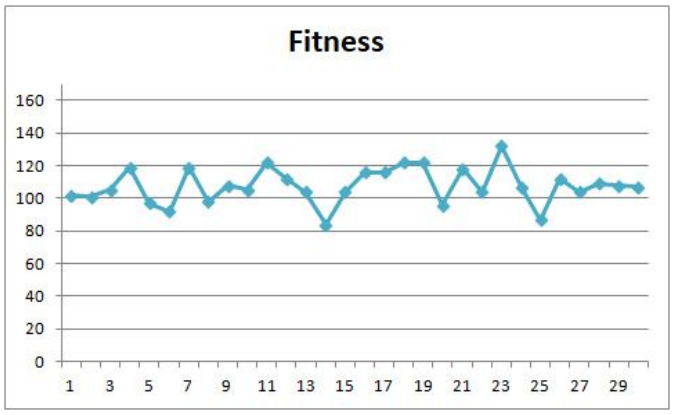

Gráfico 1. *Fitness média de 107 com N = 10.*

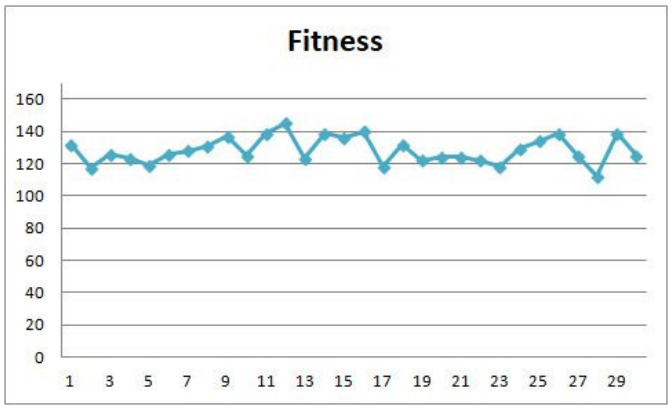

Gráfico 2. *Fitness média de 128 com N = 50.*

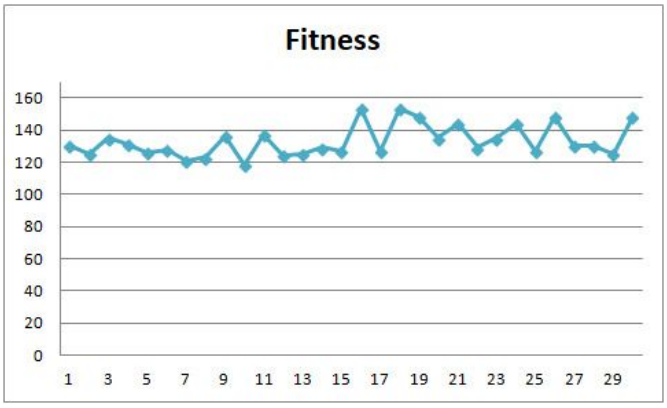

Gráfico 3. *Fitness média de 133 com N = 100.*

Podemos perceber que aumentar a quantidade de formigas influencia bastante no resultado. Por outro lado, quando elevamos demais esse parâmetro, os ganhos começam a não ser tão significativos e pode não mais valer a pena dado o tempo a mais que o algoritmo consome para executar.

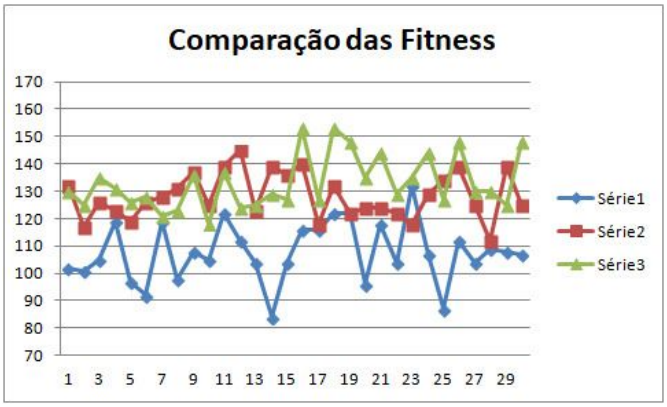

Gráfico 4. *Comparação das Fitness mostradas anteriormente.*

2. Variando Número de Gerações (N)

Valor fixo para número de formigas: 50
Valor fixo para taxa de evaporação do feromônio: 0.1

A mesma análise feita acima foi aplicada ao número de gerações. Fixando os outros parâmetros e variando esse utilizando as mesmas taxas mostradas acima, obtemos os seguintes resultados:

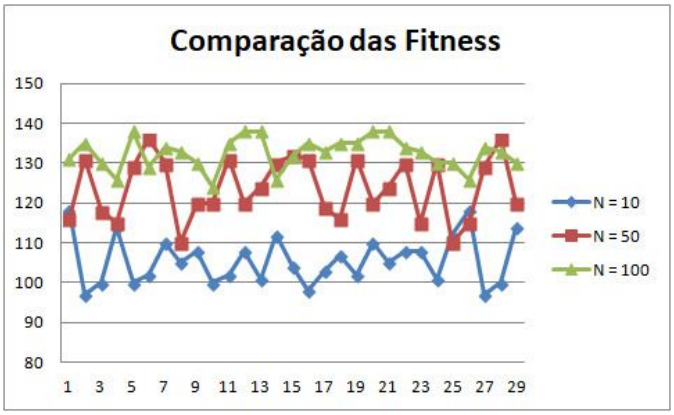

Gráfico 5. *Comparação das Fitness variando o número de gerações. Médias de 105, 123 e 132 para N = 10, 50 e 100, respectivamente.*

3. Variando Taxa de Evaporação do Feromônio (N)

Valor fixo para número de formigas: 50
Valor fixo para número de gerações: 50

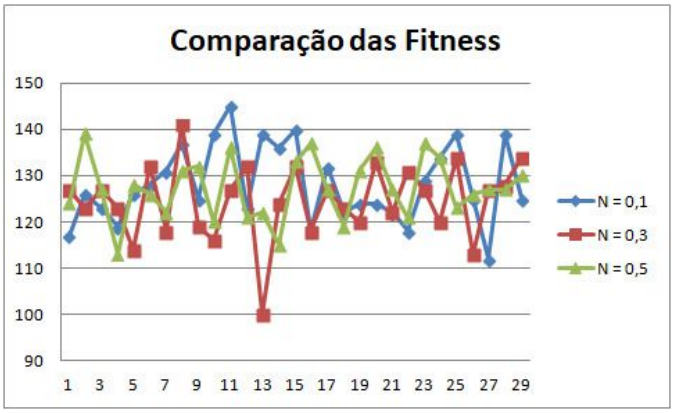

Gráfico 6. *Comparação das Fitness variando a taxa de evaporação do feromônio. Médias de 128, 124 e 127 para N = 0.1, 0.3 e 0.5.*

Podemos notar que, da forma como o algoritmo foi implementado, variar a taxa de evaporação do feromônio, pelo menos nessa velocidade, faz pouca ou quase nenhuma diferença no resultado final. Isso pode se dar possivelmente pelo fato do feromônio ser atualizado apenas após o movimento das formigas que formaram uma solução válida. De qualquer forma, usar um valor pequeno como 0.1, por exemplo, parece ser melhor do que usar valores maiores.

Observando os gráficos e a implementação no geral, pode-se dizer que os resultados ficaram relativamente dentro do esperado. A utilização de mecanismos como

elitismo, por exemplo, poderia ser uma boa alternativa para buscar o aumento das melhores fitness obtidas ao longo das gerações e pode ser implementado em projetos futuros.

Seguem abaixo os resultados obtidos nas bases de dados 1 e 3:

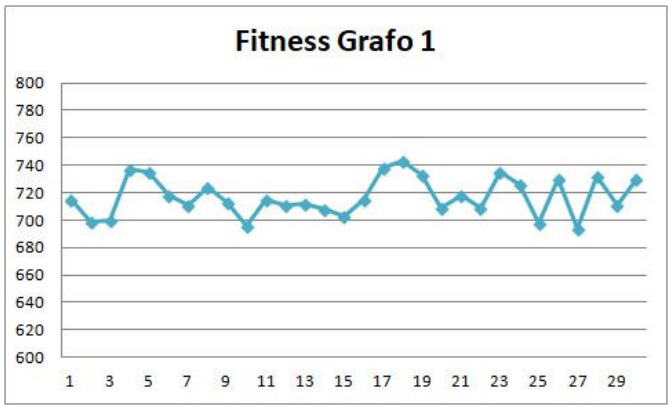

Gráfico 7. *Fitness dos indivíduos no grafo 1.*
Número de Formigas = 100, Número de Gerações = 100, Taxa de Evaporação = 0,1
e média de 717.

Gráfico 8. *Fitness dos indivíduos no grafo 3.*
Número de Formigas = 50, Número de Gerações = 50, Taxa de Evaporação = 0,1
e média de 7037.

4. Conclusão

Implementar e analisar esse trabalho foi uma tarefa complexa e cheia de desafios mas, apesar de tudo, foi muito gratificante chegar ao final com um algoritmo pronto.

Com base nos testes feitos, podemos concluir que a implementação cumpriu seu objetivo, dado que apesar das médias não tão altas, obtivemos algumas fitness satisfatórias durante as execuções.

O estudo dos parâmetros foi bastante interessante. Resumidamente, toda a teoria e principalmente a prática aprendida com relação a otimização de colônia de formigas e

teoria de grafos será de grande valia para o futuro, tanto nessa área como em outras tangentes.

5. Bibliografia

1. Marco Dorigo - The Ant System: Optimization by a colony of cooperating agents
2. John Kenneth Scholvin - Approximating the Longest Path Problem with Heuristics: A Survey
3. Colônia de Formigas (Otimização) https://pt.wikipedia.org/wiki/Col%C3%B4nia_de_formigas_(otimiza%C3%A7%C3%A3o)
4. Moodle da disciplina de Computação Natural 2018/2