Análise e Otimização de Malhas de Trânsito de Uberlândia

Nome do aluno: José Cleber Alves da Cruz Mendes – 5161815

Joaquim Marcelo Zaiden - 5162015

Luiz Gustavo Moreira Lima – 5162625

Bryan Henrique Oliveira Evangelista - 5161607

UNIUBE

2024

Os grafos são estruturas fundamentais na ciência da computação, representando conjuntos de elementos conectados por relações (arestas). Na modelagem de malhas viárias, os grafos são utilizados para mapear interseções como nós e ruas como arestas. Isso permite realizar análises como o cálculo de rotas mais curtas, simulações de tráfego e avaliação do fluxo máximo em vias específicas.

Este projeto aborda o uso de **Python**, juntamente com bibliotecas como **OSMnx**, **NetworkX**, **Matplotlib**, **Folium** e **Geopandas**, para manipular grafos representando a malha viária de Uberlândia, MG. Incluímos também a análise de congestionamento nas ruas com base na relação entre o volume de tráfego e a capacidade das vias.

Objetivo Geral

Modelar, manipular e analisar a malha viária de Uberlândia, utilizando grafos para identificar rotas, simular congestionamentos e avaliar mudanças no tráfego.

Objetivos Específicos

- Baixar e estruturar a malha viária utilizando dados do OpenStreetMap.
- Calcular e visualizar a rota mais curta com base no tempo estimado de viagem.
- Adicionar capacidades e volumes às ruas e identificar congestionamentos.
- Visualizar congestionamentos em um mapa com coloração indicativa.
- Implementar um mapa interativo com Folium para visualização de rotas.
- Realizar comparações entre diferentes cenários de tráfego.

Linguagem e Bibliotecas

- **Python 3.9+**: Linguagem principal do projeto.
- **OSMnx**: Para download e manipulação de dados geográficos.
- **NetworkX**: Para manipulação de grafos.
- **Matplotlib**: Para visualização de dados.
- **Folium**: Para criação de mapas interativos.
- **Geopandas**: Para análise espacial.

Ambiente de Desenvolvimento

- **Google Colab**: Para execução de código Python.

O projeto é dividido em etapas, com cada uma representando uma funcionalidade específica do grafo. A seguir, detalhamos os passos necessários para a implementação.

Configuração Inicial

Primeiro, instale as bibliotecas necessárias para o projeto.

Código

```
!pip install osmnx networkx matplotlib geopandas folium
```

Este comando instala as dependências do projeto, permitindo acesso a ferramentas para manipulação de grafos e mapas.

Baixar e Estruturar o Grafo

Baixamos a malha viária de Uberlândia diretamente do OpenStreetMap e adicionamos informações como velocidades médias e tempos estimados de viagem.

Código

```
import osmnx as ox

import networkx as nx

import matplotlib.pyplot as plt


# Baixar a malha viária de Uberlândia

cidade = "Uberlândia, Minas Gerais, Brasil"

grafo = ox.graph_from_place(cidade, network_type='drive')


# Adicionar velocidades médias e tempos de viagem

grafo = ox.add_edge_speeds(grafo)  # Velocidade média estimada

grafo = ox.add_edge_travel_times(grafo)  # Tempo estimado de viagem


# Visualizar o grafo

fig, ax = ox.plot_graph(grafo, node_size=0, edge_color='blue', edge_linewidth=0.5)

plt.show()
```

Calcular a Rota Mais Curta

Escolhemos dois pontos (nós) no grafo como origem e destino e calculamos o caminho mais curto em tempo.

Código

```
# Escolher dois pontos de interesse
origem = list(grafo.nodes())[0]  # Primeiro nó
destino = list(grafo.nodes())[-1]  # Último nó

# Calcular o caminho mais curto com base no tempo de deslocamento
rota_caminho_minimo = nx.shortest_path(grafo, origem, destino, weight="travel_time")
tempo_caminho_minimo = nx.shortest_path_length(grafo, origem, destino,
weight="travel_time")

print(f"Caminho mais curto entre os nós {origem} e {destino}:
{rota_caminho_minimo}")
print(f"Tempo total estimado: {tempo_caminho_minimo / 60:.2f} minutos")
```

Visualizar a Rota

Plotamos a rota mais curta em um mapa interativo utilizando a biblioteca Folium.

Código

```
import folium
from IPython.display import display

# Converter a rota em coordenadas
rota_coords = [(grafo.nodes[n]['y'], grafo.nodes[n]['x']) for n in rota_caminho_minimo]

# Criar mapa interativo
mapa = folium.Map(location=rota_coords[0], zoom_start=13)

# Adicionar rota ao mapa
```

```
folium.PolyLine(rota_coords, color='red', weight=5, opacity=0.7, tooltip="Caminho Mais Curto").add_to(mapa)

# Exibir mapa
display(mapa)
```

Adicionar Capacidades e Identificar Congestionamentos

Capacidades e volumes de tráfego são atribuídos às ruas com base em seu comprimento. Depois, identificamos vias congestionadas (onde o volume ultrapassa a capacidade) e visualizamos o congestionamento em um mapa.

Código

```
import geopandas as gpd

# Adicionar capacidade e volume às arestas
for u, v, data in grafo.edges(data=True):
    data['capacity'] = data.get('length', 50) * 0.1  # Exemplo: capacidade proporcional ao comprimento
    data['volume'] = data.get('length', 50) * 0.05  # Exemplo: tráfego proporcional ao comprimento

# Identificar ruas congestionadas
for u, v, data in grafo.edges(data=True):
    if data['volume'] > data['capacity']:
        print(f"Rua entre {u} e {v} está congestionada.")

# Visualizar congestionamento no mapa
edges = ox.graph_to_gdfs(grafo, nodes=False)
edges['congestion'] = edges['volume'] / edges['capacity']
edges.plot(column='congestion', cmap='Reds', legend=True, figsize=(12, 8),
           legend_kwds={'label': "Índice de Congestionamento"})
plt.title("Análise de Congestionamento")
```

```
plt.show()
```

Capacidade: Calculada como uma fração do comprimento da rua.

Volume: Calculado como outra fração do comprimento.

Congestionamento: Identificado onde o volume é maior que a capacidade.

Visualização: Um mapa de calor mostra ruas congestionadas com cores mais intensas.

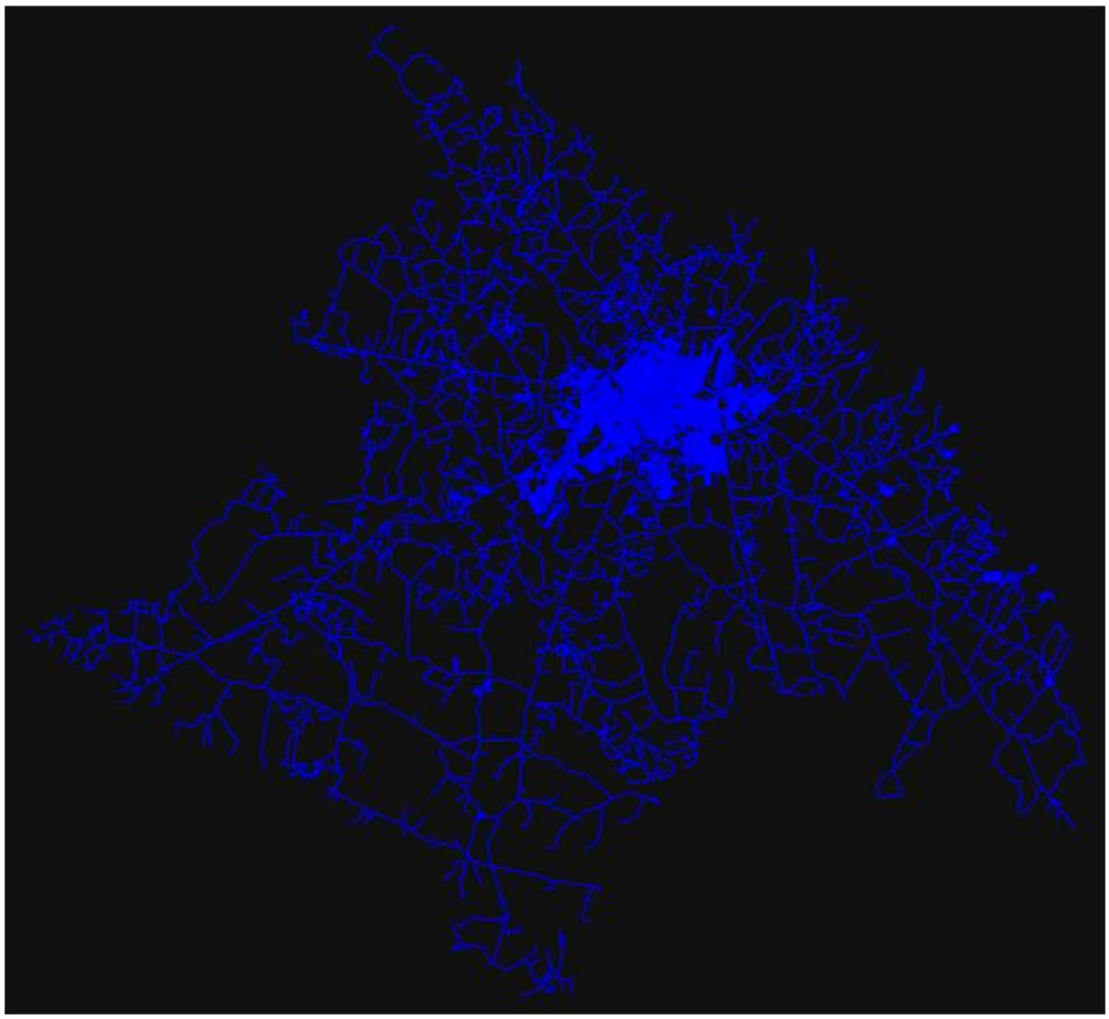
Comparar Cenários

Criamos gráficos para comparar o impacto de alterações no tráfego.

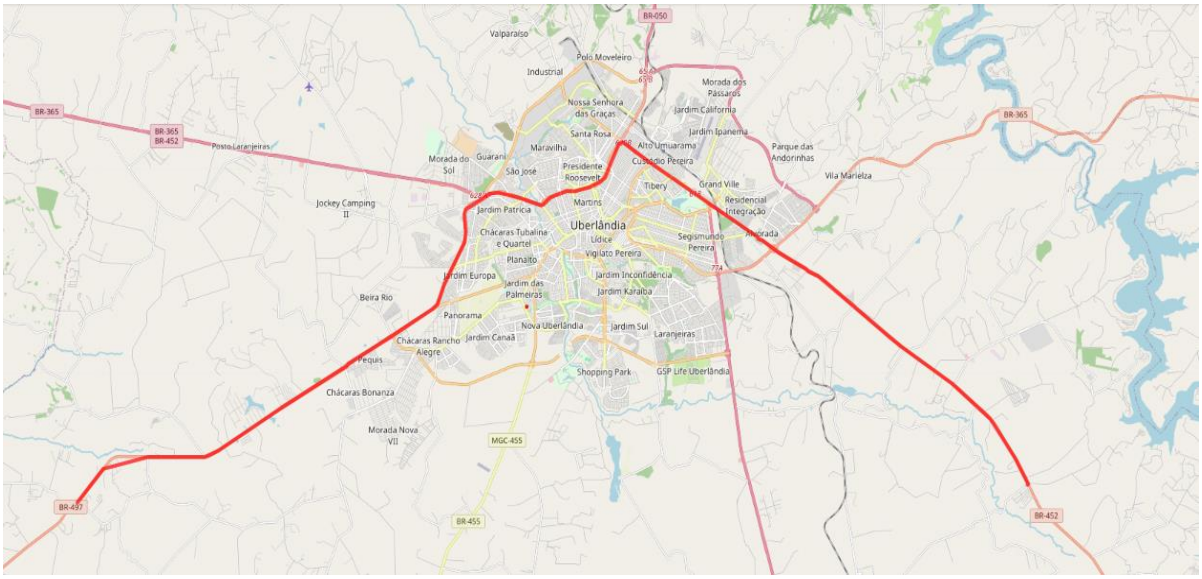
Código

```
import matplotlib.pyplot as plt


# Dados de tempo
tempos = [tempo_caminho_minimo, tempo_caminho_minimo * 1.2]  # Exemplo com aumento de 20%
cenarios = ["Original", "Congestionado"]


# Gráfico comparativo
plt.bar(cenarios, [t / 60 for t in tempos], color=['blue', 'red'])
plt.title("Comparação de Tempos de Viagem")
plt.ylabel("Tempo (minutos)")
plt.show()
```

Grafo:



Mapa interativo:



Mapa congestionamento:

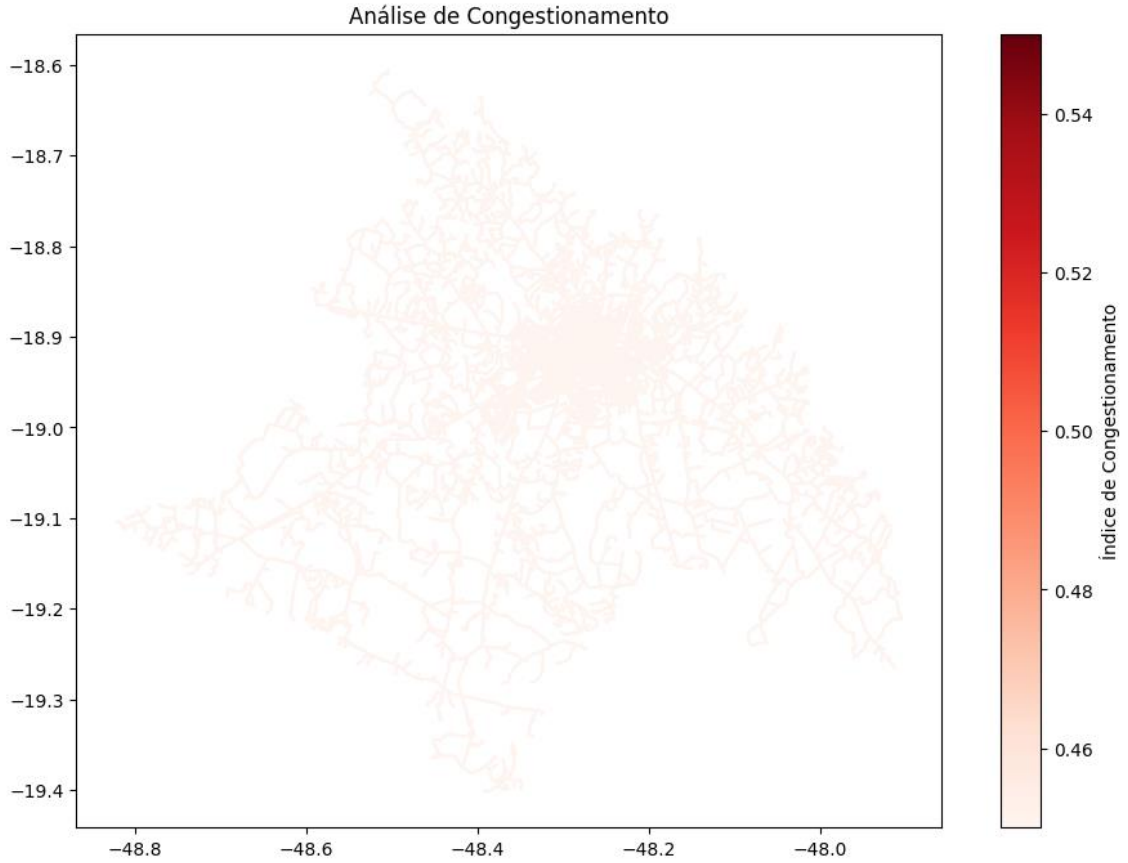
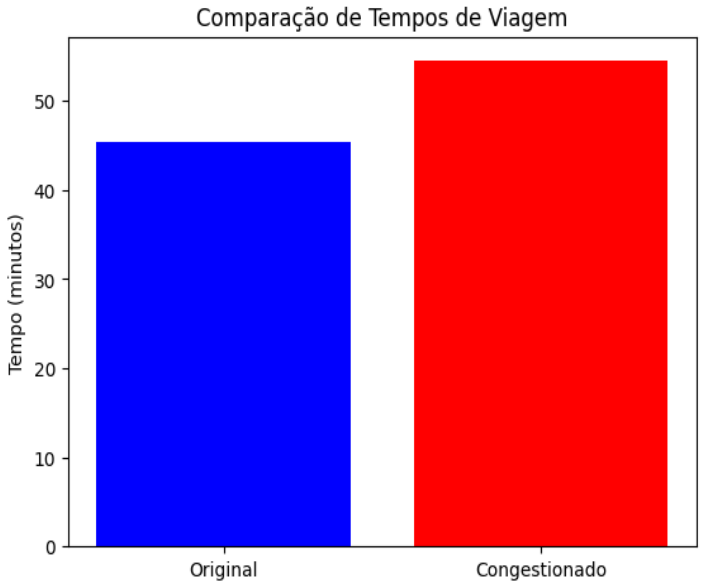


Gráfico para comparar os cenários:



Este projeto apresentado como a modelagem de malhas viárias como gráficos pode ser uma ferramenta poderosa para análise e otimização do tráfego urbano. Utilizando as ferramentas **OSMnx**, **NetworkX** e **Matplotlib**, conseguimos coletar, processar e visualizar dados de tráfego de Uberlândia, oferecendo insights valiosos sobre possíveis melhorias na infraestrutura da cidade. A utilização de algoritmos de otimização como o design do **caminho mais curto** e o **fluxo máximo** permitiu identificar pontos críticos na malha viária e sugerir rotas mais eficientes.

Uma análise de congestionamentos também foi mostrada essencial para a identificação de **gargalos** e vias que foram detectadas de intervenções, além de possibilitar simulações de diferentes cenários de tráfego. Isso demonstra a importância de se aplicar conceitos de gráficos para tomar decisões informadas sobre o planejamento e a expansão das infraestruturas viárias.

No entanto, como o tráfego é dinâmico e as cidades estão em constante mudança, uma melhoria futura seria integrar **dados em tempo real** para uma análise mais precisa e contínua. A implementação de soluções de **machine learning** para prever padrões de tráfego e melhorar rotas em tempo real também seria um avanço significativo.

Em resumo, este projeto não apenas apresentou uma solução para o **planejamento urbano e otimização de tráfego**, mas também contribuiu para a **mobilidade urbana** de forma prática, proporcionando uma abordagem eficiente para a gestão do tráfego e melhor distribuição de veículos nas vias.