Optimized Delivery Route Planning with Traffic Congestion Simulation

Objective:

The primary objective of this project was to develop a system capable of planning optimized delivery routes for vehicles in a logistics or supply chain operation, considering both the distance and traffic congestion. The system aimed to minimize transportation costs and delivery times by utilizing advanced optimization techniques, specifically the Traveling Salesman Problem (TSP). Additionally, it integrated real-time traffic congestion simulation to account for varying traffic conditions during different times of the day.

This approach aimed to provide businesses with a tool that could optimize delivery schedules and routes while also ensuring efficient fuel consumption and timely deliveries.

Abstract:

Efficient route planning is a key factor in reducing costs and enhancing customer satisfaction in logistics. However, traditional optimization techniques, such as the Traveling Salesman Problem (TSP), typically ignore the dynamic nature of traffic congestion. This project integrates real-world traffic data (simulated for the purposes of this project) into TSP to model delivery routes that are not only the shortest but also the fastest, considering peak and off-peak traffic conditions. Using Python libraries such as networkx and folium, the project constructs a graph of delivery points, simulates traffic congestion, and optimizes the route accordingly. The result is a more accurate and realistic routing solution for businesses in logistics.

Methodology:

1. Data Collection: The project began by gathering data related to delivery points. These points were defined as coordinates (latitude and longitude) for different locations in a city or service area. In addition to the coordinates, the traffic congestion levels at different times of day were simulated, including factors such as mild, moderate, and heavy traffic. The traffic congestion data was used to adjust the weights (time cost) of the edges between delivery points.

Example of Data:

- Delivery Points: Coordinates representing locations for delivery.
- Traffic Data: Simulated traffic data representing traffic levels and delays for each time of day.

2. Traffic Congestion Simulation:

- The project implemented a traffic congestion simulation, where different times of the day were simulated to reflect varying traffic conditions. For each edge between two nodes (delivery points), a congestion factor was applied, increasing the travel time between points based on traffic levels.

Code for Traffic Congestion Simulation:

```
python
CopyEdit
def simulate_traffic_condition(time_of_day):
    # Simulate mild, moderate, or heavy traffic conditions
    if time_of_day in ['8-9AM', '5-6PM']:  # Peak hours
        return 1.5  # 50% more time due to traffic
    elif time_of_day in ['12-1PM']:  # Moderate traffic
        return 1.2  # 20% more time
    else:
        return 1  # No additional time in off-peak hours
```

3. Graph Construction: A graph was created where each delivery point was a node and the edges represented the routes between delivery points. The weights of the edges were adjusted based on the traffic simulation, representing the adjusted travel times during peak and off-peak traffic conditions.

Graph Construction Code:

```
python
CopyEdit
import networkx as nx

# Create a graph
G = nx.Graph()
```

```
# Add delivery points (nodes)
for point in delivery_points:
    G.add_node(point)


# Add edges between delivery points with travel times adjusted for traffic
for i in range(len(delivery_points)):
    for j in range(i + 1, len(delivery_points)):
        travel_time = calculate_travel_time(delivery_points[i], delivery_points[j])
        # Adjust travel time based on traffic congestion
        adjusted_travel_time = travel_time *
simulate_traffic_condition(current_time_of_day)
        G.add_edge(delivery_points[i], delivery_points[j],
weight=adjusted_travel_time)
```

4. Optimization with the Traveling Salesman Problem (TSP): The Traveling Salesman Problem (TSP) was applied to find the most optimal route to visit all delivery points. The objective was to minimize the total travel time, considering both the distances and the traffic congestion factors.

The networkx library was used to solve the TSP. Since the problem is NP-hard, an approximation algorithm was applied to find a near-optimal solution.

TSP Optimization Code:

python

```
CopyEdit
# Use the TSP approximation to find the shortest route
optimized_route = nx.approximation.traveling_salesman_problem(G,
cycle=False, weight='weight')
```

5. Visualization and Mapping: The optimized route was visualized on a map using the folium library. This allowed for the interactive display of the delivery points and the

optimal route on a map, which can be easily shared or integrated into a logistics application.

Code for Visualization with Folium:

python

CopyEdit

```
import folium


# Initialize the map
m = folium.Map(location=[delivery_points[0][0], delivery_points[0][1]], zoom_start=12)


# Add markers for each delivery point
for point in optimized_route:
    folium.Marker([point[0], point[1]]).add_to(m)


# Add the optimized route as a polyline
folium.PolyLine(locations=optimized_route, color='blue').add_to(m)


# Save the map as an HTML file
m.save("optimized_route_map.html")
```

6. Cost and Time Calculation: After obtaining the optimized route, the total travel time was calculated by summing the travel times for each segment of the route. This total time was then used to calculate the overall cost, factoring in fuel costs, driver wages, and other relevant factors.

Cost Calculation Code:

python

CopyEdit

```
total_travel_time = sum([G[optimized_route[i]][optimized_route[i+1]]['weight']
for i in range(len(optimized_route)-1)])

total_cost = total_travel_time * fuel_cost_per_hour
```

Results:

1. Optimized Route:
   - The TSP algorithm successfully generated the most efficient route for visiting all the delivery points. The traffic congestion simulation effectively adjusted the travel times for each leg of the journey, ensuring that the route accounted for peak and off-peak traffic conditions.
2. Cost and Time Analysis:
   - The project's output showed a reduction in the total travel time and cost compared to an initial route (without traffic consideration). The congestion factor significantly impacted the total travel time during peak hours, confirming that traffic conditions must be factored into route planning.
3. Visualization:
   - The optimized route was visualized on an interactive map, allowing stakeholders to visualize the delivery points and the most efficient route. The map was saved as an HTML file for easy access and integration into logistics management systems.

Conclusion:

This project successfully developed an optimized delivery route planning system by incorporating traffic congestion into the Traveling Salesman Problem. The integration of simulated traffic congestion allowed for a more realistic and effective solution for businesses looking to reduce their transportation costs and improve delivery efficiency. By using networkx and folium in Python, the system was able to not only calculate the optimized route but also provide a visual representation that could be easily used by logistics companies.

Future work could involve integrating real-time traffic data through APIs (such as Google Maps or OpenStreetMap) for dynamic updates to the delivery routes. Additionally, the optimization algorithm could be extended to account for additional variables, such as fuel consumption, vehicle capacities, and customer time windows, to make the solution even more robust and adaptable.