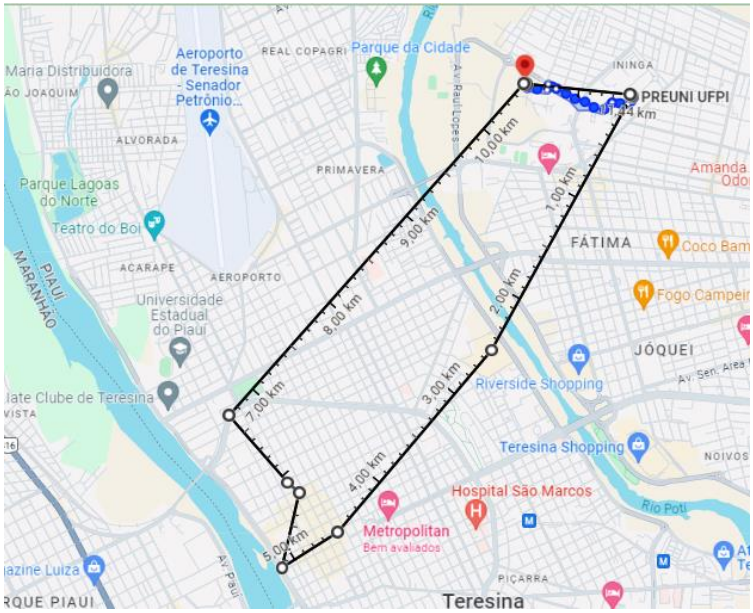
Resolução do Problema do Caixeiro Viajante por Inteligência de Enxames

INTRODUÇÃO

Em acordo com artigo da Wikipédia[1], o problema do caixeiro viajante tem um histórico de formulação matemática por sua natureza dinâmica e gráfica (na teoria de grafos), nele, é proposto o caso de um vendedor que precisa percorrer diferentes cidades para vender seus produtos, a necessidade é de escolher a sequência de cidades de maneira que a distância percorrida do ponto inicial, passando por todas as cidades, e retornando seja a menor possível. Tomando como exemplo o caminho escolhido para esse trabalho, podemos exemplificar o seguinte grafo:

Figura 1 – Percurso escolhido para análise.



Fonte: Autoria própria usando Google Maps.

Para este trabalho, será empregado o algoritmo colônia de formigas, que é uma classe do algoritmos de enxame cujo princípio faz uso de múltiplos agentes para reforço de uma sequência encontrada heuristicamente com base na minimização (ou maximização) de uma função.

Uma roupagem das ciências biológicas no que concerne o comportamento de formigas é utilizado para promover a elaboração de um procedimento matemático para o algoritmo. Neste caso, a taxa de evaporação do feromônio da formiga, o feromônio em cada trilha, a visibilidade da trilha e a constante de atualização do feromônio são alguns parâmetros tomados do escopo biológico;

METODOLOGIA

Para o emprego do algoritmo de colônia de formigas precisamos, assim como no caso do caixeiro viajante, selecionar as cidades, ou neste caso os pontos, de interesse para minimização de percurso. Em seguida, coletar a distância entre os pontos distintos dois-a-dois.

A base teórica para implementação do algoritmo será o material de aula da disciplina de Inteligência Computacional Aplicada (2023) da UFPI, bem como a apostila da disciplina IA013 do professor Fernando J. Von Zuben em conjunto com o professor Levy Boccato, ambos da Unicamp. O

[1] O artigo completo em: https://en.wikipedia.org/wiki/Travelling_salesman_problem

ambiente de execução será Python com a importação de bibliotecas Random e Numpy, em particular usaremos o ambiente do Google Colab.

PROCEDIMENTO

Visando trazer para uma aplicação cotidiana foram selecionados alguns pontos comuns (para o autor), sendo eles:

1) Prefeitura Universitária (PREUNI) da UFPI;
2) Centro de Tecnologia (CT) da UFPI;
3) Shopping Rio Poty;
4) Kina Kana da Rua Álvaro Mendes;
5) SETUT;
6) Banco do Brasil da Rua Desembargador Freitas;
7) Antenão;
8) Estação de Metrô da Matinha;

Para a medição de distância foi usado o site do Google Maps com uso da função medir distância, esta permite o cálculo aproximado[2] com base nos dados de latitude e longitude, uma melhor descrição nesse assunto de distâncias em superfícies elipsoides pode ser visto em [1]. Assim, A foi preenchido a seguinte tabela com distância em quilômetros:

Tabela 1: Distâncias em km entre os pontos elencados.

	CT	PREUNI	SPP. RP.	KINA KANA	BB	SETUT	EST. MAT.	ANTENÃO
CT	0	0,956	2,32	4,13	4	4,63	3,84	4,01
PREUNI	0,956	0	2,54	4,51	4,49	5,06	4,47	4,48
SPP. RP.	2,32	2,54	0	2,08	2,08	2,61	2,3	2,11
KINA KANA	4,13	4,51	2	0	0,616	0,616	1,38	0,616
BB	4	4,49	2,08	0,507	0	0,68	0,9156	0,093
SETUT	4,63	5,06	2,61	0,616	0,68	0	1,41	0,733
EST. MAT.	3,84	4,47	2,3	1,38	0,915	1,41	0	0,778
ANTENAO	4,01	4,48	2,11	0,616	0,093	0,733	0,778	0

Fonte: Autoria própria usando 'medir distância' do Google Maps.

Para efeitos estéticos foram usadas abreviações na tabela acima, das quais SPP. RP indica Shopping Rio Poty, BB indica Banco do Brasil e EST. MAT. indica Estação de Metrô da Matinha.

Quanto ao algoritmo de otimização por colônia de formigas, este é apresentado (código) na seção a seguir, onde os principais parâmetros ajustáveis são mantidos no começo do código para ajuste conforme necessidade. Os dados da tabela serão usados para definição da matriz de distâncias.

DESENVOLVIMENTO E CÓDIGO

O código é mostrado abaixo, para evitar mistura entre texto da apresentação e do código será empregado o uso de comentários do próprio Python para descrever cada etapa.

.

[2] O detalhe do cálculo fornecido pela própria empresa: https://cloud.google.com/blog/products/maps-platform/how-calculate-distances-map-maps-javascript-api

```
import numpy as np
import random
# Ordem de pontos da matriz de distancias
# CT|PREUNI|SPP RP|KINA KANA|BB|SETUT|EST.MAT|ANTENAO

matriz_dist=np.array([[0, 0.956, 2.32, 4.13, 4, 4.63, 3.84,4.01],
                      [0.956, 0, 2.54, 4.51, 4.49, 5.06,4.47,4.48],
                      [2.32, 2.54, 0, 2, 2.08, 2.61, 2.3, 2.11],
                      [4.13, 4.51, 2, 0, 0.507,  0.616,1.38,0.619],
                      [4, 4.49, 2.08, 0.507, 0, 0.68, 0.905,0.093],
                      [4.63, 5.06, 2.61, 0.616, 0.68,0,1.41,0.733],
                      [3.84, 4.47, 2.3, 1.38, 0.915, 1.41,0,0.778],
                      [4.01,4.48, 2.11,0.619,0.093,0.733,0.778,0]])


#Parametros
num_pontos = matriz_dist.shape[0]
num_formigas = 10
num_iter = 100
coef_evaporacao = 0.5
alfa = 1
beta = 2
Q = 100

#Variaveis de otimizacao
matriz_feromonio = np.ones((num_pontos, num_pontos))
melhor_distancia = float('inf')
melhor_caminho = []

#Loop iterativo
for iteration in range(num_iter):


    # Variavaveis de cada formiga
    distancias_formigas = np.zeros(num_formigas)
    caminhos_formigas = np.zeros((num_formigas, num_pontos),
dtype=int)

    for formiga in range(num_formigas):
     # Vetor de tamanho igual numero de pontos, iniciados com False
        visitado = [False] * num_pontos
        # Escolhe um ponto inicial
        ponto_atual = random.randint(0, num_pontos - 1)
        # Atribui como sendo o primeiro para essa formiga
        caminhos_formigas[formiga][0] = ponto_atual
        visitado[ponto_atual] = True

        # Construindo a trilha para essa formiga
        for i in range(1, num_pontos):
```

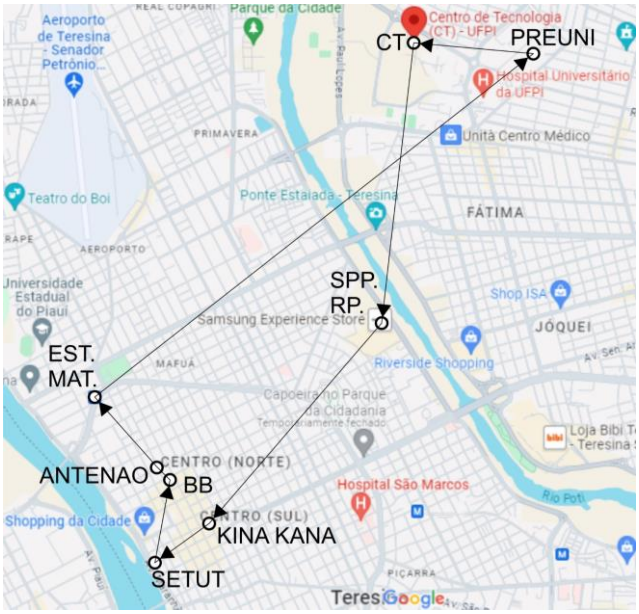
```
            # Calculando as probabilidade para prox ponto
            probabilidade_escolha = np.zeros(num_pontos)
            for ponto in range(num_pontos):
                if not visitado[ponto]:
                    feromonio =
matriz_feromonio[ponto_atual][ponto]
                    distancia = matriz_dist[ponto_atual][ponto]
                    probabilidade_escolha[ponto] = (feromonio **
alfa) * ((1.0 / distancia) ** beta)

# Escolhe o proximo ponto pelo metodo da roleta, usa as
probabilidades para enviesar
            proximo_ponto = np.random.choice(range(num_pontos),
p=(probabilidade_escolha / np.sum(probabilidade_escolha)))
            caminhos_formigas[formiga][i] = proximo_ponto
            visitado[proximo_ponto] = True
            distancias_formigas[formiga] +=
matriz_dist[ponto_atual][proximo_ponto]
            ponto_atual = proximo_ponto

    # Ajustando matriz de feromonio

    matriz_feromonio *= coef_evaporacao

    # Toma as trilha de cada formiga para atualizar o feromonio
    for formiga in range(num_formigas):
        for i in range(num_pontos - 1):
            ponto_atual = caminhos_formigas[formiga][i]
            proximo_ponto = caminhos_formigas[formiga][i + 1]
            #Q/L
            matriz_feromonio[ponto_atual][proximo_ponto] += Q /
distancias_formigas[formiga]

        # Atualizar o feromonio do ultimo paro o primeiro ponto
(fechar o grafo)
        ponto_atual = caminhos_formigas[formiga][-1]
        proximo_ponto = caminhos_formigas[formiga][0]
        #Q/L
        matriz_feromonio[ponto_atual][proximo_ponto] += Q/
distancias_formigas[formiga]

    # Atualizacao para melhor solucao
    if np.min(distancias_formigas) < melhor_distancia:
        melhor_distancia = np.min(distancias_formigas)
        melhor_caminho =
caminhos_formigas[np.argmin(distancias_formigas)]

# printa a melhor distancia (no caso menor)
# printa sequencia que da a melhor distancia
```

```
print("Melhor distancia:", melhor_distancia)
print("Melhor caminho:", melhor_caminho)
```

RESULTADOS

A execução do código retornou como menor distância o valor de 7,443 km e como sequência do grafo [1 0 2 3 5 4 7 6], indicando os pontos, a numeração segue a mesma ordem dos elementos da tabela, portanto temos como trajeto o percurso descrito na Fig. 2 abaixo:

Figura 2 – Percurso ótimo dado pelo algoritmo.



Fonte: Autoria própria com mapeamento do Google Maps.

Por questão de espaçamento, o segmento orientado BB→ANTENAO não foi incluído, mas segue no mesmo movimento horário.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Geodesics on an ellipsoid. In: WIKIPÉDIA: a enciclopédia livre. Wikimedia, 2023. Disponível em: https://en.wikipedia.org/wiki/Geodesics_on_an_ellipsoid. Acesso em: 2 jan. 2024.