FedEx SMART

Supply Chain Modelling, Algorithms, Research and Technology Centre

Indian Institute of Technology Madras

Date: 20 July, 2024

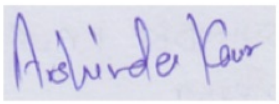
Internship Certificate

This is to certify that ***Sathyadharini S,*** from the *College of Engineering, Anna University*, pursued the summer internship at **FedEx SMART Centre, IIT Madras**. This internship was for two months starting from 20 May 2024 to 20 July 2024.

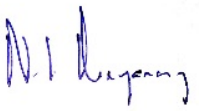
The work was done under the guidance of ***Prof. Arshinder Kaur***

Domain of the work: *Optimizing Route Planning from Voice Data Using Dynamic TSP and MetaHeuristic Algorithms.*

Her performance was satisfactory. We wish her all the best for her future endeavors.



Guide: Prof. *Arshinder Kaur*
Professor, Department of Management Studies
IIT Madras



Prof. N S Narayanaswamy
Head of FedEx SMART Centre, IIT Madras

Optimizing Route Planning from Voice Data Using Dynamic TSP and MetaHeuristic Algorithms

SATHYADHARINI S

July 2024

Technical Paper

1. Abstract

This thesis presents an innovative approach to dynamic route planning using voice data and advanced optimization algorithms. The system processes recorded conversations to transcribe and extract addresses using Named Entity Recognition (NER) model. These addresses are geocoded and integrated into an updated dataset of locations. Traffic conditions are analyzed to generate a dynamic time matrix, which, along with historical data correlations, serves as input for solving the Traveling Salesman Problem (TSP). The Ant Colony Optimization (ACO) algorithm along with other meta-heuristic algorithms were employed to find efficient routes, showcasing improved operational efficiency and decision-making in urban logistics.

2. Literature Review

The Traveling Salesman Problem (TSP) is an NP-hard problem with complexity increasing factorially with the number of nodes. Meta-heuristic algorithms are optimization techniques that can provide satisfactory solutions to TSP. This paper examines six heuristic algorithms: Nearest Neighbor, Genetic Algorithm, Simulated Annealing, Tabu Search, Ant Colony Optimization, and Tree Physiology Optimization.The study [1] compares these algorithms in terms of computation, accuracy, and convergence , tested based on Euclidean distance.

The Traveling Salesman Problem (TSP) is crucial for modeling various real-world problems but becomes increasingly complex with more nodes. This research [2] explores the Dynamic TSP (DTSP) in urban environments where travel times are unpredictable and modeled as random variables. Using real-time traffic data, the study aims to optimize route planning dynamically, leveraging heuristic search algorithms to solve DTSP efficiently. The findings suggest significant productivity gains from dynamic routing, though practical implementation challenges remain.

Travel planning is a key issue in location-based services (LBS). Due to TSP's NP-hard nature, it’s not widely studied for online LBS, and the nearest-neighbor heuristic is commonly used. This study here in [3] questions its accuracy and examines which heuristics are best for online TSP queries in LBS. By investigating 22 heuristics and conducting extensive performance studies on real and synthetic datasets, the research reveals that the best heuristics for LBS differ from those suggested by existing benchmarks. Additionally, it demonstrates that high-quality TSP solutions can be achieved through precomputation and indexing.

Named Entity Recognition (NER) is crucial for extracting names like Persons, Locations, and Organizations from text, benefiting areas such as Question Answering, Information Retrieval, and Machine Translation. This paper [4] reviews and compares Rule-based, Machine Learning-based, and Hybrid NER methods using the Message Understanding Conference (MUC) standards. It identifies their strengths and weaknesses and proposes a novel Machine Learning method called Fuzzy Support Vector Machine (FSVM) for improved NER. The research evaluates these methods on precision and portability, highlighting the importance of accurate named entity extraction in various NLP tasks.

This paper [5] examines the in-context learning abilities of OpenAI's Whisper ASR models,introducing a novel speech-based in-context learning (SICL) approach for test-time adaptation. SICL reduces word error rates (WERs) significantly with minimal labeled speech samples and without gradient descent. Experiments with Chinese dialects demonstrate consistent WER reductions of 32.3% on average across various Whisper model sizes. Using a k-nearest-neighbors example selection technique further enhances SICL's efficiency, increasing average WER reduction to 36.4%. The approach is validated for speaker adaptation and continuous speech recognition tasks, both showing significant WER reductions.

This article [6] discusses the Google Maps API, a free function set for network mapping developed by Google. It highlights the features, functions, and development methods for WebGIS based on the Google Maps API, and demonstrates its application in developing a logistics network system. The study shows that the Google Maps API can easily embed maps into logistics web pages, offering high-resolution data and a user-friendly interface. Utilizing the Google Maps API and WebGIS enhances the visualization of logistics rationing systems, indicating a future trend in logistics network system development.

3. Introduction

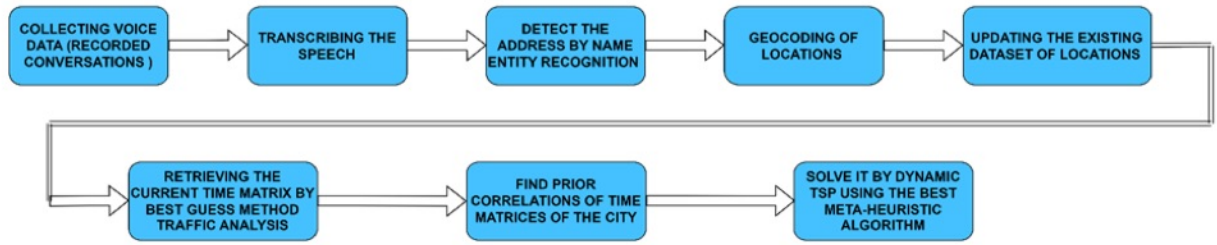
In the era of rapid technological advancements and increasing urbanization, efficient logistics and transportation systems have become crucial for enhancing operational efficiencies and reducing costs. This project aims to address the challenges of dynamic route planning in urban logistics by integrating advanced technologies such as voice recognition, Named Entity Recognition (NER), geocoding, and meta-heuristic algorithms.

The process begins with collecting voice data from recorded conversations, which are then transcribed into text. Through Named Entity Recognition, the addresses mentioned in these conversations are detected and geocoded to obtain precise geographical coordinates. These locations are integrated into an updated dataset, ensuring the system has the most current information.

Real-time traffic conditions are analyzed to generate a dynamic time matrix, reflecting the current travel times between locations. Additionally, historical correlations of city time matrices are considered to enhance the accuracy of predictions.

The core challenge of optimizing routes is approached by solving the Traveling Salesman Problem (TSP), a well-known NP-hard problem, using advanced meta-heuristic algorithms. The project evaluates various algorithms to determine the most effective one for dynamic TSP, ensuring that the chosen solution minimizes travel time and maximizes efficiency.

This integration of voice recognition, real-time data analysis, and sophisticated optimization algorithms represents a significant step forward in the development of intelligent logistics systems. The proposed solution not only improves the accuracy and efficiency of route planning but also adapts dynamically to changing conditions, providing a robust and scalable approach for urban logistics.



EXACT CURRENT PROBLEM & What it exactly solves:

- Most of the delivery partners associated with courier and other delivery services are new to most of the places in the city and everyone rely upon G-MAP services for identification of location that they need to delivery.
- And most of the customers alter their destinations, hence making the situation tougher to locate the addresses correctly.
- For this Automating from voice calls in the detection of addresses correctly , then validating addresses and geocoding the addresses for faster computation in the further steps .
- Next steps involves fetching the real-time traffic matrices (time matrix) and computing using set of locations as TSP.
- Since there is no application and testing of Meta-hueristics in a real time scenario (Previous research where just based on using EUCLIDEAN distances , This paper aims to test the efficiency of solving faster and which algorithm performs the best with actual result(Brute-force).
- Ant colony algorithm goes best in this scenario giving almost 98 percent closeness with the actual result in less time.(result)

4.TRANSCRIBING THE SPEECH

4.1 OPEN AI'S WHISPER MODEL:

Whisper is an automatic speech recognition (ASR) model developed by Open AI. It is designed for speech transcription, converting spoken language into written text. Whisper aims to provide high accuracy in transcription and supports multiple languages and various accents, making it versatile for different applications in speech recognition.

```
import whisper
model=whisper.load_model('large')
result=model.transcribe(r'C:\Users\sthdh\OneDrive\Documents\Sound Recordings\Recording (4).m4a',fp16=False)
[28] ✓ 2m 28.5s

result['text']
[29] ✓ 0.0s
'  Local Apartments, Kotiwakam, Kuppam Road, Srinivasaburam, Trivandrum, Chennai, Tamil Nadu, 600041'
```

4.2.NLP TECH

A Named Entity Recognition (NER) model is a type of Natural Language Processing (NLP) tool used to identify and classify named entities mentioned in text into predefined categories such as names of people, organizations, locations, dates, times, quantities, monetary values, percentages, and more. NER is a crucial component in various applications like information retrieval, question answering, and data extraction.

Annotation:

Annotating a dataset involves labeling text with entity tags to create a training set for the NER model. Example: "Barack Obama was the 44th President of the United States" might be annotated as "Barack Obama [PER] was the 44th President of the United States [LOC]."

Annotations and labels in Named Entity Recognition (NER) play a crucial role in detecting addresses in texts. Here's how the different labels help in this process:

4.2.1. GPE (Geopolitical Entity):

Identifies geopolitical locations such as cities, states, and countries.Essential for pinpointing specific parts of an address. For instance, recognizing "Cicero", "Illinois", and "United States" helps to identify the city, state, and country components of the address.

4.2.2 FAC (Facility):

Identifies named buildings, airports, highways, bridges, and other facilities.Useful for detecting parts of addresses that refer to specific structures or significant locations within an address. For example, "Cicero Avenue" is identified as a facility, which helps in understanding that it is a specific location within a city.

4.2.3 ORG (Organization)

Identifies names of companies, institutions, and organizations.In some addresses, organizations are part of the address details. For example, if an address is located within a company or institution's premises, recognizing the organization helps in correctly identifying the full address.

4.2.4. CARDINAL

Identifies numerals that do not fall into other specific categories like dates or percentages. Crucial for recognizing street numbers, ZIP codes, or any numeric part of the address. For instance, "2601" is recognized as a cardinal number, which helps in identifying the street number in the address.

4.2.5 Process of Detecting Addresses in Texts Using Annotations:

1. Text Processing:

The text is processed using an NER model like spaCy, which scans through the text and tags entities with appropriate labels.

2. Entity Extraction:

Entities matching the desired labels (GPE, FAC, ORG, CARDINAL) are extracted from the text. This filtering helps in focusing only on the parts of the text that are relevant to forming an address.

3. Address Construction:

The extracted entities are then combined to construct the address. For example, combining entities like "2601" (CARDINAL), "S Cicero Avenue" (FAC), "Cicero" (GPE), "Illinois" (GPE), and "United States" (GPE) to form the complete address.

4.2.6 Contextual Understanding:

The combination of different types of entities allows the system to understand the context and structure of the address. For instance, knowing that a CARDINAL followed by a FAC typically indicates a street address.

Hello, this is Bluedart calling. Am I speaking with Cleve?

Yes, this is Cleve. How can I help you?

Great! I'm calling to confirm the delivery details for your parcel. The address we have on file is 2601 S Cicero Avenue, Cicero, Illinois 60804, United States. Is that correct?

Yes, that's correct. Can you deliver it today?

Absolutely Cleve, our delivery team will make sure your parcel is delivered to 2601 S Cicero Avenue, Cicero, Illinois 60804, United States, by the end of the day.

Overall, NER annotations and labels provide a structured way to identify and extract addresses from unstructured text, making it easier to automate processes that depend on accurate address information.

4.3 VALIDATING AND STRUCTURING THE ADDRESSES:

This involves proper formatting and validation of addresses through the service of GMAP APIs and produces a formatted address ready to be gecoded as longitudes and latitudes.

```
[44]  ✓ 0.3s

    Formatted Address: 5908 W 26th St, Cicero, IL 60804, USA
    Latitude: 41.8437866
    Longitude: -87.7715113
    Partial Match: False

    Address Components:
     - 5908 (street_number)
     - West 26th Street (route)
     - Cicero (locality, political)
     - Cicero Township (administrative_area_level_3, political)
     - Cook County (administrative_area_level_2, political)
     - Illinois (administrative_area_level_1, political)
     - United States (country, political)
     - 60804 (postal_code)
```

4.4 Geocoding

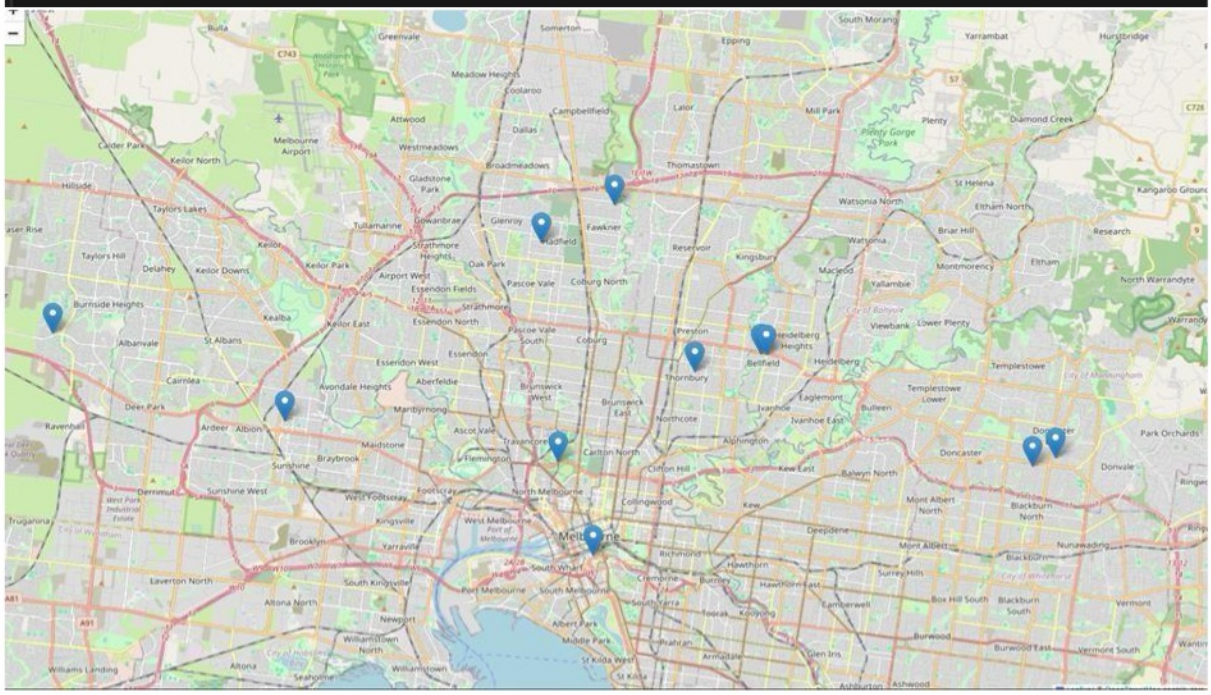
Geocoding is the process of transforming text-based addresses into their corresponding pairs of latitude and longitude. This conversion facilitates easier and more efficient computations, as well as more effective storage and retrieval of location data.

```
    "Eureka Tower, Level 89/7 Riverside Quay, Southbank VIC 3006, Australia",
    "Elliott Ave, Parkville VIC 3052, Australia",
    "14 Blenheim Way, Caroline Springs VIC 3023, Australia",
    "16 Barbara St, Hadfield VIC 3046, Australia",
    "112 Flinders St, Thornbury VIC 3071, Australia",
    "12 Rosamond Cres, Doncaster East VIC 3109, Australia",
    "11 Beverley St, Doncaster East VIC 3109, Australia",
    "141 Oriel Rd, Bellfield VIC 3081, Australia",
    "6 Boronia Ct, Bellfield VIC 3081, Australia",
    "27 Apollo Cres, Dallas VIC 3047, Australia"
]

# Function to get geocode information
CodeMate
def get_geocode(address):
    geocode_result = gmaps.geocode(address)
    if geocode_result:
        lat = geocode_result[0]['geometry']['location']['lat']
        lng = geocode_result[0]['geometry']['location']['lng']
        return lat, lng
    else:
        return None, None

# Get geocode information for all addresses
geocodes = []
for address in addresses:
    lat, lng = get_geocode(address)
    geocodes.append((lat, lng))

# Print the results
for address, geocode in zip(addresses, geocodes):
    print(f"{address} -> Latitude: {geocode[0]}, Longitude: {geocode[1]}")

[46]  ✓ 2.5s

16 Garnet St, Sunshine North VIC 3020, Australia -> Latitude: -37.7740202, Longitude: 144.8315224
Eureka Tower, Level 89/7 Riverside Quay, Southbank VIC 3006, Australia -> Latitude: -37.8213341, Longitude: 144.9646982
Elliott Ave, Parkville VIC 3052, Australia -> Latitude: -37.7884387, Longitude: 144.9493645
14 Blenheim Way, Caroline Springs VIC 3023, Australia -> Latitude: -37.7430971, Longitude: 144.7312587
16 Barbara St, Hadfield VIC 3046, Australia -> Latitude: -37.711003, Longitude: 144.9423416
112 Flinders St, Thornbury VIC 3071, Australia -> Latitude: -37.7565525, Longitude: 145.0084381
12 Rosamond Cres, Doncaster East VIC 3109, Australia -> Latitude: -37.7869128, Longitude: 145.1643381
11 Beverley St, Doncaster East VIC 3109, Australia -> Latitude: -37.7900897, Longitude: 145.1543889
141 Oriel Rd, Bellfield VIC 3081, Australia -> Latitude: -37.7506727, Longitude: 145.0394974
6 Boronia Ct, Bellfield VIC 3081, Australia -> Latitude: -37.7503033, Longitude: 145.0370439
27 Apollo Cres, Dallas VIC 3047, Australia -> Latitude: -37.6740517, Longitude: 144.9314236
```

4.5 RANDOM COLLECTION OF ADDRESSES AND GEOCODING IT

The set of addresses are marked on a static map using folium library and this set was the sample input for further tsp algorithms.

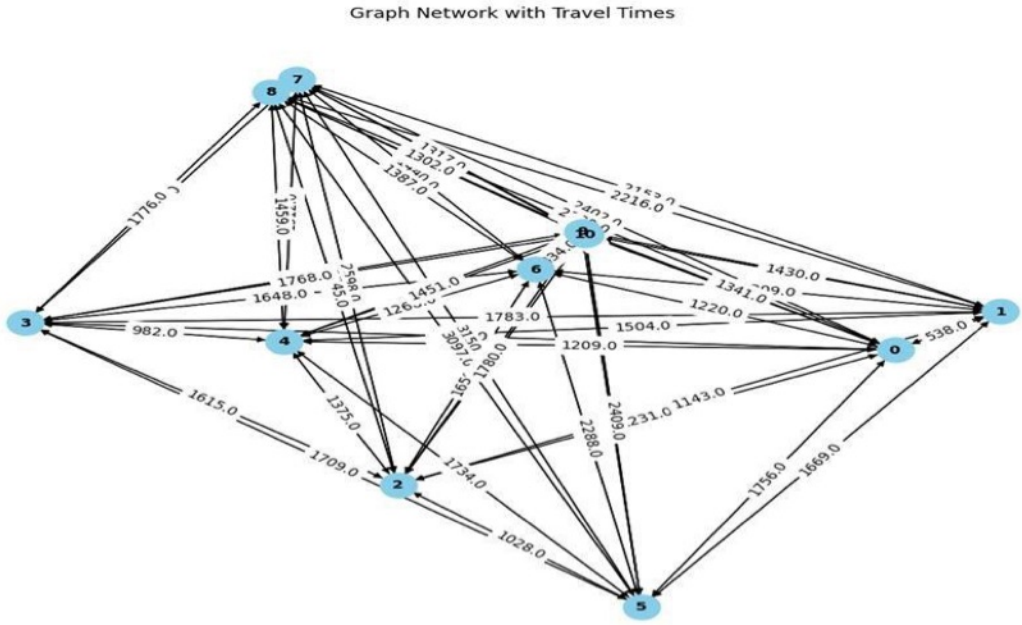
```python
# Given geocodes array
geocodes = np.array([
    [-37.711003, 144.9423416],
    [-37.6975672,144.9738736],
    [-37.7740202, 144.8315224],
    [-37.8213341, 144.9646982],
    [-37.7884387, 144.9493645],
    [-37.7430971, 144.7312587],
    [-37.7565525, 145.0084381],
    [-37.7869128, 145.1643381],
    [-37.7900897, 145.1543889],
    [-37.7506727, 145.0394974],
    [-37.7503033, 145.0370439],

])
```



5.TRAFFIC DATA FETCH AND TIME MATRIX

Using the Google Maps API, traffic data can be fetched by inputting a set of latitude and longitude coordinates. The API provides real-time traffic conditions and computes travel times between these points, forming a time matrix that helps in optimizing routes based on current traffic conditions. This dynamic approach enhances route planning and decision-making for logistics and transportation systems.

```
Travel Time Matrix (in minutes):
[[ 0.         20.26666667 21.96666667 22.48333333 22.3        20.98333333]
 [20.15       0.          2.26666667  5.86666667 15.86666667 16.08333333]
 [21.83333333  2.43333333  0.1         4.06666667 14.38333333 14.1       ]
 [20.71666667  6.28333333  4.4         0.06666667 11.28333333 11.21666667]
 [21.08333333 17.06666667 14.8        11.3         0.          4.21666667]
 [20.05       15.63333333 14.71666667 11.35        4.51666667  0.        ]]
```



This represents the graph network with travel times between each pair of locations in the set of sample data1.

6.ROUTE PLANNING

Solving TSP efficiently is crucial for various applications in logistics, transportation, and delivery services. This project leverages multiple approaches, including heuristic and metaheuristic algorithms, to find optimal or near-optimal solutions for TSP.

The project workflow involves several key steps: creating a distance matrix using the Google Maps Distance Matrix API, implementing both exact and heuristic algorithms to solve TSP, measuring their execution times, and visualizing the computed routes. The algorithms include brute force, Ant Colony Optimization (ACO), Genetic Algorithm, and Nearest Neighbor heuristic. Each algorithm's performance is evaluated based on solution quality and computational efficiency, providing insights into their applicability for real-world routing problems.

STEPS FOR EXECUTION

1. Distance Matrix Creation: This function creates the distance matrix by querying the Google Maps Distance Matrix API in batches to avoid exceeding the API's limits.
2. Brute Force TSP: This implementation finds the exact solution by checking all possible permutations of the route.
3. Ant Colony Optimization: The ACO class simulates ant behavior to find a near-optimal solution for the TSP.
4. Genetic Algorithm TSP: This algorithm uses principles of evolution (selection, crossover, mutation) to find a near-optimal solution.
5. Nearest Neighbor TSP: This heuristic algorithm builds a route by always moving to the nearest unvisited city.

6. Execution Time Measurement: The measure_time function measures the time taken by each algorithm to solve the TSP.

Plotting: The plot_route function visualizes the routes found by each algorithm on a map

RESULTS OBTAINED FOR DATASET1:

Brute Force Result: ([0, 3, 1, 2, 7, 6, 9, 8, 5, 4, 10, 0], 151.70000000000002)

Time: 15.569929361343384 seconds

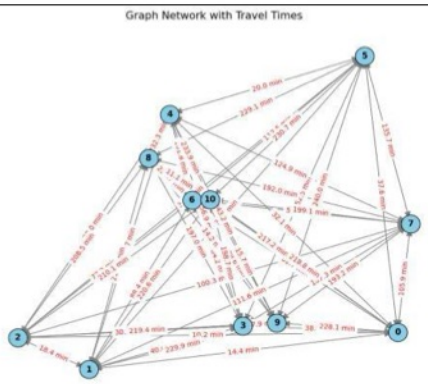
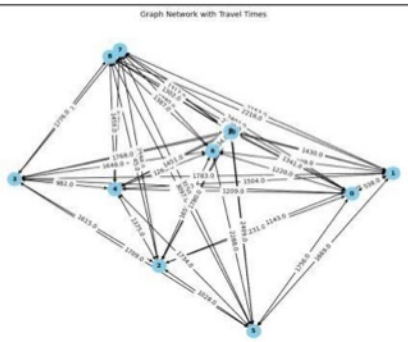
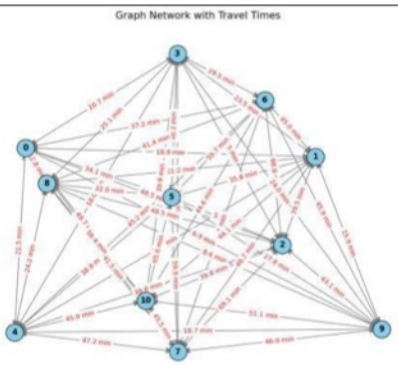
ACO Result: ([(0, 3), (3, 1), (1, 2), (2, 6), (6, 7), (7, 9), (9, 8), (8, 5), (5, 4), (4, 10), (10, 0)], 152.51666666666665)
Time: 0.5671834945678711 seconds

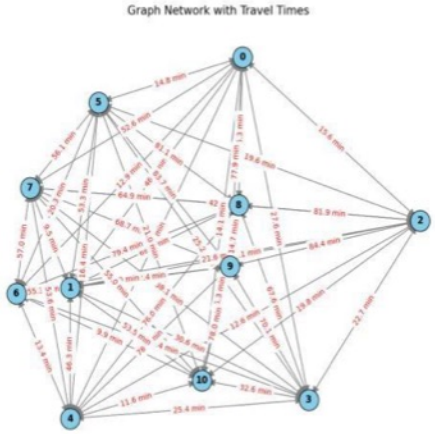
Genetic Algorithm Result: ([0, 3, 10, 4, 1, 2, 7, 6, 9, 8, 5, 0], 162.01666666666668)

Time: 1.1032276153564453 seconds

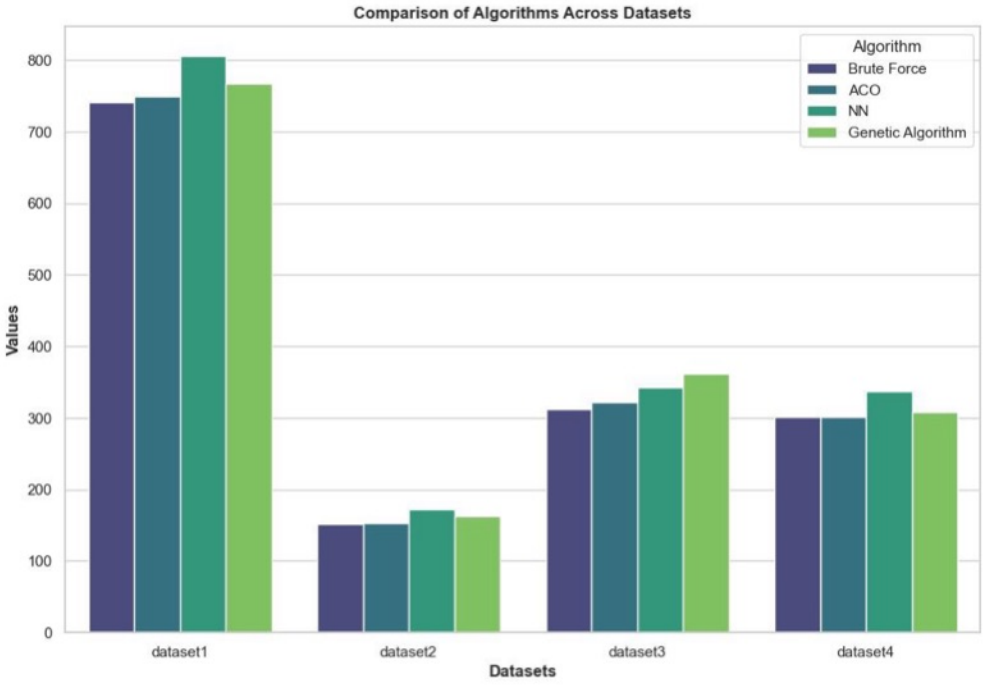
Nearest Neighbor Result: ([0, 3, 10, 4, 2, 1, 5, 9, 8, 7, 6, 0], 172.01666666666662) Time:

0.0 seconds

DATASET	BRUTEFORCE	ANTCOLONY OPTIMISATION	NEAREST NEIGHBOUR	GENETIC ALGORITHM
	741.5 (15.63)	749.15 (0.56)	806.199 (1.4)	767.834(1.12)
	151.77	152.516	172.0166	162.0167
	312.249	322.45	342.716	362.3499

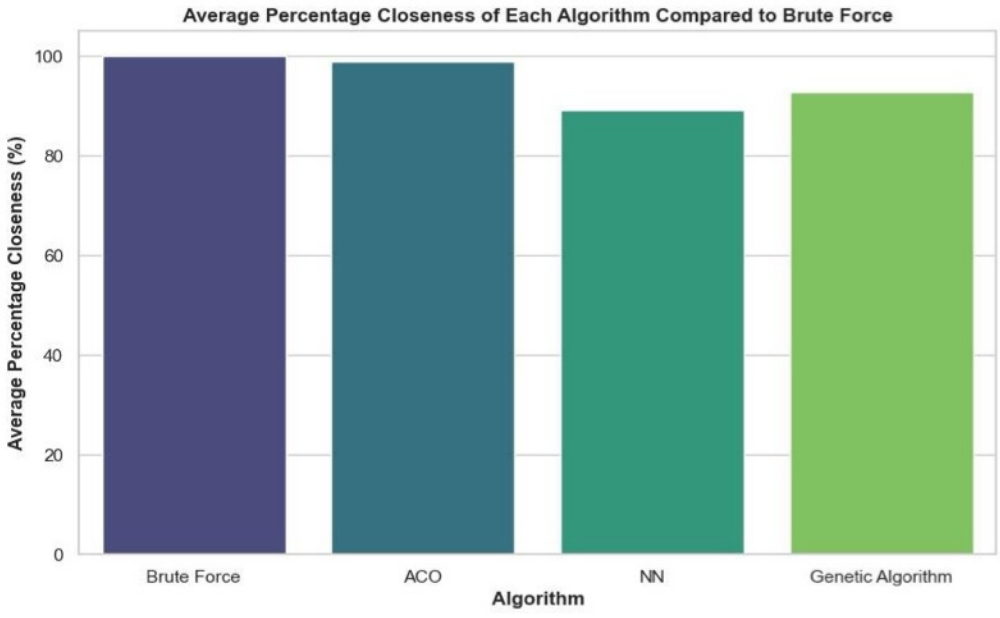
	301.55	301.716	337.48	308.2166

DIFFERENT DATASET HAS BEEN TESTED OVER THIS AND RESULTS ARE NOTED



Average percentage closeness of each algorithm compared to Brute Force:

Brute Force	100.000000
ACO	98.788695
NN	89.065470
Genetic Algorithm	92.860283



Based on the visualizations and percentage closeness data provided, here are concluding paragraphs summarizing the insights and observations from the comparison of algorithm performance across datasets:

The comparative analysis of algorithm performance across multiple datasets reveals several key insights. The bar chart depicting the raw performance values of the Brute Force, ACO, NN, and Genetic Algorithm across four datasets shows noticeable variations. Dataset1 exhibits

the highest values across all algorithms, with Brute Force and ACO closely matched, followed by NN and the Genetic Algorithm. In contrast, Dataset2 presents significantly lower values, indicating that this dataset poses less computational complexity or is less demanding for these algorithms.

When examining the average percentage closeness of each algorithm compared to Brute Force, which is considered the baseline, the ACO algorithm demonstrates the closest performance with an average closeness of 98.79%. This indicates that ACO nearly matches the Brute Force algorithm's performance while likely offering superior efficiency. The Genetic Algorithm and NN also show strong performance, with average closeness values of 92.86% and 89.07%, respectively. These results suggest that while Brute Force remains a robust approach, ACO, NN, and the Genetic Algorithm offer competitive alternatives with varying degrees of tradeoffs between efficiency and performance.

Overall, these insights can guide future algorithm selection and optimization efforts. For scenarios where computational resources are constrained, and efficiency is paramount, ACO may serve as a suitable alternative to Brute Force. Meanwhile, NN and the Genetic Algorithm provide additional options, particularly when specific characteristics of the dataset favor their unique approaches. Future work could further explore the conditions under which each algorithm excels, ensuring optimal performance across diverse datasets and applications.

7.THE MODIFIED DYNAMIC APPROACH

Dynamic nature of traffic makes it necessary to re-compute tsp each time after reaching each location on the list.

Traditional tsp : Given a set of locations and the distance/time between every pair of location, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

Initialization:

- Define a list of all cities.
- Define a matrix to store the distances between each pair of cities.

Brute Force Approach/OTHER META-HEURISTICS ALGORITHMS

- Generate all possible permutations of the cities.
- For each permutation, calculate the total distance of the route.
- Track the permutation with the shortest total distance.

Return the Result:

- Return the shortest route and its total distance.

CORRELATION:

Correlation is a statistical measure that describes the degree to which two variables move in relation to each other. It is often used to understand the relationship between variables in various fields such as finance, economics, and science. The most common measure of correlation is the Pearson correlation coefficient, which ranges from -1 to 1.

The time matrices compared varies by 90 mins.

The correlation of time matrices for the set of locations has been identified,however there is a minute deviation which in turn affects the ordering /scheduling of locations if computed again.Thus making the necessity to change over traditional tsp computation in sector of logistics and delivery.

Correlation Matrix of Time Matrices:

[[1. 0.97520678] [0.97520678 1.]]

MODIFIED VERSION OF COMPUTATION:[BECAUSE OF VARIATIONS IN TIME MATRIX]

1) Initialize the set of coordinates for the locations.

2) Retrieve the time matrix using the best_guess method and the current departure time to reflect accurate traffic conditions.

3) Solve the TSP using a meta-heuristic algorithm.

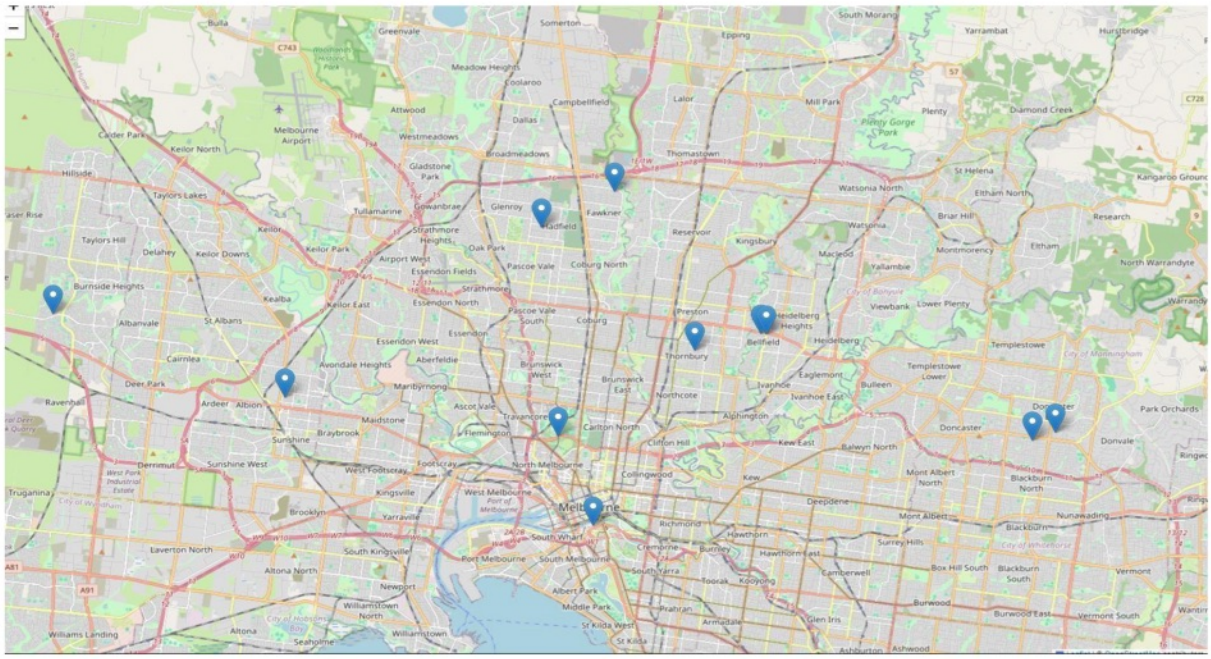
4) Store the result and fetch the updated time matrix. Check the percentage change in travel times to assess if the updates are significant.

5) If the correlation is less than 0.97,

6) Recompute and solve the TSP, then store the new result.

7)Retrieve all the first elements from the stored results.

INPUTDATASET2:



FEW ITERATIONS USING BRUTE-FORCE

```
Initial Distance Matrix:
[[ 0.         8.08333333 17.13333333 20.03333333 13.3        25.51666667
  15.33333333 31.48333333 30.96666667 16.95       16.6        ]
 [ 7.86666667  0.         17.06666667 25.18333333 18.58333333 25.43333333
  16.83333333 32.15       32.2        18.56666667 18.18333333]
 [18.01666667 17.3         0.         22.46666667 15.43333333 14.03333333
  23.75       35.03333333 34.33333333 25.16666667 24.95       ]
 [21.33333333 26.88333333 24.56666667  0.         12.65       26.86666667
  16.76666667 22.83333333 21.91666667 17.88333333 18.75       ]
 [16.45       19.13333333 18.95       13.63333333  0.21666667 28.55
  13.66666667 20.21666667 19.11666667 15.05       15.96666667]
 [25.13333333 24.36666667 13.43333333 26.86666667 24.5         0.
  30.61666667 44.48333333 43.05       32.23333333 31.96666667]
 [15.56666667 17.28333333 23.13333333 17.9        14.61666667 31.33333333
  0.4        20.21666667 19.41666667 6.98333333 6.46666667] [31.26666667
 31.06666667 35.55       23.55       20.85       46.
  19.75       0.         2.63333333 16.95       17.15       ]
 [31.08333333 32.45       34.88333333 23.         20.21666667 45.33333333
  19.15       3.05       0.         17.33333333 16.66666667]
 [16.46666667 18.08333333 23.95       20.83333333 18.93333333 32.23333333
  6.08333333 16.4        15.93333333 0.08333333 1.8        ]
 [16.4        18.01666667 23.9        20.83333333 18.11666667 32.18333333
  6.15       17.45       16.95       1.31666667  0.        ]]
Initial Shortest Route:  [0, 1, 2, 5, 3, 4, 8, 7, 10, 9, 6, 0]
Initial Total Travel Time:  140.98333333333332  minutes
```

Travel Time from First to Second Node: 8.083333333333334 minutes
Updated Current Time: 2024-07-22 20:04:21.720130 Updated
Distance Matrix:
[[0. 8.2 17.05 19.93333333 13.41666667 25.33333333
15.25 31.4 30.85 16.81666667 16.53333333]
[7.83333333 0. 17.08333333 25.03333333 18.55 25.3
16.71666667 32.86666667 32.23333333 18.46666667 18.08333333]
[18.06666667 17.3 0. 22.2 15.4 13.95
23.71666667 35.03333333 34.06666667 25.1 24.91666667]
[21.3 26.76666667 24.48333333 0. 12.55 26.88333333
16.65 22.71666667 21.73333333 17.71666667 18.58333333]
[16.36666667 18.96666667 18.91666667 13.5 0. 28.36666667
13.6 20.1 19.11666667 15.15 16.03333333]
[25.11666667 24.31666667 13.36666667 26.78333333 24.48333333 0.
30.63333333 44.66666667 43.5 32.2 32.]
[15.6 17.31666667 23.11666667 17.86666667 14.56666667 31.46666667
0. 20.16666667 19.33333333 6.9 6.5]
[31.28333333 31.13333333 35.55 23.56666667 20.83333333 45.16666667
19.85 0. 2.65 16.8 16.98333333]
[30.88333333 32.28333333 35.05 23.01666667 20.3 45.25
19.18333333 3.05 0. 17.21666667 16.61666667]
[16.41666667 18.03333333 23.8 20.78333333 18.83333333 32.1
6.05 16.3 15.83333333 0. 1.8]
[16.38333333 17.96666667 23.65 20.71666667 17.93333333 32.03333333
6.1 17.41666667 16.9 1.3 0.]]
New Shortest Route: [1, 5, 2, 4, 3, 8, 7, 10, 9, 6, 0]CHANGE
New Total Travel Time: 140.48333333333332 minutes
Travel Time from Previous First to Second Node: 25.3 minutes
Updated Current Time for Next Route Calculation: 2024-07-22 20:29:39.720130 Next
Distance Matrix:
[[0. 8.13333333 16.68333333 20.28333333 13.5 25.21666667
14.91666667 30.63333333 30.43333333 16.41666667 16.1]
[7.71666667 0. 16.48333333 25.2 18.36666667 25.01666667
16.35 31.68333333 31.86666667 18.01666667 17.63333333]
[17.71666667 17.03333333 0. 22.63333333 15.58333333 13.71666667
23.46666667 35.86666667 35. 24.78333333 24.66666667]
[20.96666667 26.38333333 24.41666667 0. 12.11666667 27.03333333
16.05 22.75 21.61666667 17.36666667 18.21666667]
[15.95 19.23333333 18.75 13.05 0. 27.78333333
13.15 20.36666667 19.06666667 15.01666667 15.98333333]
[25.05 24.26666667 13.25 27.01666667 24.6 0.
30.86666667 44.4 43.66666667 32.38333333 32.01666667]
[15.16666667 16.8 22.36666667 17.7 14.56666667 30.96666667
0. 20.58333333 19.26666667 6.76666667 6.41666667]
[31.01666667 31. 36.35 23.76666667 21.16666667 47.8
19.96666667 0. 2.81666667 16.61666667 16.8]

[30.58333333 30.78333333 36.51666667 23.03333333 20.5 45.08333333
19.26666667 2.76666667 0. 17. 16.06666667]
[16.15 17.76666667 23.25 20.63333333 18.66666667 31.93333333
5.83333333 16.03333333 15.56666667 0. 1.73333333]
[16.11666667 17.75 23.33333333 20.6 18.2 31.9
5.91666667 17.06666667 16.61666667 1.3 0.]]
Next Shortest Route: [5, 2, 4, 3, 8, 7, 10, 9, 6, 0]
Next Total Travel Time: 130.58333333333334 minutes
Travel Time from Previous First to Second Node: 13.366666666666667 minutes Updated
Current Time for Next Next Route Calculation: 2024-07-22 20:17:43.720130 Next Next
Distance Matrix:
[[0. 8.11666667 17.01666667 20.28333333 13.46666667 25.38333333
15.1 30.91666667 30.6 16.75 16.33333333]
[7.85 0. 16.96666667 25.31666667 18.58333333 25.41666667
16.71666667 32.08333333 31.8 18.43333333 18.03333333]
[17.96666667 17.16666667 0. 22.43333333 15.5 13.81666667
23.58333333 35.43333333 34.6 25.06666667 24.76666667]
[21.18333333 26.65 24.58333333 0. 12.51666667 27.11666667
16.55 23.03333333 21.75 17.78333333 18.66666667]
[16.23333333 18.75 18.8 13.23333333 0. 28.05
13.28333333 20.11666667 18.86666667 15.03333333 15.95]
[24.96666667 24.21666667 13.33333333 26.88333333 24.43333333 0.
30.6 44.46666667 43.51666667 31.9 31.93333333]
[15.55 17.2 22.68333333 17.68333333 14.43333333 31.23333333
0. 20.55 19.26666667 6.91666667 6.48333333]
[31.25 31.21666667 36.3 23.76666667 21.03333333 45.63333333
19.85 0. 2.75 16.81666667 17.]
[30.75 32.05 35.95 23.2 20.48333333 45.21666667
19.23333333 2.93333333 0. 17.1 16.43333333]
[16.26666667 17.9 23.38333333 20.66666667 18.7 31.88333333
5.96666667 16.61666667 15.91666667 0. 1.81666667]
[16.21666667 17.9 23.4 20.65 18.08333333 31.86666667
6.05 17.6 16.96666667 1.3 0.]]
Next Next Shortest Route: [2, 4, 3, 8, 7, 10, 9, 6, 0]
Next Next Total Travel Time: 110.24999999999999 minutes
Travel Time from Previous First to Second Node: 15.4 minutes
Updated Current Time for Next Next Next Route Calculation: 2024-07-22 20:19:45.720130
Next Next Next Distance Matrix:
[[0. 8.1 16.95 20.28333333 13.53333333 25.31666667
15.15 30.81666667 30.55 16.73333333 16.35]
[7.81666667 0. 16.85 25.3 18.58333333 25.35
16.73333333 31.7 31.96666667 18.43333333 18.01666667]
[17.9 17.16666667 0. 22.45 15.53333333 13.83333333
23.65 35.48333333 34.53333333 25. 24.86666667]
[21.13333333 26.63333333 24.58333333 0. 12.5 27.11666667
16.5 23.05 21.68333333 17.76666667 18.63333333]
[16.11666667 18.71666667 18.71666667 13.18333333 0. 27.95

13.25 20.15 18.8 14.93333333 15.88333333]
[24.95 24.2 13.31666667 26.81666667 24.45 0.
30.48333333 44.38333333 43.43333333 32.01666667 31.8]
[15.53333333 17.16666667 22.6 17.71666667 14.38333333 31.15
0. 20.56666667 19.25 6.91666667 6.48333333]
[31.28333333 31.16666667 35.9 23.75 21.1 45.76666667
19.93333333 0. 2.75 16.9 17.08333333]
[30.73333333 31.98333333 35.98333333 23.16666667 20.56666667 45.25
19.25 2.91666667 0. 16.83333333 16.33333333]
[16.2 17.85 23.21666667 20.71666667 18.68333333 31.91666667
5.91666667 16.5 15.85 0. 1.81666667]
[16.18333333 17.85 23.28333333 20.66666667 18.11666667 31.91666667
6.01666667 17.51666667 16.93333333 1.3 0.]]
Next Next Next Shortest Route: [4, 3, 8, 7, 10, 9, 6, 0]
Next Next Next Total Travel Time: 91.14999999999999 minutes

CONTINUED COMPUTING TILL …, FINAL NODE.

INITIAL TSP SOLUTION: [0, 1, 2, 5, 3, 4, 8, 7, 10, 9, 6, 0]

140.9mins

AFTER COMPUTING BY TAKING THE LIVE TRAFFIC CONDITIONS:

THE FIRST NODE OF EVERY ITERATION IS TAKEN:

MODIFIED TSP SOLUTION:[0, 1, 2, 5, 4, 3, 9, 6, 7, 8, 10, 0]

159 mins

This approach ensures that the TSP solutions are continually optimized based on the most current traffic data, enhancing the accuracy and reliability of the route planning.

8.CONCLUSION:

In this thesis, we explored multiple algorithmic approaches to solving the Traveling Salesman Problem (TSP) with applications in logistics and transportation. By leveraging the Google Maps Distance Matrix API, we were able to create precise distance matrices that form the foundation of our analysis. The methodologies investigated include brute force, Ant Colony Optimization (ACO), Genetic Algorithm, and the Nearest Neighbor heuristic.

Each algorithm was evaluated based on its accuracy in finding the optimal route and its computational efficiency. The brute force approach, while providing exact solutions, proved impractical for larger datasets due to exponential growth in computational time. In contrast, heuristic and metaheuristic algorithms like ACO, Genetic Algorithm, and Nearest Neighbor provided near-optimal solutions with significantly reduced computational demands.

The results demonstrated that Ant Colony Optimization and Genetic Algorithms are particularly effective for solving TSP in real-world scenarios, striking a balance between solution quality and computational efficiency. The Nearest Neighbor heuristic, while not always as precise, offered a simple and fast alternative suitable for applications requiring quick approximations.

In conclusion, this study has highlighted the strengths and weaknesses of various TSP solving algorithms, providing a comprehensive framework for addressing complex routing problems in logistics. Future work may involve integrating real-time traffic data and exploring hybrid approaches to further enhance the efficiency and accuracy of TSP solutions in dynamic environments.

9.FUTURE WORK

1. Integration of Real-Time Traffic Data:

- Incorporate real-time traffic information into the TSP algorithms to dynamically adjust routes based on current road conditions. This could significantly enhance the practicality and accuracy of routing decisions in urban environments.

2. Hybrid Algorithm Development:

- Explore the combination of different heuristic and metaheuristic algorithms to create hybrid models that can leverage the strengths of each method. For instance, using a Genetic Algorithm to refine the solutions generated by Ant Colony Optimization.

3. Scalability Enhancements:

- Investigate methods to improve the scalability of the algorithms, allowing them to handle larger datasets efficiently. This could involve parallel computing techniques or distributed computing frameworks to manage the increased computational load.

4. Machine Learning Integration:

- Utilize machine learning techniques to predict traffic patterns and travel times based on historical data. These predictions can then be incorporated into the TSP algorithms to improve route planning accuracy.

5. Customization for Specific Applications:

- Tailor the TSP solutions to specific industry needs, such as logistics, ride-sharing, or delivery services. Customizing the algorithms for specific constraints and requirements of different applications can lead to more effective and practical solutions.

6. Robustness and Reliability:

Conduct extensive testing and validation to ensure the robustness and reliability of the algorithms under various conditions, including different traffic scenarios and unexpected disruptions.

7. Environmental Impact Considerations:

Integrate considerations for environmental impact, such as minimizing fuel consumption and emissions, into the TSP algorithms. This aligns with the growing emphasis on sustainable and eco-friendly logistics solutions.

By addressing these future directions, the project can be extended to provide more comprehensive, efficient, and practical solutions for the Traveling Salesman Problem in realworld applications.

10. REFERENCES:

[1] Halim, A.H., Ismail, I. Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem. Arch Computat Methods Eng 26, 367–380 (2019)

[2] T. Cheong and C. C. White, "Dynamic Traveling Salesman Problem: Value of RealTime Traffic Information," in IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 2, pp. 619-630, June 2012.

[3] Huang, W., Yu, J.X. Investigating TSP Heuristics for Location-Based Services. *Data Sci. Eng.* 2, 71–93 (2017).

[4] IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.2, February 2008 339 Manuscript received February 5, 2008 Manuscript revised February 20, 2008 Named Entity Recognition Approaches Alireza Mansouri, Lilly Suriani Affendey, Ali Mamat.

[5] S. Wang, C. -H. Yang, J. Wu and C. Zhang, "Can Whisper Perform Speech-Based In-Context Learning?," ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Korea, Republic of, 2024.

[6] C. Fu, Y. Wang, Y. Xu and Q. Li, "The logistics network system based on the Google Maps API," 2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM), Harbin, China, 2010.