Universidade Tecnológica Federal do Paraná – UTFPR
Câmpus Ponta Grossa – Paraná – Brasil

Relatório de Atividade

Ponta Grossa / 2022

LISTA DE EXERCÍCIOS 2

Júlio César Guimarães Costa[1]; Francisco Correa Neto[1];
juliocosta@alunos.utfpr.edu.br; franet@alunos.utfpr.edu.br;
[1]Universidade Tecnológica Federal do Paraná – UTFPR – Ponta Grossa – Paraná

Resumo

Neste relatório foi abordada a implementação e os testes do algoritmo da colônia de formigas (ACO) para a problemática do caixeiro viajante e então documentado neste documento os resultados obtidos.

Palavras-chave: *metaheurística; otimização; algoritmo da colônia de formigas; ACO.*

1. Introdução

O problema do Caixeiro Viajante que será abordado neste trabalho consiste em um problema logístico, NP-Completo que é descrito da seguinte maneira: Dado um conjunto de cidades geolocalizadas, um caixeiro deve atravessar todas as cidades percorrendo a menor distância possível.

Com base então no problema apresentado foi proposta a implementação do algoritmo bio inspirado Colônia de Formigas, para tal foram analisados dois comportamento das formigas em âmbito social: a criação de um caminho eficiente entre a fonte de alimento e o ninho, e a interação microscópica, através do feromônio deixado no ambiente para a indicação do caminho percorrido.

Neste relatório, apresentamos a resolução da segunda lista de exercícios da disciplina de Metaheurísticas de Otimização Bio-Inspiradas, onde nesta, foram utilizadas técnicas de análise e estatística na aplicação do algoritmos ACO para a problemática do Caixeiro Viajante.

2. Arquivos Utilizados

Foram utilizados 3 arquivos contendo dados sobre a latitude e longitude de 38, 634 e 1979 cidades respectivamente, representado as informações de três localidades: djibout, Luxemburgo e Omã.

3. Ferramentas Utilizadas

Foi escolhida a linguagem de programação python para a implementação pela fácil manipulação de dados que a linguagem proporciona. Para criação de um ambiente isolado para o desenvolvimento do estudo foi utilizada a ferramenta anaconda environments, onde foram então instaladas as bibliotecas utilizadas: numpy (funções aritméticas), pandas (manipulação e visualização) e tqdm (monitoramento do processo).

4. Pré Processamento

Com o ambiente preparado, em um primeiro momento foram removidas todas as cidades que se repetiam nos arquivos, o que resultou no decréscimo do número de cidades do arquivo de Luxemburgo de 980 para 634, ainda no pré processamento foi implementado um programa para criação de uma tabela de distâncias euclidianas entre as cidades, buscando desta forma permitir que durante o processamento do ACO fosse possível realizar somente uma consulta, e assim economizar processamento, para tal foi utilizado o conceito da matriz triangular sem a diagonal principal, essa escolha decorreu do fato da distância entre duas cidades não ser afetada pela ordem de origem e chegada, a remoção da diagonal principal por fim ocorreu devido a distância entre uma cidade e ela mesma ser sempre 0 e não ser relevante para o estudo.

O algoritmo portanto teve o arquivo da figura 1 como entrada e o arquivo representado pela figura 2 como saída, sendo este representado da seguinte maneira, primeira cidade, segunda cidade e distância euclidiana:

Figura 1 - djibout source file

djibout - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

ID	LATITUDE	LONGITUDE
1	11003.611100	42102.500000
2	11108.611100	42373.888900
3	11133.333300	42885.833300
4	11155.833300	42712.500000
5	11183.333300	42933.333300
6	11297.500000	42853.333300
7	11310.277800	42929.444400
8	11416.666700	42983.333300
9	11423.888900	43000.277800
10	11438.333300	42057.222200
11	11461.111100	43252.777800
12	11485.555600	43187.222200
13	11503.055600	42855.277800
14	11511.388900	42106.388900
15	11522.222200	42841.944400
16	11569.444400	43136.666700
17	11583.333300	43150.000000
18	11595.000000	43148.055600

Fonte: Arquivos da disciplina.

Figura 2 - djibout distance table

djiboutDist - Bloco d

Arquivo Editar Form

```
1,2,290.99
1,3,794.00
1,4,628.71
1,5,850.05
1,6,806.30
1,7,881.98
1,8,972.87
1,9,991.28
1,10,437.07
1,11,1237.92
1,12,1186.97
1,13,903.39
1,14,507.79
1,15,903.18
1,16,1178.84
1,17,1197.22
1,18,1201.22
1,19,1205.38
1,20,901.74
```

Fonte: Os Autores

5. Algoritmo ACO

Foi então implementado o algoritmo ACO e utilizados os seguinte parâmetros para os testes:

- Relevância do Feromônio (α): 1.
- Relevância da Distância (β): 4.
- Convergência: 50 Gerações ou até que a diferença entre o fitness máximo e mínimo da geração fosse inferior a 0.01.
- Índice de Evaporação: 0,5.
- Número de "caminhos de formigas" por geração: número de cidades, com cada uma iniciando em uma cidade.

O cálculo do fitness foi realizado da seguinte maneira:

Figura 3 - Fórmula Fitness

$$fitness = 1 - \frac{distPercorrida}{\sum_{f=1}^{nFGeracao} distPercorrida}$$

Fonte: Os Autores

6. Análise de Eficiência do Algoritmo

Em um primeiro momento para entendermos melhor a complexidade do problema abordado foi realizado o cálculo da quantidade total de caminhos possíveis (arestas) entre as cidades para obter a quantidade de passos que a formiga deveria atravessar, logo após foi executado outro teste na execução do programa, foram analisados desta vez o número de iterações para que uma única formiga percorresse todas as cidades.

Tabela 1 - Informações sobre os arquivos

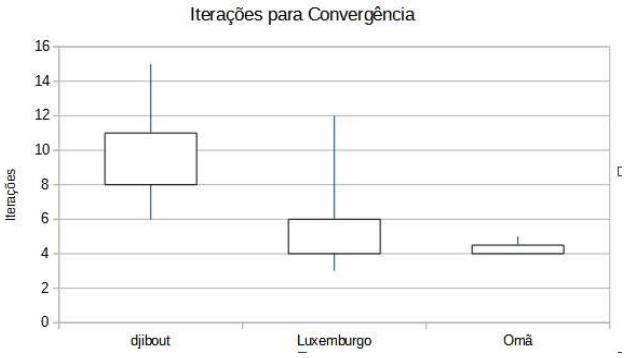
	Número de cidades	Número de Arestas	Tempo por Formiga
Djibout	38	703	< 1 segundos
Luxemburgo	634	200.661	3 segundos
Omã	1.979	1.957.231	7 segundos

Fonte: Os Autores

7. Resultados

Dos resultados obtidos o primeiro consiste no gráfico box plot relativo ao número de iterações necessários para convergência, importante salientar que devida a complexidade e o tempo necessário para os testes o número de testes foram 5 testes para Djibout e Luxemburgo e somente 3 para Omã, e os resultados estão apresentados na figura 4.

Figura 4 - Iterações para Convergência



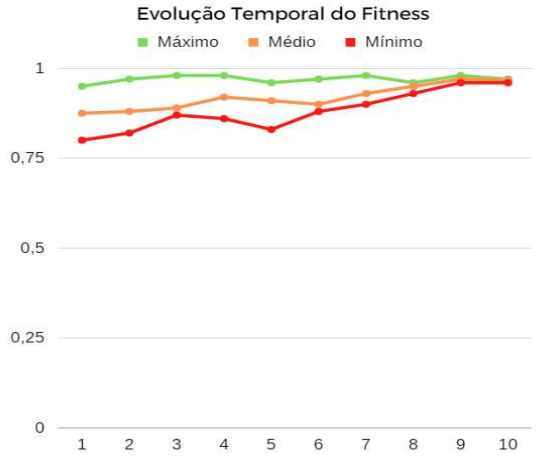
Fonte: Os Autores

A média do número de iterações até a convergência foi:

- Djibout: 10 iterações;
- Luxemburgo: 6 iterações;
- Omã: 4.3 iterações.

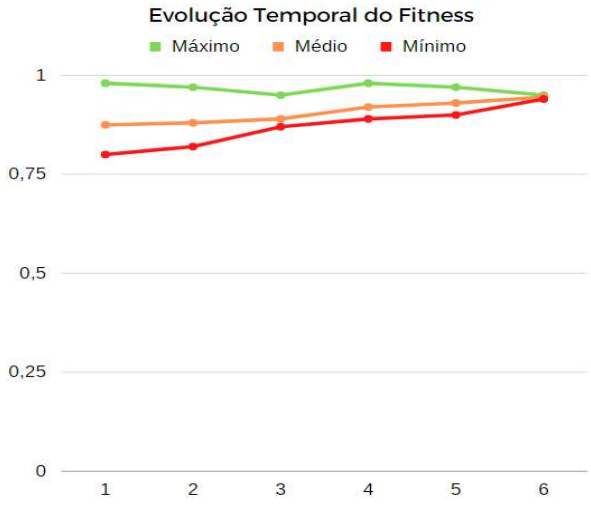
Outra análise feita foi a evolução temporal do fitness máximo, médio e mínimo por geração:

Figura 5 - Evolução Temporal do fitness - djibout



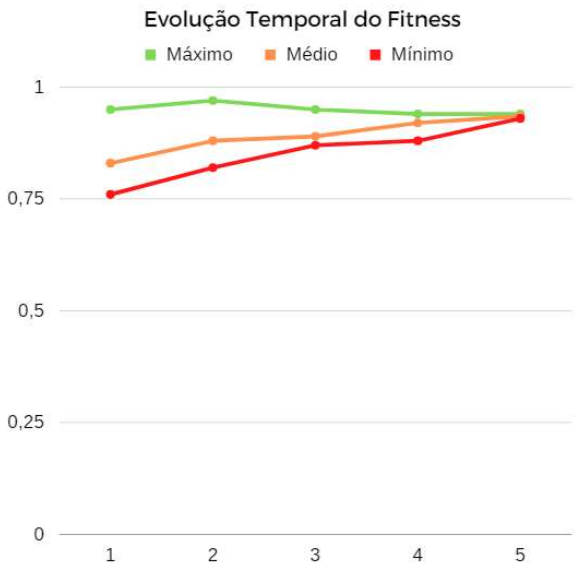
Fonte: Os Autores

Figura 6 - Evolução Temporal do fitness - Luxemburgo



Fonte: Os Autores

Figura 7 - Evolução Temporal do fitness - Omã



Fonte: Os Autores

A distância percorrida pelo melhor indivíduo "caminho" e consequentemente obteve o maior fitness nos testes feitos foram:

- Djibout: 6753 u.e.;
- Luxemburgo: 12630 u.e.;
- Omã: 104740 u.e.

Repositório

Código disponível em: https://github.com/JCGCosta/Metaheuristic_of_Bio-Inspired_Optimization