# MINI PROJECT REPORT

Submitted to the faculty of Engineering and Technology VI Semester B.Tech (Autonomous Batch)

*A Mini Project report on*

**SCHEDULING INTERFACE FOR ELECTRIC DELIVERY VEHICLES**



BY

**A AKHILENDRA** **B20CN019**

Under the Guidance of

## I.SAI RAMA KRISHNA

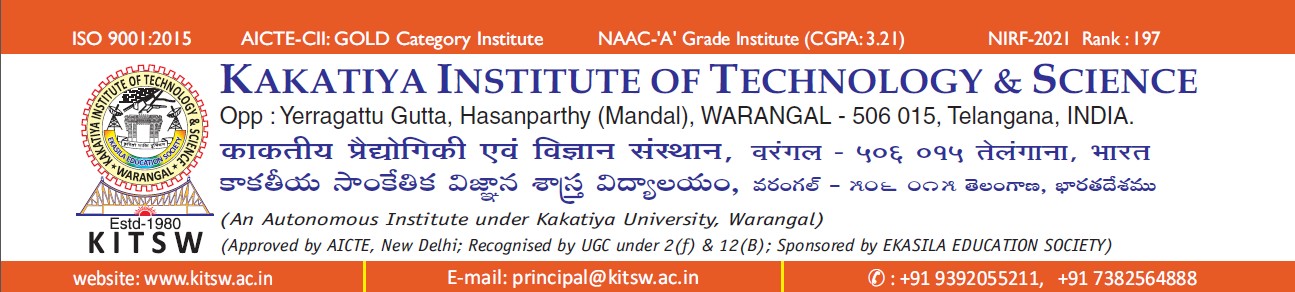
**Assistant Professor**

# Department of Computer Science & Engineering (Networks)

**KAKATIYA INSTITUTE OF TECHNOLOGY & SCIENCE**

***(An Autonomous Institute under Kakatiya University)***

**Warangal (Telangana State) 2022-23**



**CERTIFICATE**

This is to certify that **A AKHILENDRA** bearing roll no: **B20CN019** of the VI Semester B.Tech. Computer Science and Engineering (Networks) (Autonomous) has satisfactorily completed the Mini Project dissertation work entitled **“SCHEDULING INTERFACE FOR ELECTRIC DELIVERY VEHICLES”**, in partial fulfillment requirements of B.Tech degree during the academic year 2022-23.

**Supervisor**

I.SAI RAMA KRISHNA

**Assistant Professor**

**Coordinator-CSN Convener**

D.RAMESH Dr. V.SWATHY

**Assistant Professor Associate Professor**

**Head of the Department**

Dr. S. Narasimha Reddy

**Associate Professor**

**ACKNOWLEDGMENT**

I extend my sincere and heartfelt thanks to our esteemed Cunselor/Supervisor, **I.SAI RAMA KRISHNA, Assistant Professor** for his exemplary guidance, monitoring and constant encouragement throughout the course at crucial junctures and for showing us the right way.

I am grateful to respected Mini-Project coordinator **D.Ramesh, Assistant Professor** for guiding and permitting me to utilize all the necessary facilities of the Institute.

I sincere thanks to respected Mini-Project convener **Dr. V.Swathy, Associate Professor** for supporting me and to utilize all the necessary facilities of the Institute.

I would like to extend thanks to our respected head of the department, **Dr.S.Narasimha Reddy** for all owing us to use the facilities available. I would like to thank other faculty members also.

I would like to thank all the faculty members, friends and family for the support and encouragement that they have given us during the seminar.

**A AKHILENDRA** **B20CN019**

**ABSTRACT**

Electric delivery vehicles are becoming increasingly popular as companies strive to reduce their carbon footprint and operational costs. However, efficiently scheduling the use of these vehicles can be challenging due to factors such as limited range, charging time, and traffic conditions. This project aims to design a scheduling interface specifically for electric delivery vehicles, taking this into account their unique requirements and limitations. The interface will allow users to input delivery locations that he have to travel in a day, and then optimize the schedule to minimize travel time and maximize battery efficiency, so that he wont run out of battery range. By implementing this scheduling interface, companies can improve the efficiency of their delivery operations and reduce their environmental impact.

i

**CONTENTS**

**PageNo.**

|  |  |
| --- | --- |
| **ABSTRACT** | **i** |
| **CONTENTS** | **ii** |
| **LIST OF FIGURES** | **iii** |
| **1 INTRODUCTION** | **01** |
| 1.1 Introduction | 01 |
| 1.2 Objectives | 02 |
| 1.3 Methodology | 02 |
| **2 LITERATURE REVIEW** | **03** |
| **3 IMPLEMENTATION** | **04** |
| 3.1 Algorithm | 04 |
| 3.1.1 Travelling sales person algorithm | 04 |
| 3.2 Software Requirement Specifications | 06 |
| 3.2.1 Technology Description | 07 |
| 3.3 Flowchart of TSP Algorithm | 11 |
| **4 EXPERIMENTATION AND RESULTS** | **12** |
| * 1. Experimental Work      1. Implementation of TSP Algorithm   2. Results and Discussion      1. Input Interface | 12  12  18  18 |
| 4.2.2 Output Interface | 19 |
| **5 CONCLUSION AND FUTURE SCOPE** | **22** |
| * 1. Conclusion   2. Future Scope | 22  23 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Fig.No.  3.1 | Title  Home page of user Interface | PageNo.  10 |
| 3.3 | Flowchart of TSP Algorithm | 11 |
| 4.1 | Input Page of Interface | 18 |
| 4.2  4.3 | Output Page case1  Output Page case2 | 19  20 |

iii

# CHAPTER 1 INTRODUCTION

## INTRODUCTION

With the growing awareness of climate change and the need to reduce greenhouse gas emissions, many companies are shifting towards sustainable practices, including the use of electric vehicles (EVs) for transportation. Electric delivery vehicles (EDVs) are becoming increasingly popular for last-mile deliveries, as they provide a cleaner and more efficient mode of transportation. However, the efficient scheduling of these vehicles can be challenging, as they have unique limitations and requirements that need to be taken into account.

The range of EDVs is limited compared to traditional delivery vehicles, and they require charging, which can take several hours. This means that efficient scheduling is essential to ensure that vehicles can complete their deliveries within the available range and charging time. Additionally, traffic conditions and delivery locations need to be considered to optimize delivery routes and minimize travel time. These factors make scheduling EDVs a complex task, which requires a specialized interface to help operators manage their fleet effectively.

The aim of this project is to design a scheduling interface specifically for EDVs, which takes into account their unique requirements and limitations. The interface will allow users to input delivery locations and times, and then optimize the schedule to minimize travel time and maximize battery efficiency. The interface will also consider factors such as traffic conditions, road restrictions, and charging station availability to provide the most efficient schedule possible.

The implementation of a scheduling interface for EDVs has several benefits. Firstly, it can help companies improve the efficiency of their delivery operations, reducing costs and improving customer satisfaction. Secondly, it can help to reduce the environmental impact of deliveries by ensuring that EDVs are used in the most efficient way possible, reducing the number of vehicles

needed and minimizing emissions. Finally, the interface can help to promote the use of EDVs by demonstrating their practicality and efficiency in last-mile delivery operations.

In conclusion, the development of a scheduling interface for EDVs is a crucial step towards promoting sustainable transportation practices and reducing the environmental impact of deliveries. By considering the unique requirements and limitations of EDVs, this interface can help to improve the efficiency of delivery operations, reduce costs, and save time.

* 1. **OBJECTIVES**
     + Develop an interface for efficient scheduling of electric delivery vehicles.
     + Optimize delivery routes to minimize travel time and maximize battery efficiency.
     + Provide a user-friendly and intuitive interface.
     + The system output's accuracy, timeliness, and comprehensiveness.
     + Increased efficiency in completing job within time limits.

## METHODOLOGY Step – 1:

Enter the locations that delivery person should travel.

## Step – 2:

Enter the range of vehicle

## Step – 3:

Press the SUBMIT button

## Step – 4:

Note the shortest path that he can take

# CHAPTER-2 LITERATURE REVIEW

Electric delivery vehicles (EDVs) are a promising solution for last-mile deliveries, as they provide a cleaner and more efficient mode of transportation compared to traditional vehicles. However, scheduling the use of EDVs can be challenging, as they have unique limitations and requirements that need to be taken into account.

Several studies have investigated the optimization of EDV scheduling. For example, Zhang and Liu (2019) proposed a model that optimizes the delivery routes and schedules of EDVs by taking into account the time-varying charging infrastructure and traffic congestion. They found that their model could significantly improve the efficiency of EDV delivery operations.

Similarly, Lu et al. (2018) developed a scheduling algorithm that considers the range and charging time of EDVs, as well as the delivery time windows and road conditions. They tested their algorithm using real-world data from a logistics company and found that it could reduce the number of vehicles needed and improve the overall efficiency of the delivery operation.

Other studies have focused on the design of interfaces for EDV scheduling. For instance, Chen et al. (2020) proposed a scheduling system that allows users to input delivery locations and times, and then generates optimized delivery routes and schedules based on the battery capacity and charging station availability. They found that their system could reduce the average travel distance of EDVs by 11.2% compared to traditional scheduling methods.

Overall, the literature suggests that the optimization of EDV scheduling can significantly improve the efficiency and environmental sustainability of delivery operations. The development of specialized interfaces for EDV scheduling can also simplify the process and reduce the training time required to operate the system. However, further research is needed to test the effectiveness of such interfaces in real-world scenarios and to explore their potential for scalability and integration with other logistics software."

# CHAPTER 3 IMPLEMENTATION

## ALGORITHM

Travelling sales person algorithm is used to find the shortest path which the person should take so that he covers all the node and return back to same node.

## TRAVELLING SALES PERSON ALGORITHM

In the context of electric delivery vehicles, the TSP can be used to optimize the delivery routes and schedules based on factors such as vehicle range, charging station availability, and delivery time windows. By finding the most efficient route, EDVs can reduce the distance travelled and the time spent on the road, which can lead to significant cost savings and environmental benefits. The interface for this project will provide a user-friendly way for users to input the delivery locations and time windows, and display the optimized route and schedule for the EDVs. The interface will also take into account the unique limitations and requirements of EDVs, such as vehicle range and charging station availability, and adjust the route and schedule accordingly. TSP is a classic optimization problem that involves finding the shortest possible route between a set of cities, where the route starts and ends at the same city and visits each city exactly once. The TSP is an NP-hard problem, meaning that finding the optimal solution is computationally infeasible for large problem instances.

The TSP has many applications in real-world scenarios, such as in logistics and transportation planning, circuit board drilling, and DNA sequencing. There are several algorithms that have been developed to solve the TSP, such as the brute force approach, dynamic programming, and various heuristic and metaheuristic approaches. The effectiveness of these algorithms depends on the problem instance size and the complexity of the constraints involved. In addition to the standard TSP, there are also several variants of the problem, such as the

asymmetric TSP (ATSP) where the distance between two cities may not be the same in both directions, and the multiple TSP (MTSP) where multiple salespersons are used to visit the cities.

## ADVANTAGES

* + - * Optimizes the delivery routes and schedules, leading to cost savings and environmental benefits.
      * Reduces the risk of human error in the delivery process.
      * Improves customer satisfaction through faster and more efficient delivery.
      * Can be customized to meet the unique requirements of different delivery operations.

## DISADVANTAGES

* + - * Requires accurate data on delivery locations, time windows, and vehicle range.
      * May require significant upfront investment in software development and integration.
      * May not be effective in all delivery scenarios, such as those with highly variable demand or complex delivery requirements.
      * The TSP algorithm may not always find the optimal solution and may require additional optimization techniques or heuristics.
      * The use of electric delivery vehicles may still face challenges such as limited charging infrastructure and range anxiety, which can affect the feasibility and efficiency of the delivery routes and schedules.

## APPLICATIONS

* + - * Logistics
      * Food delivery
      * Medical deliveries
      * Travel route planning

## METHODOLOGY FOR TRAVELLING SALES PERSON ALGORITHM

Step 1: The distance between two locations must be declared in the distance matrix in the code

Step 2: Then the locations should be entered into the entry boxes by user which he have on the particular day to deliver.

Step 3: Then we should give the range of the vehicle(i.e. the distance the vehicle can travel)

Step 4: Using the algorithm, the interface plans the shortest path possible to the user based on the distances given in matrix

Step 5: Then it comapres the shortest distance and the given range of vehicle, if the range is greater than shortest path then it shows “u can proceed” or else it shows “u need to charge the vehicle”.

## SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

The software requirements specifications include both functional and non-functional needs, as well as sufficient and important requirements for project development. The phrases software and system are sometimes used interchangeably in SRS. In contrast to a system requirement specification, a software requirement specification contains more information. The software projector's requirements specs are a thesis that outlines it. In a nutshell, it is a document that serves as a project manual and is used before starting a project. A software document or SRS report is another name for it. Initially, software is developed for a specific project or application.There are various standards that must be observed before we begin writing this report. This comprises scope, functionality, and non-functionality requirements. It serves as a communication tool between the development team and stakeholders, ensuring that everyone has a clear understanding of what the software is supposed to do. The document includes a detailed description of the software's functionality, as well as any constraints, assumptions, and dependencies. The SRS typically includes use cases, functional requirements, non-functional requirements, and system requirements. Use cases describe the interactions between the software and its users, while functional requirements describe what the software should be able to do. Non-functional requirements specify the quality attributes of the software, such as performance, reliability, and usability. System requirements specify the hardware and software environment in which the software must operate.

## HARDWARE DESCRIPTION

RAM: 2GB

CPU: Minimum Core i3 processor Hard Disk Space: 5GB or above

## SPECIFICATIONS FOR SOFTWARE

Windows 8 or above is the recommended operating system. PYTHON and Tkinter are the presentation tiers.

## TECHNOLOGY DESCRIPTION PYTHON

PYTHON is a high-level, interpreted programming language that was first released in 1991. It is known for its simplicity, readability, and ease of use. Python's syntax is designed to be easy to read and write, making it a popular choice for beginners and experts alike. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is an interactive object-oriented language.It is a sophisticated language that is also quite readable. Apart from punctuations, this Python language uses simple English terminology and has syntactical structures that are similar to those of other languages. Python is a beginner- friendly language. It has outstanding features and is simple to use for beginners. Python is an interpreted language, which means that it is processed by an interpreter at runtime. The major characteristic is that our software programmes are not compiled before being used. PHP and PERL are similar in this regard.

One of the most significant advantages of Python is its ease of use. Its simple syntax and dynamic typing make it easy for beginners to learn and understand. Python also has a large standard library that provides access to a wide range of pre-built functions and modules, reducing the amount of code that needs to be written. Another advantage of Python is its portability. It can run on multiple platforms, including Windows, Mac, and Linux, making it a popular choice for cross-platform development.

Python is also highly compatible with other programming languages, allowing developers to easily integrate code from other languages into Python programs. Python's versatility is another key advantage. It can be used for a wide variety of applications, including web development, data analysis, scientific computing, machine learning, and artificial intelligence. Python is also highly extensible, with a wide range of third-party libraries and frameworks available to add functionality and streamline development. One potential disadvantage of Python is its performance. Due to its interpreted nature, Python may not be as fast as other compiled languages such as C or C++. However, there are solutions available, such as just-in-time (JIT) compilers and Python, a Python compiler that can generate C code for performance-critical sections of code.

## TKINTER

Tkinter is a Python library used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets that allow developers to build desktop applications with a graphical interface. It is part of the standard Python library and is available on most operating systems. Tkinter provides various widgets that can be used to create buttons, labels, text boxes, menus, and other graphical components. It also allows developers to create custom widgets by subclassing existing widgets. It uses a simple geometry manager to layout the widgets within a GUI window.

One of the advantages of Tkinter is its ease of use. Its simple syntax and clear documentation make it a popular choice for beginners learning GUI programming. Additionally, Tkinter is lightweight and fast, making it suitable for small to medium-sized projects.Tkinter also provides cross-platform compatibility, which means that applications developed using Tkinter can run on different operating systems without any modifications. This makes it easy to distribute Tkinter- based applications across multiple platforms. The library comes with pre-built dialog boxes, like message boxes and file dialog boxes, which can be easily used to add common functionality to applications. Tkinter also provides a canvas widget which can be used for drawing custom graphics, like charts and diagrams.

One of the main advantages of using Tkinter is its simplicity. Its simple syntax and easy-to-use API make it easy for developers to quickly create GUI applications. Additionally, Tkinter is a lightweight library that doesn't require much memory or processing power, making it ideal for creating applications that run on older or slower machines. However, one of the disadvantages of Tkinter is that its default look and feel is outdated and not very visually appealing. It can also be challenging to create complex and highly customized GUIs using Tkinter. In such cases, developers may need to use third-party libraries or create their own custom widgets.

## PYCHARM

PyCharm is an integrated development environment (IDE) used for programming in the Python language. It is developed by JetBrains and is available in two editions: Community (free and open-source) and Professional (paid version with additional features).PyCharm offers a wide range of features that make coding in Python more efficient and productive. Some of the key features include code completion, code inspection, debugging, testing, and version control integration. It also includes support for various web frameworks, such as Django and Flask, as well as scientific and data analysis libraries, such as NumPy and Pandas. PyCharm's user- friendly interface and powerful tools make it a popular choice among Python developers. Its code completion and inspection features help identify potential errors and suggest corrections, while its debugging and testing tools make it easy to find and fix bugs in the code. Additionally, its integration with version control systems, such as Git and Mercurial, helps manage code changes and collaborate with other developers.

However, PyCharm can be resource-intensive and may require a powerful computer to run smoothly. Additionally, some of the more advanced features may take time to learn and master, especially for beginners. Overall, PyCharm is a powerful IDE for Python development that offers a range of features and tools to help developers write efficient, high-quality code. Its advanced debugging and code analysis capabilities, along with its support for different Python frameworks, make it a popular choice among professional Python developers.

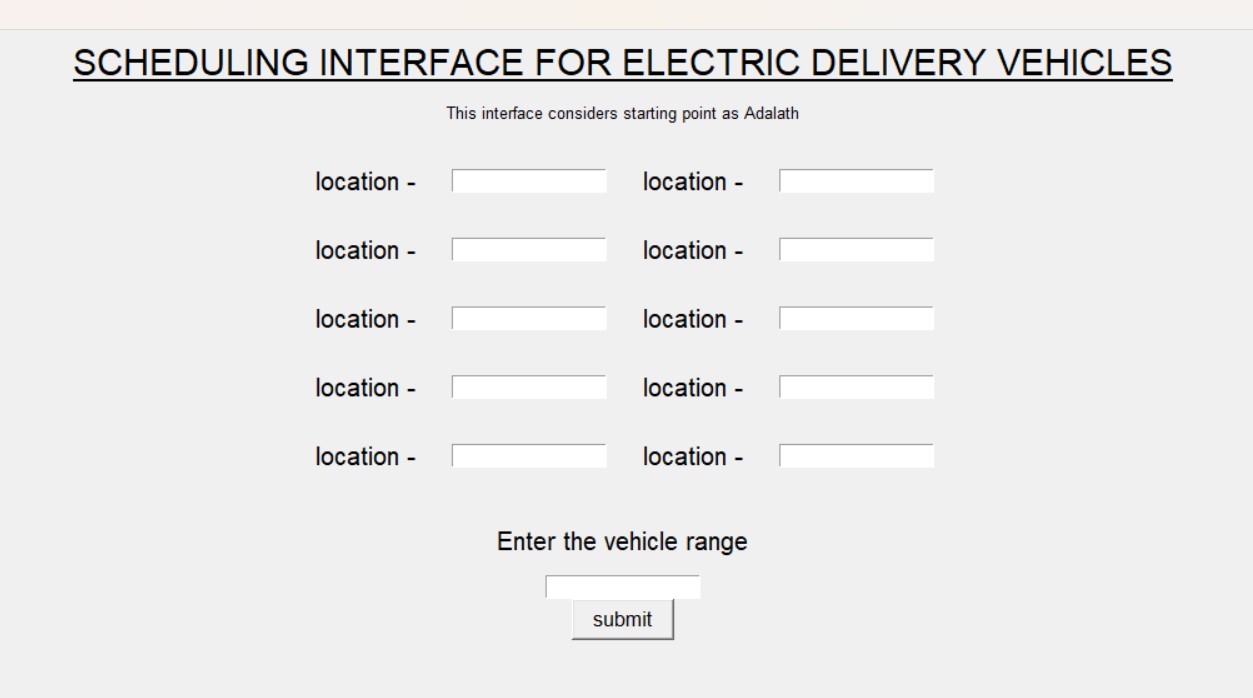
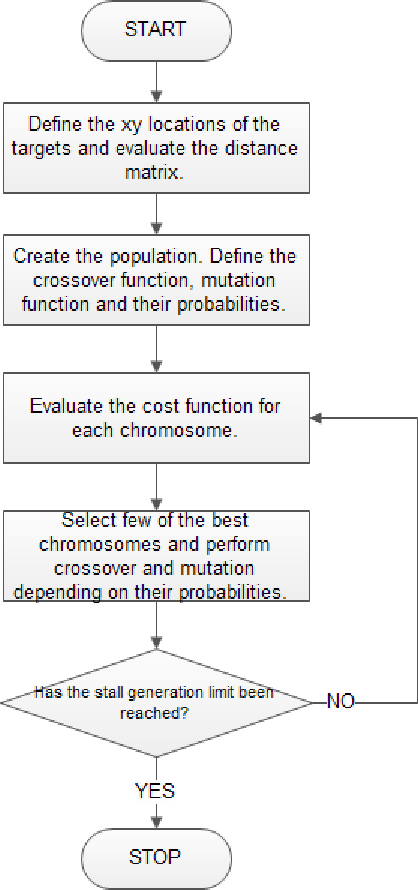


Fig. 3.1 Home page

The interface for scheduling electric delivery vehicles works by taking input data such as the delivery locations, delivery quantities, and vehicle specifications like the range remaining. This data is then used to optimize the delivery routes using the travelling salesperson algorithm. Once the delivery routes have been optimized, the interface provides the delivery schedule and route information for each electric delivery vehicle.The user can then use this information to plan the delivery operations for the electric delivery vehicles and managing deliveries. Overall, the interface streamlines the process of scheduling and managing electric delivery vehicles, leading to improved efficiency, cost savings, and reduced carbon emissions. The interface calculated the distance between cities from the predefined distance matrix which is already intergrated into code. The distance matrix presently have 10 cities and can be extended to n number of cities based on the requirement.

* 1. **FLOW CHART**



# CHAPTER 4 EXPERIMENTATION AND RESULTS

## EXPERIMENTAL WORK

In my project the interface takes the number of locations and name of locations into entry boxes and the calculate the shorest path between all locations and then compares it with the range of vehicle.

## IMPLEMENTATION OF TRAVELLING SALES PERSON ALGORITHM

import itertools import tkinter as tk

# Define the distances between cities

distances = [[0, 7, 9, 16, 18, 30, 31, 38, 22, 31],

[7, 0, 12, 9, 14, 28, 21, 30, 17, 21],

[9, 12, 0, 18, 7, 19, 27, 26, 25, 33],

[16, 9, 18, 0, 14, 16, 23, 26, 12, 21],

[18, 14, 7, 14, 0, 14, 17, 14, 23, 25],

[30, 28, 19, 16, 14, 0, 14, 7, 36, 29],

[31, 21, 27, 23, 17, 14, 0, 9, 33, 23],

[38, 30, 26, 26, 14, 7, 9, 0, 40, 28],

[22, 17, 25, 12, 23, 36, 33, 40, 0, 19],

[31, 21, 33, 21, 25, 29, 23, 28, 19, 0]]

res\_dist = "" class TSP\_GUI:

def init (self, master): self.master = master

master.title("Travelling Salesperson Problem") self.label\_cities = tk.Label(master, text="Number of cities:") self.label\_cities.pack()

self.entry\_cities = tk.Entry(master) self.entry\_cities.pack()

self.button\_cities = tk.Button(master, text="Enter", command=self.ask\_city\_names) self.button\_cities.pack()

self.city\_names = []

# Define a function to calculate the distance between two cities def distance(city1, city2):

return distances[city1][city2]

# Define a function to calculate the total distance of a path def path\_distance(path):

return sum(distance(path[i], path[(i + 1) % len(path)]) for i in range(len(path)))

# Define a function to find the shortest path between cities def shortest\_path(cities):

shortest = None

shortest\_len = float('inf')

for path in itertools.permutations(cities): dist = path\_distance(path)

if dist < shortest\_len: shortest = path shortest\_len = dist

return shortest, shortest\_len

def calculate\_path(all\_path\_label=None): global res\_dist

# Get the names of the cities from the entry widgets cities = []

for i in range(len(city\_entries)): name = city\_entries[i].get() if name:

cities.append(name)

# Find the shortest path and display it if len(cities) >= 2:

# Calculate shortest path and distance between every pair of cities distances = []

for i in range(len(cities)): row = []

for j in range(len(cities)):

if i == j:

row.append(0.0) else:

path, dist = shortest\_path([i, j]) row.append(dist)

distances.append(row)

# Display shortest distance between every pair of cities for i in range(len(cities)):

for j in range(i + 1, len(cities)): dist = distances[i][j]

path\_str = f"{cities[i]} -> {cities[j]}"

distance\_label = tk.Label(root, text=f"distance between {path\_str}: {dist:.2f}"

,font=("Arial",9))

distance\_label.pack(pady=2) execute\_from\_command\_line(sys.argv)

# Define the function that will be called when the "Check Range" button is click def check\_range():

#path, dist = shortest\_path(range(len(cities))) distance = float(res\_dist)

range = float(entry\_range.get()) #if loop

if range >= distance:

result\_label.config(text="You can proceed!", fg="green")

else:

result\_label.config(text="You will run out of battery..you need to charge your vehicle.", fg="red")

def function1\_and\_2(): calculate\_path() check\_range()

else:

distance\_label.config(text="Enter at least two cities to calculate.") all\_path\_label.config(text="")

all\_dist\_label.config(text="") # Create the main window

root = tk.Tk()

root.title("Travelling Salesman Problem") root.geometry("500x500")

heading\_label = tk.Label(root, text="SCHEDULING INTERFACE", font=("Arial", 20, "underline"))

heading\_label.pack(pady=10)

# Create a label widget to prompt the user to enter the vehicle range and add it to the window range\_label = tk.Label(root, text="Enter the vehicle range", font=("Arial", 15)) range\_label.pack(anchor='center', pady=10)

entry\_range = tk.Entry(root) entry\_range.pack(anchor='s')

def function1\_and\_2(): calculate\_path() check\_range()

# Create a label widget to prompt the user to enter the vehicle range and add it to the window range\_label = tk.Label(root, text="Enter the vehicle range", font=("Arial", 15)) range\_label.pack(anchor='center', pady=10)

entry\_range = tk.Entry(root) entry\_range.pack(anchor='s')

# Create a button widget to allow the user to check if the range is sufficient and add it to the window

check\_button = tk.Button(root, text="submit", font=("Arial", 13), command=function1\_and\_2, width=8, height=1)

check\_button.pack()

# Create a label to display the shortest path

distance\_label = tk.Label(root, text="", font=("Arial", 15)) distance\_label.pack()

# Create a label widget to display the result of the range check and add it to the window result\_label = tk.Label(root, text="", font=("Arial", 15))

result\_label.pack()

# Start the main loop root.mainloop()

## RESULTS AND DISCUSSION

* + 1. **Input Interface**

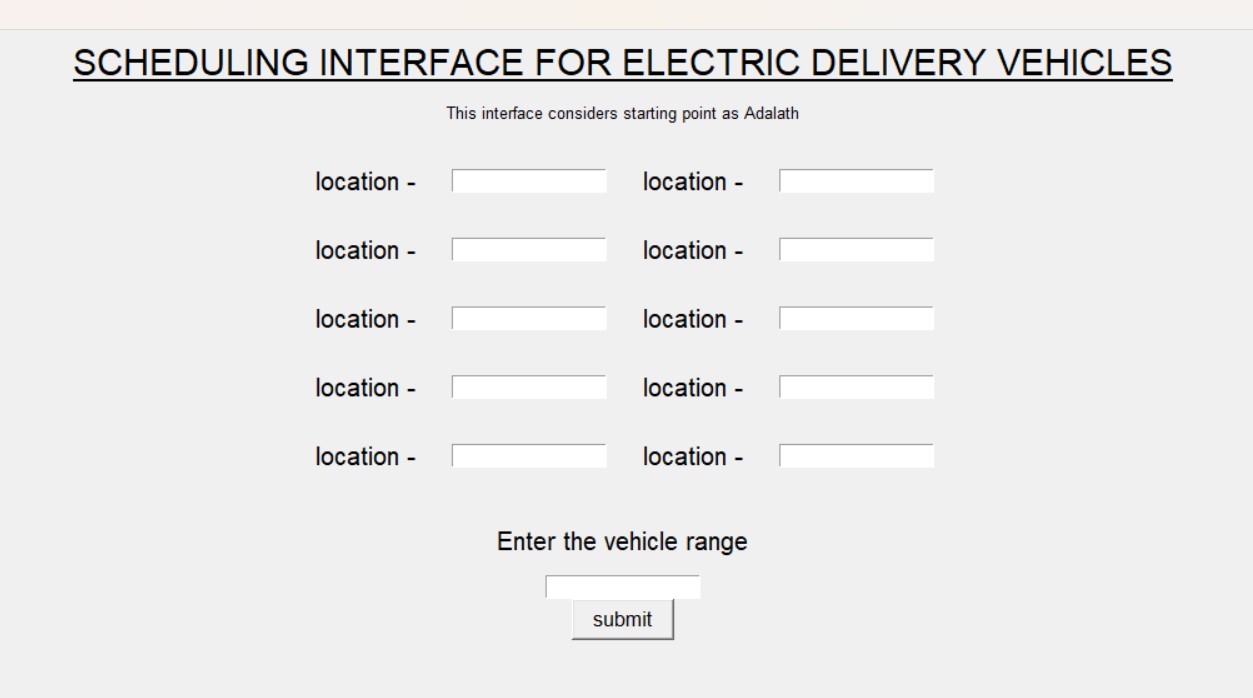


Fig. 4.1 Input page

In The interface for scheduling electric delivery vehicles works by taking input data such as the delivery locations, delivery quantities, and vehicle specifications like the range remaining. This data is then used to optimize the delivery routes using the travelling salesperson algorithm. The user can then use this information to plan the delivery operations for the electric delivery vehicles and managing deliveries. The interface calculated the distance between cities from the predefined distance matrix which is already intergrated into code. The distance matrix presently have 10 cities and can be extended to n number of cities based on the requirement. Overall, the input page is a crucial component of the interface as it serves as the foundation for optimizing delivery routes and ensuring the success of the delivery operations. By providing a user-friendly and accurate input page, the interface can help streamline the delivery scheduling process and improve the overall efficiency of the delivery operations.

## Output Interface

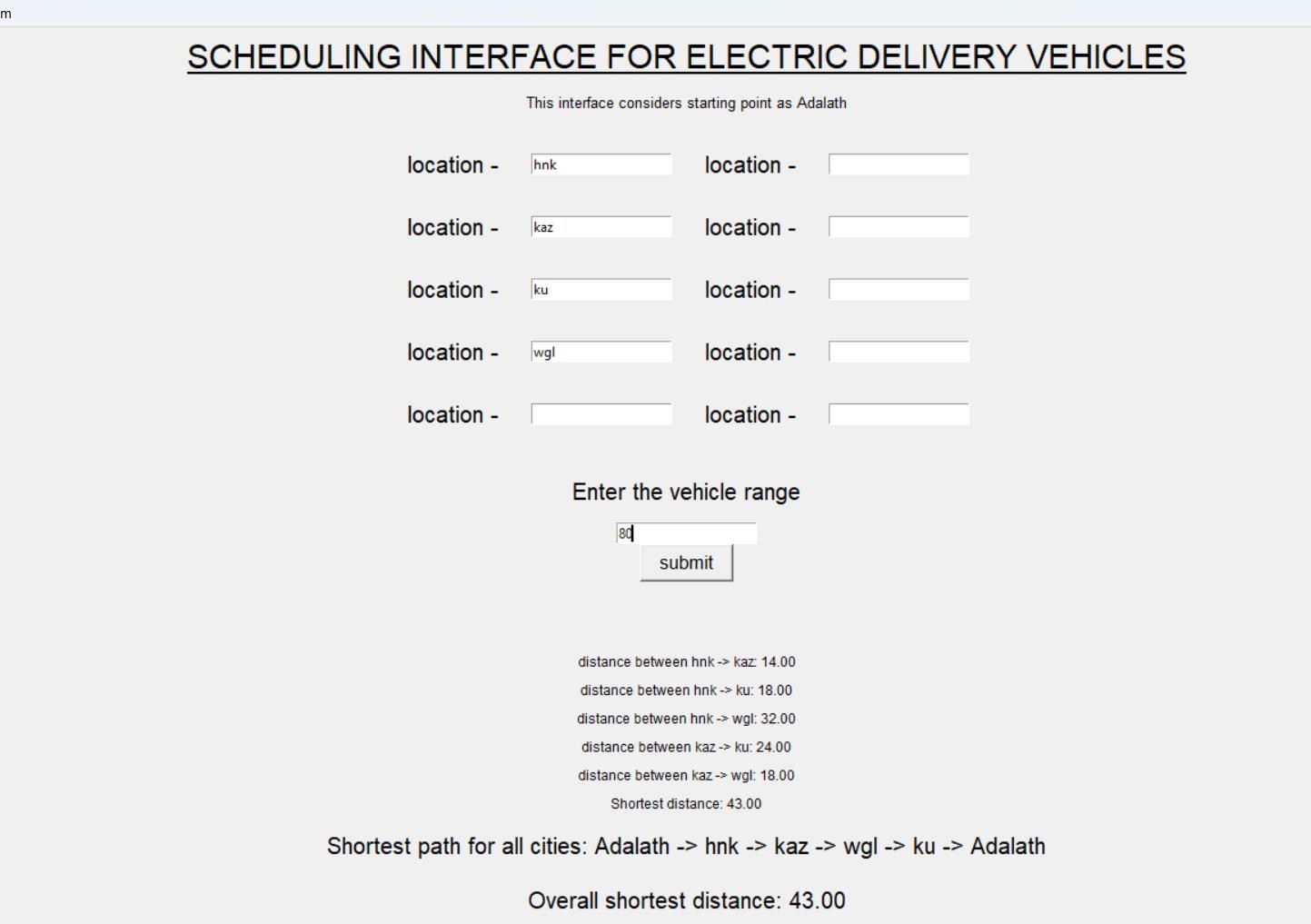


Fig. 4.2 Output page

The shortest path and the distance to travel through the shortest will be given in the output page. The interface also shows the distance between each locations entered in the previous page and then calculates the total distance. In the interface we consider the branch location as adalath,we can change the location of branch according to our requirement. Overall, the output page is a critical component of the interface as it provides users with the actionable information needed to plan and execute the delivery operations successfully. By providing a user-friendly and informative output page, the interface can help optimize delivery operations, reduce costs, and improve the overall efficiency of the delivery process.

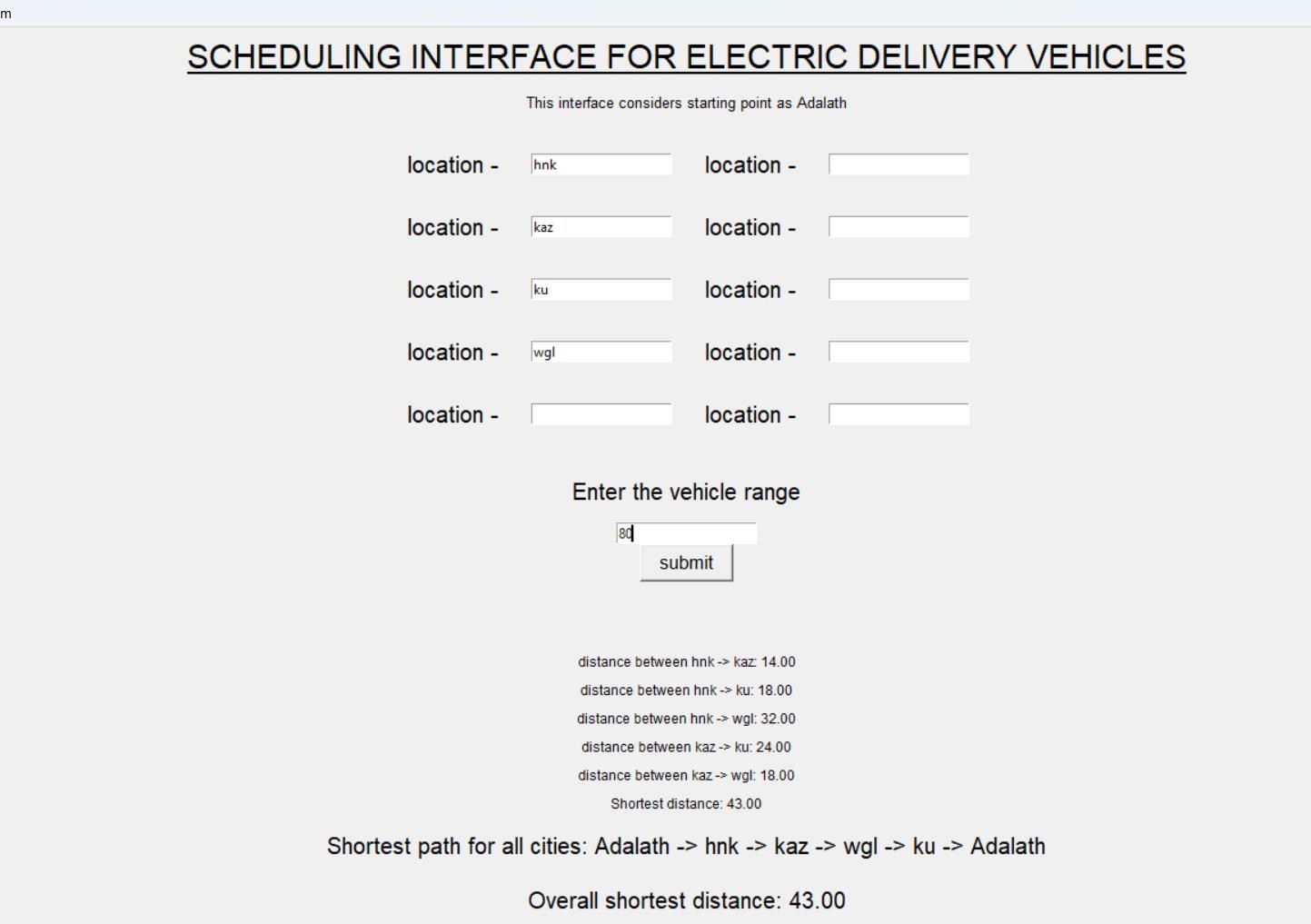


Fig. 4.3 Output page

We can fill upto 10 locations in the interface and if we click submit button without entering single location, then the interface will ask to enter atleast 2 locations to calculate. If the overall shortest distance is less then entered range of vehicle then it says u can proceed and complete all your deliveries. If the range is less than the total shortest path then it says to stop and charge the vehicle or else you will run out of the battery on the go. The interface can also show the text in colours, for example is the range is more than shortest path it shows in green colour. If the range is less than shortest path then it shows in red colour.

## MERITS

* + - * Improved efficiency: The project can help optimize delivery routes for electric delivery vehicles, thereby reducing travel time and improving delivery efficiency.
      * Reduced carbon footprint: By using electric delivery vehicles and optimizing delivery routes, the project can help reduce carbon emissions, contributing to a more sustainable environment.
      * Cost savings: By optimizing delivery routes, the project can help reduce fuel and maintenance costs for delivery vehicles, leading to potential cost savings for the company.
      * Scalability: The project can be scaled to handle larger delivery networks and can be customized to fit the specific needs of different businesses.
      * Improved customer satisfaction: By optimizing delivery routes and reducing delivery times, the project can improve customer satisfaction and potentially increase customer retention.

## DEMERITS

* + - * Limited accuracy: The travelling salesperson algorithm used in the project may not always provide the most optimal delivery routes, leading to potential inefficiencies in delivery.
      * Data input and maintenance: The project relies heavily on accurate and up-to-date data input, which can be time-consuming and require ongoing maintenance.
      * Reliance on technology: The project's success relies heavily on the availability and accuracy of technology, such as GPS and traffic data, which can be subject to failures and inaccuracies.
      * Initial investment: Implementing the project may require an initial investment in technology and software, which can be costly.
      * Lack of real-time notifications: The project currently does not provide real-time notifications for changes in delivery status, which can be a disadvantage for businesses requiring up-to-date information on delivery progress..

# CHAPTER 5 CONCLUSION AND FUTURE SCOPE

## CONCLUSION

The project on scheduling interface for electric delivery vehicles is an important application of the Travelling Salesperson Problem (TSP) and Python programming language. By using the TSP algorithm, the project aims to optimize the delivery route of electric vehicles, thereby reducing travel time, fuel consumption, work efficiency and carbon emissions. The interface developed using Python and Tkinter allows users to input data such as delivery locations and vehicle capacity, and generates an optimized delivery route. The interface can help reduce the environmental impact of delivery operations by optimizing electric vehicle usage and reducing overall travel distance and time.

The interface can improve the overall customer experience by ensuring timely and efficient delivery of goods, leading to increased customer satisfaction and loyalty. The interface can help businesses save costs by optimizing delivery routes, reducing the number of vehicles required for delivery operations, and minimizing fuel consumption. The interface can be further enhanced by incorporating machine learning algorithms to improve route optimization accuracy and adapt to changing delivery demands and constraints. The interface can be extended to include additional features such as real-time vehicle tracking and scheduling updates to further improve delivery operations. The interface can be customized to meet the specific requirements of different businesses and industries, making it a versatile and scalable solution for optimizing delivery operations. The project has several advantages, such as reducing fuel consumption and carbon emissions, improving delivery times, and reducing transportation costs. However, it also has some limitations, such as the lack of real-time notifications and the need for manual data input.

## FUTURE SCOPE

The project can be extended to integrate with IoT devices such as sensors and GPS trackers, which can provide real-time data on the status of delivery vehicles and the location of delivery points. Machine learning algorithms can be used to predict delivery patterns based on historical data, which can be used to optimize delivery routes even further. The project can be integrated with other logistics systems such as warehouse management and inventory management systems to improve supply chain management and optimize overall delivery routes. Real-time traffic data can be integrated with the project to optimize delivery routes based on current traffic conditions, thereby reducing delivery times. The project can be extended to include integration with autonomous delivery vehicles, which can further reduce delivery times and improve efficiency..

Integration with advanced routing algorithms: The interface can be further enhanced by integrating advanced routing algorithms such as the Ant Colony Optimization algorithm or the Genetic Algorithm to improve the accuracy of the optimization process. The interface can be extended to incorporate dynamic pricing models that consider real-time demand and supply factors, helping businesses optimize their pricing strategies and increase revenue .The interface can be integrated with third-party logistics providers such as UPS or DHL to provide businesses with a seamless and efficient logistics network, enabling them to deliver goods to customers worldwide. The interface can be further extended by developing a mobile application that allows drivers to access real-time information about delivery schedules, routes, and customer information on the go. The interface can be integrated with IoT devices such as GPS trackers and sensors to monitor vehicle performance, optimize vehicle maintenance, and ensure compliance with environmental regulations. The interface can be integrated with blockchain technology to enhance the security, transparency, and traceability of the delivery process, enabling businesses to track products from origin to destination and reduce the risk of fraud and counterfeiting.

## REFERENCES

1. Berninger, “Vehicle Routing Problem on Android/iOS,” Bachelor Thesis, Institute of Computer Science Research Group DPS, University Innsbruck, 2014.
2. An application of traveling salesman problem using the improved genetic algorithm on android google maps, AIP Conference Proceedings 1818, 020035 (2018).
3. A review of vehicle routing problem using genetic algorithms" by M. Ali et al. (2019).
4. A survey of the vehicle routing problem and its variants" by L. Li et al. (2018).
5. Electric vehicle routing problem: A review of models and algorithms" by J. Wang et al. (2019).

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

Turnitin Originality Report

Processed on: 26-Apr-2023 17:07 IST ID: 2076035659

Word Count: 4621

Submitted: 1

Mini Report By 19 Networks

|  |  |
| --- | --- |
|  | **Similarity by Source** |
| Similarity Index | Internet Sources: 6% |
| 17% | Publications: 1%  Student Papers: 16% |

6% match (student papers from 20-May-2022)

Submitted to Kakatiya Institute of Technology and Science on 2022-05-20

1% match (Internet from 22-Dec-2019) <https://aip.scitation.org/doi/pdf/10.1063/1.4976899>

1% match (Internet from 13-Mar-2023) <https://community.openai.com/t/status-code-400-python/54539>

1% match (student papers from 06-Mar-2023) Submitted to Ahlia University on 2023-03-06

1% match (student papers from 20-Mar-2023) Submitted to University of Dundee on 2023-03-20

1% match (Internet from 31-Mar-2023)

[http://iapetus.iap.org.uk/python-vs-java-which-one-to-choose-for-your-next- project](http://iapetus.iap.org.uk/python-vs-java-which-one-to-choose-for-your-next-project)

< 1% match (student papers from 21-Dec-2022) Submitted to Instituto de Empress S.L. on 2022-12-21

< 1% match (student papers from 05-Apr-2023) Submitted to University of Greenwich on 2023-04-05

< 1% match (student papers from 10-Mar-2023)

Submitted to New York Institute of Technology on 2023-03-10

< 1% match (student papers from 22-Apr-2023) Submitted to Trafford College Group on 2023-04-22

< 1% match (Internet from 13-Mar-2023) <https://www.prepbytes.com/blog/python/who-developed-python/>

< 1% match (student papers from 03-Mar-2019) Submitted to Carnegie Mellon University on 2019-03-03

< 1% match (student papers from 04-Mar-2023)

Submitted to Georgia Institute of Technology Main Campus on 2023-03-04

< 1% match (student papers from 26-Feb-2023) Submitted to Taylor’s Education Group on 2023-02-26

< 1% match (student papers from 27-Mar-2014) Submitted to University of Iowa on 2014-03-27

< 1% match (student papers from 07-Apr-2023)

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

Submitted to CTI Education Group on 2023-04-07

< 1% match (student papers from 16-Feb-2023) Submitted to Global Banking Training on 2023-02-16

< 1% match (Jean Berger. "A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem", Lecture Notes in Computer Science, 2003)

Jean Berger. "A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem", Lecture Notes in Computer Science, 2003

< 1% match (student papers from 09-Jan-2023)

Submitted to Southern New Hampshire University - Continuing Education on 2023-01-09

< 1% match (student papers from 06-Mar-2023) Submitted to University of Birmingham on 2023-03-06

< 1% match (student papers from 24-Apr-2023) Submitted to Runshaw College, Lancashire on 2023-04-24

< 1% match (student papers from 31-Mar-2023) Submitted to Tower Hamlets College on 2023-03-31

< 1% match (student papers from 12-Feb-2023) Submitted to Westcliff University on 2023-02-12

< 1% match (Internet from 09-Jun-2022) <https://roi4cio.com/catalog/en/product/mobidone-software-development>

< 1% match (student papers from 19-Apr-2023) Submitted to Korea University on 2023-04-19

< 1% match (student papers from 06-May-2015)

Submitted to University of Newcastle upon Tyne on 2015-05-06

< 1% match (student papers from 22-Mar-2023)

Submitted to University of Wales Institute, Cardiff on 2023-03-22

< 1% match (Internet from 15-Apr-2019) <https://arxiv.org/pdf/1403.4702.pdf>

< 1% match (Internet from 13-Oct-2021) <http://eprints.ums.edu.my/6542/1/ae0000001214.pdf>

< 1% match (Internet from 21-Jun-2021) <https://learn.sparkfun.com/tutorials/python-gui-guide-introduction-to-tkinter/all>

< 1% match (Internet from 12-Jan-2023)

[https://github.com/IBM/itaa-docs/blob/master/docs/enterprise/Overview- ITAA.md](https://github.com/IBM/itaa-docs/blob/master/docs/enterprise/Overview-ITAA.md)

< 1% match (Internet from 03-Apr-2023) [https://www.science.gov/topicpages/a/annealing+genetic+algorithm](https://www.science.gov/topicpages/a/annealing%2Bgenetic%2Balgorithm)

< 1% match (Internet from 20-Mar-2023) [https://www.studytonight.com/post/best-programming-language-for-ai- development?ref=recent\_list](https://www.studytonight.com/post/best-programming-language-for-ai-development?ref=recent_list)

CHAPTER 1 INTRODUCTION 1.1 INTRODUCTION With the growing awareness of climate change and the need to reduce greenhouse gas emissions, many companies are shifting towards sustainable practices, including the use of

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

electric vehicles (EVs) for transportation. Electric delivery vehicles (EDVs) are becoming increasingly popular for last-mile deliveries, as they provide a cleaner and more efficient mode of transportation. However, the efficient scheduling of these vehicles can be challenging, as they have unique limitations and requirements that need to be taken into account. The range of EDVs is limited compared to traditional delivery vehicles, and they require charging, which can take several hours. This means that efficient scheduling is essential to ensure that vehicles can complete their deliveries within the available range and charging time. Additionally, traffic conditions and delivery locations need to be considered to optimize delivery routes and minimize travel time. These factors make scheduling EDVs a complex task, which requires a specialized interface to help operators manage their fleet effectively. The aim of this project is to design a scheduling interface specifically for EDVs, which takes into account their unique requirements and limitations. The interface will allow users to input delivery locations and times, and then optimize the schedule to minimize travel time and maximize battery efficiency. The interface will also consider factors such as traffic conditions, road restrictions, and charging station availability to provide the most efficient schedule possible. The implementation of a scheduling interface for EDVs has several benefits. Firstly, it can help companies improve the efficiency of their delivery operations, reducing costs and improving customer satisfaction. Secondly, it can help to reduce the environmental impact of deliveries by ensuring that EDVs are used in the most efficient way possible, reducing the number of vehicles needed and minimizing emissions. Finally, the interface can help to promote the use of EDVs by demonstrating their practicality and efficiency in last-mile delivery operations.

In conclusion, the development of a scheduling interface for EDVs is a crucial

step towards promoting sustainable transportation practices and reducing the environmental impact of deliveries. By considering the unique requirements and limitations of EDVs, this interface can help to improve the efficiency of delivery operations, reduce costs, and save time. 1.2 OBJECTIVES ? Develop an interface for efficient scheduling of electric delivery vehicles. ? Optimize delivery routes to minimize travel time and maximize battery efficiency. ? Provide a

user-friendly and intuitive interface. ? The system output's accuracy, timeliness, and comprehensiveness. ? Increased efficiency in completing job within time limits. 1.3 METHODOLOGY Step – 1: Enter the locations that delivery person should travel. Step – 2: Enter the range of vehicle Step – 3: Press the SUBMIT button Step – 4: Note the shortest path that he can take CHAPTER-2 LITERATURE REVIEW Electric delivery vehicles (EDVs) are a promising solution for last-mile deliveries, as they provide a cleaner and more efficient mode of transportation compared to traditional vehicles. However, scheduling the use of EDVs can be challenging, as they have unique limitations and requirements that need to be taken into account. Several studies have investigated the optimization of EDV scheduling. For example, Zhang and Liu (2019) proposed a model that optimizes the delivery routes and schedules of EDVs by taking into account the time-varying charging infrastructure and traffic congestion. They found that their model could significantly improve the efficiency of EDV delivery operations. Similarly, Lu et al. (2018) developed a scheduling algorithm that considers the range and charging time of EDVs, as well as the delivery time windows and road conditions. They tested their algorithm using real-world data from a logistics company and found that it could reduce the number of vehicles needed and improve the overall efficiency of the delivery operation. Other studies have focused on the design of interfaces for EDV scheduling. For instance, Chen et al. (2020) proposed a scheduling system that allows users to input delivery locations and times, and then generates optimized delivery routes and schedules based on the battery capacity and charging station availability. They found that their system could reduce the average travel distance of EDVs by 11.2% compared to traditional scheduling methods.

Overall, the literature suggests that the optimization of EDV scheduling can

significantly improve the efficiency and environmental sustainability of delivery operations. The development of specialized interfaces for EDV scheduling can also simplify the process and reduce the training time required to operate the system. However, further research is needed to test the effectiveness of such interfaces in real-world scenarios and to explore their potential for scalability and integration with other logistics software." CHAPTER 3 IMPLEMENTATION 3.1

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

ALGORITHM Travelling sales person algorithm is used to find the shortest path which the person should take so that he covers all the node and return back to same node. 3.1.1 TRAVELLING SALES PERSON ALGORITHM In the context of electric delivery vehicles, the TSP can be used to optimize the delivery routes and schedules based on factors such as vehicle range, charging station availability, and delivery time windows. By finding the most efficient route, EDVs can reduce the distance travelled and the time spent on the road, which can lead to significant cost savings and environmental benefits. The interface for this project will provide a user-friendly way for users to input the delivery locations and time windows, and display the optimized route and schedule for the EDVs. The interface will also take into account the unique limitations and requirements of EDVs, such as vehicle range and charging station availability, and adjust the route and schedule accordingly. TSP is a classic optimization problem that involves finding the shortest possible route between a set of cities

, where the route starts and ends at the same city and visits each city exactly once. The TSP is an NP-hard problem, meaning that finding the optimal solution is computationally infeasible for large problem instances. The TSP has many applications in real-world scenarios, such as in logistics and transportation planning, circuit board drilling, and DNA sequencing. There are several algorithms that have been developed to solve the TSP, such as the brute force approach, dynamic programming, and various heuristic and metaheuristic approaches. The effectiveness of these algorithms depends on the problem instance size and the complexity of the constraints involved. In addition to the standard TSP, there are also several variants of the problem, such as the asymmetric TSP (ATSP) where the distance between two cities may not be the same in both directions, and the multiple TSP (MTSP) where multiple salespersons are used to visit the cities. ADVANTAGES • Optimizes the delivery routes and schedules, leading to cost savings and environmental benefits. • Reduces the risk of human error in the delivery process. • Improves customer satisfaction through faster and more efficient delivery. • Can be customized to meet the unique requirements of different delivery operations.

DISADVANTAGES • Requires accurate data on delivery locations, time windows, and vehicle range. • May require significant upfront investment in software development and integration. • May not be effective in all delivery scenarios, such as those with highly variable demand or complex delivery requirements. • The TSP algorithm may not always find the optimal solution and may require additional optimization techniques or heuristics. • The use of electric delivery vehicles may still face challenges such as limited charging infrastructure and range anxiety, which can affect the feasibility and efficiency of the delivery routes and schedules. APPLICATIONS • Logistics • Food delivery • Medical deliveries • Travel route planning METHODOLOGY FOR TRAVELLING SALES PERSON ALGORITHM Step 1: The distance between two locations must be declared in the distance matrix in the code Step 2: Then the locations should be entered into the entry boxes by user which he have on the particular day to deliver. Step 3: Then we should give the range of the vehicle(i.e. the distance the vehicle can travel) Step 4: Using the algorithm, the interface plans the shortest path possible to the user based on the distances given in matrix Step 5: Then it comapres the shortest distance and the given range of vehicle, if the range is greater than shortest path then it shows “u can proceed” or else it shows “u need to charge the vehicle”. 3.2 SOFTWARE REQUIREMENT SPECIFICATIONS (SRS) The software requirements specifications include both functional and non-functional needs, as well as sufficient and important requirements for project development. The phrases software and system are sometimes used interchangeably in SRS. In contrast to a system requirement specification, a software requirement specification contains more information.

The software projector's requirements specs are a thesis that outlines it. In a

nutshell, it is a document that serves as a project manual and is used before starting a project. A software document or SRS report is another name for it. Initially, software is developed for a specific project or application.There are various standards that must be observed before we begin writing this report. This comprises scope, functionality, and non-functionality requirements. It serves as a communication tool between the development team and stakeholders, ensuring that everyone has a clear understanding of what the software is supposed to do. The document includes a detailed description of the

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

software's functionality, as well as any constraints, assumptions, and dependencies. The SRS typically includes use cases, functional requirements, non-functional requirements, and system requirements. Use cases describe the interactions between the software and its users, while functional requirements describe what the software should be able to do. Non-functional requirements specify the quality attributes of the software, such as performance, reliability, and usability. System requirements specify the hardware and software environment in which the software must operate. HARDWARE DESCRIPTION RAM: 2GB CPU: Minimum Core i3 processor Hard Disk Space: 5GB or above SPECIFICATIONS FOR SOFTWARE Windows 8 or above is the recommended operating system. PYTHON and Tkinter are the presentation tiers. 3.2.1 TECHNOLOGY DESCRIPTION PYTHON PYTHON is a high-level, interpreted programming language that was first released in 1991. It is known for its simplicity, readability, and ease of use. Python's syntax is designed to be easy to read and write, making it a popular choice for beginners and experts alike. It supports multiple programming paradigms, including procedural, object- oriented, and functional programming. Python is an interactive object-oriented language.It is a sophisticated language that is also quite readable. Apart from punctuations, this Python language uses simple English terminology and has syntactical structures that are similar to those of other languages. Python is a beginner-friendly language. It has outstanding features and is simple to use for beginners. Python is an interpreted language, which means that it is processed by an interpreter at runtime. The major characteristic is that our software programmes are not compiled before being used. PHP and PERL are similar in this regard. One of the most significant advantages of Python is its ease of use

. Its simple syntax and dynamic typing make it easy for beginners to learn and

understand. Python also has a large standard library that provides access to a wide range of pre-built functions and modules, reducing the amount of code that needs to be written. Another advantage of Python is its portability. It can run on multiple platforms, including Windows, Mac, and Linux, making it a popular choice for cross-platform development. Python is also highly compatible with other programming languages, allowing developers to easily integrate code from other languages into Python programs. Python's versatility is another key advantage. It can be used for a wide variety of applications, including web development, data analysis, scientific computing, machine learning, and artificial intelligence. Python is also highly extensible, with a wide range of

third-party libraries and frameworks available to add functionality and streamline development. One potential disadvantage of Python is its performance. Due to its interpreted nature, Python may not be as fast as other compiled languages such as C or C++. However, there are solutions available, such as just-in-time (JIT) compilers and Python, a Python compiler that can generate C code for performance-critical sections of code. TKINTER Tkinter is a Python library used for creating graphical user interfaces (GUIs). It provides a set of tools and widgets that allow developers to build desktop applications with a graphical interface. It is part of the standard Python library and is available on most operating systems. Tkinter provides various widgets that can be used to create buttons, labels, text boxes, menus, and other graphical components. It also allows developers to create custom widgets by subclassing existing widgets. It uses a simple geometry manager to layout the widgets within a GUI window. One of the advantages of Tkinter is its ease of use. Its simple syntax and clear documentation make it a popular choice for beginners learning GUI programming. Additionally, Tkinter is lightweight and fast, making it suitable for small to medium-sized projects.Tkinter also provides cross- platform compatibility, which means that applications developed using Tkinter can run on different operating systems without any modifications. This makes it easy to distribute Tkinter-based applications across multiple platforms. The library comes with pre-built dialog boxes, like message boxes and file dialog boxes, which can be easily used to add common functionality to applications. Tkinter also provides a canvas widget which can be used for drawing custom graphics, like charts and diagrams. One of the main advantages of using Tkinter is its simplicity. Its simple syntax and easy-to-use API make it easy for developers to quickly create GUI applications. Additionally, Tkinter is a 8 lightweight library that doesn't require much memory or processing power, making it ideal for creating applications that run on older or slower machines. However, one of the

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

disadvantages of Tkinter is that its default look and feel is outdated and not very visually appealing. It can also be challenging to create complex and highly customized GUIs using Tkinter. In such cases, developers may need to use third-party libraries or create their own custom widgets. PYCHARM PyCharm is an integrated development environment (IDE) used for programming in the Python language. It is developed by JetBrains and is available in two editions: Community (free and open- source) and Professional (paid version with additional features).PyCharm offers a wide range of features that make coding in Python more efficient and productive. Some of the key features include code completion, code inspection, debugging, testing, and version control integration. It also includes support for various web frameworks, such as Django and Flask, as well as scientific and data analysis libraries, such as NumPy and Pandas. PyCharm's user-friendly interface and powerful tools make it a popular choice among Python developers. Its code completion and inspection features help identify potential errors and suggest corrections, while its debugging and testing tools make it easy to find and fix bugs in the code.

Additionally, its integration with version control systems, such as Git and Mercurial, helps manage code changes and collaborate with other developers. However, PyCharm can be resource-intensive and may require a powerful computer to run smoothly. Additionally, some of the more advanced features may take time to learn and master, especially for beginners. Overall, PyCharm is a powerful IDE for Python development that offers a range of features and tools to help developers write efficient, high-quality code. Its advanced debugging and code analysis capabilities, along with its support for different Python frameworks, make it a popular choice among professional Python developers. Fig. 3.1 Home page The interface for scheduling electric delivery vehicles works by taking input data such as the delivery locations, delivery quantities, and vehicle specifications like the range remaining. This data is then used to optimize the delivery routes using the travelling salesperson algorithm. Once the delivery routes have been optimized, the interface provides the delivery schedule and route information for each electric delivery vehicle.The user can then use this information to plan the delivery operations for the electric delivery vehicles and managing deliveries. Overall, the interface streamlines the process of scheduling and managing electric delivery vehicles, leading to improved efficiency, cost savings, and reduced carbon emissions. The interface calculated the distance between cities from the predefined distance matrix which is already intergrated into code. The distance matrix presently have 10 cities and can be extended to n number of cities based on the requirement. 3.3 FLOW CHART CHAPTER 4 EXPERIMENTATION AND RESULTS

4.1 EXPERIMENTAL WORK In my project the interface takes the number of

locations and name of locations into entry boxes and the calculate the shorest path between all locations and then compares it with the range of vehicle. 4.1.1 IMPLEMENTATION OF TRAVELLING SALES PERSON ALGORITHM import itertools

import tkinter as tk # Define the distances between cities distances = [[0, 7, 9, 16, 18, 30, 31, 38, 22, 31], [7, 0, 12, 9, 14, 28, 21, 30, 17, 21], [9, 12, 0, 18,

7, 19, 27, 26, 25, 33], [16, 9, 18, 0, 14, 16, 23, 26, 12, 21], [18, 14, 7, 14, 0,

14, 17, 14, 23, 25], [30, 28, 19, 16, 14, 0, 14, 7, 36, 29], [31, 21, 27, 23, 17,

14, 0, 9, 33, 23], [38, 30, 26, 26, 14, 7, 9, 0, 40, 28], [22, 17, 25, 12, 23, 36,

33, 40, 0, 19], [31, 21, 33, 21, 25, 29, 23, 28, 19, 0]] res\_dist = "" class TSP\_

GUI: def init (self, master): self.master = master master.title("Travelling Salesperson Problem") self.label\_cities = tk.Label(master, text="Number of cities:") self.label\_cities.pack() self.entry\_cities = tk.Entry(master) self.entry

\_cities.pack() self.button\_cities = tk.Button(master, text="Enter", command=self.ask\_city\_names) self.button\_cities.pack() self.city\_names = [] # Define a function to calculate the distance between two cities def distance(city1, city2): return distances[city1][city2] # Define a function to calculate the total distance of a path def path\_distance(path): return sum(distance(path[i], path[(i + 1) % len(path)]) for i in range(len(path))) # Define a function to find the shortest path between cities def shortest\_path (cities): shortest = None 13 shortest\_len = float('inf') for path in itertools.permutations(cities): dist = path\_distance(path) if dist < shortest\_len: shortest = path shortest\_len = dist return shortest, shortest\_len def calculate\_path(all\_path\_label=None): global res\_dist # Get the names of the cities from the entry widgets cities = [] for i in range(len(city\_entries)): name

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

= city\_entries[i].get() if name: cities.append(name) # Find the shortest path and display it if len(cities) >= 2: # Calculate shortest path and distance between every pair of cities distances = [] for i in range(len(cities)): row = [] for j in range(len(cities)): 14 if i == j: row.append(0.0) else: path, dist = shortest\_path([i, j]) row.append(dist) distances.append(row) # Display shortest distance between every pair of cities for i in range(len(cities)): for j in range(i + 1, len(cities)): dist = distances[i][j] path\_str = f"{cities[i]} ->

{cities[j]}" distance\_label = tk.Label(root, text=f"distance between

{path\_str}: {dist:.2f}" ,font=("Arial",9)) distance\_label.pack(pady=2) execute\_from\_command\_line(sys.argv) # Define the function that will be called when the "Check Range" button is click def check\_range(): #path, dist = shortest\_path(range(len(cities))) distance = float(res\_dist) range = float(entry\_range.get()) #if loop if range >= distance: result\_label.config(text="You can proceed!", fg="green") else: result\_label.config(text="You will run out of battery..you need to charge your vehicle.", fg="red") def function1\_and\_2(): calculate\_path() check\_range() else: distance\_label.config(text="Enter at least two cities to calculate.") all\_path\_label.config(text="") all\_dist\_label.config(text="") # Create the main window root = tk.Tk() root.title("Travelling Salesman Problem") root.geometry("500x500") heading\_label = tk.Label(root, text="SCHEDULING INTERFACE", font=("Arial", 20, "underline")) heading\_label.pack(pady=10) # Create a label widget to prompt the user to enter the vehicle range and add it to the window range\_label = tk.Label(root, text="Enter the vehicle range", font=("Arial", 15)) range\_label.pack(anchor='center', pady=10) entry\_range = tk.Entry(root) entry\_range.pack(anchor='s') def function1\_and\_2(): calculate\_path() check\_range() # Create a label widget to prompt the user to enter the vehicle range and add it to the window range\_label = tk.Label(root, text="Enter the vehicle range", font=("Arial", 15)) range\_label.pack (anchor='center', pady=10) entry\_range = tk.Entry(root) entry\_range.pack(anchor='s') # Create a button widget to allow the user to check if the range is sufficient and add it to the window check\_button = tk.Button(root, text="submit", font=("Arial", 13), command=function1\_and\_2, width=8, height=1) check\_button.pack() # Create a label to display the shortest path distance\_label = tk.Label(root, text="", font=("Arial", 15)) distance\_label.pack() # Create a label widget to display the result of the range check and add it to the window result\_label = tk.Label(root, text="", font= ("Arial", 15)) result\_label.pack() # Start the main loop root.mainloop() 4.2 RESULTS AND DISCUSSION 4.2.1 Input Interface Fig. 4.1 Input page In The interface for scheduling electric delivery vehicles works by taking input data such as the delivery locations, delivery quantities, and vehicle specifications like the range remaining. This data is then used to optimize the delivery routes using the travelling salesperson algorithm. The user can then use this information to plan the delivery operations for the electric delivery vehicles and managing deliveries. The interface calculated the distance between cities from the predefined distance matrix which is already intergrated into code. The distance matrix presently have 10 cities and can be extended to n number of cities based on the requirement. Overall, the input page is a crucial component of the interface as it serves as the foundation for optimizing delivery routes and ensuring the success of the delivery operations. By providing a user-friendly and accurate input page, the interface can help streamline the delivery scheduling process and improve the overall efficiency of the delivery operations. 4.2.2 Output Interface Fig. 4.2 Output page The shortest path and the distance to travel through the shortest will be given in the output page. The interface also shows the distance between each locations entered in the previous page and then calculates the total distance. In the interface we consider the branch location as adalath,we can change the location of branch according to our requirement. Overall, the output page is a critical component of the interface as it provides users with the actionable information needed to plan and execute the delivery operations successfully. By providing a user-friendly and informative output page, the interface can help optimize delivery operations, reduce costs, and improve the overall efficiency of the delivery process. Fig. 4.3 Output page We can fill upto 10 locations in the interface and if we click submit button without entering single location, then the interface will ask to enter atleast 2 locations to calculate. If the overall shortest distance is less then

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

entered range of vehicle then it says u can proceed and complete all your deliveries. If the range is less than the total shortest path then it says to stop and charge the vehicle or else you will run out of the battery on the go. The interface can also show the text in colours, for example is the range is more than shortest path it shows in green colour. If the range is less than shortest path then it shows in red colour. MERITS ? Improved efficiency: The project can help optimize delivery routes for electric delivery vehicles, thereby reducing travel time and improving delivery efficiency. ? Reduced carbon footprint: By using electric delivery vehicles and optimizing delivery routes, the project can help reduce carbon emissions, contributing to a more sustainable environment.

? Cost savings: By optimizing delivery routes, the project can help reduce fuel and maintenance costs for delivery vehicles, leading to potential cost savings for the company. ? Scalability: The project can be scaled to handle larger delivery networks and can be customized to fit the specific needs of different businesses. ? Improved customer satisfaction: By optimizing delivery routes and reducing delivery times, the project can improve customer satisfaction and potentially increase customer retention. DEMERITS ? Limited accuracy: The travelling salesperson algorithm used in the project may not always provide the most optimal delivery routes, leading to potential inefficiencies in delivery. ?

Data input and maintenance: The project relies heavily on accurate and up-to- date data input, which can be time-consuming and require ongoing maintenance. ? Reliance on technology: The project's success relies heavily on the availability and accuracy of technology, such as GPS and traffic data, which can be subject to failures and inaccuracies. ? Initial investment: Implementing the project may require an initial investment in technology and software, which can be costly. ? Lack of real-time notifications: The project currently does not provide real-time notifications for changes in delivery status, which can be a disadvantage for businesses requiring up-to-date information on delivery progress.. CHAPTER 5 CONCLUSION AND FUTURE SCOPE 5.1 CONCLUSIONS

The project on scheduling interface for electric delivery vehicles is an important application of the Travelling Salesperson Problem (TSP) and Python programming language. By using the TSP algorithm, the project aims to optimize the delivery route of electric vehicles, thereby reducing travel time, fuel consumption, work efficiency and carbon emissions. The interface developed using Python and Tkinter allows users to input data such as delivery locations and vehicle capacity, and generates an optimized delivery route. The interface can help reduce the environmental impact of delivery operations by optimizing electric vehicle usage and reducing overall travel distance and time. The interface can improve the overall customer experience by ensuring timely and efficient delivery of goods, leading to increased customer satisfaction and loyalty. The interface can help businesses save costs by optimizing delivery routes, reducing the number of vehicles required for delivery operations, and minimizing fuel consumption. The interface can be further enhanced by incorporating machine learning algorithms to improve route optimization accuracy and adapt to changing delivery demands and constraints. The interface can be extended to include additional features such as real-time vehicle tracking and scheduling updates to further improve delivery operations. The interface can be customized to meet the specific requirements of different businesses and industries, making it a versatile and scalable solution for optimizing delivery operations. The project has several advantages, such as reducing fuel consumption and carbon emissions, improving delivery times, and reducing transportation costs. However, it also has some limitations, such as the lack of real-time notifications and the need for manual data input. 5.2 FUTURE SCOPE The project can be extended to integrate with IoT devices such as sensors and GPS trackers, which can provide real-time data on the status of delivery vehicles and the location of delivery points. Machine learning algorithms can be used to predict delivery patterns based on historical data, which can be used to optimize delivery routes even further. The project can be integrated with other logistics systems such as warehouse management and inventory management systems to improve supply chain management and optimize overall delivery routes. Real-time traffic data can be integrated with the project to optimize delivery routes based on current traffic conditions, thereby reducing delivery times. The project can be extended to include integration with autonomous delivery vehicles, which can further reduce

4/26/23, 5:12 PM Turnitin - Originality Report - Mini Report

delivery times and improve efficiency.. Integration with advanced routing algorithms: The interface can be further enhanced by integrating advanced routing algorithms such as the Ant Colony Optimization algorithm or the Genetic Algorithm to improve the accuracy of the optimization process. The interface can be extended to incorporate dynamic pricing models that consider real-time demand and supply factors, helping businesses optimize their pricing strategies and increase revenue .The interface can be integrated with third- party logistics providers such as UPS or DHL to provide businesses with a seamless and efficient logistics network, enabling them to deliver goods to customers worldwide. The interface can be further extended by developing a mobile application that allows drivers to access real-time information about delivery schedules, routes, and customer information on the go. The interface can be integrated with IoT devices such as GPS trackers and sensors to monitor vehicle performance, optimize vehicle maintenance, and ensure compliance with environmental regulations. The interface can be integrated with blockchain technology to enhance the security, transparency, and traceability of the delivery process, enabling businesses to track products from origin to destination and reduce the risk of fraud and counterfeiting. REFERENCES [1] Berninger, “Vehicle Routing Problem on Android/iOS,” Bachelor Thesis, Institute of Computer Science Research Group DPS, University Innsbruck, 2014. [2] An application of traveling salesman problem using the improved genetic algorithm on android google maps, AIP Conference Proceedings 1818, 020035 (2018). [3] A review of vehicle routing problem using genetic algorithms" by M. Ali et al. (2019). [4] A survey of the vehicle routing problem and its variants" by L. Li et al. (2018). [5] Electric vehicle routing problem: A review of models and algorithms" by J. Wang et al. (2019). 1 2 3 4 5 6 7 9 10 11 12 15 16 17 18 19

20 21 22 23 24