A SMART CLOUD PLATFORM SERVICE FOR SOCIALIZED TRAVEL AND TRANSPORTATION WITH MOBILE SUPPORT

A Thesis

by

Yaprak Ayazoğlu Yazıcı

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master's Thesis

in the
Department of Computer Engineering

Özyeğin University
September 2013

A SMART CLOUD PLATFORM SERVICE FOR SOCIALIZED TRAVEL AND TRANSPORTATION WITH MOBILE SUPPORT

Approved by:

Professor İsmail Arı, Advisor
Department of Computer Science
Özyeğin University

Professor Barış Aktemur
Department of Computer Science
Özyeğin University

Professor Enis Kayış
Department of Industrial Engineering
Özyeğin University

Date Approved: 15 August 2013

To My Dearest Family, Science and To the Force...

ABSTRACT

It is now clear that social networking services are evolving towards mobile web applications and continuous location sharing is also becoming a trend. In this evolution, we believe that the next step will be complete and continuous route and experience sharing. A route-based social networking cloud platform is a service in which users share their travel routes as well as experiences and stories using their mobile devices, or search for and get matched with people with similar travel and transportation habits in real-time. This work mainly focuses on the development of an end-to-end route-based social networking cloud platform service and its client-side mobile application. First, we describe the design and implementation of our real-time taxi ride-sharing service for metropolitan areas, which is a specific application of the route-sharing service. This system is composed of 1) taxi location data collection system, 2) real-time passenger coupling service and 3) a dashboard for visualization and analysis of the results. Second, we describe our location tracking mobile application that captures rich location-based information and saves it to the cloud.

ÖZETÇE

Günümüzde sosyal paylaşım servisleri mobil cihaz kullanımını ön plana çıkartmaktadır ve kullanıcıların da konum bazlı içerik paylaşımları giderek yaygınlaşmaktadır. Bu değişim sürecinde bir sonraki adım yaşadığı deneyimleri bütün bir yol hikayesi şeklinde paylaşımına olanak sağlayan servisler olacaktır. Rota tabanlı sosyal paylaşım hizmeti, kullanıcıların mobil cihazlarını kullanarak gerçek zamanlı olarak deneyimlerini gerek belli başlı noktalar, gerekse bütün olarak ifade edebilmelerini ve ortak gezi zevklerine, ulaşım güzergahlarına sahip diğer insanlarla tanıştırılmasını ve eşleştirilmesini sağlayacak bir hizmettir. Bu çalışmada uçtan uca rota tabanlı sosyal paylaşım hizmetini bulut üzerinde gerçeklenecek ve bu hizmete uygun bir mobil uygulama geliştirilecektir. Birinci olarak, gerçek zamanlı taksi paylaşımı servisi tanımlanacaktır. Bu sistem 1) Taksi jeolokasyon verisi toplama birimi, 2) gerçek zamanlı yolcu eşleme sistemi, 3) yolcu eşleştirme ve birleştirme algoritmalarının görsel ve sonuç analizi için web tabanlı panel sisteminden oluşmaktadır. İkinci olarak, takip edilen güzergahın gerçek zamanlı olarak bulut üzerinde kaydedilmesini sağlayan mobil uygulamadan bahsedilecektir.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank to my thesis supervisor, Asst. Prof. İsmail Arı for the valuable guidance and advice. His kind willingless to teach, all his supports, comments, deepest knowledge and all the generous time he spent for my work helped me to improve my knowledge. My master's study and thesis wouldn't be completed without his support and help.

I would like to thank to all of my instructors in Özyeğin University for giving me the chance to improve my technical knowledge about Computer Science. Further, their special attention to teach the current problems of our age and giving insights for the future is invaluable. I appreciate the support of my thesis committee members Prof. Barış Aktemur, Prof. Murat Şensoy and Prof. Enis Kayış who helped me to improve the quality of this thesis.

I would also like to thank to all my friends at Özyeğin University. Especially with Uğur Koçak, Erdi Ölmezoğulları, Ali Arsal, Kıvanç Çakmak, I had great time with them both at university: at lunch, coffee breaks; and out the university: new year celebrations, bicycle trips with all, long and adventurous motorcycle trips with Uğur Koçak. All the moments with my friends are unforgetable.

I would also like mention my gratitude to my family. I really appreciate the support and help of my dearest husband Volkan Yazıcı. His kind toleration and love always be the driving force. I would like to thank to my mom and sister for encouragement to reach my best and always being there where I need help.

This project has been sponsored in part by Turkish Telecom R&D Program, EU-FP7 Marie Curie BI4MASSES project, and IBM SUR (Shared University Research) award. I thank all our sponsors for their continued support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The top 10 social networking sites as of March 2012 are listed in Table 1. The striking fact is that these social platforms have attached 12-750+ Million users in a relatively short time, which shows their popularity and the need for such services. While, many of these cloud web sites also have accompanying mobile applications, none of them have end-to-end route-based travel experience sharing features. Some sites have only recently added pin-point location publishing support especially for mobile devices, which is also shown in Table 1.

The first motivation of our project is to reach tech-savvy traveling crowds by adding the mentioned route-based socialization features and potentially generate a new revenue stream for all. The second motivation is to contribute to the new wave of "location-based social networking" trends currently set by Foursquare.com and alike. Note that, even Foursquare currently does not have a continuous route-based experience sharing feature. Combination of mobile-driven information, backed by scalable Cloud Infrastructure (IaaS) and Platform Services (PaaS) and recommendation systems constitutes the core of our work as well as Web 3.0.

The Geographic Information Systems (GIS) date back to 1854. With rapid growth of the web, users have begun to use GIS over the Internet which led to standardization of geography-related data and message transfer formats. Geography Markup Language (GML), which was developed in 2007, is the most commonly used modeling language for geographic objects and a standard for geographic transactions on the Internet. When it comes to route-sharing alone, GML is sufficient. However, our project also aims to include text and media content generated or collected over routes,

Table 1: Top 10 social networking sites based on user count [1].

Rank	**Name**	**Users**	**Location-Support**
1	Facebook	750 M	Yes
2	Twitter	250 M	Yes
3	LinkedIn	110 M	No
4	Pinterest	85.5 M	Yes
5	MySpace	70.5 M	Yes
6	Google Plus+	65 M	Yes
7	DeviantArt	25.5 M	No
8	LiveJournal	20.5 M	Yes
9	Tagged	19.5 M	Yes
10	Orkut	17.5 M	Yes

for which GML is inadequate. When "experience sharing" is concerned, yelp.com, forums, *etc.* are suitable web sites and services. Forums contain unstructured travel-related data and websites similar to yelp.com usually contain experiences without any geographic markup support. The closest project that allows us to *define routes with user content* is Open Street Map[1], however there is no travel or transportation web service that implements the Open Street Map and can serve real-time route-based queries. This thesis is a step towards this ideal goal.

[1] http://www.ebizmba.com/articles/social-networking-websites

CHAPTER II

MOBILE RIDE-SHARING

This thesis addresses the technical issues that hinder massive adoption of emerging mobile ride-sharing applications used in metropolitan areas. There can be variety of spatio-temporal passenger matching applications ranging from short-term, ad-hoc taxi ride-sharing to long-term carpooling. Yet, the most challenging technical issues occur for the city-wide, massive-scale and dynamic passenger coupling scenarios. To analyze and address these issues, we created a metropolitan taxi ride-sharing scenario involving 1000s of vehicles traveling in the city of Istanbul by making real web service calls to the Google Directions service. We describe three passenger matching algorithms called *Source-based, Source-and-Destination-based* and *OnRoute-based* matching and compare their performances using occupancy rates, total travel distances, travel time, monetary savings and effects on traffic. We defined two new success metrics for occupancy rates and called them *transportation throughput* and *goodput*. The results show that ∼30-50% improvement in occupancy rates and monetary savings are possible at the same time. We hope that our findings will encourage people (drivers and passengers equally) to adopt mobile application and cloud services for ride-sharing, thus removing a substantial number of private cars from the traffic. Our findings clearly demonstrate the power and social impacts of mobile, ubiquitous and cloud computing technologies when used in harmony.

2.1 Motivations for Ride-sharing

As of 2013, there are approximately 4-6 million registered vehicles in the largest metropolitan areas of the world including Los Angeles, Istanbul, Moscow, and Beijing and this number increases by ∼10% each year. Adding the fact that the population

of these cities are around 12-20 million, but vehicle occupancy rates are only ∼1.5 seats/car, the traffic congestions are inevitable. While different modes of mass transportation have the biggest share in providing mobility in these cities, ride-sharing has a bigger potential for reducing the traffic congestion problems as it directly replaces existing vehicles from the traffic or avoids new ones to enter. Mass transportation is usually not convenient for several reasons including crowdedness, spatial inconvenience (*i.e.* farness to origins and destinations of intended travel), temporal inconvenience (*i.e.* unavailability when needed; possibility of long waits). Bad weather conditions exacarbate all of these problems. Ride-sharing applications are gaining momentum due to proliferation of smart mobile devices [2][3]. However, they have not been deployed in massive scales yet despite their potential, due to computational complexities and other social issues.

In this chapter we compare three passenger matching strategies called *Source-based Matching (SM), Source-and-Destination-based Matching (SDM)* and *OnRoute-based Mathcing (ORM)* in order to:

- reduce the number of cars in traffic,
- save money and time to ride-sharing passengers,
- increase taxi fill rates and income of the taxi drivers, and
- save gas and energy for the overall society.

Unless otherwise stated, the word taxi may also refer to private vehicles throughout this thesis.

2.2 Design of Taxi Ride-sharing Scenarios

Our mobile ride-sharing service that depends on continuous route-sharing has two parts:

- the smart mobile application, and
- the backend cloud platform service.

Passengers are assumed to have smart mobile devices (Android, iPhones, iPads, *etc.*) with networking and global positioning system (GPS) capabilities. The mobile ride-sharing applications installed inside allow them to make taxi requests from anywhere, anytime. The request will have information about the pickup location, destination location and possibly deadlines for pickup and drop-off times. Additional personal profile information on ride-sharing preferences can also be added. Note that we assume continuous mobile networking ability (*i.e.* access to a 2G-3G-4G or a WiMax line) throughout the trip so that the passenger matching and fare calculations can be done accurately.

Taxis also need to continuously and accurately feed their geospatial locations to the cloud service, so that best matches can be made. This information is stored in a database (or in our case MongoDB NoSQL engine [4]) with spatial query support at the cloud service side. Spatial (*i.e.* location) indexing techniques for fast K-Nearest Neighbor (KNN) queries include R-trees[5][6], kd-trees[7], and Voronoi[8] diagrams. MongoDB's proximity query($near) uses R-tree.

Since KNN query support is a well-plowed research area and is currently available inside the database engines, we do not study this topic in further detail. The cloud service collects these taxi requests and primarily matches people with the closest taxis. It informs both parties about the locations, duration of travel, fare, *etc*, details of which are described in the following sections. More importantly, the service continuously checks the progress of the trip and groups (matches & merges) people that will be traveling over similar routes. Similarity of trips is measured by the proximity of origination, destination or on-route points by different strategies. A fare is also calculated by the service for each passenger such that everyone can agree on it without a dispute (since they already accepted to use the service and are aware of

the savings).

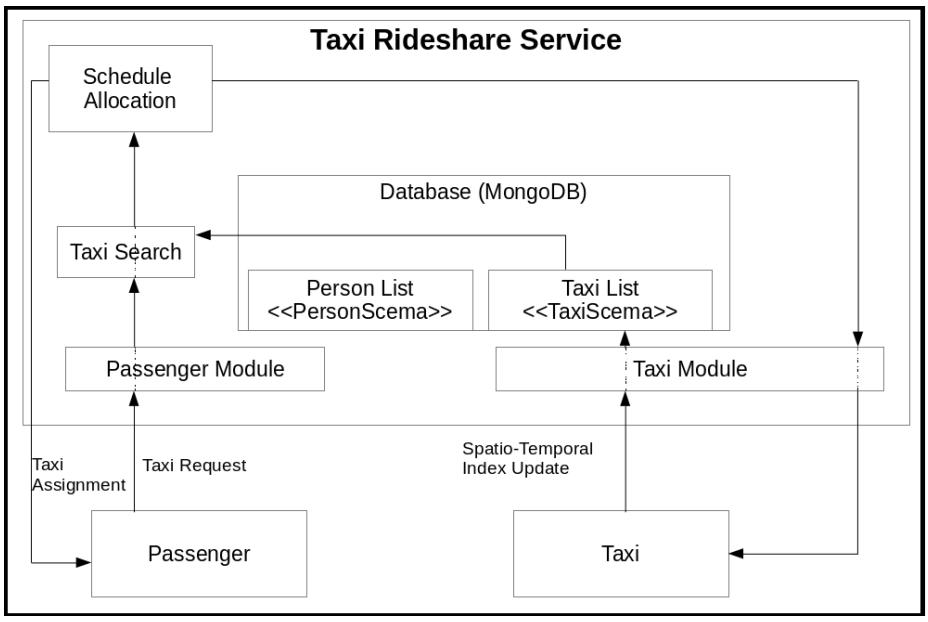

Figure 1: System design and components for taxi ride-sharing

Figure 1 shows a simplified operational workflow of the system. There are two types of users and respective workflows: taxis and passengers. Passenger flow has two steps: dial-a-taxi (inform taxi) and get information about assigned taxi (license plate, estimated arrival time, *etc.*). Taxis with mobile support feed location (and possibly occupancy) information to the cloud and they do not explicitly search for passengers.

2.2.1 Taxis With and Without Mobile Support

The taxi drivers are assumed (by default) to have the application installed on their mobile phones or vehicular tablets. There are several advantages to this. Having the mobile application allows taxi drivers to:

- Be called (*i.e.* dial-a-ride) from anywhere by the application's users,
- See the profile of users and select accordingly,
- Get involved in fare calculations,
- Create their own profiles, and

- Collect points, recommendations, *etc.*, to create a professional (ongoing) service (with past memory).

What happens if the taxi drivers do not have smart phones or these applications installed? Can we still bootstrap this system? The answer is, yes. Until drivers obtain (or choose to obtain) the application, the first passengers (also the early adopters of the mobile application) will act as the proxy and initiator for picking other requests by passengers.

The possible constraints on real-time matching may include the maximum count or limit passengers may place (which is a fixed-static profile metric) on the number of ride-sharing people or a time deadline on the pickup or drop off times. The location information is encoded with geohashing[9][10] technique, where the latitude-longitude pairs are turned into a single globally unique identifier(GUID) via a hashing function. Geo-hashing makes it efficient to search nearest neighbors by storing multiple geo-locations as a single B-Tree. Spatial databases primarily use R-trees for spatial queries.

CHAPTER III

EVALUATION METHODOLOGY AND METRICS

We define two new metrics called throughput and goodput for measuring the success of matching strategies given in Section 4. Our transportation metrics throughput and goodput are analogous to their computer network counterparts. Goodput defines the average number of passengers delivered with route-distance overheads removed. Therefore, it is both more passenger-friendly and a stricter transport efficiency metric than throughput. Throughput defines average number of passengers (i.e. similar to network packets) that are moved from their sources to their destinations. Since route-distance overheads may be involved in this metric, it can be viewed as driver-friendly, since drivers still make money for extra travel. Figure 2 will be used to show how transportation success metrics are calculated by example in the following section. In this figure, a simplified set of routes are defined to describe some ride-sharing scenarios.

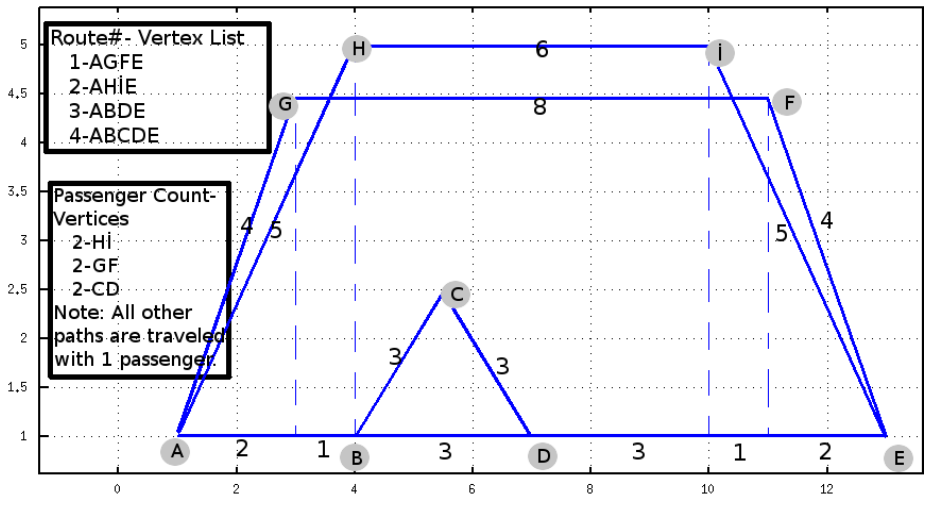

Figure 2: Sample routes used for throughput, goodput metric descriptions.

3.1 Throughput and Goodput

The formulation of **throughput metric** is defined in equation (1). First a personKM is the value calculated by the multiplication of the total distance (km) traveled by the number of people that traveled this distance at every part (pickup-drop off) of the travel.

$$Throughput = \frac{\sum personKM_i}{\sum distance_i} \tag{1}$$

For example, assume that a taxi will pick up a passenger from location A and travel through $|ABDE|$ route to drop this passenger at point E (Figure 2). The length of the route $|ABDE|$ is 12 kilometers. When a single passenger travels alone calculation is as follows,

$$\begin{aligned}
PersonKM &= \#person * |ABDE|(km) && \text{Calculate PersonKM (pkm)} \\
PersonKM &= 1person * 12km && \text{Substitude numbers} \\
&= 12pkm \\
Throughput &= \frac{\sum personKM_i}{\sum distance_i} && \text{Calculate Throughput} \\
Throughput &= \frac{12pkm}{12km} && \text{Substitute numbers} \\
&= 1p
\end{aligned}$$

Suppose that this same route is traveled by two people, then throughput would be,

$$\begin{aligned}
Throughput &= \frac{\sum personKM_i}{\sum distance_i} && \text{Calculate Throughput} \\
Throughput &= \frac{2person * 12km}{12km} && \text{Substitute numbers} \\
&= 2p
\end{aligned}$$

Lets analyze another scenario: taxi picked a passenger from point A, the destination of this passenger is E. When taxi is at point A and picked the first passenger, another passenger(2nd passenger) requested for a taxi from point G to F. Initially route was

$[ABDE]$, after this second request the final route is $[AGFE]$. For this route, the throughput can be calculated as follows,

$$
\begin{aligned}
PersonKM &= \#person * |AG| + \#person * |GF| + \#person * |FE| \\
PersonKM &= 1person * 4km + 2person * 8km + 1person * 4km \\
&= 24pkm \\
Throughput &= \frac{\sum personKM_i}{\sum distance_i} \\
Throughput &= \frac{24pkm}{16km} \\
&= 1.5p
\end{aligned}
$$

If the 2nd passenger were to travel on $[HI]$, the new route would have been $[AHIE]$ and this would affect the calculations as follows,

$$
\begin{aligned}
PersonKM &= \#person * |AH| + \#person * |HI| + \#person * |IE \\
PersonKM &= 1person * 5km + 2person * 6km + 1person * 5km \\
&= 22pkm \\
Throughput &= \frac{\sum personKM_i}{\sum distance_i} \\
Throughput &= \frac{22pkm}{16km} \\
&= 1.4p
\end{aligned}
$$

If the 2nd passenger were traveling from point C to D ($[CD]$), then the throughput

will be,

$$
\begin{aligned}
PersonKM &= \#person * |AC| + \#person * |CD| + \#person * |DE| \\
PersonKM &= 1person * 6km + 2person * 3km + 1person * 6km \\
&= 18pkm \\
Throughput &= \frac{\sum personKM_i}{\sum distance_i} \\
Throughput &= \frac{18pkm}{15km} \\
&= 1.2p
\end{aligned}
$$

Note that, for 2 passengers, maximum throughput is 2 and 1 is the minimum throughput. Further, in this metric the extra distances traveled benefits to the taxi driver more, therefore it is more "driver-friendly" where as the goodput metric is more "passenger-friendly" as shown next.

Goodput metric minimizes the distance traveled for transporting multiple passengers, thus it also saves gas. The formula for goodput is defined at Equation 2.

$$
Goodput = \frac{\sum originalDistance_i}{\sum distance_i} \tag{2}
$$

If we re-calculate the scenarios defined at the begining, For **scenario 1** (1 passenger travels the $[ABDE]$ path.),

$$
\begin{aligned}
TotalKM &= |ABDE| = 12km \\
OriginalDistances &= passenger1.dist \\
&= 12km \\
Goodput &= \frac{\sum originalDistance_i}{\sum distance_i} \\
&= \frac{12km}{12km} = 1
\end{aligned}
$$

For **scenario 2** (2 passengers want to go from A to E)

$$
\begin{aligned}
TotalKM &= |ABDE| = 12km \\
OriginalDistances &= passenger1.dist + passenger2.dist \\
&= 12 + 12 = 24km \\
Goodput &= \frac{\sum originalDistance_i}{\sum distance_i} \\
&= \frac{24km}{12km} = 2
\end{aligned}
$$

For **scenario 3** (Passenger 1 wants to go from A to E, Passenger 2 wants to go from G to F)

$$
\begin{aligned}
TotalKM &= |AGFE| = 16km \\
OriginalDistances &= passenger1.dist + passenger2.dist \\
&= 12km + 8km = 20km \\
Goodput &= \frac{\sum originalDistance_i}{\sum distance_i} \\
&= \frac{20km}{16km} = 1.25
\end{aligned}
$$

We skip calculation of goodput for scenarios 4-5 and conclude with the Table 2 that compares the throughput and goodput of given routes. According to this table, route 5 has the worst goodput since it diverts significantly from its original route to deliver 2nd passenger to a short distance. Goodput values are always smaller than or equal to the throughput values, as goodput denotes a more conservative utility. Note that, the first scenario is the calculation for the 1-passenger trip.

3.2 Money Calculation

Our goal is to propose a practical taxi ride-sharing service, thus in our work we also define a practical pricing model that considers the benefits of both riders and taxi drivers. In this model shown in Algorithm 1, 1) the initial fee is paid by both

Table 2: Throughput and Goodput for the Scenarios Described in Section 3.1

Scenario#	**Throughput**	**Goodput**
1	1	1
2	2	2
3	1.5	1.25
4	1.4	1.125
5	1.2	1

passengers, 2) fee is incremented every 100 meters, 3) the total fare is proportionally divided among passengers based on their travel distances.

Algorithm 1 calculateMoney()

1: $Total fee = incremental fee * (distance(m)/100m) + (initial fee * \#passengers)$

CHAPTER IV

PASSENGER MATCHING AND MERGING

4.1 Passenger Matching Strategies

In this section, we describe our passenger matching strategies.

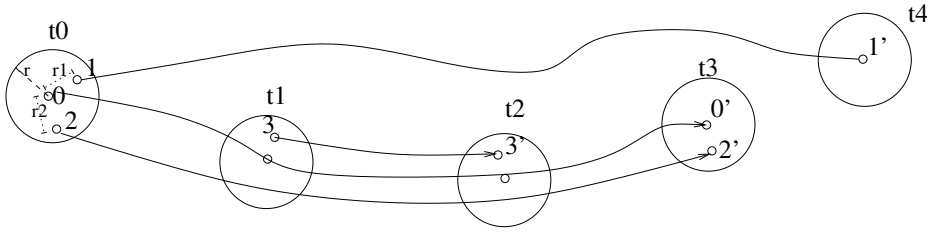

Figure 3: Routes for 4 passengers (0-4) and their movements in time (t0-t4). r = 600m, r1 = 230m, r2 = 350m.

In Figure 3, assuming $t_0 < t_1 < t_2 < t_3$, we show movements of four passengers 0,1,2, and 3 at different times t_0, t_1, t_2 and t_3. At t_0, passenger 0 found a taxi and he/she wants to reach 0'. The approximate time to reach 0' is t_3. At this same moment(t_0), passenger 1 is makes a taxi request to transport him/her to 1' and passenger 2 calls for a taxi to the destination point of 2'. At t_1, passenger 3 asks for a taxi to destination point 3'. Taxi starts its journey from 0 to 0' in 3 **steps**. Path covered between $[t_0, t_1]$ is step0, between $[t_1, t_2]$ is step1, between $[t_2, t_3]$ is step2. Passenger 0 starts the journey alone in the taxi. At t_0 passengers 1 and 2 also make a taxi call. We assume that these emerging requests can actually represent either calls for taxis or attempts to ride private vehicles. Depending on the strategy to be selected, passenger grouping decisions may change as described next.

4.1.1 Source-Based Matching (SM)

Algorithm 2 gives the psuedo-code for SM strategy that continuously makes nearest passenger or nearest neighbor(NN) queries to the database with the current location

of the moving vehicle at the center and a radius parameter (*e.g.* r = 600 m) as shown in Figure 3. At t_0, when passengers 1 and 2 made their taxi calls and got into a request queue, the algorithm sorts the returned lists in the increasing order of their distances to the current location. For instance, in Figure 3, since r1 (distance of passenger 1 to passenger 0) is smaller than r2 (distance of passenger 2 to passenger 0) the Euclidian distance-sorted list becomes $< passenger1, passenger2 >$, in order. The vehicles that are already carrying passengers are not considered as candidates for merging and are removed from this list. As this strategy only considers source proximity, it is prone to make large errors when the destinations of the two passengers are far apart.

Algorithm 2 srcMatch(taxi t, queryRadius r)

1: $init : Lsrc = \emptyset, sortedList = \emptyset$

2: $Lsrc = queryNearPassengers(t.currentLocation, r)$

3: {Sort in ascending order of Euclidean distance}

4: $sortedList = sort(Lsrc)$

5: **return** *sortedList*

4.1.2 Source & Destination-Based Matching (SDM)

SDM strategy can be considered an extension to the SM strategy. As shown in Algorithm 3, this strategy also continuously makes NN queries to the database with the current location of the moving vehicle at the center and a radius parameter to find the closest source-based matching candidates. Additionally, it makes an NN query based on the destination of the current taxi and destinations of all "emerging" requests of passengers for vehicles to obtain a destination-based candidate list. Next, it takes the intersection of the two lists to find passengers that are close both source and destination-wise. Vehicles that are already carrying passengers are eliminated from the candidate list again. Intersection list is sorted in the ascending order of

both source and destinaton based Euclidean distances (*i.e.* sum of distances) to the current taxi.

Algorithm 3 srcDstMatch(taxi t, queryRadius r)

1: $init : Lsrc = \emptyset, Ldst = \emptyset, sortedList = \emptyset, Lsrcdst = \emptyset$

2: $Lsrc = queryNearPassengers(t.currentLocation, r)$

3: $Ldst = queryNearPassengers(t.destination, r)$

4: $Lsrcdst = Lsrc \cap Ldst$

5: {Sort in the ascending order of sum of src+dst Euclidean distances}

6: $sortedList = sort(Lsrcdst)$

7: **return** *sortedList*

4.1.3 On Route-Based Matching (ORM)

In some scenarios, the destination of the second candidate vehicle for merging may not be within the proximity radius of the destination of the current vehicle. However, it may be on the route of the first vehicle while it is traveling towards its destination. ORM strategy searches for all candidate vehicles with destinations in the radius proximity of the travel path of the current taxi as shown in Algorithm 4. The algorithm can also be extended (for every source-based candidate) to check if the destination of the first passenger is on the route of the second candidate passenger, so that the case of partially-overlapping routes is also covered. Note that if there are many candidates, then the cost of these calculations can be high.

Algorithm 4 onRouteMatch(taxi t, queryRadius r)

1: $init : Lsrc = \emptyset, LonRoute = \emptyset, sortedList = \emptyset, Lsrcdst = \emptyset$

2: $Lsrc = queryNearPassengers(t.currentLocation, r)$

3: **for all** $step : t.taxiRoute$ **do**

4: LonRoute = LonRoute ∪ queryNearPassengers(step.startLocation, r)

5: $Lsrc\_OnRoute = Lsrc \cap LonRoute$

6: $sortedList = sort(Lsrc\_OnRoute)$

7: **return** $sortedList$

4.2 Decision to Merge or Not? (After Matching)

At the initial attempts of our taxi ride-sharing simulations we found that the utility (see Section 3.1 for definition of goodput) of the ride-sharing could be lower than 1 when non-ideal matches were made. To improve the utility an additional merging step was added. The major criteria is to merge the routes that together have a goodput greater than 1. As an additional second step, the route that maximizes the utility among all will be selected. With 2 passengers, there are 4 ways to define a merged route as shown in Figure 4. In the first two cases, origin "a" and destination "d" is constant and [c,b] in between can permute. Thus, two possible schedules are listed in item 1 and 2 below. In list item 3,4 the origin "a" and destination "b" is constant and [c,d] can permute.

1. a,c,b,d
2. a,b,c,d
3. a,c,d,b
4. a,d,c,b Eliminated

First of all, Route 4 is **eliminated** from the begining, since it requires 2nd passenger to be dropped off before she/he were picked up, which is not logically possible. For

each of those 3 remaining alternatives, goodput value will be calculated and the best merged route will be scheduled.

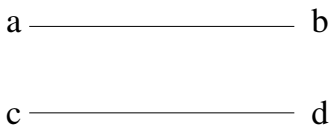

Figure 4: 2 sample routes. a-b is the route of first passenger, c-d is the route of second passenger.

CHAPTER V

IMPLEMENTATION

5.1 Route Generation

Data set to run tests are collected by using Google Directions service. In Google Directions Service input for a route is two $< lat, long >$ pairs for source and destination, respectively. Output is a complete route that is optimized by Lazy Shortest Path Strategy. We first took the big rectangular area denoted with north-west(41.434, 28.410) and south-east (40.784, 29.660) which covers the city of Istanbul.Next we randomly picked two "start" and "end" locations inside this box. With this method, most of the output routes were somewhat unreasonable from the perspective of anyone who knows the city well,*e.g.* you would not travel from Zekeriyaköy to Şile by taxi as it would be too costly. Initially, we used the well-known Voronoi diagrams to generate

Figure 5: Istanbul regions generated using Voronoi diagrams.

such zoning and we used the QuichHull[11] library to generate Voronoi regions by giving it a set of landmarks. However, since the zones were automatically generated (with no topological concerns) many of them were overlaid on top of the sea as can be seen in Figure 5. Routes that originated from or were destined to a random location on the sea returned faulty results from Google Directions service. Thus, we decided to tune the zones generated by hand. Figure 6 shows the map of Istanbul with 24 overlaid zones defined based on population density. To define a realistic origin point

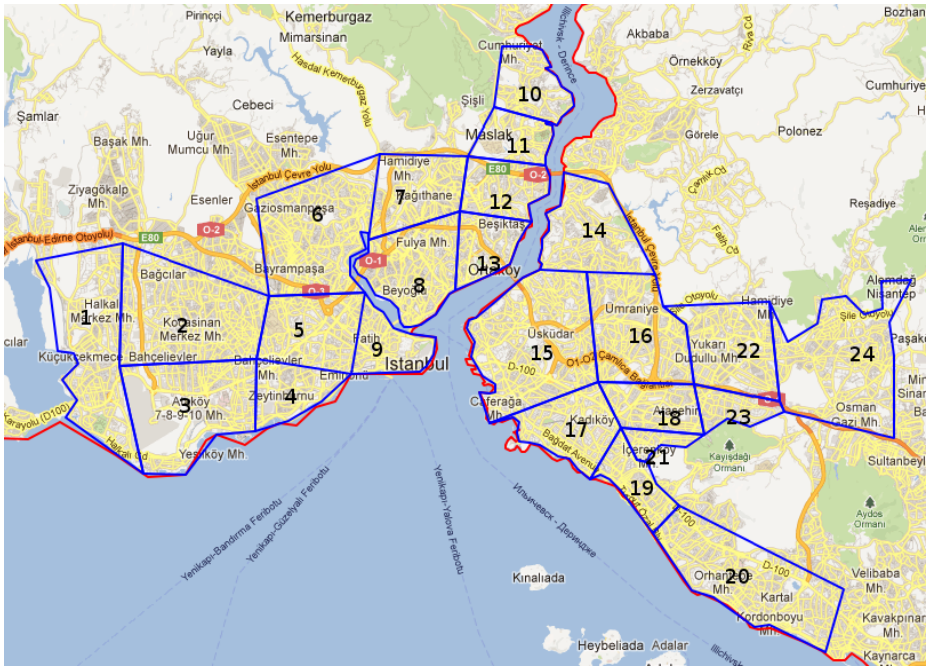

Figure 6: Istanbul regions generated manually

for a passenger, a random lat-long pair will be selected in a randomly chosen region (among the 24 zones). Next we needed to find a realistic way to decide on the possible destination points. For instance, the probability to traveling from region 1 to 1 and from region 1 to 24 shouldn't be equal. Thus, inter-regional probabilities were determined to define the likelihood of general passenger travel habits. Table 3 shows the matrix for probability of travel between two selected zones. For the total table please refer to the Table 4 at Appendix A. For instance, intra-zone travels are the most probable and trips to neighbor zones occur relatively more infrequently. Table

3 can be read as follows: The probability of a taxi to travel inside region 3 is 100/236, whereas it is 20/236 for a travel from region 3 to 4. Additional biasing may include landmarks of extreme popularity. For example, there is an airport in region 3 and the probability of travel to this region is higher. Regions 8-9-12-13 have touristic attractions, universities, famous entertainment and food locations which increases probability of travel to these place. In some metropolitan areas, taxis (or minibuses) may be constraint to operate in certain zones or routes only (that they paid for). We currently did not place such constraints, but will discuss such issues later.

Table 3: Inter-region travel probabilities between the regions 15 to 18

Region#	**15**	**16**	**17**	**18**
15	0.75	0.07	0.07	0.04
16	0.04	0.88	0.02	0.01
17	0.04	0.01	0.81	0.01
18	0.01	0.01	0.08	0.80

Defining all the building blocks in the data generation process, the main procedure can be summarized as follows:

Algorithm 5 routeGeneration

1: Pick a random region number and a random geo-location in that region.
2: Based on the biases randomly pick a destination region and a random geo-location in that region.
3: Make a Google Directions web service call using the start and end geo-points.

There are nearly 18,000 registered taxis in İstanbul. Thus, using the algorithm above, more than 20,000 taxi routes are collected from Google Directions Service. However, due to 2500 query limit Google Directions Service, tests with 20,000 routes takes days to finish. Thus, a 3000 subset of routes are extracted out of 20,000 routes. The origin and destination points of those 3000 routes are selected from regions 15 to 18. To check the validity of the generated taxi routes, distance and duration

histograms of 20.000 routes are plotted in Figure 7. This same histogram is also plotted for 3000 routes in Figure 8. As observed in those figures, the average duration for a route is around 10mins and average distance is around 4km.

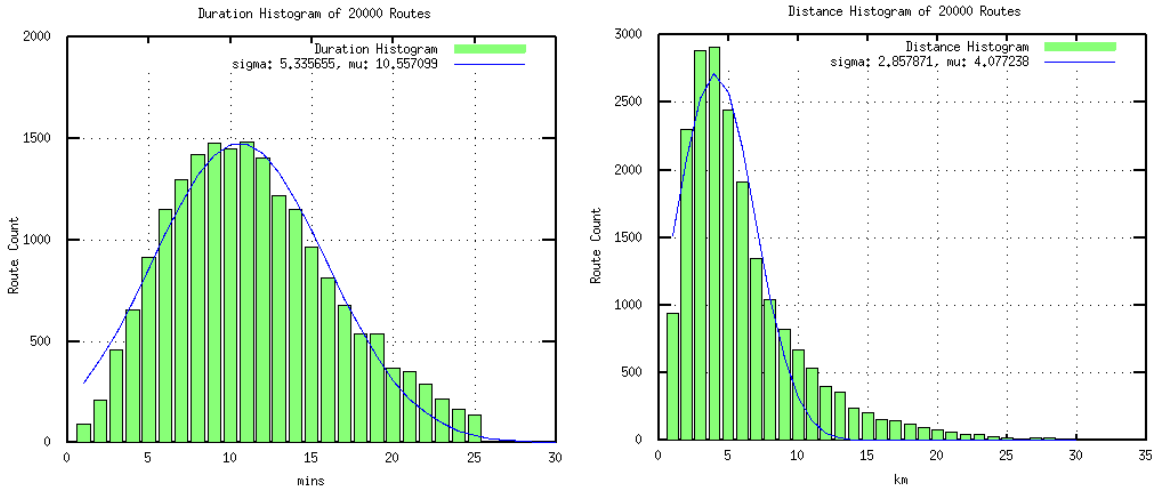

Figure 7: Distance and duration histogram for 20000 Routes

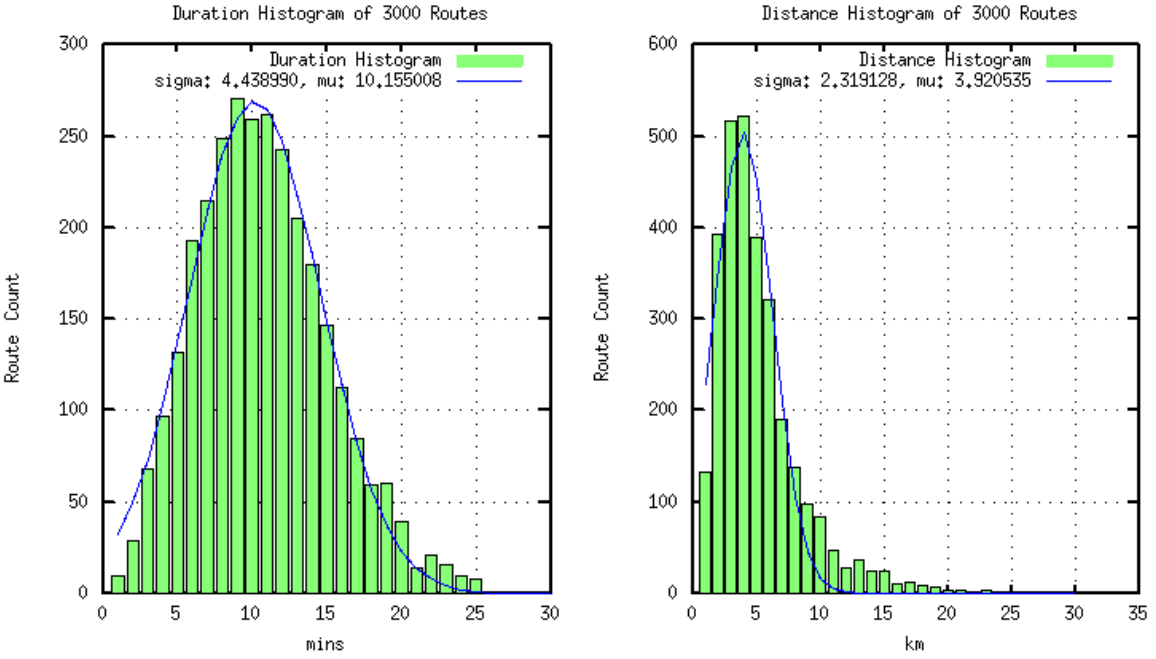

Figure 8: Distance and duration histogram for 3000 Routes

For those plots, Gaussian fitting is calculated using the formula given in equation 3 where the parameters μ and σ denote the mean and variance of the distribution.

We find that for both 20000 and 3000 taxis, route durations (10.5 ± 5.5 minutes) and the travel distances (4 ± 2.5 km) are quite similar. Therefore, the system seems to be self-similar in nature.

$$d(x) = c1 * exp(-\mu/\sigma^2) \tag{3}$$

With this taxi route set, we generated two large-scale simulation scenarios. The first scenario consists of 20,000 routes all over Istanbul (24 regions in Figure 6) and in the second one, the 3000 routes that reside in region [15-16]. Due to 2500 daily query limit to Google servers, most experiments were run on 3000 route set.

5.2 Simulation Model

The next step is to develop a simulation setup based on the raw data that is collected with the procedure defined in Section 5.1. Initially, the route data simply consists of an array of GeoPoints, a data structure stores the coordinate of a location in $< lattitute, longitude >$ format. In the simulation model, this route will be transformed into a taxi journey data by injecting time stamps and passenger information. First of all, all taxis will be started in 60 seconds. Thus, the initial time stamp will be randomly defined between 0 and 59 seconds. The duration and distance of rest of the steps is already defined in data that is collected from Google Directions Service. Thus, updates are done just by adding the duration of the step by the starting time stamp. The basic attributes of a raw route data shall be defined as shown below. Since this route data is stored inside the MongoDB NoSQL engine [4], its format was intentionally selected as JSON (JavaScript Object Notation), which is the native/default format for MongoDB.

```
{"route": {
  "steps": {
    [ { "GeoPoint": {"lat":40.9938,"lng":29.1028} },
        { "GeoPoint": {"lat":40.9943,"lng":29.1034} },
        { "GeoPoint": {"lat":40.9947,"lng":29.1040} },
        { "GeoPoint": {"lat":40.9939,"lng":29.1067} },
        { "GeoPoint": {"lat":40.9939,"lng":29.1069} },
        { "GeoPoint": {"lat":40.9935,"lng":29.1081} },
```

```
      { "GeoPoint": {"lat":40.9934,"lng":29.1086} }
    ]
  }
}}
```

The taxi simulator software is used to generate a taxi object out of this raw route data and inject time and passenger values at the raw JSON format. The resulting document will be as follows:

```
{taxi: {"route": {
    "utid" : "a4e68ad4-1f35-4c5a-a3e7-90fcc21df66d",
    "curLoc":{"GeoPoint":{"lat":40.9938,"lng":29.1028}},
    "steps": {
      [ {"GeoPoint":{"lat":40.9938,"lng":29.1028}, "startTimeInMillis":30000, "passenger": 1 },
        {"GeoPoint":{"lat":40.9943,"lng":29.1034}, "startTimeInMillis":47796, "passenger": 1 },
        {"GeoPoint":{"lat":40.9947,"lng":29.1040}, "startTimeInMillis":62896, "passenger": 1 },
        {"GeoPoint":{"lat":40.9939,"lng":29.1067}, "startTimeInMillis":119687,"passenger": 1 },
        {"GeoPoint":{"lat":40.9939,"lng":29.1069}, "startTimeInMillis":122888, "passenger": 1 },
        {"GeoPoint":{"lat":40.9935,"lng":29.1081}, "startTimeInMillis":161416, "passenger": 1 },
        {"GeoPoint":{"lat":40.9934,"lng":29.1086}, "startTimeInMillis":175093, "passenger": 1 }
      ]
    }
  }
}}
```

As observed in the listings above, a "taxi route" is represented as a set of steps. A step is represented by startTimeInMillis, passengerCount and a GeoPoint. StartTimeInMillis is the time when the taxi is at that GeoPoint. Passenger count shows how many passengers are in the taxi in that step. This information is used for calculating the success/performance/utilization metrics. A "taxi" object itself is named with a unique identifier (utid), current location is assigned to the first step. The simulation workflow is described in Figure 9.

The scenario time is set to 08:00 am and the 3000 taxis start within a minute and start to transport their passengers. Remember that, in the time injection part, the time when passengers start their journey with a taxi is picked randomly from the [0-59] second range. In the simulation setup, the previous states of the taxis are not considered and whenever the first customer starts her journey, it is assumed that the taxi suddenly appears at that location. When the passengers are delivered the taxi stops or disappears from the simulation (*i.e.* does not carry more passengers).

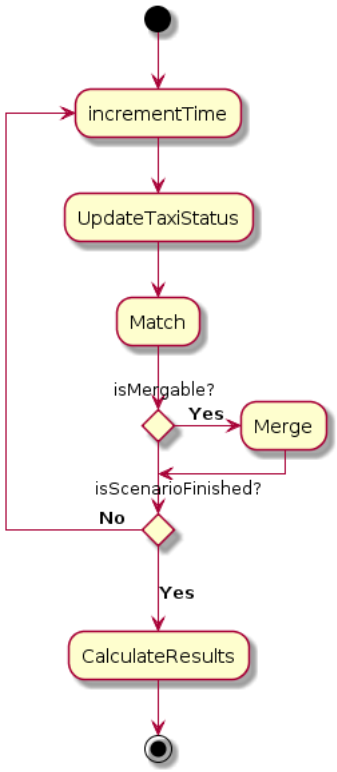

Figure 9: Simulation workflow

This way we can compare the fill rate efficiency of different strategies. Another way would be to let all taxis go and deliver as many passengers as they can and report this value. In this context, **passenger taxi request** can be defined as one of the 3000 taxi routes that has just started. In other words, it means that current time is equal to the starting time of the taxi route. If there is a passenger taxi request, the system will find the list of "available" moving taxis that is physically in the range of a certain radius. In our simulation, we run tests with 300m, 600m, and 1000m search ranges. The "available" in simulation setup means taxi is not merged before, however, in real setup, "available" may mean the taxi is either unoccupied or has available seats. Thus, it is important to note that, in the *simulation setup* a taxi shall have 2 passengers at most.

Whenever there is a taxi request, the system will **match** this passenger with a list of candidate taxis. If one found, the system will continue and calculate the feasibility of the **merge**. Currently, if the goodput is greater than 1, two passengers will be merged into a single taxi.

As stated above, the scenario will start at 08.00 am and taxi locations will be updated and new taxi request will be handled at each second. Checks will continue till all the taxis drop their passengers.When the scenario finishes, the evaluation tool will calculate and display the analysis of the scenario.

The results of scenarios are debugged, analyzed and visualized in the web-based dashboard, whose snapshots are given at Figure 10, 11 and 12. As shown in Figure 10, there is just one taxi and one passenger. As time passes, there is a taxi request at some certain distance, and the possible route for it is shown in Figure 11. However, this second passenger is merged with the first taxi and the resulting route is shown with the red route in Figure 12.

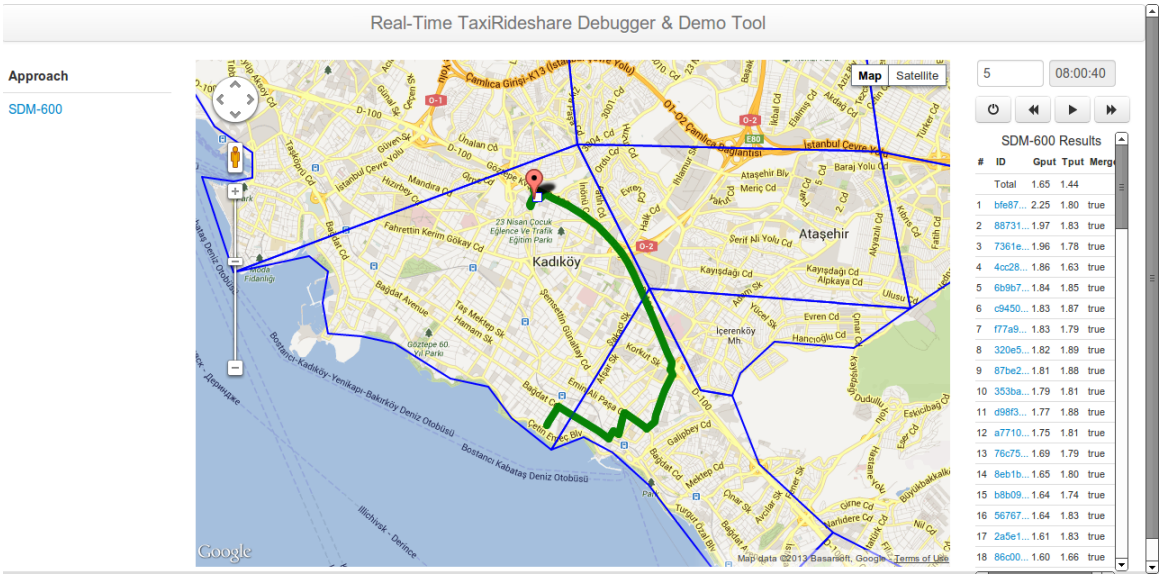

Figure 10: Demo: Taxi arranged for the 1st passenger

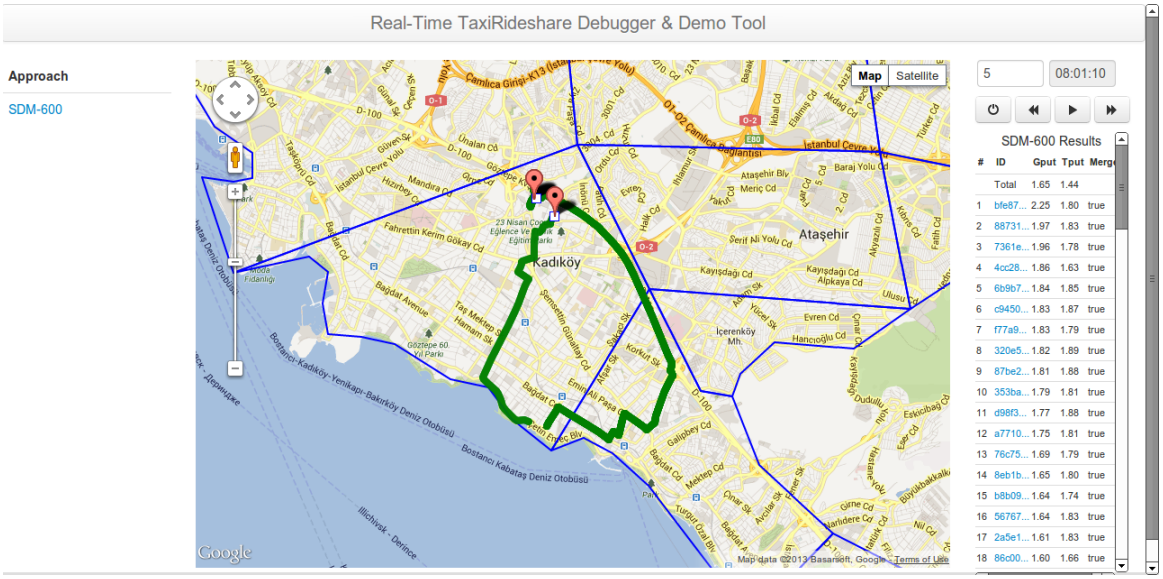

Figure 11: Demo: 2nd passenger requests a taxi and she is merged with the 1st passenger.

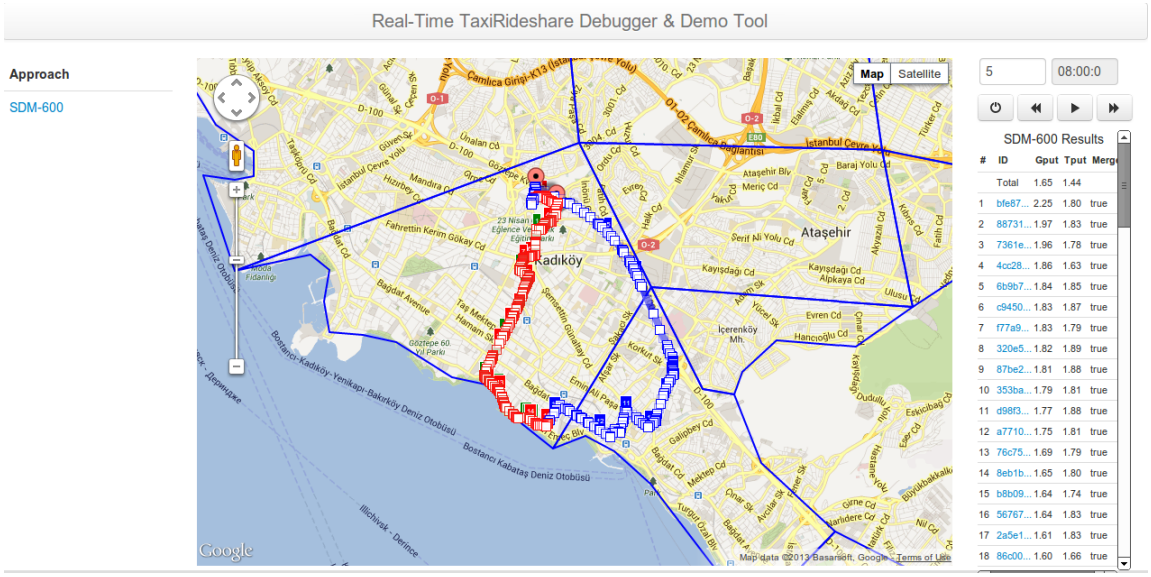

Figure 12: Demo: Offline view of merged passengers.

5.3 Mobile Location Tracker Application

This section describes the mobile application developed for collecting the route information in real-time. It uses the PhoneGap API.

The technology that enables real-time taxi ride-sharing is smart phones. For this

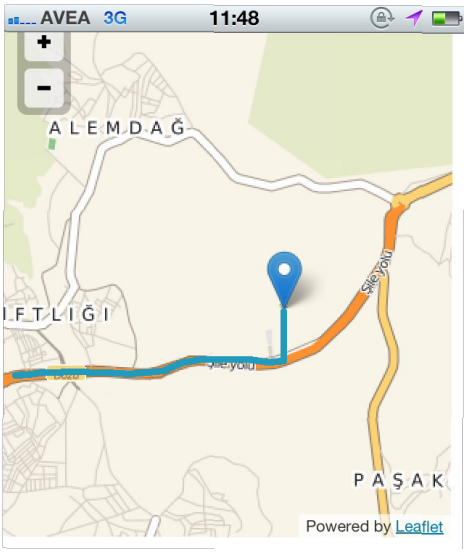

Figure 13: Location tracker application developed with PhoneGap

reason, a mobile location tracker application is developed. To easily deploy a single application in various devices, PhoneGap API[12] is preferred. However, we need to be careful that the location tracker function should continue to work in the background when the device goes to sleep mode. Otherwise, no location data will be captured during that time. We implemented this functionality during development.

The screenshot of this application is given in Figure 13.

CHAPTER VI

RESULTS

6.1 Experimental Setup

This system is tested for 3000, 20000 taxi setup in Istanbul. For route merging Google Directions Service is used. Since there is 2500 Google Directions Service query limit, simulations run mostly on 3000 taxi setup. For all the results of the tests refer to Appedix B. 3 different matching approaches(SM, SDM and ORM) are tested. All of those approaches search for available taxis within a certain range. In this work, the performance of 300m, 600m and 1000m search range is tested. We also use the merge strategy "$goodput < 1$ not accepted" and call these **missed merges**. Attemps to merge NN-passengers, whether hit ($goodput > 1$) or missed ($goodput < 1$) lead to two path queries to Google Directions Service. The first path query uses the destination of the riding passenger as the waypoint and the second path query uses the destination of the pickup passenger as the waypoint; we pick the path leading to higher goodput, if of course the goodput is higher than 1.

6.2 Evaluation

As observed in Figure 14, at 300m SM strategy serves the maximum count of passengers. ORM and SDM follow SM, respectively. It is more likely to find a source matched passenger than finding a source and destination matched passenger. Note that if the taxi search range is increased, the amount of passengers served increase as well.

For 3000 taxi setup, 1500 merges can occur at maximum. The percentage of the passengers served is given in Figure 15 and when the taxi search range is increased

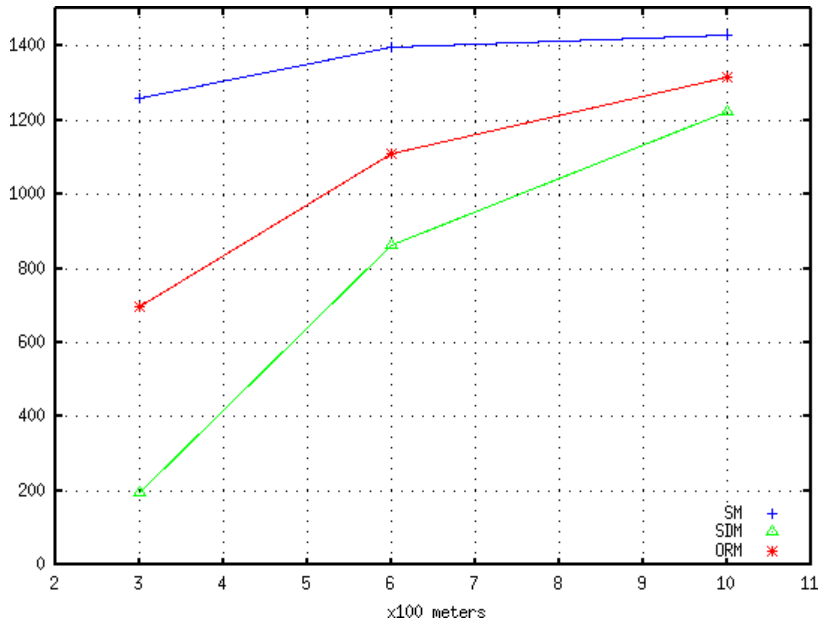

Figure 14: Merged taxi count for SM, SDM and ORM strategies on 3000 taxi setup.

to 1000m, almost 80% of the passengers are served for both strategies. The good-

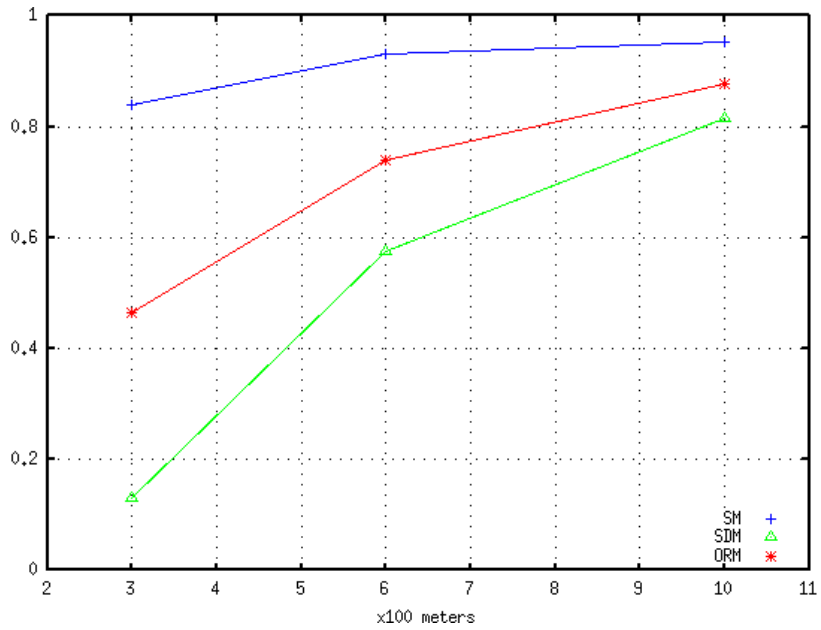

Figure 15: Ratio of merged taxis over maximum merge count for SM, SDM and ORM strategies on 3000 taxi setup.

put, throughput and extra time of tests are analyzed in 2 different ways: Total and Merged. In the "merged" calculation, only the routes of the merged passengers will be calculated. For instance, in 300m range SM test for 3000 taxi test setup there are

1259 merged and 482 unmerged passengers(Table 6 in Appendix B). While calculating "merged" results only the 1259 merged passenger routes will taken into account. Whereas, on "total" results all of the passenger routes will be considered.

As observed in Figure 16, the best goodput is at 300m search range of SDM strategy. ORM and SM follow SDM, respectively. It is also observed that SDM strategy performs better in merged goodput value than ORM and and SM for all the tested search ranges. Similarly, ORM is better than SM for both search ranges when goodput metric is compared. Best goodput value is obtained with SDM at 1000m search range. In total goodput 600m search range for all strategies is the optimal solution. When the amount of unmerged taxis increases, the total goodput converges to 1. If SDM strategy is considered, since the ratio of passengers served in 300m search range is lower than 0.2, the difference between total goodput and merged goodput is huge. For 1000m search range, more than 80% percentage of the passengers are served, the merged, total goodput difference is relatively less.

The throughput of the system is given in Figure 17. This figure looks similar to the goodput figure (Figure 16). However, since the overhead distances are also added to the calculations, throughput is relatively larger than goodput results.

In taxi ride-sharing service, the cost for the passenger is the extra time she spent for a ride. SDM with 300m search range has the minimum time cost as observed in Figure 18.

The system cost in terms of Google Direction Service request count is given in Figure 19. SDM strategy has the minimum request count. The maximum request count is in the 1000m search range of SM strategy and it is close to 20,000 requests. The reason for this relatively astronomic number is the merging condition that goodput must be greater than 1. It is obvious that there will be numerous source matched taxis for a passenger, however among those matched taxis, it takes a relatively large number of Google Directions Service queries to find a merge with goodput greater

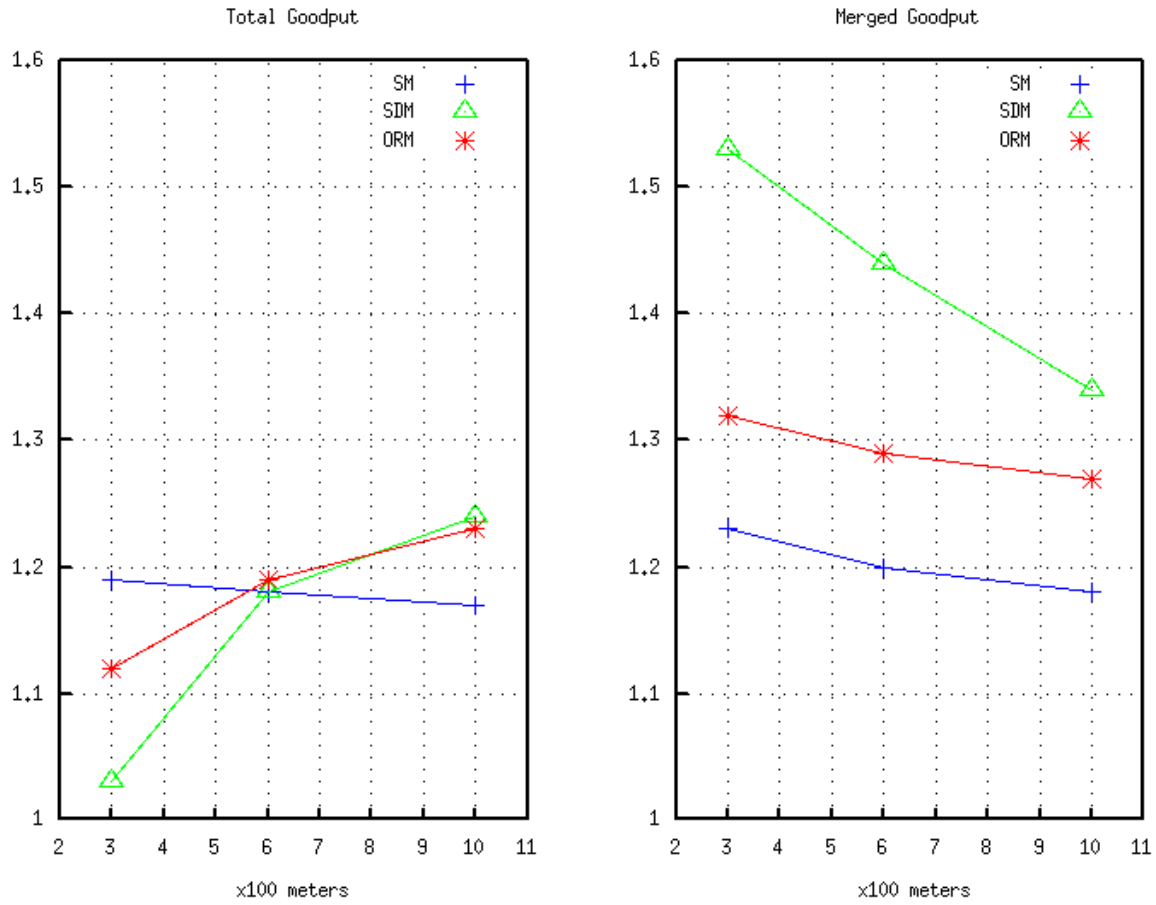

Figure 16: Goodput comparison for SM, SDM and ORM strategies for 3000 taxi setup.

than 1. Consider the 600m NN-search scenario in Table 7 in Appendix B. 1272/3000 trips were unmerged, meaning they served 1272 passengers only (1 per car), whereas for 1000m NN-search the unmerged number dropped to 586/3000. The 600m NN-search range led to 864 hit merges, *i.e.* $864*2 = 1728, 1728/3000$ requests were served, whereas the 1000m NN-search led to 1207 hit merges, *i.e.* $1207*2 = 2414, 2414/3000$ requests were served. For the 864 merges of 600m NN-search we made 2434 Google Directions service queries (including 2 for unshown 353 missed merges) and for the 1207 merges of 1000m NN-search we made 6080 Google queries. Making these Google queries can have both latency and monetary costs (currently the service rate is approximately $10,000/month for 100K requests/day). This cost has to be justified by

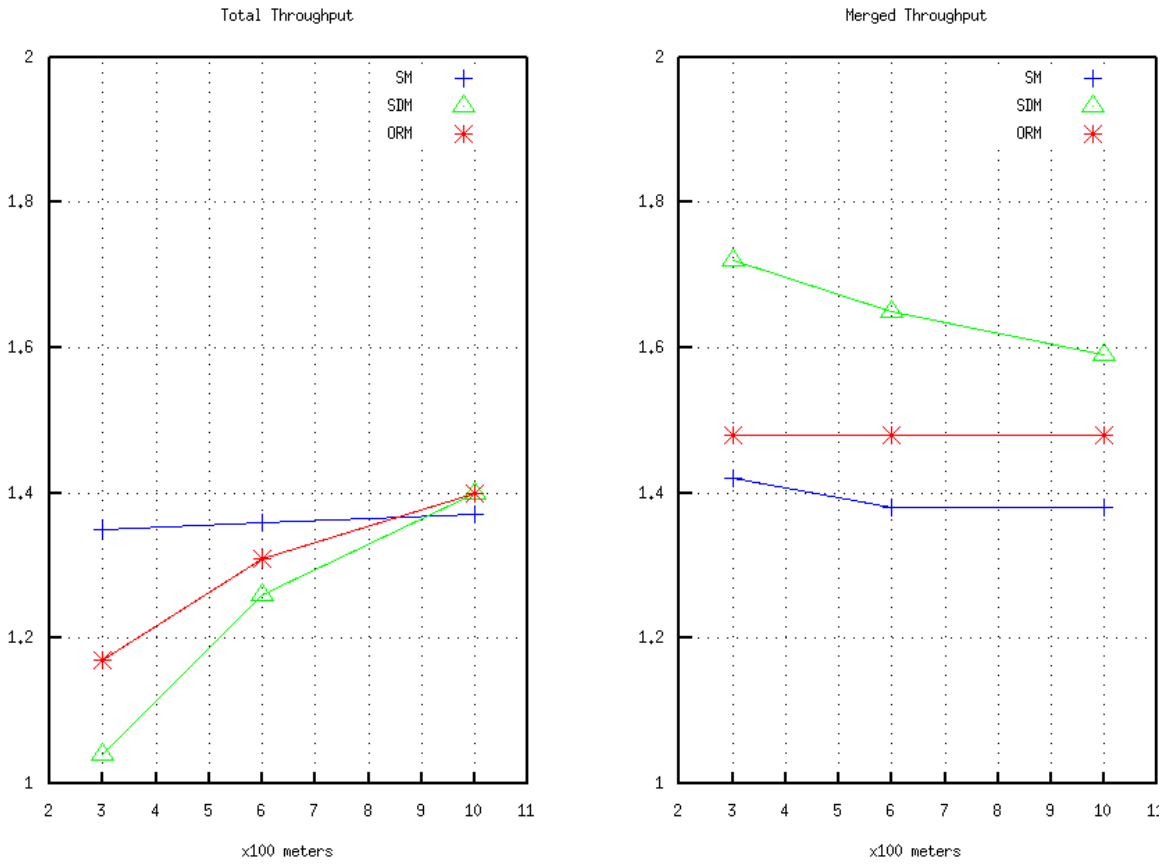

Figure 17: Throughput results for 3000 taxi test

increased throughput and goodput metrics (time, money, gas savings via carpooling). However, we see that both the throughput and the goodput of 600m NN-query searches are higher (1.65, 1.44 resp.) than 1000m searches (1.59, 1.34 resp.). This clearly shows that the quality of the merges increases as the search distance decreases. However, limiting the search distance too much could result in lack of "good" merging opportunities (*i.e.* $goodput > 1$), so we need to strike a balance in selecting this parameter.

Considering SDM has the best goodput and throughput results, least cost of time and Google Directions Service, it is the best strategy for the taxi ridesharing service.

The most important incentive for both passengers and drivers is the money paid and gained. 4 different strategies for money calculation were tried and displayed at Figure 20. As observed in this figure, there are 4 different calculations for 300m,

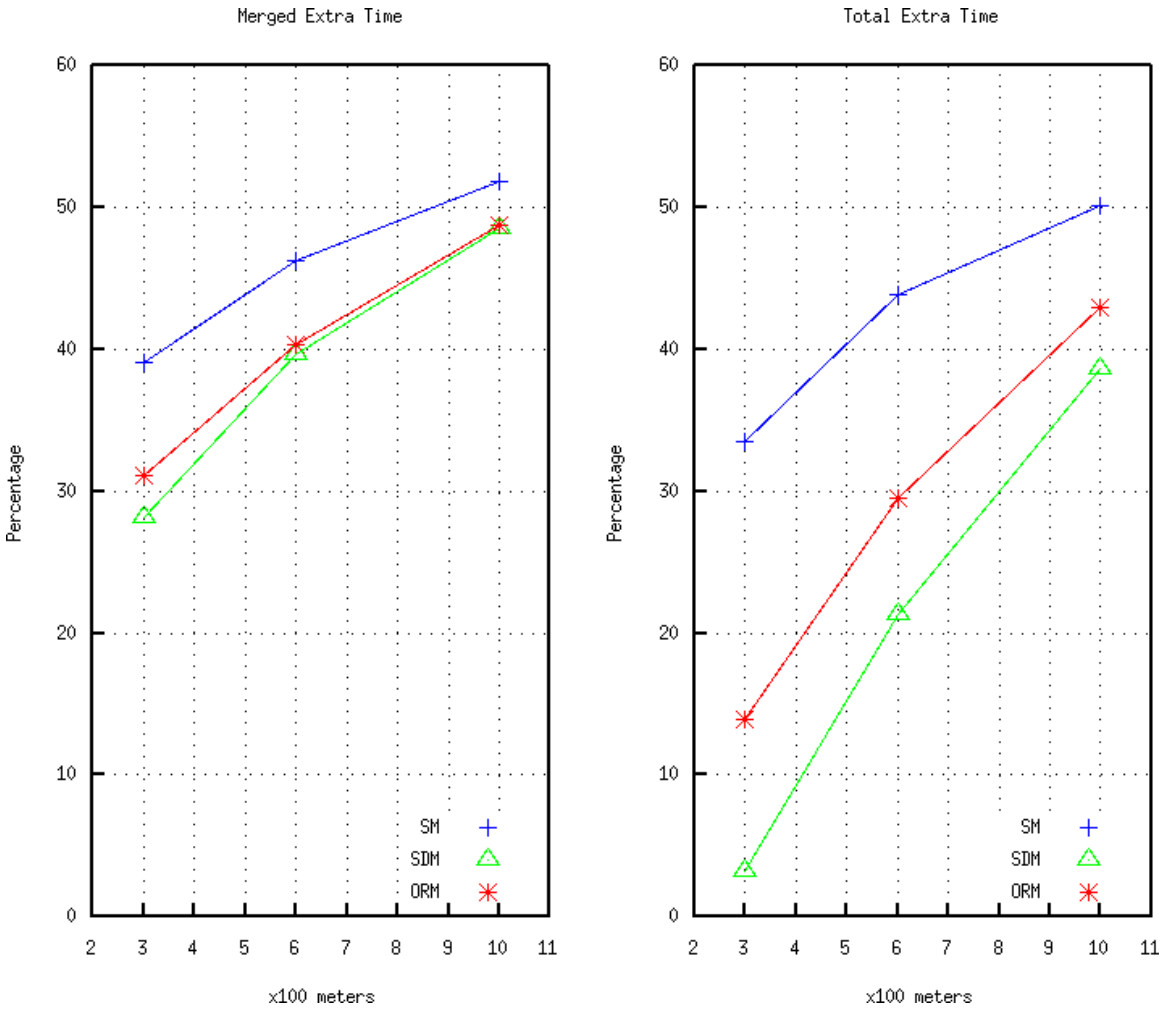

Figure 18: Time cost for SM, SDM and ORM strategies for 3000 taxi setup.

600m and 1000m search ranges. In the **1st payment strategy**, the 2 passengers that share the ride will also share the initial fee(2.7 units of money) in half and money increment in 100m is still the same(0.17 units of money in our system). In the **2nd strategy**, the initial fee of the first will be divided evenly to the ridesharing passengers. However, this time, the money increment for 100m will be 0.21 units of money. In the **3rd strategy**, each passenger will pay their initial fee. For instance, if there are two passengers sharing the ride, the driver will be paid $2.7 * 2 = 5.4$ units of money. The 100m distance will cost 0.17 units of money. In the **4th strategy**, different from 3rd strategy, 100m distance will cost 0.21 units of money.

As observed in Figure 20, passengers will earn more when the search range is

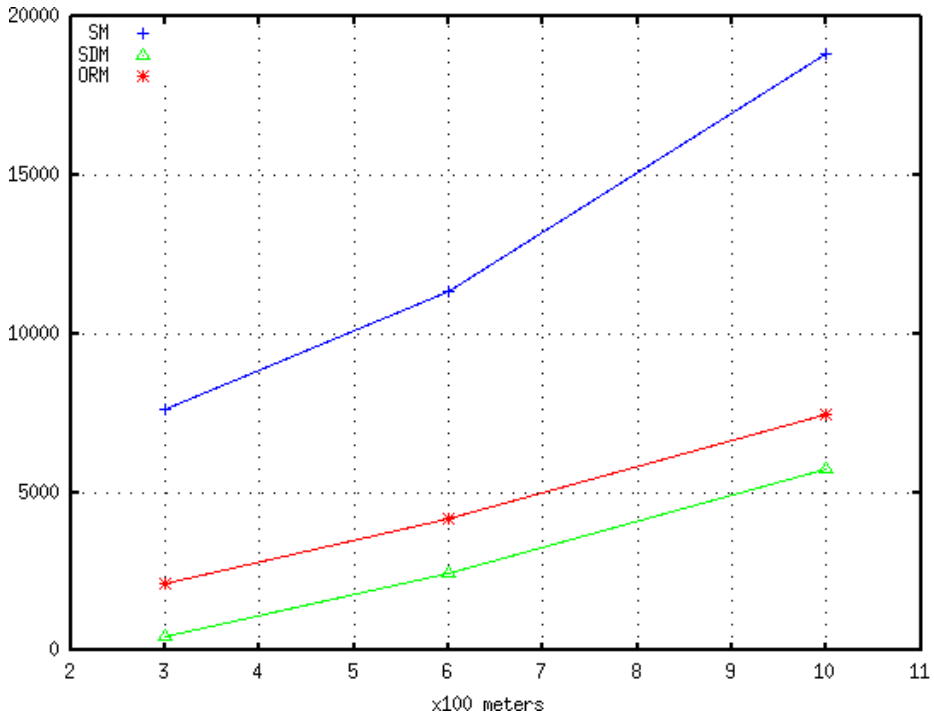

Figure 19: Google Directions Service query count for SM, SDM and ORM strategies on 3000 taxi setup

decreased and similarly larger search range work better for taxi drivers. In the SM approach there is no crossing (or equal benefits) point for both taxi drivers and passengers. In SDM approach, 300m search range and 3rd payment strategy (all passengers pay the initial fee, 0.17 units of money for 100m) is equally adventageous for both drivers and passengers. In ORM, this value is in the middle of 3rd and 4th payment strategies.

There are several parameters (or knobs) we can adjust to distribute the savings among taxi drivers and passengers. For example, if we charge the initial fee to all passengers the driver will earn higher but the monetary savings for passengers may diminish. If we divide the savings only among passengers then the drivers make no additional income and are not motivated to carry carpooling passengers around. In fact, recent news suggest that there is a growing unrest among taxi drivers (in New York) against the users of these mobile carpooling applications. We believe the drivers who are also stakeholders need to be included in this game, either as users of

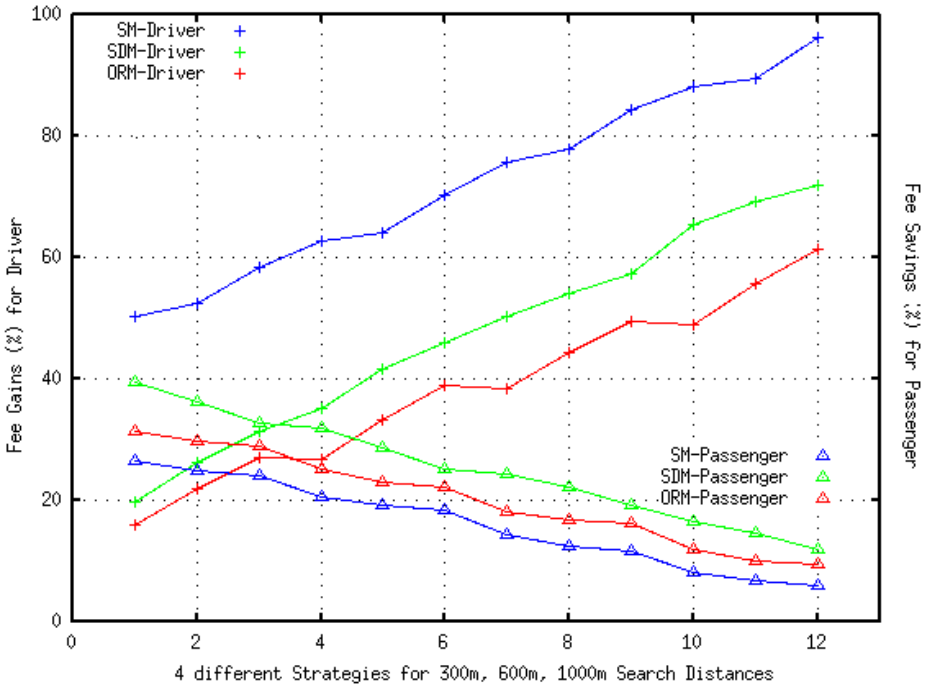

Figure 20: Percentage of the fee gains for drivers and savings for passengers for 3000 taxi setup.

the application collecting points-ratings or as an earning party.

6.3 Background and Related Work

Three categories of related works are studied to position this thesis in the research community.

6.3.1 Recurring Ride-sharing

Recurring rideshares concerns routine commutes. Carpooling services such as 511.org [13] and Avego[14] are the web services that also provide an official mobile support. For static and small size of the problem, it is possible to find an optimal solution [15]. Compared to static recurring ride-sharing in which all routes and time schedules are known in advance, our work that focuses on dynamic and real-time taxi ride-sharing poses a more difficult challenge due to dynamically changing routes, time constraints, *etc.*

6.3.2 Dial-A-Ride Problem

Taxi ride-sharing problem can be viewed as a special member of the Dial-A-Ride (DARP)[3] problem class. This problem also shall be named as a member of Traveling Salesman Problem, or more specifically Vehicle Routing With Time Windows[16]. These problems are mainly focused on planing vehicles to a set of geographically scattered passengers with certain constraints such as time windows, while minimizing an objective, minimizing the path traveled, a.k.a shortest path as in the traveling salesman problem. DARP has been analyzed for various transport scenarios such as school bus routing[16], transits for handicapped and elderly people [17], *etc.* Solving such optimization problems globally with 20K taxis and millions of people is still challenging. Yet, in our scenario all of these entities are constantly moving, which makes global optimization an impractical approach. Optimization techniques can be layered on top of our system for offline analytics and optimization.

6.3.3 Real-Time Taxi Recommender Systems

Recently there are several works attacking the real-time and dynamic version of dial-a-ride problem (static). For instance, the web service BiTaksi searches for available taxis at real-time. This system is no different than an online cab stand. Applications like BiTaksi aim to find one more passenger to taxis when the taxi is returning back from a service. Different from those applications in our thesis, we further want to improve the passenger count in a taxi.

In the literature, the closest work to ours is T-Share[2] project of Microsoft Research Group which develops a taxi ride-sharing system in Beijing that aims to serve real-time requests sent by taxi users and generates ride-sharing schedules. This system is built on an offline set of taxi data that is provided by the Beijing municipality. In comparison, we built a system that works and generates online taxi data with Google Directions service.

CHAPTER VII

CONCLUSION & FUTURE WORK

Social networking services are evolving towards mobile web applications and location based services is becoming popular. In future, continous route and experience sharing based applications will be the trend. Such applications can be applied in many fields ranging from tourism to transportation. In this work, all of the building blocks to develop an "end-to-end location-based application" was described. OpenStreetMap was used due to its rich schema and extensibility. MongoDB was used for its geospatial support, scalability and flexibility to define documents. PhoneGap was preferred because of its portability and scalability. Using all of those building blocks, a real-time taxi ride-sharing mobile cloud service was developed and its technical problems were addressed. In this application, more than 20,000 realistic taxi routes were generated, a simulator is developed to setup a scenario. In this simulator new passenger coupling and merging strategies were implemented, those strategies were compared and results were evaluated. To validate and debug the routes, merging and matching strategies, a web-based dashboard was developed.

Based on our results, this system can reduce up to 40% of the city traffic using Source & Destination based matching strategy when candidate passengers are searched within 600m range over the Istanbul metropolitan area. In addition to the promise for reducing the traffic, the system has the potential to be deployed easily in any city in the world.

In the future, we recommend the following research actions to be taken:

- Current recommendation engines all focus on item sales from online stores (e-stores) such as Amazon, e-bay, *etc.* If our system is deployed, route based

recommendation can be added. With data obtained for passenger and taxi travel habits, a recommender engine shall be developed.

- In this thesis, we only showed applications for geolocation-based matching strategies. In the future, we plan to extend the matching criteria to include rider profile information and add rich metadata to the shared travel information including images and videos of visited places.

- In this thesis, we did not address the security, privacy, and trust issues in mobile ride-sharing applications and cloud services, which can be critical for the safety and usability of the service. While this is an orthogonal design issue out of our current scope, we do foresee a couple of mechanisms that will be useful. First of all, just like any other product or service, there will be recommendation (star-rating, reviewing, commenting) mechanisms in place to eliminate bad eggs from the system. Second, messages will be encrypted, signed and authenticated by their sender-receivers so that illegal location-tracking by eavesdroppers or content modifications will not be possible. Finally, the cloud service can be distributed and made fail-proof against Denial of Service (DoS) and distributed DoS (DDoS) attacks with known techniques. We skip details here for brevity and note these topics as interesting future work.

7.1 Threats to Validity

In this thesis, new matching and merging strategies are developed and tested. In the simulation setup there are several assumptions made to make the system easily tractable. However, these assumptions, that are listed below, can be a threat to the validity of the system, as well.

- In taxi route generation process, the regioning and inter-region probabilities may be different for Istanbul or other cities.

- Taxis are assumed to have 2 passengers at most, which can in reality be a higher number.

- After a taxi delivers their passengers, this taxi is not set as available in our simulations whereas taxis will continue to carry more passengers in real life.

- In real life, passengers should have time and personal preferences. For instance, a woman may not want to travel with a men, a passenger may request to be picked up in a certain time, or may have an arrival deadline. Such preferences were not taken into account in this thesis.

- All the taxis are dispatched in a 1 minute period in the simulation.. In this setup, for instance, in the 3000 taxi test there shall be around 1500 active taxis at a certain time. If the dispatch period was even longer(*e.g.* 5 mins or 10 mins), the active taxi count, and thus, the merge and match counts would decrease. We experimented these scenarios as well but found no additional benefits to ridesharing.

APPENDIX A

INTER-REGION PROBABILITIES

Table 4: Inter-region travel probabilities

Region#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0.73	0.07	0.07	0.04	0.04	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.06	0.61	0.12	0.03	0.12	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.08	0.08	0.42	0.08	0.08	0.04	0.04	0.01	0.04	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.01	0.03	0.12	0.62	0.12	0.01	0.01	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.04	0.02	0.04	0.78	0.04	0.02	0.02	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.00	0.01	0.00	0.01	0.07	0.75	0.07	0.04	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.07	0.71	0.07	0.00	0.01	0.01	0.07	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	0.00	0.00	0.00	0.01	0.01	0.01	0.07	0.70	0.07	0.01	0.01	0.04	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.04	0.08	0.02	0.00	0.08	0.77	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.88	0.09	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.08	0.82	0.08	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.07	0.01	0.01	0.07	0.71	0.07	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.10	0.01	0.01	0.01	0.10	0.67	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.79	0.08	0.08	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.00
15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.75	0.07	0.07	0.04	0.01	0.00	0.00	0.00	0.00	0.00
16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.88	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00
17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.01	0.81	0.01	0.12	0.00	0.00	0.00	0.00	0.00
18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.08	0.80	0.00	0.00	0.08	0.01	0.01	0.00
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.01	0.73	0.04	0.07	0.00	0.00	0.00
20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.90	0.00	0.00	0.00	0.00
21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.04	0.04	0.00	0.83	0.00	0.04	0.00
22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.80	0.04	0.08
23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.01	0.08	0.76	0.08
24	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.08	0.83

APPENDIX B

SM, SDM & ORM RESULTS

Table 5: SDM results for 20000 taxis in Istanbul

Distance		**600 m**	
UnMerged		10316	
Merged		4842	
Google Queries		14838	
M-Throughput		1.66	
T-Throughput		1.18	
M-Goodput		1.46	
T-Goodput		1.12	
M-Extra Time (mins)		37.42%	
T-Extra Time (mins)		15.26%	
Initial Fee	$\frac{Incr}{100m}$	**Driver**	**Passenger**
One Fee(2.7)	0.17	25.43%	36.66%
	0.21	40.85%	28.87%
Per Person Fee(5.4)	0.17	52.65%	22.92%
	0.21	68.07%	15.13%

Table 6: SM results for Uskudar Taxis (3000 taxis). P: Passenger, D: Driver

Distance		**300 m**		**600 m**		**1000 m**	
UnMerged		482		204		142	
Merged		1259		1398		1429	
Google Queries		7628		11364		18858	
M-Throughput		1.42		1.38		1.38	
T-Throughput		1.35		1.36		1.37	
M-Goodput		1.23		1.20		1.18	
T-Goodput		1.19		1.18		1.17	
M-Extra Time (mins)		39.09%		46.19%		51.84%	
T-Extra Time (mins)		33.48%		43.83%		50.13%	
Initial Fee	$\frac{Incr}{100m}$	**D**	**P**	**D**	**P**	**D**	**P**
One Fee(2.7)	0.17	50.40%	26.64%	52.60%	24.96%	58.39%	24.22%
	0.21	62.78%	20.60%	64.18%	19.27%	70.34%	18.50%
Per Person Fee(5.4)	0.17	75.71%	14.30%	77.89%	12.53%	84.40%	11.77%
	0.21	88.09%	8.25%	89.47%	6.84%	96.35%	6.06%

Table 7: SDM results for Uskudar Scenarios (3000 taxis). D: Driver, P: Passenger

Distance		**300 m**		**600 m**		**1000 m**	
UnMerged		2608		1272		552	
Merged		196		864		1224	
Google Queries		460		2434		5758	
M-Throughput		1.72		1.65		1.59	
T-Throughput		1.04		1.26		1.40	
M-Goodput		1.53		1.44		1.34	
T-Goodput		1.03		1.18		1.24	
M-Extra Time (mins)		28.15%		39.64%		48.53%	
T-Extra Time (mins)		3.2%		21.34%		38.58%	
Initial Fee	$\frac{Incr}{100m}$	**D**	**P**	**D**	**P**	**D**	**P**
One Fee(2.7)	0.17	19.85%	39.54%	26.37%	36.24%	31.46%	32.67%
	0.21	35.11%	31.84%	41.54%	28.58%	45.89%	25.28%
Per Person Fee(5.4)	0.17	50.20%	24.23%	54.23%	22.18%	57.48%	19.35%
	0.21	65.46%	16.54%	69.39%	14.53%	71.92%	11.95%

Table 8: ORM results for Uskudar Taxis (3000 taxis) P: Passenger, D: Driver

Distance		**300 m**		**600 m**		**1000 m**	
UnMerged		1608		782		370	
Merged		696		1109		1315	
Google Queries		2128		4162		7442	
M-Throughput		1.48		1.48		1.48	
T-Throughput		1.17		1.31		1.40	
M-Goodput		1.32		1.29		1.27	
T-Goodput		1.12		1.19		1.23	
M-Extra Time(mins)		31.08%		40.27%		48.75%	
T-Extra Time (mins)		13.97%		29.53%		42.97%	
Initial Fee	$\frac{Incr}{100m}$	**D**	**P**	**D**	**P**	**D**	**P**
One Fee(2.7)	0.17	16.07%	31.34%	21.94%	29.64%	26.98%	28.90%
	0.21	26.71%	25.04%	33.33%	23.07%	38.86%	22.25%
Per Person Fee(5.4)	0.17	38.41%	18.13%	44.45%	16.66%	49.59%	16.24%
	0.21	49.05%	11.83%	55.84%	10.08%	61.47%	9.59%

References

[1] "Open street map." http://www.openstreetmap.org/wiki/Main_Page.

[2] Shuo Ma, Yu Zheng and Ouri Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *International Conference on Data Engineering*, 2013.

[3] Jean-François Cordeau, Gilbert Laporte, "The dial-a-ride problem: models and algorithms," in *Annals of Operations Research*, vol. 153, Springer, September 2007.

[4] 10gen, "MongoDB." http://www.mongodb.org.

[5] Nick Roussopoulos, Stephen Kelley, Frédéric Vincent, "Nearest neighbor queries," in *ACM SIGMOD international conference on Management of data*, pp. 71–79, 1995.

[6] Anirban Mondal, Yi Lifu, Masaru Kitsuregawa, "P2pr-tree: An r-tree-based spatial index for peer-to-peer environments," in *Current Trends in Database Technology*, pp. 516–525, 2005.

[7] Peter N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *ACM-SIAM Symposium on Discrete algorithms*, pp. 311–321, 1993.

[8] Afsin Akdogan, Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi, "Voronoi-based geospatial query processing with mapreduce," in *Cloud Computing Technology and Science (CloudCom)*, pp. 9–16, 2010.

[9] Zoran Balkić, Damir Šoštarić, Goran Horvat, "Geohash and uuid identifier for multi-agent systems," in *Agent and Multi-Agent Systems. Technologies and Applications*, pp. 290–298, 2012.

[10] "GeoHashing." http://en.wikipedia.org/wiki/Geohash.

[11] "QuickHull." http://www.qhull.org/.

[12] "Phonegap." http://phonegap.com/.

[13] "511.org." http://511.org/.

[14] "Avego." https://www.avego.com/.

[15] R. W. Calvo, F. de Luigi, P. Haastrup, and V. Maniezzo, "A distributed geographic information system for the daily carpooling problem," in *Computer Operation Research*, Elsevier Science Ltd., 2004.

[16] M. Desrochers, J. Lenstra, M. Savelsbergh and F. Soumis, "Vechicle routing with time windows: Optimization and approximation," in *Vehicle Routing: Methods and Studies*, pp. 65–84, 1988.

[17] Alexandre Beaudry, Gilbert Laporte, Teresa Melo, Stefan Nickel, "Dynamic transportation of patients in hospitals," in *OR Spectrum*, pp. 77–107, 2010.

[18] Jean-François Cordeau, Gilbert Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem," in *Transportation Research Part B: Methodological*, pp. 579–594, July 2003.

VITA

I get my B.S. licence from Electrical and Electronics Engineering in Bilkent University. After graduation I worked for 2 years at Ayyazılım, an SME at METU Teknokent. In this 2 years, I completed several projects to Nurol Makina, Roketsan, Mikes and Aselsan. Now, I'm a member of the Cloud Computing Research Lab of Özyeğin University and working under the supervision of Asst. Prof. Ismail Arı. In my master's studies, I focused on mobile cloud services, recommender systems and geographical information systems.