ABSTRACT

The project aims at designing an algorithm that would optimize the delivery of orders by suggesting the best possible rider for a particular order and also suggesting the best possible shortest route for that particular rider to complete its set of deliveries.
Chapter 1 of this report highlights the problem and challenges behind this project. The literature review of related work is presented in Chapter 2. The system's basic model is presented in Chapter 3, along with the model's functional and non-functional requirements. On the other side, Chapter 4 discusses the system's detailed design, which can assist developers in implementing the system. The porotype and the development of the system are discussed in Chapters 5 and 6. This report comes to a close with Chapter 7 including conclusion and future work that can be carried out in the domain.

DECLARATION

I hereby declare that the work has been done by myself to fulfill the requirement of the BS (Software Engineering) and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

I hereby further declare that in the event of any infringement of the provision of the Act whether knowingly or unknowingly the university shall not be liable for the same in any manner whatsoever and undertake to indemnify and keep the university indemnified against all such claims and actions.

© USAMA BIN SULTAN	[021 – 18 – 43144]
© HANZALAH AHMED KHURSHID	[021 – 18 – 44818]
© MUHAMMAD WALEED IQBAL	[021 – 18 – 44826]
© SYED MUHAMMAD HAMZA HUSSAIN	[021 – 18 – 45006]

ACKNOWLEDGEMENT

First, we thank Almighty Allah who praise us with the ability to think, work and deliver what we are assigned to do. Secondly, we must be grateful to our internal Supervisor **Dr. Atiya** and our project coordinator **Dr. Mansoor Ibrahim** who helped us in this project. We also acknowledge our teachers that throughout our studies helped us and guided us, departmental staff, university staff or other than this.

LIST OF ACRONYMS

Table 1: List of Abbreviations

GL	Group Leader
GM	Group Members
NP-Hard	Non-Deterministic Polynomial
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
VRO	Vehicle Routing Optimization
GIS	Geographic Information System
SDSS	Spatial Decision Support System
LSVRP	Large Scale Vehicle Routing Problem
LS	Local Search
HH	Hyper Heuristics
GA	Genetic Algorithm
GP	Genetic Programming
GLS	Guided Local Search
KGLS	Knowledge-Guided Local Search
SMOTE	Synthetic Minority Oversampling Technique
ERD	Entity Relationship Diagram

Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

Route optimization is the process of finding the best (shortest and cheapest) route for your vehicles in a way that we can optimize and find optimal solution. A route optimization tool can find the most efficient routes to travel based on the constraints and goals of a business. This includes taking into account the number of stops required, their locations, and the time windows for deliveries.

However, it is not that simple. If you have a business, you may know that route planning includes deciding on the best way to get customers to your business. You might have had difficulty following the proper route. There should be challenges to face if you are not likely to take them more seriously.

We are used to requesting multiple pick-ups and drop-off requests for our packages, because we want them to be delivered to any location. Also, handling multiple clients may increase your workload. Managing all these tasks one-handed can lead to serious bugs and problems. A carpenter cannot do the job of an electrician. People who are skilled at performing tasks quickly are able to do so effectively and with less time spent on the task. Likewise, vehicles are also made for specific functions. Reefers are designed to carry cold items, trucks are made for carrying heavy items, and so on. Therefore, it is important to choose the right vehicle for the job. It is important to find a driver who is communicative and able to complete the process easily.

The service time includes the travel time and a fixed stop time per customer. There is no one answer to this question since it depends on the individual's needs. You may also want to consider following through with your plan. It is difficult to determine which route and at what time a driver delivers the products. If an average time doesn't seem to be working, there might be a time crisis looming.

Accounting is essential for ensuring that all financial matters are taken care of properly. It can be difficult to gather all of the details necessary for a report. Accounting is important for keeping track of the costs associated with your business. If you neglect this part of your business, it could bring your company down. You may have assets with their own limitations. The volume, weight, and load meter are a part of the limitations of these devices. It is easy to manage the limitations of a single vehicle when doing so. However, when the number changes, the task becomes very tiring and difficult.

We often change the delivery destination at the last minute. This can be a serious problem for your logistics company. Since finding an optimized route at the last minute can be quite difficult. Moreover, there is a high chance of customer dissatisfaction if you fail to deliver your goods on time.

So, how one can overcome all these problems? Definitely all the above problems will manage with the help of a routing optimization system.

1.1. Optimization

Maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation. The function allows comparison of the different choices for determining which might be the "best".

Some Common goals of Optimization are minimal cost, maximal profit, minimal error, optimal design and optimal management.

Usually optimization is of two types:

- Mathematical Optimization.
- Heuristic Optimization

1.1.1. Mathematical Optimization

Mathematical optimization is the process of finding the best possible solution to a problem by adjusting variables in a way that makes the problem as efficient as possible. Some variation of optimization is required for all deep learning models to function, whether using supervised or unsupervised learning.

There are many optimization techniques to choose from, but all require a starting point and a goal. In order to optimize something, you need to define an objective function that will tell you what you want to achieve.

This function can produce a specific result or a probability threshold.

Inputs can be either discrete or continuous.

Limits on the size of variables are important in order to keep the data in the model accurate. Equality constraints are usually noted with an hn (x), and inequality constraints are noted with a gn (x).

Unlimited optimization problems are those that do not impose any constraints on the variables.

1.1.2. Heuristic Optimization

Heuristic design is a computational procedure that tries to find an optimal solution by iteratively improving a candidate solution according to a given measure of quality. Heuristics make little or no assumptions about the problem, being optimized and can search large areas of candidate solutions to find the optimal or near-optimal solutions at a reasonable computational cost with no guarantee of feasibility or optimality, or even in many cases the state, as in near optimality is a particularly viable solution. Heuristics implement some form of search optimization, such as evolutionary programming, evolution strategy, genetic algorithms, genetic programming, and differential evolution.

Basically every supply chain planning and scheduling problem is at its core an optimization problem. The company's solution involves determining the best way to synchronize the supply and demand across the supply chain network in order to boost customer satisfaction and bottom-line results. One common way businesses try to solve their supply chain planning and scheduling problems is by using heuristics. Simply put, a heuristic is a problem solution that uses a practical process (often referred to as a "rule of thumb" or "best practice") to produce a workable solution good enough to quickly solve a specific problem and immediate goals - but not necessarily an optimal solution.

In contrast, an optimization model uses an intelligent, automated process to create an optimal solution to a specific problem, given decision variables such as production, inventory, and shipment volumes, as well as constraints and key performance indicators (KPIs). Supply chain optimization solutions aim to improve the performance of your procurement, production, inventory, and distribution operations so that you can achieve the best possible delivery performance and overall profitability.

1.2. Route Optimization System

When you are running a logistics company, you know that your destination may not always be where you plan to take your products. Even if it is not a pickup point, the delivery point can be multiple. How can you find the shortest route that covers all of your points? How can you know about the road conditions remotely? To optimize a route, you will need a route optimization system. Delivery route optimization software can help you find the shortest distance between two or more locations. A truck routing software can also analyze current dicey situations and business limitations, like the available vehicles and drivers, traffic conditions, etc. This will help you get your goods to their destination in a timely manner. The best route optimization software generally follows three primary strategies: optimizing routes based on available data, optimizing routes based on user preferences, and optimizing routes to minimize travel time.

The three types of routing are:
Static, which is based on requirements such as quarterly or monthly plans.
Dynamic, which is based on daily developments.
Real-time dynamic routing – Developing routes based on current road and traffic conditions in real-time to help drivers on the ground.

1.3. Benefits of a Route Optimization System

The global best route optimization software market is projected to grow by 24.7% over the next four years. This is equal to $5.32 billion! This market has a high dominance in certain regions.

We must think of why do people go after the best path improvement tool? Does the best route optimization software have any benefits for travelers? Yes, I am excited about the new product. Route optimization software can also have benefits, such as saving you time. This section provides information about all of the different types of yoga. Invest less, but get a bigger return (higher return on investment)

- Reduce human error.
- Provides you with a clear insight into your overall work operations
- Save fuel costs and manage budgets efficiently
- Reduced travel time for each journey

- Can help staff complete their working hours.
- Save on redelivery costs and stockpiled first-try shipments
- Manage multiple pickups and drop off points at the same time
- Get instant alerts on route differences and incidents without incident
- Replace your operator's route optimization system to save labor costs
- Smaller distances mean low carbon emissions and pollution

Helps deliver efficient last-mile deliveries to enhance customer satisfaction

1.4. Ways to Optimize Route Planning

Use software to plan your logistics routes and improve your delivery services. Access a centralized dashboard that shows dispatch and other details of the orders in real-time. Use a logistics route optimization software to manage resources, assign tasks, and allocate orders from a single dashboard. Use a vehicle/Package route optimization platform to improve fleet performance by gathering data analytics and other reports. Field agents can easily navigate and deliver orders on time with effective vehicle route optimization. The location intelligence feature of the logistics route planning software can help you avoid congestion and optimize movement times. There are alternate routes you can take to optimize your vehicle route. Understand the factors that influence the ETA, including the time it takes to make the signals. Use logistics route optimization software to uncover hidden data points that can help improve supply chain and cost efficiency.

1.5. Uses of Route Optimization Systems

OTL LOGISTIC

OTL is a logistics company in Malaysia. The bank has over 250 branches in different parts of the world and operates over 300 vehicles. The company had difficulty managing its fleet efficiently due to an outdated fleet management system. The company had trouble repeating routes, and was unsure of where the vehicles were. After using a route optimization system, the company was able to improve its service and tracking. Moreover, they benefited hugely. Let's take a look at the various solutions.

Solution:
Optimize the route
15% savings in fuel costs
Reduced number of trucks as there are few enough trucks to reach multiple destinations.

CRATE AND BARREL:

Crate & Barrel is a well-known American-based home decor and furniture company founded in 1962. Currently, the store has over 160 stores and 7,000 associates. It also has over 80 delivery trucks. The main distribution centers of this company are in New Jersey, San Francisco, and Chicago. The coverage of the insurance policy is 300,000 per year. However, the company faced a major delivery problem. Their fleet management system was the reason for this. The problems they faced are listed below. The company sought a route optimization solution to help improve vehicle and driver utilization, inefficient vehicle tracking, failure to track data properly, and poor customer experience. The solutions they got were incredibly mesmerizing.

Solutions:
Enhanced vehicle tracking
Real-time insight into delivery statistics
Delivery time reduced by half
Real-time delivery tracking
Improved customer satisfaction

THE FOODERY:
The Foodery is a home delivery service based in Boston. This is a home delivery service. They deliver nourishing food to people. Moreover, their target audience consists of mainly working individuals who do not get the time to prepare food. Foodery was using a manual fleet management system. This created a lot of challenges for them. Let’s check them out.

Challenges:
Finding experienced drivers who were well-versed with the routes
Delivering food on time

Solutions:
Optimized routes
Up to 28% of delivery cost savings
Reduction of delivery time by half
On-time delivery

1.6. PROBLEM STATEMENT

The delivery industry is growing massively as the new newly "online" trend has started due to the Covid. Most of the businesses have now shifted to "online" platforms. This increased the demand for the packages to be delivered. Therefore, finding an optimized route for delivering the packages has become more critical and harder than ever for delivery businesses. But today almost every business uses route optimization software from small cottage industries and stores to large B2B enterprises of all industries.

In delivery system, when done manually, finding the best route is near to impossible. Even if we have few vehicles with 10 stops each there can be million routes. While planning a route is very hard having hundreds of different routes without having the right tools. Also, the biggest challenge with last mile delivery includes Delivery efficiency, margins, customer demands, delivery agility, costs, end customer interactions, missed deliveries etc. Companies also risk inflating their operational costs, often in the form of too many vehicles in their fleet and/or wasted fuel and wages due to longer than necessary routes. Delivery businesses face these types of problems every day. Effective route planning would save fuel, time, cost, employee expense, transport expense, maintenance expense. So, if we use proper route optimization mechanism we can overcome and save all above things which will help businesses to generate more revenue and let them make profitable.

An effective route optimization solution will help delivery businesses minimize wages, driving time, and fuel consumption by finding the most efficient route for the entire fleet in a matter of minutes. Thus, we will formulate an algorithm that would optimize the route for a delivery company, hence minimizing the operational costs incurred by the company and the time taken to deliver a certain number of parcels.

Now a day many companies like Food panda, UberEATS, Groceries, bottled beverages, CSA farm and others are using software that helps them in planning their route for deliveries. These companies having an optimized route can create a valuable effect on cost and time taken by a package to be delivered or collected, thus minimizing the operational cost for the delivery companies.

As we are targeting optimization in route planning. Our main target is minimizing travelling time. To minimize travelling time by using optimization of all the routes and riders we will design algorithm in such a way that when we try to plan a route our optimization system selects best rider from all available riders of particular area and assign deliveries to them.

1.7. Motivation and Challenges

It is observed by the people who order food online that the time taken by the delivery company often exceeds the expected time. The main causes behind this are often longer or incorrect routes, or sometimes due to the inexperience riders. These significant delays often bother people a lot.

The major challenging part here is the data collection. To carry out experiments on our formulated algorithm, we need some real-time data. Getting this real-time data is very difficult as no company will be willing to provide its data due to security reasons.

Our ultimate goal is to minimize the delivery time by minimizing the operational costs incurred in terms of travel time and fuel costs due to longer or ineffective routes or even due to an inexperience ride.

1.8. Project Objectives

- To collect data related to VRP i.e. Vehicle Routing Problem, that would contain the information regarding delivery and depot locations, the rider/delivery vehicle capacity and other customer related data.
- To apply different clustering techniques to divide the customers or the delivery locations according their delivery needs or their locations or the like features.
- To minimize time by formulating an algorithm that would suggest the best possible route given the locations for delivery and the capacity of the delivery entity (i.e. the delivery van or the delivery boy).
- To develop a web API that would implement the designed algorithm so that the

route data can be fetched when needed.

- To develop a demonstration app with maps implemented, that would visually display the routes suggested by the algorithm.

Chapter 2

2. BACKGROUND

Route optimization finds the best route to reduce travel cost, fuel consumption and the time taken to deliver or collect some packages. Due to its non-deterministic polynomial time (NP-Hard) complexity, it requires a lot of effort in terms of computing time to find the best route under a given set of circumstances.

A lot of work has been carried out by a number of scientists around the world but the problem still remains a critical subject in the field of optimization, especially when it comes to multidimensional optimization.

2.1. Literature Review

This section reviews the major approaches and methodologies used earlier to solve this problem.

2.1.1. Heuristic Algorithms:

There are two major categories of the heuristic algorithms being widely used for the purpose of solving route optimization i.e. Local Search and the Evolutionary Search.

2.1.1.1. Local Search

Starting from an initial solution, Local search moves from a current solution to another solution in the neighborhood. This is an iterative process in which the solution starts from a candidate node and exchanges nodes or local routes to gradually move towards a better solution. This iterative process sometimes gets trapped in the local optimum solution; therefore to overcome this many other intelligent strategies have been formulated to improve overall solution quality. These include simulated annealing [3], iterated local search [4], large neighborhood search [5], variable neighborhood search [6], and tabu search [7]. Many experiments on small and large sized VRPs (i.e. for 25 - 100 customers), using Local Search Heuristics [3, 5, 6], are found to have performed well.

2.1.1.2. Evolutionary Search

In an Evolutionary Search the whole solution space is divided into many small solutions, and then the evolutionary algorithm concurrently optimizes many solutions eventually reaching high quality solutions. It has four major steps involved:

- Representation
- Selection
- Combination
- Mutation

A few most effective evolutionary algorithms for VRP optimization are provided by Repoussis et al. [8] and Vidal et al. [9]. Mester and Bräysy [10] used the standard evolutionary optimization framework to guide exploration in the VRPTW solution space

with solution initialization and evolution. Gehring and Homberger [11] parallelized the genetic algorithm and were the first to solve VRPTW instances up to 1000 customers.
The only work on the voronoi diagram was done by Milthers [34], who split the VRPTW into sub problems and then solved them with large neighborhood search heuristics. The Voronoi diagram was found to be effective in guiding the search process. However, this study only scoped the decomposition of the problem in the solution construction stage. It can be further improved.

2.1.2. Integration of VRO and GIS

A few effective approaches for VRP have been integrated with GIS-based SDSS to cater some real world applications. Spatial data management, processing, and visualization tools are used to collect customer orders, georeference related data, activate the solving process, and display routes for VROs, as in GIS software such as ArcGIS and TransCAD. Along with Local search, Weigel and Cao [2] first introduced implementation of a tabu search heuristics approach in a GIS environment to deal with VRO for an American retailer. Mendoza et al. [1] integrated a customized routing module, which improved solution quality with commercial solutions such as SAP/R3 and ArcGIS to cope with VROs in public utilities. Experiments on a real-world case in Bogotá, Colombia with 323–601 customers verified the effectiveness of evolutionary optimization and GIS software.

Santos et al. [12], developed a web-based user-friendly SDSSs embedded with VRO to cater a trash collection task in Coimbra, Portugal. TU et al. [13] used historical traffic information to present a cloud GIS-based spatial decision support framework with variable neighborhood search heuristics for dynamic vehicle routing. All this shows the dominance of VRO in real-life transportation applications. However, they also indicated that current spatial intelligence should be improved to cater the ever increasing number of customers in the different transportation sectors.

The classical variant of the VRP has limited capacity on their vehicles. The goal of the VRP is to find the optimal routes which visit all customers at once, respecting the capacity of each vehicle, with routes starting and ending at the depot. The classical optimization task is to minimize the overall distance. For the full model, consider the work of [14].

Some mathematical formulations based methods will thoroughly search for the values of the variables to find the mathematical proven optimal. But the problem was proven to be NP-hard [15], and finding the best routes is a hard task. For instances with more than 200 customers, the problem is usually referred to as Large-Scale VRP (LSVRP) [16]–[19]. This larger size requires extra care regarding the amount of search effort given. Therefore, such a heuristic method is needed that finds good solutions in faster time instead of an optimal solution.

Heuristics are usually based on simple moves which can be searched fast and repeatedly. These are known as neighborhoods, perturbation heuristics, and local search, and they can be classified as intra-route and inter-route. Examples of such moves are the classics 2-Opt [20], Cross Exchange [21], among others.

Most effective and efficient methods for solving the VRP use a Local Search (LS) based approach [22]. LS heuristics quality is defined by the neighborhoods utilized, ranging from the classic and simple moves to more elaborated ones. In start, the LS explores new solutions by making small moves, being robust across different problems and instances, as well as being able to find high-level solutions [23].

Works from [29] and [30] applied LS in combination with other techniques, such as Genetic Algorithm and Set Partitioning, respectively, to lead the solution or set of solutions towards better solutions. However, when it comes to large scale, as shown in [19], they lose their ability to efficiently solve the problem within a few minutes, reaching to several hours. This can be explained by applying LS to these scales, the large number of neighboring solutions makes it too costly for the full search in each neighborhood. To overcome this, some LS-based methods apply some kind of heuristic pruning, reducing the number of solutions searched.

A recent case of success [31] can find solutions for up to 30000 customers within minutes of execution time, by considering move-specific pruning techniques. More methods apply limits to the search space, usually by grouping the customers or by some sort of threshold as reviewed in [17]. However, limiting the search space is not a trivial task, since if poorly done can avoid good solutions from being found.

Hyper-heuristics (HHs) reduce the level of domain knowledge to create a good heuristic. HHs have been applied to automate heuristic sequencing, planning systems, parameter control and heuristic learning methods [32], with several cases of success. When learning heuristics, factors such as the types of components to be considered, the techniques used and which parameters should this algorithm use, need to be considered. One popular approach for building HHs are the EC techniques, such as Genetic Algorithm (GA) and Genetic Programming (GP). GA has been applied for several search problems including searching for optimal heuristic sequencing, such as [24] for the bin-packing problem. GP is more used for creating a heuristic rule which builds a solution [25], rather than improving it, such as in [26] for the Dynamic Job-Shop Scheduling. More on Hyper-heuristics can be found in the review of [32] and the book of [33].

For large-scale problems (focusing on the VRP), however, HHs have not been well explored. One example of HH being applied to an LSVRP with Time Windows [18], where the large problem size is handled before the solution is fed to the HH. The approach solves the problem and search space with a column generation technique.

Guided Local Search (GLS) is a deterministic algorithm that attempts to escape the local optimum that LS algorithms inevitably fall into [27]. GLS has a set of features that can be selected to penalize the current solution, moving it away from the pitfall. This is done by using different objective functions to guide the solution, rather than changing the solution itself [31], [27].

Knowledge-Guided Local Search (KGLS) is presented in [31] where the authors apply the Guided Local Search (GLS) with a newly introduced operator and penalization functions. This was later adapted to large-scale in [19]. The KGLS operates by sequential applications of a Local Search algorithm. These phases remove the undesired edges in order to find new solutions which can potentially lead to a better overall solution. These penalization functions were based on a study by the same authors [28] in which they investigate similarities across different VRP solutions, according to several metrics. One of the most effective metrics was the width of the routes.

2.2. Background Technologies

2.2.1. Ant Colony Optimization

Both scientific and industrial fields rely heavily on optimization problems. Time table scheduling, nursing time distribution scheduling, railway scheduling, capacity planning, travelling salesman problems, vehicle routing problems, Group-shop scheduling problems, portfolio optimization, and so on are some real-life examples of optimization problems. For this reason, many optimization techniques have been created. One of them is ant colony optimization. Ant colony optimization is a probabilistic method for determining the best paths to take. The ant colony optimization approach is used in computer science and research to solve a variety of computational challenges.

Marco Dorigo introduced ant colony optimization (ACO) in his Ph.D. thesis in the 1990s. This method is based on an ant's foraging activity when looking for a path between their colony and a source of food. It was first used to solve the well-known dilemma of the travelling salesman. It is then utilized to solve a variety of difficult optimization problems. Ants are social insects that live in colonies. They reside in groups called colonies. The purpose of the ants is to find food; hence their behavior is governed by this goal. Ants crawling about their colonies while searching. An ant hops from one location to the next in search of food. It leaves a pheromone-like chemical substance on the ground as it moves. Pheromone trails are used by ants to communicate with one another. When an ant comes across some food, it carries as much as it can. It deposits pheromone on the pathways according on the quantity and quality of the food when it returns. Pheromone can be detected by ants. As a result, additional ants will be able to smell it and will follow that path. The higher the pheromone level has a higher probability of choosing that path and the more ants follow the path, the amount of pheromone will also increase on that path.

Chapter 3

3. PROJECT PLAN AND INITIAL DESIGN

3.1. Introduction

In this chapter we will discuss how much work we have done till now on research and development as according to our project plan. In this chapter we are going to discuss in details about timeline activity, Gantt chart, functional requirement, non-functional requirement and other requirement of project.

3.2. Summary of Activity Schedule

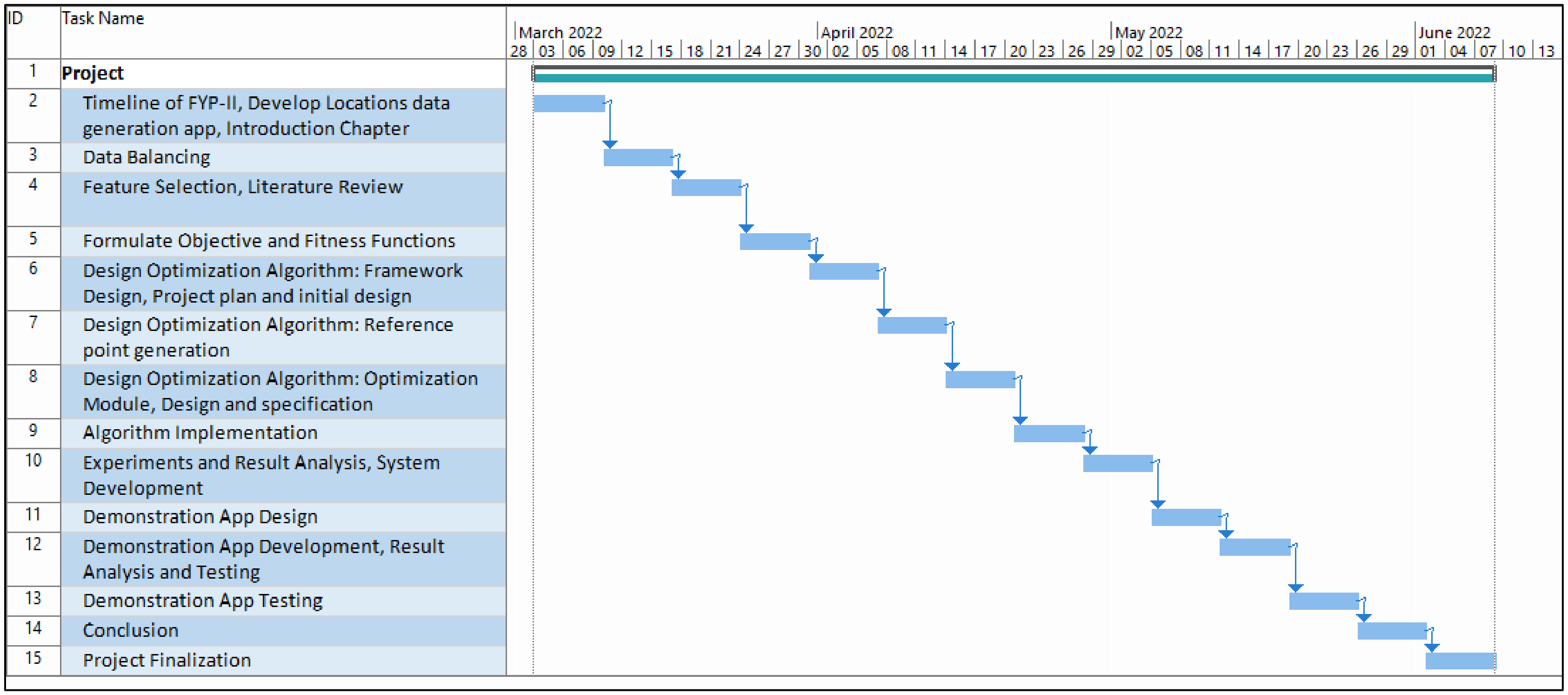
Following timeline was developed to define the sequence of tasks.

Table 2: Timeline of the Project

No.	Elapsed Time	Task	Milestones	Chapters/Deliverables
1	Week 1	Timeline of FYP-II, Develop Locations data generation app, Introduction Chapter		Timeline
2	Week 2	Data Balancing	Get equivalent data for each feature	Chapter 1
3	Week 3	Feature Selection, Literature Review	Select features to be used	
4	Week 4	Formulate Objective and Fitness Functions	Design objective and fitness functions	Objective and Fitness Functions, Chapter 2
5	Week 5	Design Optimization Algorithm: Framework Design, Project plan and initial design		
6	Week 6	Design Optimization Algorithm: Reference point generation		Chapter 3
7	Week 7	Design Optimization Algorithm: Optimization Module, Design and specification	Design Optimization Algorithm	Optimization Algorithm

8	Week 8	Algorithm Implementation	Develop an API for the optimization algorithm	Chapter 4
9	Week 9	Experiments and Result Analysis, System Development	Test the performance of the algorithm	
10	Week 10	Demonstration App Design	Design the interface of the demo app	UI Design, Chapter 5
11	Week 11	Demonstration App Development, Result Analysis and Testing	Develop the backend of the demonstration app	
12	Week 12	Demonstration App Testing	Test the performance of demo app	Demonstration App, Chapter 6
13	Week 13	Conclusion	Project conclusion and future enhancement	
14	Week 14	Project Finalization	Project Completion	Project, Chapter 7

3.2.1. Gantt Chart



ID	Task Name
1	**Project**
2	Timeline of FYP-II, Develop Locations data generation app, Introduction Chapter
3	Data Balancing
4	Feature Selection, Literature Review
5	Formulate Objective and Fitness Functions
6	Design Optimization Algorithm: Framework Design, Project plan and initial design
7	Design Optimization Algorithm: Reference point generation
8	Design Optimization Algorithm: Optimization Module, Design and specification
9	Algorithm Implementation
10	Experiments and Result Analysis, System Development
11	Demonstration App Design
12	Demonstration App Development, Result Analysis and Testing
13	Demonstration App Testing
14	Conclusion
15	Project Finalization

Figure 1: Gantt chart

3.2.2. Functional Requirements

Following are a few functional requirements of the system:

- The app should devise a route that minimizes the travel time.
- The app should be able to plan effective routes.
- The app should suggest the best rider in a particular time.

3.2.3. Non-Functional Requirements

Following are a few non-functional requirements of the system:

- The application should provide full data security.
- Accuracy in planning routes should be highly ensured
- The app should be efficient enough to run on maximum devices.
- The response time of the app should not exceed a few seconds.

3.2.4. Hardware Requirements

Following are a few Hardware Requirements

- core i3 OR more
- X86_64 CPU ARCHITECHTURE, 2ND Generation Intel Core or newer.
- 4 GB RAM or More
- 20 GB of available disk space minimum (Visual Studio IDE)

Chapter 4

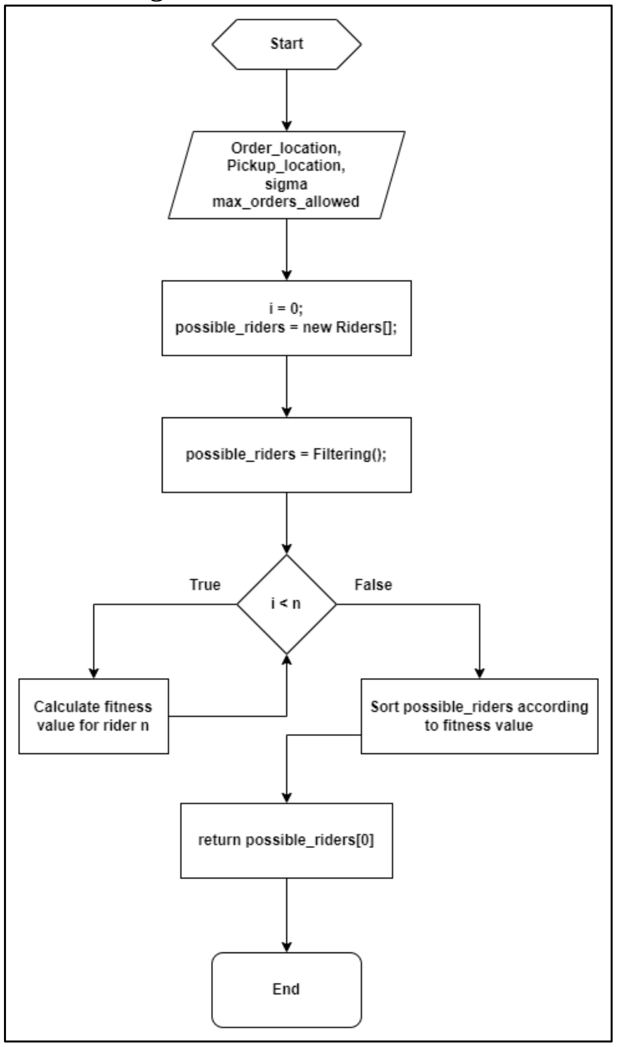
4. DESIGN AND SPECIFICATION

4.1. Introduction

Design and specification is an essential part of any implementation of project. In this chapter we will elaborate our project deeply with the help of diagram that is based according to our based research. We are going to explain our algorithm with the help of flow chart to show the flow of algorithm that will work in optimization of route.
The purpose of making the flow chart diagram to guide the direction of algorithm and to how the system will work in the implementation of all possible steps.

4.2. Algorithm Diagram

Following is a general flow of the algorithm:



Start
Order_location,
Pickup_location,
sigma
max_orders_allowed
i = 0;
possible_riders = new Riders[];
possible_riders = Filtering();
True
i < n
False
Calculate fitness value for rider n
Sort possible_riders according to fitness value
return possible_riders[0]
End

Figure 2: Flow diagram: Main Algorithm

Following is a general flow of steps of the Filtering process, which is a part of the main algorithm:

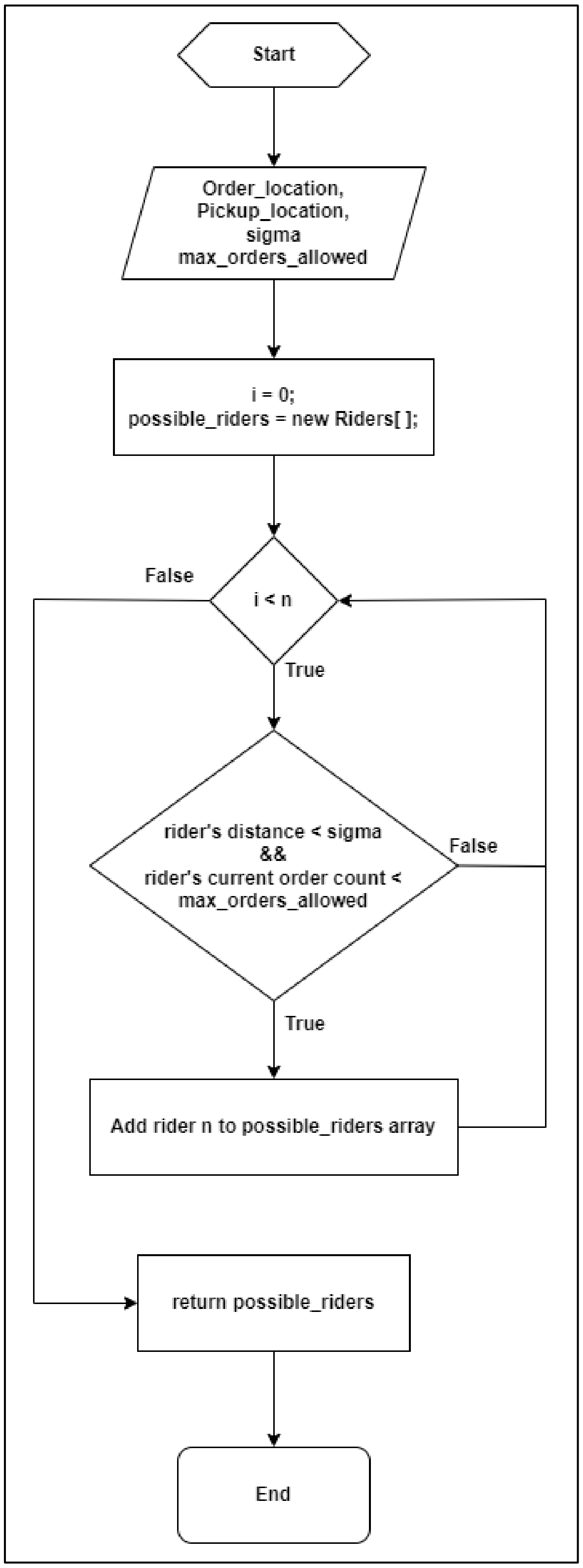


Figure 3: Flow diagram: Filtering process

This algorithm has four main steps:

- Filtering
- Fitness Value calculation
- Ranking
- Selection

Each of these steps are explained below in detail in chapter 5.

4.3. Summary

This is research based project so we don’t have any practical implementation except our research. The above flow chart of algorithm is drawn according to our research on optimization of route. We will discuss this algorithm in detail in next chapter.

Chapter 5

5. SYSTEM PROTOTYPE AND DEVELOPMENT

5.1. Introduction

In This chapter we are discussing about the flow and step of algorithm that we have designed. We are also discussing about the database queries which are used in SQL Server and we are showing screenshots of our application that shows functionalities. We have briefly provided a few clarifications about the sort of functionalities available on the system and source code.

5.2. Algorithm

Following is a basic structure of Algorithm:

- **<u>Input:</u>** Order, Order_Location, Restaurant_Location, sigma(radius) value, max_orders_allowed
- **<u>Output:</u>** Rider, Optimized_Route
- **<u>Steps:</u>**
    - Filtering(Restaurant_Location, sigma, max_orders_allowed): possible_riders
    - Foreach possible_rider:
        - Calculate the fitness_function (Rider, Order_Location).
    - Rank the rider or arange the possible_riders array on the basis of their fitness value.
    - Select the top most rider.

In the above steps, **Filtering** is the step where we select the riders which could possible minimize the total time taken to deliver a parcel. The steps of the Filtering process are as follows:

- **Filtering(Restaurant_Location, (σ)sigma, max_order_allowed):**
- **<u>Input</u>:** Restaurant_Location
- **<u>Output:</u>** Possible_riders
- **<u>Steps:</u>**
    - Select all the riders present in the same area.
    - Initialize an empty array of riders; assign to possible_riders
    - Foreach rider:
        - Calculate the distance of the rider from the restaurant location; assign to rider_Distance
        - If (ride_Distance <= sigma and rider_order_count < max_order_allowed):
            - Add rider to possible_riders
        - Return possible_riders.

Each step of the algorithm is explained below in detail.

5.2.1. Filtering

This step selects the riders from the overall population and filters out only those fewer riders which could possible minimize the delivery time.

The above function works by selecting the riders present in the radius as specified by **σ** which in our case we have fixed it to 5 km. The value of **σ** can be increased or decreased if we are not getting the enough numbers of riders in a particular area. An empty array of **possible_riders** is initialized in the beginning. At a given particular time, we would have at most ten riders to compare and find the best possible rider. While filtering out the riders, we also check if the rider is allowed to take more order or not i.e. if the rider has exceeded its max order limit or not which in our case we have decided as 5 i.e. a rider is allowed to have maximum 5 orders at a time. For the time being, we are assuming the values (such as **σ**, **max_orders** and the number of riders to search for) as per used by most of the real world companies such as Uber Eats and foodpanda, but later on, we will perform a sensitivity analysis to define these values.
If a rider fits the above criteria, it is added to the possible_riders array. The final **possible_riders** array is returned from the **Filtering** function.

5.2.2. Calculation of Fitness value for each rider

In this step, the array of riders got in step1 will be used and **for each rider**, fitness values will be calculated by calculating the total path length of the rider and then dividing it by the rider's rating.

5.2.3. Final Rider Selection

After getting the fitness values, the riders will be ranked according to their fitness values. After the ranking step, the top most rider will be selected and suggested as the best possible rider.

5.3. Backend Design

For the backend design purpose, the implementation of the algorithm is shown below in C#:

```
using RouteOptimization.App_Files;
using RouteOptimization.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace RouteOptimization
{
    public static class Optimization
    {
        // location format used for this project is [lat, long]

        // constant for changing the status of order to "ASSIGNED"
        private const string ORDER_STATUS_ASSIGNED = "ASSIGNED";

        // number of kms upto which the program should search for the riders.
        private const int SIGMA = 5;

        // maximum number of orders a rider can have.
        private const int MAX_ORDERS = 5;

        // main optimization algorithm.
        public static void Optimize(orders_processed order)
        {
            // get restaurant location from the order
            double[] pickup_location = new double[] { (double)order.hub_latitude,
(double)order.hub_longitude };

            // get the rides that could possible be the best rider for the job.
            var possible_riders = Filtering(pickup_location);

            // empty array for storing fitness values of the riders.
            var rider_fitnesses = new List<RiderFitness>();

            // for every rider...
            foreach (var item in possible_riders)
            {
                // get the shortest route length after this order.
                var shortestRouteLength = GetShortestRoute(item, order);

                // get the fitness value for this rider with this order.
                var item_fitness = GetFitnessValue((short)item.driver_rating,
shortestRouteLength.Sum(s => s.temp_distance));

                // add to the fitness values array for comparison.
                rider_fitnesses.Add(new RiderFitness() { driver_id = item.driver_id,
rider_fitness = item_fitness });
            }

            // sort the array of fitness values according to their fitness values
and select the first one.
            var rider_id = rider_fitnesses.OrderBy(o =>
o.rider_fitness).FirstOrDefault().driver_id;

            // assign the current order to the selected rider and change the order
status to "ASSIGNED".
            order.driver_id = rider_id;
            order.order_status = ORDER_STATUS_ASSIGNED;
        }

        // Filter out the rider that could possibly be the best rider.
        private static List<driver> Filtering(double[] pickup_location)
        {
            // empty list for riders to be returned
            List<driver> possible_riders = new List<driver>();

            // empty list of riders for all riders got from the database.
            List<driver> all_riders = new List<driver>();

            using (var db = new Optimization_RWEntities())
            {
                // define parameters for the stored procedure to be called.
                SqlParameter[] param = new SqlParameter[] {
                    // new SqlParameter("@lat", -30.0474147),       // hard coded
referene points
                    // new SqlParameter("@long", -51.2135086),      // for testing
only.
                    new SqlParameter("@lat", pickup_location[0]),
                    new SqlParameter("@long", pickup_location[1])
                };
                // call stored procedure that will return all the riders within an
area of 5 km in the form of a list.
```

```
                all_riders = db.Database.SqlQuery<driver>("GetRidersIn5KmRadius
@lat,@long", param).ToList();

                // filter only the riders having orders less than 5
                for (int i = 0, j = 0; i < all_riders.Count; i++)
                {
                    var rider_id = all_riders[i].driver_id;     // get rider id

                    //// get the order count for this rider
                    //var order_count = db.orders_processed.Where(w => w.driver_id
== rider_id &&
                    //!w.order_status.Equals("FINISHED")).Count();

                    // define parameters for the stored procedure to be called.
                    SqlParameter[] param2 = new SqlParameter[] { new
SqlParameter("@rider_id", rider_id) };
                    // call stored procedure that will return all the riders within
an area of 5 km in the form of a list.
                    // var order_count =
db.Database.SqlQuery<Item>("GetRidersCurrentOrderCount @rider_id", param2);

                    var data = db.Database.SqlQuery<int>(@"declare @num int exec
@num = GetRidersCurrentOrderCount @rider_id select @num", param2);

                    var order_count = data.First();

                    // if the orders of this rider is less than 5, then add to the
possible riders array
                    if (order_count < MAX_ORDERS)
                    {
                        // add current rider to the possible riders array.
                        possible_riders.Add(all_riders[i]);
                        j++;
                    }

                    // j is used to limit the number of rider that are going to be
returned.
                    if (j >= 10)
                        break;
                }
            }

            // return the possible riders.
            return possible_riders;
        }

        // get the shortest path/sequence of deliveries for a rider
        public static List<Location> GetShortestRoute(driver rider, orders_processed
new_order = null)
        {
            // empty list for all the locations to be searched.
            List<Location> locations = new List<Location>();

            // empty list to keep the sequence of locations for the shortest path.
            List<Location> path_sequence = new List<Location>();

            // index variable for path_sequence
            int i = 0;
            // current location of this rider. (format => lat, long)
            double[] ref_location = new double[2] { (double)rider.current_latitude,
(double)rider.current_longitude };

            // get all the current orders of this rider.
            // var current_orders = db.orders_processed.Where(w =>
!w.delivery_status.Equals("FINISHED")).ToList();
            List<orders_processed> current_orders =
UtilityClass.GetUnfinishedOrdersForRider(rider.driver_id);

            // if there is any new order, add it to the list of current orders
            if (new_order != null)
            {
                current_orders.Add(new_order);
            }

            // add all the pickups of the orders to the locations list
            foreach (var item in current_orders)
            {
                locations.Add(new Location()
                {
                    orderId = item.order_id,
                    locationType = 'P',
                    location = new double[2] { (double)item.hub_latitude,
(double)item.hub_longitude }
                });
            }

            // loop on until location array is empty
            while (locations.Count != 0)
            {
                // find distance from the reference point for every location.
                foreach (var item in locations)
                {
                    item.temp_distance = Coordinates.distance(ref_location[0],
item.location[0], ref_location[1], item.location[1]);
                }

                // get the location with the minimum location from the reference
point.
                var min_location = locations.OrderBy(o =>
o.temp_distance).FirstOrDefault();


                if (min_location != null)
                {
                    // if the last minimum location is a pickup location, then add
its respective delivery location.
                    if (min_location.locationType == 'P')
                    {
                        // get the respective delivery location data.
                        var order_delivery = current_orders.FirstOrDefault(f =>
f.order_id == min_location.orderId);

                        // add the delivery location data to the locations array to
be evaluated.
                        locations.Add(new Location()
                        {
                            orderId = order_delivery.order_id,
                            locationType = 'D',
                            location = new double[2] {
Convert.ToDouble(order_delivery.store_latitude),
                                Convert.ToDouble(order_delivery.hub_longitude) }
                        });
                    }

                    // change the reference location to the last minimum location.
                    ref_location = min_location.location;

                    // add the current minimum location with its serial number to
the path_sequence array.
                    min_location.index = i;
                    path_sequence.Add(min_location);

                    // remove the current minimum from locations array.
                    locations.Remove(min_location);

                    // increment index variable.
                    i++;
                }
            }

            return path_sequence;
        }

        // get the fitness value for a rider.
        private static double GetFitnessValue(short driver_rating, double
shortestRouteLength)
        {
            // we are going to minimize the following function.
            // f = total route length / rider rating
            //  -- where  total route length = current route lenght of all orders +
            //                                  route lenght of pickup +
            //                                  route length of delivery location.

            return (shortestRouteLength / driver_rating);
        }
    }
}
```

5.4. Database Queries

Following query was used to create a stored procedure that was used to retrieve riders from the database which are present in the area of 5 km.

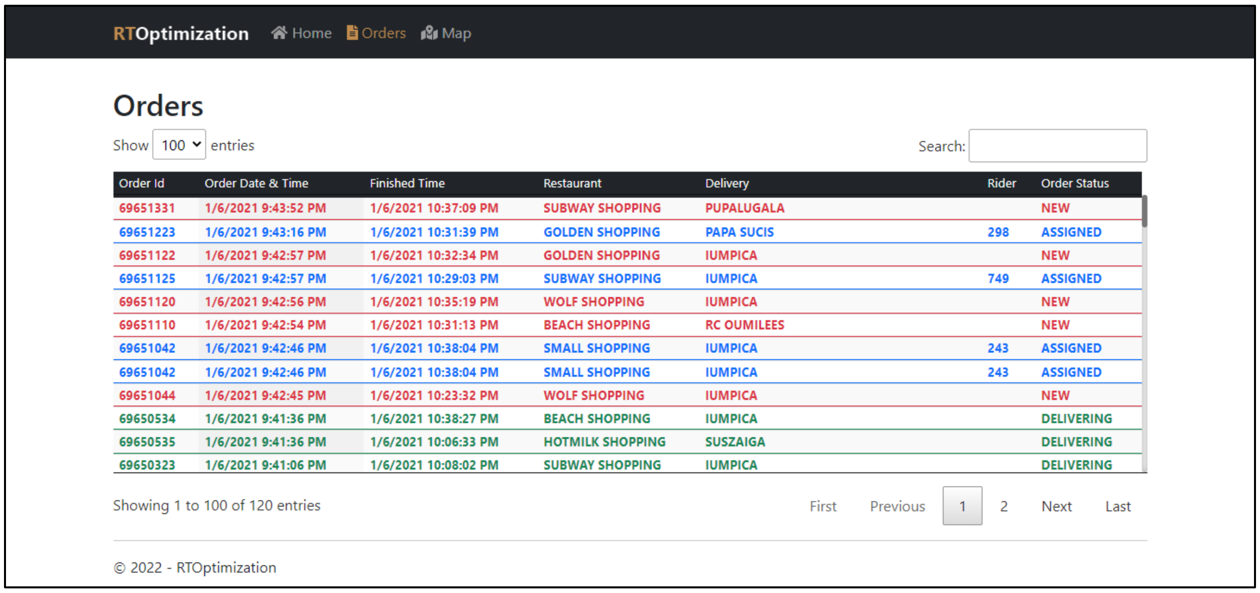
```
USE [Optimization_P2]
GO

/****** Object:  StoredProcedure [dbo].[GetRidersIn5KmRadius]    Script Date: 06/08/2022
8:19:42 pm ******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[GetRidersIn5KmRadius]
    -- Add the parameters for the stored procedure here
    @lat AS float = -30.0374145507813,
       @long AS float = -51.2035217285156
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

       DECLARE @ref_lat AS float = @lat * (PI()/180);
       DECLARE @ref_long AS float = @long * (PI()/180);

    -- Insert statements for procedure here
       SELECT * FROM drivers
       WHERE acos(sin(@ref_lat) * sin(current_latitude * (PI()/180)) +
       cos(@ref_lat) * cos(current_latitude * (PI()/180)) * cos(current_longitude *
(PI()/180) - (@ref_long))) * 6371 <= 5;
END
GO
```

Following query was used to create a stored procedure that was used to return current number of unfinished orders that a rider has at that particular time.

```
USE [Optimization_P2]
GO

/****** Object:  StoredProcedure [dbo].[GetRidersCurrentOrderCount]    Script Date:
06/08/2022 8:21:35 pm ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[GetRidersCurrentOrderCount]
@rider_id numeric(18, 0)
AS

select Count(*) from orders_processed where
driver_id = @rider_id and order_status not in ('FINISHED')

GO
```

5.5. Screenshots



Orders

Show 100 entries Search:

Order Id	Order Date & Time	Finished Time	Restaurant	Delivery	Rider	Order Status
69651331	1/6/2021 9:43:52 PM	1/6/2021 10:37:09 PM	SUBWAY SHOPPING	PUPALUGALA		NEW
69651223	1/6/2021 9:43:16 PM	1/6/2021 10:31:39 PM	GOLDEN SHOPPING	PAPA SUCIS	298	ASSIGNED
69651122	1/6/2021 9:42:57 PM	1/6/2021 10:32:34 PM	GOLDEN SHOPPING	IUMPICA		NEW
69651125	1/6/2021 9:42:57 PM	1/6/2021 10:29:03 PM	SUBWAY SHOPPING	IUMPICA	749	ASSIGNED
69651120	1/6/2021 9:42:56 PM	1/6/2021 10:35:19 PM	WOLF SHOPPING	IUMPICA		NEW
69651110	1/6/2021 9:42:54 PM	1/6/2021 10:31:13 PM	BEACH SHOPPING	RC OUMILEES		NEW
69651042	1/6/2021 9:42:46 PM	1/6/2021 10:38:04 PM	SMALL SHOPPING	IUMPICA	243	ASSIGNED
69651042	1/6/2021 9:42:46 PM	1/6/2021 10:38:04 PM	SMALL SHOPPING	IUMPICA	243	ASSIGNED
69651044	1/6/2021 9:42:45 PM	1/6/2021 10:23:32 PM	WOLF SHOPPING	IUMPICA		NEW
69650534	1/6/2021 9:41:36 PM	1/6/2021 10:38:27 PM	BEACH SHOPPING	IUMPICA		DELIVERING
69650535	1/6/2021 9:41:36 PM	1/6/2021 10:06:33 PM	HOTMILK SHOPPING	SUSZAIGA		DELIVERING
69650323	1/6/2021 9:41:06 PM	1/6/2021 10:08:02 PM	SUBWAY SHOPPING	IUMPICA		DELIVERING

Showing 1 to 100 of 120 entries

© 2022 - RTOptimization

Figure 4: Main orders screen

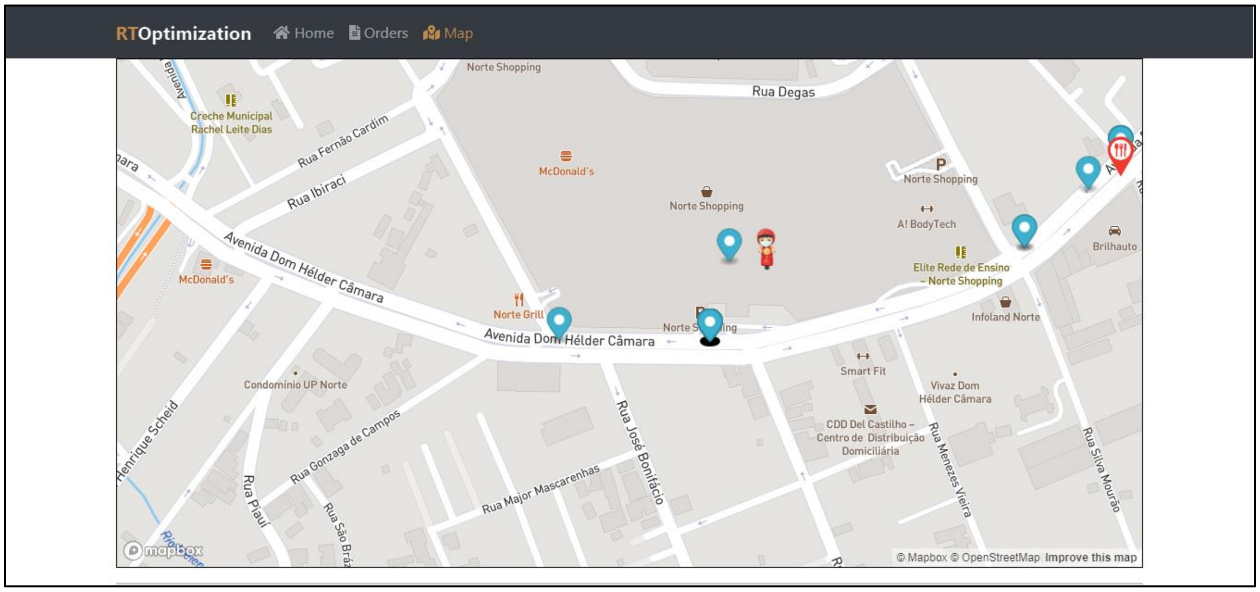


Figure 5: Maps Screen

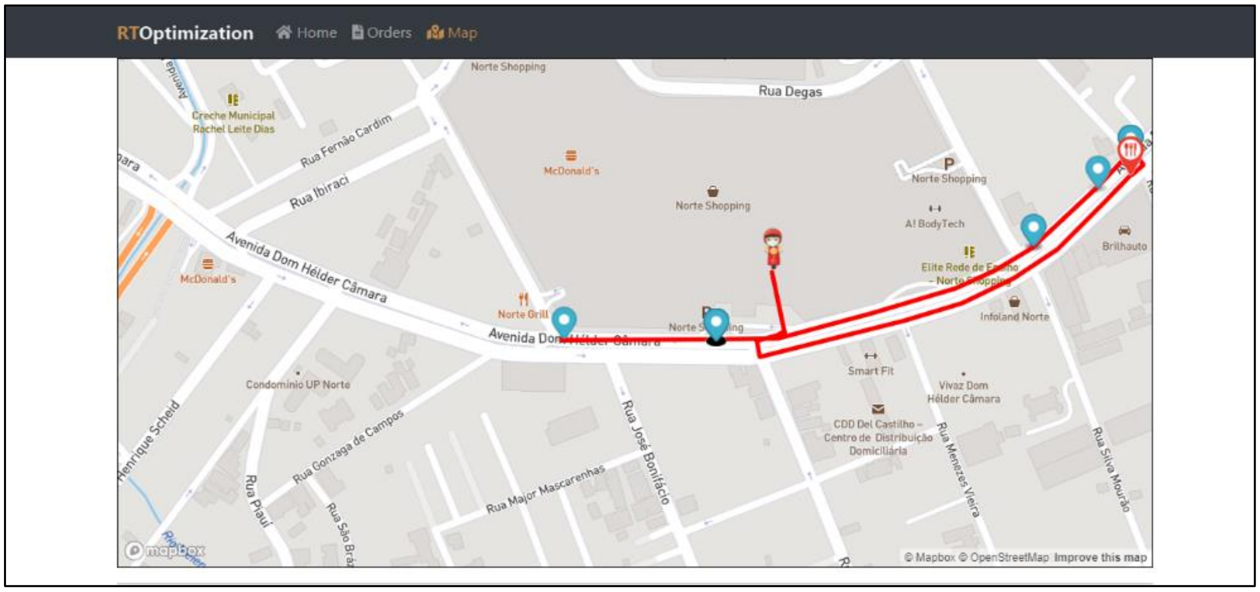


Figure 6: Routes Screen

5.6. Summary

This Chapter consist of system prototype and development in which we explained about the frontend and backend design some database queries and some screenshot of our application. We have also attach details of algorithm step by step, attached screen shot of front-end, queries of database and back-end. The database queries are also provided to show how the system retrieve the data while functioning.

Chapter 6

6. Testing

6.1. Test Cases

Table 3: Test Case 1.1

Requirement Reference	1	**Project Name**	Routing Optimization System
Test Case Id	1.1	**Test Type**	Functionality
Test Case Description	To test that the maps button and it shows maps, riders on it, deliveries and pickup.		
Test Steps	• Get on Maps button and check is it clickable and then check it shows map, riders on it, deliveries and pickups		
Expected Result	• Maps button should show map, available riders, pickups and deliveries.		
Actual Result	Maps is working.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 4: Test Case 1.2

Requirement Reference	1	**Project Name**	Routing Optimization System
Test Case Id	1.2	**Test Type**	Functionality
Test Case Description	To test that the working of dropdown button working perfectly or not.		
Test Steps	• Get on dropdown button and check is it clickable and then check its functionality.		
Expected Result	• Click on dropdown and check it show max 100 values.		
Actual Result	Table is displaying the values.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 5: Test Case 1.3

Requirement Reference	1	**Project Name**	Routing Optimization System
Test Case Id	1.3	**Test Type**	Functionality
Test Case Description	To test that the maps screen is showing the shortest route for a particular rider or not.		
Test Steps	• Get on routes screen and enter any rider's id		
Expected Result	• The maps screen should display the shortest possible route for that rider.		
Actual Result	Routes screen is displaying the routes.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 6: Test Case 7.1

Requirement Reference	7	**Project Name**	Routing Optimization System
Test Case Id	7.1	**Test Type**	Functionality
Test Case Description	To test that the home page screen working fine or not		
Test Steps	• Get on Home page and check whether working properly by clicking on logo.		
Expected Result	• Home page should reload properly		
Actual Result	Home Page is working fine.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 7: Test Case 7.2

Requirement Reference	7	**Project Name**	Routing Optimization System
Test Case Id	7.2	**Test Type**	Functionality
Test Case Description	To test that all the buttons including home, orders, maps, search bar, dropdown and paging are working perfectly or not		
Test Steps	• Get on home button and check is it clickable and then check it proceeds to home page or not.		
Expected Result	• Home page should reload.		
Actual Result	Home page is reloading.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 8: Test Case 7.3

Requirement Reference	7	**Project Name**	Routing Optimization System
Test Case Id	7.3	**Test Type**	Functionality
Test Case Description	To test that the order button proceeds to orders list and refreshing orders or not		
Test Steps	• Get on order button and check is it clickable and then check list of orders showing or not. Also check is it refreshing after 30 seconds or not.		
Expected Result	• By clicking orders there must be list of orders available. Orders should refresh after 30 seconds.		
Actual Result	Orders list is loading and refreshing every 30 seconds.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Table 9: Test Case 7.4

Requirement Reference	7	**Project Name**	Routing Optimization System
Test Case Id	7.4	**Test Type**	Functionality
Test Case Description	To test that the search bar and it's functionality working perfectly or not.		
Test Steps	• Get on search bar button and check is it clickable and then check is it searching anything available on table or not.		
Expected Result	• Search bar should search anything available in table.		
Actual Result	Search bar is fully functional.		
/			
Date Prepared	06/08/2022		
Date Run	07/08/2022		
Prepared By	M. Waleed Iqbal		
Tested By	Hanzalah Ahmed Khursheed		

Chapter 7

7. Conclusion

7.1. Introduction

This chapter will summarize all of the work completed during the final year project, as well as the challenges, limitations, and future work for this project. In this chapter all the major and minor work will be discussed. This chapter also includes the limitation of research in order to help the user to understand the basics of algorithms. The Future work section will provide an overview of algorithm's enhancements that can be made in future.

This route optimization system algorithm has the basic features of optimization that is perfect for all industries that have transportation model which includes a pickup and a delivery location. In order to optimize time, we have designed an algorithm that is more convenient to the companies with a central hub to optimize time and for the better customer experience. We aim to make the algorithm more efficient and accurate with future enhancements.

7.2. System Limitations and Challenges

The problem that we currently worked on addresses only the orders received till a particular time and suggest the possible riders and the shortest path for that rider. The shortest path suggested by the algorithm does not address the issues such as delay time, traffic congestion, road conditions and other real-time factors that may delay the actual delivery time.

7.3. Future Work

In future, further work could be carried out in this domain to make the solutions more effective. An improved solution could be the one that could predict the number of orders for each restaurant in the future using some ML prediction techniques. This way, if we know that a restaurant is going to get crowded with orders in the next hour, we can assign the current orders to more far away riders so that we would have more riders in the next hour closer to that particular restaurant.

Another room of improvement could be to bring into consideration the real-time road conditions such as broken or crowded roads, traffic jams, road closures, construction etc. If we have the real-time road conditions, we would be able to calculate the time taken by each possible route and thus, would be able to suggest shortest routes in terms of time i.e. sometime longer but cleaner route may get the job done in less time than the shorter but bad routes.

7.4. Conclusion

We can conclude that an optimization algorithm plays vital role for the delivery or transport industry. An effective route optimization algorithm can not only increases efficiency of a company or an organization but can also prove to be fruitful in term of monitory benefits. The algorithm designed by us for the purpose of this project address static orders only i.e. an order is received and optimized. This approach also solves the problem to some extent. But addressing the dynamic orders could help find much better solutions. By dynamic orders we mean the orders received at the moment when the current orders are being processed i.e. when the current orders are being processed, any order received during the time of processing, should also be considered for processing. This would help suggest more optimal solutions. Another great improvement could be to predict future orders and schedule the current orders accordingly or taking into account the real-time road conditions or other such factors that may affect the overall job completion time. In the end, this is a topic with a never ending room of improvement.

References

1. Mendoza, J.E.; Medaglia, A.L.; Velasco, N. An evolutionary-based decision support system for vehicle routing: The case of a public utility. Decis. Support. Syst. 2009, 46, 730–742.
2. Weigel, D.; Cao, B. Applying GIS and OR techniques to solve sears technician-dispatching and home-delivery problems. Interfaces 1999, 29, 112–130.
3. Baños, R.; Ortega, J.; Gil, C.; Fernández, A.; de Toro, F. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. Expert. Syst. Appl. 2013, 40, 1696–1707.
4. Alabas-Uslu, C.; Dengiz, B. A self-adaptive local search algorithm for the classical vehicle routing problem. Expert. Syst. Appl. 2011, 38, 8990–8998.
5. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 2006, 40, 455–472.
6. Polacek, M.; Benkner, S.; Doerner, K.F.; Hartl, R.F. A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. BuR-Business Res. 2008, 1, 207–218.
7. Cordeau, J.F.; Laporte, G.; Mercier, A. A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. 2001, 52, 928–936.
8. Repoussis, P.P.; Tarantilis, C.D.; Ioannou, G. Arc-guided evolutionary algorithm for the vehicle routing problem with time windows. IEEE. Trans. Evol. Comput. 2009, 13, 624–647.
9. Vidal, T.; Crainic, T.G.; Gendreau, M.; Prins, C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. 2012, 40, 475–489.
10. Mester, D.; Bräysy, O. Active guided evolution strategies for large-scale vehicle routing problems with time windows. Comput. Oper. Res. 2005, 32, 1593–1614.
11. Gehring, H.; Homberger, J. Parallelization of a two-phase metaheuristic for routing problems with time windows. J. Heuristics. 2002, 8, 251–276.
12. Santos, L.; Coutinho-Rodrigues, J.; Antunes, C.H. A web spatial decision support system for vehicle routing using google maps. Deci. Support. Syst. 2011, 51, 1–9.
13. Tu, W.; Li, Q.; Chang, X.; Yue, Y.; Zhu, J. A Spatio-temporal decision support framework for large scale logistics distribution in Metropolitan area. In Advances in Spatial Data Handling and Analysis; Harvey, F., Leung, Y., Eds.; Springer: Berlin, Germany, 2015; pp. 193–206.
14. G. Laporte, "Fifty years of vehicle routing," Transportation Science, vol. 43, no. 4, p. 408–416, nov 2009.
15. J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems." Networks, vol. 11, no. 2, p. 221–227, 1981.
16. M. Gendreau and C. D. Tarantilis, Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Cirrelt Montreal, 2010.

17. M. Huang and X. Hu, “Large scale vehicle routing problem: An overview of algorithms and an intelligent procedure,” International Journal of Innovative Computing, Information and Control, vol. 8, no. 8, p. 5809–5819, 2012.
18. N. R. Sabar, X. J. Zhang, and A. Song, “A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows,” in CEC. IEEE, 2015, p. 830–837.
19. F. Arnold, M. Gendreau, and K. S¨orensen, “Efficiently solving very large-scale routing problems.” Computers & OR, vol. 107, p. 32–42, 2019.
20. M. M. Flood, “The traveling-salesman problem,” Operations Research, vol. 4, no. 1, p. 61–5, feb 1956.
21. ´ Eric Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin, “A tabu search heuristic for the vehicle routing problem with soft time windows,” Transportation Science, vol. 31, no. 2, p. 170–186, may 1997.
22. T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, “Heuristics for multiattribute vehicle routing problems: A survey and synthesis,” European Journal of Operational Research, vol. 231, no. 1, p. 1–21, nov 2013.
23. E. Aarts and J. K. Lenstra, Local Search in Combinatorial Optimization. Princeton University Press, 2003.
24. N. Pillay, “A study of evolutionary algorithm selection hyperheuristics for the one-dimensional bin-packing problem.” South African Computer Journal, vol. 48, p. 31–40, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/saj/saj48.html#Pillay12
25. H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, Y. Sun, B. Tran, B. Xue, and M. Zhang, “A survey on evolutionary machine learning,” J. Roy. Soc. New Zeal., vol. 49, no. 2, p. 205–228, 2019.
26. F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, “Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling,” IEEE Transactions on Cybernetics, 2020. Doi: 10.1109/TCYB.2020.3024849.
27. C. Voudouris, E. P. Tsang, and A. Alsheddy, “Guided local search,” in Handbook of Metaheuristics. Springer US, 2010, p. 321–361.
28. F. Arnold and K. S¨orensen, “What makes a vrp solution good? The generation of problem-specific knowledge for heuristics,” Computers & Operations Research, vol. 106, p. 280–288, 2019.
29. T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, “A unified solution framework for multi-attribute vehicle routing problems,” European Journal of Operational Research, vol. 234, no. 3, p. 658–673, may 2014.
30. A. Subramanian, E. Uchoa, and L. S. Ochi, “A hybrid algorithm for a class of vehicle routing problems,” Computers & Operations Research, vol. 40, no. 10, p. 2519–2531, oct 2013.
31. F. Arnold and K. S¨orensen, “Knowledge-guided local search for the vehicle routing problem,” Computers & Operations Research, vol. 105, p. 32–46, may 2019.

32. E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. O¨ zcan, and R. Qu, “Hyper-heuristics: A survey of the state of the art,” Journal of the Operational Research Society, vol. 64, no. 12, p. 1695–1724, 2013.
33. N. Pillay and R. Qu, “Theoretical aspect—a formal definition,” Hyper-Heuristics: Theory and Applications, p. 37–48, 2018.
34. Milthers, N.P.M. Solving VRP Using Voronoi Diagrams and Adaptive Large Neighborhood Search. Master’s thesis, University of Copenhagen, Copenhagen, Denmark, 2009.