T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

DYNAMIC ROUTE PLANNER

MUHARREM OZAN YEŞILLER

SUPERVISOR
PROF. DR. FATIH ERDOĞAN SEVİLGEN

GEBZE
2022

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

DYNAMIC ROUTE PLANNER

MUHARREM OZAN YEŞILLER

SUPERVISOR
PROF. DR. FATIH ERDOĞAN SEVİLGEN

2022
GEBZE

GEBZE TECHNICAL UNIVERSITY	GRADUATION PROJECT JURY APPROVAL FORM

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 31/08/2021 by the following jury.

JURY

Member (Supervisor)	:	Prof. Dr. Fatih Erdoğan SEVİLGEN
Member	:	Prof. Dr. Fatih Erdoğan SEVİLGEN
Member	:	Prof. Dr. Erchan APTOULA

ABSTRACT

Nowadays, smartphones have become the most important part of our daily life. Smartphones are getting stronger day by day and are equipped with various content that will be useful to people. One of them is the map applications that drivers generally use. The majority of the use of map applications; The goal is to reach optimized routes from one point to another. By following these routes, the user reaches their destination at reasonable costs. Such applications also calculate routes from one point to many points in order, but present these routes to the user in the order of the given points.

Calculating a route with the condition of passing many routes is a complex problem and provides a solution in non-deterministic polynomial time. Meta-heuristic algorithms, on the other hand, consist of designs that use heuristic approaches, created to find a good solution to such problems.

As smartphones improve in computing power over time, solving complex problems is becoming more common. The motivation behind this work is to create an optimized route by stopping from one point to another point to other points via smartphones and map applications libraries. The points of contact may increase, decrease or change their location. Providing a dynamic optimization that adapts to such changing conditions is another motivation of this study.

In this study, an online platform is presented that allows drivers to reach their destination by collecting passengers and in this context, calculates the route with dynamic optimization techniques. This system consists of a cross-platform application that includes driver and passenger interfaces.

ÖZET

Günümüzde akıllı telefonlar günlük hayatımızın en önemli parçası haline geldi. Akıllı telefonlar her geçen gün daha da güçlenmekte ve insanların işine yarayacak çeşitli içeriklerle donatılmaktadır. Bunlardan birisi de genel olarak sürüclerin kullandığı harita uygulamalarıdır. Harita uygulamalarının kullanımının çoğunluğunu; bir noktadan başka bir noktaya giden, optimize edilmiş rotalara ulaşmak amacı oluşturur. Kullanıcı bu rotaları takip ederek hedeflerine makul maaliyetlerle ulaşırlar. Bu tarz uygulamalar bir noktadan bir çok noktaya sırayla gitmek amacıyla da rotalar hesaplar fakat bu rotaları, verilen nokta sırasına göre kullanıcıya sunar.

Bir çok rotadan geçmek koşulu ile bir rota hesaplamak kompleks bir problemdir ve deterministik olmayan polinom zamanda çözüm sağlar. Metasezgisel algoritmalar ise bu tarz problemlere iyi bir çözüm bulmak için oluşturulan, sezgisel yaklaşımları kullanan tasarımlardan oluşmaktadır.

Akıllı telefonlar zaman içinde bilgi işleme gücü konusunda geliştikçe, karmaşık problemleri çözdürmek daha yaygın hale geliyor. Bu çalışmanın arkasındaki motivasyon, akıllı telefonlar ve harita uygulamaları kütüphaneleri aracılığıyla bir noktadan bir noktaya, başka noktalara da uğramak suretiyle, optimize edilmiş bir rota oluşturmaktadır. Uğranacak noktalar artabilir, azalabilir ya da yerleri değişebilir. Bu tarz değişken koşullara uyum sağlayan dinamik bir optimizasyon sağlamak bu çalışmanın bir diğer motivasyonudur.

Bu çalışmada, sürücülerin uğrayacağı hedefe, yolcuları toplayarak ulaşmasını sağlayan ve bu bağlamda rotayı dinamik optimizasyon teknikleriyle hesaplayan çevrimiçi bir platform sunulmuştur. Bu sistem sürücü ve yolcu arayüzlerini içeren, çapraz platformlarda çalışan bir uygulamadan oluşmaktadır.

ACKNOWLEDGEMENT

First of all, I would like to thank my advisors Prof. Dr. Fatih Erdoğan SEVİLGEN and Prof. Dr. Erchan APTOULA for their invaluable advice, continuous support and patience during this study. I would also like to thank all academic staffs in the Department of Computer Engineering at Gebze Technical University. Their vast knowledge and experience has always encouraged and led me to success throughout my academic research, industry career and daily life. Finally, I would like to express my gratitude to my family and friends who have supported me throughout my life.

Muharrem Ozan Yeşiller

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
SDK	:	Software Development Kit
API	:	Application Programming Interface
ACO	:	Ant Colony Optimization
ETA	:	Estimated Time of Arrival
TSP	:	Travelling Salesman Problem

CONTENTS

Abstract **iv**

Özet **v**

Acknowledgement **vi**

List of Symbols and Abbreviations **vii**

Contents **ix**

List of Figures **x**

List of Compare Tables and Graphs **xi**

1 INTRODUCTION **1**

1.1 Project Definition 1

1.2 System Requirements 2

1.2.1 Functional Requirements 2

1.2.2 Non-Functional Requirements 3

1.3 Problem Definition 4

1.3.1 Input Domain Set, Solution Format 4

1.3.2 Decision Variables 5

1.3.3 Constraints 5

1.3.4 Objective Function 6

2 META-HEURISTIC DESIGN **7**

2.1 Overview of the Ant Colony Optimization 7

2.1.1 How the Ant Colony Optimization Algorithm Works 7

2.2 Adaption of the Travelling Salesman Problem to a Shortest Path Problem 10

2.2.1 What is Travelling Salesman Problem? 10

2.2.2 Adaptation to the Shortest Path Problem 10

2.3 Dynamic Applying and Design of the Algorithm 12

3 Mobile Application Side **14**

3.0.1 User Interface Side 14

3.0.1.1 Passenger Side 14
3.0.1.2 Driver Side 14
3.0.2 Backend / Database Side 15

4 Comparison Charts **16**

5 Conclusions **19**

Bibliography **20**

LIST OF FIGURES

1.1 Adding new passenger in the optimized route 2
1.2 n is a total vertexes of the graph 5
1.3 Solution constraints formulas 5
1.4 Objective function 6

2.1 n is the amount of pheromone that each ant leaves on the road, and d is the distance between two cities 7
2.2 n is the amount of pheromone that each ant leaves on the road, and d is the distance between two cities 8
2.3 Rho is the evaporation coefficient 8
2.4 n: pheromone, d: distance, alpha: pheromone effect, beta: cost effect . 9
2.5 Travelling Salesman Problem Example 10
2.6 Standart TSP Solution vs Adapted TSP Solution 11

4.1 Time Compare Dynamic ACO and Brute Force 16
4.2 Brute Force Costs 17
4.3 Dynamic ACO Costs 17
4.4 Cost Compare Dynamic ACO and Greedy Algorithm 18

LIST OF TABLES

1. INTRODUCTION

In this study, it is proposed to create a portal where online drivers receive their passengers in an optimized way. The project aims to gather these users together by offering a cross platform. The driver sees the passengers he needs to collect and has the functionality to run algorithms to collect these passengers in an optimized way. The project aims to strengthen optimization by building on previously obtained results from meta-heuristic frameworks. This article is to present powerful optimization design decisions and methods by making use of a software and meta-heuristic technique of the developed portal and previously created routes.

1.1. Project Definition

In this study, an algorithm that performs route planning dynamically, strengthens it using ant colony optimization, and a mobile application that enables users to communicate with each other and run this algorithm has been developed.

The mobile application has separate graphical interfaces for passengers and drivers. It chooses whether the users are passengers or drivers while completing their membership. Drivers can add passengers to their crew. Passengers, on the other hand, can see the instant location of the drivers (if the drivers activate it). On the driver's side, there is an action to run an algorithm for its passengers and destination. When the driver presses this action button, the algorithm runs and produces a to-target result that includes the passengers it has.

The algorithm runs the ant colony algorithm with some heuristics and meta-heuristics. The important point here is that the algorithm works dynamically. This dynamic means: Passengers can change their positions, enter information such as passengers will/will not arrive, and likewise new passengers can be added to this group. As a result of these situations, the algorithm does not work from scratch, using the previously obtained results, only intensifies the search in the relevant places, but does not focus on it blindly, and has the possibility of diversification to create a more optimized route. These procedure is shown in the Figure 2.5. All these situations are handled within the mobile application.

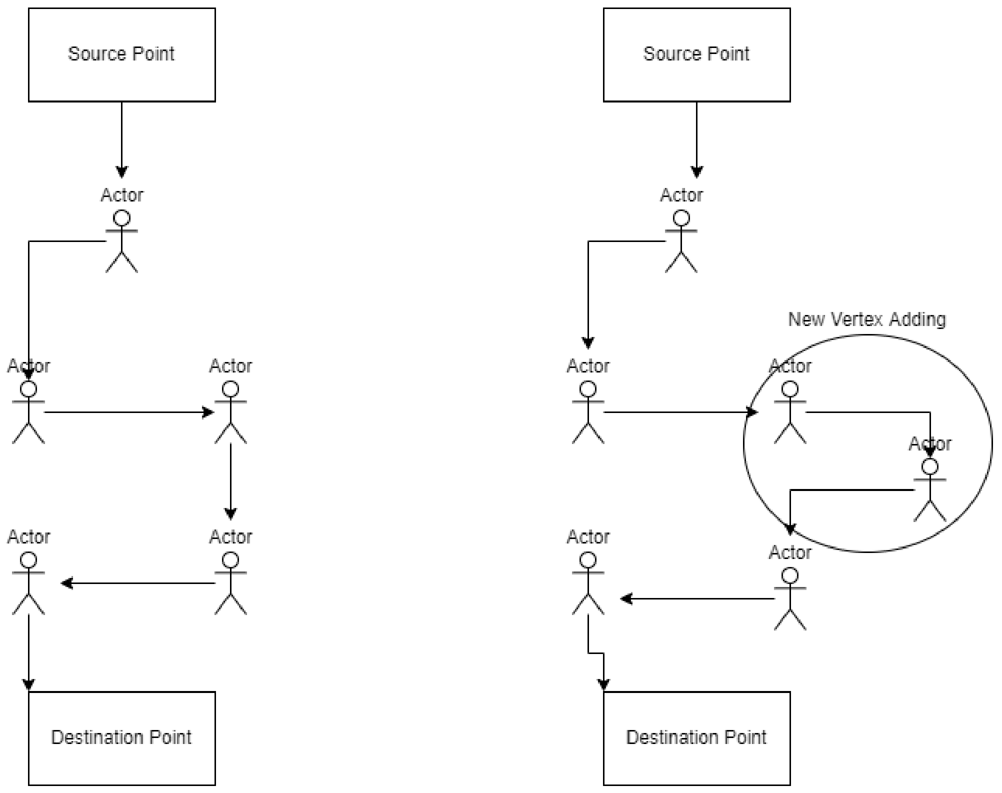

Figure 1.1: Adding new passenger in the optimized route

1.2. System Requirements

The requirements within the scope of the project are mentioned in this section. Project requirements are grouped under two separate headings as functional requirements and non-functional requirements.

1.2.1. Functional Requirements

1. The application will be used by passengers and drivers. It must support a multi-user interface. Passengers and drivers will enter the system username and password. The security will provide from Firebase Authentication API.
2. Passengers can be able to update their location information.
3. The passengers can be aware of the instant updates of the driver location.
4. The passengers and the driver can be able to see each other's location on the map.
5. Drivers will be able to add/remove passengers their passengers list, also they can using algorithm that calculate optimized route.

6. The optimization algorithm developed using meta-heuristic methods.

1.2.2. Non-Functional Requirements

1. **Usability Requirements**
   The system is user friendly in terms of usability. Users clearly perform their operations by clicking the buttons that are written in English.

2. **Performance Requirements**
   The system has very little data at first, but as the users use the system, the number of data in the system increases and here, the system needs to increase efficiency by using fast algorithms on top of these data. The system targets this in terms of efficiency.

3. **Space Requirements**
   System will hold users' data on the server side. These data are login information of the users, some driver or passenger information that users created. Since, all these data tend to be enlarged as the new users register the system, and users make use of the system. Therefore it should be scalable.

4. **Dependability Requirements**
   Application will use a map to navigate users to their desired destination. In order to use this functionality, it will depend on some third-party map and navigation data.

5. **Security Requirements**
   System must be able to store users' passwords encrypted. Posts will be edited only by its creators and administrators.

6. **Environmental Requirements**
   The interface works only visual and tactile on the computer, so the environment in which the interface is operated will not limit the users normal detection capabilities.

7. **Operational Requirements**
   Interaction of users and users is required for each method of the system to work. After all, one of the objectives of this system is to provide communication between users who use the system. It needs the operations it performs for this goal.

8. **Development Requirements**
   Since the program will be developed by single developer, the development environment must enable developer to work.

9. **Regulatory Requirements**
   System users have certain powers granted by the system. While these powers are given by the system, the system pays attention to giving it in a way that does not disturb its order and controls them.

10. **Ethical Requirements**
    Administrators will handle inappropriate behaviour, and its users. Database administrators can warn users and ban them if necessary.

11. **Accounting Requirements**
    The system must have at least one database administrator. Database administrators can approve passengers and drivers who are members of the system. Approved passengers and drivers cannot perform operations restricted by their membership (by the system).

12. **Safety/security Requirements**
    In some parts of the system, the interaction of users with each other is very much in the foreground. This also has insecurities, for example, malicious users can abuse this system. For this situation, there are measures that the system should take.

1.3. Problem Definition

The problem is to find the optimized route to the destination by collecting all intermediate passengers starting from a source. Here the source represents the location of the drive and the destination represents the destination. All intermediate passengers represent passengers who want to go to that destination. (e.g. bus driver is source point, school is destination point and students are intermediate passengers.)

The problem is similar to the traveling salesman problem, a similar technique has been applied, details will be mentioned in the Adaption of the Travelling Salesman Problem to a Shortest Path Problem section.

1.3.1. Input Domain Set, Solution Format

The designed Algorithm accepts a graph data structure as input. This graph is a full graph that contains the distances of the passengers to each other in the data structure. The algorithm adds the source and the target to the graph later, and performs its operations accordingly. If there is a previously found solution as another input, it takes it and makes initializations by referencing the previous solution. (These initializations and parameters used by ant colony optimization are mentioned in section Overview of the Ant Colony Optimization.)

The algorithm solution format is an array containing the elements of the route.

1.3.2. Decision Variables

The decision variables used in the problem are the vertexes of the graph. The vertexes selected while creating the solution are determined according to the parameters used by the algorithm.

$$\left[V_1, V_2, V_3, V_4 \dots V_n\right]$$

Figure 1.2: n is a total vertexes of the graph

1.3.3. Constraints

The constraints are as follows:

1) After visiting a city, the driver must visit the next city because he can only take a student into the vehicle once.

2) When visiting a city, the driver only needs to come from one city.

3) It is necessary to make sure that a round is fully connected, that is, there are no sub-rounds.

$$\sum_j y_{ij} = 1, \quad i = 0, 1, \dots, n-1$$
$$\sum_i y_{ij} = 1, \quad j = 0, 1, \dots, n-1$$
$$\sum_i \sum_j y_{ij} \leq |S| - 1 \quad S \subset V, 2 \leq |S| \leq n-2$$
$$y_{ij} \in \{0, 1\} \; \forall i, j \in E$$

Figure 1.3: Solution constraints formulas

1.3.4. Objective Function

While obtaining the solution, it is aimed to minimize the sum of the cost of the edge from the last selected vertex to the next vertex.

$$C(G') = \sum_{e \in E} (C(e))$$

Figure 1.4: Objective function

2. META-HEURISTIC DESIGN

2.1. Overview of the Ant Colony Optimization

Swarm intelligence is the collective behavior of systems acting together. Swarm intelligence systems consist of groups that interact with each other and with their environment. Each representative in the groups follows simple rules, and although there are no rules about how each representative should behave, interactions between representatives lead to "intelligent" behavior that the individual is not aware of.[1]

This coordination mechanism is observed in ant colonies. In many ant species, ants going to and from the food source deposit a substance called pheromone on the ground. Other ants sense the presence of the pheromone and tend to follow paths where the pheromone density is higher.[2]

As ants tend to choose the path with higher pheromone levels with a higher probability, over time, the path begins to be used by other ants. As a result, the pheromone level on the road becomes more evident and the success of the ant swarm increases. In other words, ants have found the most efficient way to reach their goals and have started to use them.

2.1.1. How the Ant Colony Optimization Algorithm Works

In order to construct the mathematical model of the ant colony system, the following two situations should be considered:

- Pheromones (Model and evaporation)
- Decision Making Phase

Mathematical Model of Pheromone Level

$$\eta_{ij} = \frac{1}{d_{ij}}$$

Figure 2.1: n is the amount of pheromone that each ant leaves on the road, and d is the distance between two cities

The staged pheromone level in the equation where the model does not move is 0. 1/d represents the distance between two points. Here 1 / the distance between two points is due to the algorithm searching for the shortest path. Thanks to this formula, more pheromone levels will be released for a shorter route.

$$\Delta \eta_{i,j} = \sum_{k=1}^{m} \left(\Delta \eta_{i,j}^{k}\right)$$

Figure 2.2: n is the amount of pheromone that each ant leaves on the road, and d is the distance between two cities

In the above equation, m represents the total number of ants. The level of pheromone secreted to the surface by each ant will evaporate over time. In the above equation, the model is established without evaporation.

What Is Evaporation?

Evaporation is the evaporation of the pheromone chemical that an ant leaves on the surface after its movement between two cities over time. By adding evaporation to the model, we can better simulate this ant-inspired algorithm.

$$\Delta \eta_{i,j} = (1 - \rho)\eta_{i,j} \sum_{k=1}^{m} \left(\Delta \eta_{i,j}\right)$$

Figure 2.3: Rho is the evaporation coefficient

In the above mathematical model (1-Rho) part represents the current pheromone level and the other part represents the pheromone level secreted by all other ants.

As seen in the model, Rho is a constant number and represents the evaporation rate. When Rho = 0, there is no evaporation. In this case, the equation gives the same result as the first non-evaporation equation we wrote. If Rho = 1, all the pheromone left by the ant will evaporate. If there is no pheromone on the ground, communication between the ants will not be possible.

If there were no evaporation, the probability of optimizing the solution would decrease. Because there is a possibility of diversification due to probability calculations. The algorithm exhibits this behavior in order not to get stuck in local optimization points, and if there were no evaporation, the probability of getting stuck in local optimizations would increase.

Calculating Probabilities

In this section, we will examine how ants' path selection probabilities are calculated.

$$P_{i,j} = \frac{(\eta_{i,j})^{\alpha}(d_{i,j})^{\beta}}{\sum\left((\eta_{i,j})^{\alpha}(d_{i,j})^{\beta}\right)}$$

Figure 2.4: n: pheromone, d: distance, alpha: pheromone effect, beta: cost effect

By reducing the ETA effect to 0 in the formula, that is, by taking beta = 1, the probabilities can only be calculated according to the pheromone level.

The roulette selection method (Roulette Wheel Selection, Fitness Proportionate Selection) is used when making a selection with probability. Roulette wheel selection is a frequently used method in genetic and evolutionary algorithms or modeling complex networks.[3] In the first place, cumulative sums are obtained from these probabilities. The selection is made on cumulative totals, not probabilities. After the cumulative totals are obtained, a random number in the range [0,1] is generated and the selection is made according to this generated number.

In line with this information, the ant colony algorithm design is shown with the pseudocode below:

Algorithm 1 Ant Colony Optimization

```
Determining ACO parameters    ▷ iteration and ant number, initial pheromone, rho, alpha, beta
while Stopping criterion not satisfied do
    while Unless all the ants reach the target do
        Generate new solution using probability distribution.
        Update local pheromone level.
    end while
    Update best solution and only update the most appropriate pheromone level.
end while
```

2.2. Adaption of the Travelling Salesman Problem to a Shortest Path Problem

2.2.1. What is Travelling Salesman Problem?

The salesperson problem can be defined as:

1. There is a salesperson;
2. This seller wants to sell his goods in n cities;
3. On the other hand, logically, this seller wants to tour these cities in the shortest possible time and with a maximum of one visit to each city.

The purpose of the problem is to be able to offer this shortest path to the seller.

To summarize, the solution to this problem is to draw a Hamiltonian Path on the relevant graph.

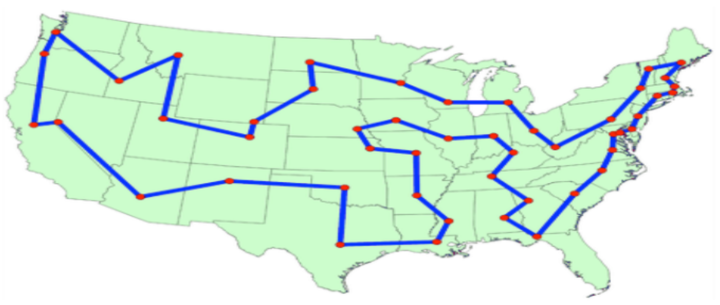

Figure 2.5: Travelling Salesman Problem Example

2.2.2. Adaptation to the Shortest Path Problem

We mentioned that ants follow each other in ant colony optimization. Ants choose cities by following pheromones, but there is a possibility for diversification. With the formula including pheromones and graph weight (Overview of the Ant Colony Optimization), the ants choose the next city.

The "dummy node" concept is visualized in the figures below.Left side of Figure 4.4 shows the source vertex as the same starting and ending vertex as in the standard TSP problem. Right side of Figure 4.4 shows the same TSP problem, but here there is a dummy node between the target and the source, and the weight of this edge must be a number very close to 0. Because if this weight is very close to zero, the probability of choosing a dummy node will converge to 100% when it reaches the target in Ant Colony Optimization. In other words, in order to hack the distance matrix of the TSP graph, a dummy node is added to the corresponding place of the matrix and its distance from the source node converges to 0.

As a result, when this technique is applied, we can adapt the TSP Problem to the Shortest Path Problem. Because an ant that reaches the target will tend to go to the Source and we have to travel all the vertexes to create a Hamiltonian Cycle. With this approach, it is inevitable that the solution starts from the source, stops at the target and returns to the source. Of course, the edge from the target to the source is not added to the solution and the solution format is target, intermediate elements, source. In this way, we reach the solution of the shortest path from a source to the destination, visiting all intermediate elements.

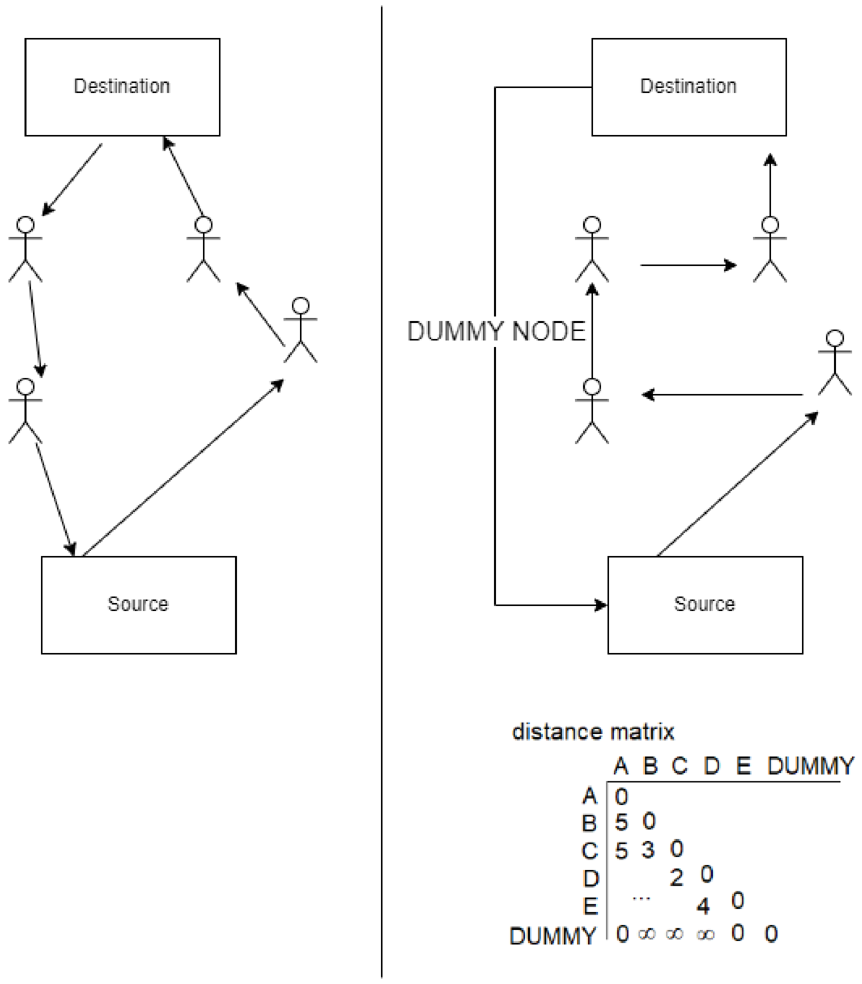

Figure 2.6: Standart TSP Solution vs Adapted TSP Solution

2.3. Dynamic Applying and Design of the Algorithm

We talked about our optimization problem, optimization method and certain adaptations, now let's talk about how this algorithm will behave dynamically.

In Ant Colony Optimization, we examined that pheromones constitute a large part of the behavior of ants (Overview of the Ant Colony Optimization). We have benefited from previously created solutions in order to ensure that the algorithm we designed does not go out of this concept and works dynamically. If the algorithm has not found a solution for a passenger group before, it works from scratch, but if the algorithm has found a solution for a passenger group before, we no longer need to create a solution for the same data group from scratch. As we mentioned, the locations of the users can be changed, users can be added or removed on the graph (The concept called adding here is that the new passenger is included in the driver group, and the concept called removing is the situation where the passenger does not want to be taken in the relevant time period).

For these cases, the dynamic behavior display mechanism of the algorithm is designed as follows:

Adding New Passenger: Leveraging the previous solution, initial pheromone levels are given slightly higher than normal pheromone levels. The ants don't bother to build the solution from scratch, it just makes intensification at the relevant place and decides which index of the newly added passenger should be added to the optimized solution we previously obtained. Thus, a solution is not built from scratch unnecessarily.

Passenger Removal: If this passenger is included in the previous solution, this passenger is removed from this range and the two passengers are joined together. Here, the combined path may increase the cost of the solution, but when the solution from which the relevant passenger was removed is reused, that is, when the initial pheromone values of the edges included in this solution are high, the ants decide what action to take instead of the removed passenger without disturbing the solution.

Passenger Position Change: The solution method applied here is the same as the solution method applied in the scope of "Passenger Removal". If the relevant passenger is present in the previous solution, it is removed and since it will already be included in the graph, it is sufficient to decide which passengers should be among while finding the solution in which the new position will be corrected.

Algorithm 2 Dynamic Ant Colony Optimization

```
Get all coming users from database
Construct graph data structure with these users
Determining ACO parameters
if Is There Previous Solution then
    if Are There Any Passenger Location Change Check then
        Remove this passenger from previous solution
    end if
    Add more pheromones to the edges contained in this solution in the pheromone graph
end if
while Stopping criterion not satisfied do
    while Unless all the ants reach the target do
        Generate new solution using probability distribution.
        Update local pheromone level.
    end while
    Update best solution and only update the most appropriate pheromone level.
end while
```

3. MOBILE APPLICATION SIDE

3.0.1. User Interface Side

The user interface consists of multiple interfaces. It is designed according to the behavior of passengers and drivers.

These behaviors are:

3.0.1.1. Passenger Side

- Passengers can see their driver on the map and can follow the instant location tracking. For this location tracking, the driver must have activated it because the information of the driver's whereabouts at unnecessary times is confidential.
- If the passenger wants to change his/her position, he/she can specify that he/she will not come in that time period.

3.0.1.2. Driver Side

- The driver can see herself/himself and her destination on the map.
- The driver can add and remove passengers to the group.
- The driver can access an algorithm-optimized route with the current arrivals list.
- It can activate the live tracking of the driver's location and stop it. At the same time, it can notify passengers of its location with only a one-time request.
- After running the algorithm, the driver sees a button that changes from the first passenger to the last passenger, and sees who to pick up in line with the information on this button, picks it up, and when he clicks this button, he sees who the next passenger is.

3.0.2. Backend / Database Side

- Firebase Authentication API is used as the Authenticate service.
- Syncing between users happens over Firebase. Passengers tracking the driver's location is an example of this. The driver writes his location to the database, and passengers receive an updated location information from this database, if available.
- Solutions previously obtained by the driver are kept by the relevant driver.Thus, even if the driver changes the device, she/he does not throw away her previous gains.

4. COMPARISON CHARTS

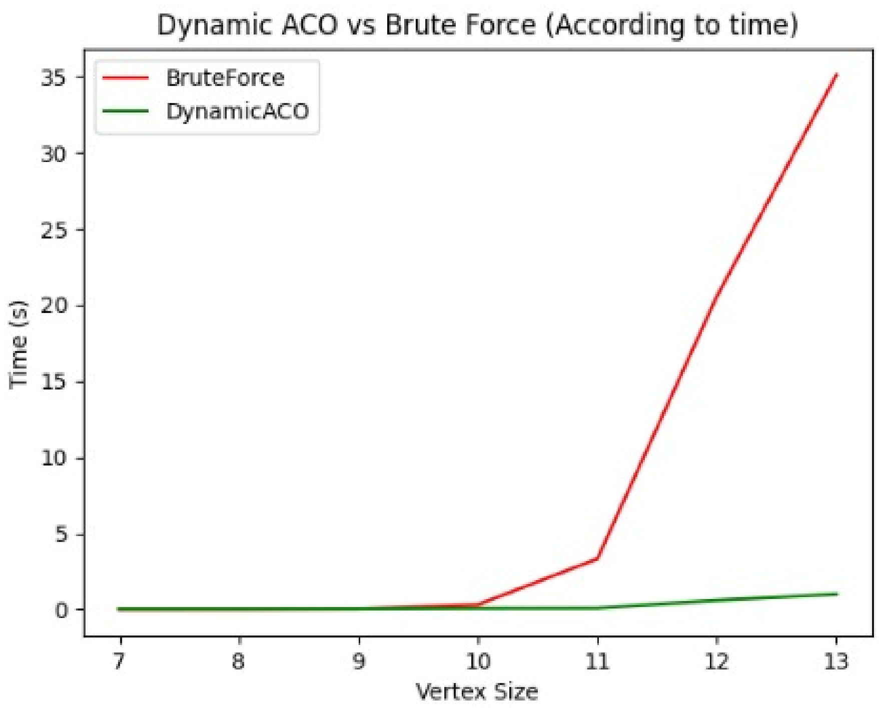

Figure 4.1: Time Compare Dynamic ACO and Brute Force

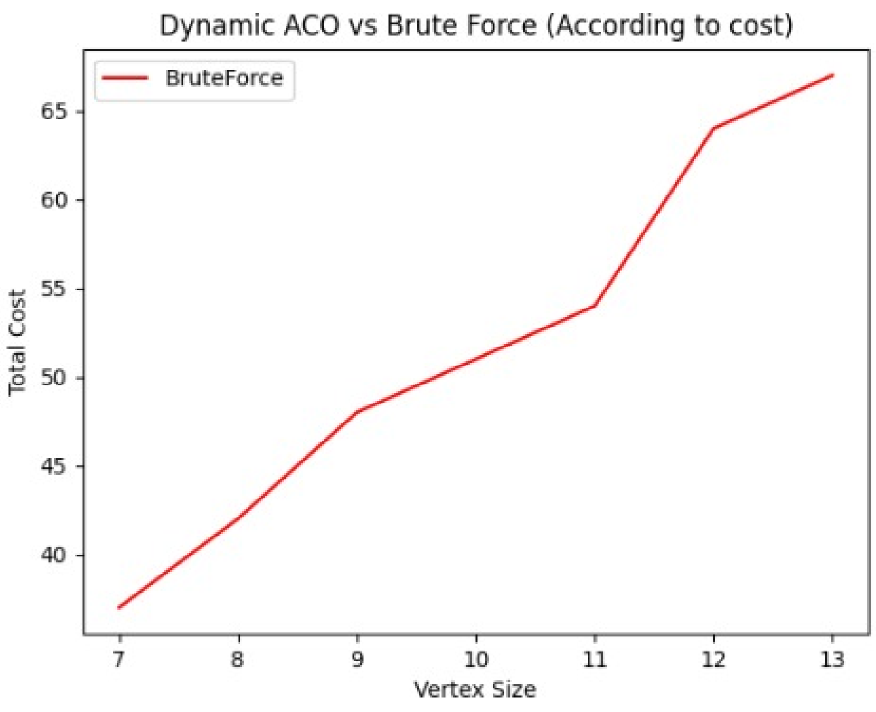

Figure 4.2: Brute Force Costs

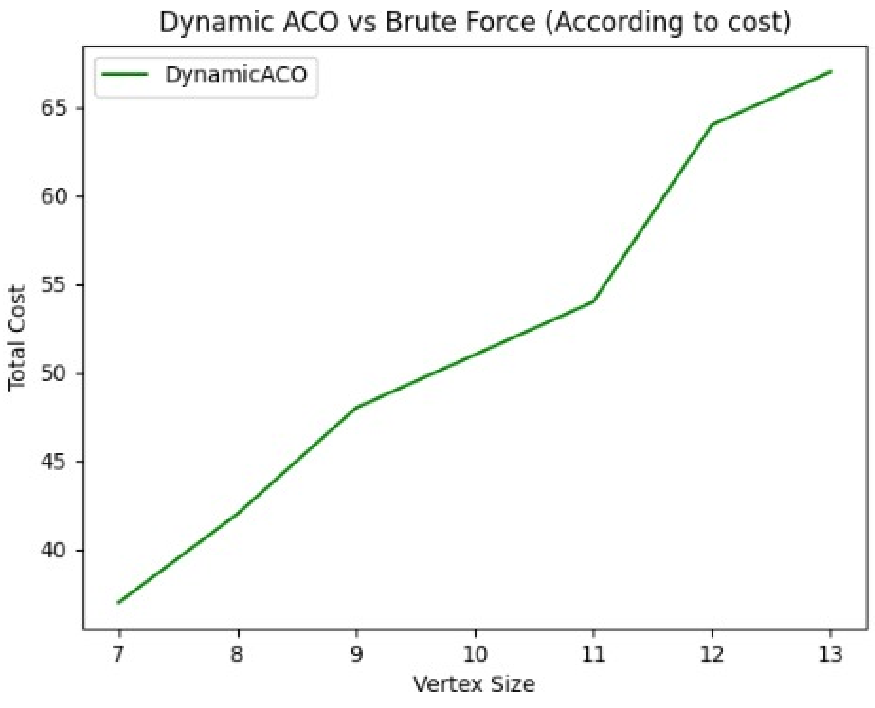

Figure 4.3: Dynamic ACO Costs

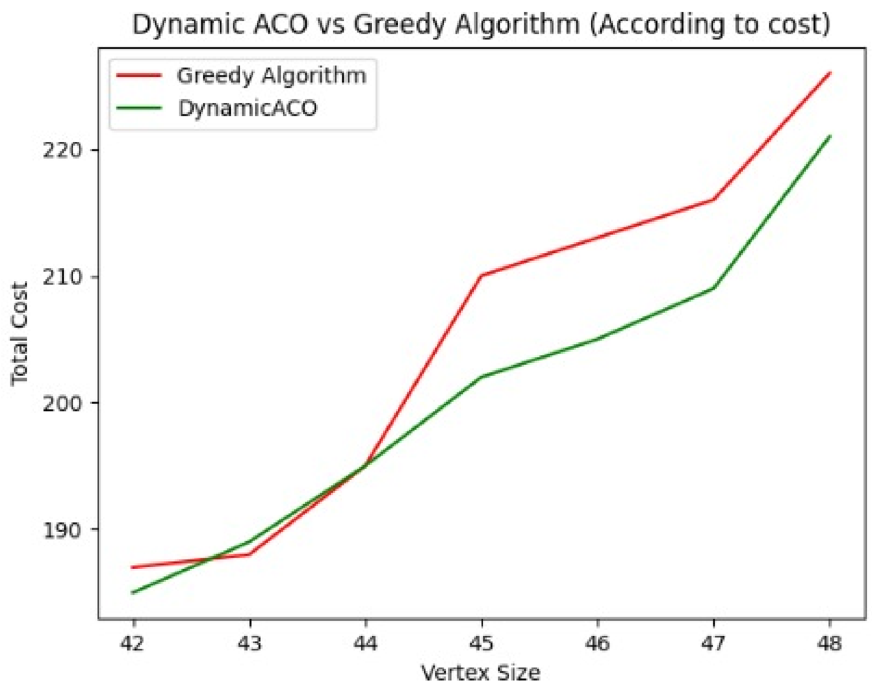

Figure 4.4: Cost Compare Dynamic ACO and Greedy Algorithm

5. CONCLUSIONS

The main motivation of this work is for a driver to collect this set of users for multiple users and take them to the destination with an optimized route, many ready-made applications (e.g. Google Maps) do this, but these ready-made applications route multiple users in the order entered. For example, it may be a wiser move to stop by C while going from A to B, but such applications route an input ordered as A, B, C in the order of A, B, C.

Another motivation is the performance of algorithms developed for problems that cannot be solved in such polynomial time on mobile devices, which have increased their power considerably today.

This study has been developed without any payment, the next version of these studies will create a better environment with a certain investment:

1. APIs that contain instant traffic information can be used.
2. APIs can also be used to simulate the result of the algorithm in the form of a directions.

Certain payments are required to use such APIs.

BIBLIOGRAPHY

[1] *Swarm Intelligence*, `https://en.wikipedia.org/wiki/Swarm_intelligence`.

[2] M. Dorigo, “Ant colony optimization,” *Scholarpedia*, vol. 2, no. 3, p. 1461, 2007.

[3] A. Lipowski and D. Lipowska, “Roulette-wheel selection via stochastic acceptance,” *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.