Ödev 2 – Gerçek Haritada TSP

Ders: Bilgisayar Oyunlarında Yapay Zeka
Öğrenci: Mine Büşra Hazer
Tarih: 29.10.2025

Amaç

Bu çalışmada, önceki ödevde rastgele üretilen noktalardan gerçek coğrafi veriye geçilmiştir. İznik (Bursa, Türkiye) bölgesinden alınan **gerçek yol ağı (osmnx)** üzerinde, **10 rastgele nokta** seçilmiş ve bu noktalar arasında **En Yakın Komşu (Nearest Neighbor)** algoritması ile TSP çözümü yapılmıştır.

Yöntem

1. **Map Data Extraction:**
   - osmnx.graph_from_place("Iznik, Bursa, Turkey", network_type="drive")
   - Gerçek yol verisi ve bağlantılar elde edildi.
2. **Node Selection:**
   - Rastgele 10 nokta seçildi (random.sample)
   - Koordinatlar G.nodes[n]['x'], G.nodes[n]['y'] ile alındı.
3. **TSP Heuristic:**
   - "En Yakın Komşu" algoritması uygulandı.
   - Kenar ağırlıkları olarak nx.shortest_path_length(..., weight="length") kullanıldı (gerçek yol uzunlukları).
4. **Visualization:**
   - Harita çizimi folium ile yapıldı.
   - Kırmızı çizgiler → TSP rotası
   - Mavi noktalar → Seçilen 10 düğüm
   - Yeşil nokta → Başlangıç noktası

Sonuç

Bu ödev, önceki aşamadan farklı olarak gerçek dünyadaki yolları dikkate alarak TSP çözümü sunmaktadır. Böylece **gerçek sürüş mesafeleri**, **şehir topolojisi** ve **yol bağlantıları** hesaba katılmıştır.

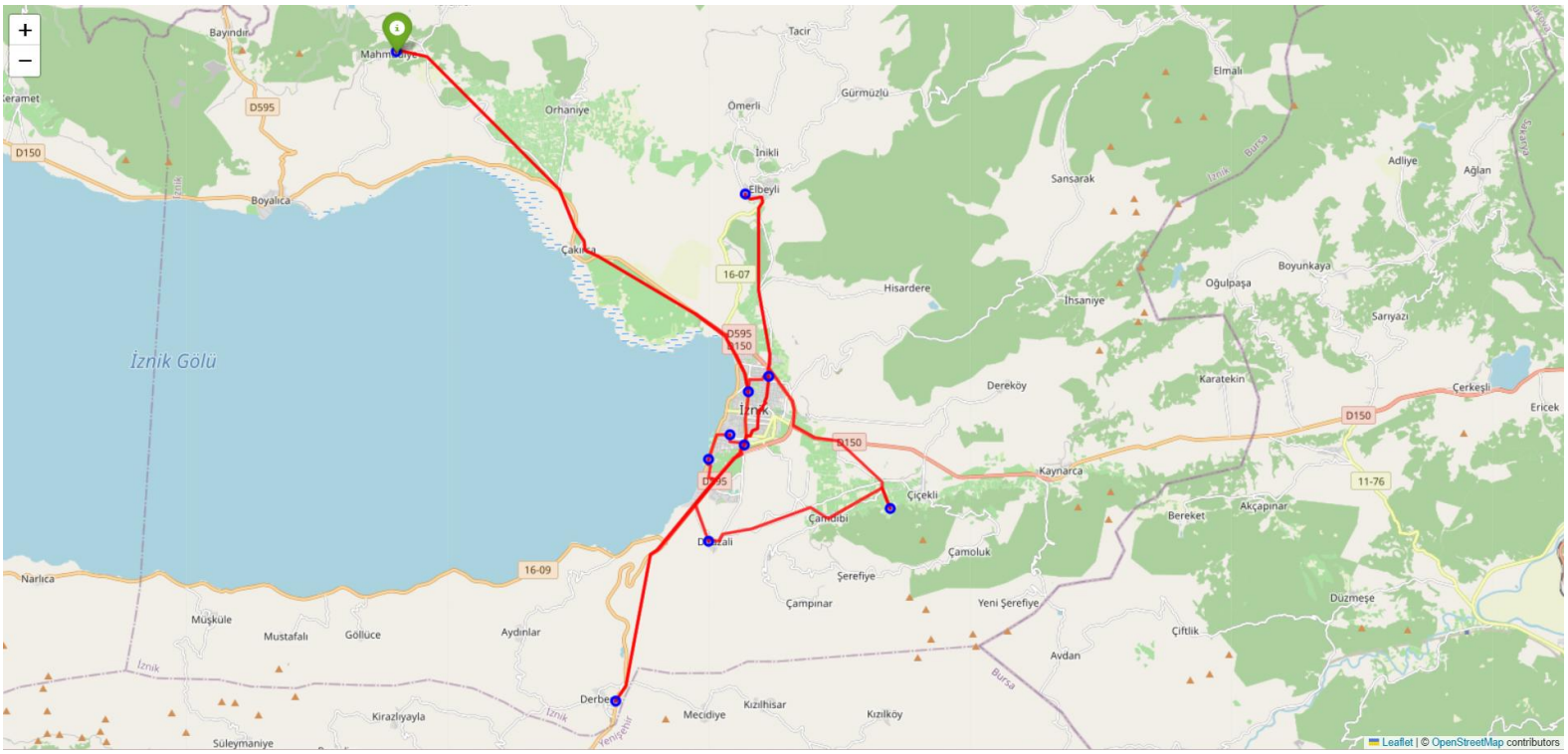
En Yakın Komşu algoritması hızlı olsa da, optimum çözüm sağlamaz ancak gerçek haritalarda uygulanabilirliği kolaydır.

Kaynak Kod

```
# ============================
# Assignment 2 – Real Map TSP
# ============================

import osmnx as ox
import random
import networkx as nx
import folium

# 1. Bursa, İznik şehrinin yollar ağını indirir
G = ox.graph_from_place("İznik, Bursa, Turkey", network_type="drive")

# 2. Grafikteki düğümleri listeler
nodes = list(G.nodes)

# 3. Rastgele 10 düğüm seçme
random.seed(42)
selected_nodes = random.sample(nodes, 10)

# 4. Gerçek mesafeleri hesaplayan yeni alt grafik
subG = nx.Graph()

for i in range(len(selected_nodes)):
    for j in range(i + 1, len(selected_nodes)):
        try:
            # Gerçek yol uzunluğu (metre)
            length = nx.shortest_path_length(G, selected_nodes[i],
selected_nodes[j], weight="length")
            subG.add_edge(selected_nodes[i], selected_nodes[j], weight=length)
        except nx.NetworkXNoPath:
            pass  # Eğer iki nokta arasında yol yoksa, geç

# 4. En Yakın Komşu (Nearest Neighbor) algoritması
def nearest_neighbor_tsp(G, start_node):
    visited = [start_node]
    current = start_node
    total_dist = 0

    while len(visited) < len(G.nodes):
        neighbors = [(n, G[current][n]['weight']) for n in
G.neighbors(current) if n not in visited]
        if not neighbors: # komsu yoksa geç
            break
        next_node, dist = min(neighbors, key=lambda x: x[1])
        visited.append(next_node)
        total_dist += dist
        current = next_node
```

```
    # Başlangıç noktasına geri dön
    total_dist += G[current][start_node]['weight']
    visited.append(start_node)
    return visited, total_dist

# 5. Algoritmayı çalıştır
start = selected_nodes[0]
tour, total_distance = nearest_neighbor_tsp(subG, start)
print("Tur sırası (node ID):", tour)
print(f"Toplam Mesafe: {total_distance:.2f} metre")

start_lat = G.nodes[start]['y']
start_lon = G.nodes[start]['x']

m = folium.Map(location=[start_lat, start_lon], zoom_start=13)

# 6. Rota çizimi
for i in range(len(tour) - 1):
    try:
        path = nx.shortest_path(G, tour[i], tour[i + 1], weight="length")
        route_coords = [(G.nodes[n]['y'], G.nodes[n]['x']) for n in path]
        folium.PolyLine(route_coords, color="red", weight=3,
opacity=0.8).add_to(m)
    except nx.NetworkXNoPath:
        continue

# 7. Düğümleri mavi dairelerle işaretle
for n in selected_nodes:
    folium.CircleMarker(
        location=(G.nodes[n]['y'], G.nodes[n]['x']),
        radius=4,
        color='blue',
        fill=True,
        fill_color='blue'
    ).add_to(m)

# 8. Başlangıç noktası yeşil
folium.Marker(
    location=(G.nodes[start]['y'], G.nodes[start]['x']),
    popup="Start Point",
    icon=folium.Icon(color='green')
).add_to(m)

# 9. Kaydet
map_path = "hw-2/real-map-tsp.html"
m.save(map_path)
```

Gerçek Harita Görüntüsü:



Ödevin GitHub Linki:

https://github.com/busrahazer/progressive-tsp-research