LB/DON/106/2016

IT 01/130

Enhanced route searching and suggesting method for carpooling system

K.B.V.D. Lakshan Dharmasiri

Index Number: 139161R

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Honours Degree of Bachelor of Science in Information Technology.

April 2016

004"16"
004 (043)

TH 3165
+ 1 DVD-ROM
(TH 3160 - TH3180)

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

K.B.V.D. Lakshan Dharmasiri ..... [illegible] ....

Supervised by

Mr. Chaman Wijesiriwardana

UOM Verified Signature

Date:

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Mr. Chaman Wijesiriwardana, for the continuous support of my MSc study and research, for his patience, motivation, enthusiasm, and immense knowledge. I have been amazingly fortunate to have a supervisor who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. Besides my supervisor, I would also like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my father and mother for the encouragement and guidance throughout my MSc.

Abstract

Carpooling is considered as a solution to address the current problems in travelling, in a context of high fuel cost, vehicular traffic congestion, which leads to air pollution, health hazard, vehicle parking problems and wasting of valuable time resulted in high opportunity cost. Carpooling is the process of utilizing a pool of passengers into matching vehicles, when there are freely available vacant seats in passenger vehicles, which could be made use of by needy passengers, minimizing the per head cost of travelling in a macroeconomic sense. The matching of drivers and passengers by maintaining a set of data, can be implemented by developing information and communication technology based applications. The drivers can post the availability of the vehicles and the number of vacant seats in their vehicles in this application and the passengers can search and contact the relevant drivers on a preferred route. By matching these needs of two parties through the proposed system, it is expected to address the above mentioned issues with a satisfactory contribution towards the optimization process of cost of travelling in a macroeconomic context. There are many carpooling systems in application in different countries and also many research and development works were done in this field to improve the quality of the carpooling systems. However, developing a system to assist the passengers as well as the drivers, searching a best and ideal route(s) to reach their destinations, with their preferred parameters is a complex and challenging task.

This research presents an improved searching method, which makes sure that the resulting routes are most suitable and identical routes, considering the route information entered by the passenger. This system will also generate and suggest alternative routes to the driver, after analyzing the lift request data entered by the passengers, for his consideration and then him to decide on the preferred route.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Prolegomena

Ever increasing population and growth of world economies lead to the increased usage of passenger vehicles. This has become a fact even in the developing countries. Increase in vehicular passenger transport resulted in increased energy cost, traffic congestions, unprecedented level of environmental pollution, thereby compelling the people and the Governments to focus their attention towards the optimal management of the ever increasing travel demand including development of alternative transport modes. In the international journal published by S.A. Shaheen and A. P. Cohen [1] and the survey undertaken by M. M. Galizzi [2], in Europe region, explains about the impact of ever increasing travel demand and resultant increase in the passenger vehicular traffic. One of the most significant fact identified in their studies was that, most vehicles on the road are not fully occupied, in most of the time and only the driver travels in the vehicle [2]. Due to this fact, more passenger vehicles are required to meet the travel demand, which could have otherwise been reduced significantly if occupancy rate is increased in passenger vehicles. This also highlights the fact that underutilized passenger vehicle capacity tends to waste the resources in a big way in the macroeconomic context. Therefore, studies undertaken in the field of transport management, have focused attention towards development of travel management methods to share the vacant capacities of passenger vehicles, with the passengers having identical destinations along the same routes. The sharing of underutilized capacities of passenger transport vehicles is widely known as "Carpooling" among the professionals, it is now believed by the professionals that this method of management can contribute significantly in addressing the above mentioned social, economic and environmental issues arising from the ever increasing vehicular traffic. Many researchers emphasized and recommended [3]–[7] Carpooling approaches can help significantly in addressing the issues already stated while yielding benefits to the passengers as well as the drivers.

The carpooling systems already developed and in use have been designed for specific purposes, such as profit making, increasing social benefits, reducing travelling time, minimize CO2 emissions, ease vehicular traffic congesticns etc. The carpooling systems currently in use such as LiftShare (https://liftshare.com/uk), BlaBlaCar (http://www.blablacar.in/), EuroLift (http://www.eurolift.com/), FreeWheelers (http://www.freewheelers.com/) etc. are designed to provide public carpooling services with different features and facilities, which help the drivers and the passengers to meet each other through the carpooling portals, enabling the drivers to find suitable passengers and vice versa.

Existing carpooling systems provide facility to the passengers to search available routes in the system through analyzing the input data fed in to the system such as origin and destination points; however, the routes suggested by the filtering process of the system, may cause disappointment as the suggested route could be totally deviated from the expected route and such route may be an elongated one. Such disappointments can occur due to the fact that different systems use different approaches, techniques and technologies to search matching routes. Many organizations, researchers and analysts have addressed such issues in carpooling systems [8]–[10], to improve the quality and the performance. Study of such systems revealed that the searching mechanism inbuilt in the system plays a significant role, in the process of route evaluation.

Another drawback found in the current systems is that, the driver has to wait until a response is registered by prospective passengers, who accept the route entered by the driver. It will be advantageous for the drivers, if the system can suggest alternative routes, by analyzing the lift requests made by passengers. This facility will definitely enhance the quality and the efficiency of the system for the advantage of the system users.

1.2. Proposed searching and suggesting method

It has already been observed that the searching and suggesting facility in a system has to play a dominant role in the carpooling systems. As such, appropriately designed searching and suggesting method in a system, can provide solutions to the drawbacks found in the current carpooling systems. The issues relating to the driver waiting for the prospective passenger response and the alternative route being suggested to be identical to the path preferred by the driver can be overcome, by improving the route suggesting mechanisms. The route suggestion method requires the system to maintain a list of expected routes from passengers in the system. The expected route details could be collected, through a facility to create alerts in the application, which notify the passengers when there are matching lifts available. By maintaining the requested route details in a database, the system can search and suggest matching alternate paths to the drivers. Accordingly, the driver can change the path to alternate routes suggested by the system, keeping the origin and the destination locations unchanged catering for the matching passengers found in the system.

The development of the carpooling system will use PHP server side scripting language and the MySQL tool for the database. The MySQL Spatial database tables were used to store the geo-coordinates and related route data; and the inbuilt Spatial functions were used to perform the Spatial related calculations. The Google Maps JavaScript API is used to show the routes and locations in the maps and the RouteBoxer JavaScript utility library used to perform boundary calculations along the routes created in the map. The JQuery JavaScript library used to improve the interactivity of the system.

When the system performs the searching, it uses the Haversine formula to calculate the great circle distance to the geo-coordinates available in the Spatial tables to filter the nearest routes. In order to improve the search results, they will be further filtered using the bounding boxes which are generated from the RouteBoxer utility library and the routes will be ranked and the most suitable routes will be shown to the passenger.

1.3. Summary

The remainder of this thesis has been structured to review the literature on carpooling systems, and followed by a comparative analysis of advantages and disadvantages of existing systems. The next section will describe the constraints and limitations of existing systems in order to identify the areas in the systems which need improvements. Then the approach of the system implementation will be discussed with a focus on analysis, design and implementation. Finally, the evaluation of the developed system will be presented and followed by conclusions with summarization of the key constraints and limitations of proposed algorithm and provide some suggestions for future research directions.

Chapter 2

Developments in route searching solutions

2.1. Introduction

Initially, the mapping models were mostly used for military and research purposes, however with the development of the information and communication technology, the existing mapping APIs were technologically advanced and new APIs were introduced with many user friendly and powerful features, which has become popular among the users as well as the professionals. Using these powerful mapping APIs, many useful systems and tools were developed to improve the performance of day to day map related tasks in a user friendly manner. The function of efficient and optimal route searching is necessary for these systems to provide value added services and therefore new algorithms, tools and techniques have been developed. In the next section of this chapter, the current approaches and solutions to solve this problem is reviewed.

2.2. Similar models

A carpooling platform called "Lift4U" developed by S. Di. Martino et al. [11], proposed a passenger searching mechanism, which uses cloud based architecture to improve the searching performance and to cater for wide range of users such as desktop, mobile and in-car telematics. Using this approach, lift offers and requests can be matched on-the-fly using the data gathered from the users' social network and mobile device GPS. The parameters such as starting point, destination point and maximum possible detour distance to pick up other passengers etc. are used for filtering matching passengers.

In this approach, developers have mainly focused on optimization of the total distance of the trip, while offering the facility to the drivers to pick up the matching passengers. The interesting feature in this model is its integration with the social network to search for the matching passengers. This system too has disadvantages; it does not focus on the passengers who are within the route, but it rather considers only the nearby passengers at the trip starting point. The other disadvantage is, its integration with the social network which allows third party applications risking the users' privacy. These disadvantages have been given due consideration in this paper.

Another matching model was developed for carpooling systems by J. Xia et al. [12], it used both optimal and heuristic approaches to solve the shortest path-finding problem. This modal too identified the constraint that, most of the existing models do not have the technical ability to match more than three passengers. This modal analyzes the optimal route searching results from both enumeration (and brute force algorithm), and linear programming; however, it has practically proved that, both approaches are not efficient for large data sets. Heuristic procedures proposed to overcome this issue by using Simulated Annealing heuristic procedure and Tabu Search procedure. In this approach, they have used Spatial and text based data for passenger matching. This application performs Spatial filtering of the passengers within the defined range of distance to the road. (See Figure 2.1)

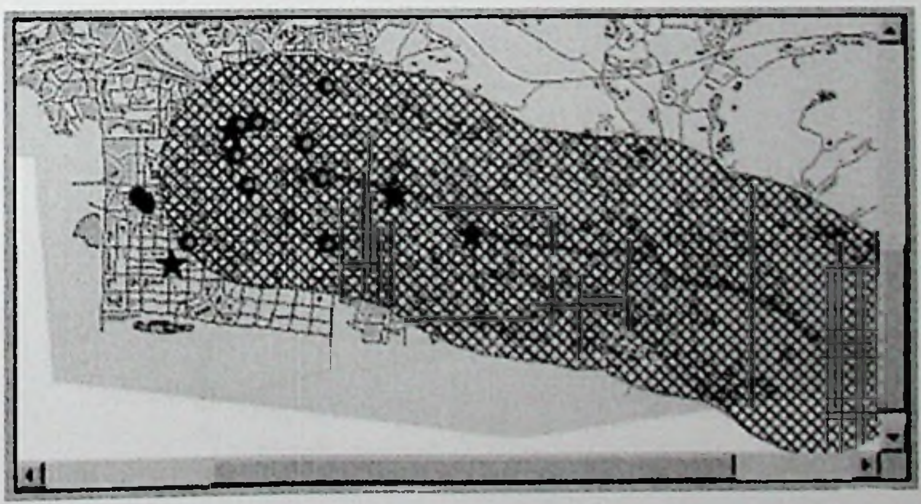


Figure 2.1: Area selected for searching

In summary, the researchers highlighted three points; first point is that the Tabu search procedure requires more execution time than the simulated annealing procedure. Secondly, if the matching passengers are less than four, there is no significant difference between two results. Thirdly, if the problem is large in scale, both procedures have similar execution time, but, the Tabu search procedure provides shorter routes.

The research undertaken by C. Mulders and B. Lambeau [13], address a similar passenger searching problem; their research too dedicated to search and group the passengers into carpools. The case study undertaken in this research was that, the colleagues of businesses/schools/works in different locations in a particular area, were grouped according to common destinations. The driver of the group has to pick up passengers in each group; therefore, the minimum detour distance is calculated and maximum of three passengers were grouped together. Figure 2.2 shows how the route calculations are done for driver (u) comparing each passenger waiting at each location closer to the starting point of the route.

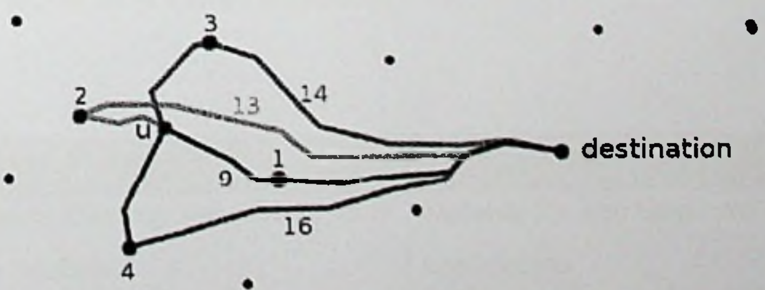


Figure 2.2: Identifying nearby passengers

In this paper, three different solutions for carpooling issues were analyzed and tested against each other. The Table 2.1 shows the summary of the three methods analyzed.

#	Method analyzed	Summary of the analysis
1	Classic filtering and sorting from the database	Works well for low number of passengers. Less efficient than other two with regards to finding a route for a big group of users.
2	Clustering algorithm to group the passengers by car	Very efficient in terms of computational time. Unable to improve the algorithm by adding features and lacks a proper optimization.
3	Constraint programming solver on a vehicular routing based model	Slowest solution in terms of computational time. Gives best results and features can be added easily.

Table 2.1: Summary of methods

According to the preceding analysis, use of these methods depends on the application. The Table 2.2 shows the possible applications.

#	Searching method	Type of applications
1	Classic filtering and sorting from the database	Suitable for web based and mobile based applications
2	Clustering algorithm to group the passengers by car	Compute the results quickly for set of users
3	Constraint programming solver on a vehicular routing based model	The applications where the time is not a main concern.

Table 2.2: Possible method of applications

In summary, this system is developed to group the passengers whose destination is same as the driver and the passengers must be in nearby starting location of the route. In this paper, it is not clearly mentioned about the best method to match and search passengers; but discuss the applicability of the given three methods. Combination of the useful features of these methods is suggested to overcome the issues in each method such as combining the clustering method with the vehicular routing method. The main disadvantage of the clustering algorithm is; that it uses the binary comparison which is not a better way to perform lager sets of data.

The Intelligent Carpooling System (ICS) developed by Shete et al. [14] provides carpool services through smart handheld devices which can be used anytime anywhere. The primary objective of this application is to match maximum number of passengers with drivers, and the secondary objective is to minimize the average distance travelled by drivers, the average waiting time of passengers, and the average travel distance of passengers. Using this mobile application, driver can offer carpool rides and passenger can send carpool requests. In this study they applied the Genetic-based Carpool Route and Matching Algorithm (GCRMA) to resolve this multi objective optimization problem, called the Carpool Service Problem (CSP). They proved that, this algorithm provides best results for matching objectives of CSP than other algorithms and GCRMA has less degree of computational complexity in responding the matched results within a reasonable period of time.

This system contains two primary modules, a mobile client (MC) module and a carpool services (CS) module. The web Hypertext Transfer Protocol (HTTP) from the mobile communication network used to communicate between the MC module and the CS module. The MC module is a mobile application which is built on Android mobile operating system and the CS module provides the Representational state transfer (RESTful) web service application interface for global implementation of the ICS on cross platform devices. The free Google Cloud Messaging (GCM) service used to send the data from server to users' mobile application and vice versa. Proposed algorithm consists of two important modules; an Evolution Initialization (EI) module and a Genetic Evolution (GE) module. EI consists of chromosome representation of user's request and feasible matches are then generated by the greedy population initialization procedure via distance based heuristics. GE module consist of six steps appended below and this procedure is repeated many times to obtain optimized solution.

1. Route and matching evaluation.

The route and matching evaluation procedure finds the shortest route and then records the routing results in the implicit routing layer of the chromosome. When the fitness value of chromosome is small, it indicates that better routing results are obtained.

2. Elitist chromosome selection.

After route and matching evaluation, chromosomes are sorted in ascending order and then partitioned into two sub populations as top tier and low tier. The top-tier chromosomes are retained for the next generation, and the lower-tier chromosomes will be recombined with the top-tier chromosomes during the next procedure to produce the potential offspring.

3. Exceptional trait crossover.

In this step, the initial parent chromosome is randomly and uniformly selected from the top-tier chromosomes, second parent is selected from the lower-tier chromosomes to introduce matching diversity.

4. Optimization-oriented mutation.

To obtain optimized results, two mutation operators insert/reset and swap used. Insert maximizes the number of matched passenger through insertion mutation. Insert operator is used to insert a new gene into the segment until its capacity is completely filled; otherwise, reset operator removes all genes for re-insertion. The second mutation operator is swap. According to the distance estimation value, the determinations of the mutation segments are selected from those segments assessed to have the worst fitness values.

5. Invalid chromosome repair.

In certain cases, a particular passenger assigned to more than one driver through above procedures and chromosome will become invalid. In this step, repeated passengers with the passengers which have not yet been allocated to any driver will be replaced.

6. Early stop option.

An early stop option is based on the differences between the fitness values of consecutive generations and it reduces the computational cost of the GCRMA.

In this approach, a mobile based application used to perform the route matching for drivers and passengers. RESTful web services are used to do the communications between the mobile application and the server. Considering the matching algorithm, the genetic based route matching approach used which is less in complexity and efficient time.

An Android based ridesharing and travel assistance mobile application developed by Arthi R. [15], used to search and find matching routes for the drivers and passengers who are involved in carpooling. In this study, it was mainly focused on the carpooling among the colleagues and the students in an educational institute. The ride updates which are offered by drivers are updated in this application in real time and will be shown to the application users in a map. Facebook integration in this application improves the trust and accountability between users and they can set the ride visibility to others by making it private or public. The private rides are only visible to the Facebook friends while public rides are visible to all users in the application. Using the ride sharing service the matching routes will be filtered considering the user's filtering preferences. After completing a ride, the passenger can rate the driver through the application which will be used to improve the search results.

There are three main modules in this application; Android base Mobile Client, Carpooling Web service and Google Fusion Tables for database. The mobile client will handle all the requests by the driver and the passengers and provide the results by communicating with the carpooling web service. The web service is the main component of the application which process the mobile client requests and provide the matching route results using the ride matching algorithm. In this algorithm, following steps are used to filter the ideal routes for the carpooling users.

1. Purpose filtering.

The algorithm identifies the carpooling request type and accordingly the data set will be identified for the user. The algorithm will select the passenger data set for drivers and the driver data set for the passengers.

2. Time filtering.

This step used to identify the latest ride offers and requests from the real time data set in the application.

3. Bounds filtering.

In this step, a virtual box is defined around the driver's route and the passengers will be identified inside this virtual box for matching.

4. Route filtering.

In this step, the algorithm identifies alternate paths for the user's route and determine the most ideal path using the following formula. The route with the highest points will be chosen as the efficient and ideal paths and it will be sent to the user through the web service.

$$P = n/(dt)$$

P - the points to the route

n - the number of users who can be pooled in the route

d - the distance of the route

t - the time to travel through the route

Google Fusion Table free service is used as the database to store route related geo-coordinate data which will be accessed by the web service to search matching results.

In this study, the gap between the driver and the passenger, is reduced by using the social media integration and suggests integrating other social medias to reduce the gap further. The main disadvantage of this application is, the users can't customize the routs by adding waypoints to the route as they prefer.

The experimental study undertaken by Teodorović D. and Dell' Orco M. [16], propose a methodology to solve the ride matching problem in carpooling, by using the Bee Colony Optimization which use collective intelligence. This algorithm has been applied in various combinatorial optimization problems. One of the primary goals of this study is to explore and identify possible applications of collective bee intelligence in solving the combinatorial problems characterized by uncertainty. Following are the main objectives of this study.

1. Minimize the total distance traveled by all participants.
2. Minimize the total delay.
3. Make vehicle utilization relatively equal.

The ride matching problem can be considered as deterministic combinatorial optimization problem, or as a combinatorial optimization problem characterized by uncertainty. In this study an attempt has been made to develop a methodology to solve the ride-matching problem.

In this approach, every passenger involved in the carpooling (including the driver) considered as a node and the matching problem is decomposed in stages. The first passenger (driver) in the car represents the first stage, the second passenger to join will be in the second stage and the third passenger to join will be in the third stage and so on. When they are matching, each artificial bee forward passes (going forward) and visit certain number of nodes, creating a partial solution and return to node zero which is considered as hive (Figure 2.3). In this hive, artificial bees are participating in a decision making process to decide following options.

1. Abandon the partially generated path and become again an uncommitted node.
2. Continue with partially discovered path without using more artificial bees.
3. Recruit more artificial bees before returning to the discovered path.

In this process, each and every artificial bee assigned with certain level of loyalty depending on the quality of the partial solutions generated. During the second forward pass, artificial bees will access few more nodes in the previously discovered and then backward pass (coming back) and return to hive to participate in a decision making process. This iteration will be continued until all nodes are discovered. Artificial bees perceive a specific node as 'less attractive', 'attractive' or 'very attractive', depending on the proximity in space and proximity in time between two passenger requests, when choosing the next node to be visited. These proximities considered as 'distance in space at origin', 'distance in space at destination', and 'distance of arrival times'.

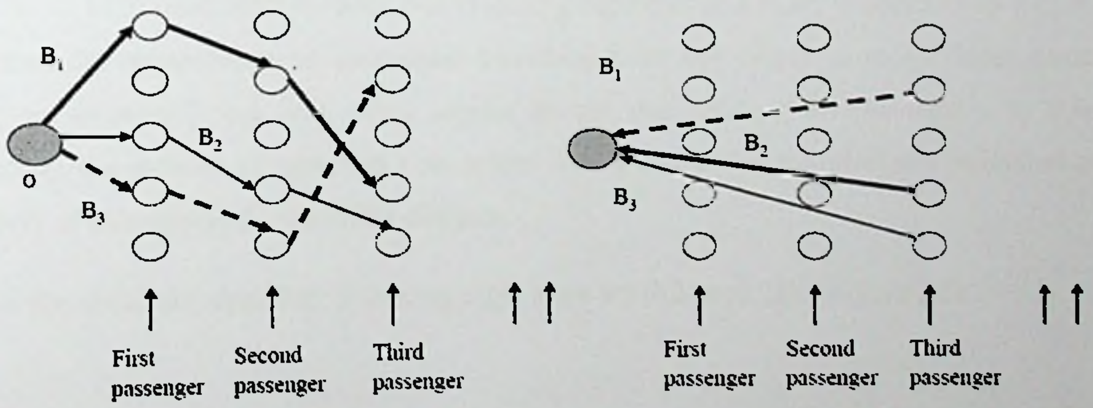


Figure 2.3: First forward pass (left) and first backward pass (right)

To determine the node attractiveness to visit, an artificial bee can perceive a particular distance between nodes as 'short', 'medium' or 'long' and following condition will be used.

1. If the distance in space at origin is SHORT, and the distance in space at destination is SHORT, and the distance of returning times is SHORT.
2. Then the node attractiveness is HIGH.

The path badness attribute is used in the corresponding approximate reasoning algorithm to determine bee's loyalty on the visited path. The attractiveness value will then be used to select the best route and will use the following conditions.

1. If the length of the advertised path is SHORT, and the number of bees advertising the path is SMALL
2. Then the advertised path attractiveness is MEDIUM

In summary, this Bee Colony Optimization method focus on three main objectives, (1) minimize the total distance travelled, (2) minimize total delay and (3) make vehicle utilization relatively equal. This ride sharing experiment is evaluated using real world scenarios and the results yielded were successful. Due to the success of this approach, the route planning facility have the capability of picking up several passengers in a route.

Tao C. [17], provided a dynamic ride matching algorithm in a study undertaken by him to meet the requirements of passengers travelling from one origin to many destinations ("one-to-many") and from many origins to one destination ("many-to-one"). In this study, the problem of matching a passenger with a driver was analyzed and refined as a way of minimizing the travelling distance.

In this matching algorithm, following eight steps are followed. (See Figure 2.4)

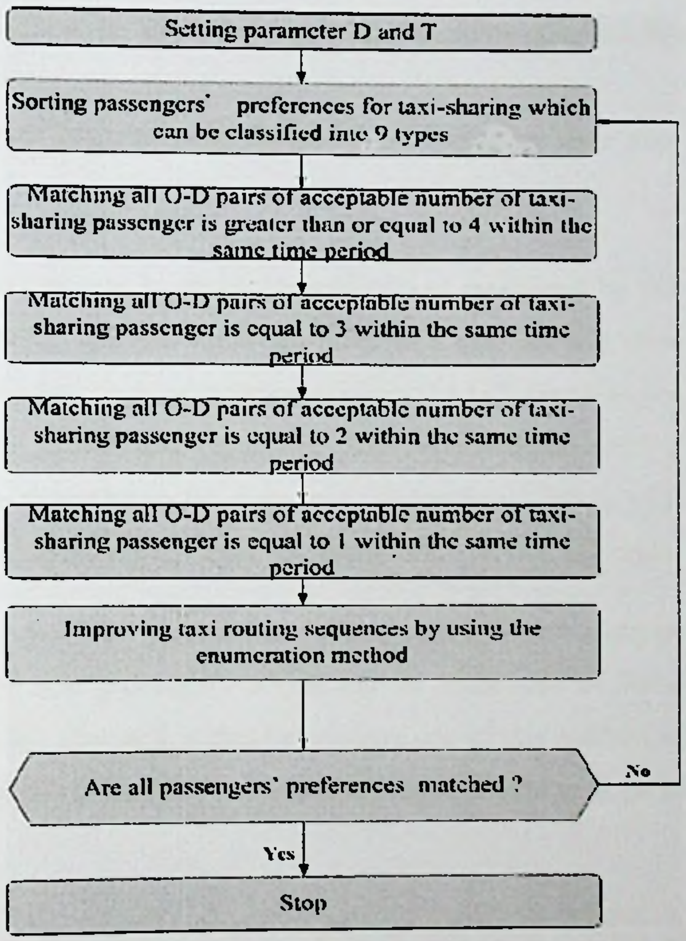

Setting parameter D and T
Sorting passengers' preferences for taxi-sharing which can be classified into 9 types
Matching all O-D pairs of acceptable number of taxi-sharing passenger is greater than or equal to 4 within the same time period
Matching all O-D pairs of acceptable number of taxi-sharing passenger is equal to 3 within the same time period
Matching all O-D pairs of acceptable number of taxi-sharing passenger is equal to 2 within the same time period
Matching all O-D pairs of acceptable number of taxi-sharing passenger is equal to 1 within the same time period
Improving taxi routing sequences by using the enumeration method
Are all passengers' preferences matched ?
No
Yes
Stop

Figure 2.4: Steps involved in dynamic ride matching

1. Setting the searchable distance and maximum passenger waiting time.
2. Matching and sorting by passenger's preferences and the results will be classified. If unable to match step 3 it will not be taken.
3. Matching all O-D pairs of acceptable number of taxi-sharing passenger is greater than or equal to four within the same time period.
4. Matching all O-D pairs of acceptable number of taxi-sharing passenger is equal to three within the same time period.
5. Matching all O-D pairs of acceptable number of taxi-sharing passenger is equal to two within the same time period.
6. Matching all O-D pairs of acceptable number of passenger for taxi-sharing is equal to one within the same time period.

7. Improving taxi routing sequences by using the enumeration method to identify the shortest route.
8. If all passengers' preferences are matched, then stop; else, go to step 1.

In this system, all the taxi locations are tracked and monitored by the system. When a passenger requests a taxi, the algorithm matches a suitable taxi from the system and compared with the taxi position information from the GIS based server. Then, matching results will be sent to the requestors and the taxi driver will also notify about the pick-up. While passengers travelling on the shared taxi, all locational details will be tracked by the system, so that the operators can track and confirm the service any time.

The main advantage of this approach is, it can produce the matching results faster and it can process over four passengers which can be made use of for any vehicle. The disadvantage of this approach is that the success rate of this method will be zero, if the number of passengers is too small or passenger locations are too dispersive.

2.3. Summary

There are many studies done in different areas of carpooling systems such as finding the shortest path, finding the quickest path, passenger matching, improve the searching performance etc. with a focus on improving the functionalities of carpooling systems. The applications of the carpooling systems are different according to the requirements. The carpooling applications can be used for private/organizational/public domains and can be implemented using different architectures and platforms such as web based, desktop, mobile etc. In the above literature review, existing route matching and passenger matching approaches reviewed.

According to the public carpooling systems discussed so far, if a particular passenger wants to find a route which is going through a given starting and destination locations, the application will provide all the routes which are going through the given start and the end locations. If the passenger is provided with a large number of matching results, to select an ideal and precise path, the passenger is compelled to browse through each and every result one by one, which is a disappointing process. Also, the resulting routes may not be closely matched to the preferred path and either it may be deviated or routes with longer distance routes may have been returned. To address this kind of problem, the enhanced searching method is important for carpooling systems.

Another drawback of the current systems is that; a driver is unable to get an idea about the available passengers in his area of preference. If the system is capable of providing relevant information of the passengers waiting in that area for lifts, it will be advantageous to the driver as well as to the community. Hence, analysis of route requests by the passengers in that area and generation of alternative routes for drivers' will be an opportunity to the drivers to adopt an appropriate system generated route, which meets matching passengers for the driver.

In the next chapter, the technologies used to develop the proposed application will be discussed.

Chapter 3

Technology adopted – Route searching and suggesting

3.1. Introduction

This section will provide the full description of the tools and technologies adopted to implement the route searching and route suggesting module. This is a web based application, as such the PHP [14] server side scripting language has been used to perform the server side processing. The MySQL [15] tool has been used as the database which will store the general data, as well as the route related Spatial data. The Apache web server [16] will be used to host the web application. The WAMP [17] server tool has been used to get above mentioned three tools together as a bundle. The Google Maps JavaScript API [18] used as a mapping tool. The bounding box calculations on the map will be performed by the RouteBoxer [19] JavaScript utility library. In order to improve the interactivity of the application and to perform the asynchronous calls to the server, JQuery JavaScript library will be used.

3.2. PHP scripting language

The proposed carpooling system was developed as client server architecture, therefore the PHP server side scripting language was adopted to process the client requests and to perform the database requests. The main reasons to use this server side scripting language over the other languages are appended below.

- Free and open source
- Easy to program since it is similar to C/C++ syntaxes
- Platform independent and supports for major web servers

3.3. MySQL database management tool

The MySQL free database tool is used to store general data of the system and route related geo-coordinate data in Spatial tables. The main reason to use this tool is, it supports the Spatial functions [20] such as calculate distance, calculate area etc. in Spatial tables.

3.4. Apache web server

To host the developed web system, Apache web server is used. This web server is available freely and it supports many operating systems.

3.5. WAMP server

The WAMP server is a free web development environment for Windows operating system. It provides many tools such as PhpMyAdmin, SQLBuddy, XDebug etc. which reduce the application development effort.

3.6. Google Maps JavaScript API

The Google Maps API used as the map tool for the system which will supports powerful object oriented features which will make it easy to develop the application.

3.7. Route Boxer utility library

This library has been used to calculate the geo-coordinates of bounds on a particular route on the map, which is used to identify the routes in the database using Spatial functions provided by MySQL database tool.

3.8. JQuery JavaScript library

JQuery JavaScript library used to increase the interactivity of the system. Built in functions in this library will be used to perform the requests from the database using the Asynchronous JavaScript and XML (AJAX).

3.9. Summary

Installing and configuration of PHP, MySQL and Apache web server is a time consuming process. Therefore, the WAMP web development environment has been used, since it has the required tools together, which reduce the installations and configurations. In the next chapter, the approach for the route searching and route suggesting modules will be discussed individually.

Chapter 4

Route searching and suggesting approach

4.1. Introduction

This section will discuss the approach which is used to implement the route searching and route suggesting modules. The route searching module will be used by the passengers to search matching and ideal routes for a given route, by analyzing the lift offers in the database. The alternate route suggesting module will be providing route suggestions to the drivers, when adding a lift offer in the application.

4.2. Route searching approach

The application requires the lift offer details of drivers in the database to perform the searching and filtering for matching routes for the prospective passengers. The relevant route details, such as starting point, destination, way points, date and time etc. can be collected from the drivers through a map interface (See Figure 4.1). Then the details gathered can be saved in the MySQL Spatial database table.

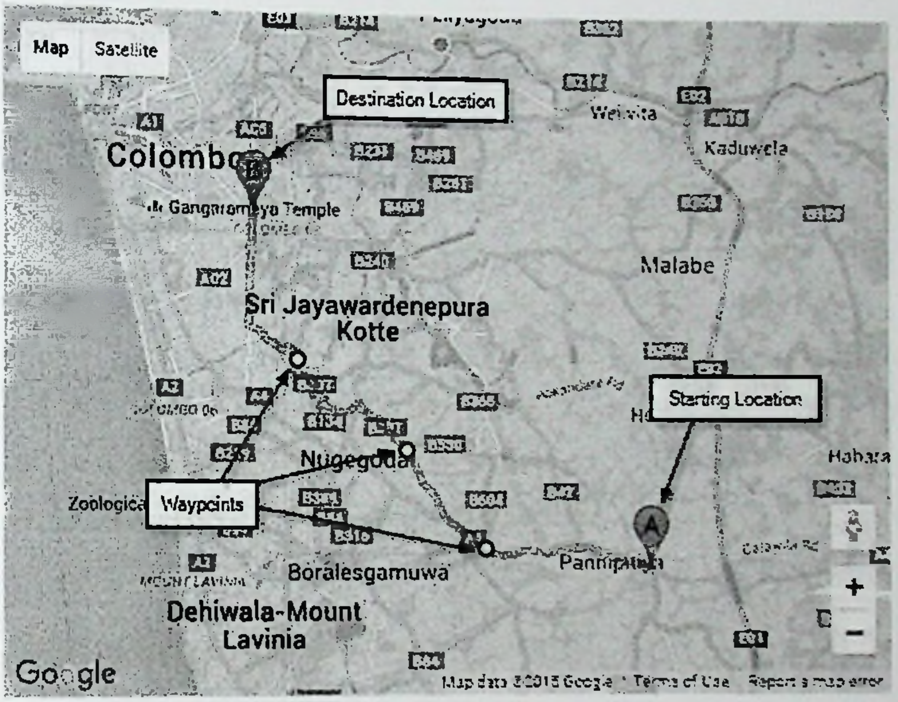


Figure 4.1: Properties of a route

The passengers can search for the matching routes, from the map provided in the application, by providing the starting point, destination point, waypoints etc. When the user clicks the search button in the interface after setting up his preferred path, the application will generate the bounding boxes for the path using the RouteBoxer utility library. Then the client side JavaScript will collect the geo-coordinate data for the relevant route and then bounding boxes will be generated. With this data, the application will generate a Java Script Object Notation (JSON) to be sent to the server from the JQuery AJAX function. The data received from the client side will be extracted from the server side, and will be prepared for the searching algorithm using a PHP script.

Following steps in the searching algorithm will be performed to filter the matching routes.

1. Define a circle by using Haversine formula [21] around the starting point of the route which has a predefined radius; and query the points inside the circle to identify the unique routes inside it.

2. Query the unique routes at the destination of the route by using the same approach referred to in the step one above.
3. From these routes, identify the routes which are going through both starting and destinations locations which were queried and identified under step1 and 2 above.
4. Select all the points for each resulting route and check the availability of each point inside each bounding box generated for a given path.
5. According to the availability of the points inside the boxes, the routes will be ranked. (E.g.: From the total points of 20, if 15 points fall inside the bounding box, the accuracy level will be (15/20) *100 = 75 %)
6. These results will be sorted according to the accuracy level and will be shown to the user.

4.3. Alternate route suggesting approach

The system needs the details of the route requests from the passengers to perform the alternate path suggestion. This route information can be collected from the passengers through a map interface to set up an alert. When the driver offers a lift by adding route details, the application will generate larger route boxes to cover a wider area for searching. This route box data and the given route data then will be sent to the server for processing, using the JQuery AJAX library functions.

Following steps will be performed in the searching algorithm to search a matching route and then to make alternate route suggestions to the driver;

1. Routes in the route alert table, which have the same starting, destination and waypoints that are stored in the bounding boxes will be filtered out.
2. Compute the bearing angle of each path and compare them with the original path to identify the routes with similar angles by filtering.
3. Eliminate the routes which are having the opposite direction against that of the original path, by comparing the geo-coordinates of starting and destination points.
4. Apply the Haversine formula to starting and destination locations to filter out unwanted routes. (See Figure 4.2)

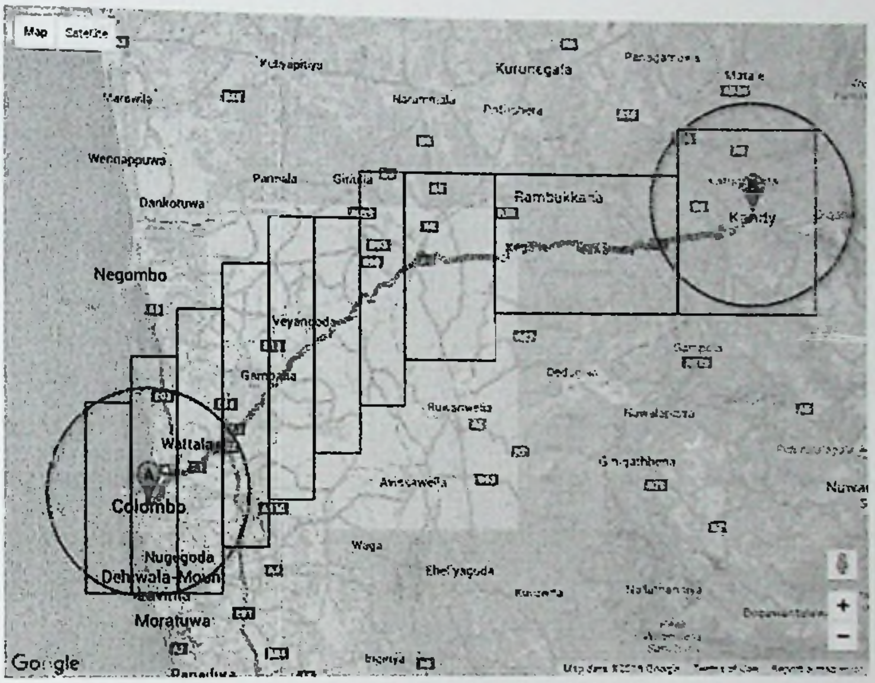


Figure 4.2: Applying Haversine formula

4.4. Summary

This section highlighted the approach of route searching algorithm and route suggestion algorithm. In the next chapter, the analysis and the design of these two modules will be discussed further. Apart from these two main modules, the analysis and design of other sub modules also will be discussed.

Chapter 5

Analysis and design

5.1. Introduction

In this chapter, the analysis and design of each module in the proposed system will be discussed. The proposed system will be a web based application. The server back-end will have route searching module, route suggesting module and Spatial database management module. The top level design of the proposed system is represented graphically in Figure 5.1. Each component and module included in the proposed system will be discussed in detail in the following sections.

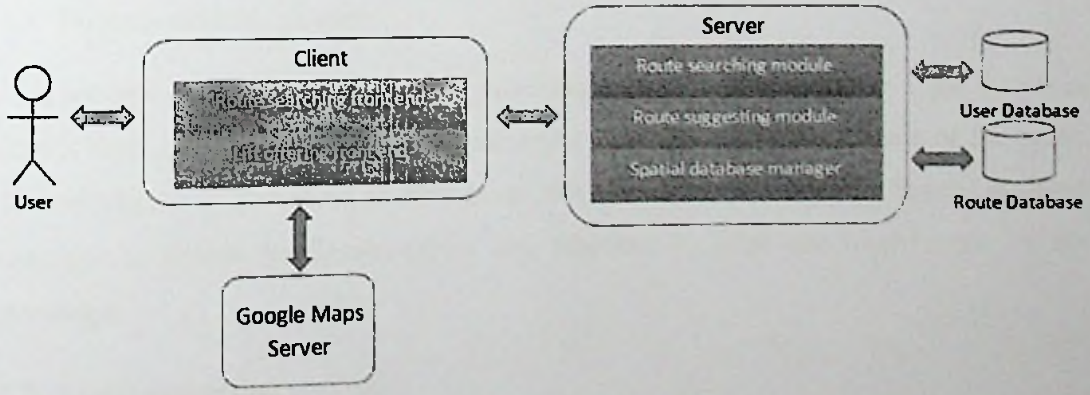


Figure 5.1: Top level system design

5.2. Route searching front-end

This component is used for collecting and showing the route details to the passenger, when a passenger performs a route searching. If the passenger wants to search a matching route in the database; in the first step, user has to setup the preferred route in the map providing the start, destination and waypoint locations. From these data, the route will be indicated on the map using the Google Maps services. This component also communicates with the server to search and show the results in the interface. In this component, the bounding boxes will be created using the RouteBoxer utility library which will be used to search matching routes in the database. The passengers can setup alerts in the application to get notification if there are matching routes available and this interface provides the functionality to setup up the preferred route in the map.

5.3. Lift offering front-end

This component will be used by the driver to add a proposed lift in the application by defining the preferred path in the map. This component uses the Google Maps services to render the routes in the map using the inputs such as origin, destination and waypoints. This component too will interact with the server to save the offered route by the driver and to show the suggested alternate paths in the map. The RouteBoxer utility library will be used to generate the bounding boxes for the preferred path by the driver for searching purposes.

5.4. Route searching module

This module is used for searching the matching routes for the passenger using the given inputs such as start point, end point, waypoints and bounding box details of the route. This module is implemented on PHP script, which will interact with the Spatial database manager to access the Spatial tables and functions to filter matching routes for the passenger.

5.5. Route suggesting module

This module is designed to search for matching routes for the route preferred by the driver, and it will be used to search suggested routes. By using the input data in the lift offering component, the matching routes will be filtered from the database with the help of the Spatial database manager module.

5.6. Spatial database manager

This module will be designed to interact with the Spatial database, for application. All the Spatial related queries from route searching module and the route suggesting module will be handled by this module.

5.7. Summary

In this chapter, the analysis and design of route searching front-end, lift offering front-end, route searching module, route suggesting module and the Spatial database manager module is discussed. In the next chapter, implementation of these modules will be discussed.

Chapter 6

Implementation

6.1. Introduction

In this chapter, the implementation details of route searching front end, lift offering front end, route searching module, route suggesting module and Spatial database manager module will be discussed. These details will be discussed in the following sections separately.

6.2. Route searching front-end

This is the front end visible to the passengers when they are login to the application. In this interface they can search matching routes by setting up the preferred route in the map provided. To setup the route in the map, user has to select the starting point and the destination location in the given fields. These fields are autocompleting fields, which provide suggestions when user starts typing the location names. These autocomplete fields are implemented using the autocomplete feature provided by the Google Maps API. (See Figure 6.1)

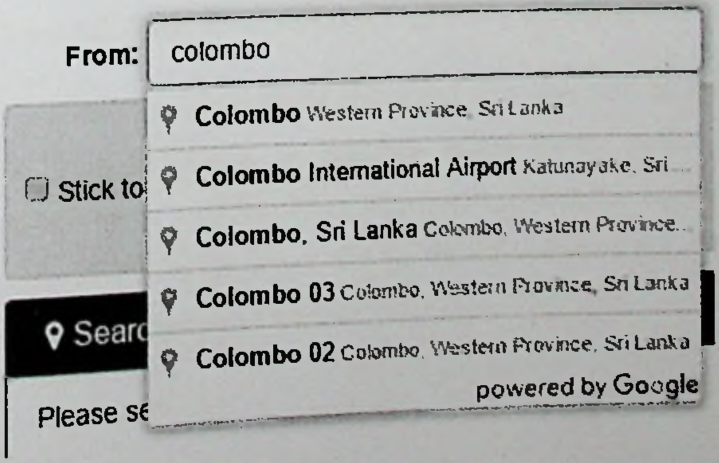

From:
colombo
Colombo Western Province, Sri Lanka
Colombo International Airport Katunayake, Sri
Colombo, Sri Lanka Colombo, Western Province
Colombo 03 Colombo, Western Province, Sri Lanka
Colombo 02 Colombo, Western Province, Sri Lanka
powered by Google
Stick to
Searc
Please se

Figure 6.1: Location suggestions for typed letters

When the user clicks the search button to search matching routes in the database, these starting and destination location names are converted to the relevant geo-coordinates using the Google Maps geocoder service. Also, bounding boxes will be generated simultaneously for the route, through the RouteBoxer library. Figure 6.2 shows the JavaScript function to generate the bounding box for a given route in the map.

```
var rboxer = new RouteBoxer();
var Rdistance = 10;
var RouteResponse;
function updateMapWhenPlaceChanges(){
    if(PlaceFrom != "" && PlaceTo != ""){
        directionsService.route({
                origin: PlaceFrom,
                destination: PlaceTo,
                travelMode: google.maps.TravelMode.DRIVING
            },
            function(response, status){
                if (status === google.maps.DirectionsStatus.OK) {
                    directionsDisplay.setDirections(response);
                    RouteResponse = response;

                    var path = response.routes[0].overview_path;
                    var boxes = rboxer.box(path, Rdistance);

                    for (var i = 0; i < boxes.length; i++) {
                        var bounds = boxes[i];
                    }
                }
            }
        );
    }
}
```

Figure 6.2: RouteBoxer function to generate bounding boxes

These data will be composed into a JSON object and then they will be sent to the route searching module using AJAX call, for matching routes for the passenger. The route searching module will return a JSON object, which will contain the search results to this front end. A sample JSON results object will be shown in Figure 6.3. The data in this object will be extracted and shown to the user.

```
{
    "0":{
        "status":"success","message":"Results found."
    },
    "1":{
        "0":{"RouteId":"2016-03-27!06:19:35","StartAddress":"Colombo","EndAddress":"Trincomalee"},
        "1":{"startLat":"6.9270974","startLng":"79.8612478"},
        "2":{"endLat":"8.5921964","endLng":"81.1965936"}
    },
    "2":{
        "0":{"RouteId":"2016-03-27!06:31:41","StartAddress":"Colombo","EndAddress":"Trincomalee"},
        "1":{"startLat":"6.9270974","startLng":"79.8612478"},
        "2":{"endLat":"8.5921964","endLng":"81.1965936"}
        },
    "3":{
        "0":{"RouteId":"2016-03-27!06:37:57","StartAddress":"Colombo","EndAddress":"Trincomalee"},
        "1":{"startLat":"6.9270974","startLng":"79.8612478"},
        "2":{"endLat":"8.5921964","endLng":"81.1965936"}
    }
}
```

Figure 6.3: Sample JSON route results object

In this front end, passengers can setup route requests/alerts in the application by setting up a route in the map. When a user adds an alert from this interface, all the route details are stored in the database. In the process of saving data, all the geo-coordinates of the route will be saved in the Spatial database table, for matching purposes. This set of geo coordinates are retrieved from the Google Maps directions result object.

6.3. Lift offering front-end

The driver can offer a ride in the system by using the front end. In this process, the user has to define the route in the map, providing the route details as in the route searching front end. (See figure 6.4) Once, a user adds a lift offer in the application, all the route details will be sent to the route suggesting module and saved to the Spatial database table. In return, the route suggesting module will send the list of suggested results as a JSON object, which will be resolved and shown to the user by this frontend component.

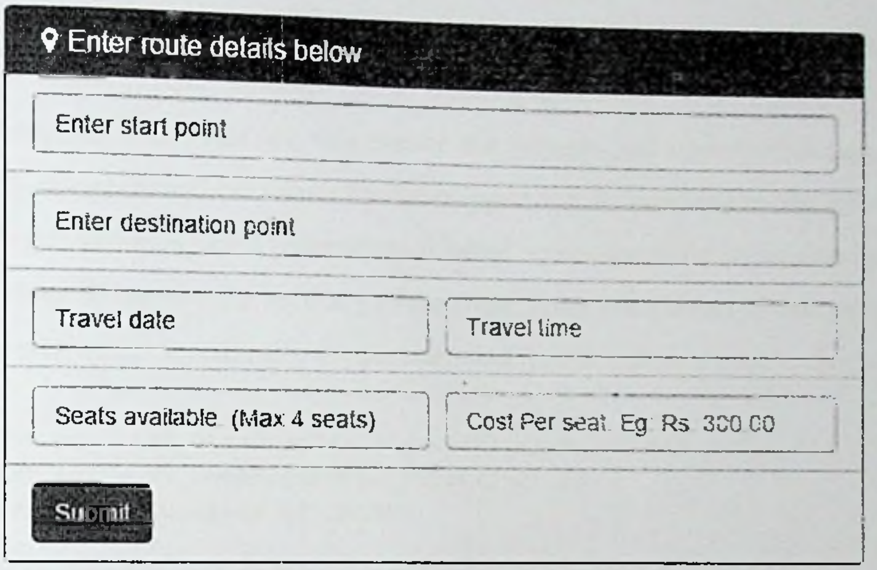

Enter route details below
Enter start point
Enter destination point
Travel date
Travel time
Seats available (Max 4 seats)
Cost Per seat. Eg: Rs. 300.00
Submit

Figure 6.4: Fields in the lift offering front-end

6.4. Route searching module

This module is used for searching matching routes to the passengers. In the process of searching for matching routes, in the first place the algorithm defines a circle around the starting point which has the user agreed radius. Then, it searches for all the points falling within the circle through the Spatial database table, which has a record of lift offers made by the drivers. To calculate distances from the starting point to the points in the Spatial table, the Haversine formula is used, which will calculate the great circle distance. From these results, the distances are less than the circle radius will be filtered. This same route searching procedure will be repeated for the end point of the passenger's route and filter the routes which are inside the end point circle. Considering these two sets of results, it will filter the routes which are going through both starting and destination point circles, to get the list of routes for the passenger who can accordingly travel from starting to destination. (See Appendix A for the SQL to filter these routes.)

The Spatial database table maintains all the points for each route, which form the route. Therefore, to get the most ideal and matching routes, each point of the route will be validated for availability inside the bounding boxes created for the route of the passenger.

This validation can be performed using the Spatial database functions which are supported by MySQL database tool. (See Figure 6.5 and Figure 6.6) The "MBRContains()" SQL function will require the polygon and a geo-coordinate point to check this validation and the values for the polygon will be provided by the RouteBoxer library. When validating each point of the selected routes, the valid count also calculated for each route to identify the accuracy of the route. After completion of this process, the results will be sorted according to the accuracy and results will be shown to the user.

```
SET @g1 = ST_GeomFromText('Polygon((1 1,1 4,4 4,4 1,1 1))');
SET @g2 = ST_GeomFromText('Point(2 2)');
SELECT MBRContains(@g1,@g2);
```

Figure 6.5: MySQL MBRContains function usage

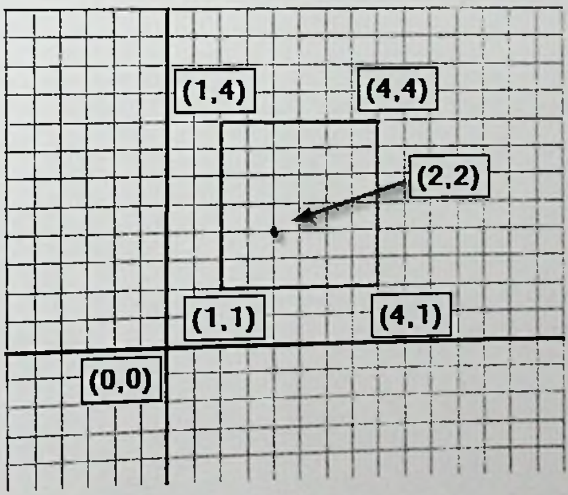


Figure 6.6: MBRContains function operation

6.5. Route suggesting module

The application, can show the route suggestions to the drivers with the help of this module, by analyzing the route requests made by the passengers, who are around the driver's path. When the driver is adding a lift to the application, bounding boxes are created to cover a larger area using the RouteBoxer library. This is the area, within which the algorithm searches for alternate paths for the driver. Then, all the bounding box data and the route data will be sent to the server for searching alternate paths within the interested area.

In the searching algorithm, the routes stored in the route alerts table will be filtered out, which contain the starting, destination and waypoints available inside the bounding boxes. This result will contain all the routes which have all the starting, destination and waypoints inside the interested area. However, the results may contain routes with different directions and routes where the driver is unable to travel. (See Figure 6.7 for possible routes that may be in the results)

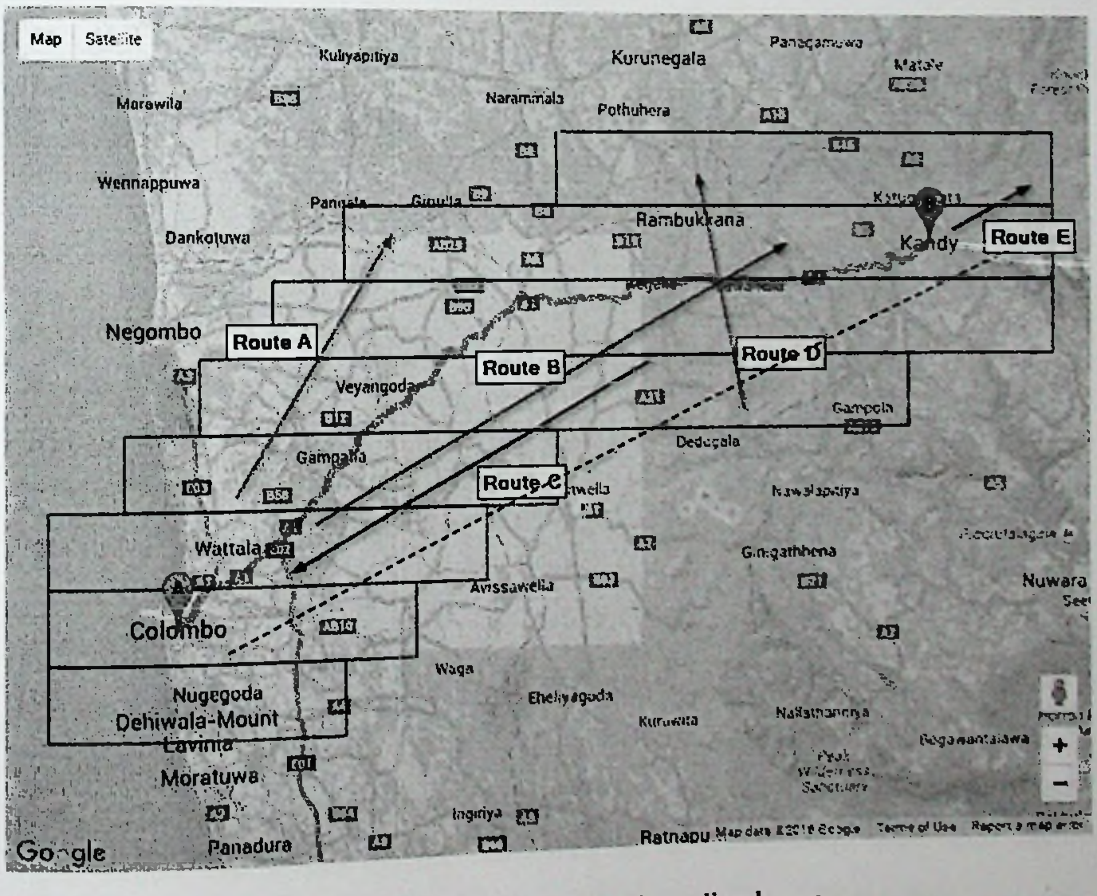


Figure 6.7: Possible routes in bounding boxes

According to the Figure 6.7, the driver's route is A to B. The routes which are perpendicular to the original route or having considerable angles of deviations, have to be removed from the results, as the driver is unable to drive along such routes. This elimination can be done using the Bearing angle [22] calculation of each path and comparing with the original path. For this Bearing angle calculation PHP function will be used. (See Figure 6.8 for the PHP function)

```
/*Calculate bearing angle of two points*/
function getGreatCircleBearing($lat1, $lon1, $lat2, $lon2){
    $bearing = (rad2deg(atan2(sin(deg2rad($lon2) - deg2rad($lon1)) * cos(deg2rad($lat2)),
    cos(deg2rad($lat1)) * sin(deg2rad($lat2)) - sin(deg2rad($lat1)) * cos(deg2rad($lat2)) *
    cos(deg2rad($lon2) - deg2rad($lon1)))) + 360) % 360;

    return $bearing;
}
```

Figure 6.8: PHP function to calculate Bearing angle

With the help of this PHP function, the Bearing angle will be calculated for each path considering the start and end points. Figure 6.9, shows the bearing angles of two points compared to each other. The angle will be counted clockwise starting from the north.

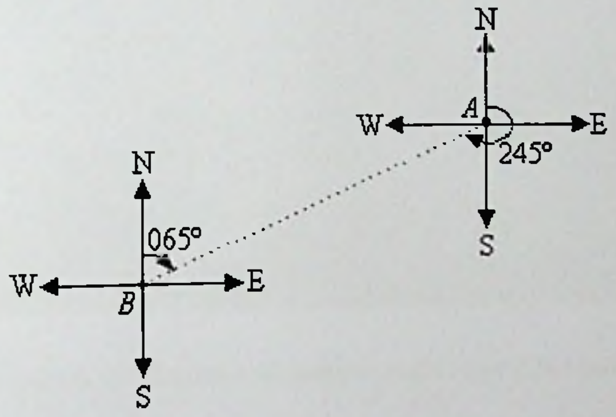


Figure 6.9: Bearing angles of two points

Considering the Bearing angle (θ) of the original path, a range of angles (θ +20° and θ - 20°) will be determined to eliminate deviated routes in the results. From this filtering, the route A and D can be removed from the results and the routes with similar angle can be acquired. (See Figure 6.10, for routes with similar angles to the original route)

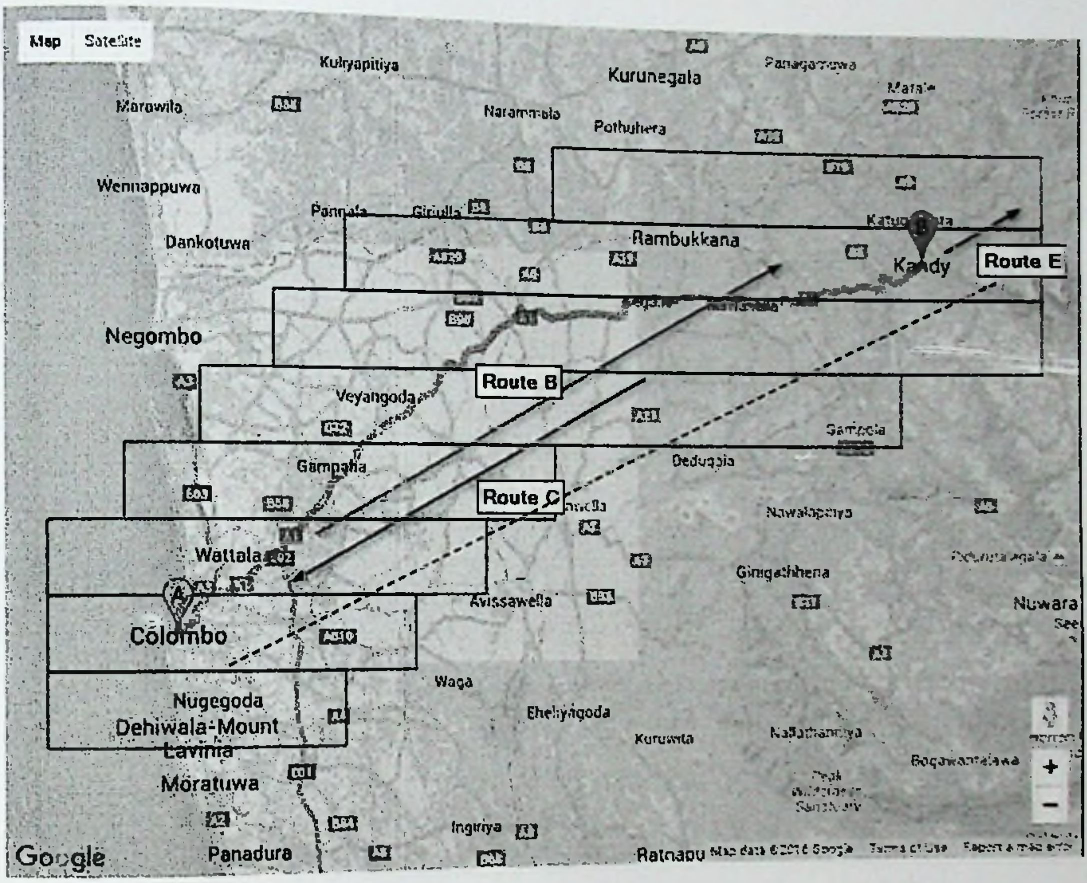


Figure 6.10: Routes with similar angle to original path

In the remaining results, there can be routes which are having opposite direction from the original path such as route C. To eliminate the routes which are having the opposite direction, the start to destination direction can be identified by comparing the geo-coordinates. After this filtering, the routes with similar angle to the original path and similar direction to the original path can be identified. See Figure 6.11, for filtered routes, which are having same angle and same direction to the original route.

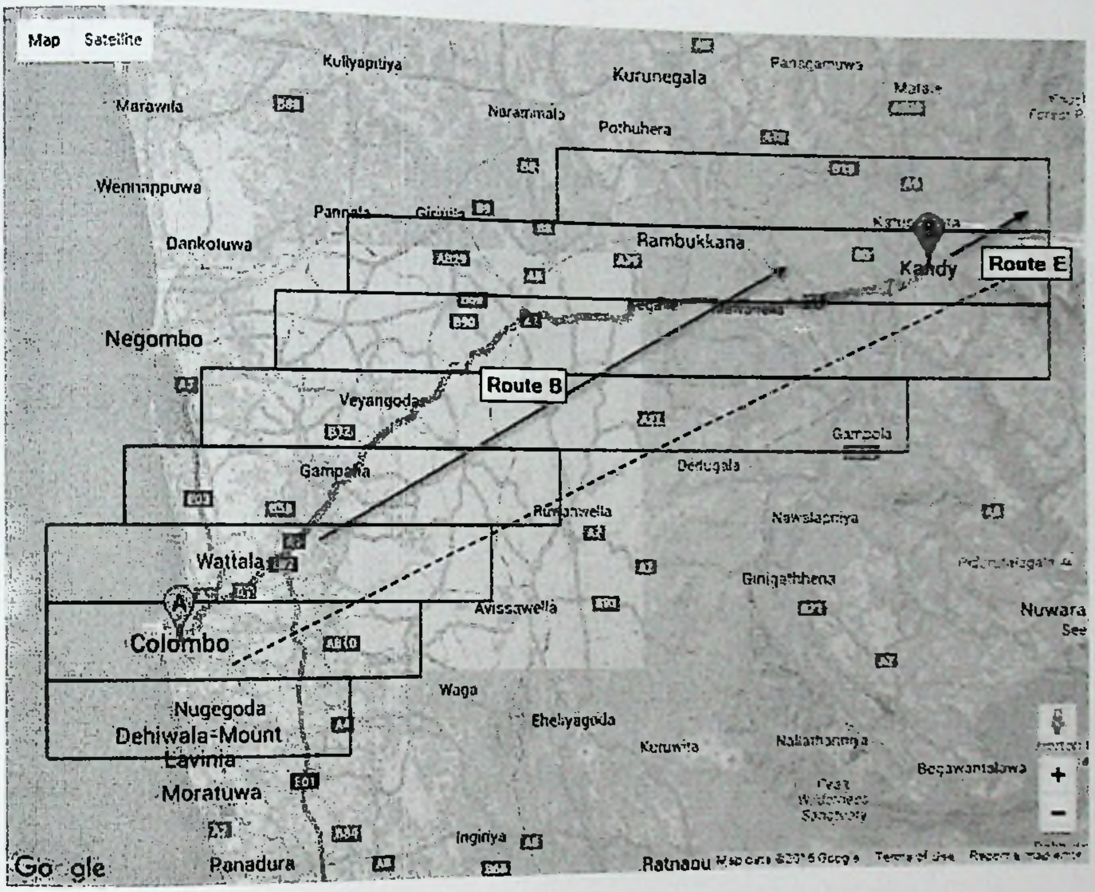


Figure 6.11: Routes with similar angle and similar direction

Analyzing the remaining results, there can be situations similar to route E, where the driver may want to drive away from the starting or destination locations, to get a passenger. These routes can be eliminated by applying the Haversine formula to starting and destination locations of the original path, considering the distance used for bounding box as radius. After this filtering, the routes with similar angle, similar direction and between the starting and destination location routes will be filtered. (See Figure 6.12).

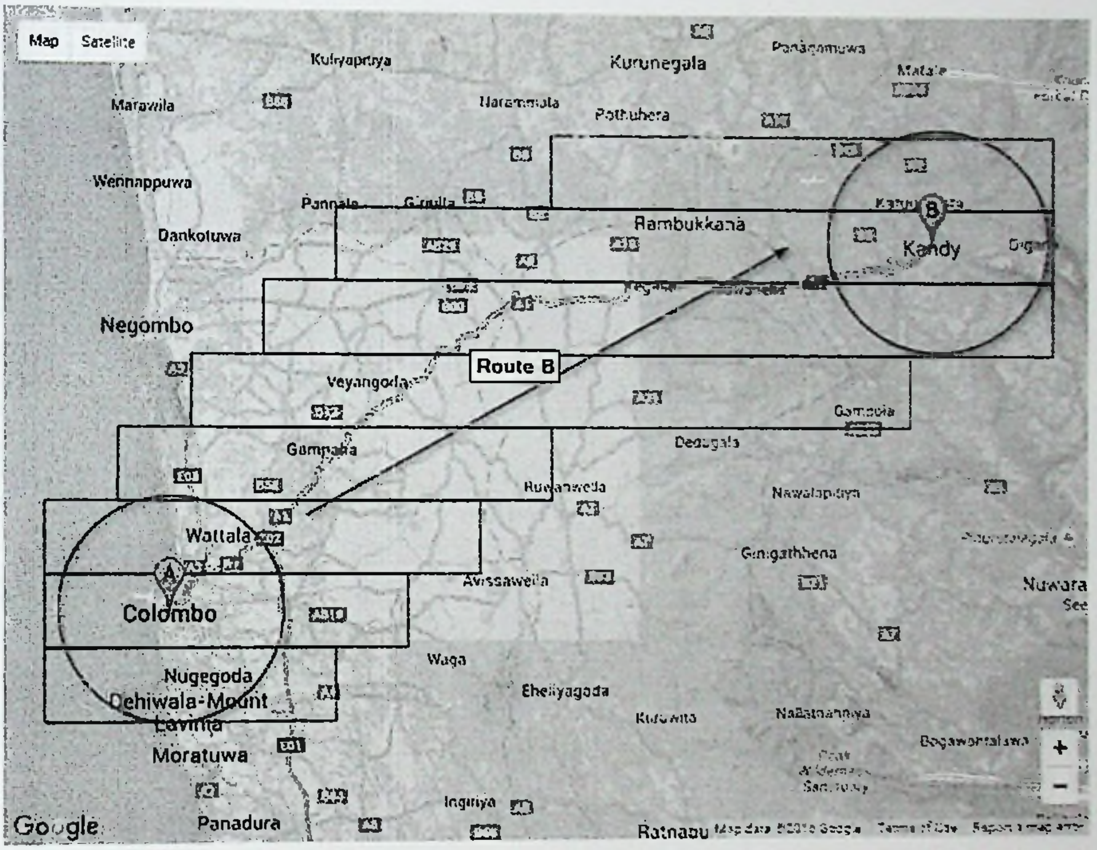
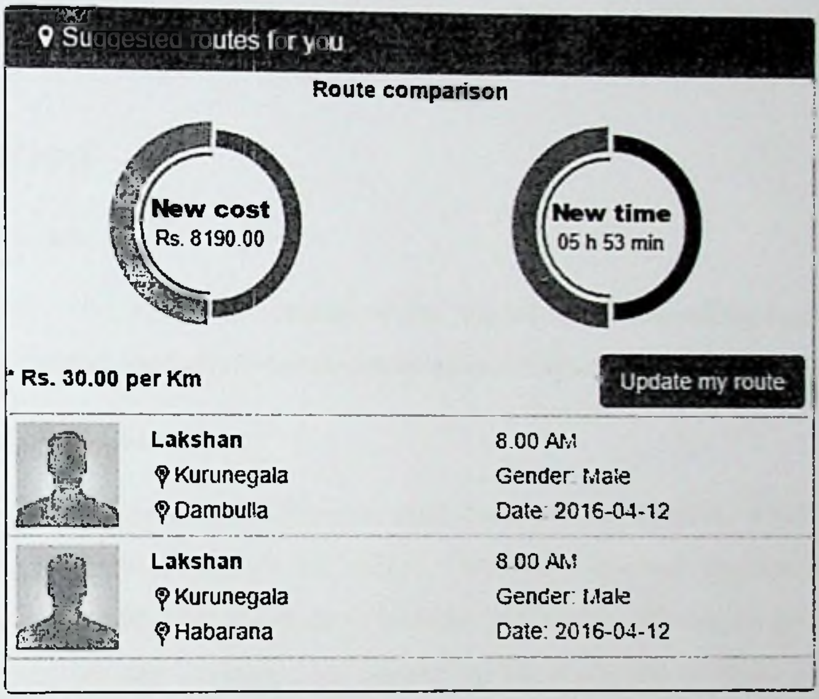


Figure 6.12: Area applied by Haversine formula

To improve the results further, all the points in each route are checked for availability inside the bounding boxes of the original route and considering the availability of points the routes will be ranked. From these set of filtering, the considered area will be inside the bounding boxes and outside the two circles in starting and destination locations. These results will be the finally suggested routes and then will be shown to the driver.

When driver select a suggested route from the suggested route list, the route comparison summery between driver entered route and the suggested route will be shown to the driver. This summery show the cost and time comparison using interactive Pie charts which will support drivers' decision making. (See figure 6.13)


Suggested routes for you

Route comparison

New cost
Rs. 8190.00

New time
05 h 53 min

* Rs. 30.00 per Km

Update my route

Lakshan	8.00 AM
Kurunegala	Gender: Male
Dambulla	Date: 2016-04-12
Lakshan	8.00 AM
Kurunegala	Gender: Male
Habarana	Date: 2016-04-12

Figure 6.13: Suggested route summary for driver

6.6. Spatial database manager module

This module is responsible for integrating the application with the external database. All the database queries performed by route searching module and the route suggesting module will access the database through this module.

6.7. Summary

In this chapter, the implementation details of each module in this application is discussed. In the next chapter, the evaluation process of the proposed system will be discussed.

Chapter 7

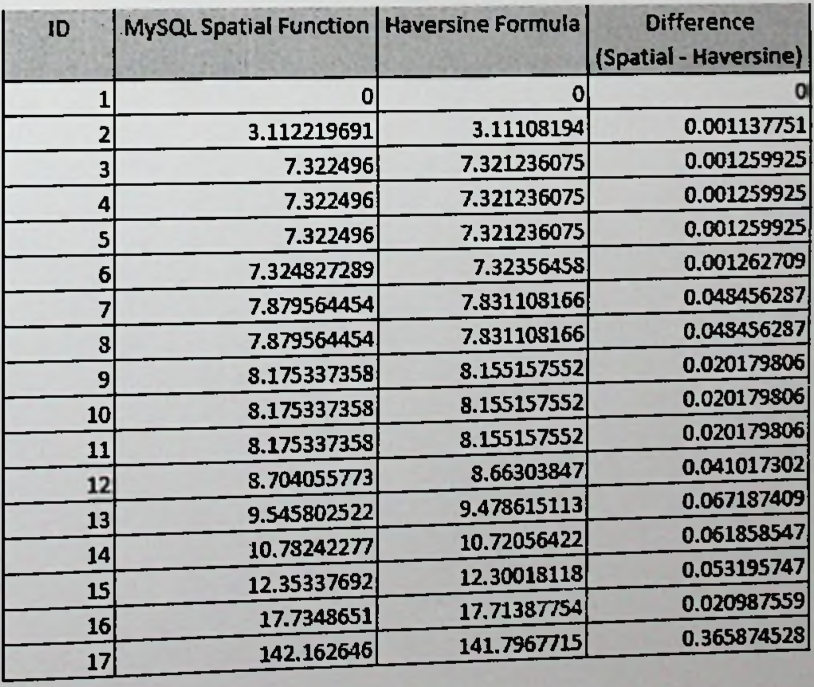
Evaluation

7.1. Introduction

In this chapter, the evaluation details of the implemented carpooling system will be discussed. Different evaluations are discussed in separate sections appended below.

7.2. Verifying the distance

In this application, the distance filtering used the Haverine formula, which is used to calculate the great circle distance in a sphere. The Spatial database engines also provide inbuilt functions to calculate the distance between two points. The results generated from these two methods are compared by calculating the distances to same points in the database. The analysis of these two values are appended below. (See Figure 7.1 and Figure 7.2)



ID	MySQL Spatial Function	Haversine Formula	Difference (Spatial - Haversine)
1	0	0	0
2	3.112219691	3.11108194	0.001137751
3	7.322496	7.321236075	0.001259925
4	7.322496	7.321236075	0.001259925
5	7.322496	7.321236075	0.001259925
6	7.324827289	7.32356458	0.001262709
7	7.879564454	7.831108166	0.048456287
8	7.879564454	7.831108166	0.048456287
9	8.175337358	8.155157552	0.020179806
10	8.175337358	8.155157552	0.020179806
11	8.175337358	8.155157552	0.020179806
12	8.704055773	8.66303847	0.041017302
13	9.545802522	9.478615113	0.067187409
14	10.78242277	10.72056422	0.061858547
15	12.35337692	12.30018118	0.053195747
16	17.7348651	17.71387754	0.020987559
17	142.162646	141.7967715	0.365874528

Figure 7.1: Distance difference of Spatial function and Haversine formula

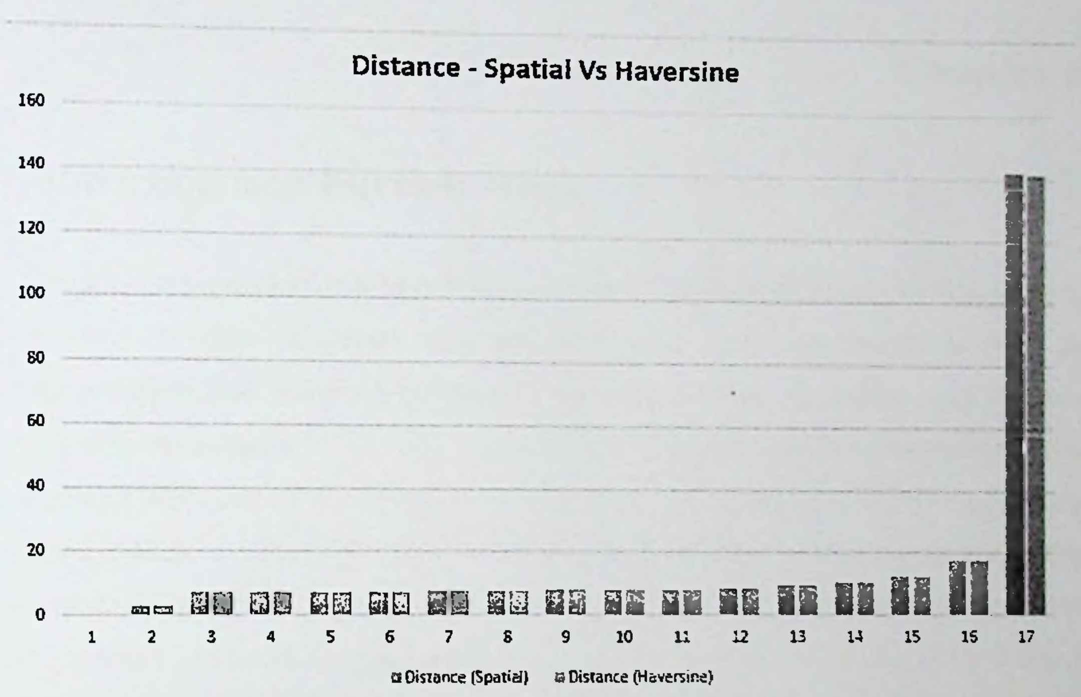


Figure 7.2: Distance comparison – Spatial vs Haversine

The results indicate that, in every scenario the values given by the MySQL Spatial function are greater compared to the Haversine results. When the distance is longer the difference also tends to increase.

Chapter 8

Conclusion and Further work

There are many solutions that have been developed to address different problem areas in carpooling systems. However, most of them have some advantages as well as disadvantages. This paper reviews and analyzes several researches and projects undertaken with regard to the route matching and passenger matching approaches. The purpose of this research is to provide an enhanced route searching and route suggesting method, which is one of the most important feature of the carpooling systems. The enhanced route searching mechanism ensures that the results are most ideal and similar to the preferred routes of the passengers. The alternative path suggesting mechanism is implemented to improve the driver's interaction with carpooling systems, which will advance the passenger matching from driver's perspective which is a new area to be considered.

As further work, applying an ideal geo-coordinate clustering method and a method to reduce the number of geo-coordinates in each route, needs to be implemented to improve the searching performance. An online payment mechanism through this carpooling application can be introduced to handle and track all the payments for rides. Also, a mobile application can be implemented to access these carpooling services through mobile devices which will improve the user satisfaction.

References

[1] S. A. Shaheen and A. P. Cohen, "Carsharing and Personal Vehicle Services: Worldwide Market Developments and Emerging Trends," *Int. J. Sustain. Transp.*, vol. 7, no. 1, pp. 5–34, Jan. 2013.

[2] M. M. Galizzi, "The economics of Car-Pooling: A survey for Europe," in *Paper for the workshop: Highways-Cost and Regulation in Europe. Universita degli Studi di Bergamo, Bergamo, Italy*, 2004.

[3] B. Srivastava, "Making Car Pooling Work–Myths and Where to Start," in *Proc. 19 ITS World Congress Semantic Cities Workshop, Vienna, Austria*, 2012.

[4] S. R. I. Sweta, M. Mounika, P. Agrawal, and G. B. Pallavi, "A Survey to Justify the Need for Carpooling."

[5] K. K. Dewan and I. Ahmad, "Carpooling: A Step To Reduce Congestion," *Eng. Lett.*, vol. 14, no. 1, 2007.

[6] D. Graziotin, "An Analysis of issues against the adoption of Dynamic Carpooling," *ArXiv Prepr. ArXiv13060361*, 2013.

[7] P. Brinckerhoff, "Greenhouse gas (ghg) and energy mitigation for the transportation sector," 2009.

[8] Y.-T. Chen and C.-H. Hsu, "Improve the Carpooling Applications with Using a Social Community Based Travel Cost Reduction Mechanism," *Int. J. Soc. Sci. Humanity*, pp. 87–91, 2013.

[9] D. Zhang, Y. Li, F. Zhang, M. Lu, Y. Liu, and T. He, "coRide: carpool service with a win-win fare model for large-scale taxicab networks," 2013, pp. 1–14.

[10] R. Manzini and A. Pareschi, "A Decision-Support System for the Car Pooling Problem," *J. Transp. Technol.*, vol. 02, no. 02, pp. 85–101, 2012.

[11] S. Di Martino, R. Galiero, C. Giorio, F. Ferrucci, and F. Sarro, *DMS 2011: proceedings : the 17th International Conference on Distributed Multimedia Systems : technical program, August 18-20, 2011, Convitto della Calza, Florence, Italy*. Skokie, IL: Knowledge Systems Institute Graduate School, 2011.

[12] J. Xia, K. M. Curtin, W. Li, and Y. Zhao, "A New Model for a Carpool Matching Service," *PloS One*, vol. 10, no. 6, p. e0129257, 2015.

[13] C. Mulders and B. Lambeau, "Carpooling, a vehicle routing approach."

[14] "PHP: Hypertext Preprocessor." [Online]. Available: http://php.net/. [Accessed: 14-Mar-2016].

[15] "MySQL." [Online]. Available: https://www.mysql.com/. [Accessed: 14-Mar-2016].

[16] "Welcome to The Apache Software Foundation!" [Online]. Available: http://www.apache.org/. [Accessed: 14-Mar-2016].

[17] "WampServer," *WampServer*. [Online]. Available: http://www.wampserver.com/en/. [Accessed: 14-Mar-2016].

[18] "Google Maps JavaScript API," *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/. [Accessed: 14-Mar-2016].

[19] "RouteBoxer Documentation: Examples." [Online]. Available: http://google-maps-utility-library-v3.googlecode.com/svn/trunk/routeboxer/docs/examples.html. [Accessed: 14-Mar-2016].

[20] "MySQL :: MySQL 5.7 Reference Manual :: 12.15.1 Spatial Function Reference." [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/spatial-function-reference.html. [Accessed: 14-Mar-2016].

[21] "Haversine formula," *Wikipedia, the free encyclopedia*. 14-Mar-2016.

[22] "Bearing (navigation)," *Wikipedia, the free encyclopedia*. 18-Feb-2016.

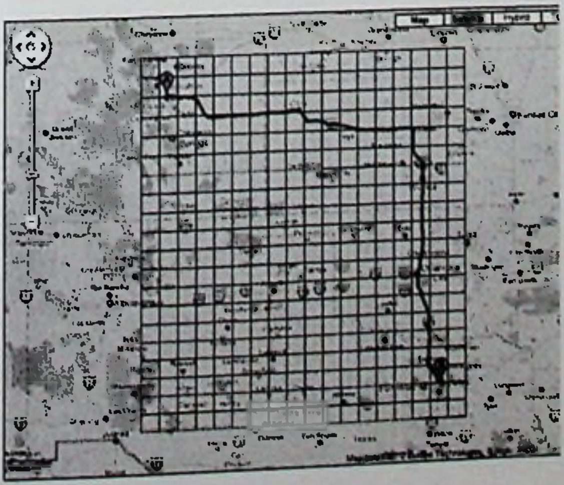
Appendix A – SQL to filter routes which are going through given two points

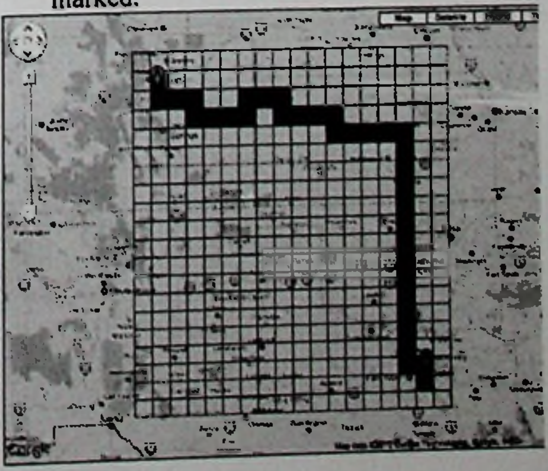
```
SELECT T1.route_id
FROM (
    SELECT distinct route_id, ( 6371 * acos( cos( radians(6.8412738) ) *
    cos( radians( X(point) ) ) * cos( radians( Y(point) ) - radians(79.9640329) ) +
    sin( radians(6.8412738) ) * sin( radians( X(point) ) ) ) ) AS distance
    FROM sp_lift_offer_points
    GROUP BY(route_id)
    HAVING distance < 20
    ORDER BY distance
) AS T1
JOIN (
    SELECT distinct route_id, ( 6371 * acos( cos( radians(6.8412738) ) *
    cos( radians( X(point) ) ) * cos( radians( Y(point) ) - radians(79.9640329) ) +
    sin( radians(6.8412738) ) * sin( radians( X(point) ) ) ) ) AS distance
    FROM sp_lift_offer_points
    GROUP BY(route_id)
    HAVING distance < 10
    ORDER BY distance
) AS T2
ON T1.route_id = T2.route_id
LIMIT 0 , 20
```

Appendix B – Steps performed by route boxer utility library

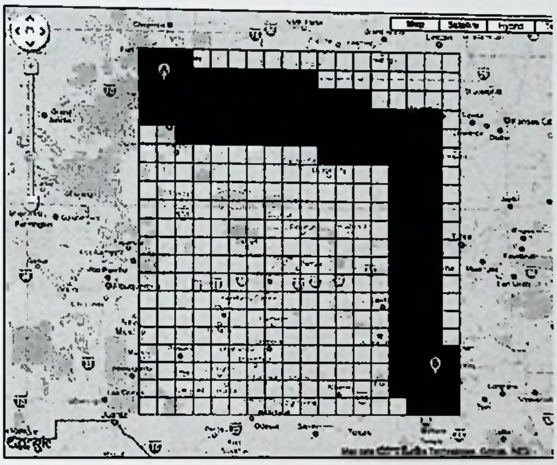
1. A grid is overlaid on the route with square cells spaced at the specified distance. The grid is centered on the bounding box of the route polyline, and extends one cell further out in each direction than is necessary to cover the entire route.

2. Every cell in the grid that the route intersects with is identified. To do this the vertices on the route polyline are traversed, and the cell containing each vertex is marked. If a cell that is marked does not share an edge with the cell for the previous vertex, the intermediate cells are also marked.

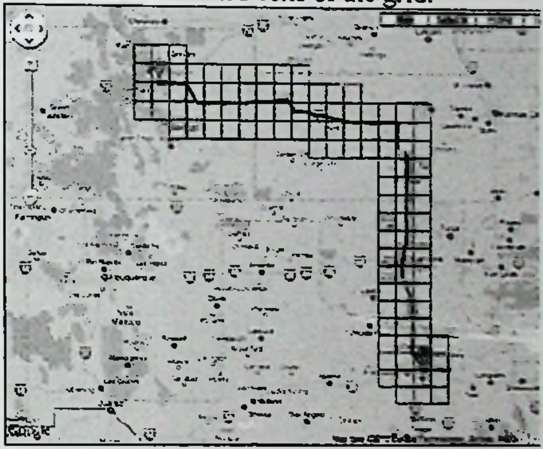




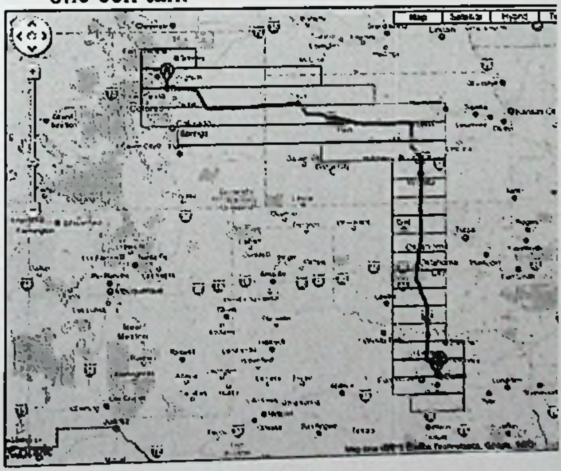
3. When a cell is marked, each of the 8 cells surrounding it are also marked.



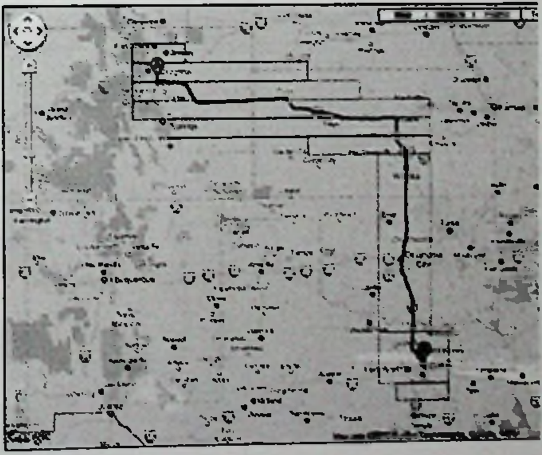
4. When all of the route vertices have been traversed, any point within the specified distance of the route is guaranteed to be in one of the marked cells of the grid.



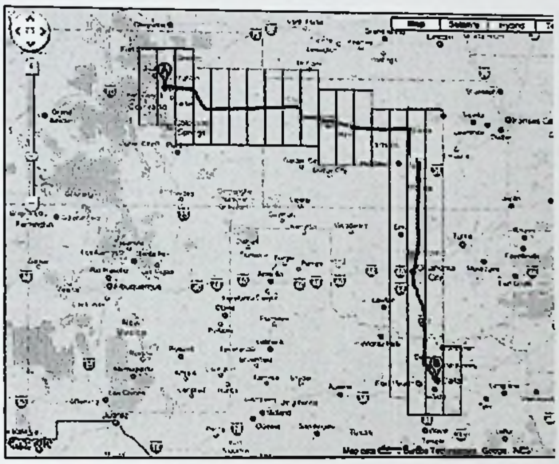
5. The marked cells are then merged into a set of non-overlapping rectangular boxes. Two different approaches are taken to this. The first approach merges cells that adjoin horizontally into a set of wide boxes, each one cell tall.



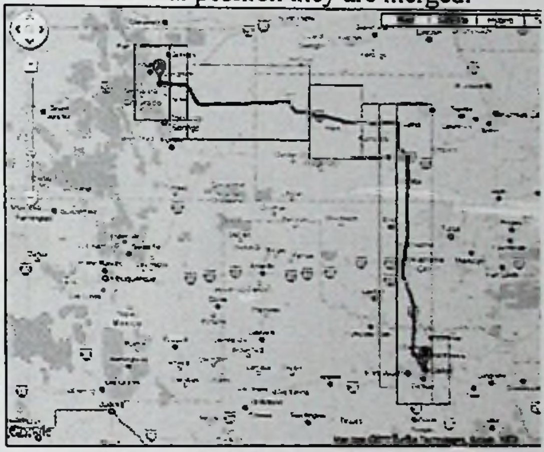
6. Each box is then compared to the boxes on the row below, and if there is a box of the same width and horizontal position they are merged.



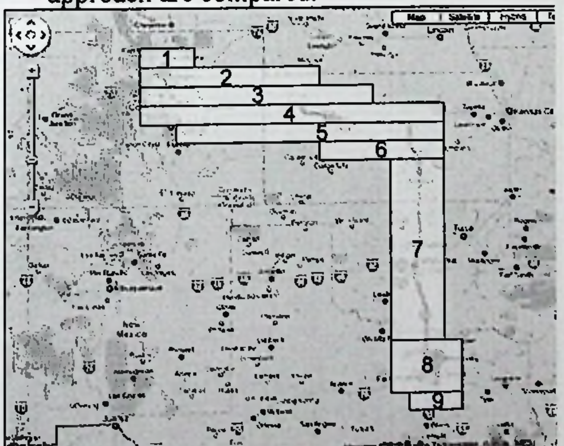
7. The second approach follows the same technique, but first merges cells vertically into a set of tall boxes, each one cell wide.



8. Each of these boxes are then compared with the boxes in the column to the left, and if there is a box with the same height and vertical position they are merged.



9. When the two approaches are complete the number of boxes that resulted from each approach are compared.



10. The set of boxes that is smallest in number are returned to the application.