ON MY WAY: SYSTEM FOR OPTIMIZING DRIVING ROUTES FOR CROWDSOURCING APPLICATIONS

BY XUEYUAN SONG

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. JANNE LINDQVIST

and approved by

_____

_____

_____

New Brunswick, New Jersey

May, 2014

ABSTRACT OF THE THESIS

On My Way: System for Optimizing Driving Routes for Crowdsourcing Applications

by Xueyuan Song

Thesis Director: Prof. JANNE LINDQVIST

Nowadays people rely on their phones for much more than just communication. Phones are used for social media, as cameras, for emailing, and since more and more people begin to use smartphones, for navigation. Conventionally, the route recommendation given by GPS has been considered as the optimal route search problem only between two locations - origin and destination [9] and to make it more efficient, the map needs to be updated manually. However, in the real cases, several restrictions may need to be considered in the route recommendation. The restrictions could be several intermediate destinations which must be visited before reaching the final destination. For example, a user wants to visit places that belong to several general types, such as a bank and a gas station before arriving at working company, and then many choices may be available along the route to company, where only one place from each general type is to be chosen. Additionally, there is no good way on mobile phones, especially on the Android platform to figure out how to get from point A to point B while navigating through intermediate waypoints belonging to user' specified general type. Hence, this thesis tries to bring this issue to light, and provides a solution to the problem. The thesis make the following three contributions: first, proposing the algorithm help search nearby places and a strategy help get the optimal route through multiple intermediate waypoints. Second, an Android based application including the design and implementation which gives the optimal route recommendation

by users' specified general types. Third, the result of the application is analyzed and discussed by using a JavaScript Application and R.

Acknowledgements

The thesis paper is made possible through the help and support from everyone, including: parents, teachers, family, friends, and in essence, all sentient beings.

Especially, please allow me to dedicate my acknowledgment of gratitude toward the following significant advisors and contributors:

First and foremost, I would like to thank Prof. Janne Lindqvist for his most support and encouragement. He kindly read my paper and offered valuable detailed advices on grammar, organization, and the theme of the paper.

Second, I would like to thank the rest of my thesis committee: Prof. Wade Trappe, Prof. Yanyong Zhang for their encouragement, insightful comments, and hard questions. As well as all the other professors who have taught me about courses over the past two years of my pursuit of the master degree.

Finally, I sincerely thank to my parents, labmates in HCI, and friends, who provide the advice and financial support. The product of this thesis paper would not be possible without all of them.

This material is based upon work supported by the National Science Foundation under Grant Number 1211079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the development of the Intelligent Transportation Systems (ITS) technologies, a vast amount of real time information is available for the travelers. Since the ITS technology powers the automotive navigation system, people rely on automotive navigation system more than ever before. An automotive navigation system is a satellite navigation system designed for use in automobiles. One of its main functions is to give the optimal route recommendation to the destination using the real time traffic information. In the system, a Global Positioning System (GPS) [1] navigation device is used to acquire position data and help to locate the user on a road in the unit's map database. Using the road database, the unit can give directions to other locations along roads also in its database.

The GPS navigation system has become one of the most popular applications for people. However, most automotive navigation devices give their route recommendation based the downloaded map, if information of the electronic map is out of date or some traffic incident or road maintenance event occurs in real time, the GPS navigation system may plan an erroneous route. Although the real-time online road information supported by the Radio Data System-Traffic Message Channel (RDS-TMC) [2] has improved this problem, the on-board GPS navigation system may not receive useful road information from RDS-TMC to help route planning.

Most smartphones have full GPS capability onboard, so navigation is a natural use for that. All the Android devices come with a turn-by-turn navigation system, and new iPhones come with turn-by-turn too. World sales of smartphones to end users totaled 968 million units in 2013, an increase of 42.3 percent from 2012. So one can safely assume that this means about 2.65 million GPS units are sold daily [3]. Moreover, the navigation apps based on the smartphone can retrieve real-time traffic information from internet to plan a faster route by

avoiding roads under construction.

Android Market provides a large number of navigation apps in the market. Navigon Mobile Navigator is one of them. With the help of Navigon Mobile Navigator, one can smoothly reach to the destination. Several navigation functions are included in this app such as Reality View Pro and Lane Assistant Pro. The realistic images of motorways signs and exits can be provided by the Reality View Pro. With Lane Assistant Pro navigate the right lane even in hectic situation. Google Maps with navigation is developed by Google Inc, it has following features: search by keyword, walking navigation, transit navigation, biking navigation, search along route, offline Reliability, and etc. Additionally, Google Maps also provides the satellite view, street view, and traffic view to help users find the destination. Glympse Navigation app has an easy way to use. With the help of Glympse, one can share their locations through Facebook, SMS, Twitter or email. Sygic is the most downloaded offline road GPS navigation app in Google Play. It provides the route to avoid town and heavy traffic and has following key features: high quality TomTom maps stored on the device, easy oriented 3D cities and landscape, junction view, SOS assistance nearby, friends on map, and etc. MapQuest provides the function to search by voice. With the help of this hands free input, drivers can focus on their driving. Copilot Live is one of the widely used navigation apps with set of features found in online and offline apps. The deepest impression function of this app is that one can access it without data connection. Android Compass is like a magnetic compass with several special features. When it is used, it determines the location with the help of Global Positioning System (GPS) or Wireless-Fidelity (Wi-Fi). The compass includes section of 4 poles like North, South, East, and West. And it provides 4 options to map or navigate the destination. Waze is an app that successfully connects social networking with traffic navigation. Users can contribute to their local driving community by just driving with Waze. Besides, Waze Navigation offers free travel directions. In this app, Waze is commanded through audio and on screen for comfort travel. This app needs to work on an active data connection.

However, the route recommendations given by GPS or Android apps as well as apps discussed above have considered as the optimal route search problem only between two location - origin and destination [9]. However, in the real cases, several restrictions may need to be applied when providing the route recommendation. The restrictions could be several intermediate

destinations which must be visited before reaching the final destination. The case of multiple waypoints destinations is much more complex than a route search only between two locations.

First, to find the optimal route to the final destination, the order of visiting the waypoints needs to be optimized. The problem may look similar to the travelling salesman problem. However, in this case, the final destination must be visited at the end. In addition, the waypoints could be considered as the selection from a set of possible choices. For example, a user wants to visit places which belong to several general types such as a bank and a gas station before arriving at final destination, then many choices may be available, where only one place from each general type is to be chosen. Moreover, the intermediate waypoints should be along with or near the way to the destination in order to further optimize the route distance and save time. Second, the driving route which goes through the optimal waypoints needs to be calculated. The shortest path calculation should be based on the actual real time driving route, and the routes information should be transformed to a human readable route in order to navigate.

Although Google Maps provide a way to calculate a route with specified waypoints, this can be only used via websites and instead of recommended waypoints, those waypoints need to be specified by users one by one manually. On the mobile platform, there is only one start and end point setting each time. Most navigation apps only care about start and end point instead of multiple waypoints too, however, it is common for users to get to the destination through multiple waypoints.

So in this thesis, an algorithm to optimize the results when searching nearby places and a method to filter the best three optimal routes is proposed, and then the thesis focuses on designing and implementing an Android-based navigation app, which has the following advantages to find the optimal route through multiple waypoints. First, recommend the nearby waypoints which belong to user's specified general types. Second, return three recommendation optimal routes and route information such as main street name, total time duration, and total distance on a satellite map. Third, an automatically notification to notify users when they are leaving home or work places. Furthermore, the strategy to design the application was evaluated and discussed by using R [10] and a JavaScript web application.

Chapter 2

Related Work

Most smartphones have some types of navigation system [16] or at least have access to a market where they can download an application that works as a navigation system. One of its main functions is to provide the optimal route to destination. The optimal route search problem has been extensively studied and then many algorithms have been proposed.

Given a graph with a set of verticals (or nodes) and edges, where each edge has a weight, single source shortest path algorithms, such as Dijkstra's [11] and Bellman-Ford's [12] [13] algorithms, compute the shortest path between a single node and every other node in the graph. In order to compute the shortest path between every node and every other node in the graph, there are all pairs shortest path algorithms, such as Johnson's [14] and Floyd-Warshall's [15] algorithms. Since these algorithms are all assuming that the weights on the edges are static, they must be re-executed whenever the weight of an edge changes.

The Dijkstra shortest path [4] and A* [5] algorithms are usually used to plan the route in the automotive navigation system. As compared to Dijkstra, A* achieves the shorter computation time by using heuristic functions. However these traditional route planning algorithms [6] [7] use static information such as the speed limit, instead of the real-time speed, of each road segment to calculate the route to the destination, which usually underestimates the actual traveling time and fuel consumption in turn.

To calculate the optimal route through multiple waypoints, Maruyama et. al. [27] proposed a method to find the optimal tour plan using genetic algorithm. Their work focuses on tour planning and users preferences for the destinations. In their work, they work on determining the optimal tour plan considering restrictions, so some destinations may be left out as a result. Kanoh and Nakamura [29] proposed a GA-based route search algorithm to find the route

considering the unspecified waypoints destination. In their method, however, it is not guaranteed that the intermediate destinations are always included in the route. Manoj and Shingo [9] proposed a method to optimize traveling times and the order of visiting multiple waypoints in order to determine the optimal route from origin to destination via multiple waypoints. An efficient Ran-Discrete Version (RasID-D) has been already proposed by Hirasawa etc. al. [20] to solve the discrete optimization problems, where the efficiency of RasID-D has already been shown. Therefore, in their paper, they mainly apply RasID-D to optimize the visiting order of the intermediate destinations. However, the paper does not mention how to select waypoints from a set of possible choices and how to get the real time routes between those waypoints. Lin Jun and Du Junping presented the system architecture, key technology and system realization for the travel route intelligent navigation system based on WEBGIS [25]. In their paper, they did the researches on the tourism route planning based on the traveler's types, preferences, characteristics and requirements. The basic function is to produce a satisfying tourism route planning for travelers based on the following: huge database of tourism destination information and database of tourism plan and product information; the result of data mining and classification of traveler's characteristic, market's hotspots and tourism products; the travelers requirements such as destination, time, and budget, etc. Jeffrey Miller and Muhammad Ali presented a fastest path algorithm that contains multiple unique destinations, which is a specialization of the Traveling Salesman Problem (TSP) with intermediate cities [8]. In their paper, they proposed the algorithm which can be used with Intelligent Transportation System (ITS) applications for determine the fastest route to travel to a set of destinations, such as required by delivery companies. Then the algorithm was executed on theoretical and actual transportation graphs in FreeSim.

To design an application, Ing-Chau Chang and Hung-Ta Tai [22] designed and implemented an Android-based navigation app, which can help reduce the travelling time, fuel consumption and emitted carbon dioxide. In their paper, they are focusing on using vehicular ad-hoc network (VANET)-based A*(VBA*) [28] route planning algorithm which is proposed to calculate the route which the shortest travelling time or the least oil consumption, depending on two real-time traffic information sources. A GPS navigation app was implemented on the Android platform to realize the VBA* route planning algorithm. First, a generic format to record the

vehicle's driving information in the Android platform carried by the vehicle's driver is designed. Second, each vehicle will distribute its recorded driving information to its neighbor vehicles within the IEEE 802.11p wireless communication range. Third, the real time traffic data of road segments accessible from Google Maps is also used in the Android-based GPS navigation App. However, the application was only designed for proving VBA* achieves significant reductions on both the average travelling time and oil consumption of the planned route, as compared to traditional route planning algorithms. In David M. Mark's [24] work, they presented the algorithm for selecting paths for travel by road. The algorithm is based on the principle that 'ease of description' is an important consideration in route selection. The route description is represented by a frame, and the route description cost is approximated by the number of slots in that frame. Then total route cost is just the sum of the route length and weighted route description cost.

In this thesis, to design an application on mobile platform, the proposed strategy should consist of three steps: first, waypoints should be recommended which belongs to user's selected types, then a real time driving route through specified multiple waypoints need to be optimized. Third, the route information is transformed to user readable routes and need to be drawn in an application.

In order to find the waypoints belong to users specified types, the local businesses and services need to be collected and use for further calculation. Yahoo PlaceFinder is a geocoding web service that recognizes a large number of place formats and returns rich geographic data about each result, including geographic coordinates, address components, and where on earth identication number(WOEID). The WOEIDs returned by the service can be passed to Yahoos GeoPlanet API for further geographic enrichment and discovery. Yahoo charge the service at least $3.00/1000 when the daily query is over 35,000. The Google Places API provides place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. Search for places can be either by proximity or a text string. In this paper, we chose to use the Google Places API, for the reason that it can reach millions of business, quickly and for free.

In real situations, the route needs to be calculated efficiently in real time. It seems impossible to collect all the actual routes information, however, this information can be provided by

third party applications. Google Maps have an easy to use website where a user can get directions from point A to B. If a user wants directions to multiple locations the user can enter the location. Bing Maps are very similar to Google Maps, but they do not give efficient routes [17]. MapQuest is very similar to Bing Maps in the way that it does it's waypoints, but it seems to be the most inaccurate. Yahoo Maps does not offer an option to choose waypoints between the users two chosen locations [18]. Results are different among the different devices, but for the most part they are not optimal. Now, the Google Direction API provides a service that calculates directions between locations using an HTTP request. The direction search could be several modes of transportation, include transit, driving, walking or cycling. Directions may specify origins, destinations and waypoints. The Directions API can return multi-part directions using a series of waypoints. Moreover, the Direction API was designed to return an optimal route through multiple waypoints. So in order to get the real time driving route, Google Direction API will be chosen in this thesis.

Chapter 3

Application Design

Traditional route recommendations are only considered about two locations which are origin and destination. So in this chapter, we are going to describe how to design an application which can recommend both intermediate waypoint and optimal route going through them. There are two parts in this chapter. In the first part, a simple scenario will be described which can provide a brief introduction of the application design. In the second part, a navigation path will be drawn to show the application design.

3.1 Simple Scenario

A user is leaving his home in the morning to go to work. The user knows that he needs a cup of coffee before reaching work, needs to put gas in the car, and needs to buy a pair of shoes because his got ruined last night. With a normal GPS system he would have to manually search for a coffee shop that is on his route to work, along with a gasoline station, and a shoe store and input their addresses all into the GPS system. With Android application in this thesis, the user puts his home address and his work address and then enters the general type of place he would like to go. The application has a drop-down auto-complete text box so that the user can pick Google Places keywords. The user enters cafe, shoe store, and gas station. The stopovers can be entered in any order, the application deals with ordering for best route. Basically, the application can calculate all the possible combinations from these stopovers. Upon pressing notification, the app will be launched and show up with best three optimal different routes that will take the user to all of the stopovers. The user clicks the radio button group on the map to switch one of three recommended routes, and at the same time, the corresponding route information will display on the information board.

3.2 Navigation Path

When first running the application, a user will be asked to type in both home and work addresses and then led to a space to enter the general types of intermediate waypoints for the journey. The user will enter type of waypoints from a drop-down auto-complete text-view. This text view gives the user the ability to pick up any of the Google Places generic keywords as a general type of waypoints. Unless being manually deleted by users, this information includes the address information and waypoint types will be stored in a SQLiteDatabase on cell phone local storage. By using nearby places searching algorithm, all the possible places which belongs to those general types will be considered. For each general type, many choices may be available, but only one is to be chosen. Then for a set of specified types, many combinations of intermediate waypoints will be available, those waypoints combinations will be optimized and further filtered by the selection algorithm. Finally, the three recommended optimal results will be showed on a satellite map.

The application can be launched simply by clicking the notification. To make it easy to use, a notification was issued usually at 9:00 am or 5:00 pm, by clicking the notification the user can directly see the map view, route information and a group of radio buttons to select the best three optimal routes on the map as it is showed in figure 3.1.

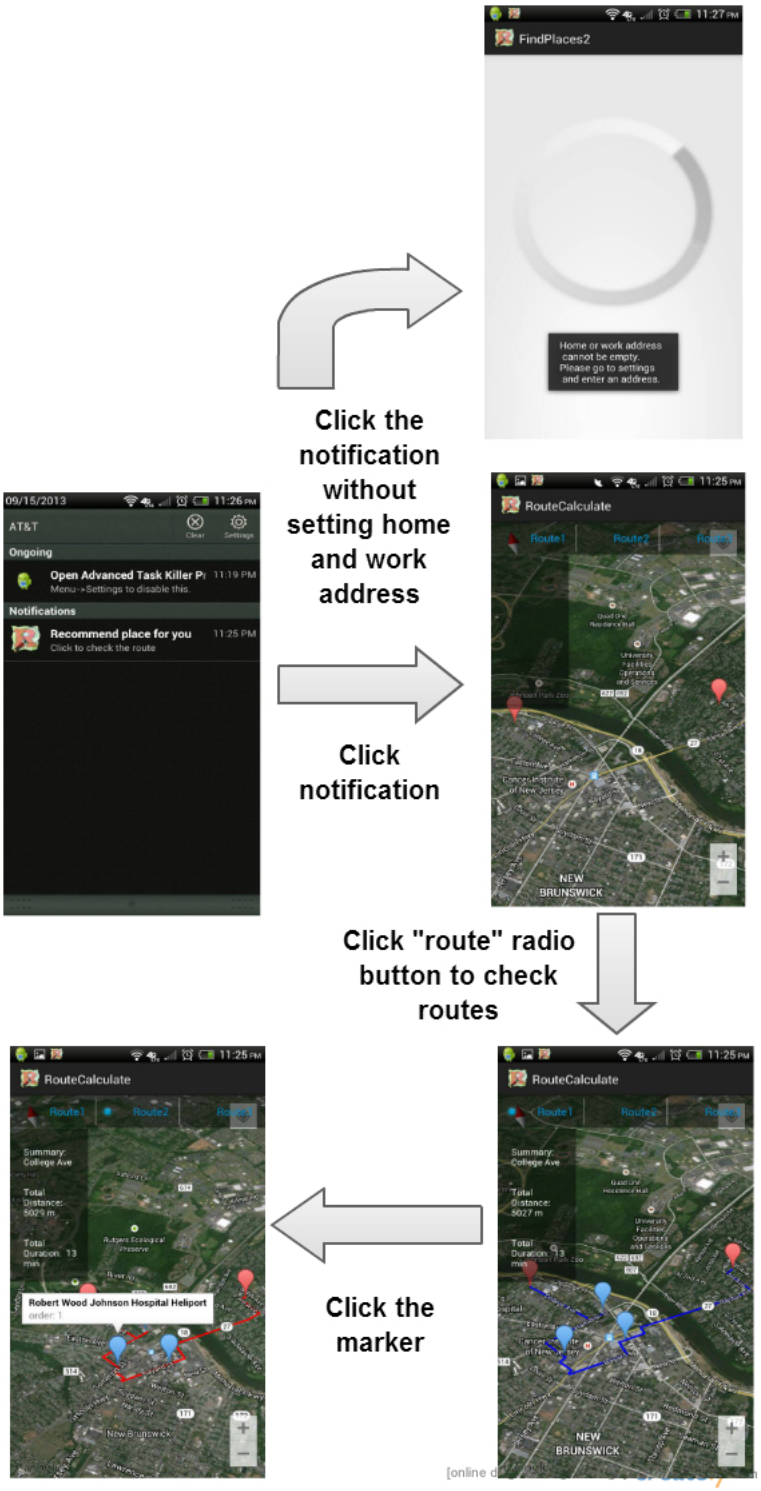

Figure 3.1: The figure consist several screenshots of the application. When the notification was launched, by clicking the notification, the user can see a map view with a group of three radio buttons on it. By clicking on each button, the optimal route will be drawn on the map, and the route information will be displayed on the information board.

Chapter 4

Application Implementation

The Android API libraries and developer tools necessary to build, test, and debug apps for Android application are provided by the Android SDK. So in this chapter, Google Direction API, Google Places API and Google Maps API v2 will be used to implement the application. The algorithm to search the nearby places and the strategy to calculate the real time driving routes will be implemented on Android platform.

4.1 Automatic Notification

Automatic notification [33] was implemented by using “IntentService” and “NotificationManager”. IntentService is a base class for Services which can handle asynchronous requests (expressed as Intents) on demand. Client can send requests through startService(Intent) calls, then the service is started as needed, handles each intent in turn using a worker thread, and stops itself when it runs out of work. This “work queue processor” pattern is commonly used to offload tasks from an application’s main thread. We chose “IntentService” class because it can simplify this pattern and take care of the mechanics. In this application, a class called “PingService” which extends IntentService is implemented. Then the notification was built by “NotificationManager” and pending activity which will be activated when users click on the notification, was added to the notification when implementing on HandleIntent(Intent). The notification building process should be like:

```
builder =
new NotificationCompat.Builder(this)
.setSmallIcon(R.drawable.ic_launcher)
.setContentTitle("Recommend place for you")
.setContentText("Click to check the route")
.setDefaults(Notification.DEFAULT_ALL);
```

The pending intent which is showing the map view can be simply added to notification by using “setContentIntent”. By using the thread.sleep() function we can set to launch the notification at any time.

4.2 Search Nearby Places Implementation and Algorithm

There are various Google APIs used in order to get efficient results while maintaining uniformity. The first API used in the application is Google Directions API [31], the Directions API allows 2,500 calls per day and individual request may contain up to 8 intermediate waypoints, there is no key needed to use it. Direction API can be used to get the directions from home to work. It will return a JSON object with the directions which are needed to get strictly from point A to point B. For example, as showed in figure 4.1, one root element of the JSON object is "Routes" which contains a set from the beginning to the end of the route. Each route can contain a legs array which includes a section on the line from information. Each element in the array is used to specify the full itinerary of the calculated route in the journey. The itinerary contains "steps" element which contains a set of sections, and these sections are used to represent each individual on the trip away sections of information. The section will include the latitude and longitude of both start and end locations.

The JSON object is parsed then. The strategy to find the closest waypoints is to search at each "end location" for the specified waypoints. The search would only occur if the previous turn was more than a specified distance from the next, this was so that there would not be redundant places to route to.

To look for the waypoints which belong to users' specified general place type, Google Places API [32] is designed for the purpose. To use Google Places API, first we need to enable the service and get the key from Google API Console. Google Places API can power location-based app. It provides detailed information about places across a wide range of categories. By using nearby search, searching for the places within specified area will be easily implemented. Google allows a daily quota of 1,000 API calls. This was more than enough for testing. The Places API also returned a JSON object. The object contained latitude and longitude, address, rating, and the name of each place. Those places returned could be organized by rating or distance. We chose distance because we're looking for the most efficient route not most pleasant place. Then three closest results on each place search will be saved for further use. This would give a total of (3*(Number of turns turns within X meters of last)) different locations, it is plenty and proved to be enough in following experiments. After removing duplicated places,

```
{
  "status": "OK",
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },
        "polyline": {
          "points": "a~l~Fjk~uOwHJy@P"
        },
        "duration": {
          "value": 19,
          "text": "1 min"
        },
        "html_instructions": "Head \u003cb\u003enorth\u003c/b\u0
        "distance": {
          "value": 207,
          "text": "0.1 mi"
        }
      },
      ...
      ... additional steps of this leg
    ...
    ... additional legs of this route
```


Figure 4.1: The figure showed the JSON Output when calling Google Direction API. Generally, only one entry in the “routes” array is returned for directions lookups, though the Directions service may return several routes if passing “alternatives = true” when sending HTTP request.

those places will be stored into a two-dimensional ArrayList according to their general types, for example, ArrayList<ArrayList <Places>>.

4.3 Find Possible Shortest Path Algorithm

The places ArrayList will be further used to make waypoints combinations. In order to find the optimal route to the final destination, the optimal waypoints need to be selected. Because for each general type, there could be many places available after nearby searching. The goal is to select one place from each type, and make them to a set of waypoints, and then each set should be a possible waypoints combination. For each waypoints combination, calculating the shortest path through the waypoints in the combination and then selecting the three best results which mean three shortest paths from those combinations. Furthermore, all the waypoints in each combination will be retrieved and sent for requesting directions by using HTTP request. Finally, the application shows users the recommended optimal actual routes of those combinations on the map.

When calculating the shortest path for all the waypoints combinations, basically there are two strategies can be used to select the best three optimal routes. The first strategy is calling Google Direction API for all the waypoints combinations, since Direction API will automatically optimize the order of visiting waypoints, it is unnecessary to do it again, then actual driving route distance of all the intermediate waypoints combinations will be retrieved so that those distances can be further used for comparison. Google Direction API will return a route with information which is automatically optimized and also taking traffic condition into consideration; it is a simple way to get the optimal routes. However, there are lots of shortcomings when using this strategy. First, calculating directions is a time and resource intensive task, so it takes too much time to get response when calling Google Direction API and if the number of waypoint combinations is too large (more than 200), the app will be crashed, because it takes too much time to execute the code in “doinbackground function”. Second, when calling the Google Direction API at such a high frequency, it may lose data and get no response from the remote server. Third, The Directions API has limited 2,500 requests per day, each time calling API counts as a single request. Individual request for driving may contain up to 8 intermediate waypoints in the request. If every possible combination needs to be called the API once, as a result it could be very expensive because of the usage fee of Google API. For example, we suppose that user select three general types of waypoints, and each general type can return 5

places along the route to destination, so the number of possible combinations will be 5^3, which means for each search if calling the Google API once, when running the app once, the average number of API requests would be 125. The second strategy is first connecting all intermediate waypoints with each other in each combination, then using shortest path algorithm to find the short path based on their bird view distance, finally only send the best three waypoints combinations to call Google Direction API. The second strategy may not be intuitive, because there is no intuitive evidence to show the bird-view distance is related with both actual route distance and driving duration. However, to make it more practical, the second strategy is implemented in the application. The result will be showed and discussed in following chapter.

Hence, the strategy is first recursively getting one place from each type, then making it to a possible waypoints combination. After connecting every place with each other in a certain combination, Kruskal Algorithm is used to calculate the shortest path of each combination. The distances between any places can be calculated by using latitude and longitude of those places.

The radius of earth R = 6378.137 km. The latitude and longitude of waypoints A and B is A(lat1, lng1), B(lat2, lng2). The code of calculating distance is:

```
private double R = 6378.137

private double rad(double d)
{
return d*MathPI/180.0;
}

public double calculateDistance(double lat1,
double lng1, double lat2, lng2)
{
double radLat1 = rad(lat1);
double radLat2 = rad(lat2);
double a = radLat1 - radLat2;
double b = rad(lng1) - rad(lng2);
```

```
double s = 2*Math.asin(Math.sqrt
(Math.pow(Math.sin(a/2), 2)
+Math.cos(radlat1)*Math.cos(radLat2)
*Math.pow(Math.sin(b/2), 2)
));
s = s*R;
return s;
}
```

Now the bird view distances are calculated and then only returned the best three combinations from the result. The waypoints of each combination will be sent to call Google Directions API. In this case, only three times of API calls are needed when running the app.

Directions API request takes the following form:

```
http://maps.googleapis.com/maps/api/directions/ output ? Parameters
```

Wherein, output can be JSON and XML. In our case, we use JSON as the format of response. It can be easily parsed by using JSON object. There are several parameters that should be used in order to get the correct information. Three mandatory parameters are origin, destination and sensor. So the origin is set to home address, destination is set to work address, and sensor should be false. One can also set many optional parameters (notification), that we want to specify here like mode and waypoints. To calculate the route, we can specify the mode of transportation mode which includes Driving, Walking, Bicycling and Transit. By default, the route calculation for the driving route. Waypoints parameter can be simply used by adding one or more to the pipe character (—) separated by address or location. By specifying waypoints parameter, the return route will pass through those given places. By default, route service is provided by the places of a given order. We set “optimize:ture” as waypoints parameters first parameter passed to press more effectively rearrange the order of places, so that it can return the optimal route. By examining the calculated route, the “Waypoint_order” element from response JSON Object will show the sequence of those waypoints.

```
"Waypoint\_order" : [ 1 , 0 , 2 , 3 ]
```

4.4 Display Routes On Map

Google Maps fragment can provide a map view in an app. The map fragment can be added to an activity's layout file simply with XML. To use Google Maps API [30], again the application needs to be registered and Google Maps API v2 needs to be enabled, and the courtesy limit is 25,000 requests/day. To display route on map, first GoogleMap object needs to be created, and then the makers which identify locations can be easily added or removed on the map. Polyline class defines a set of connected line segments on the map. Polyline objects consist of a set of LatLng locations, and create a series of line segments that connect those locations in an ordered sequence. Markers can be used for marking the waypoints on the route while polyline can be used for drawing the route on the map. To create a Polyline, we first create a PolylineOptions object and then places points which parsed from HTTP request need to be added. Points represent a point on the earth's surface, and are expressed as a LatLng object. Line segments are drawn between points according to the order in which we add them to the PolylineOptions object. Hence all points need to be added to a PolylineOptions object by using PolylineOptions.add() function according to the order returned by the Directions API. At last polyline needs to be added to the map object by calling GoogleMap.addPolyline (PolylineOptions). The method also returns a Polyline object with which polyline can be altered at a later time. The route is showed as figure 4.2.

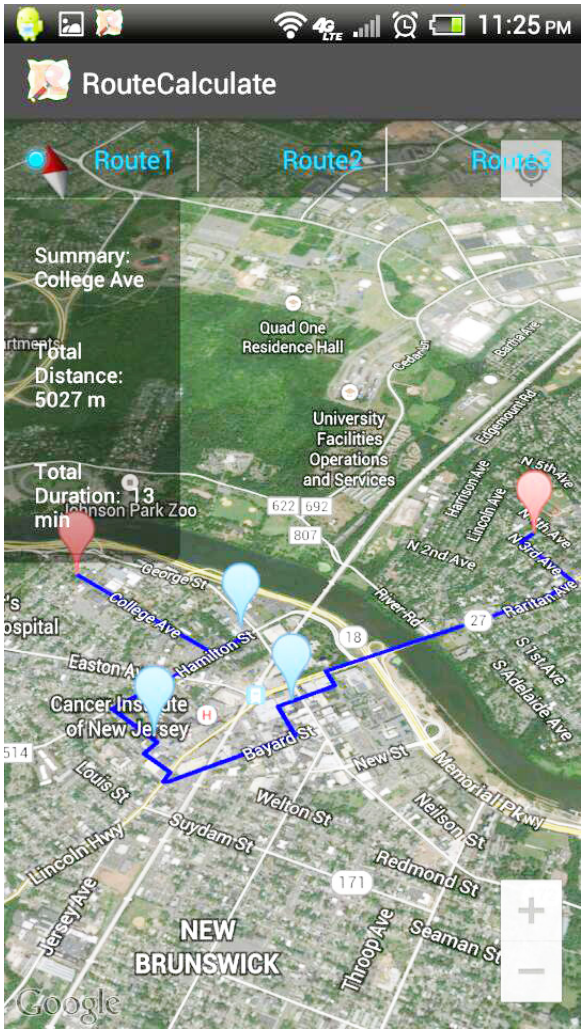

Figure 4.2: On the top of the screenshot, a group of three radio buttons can be used for switching three recommended routes, below them there is a transparent information board which is showing the corresponding routes information include main routes name, time duration, and distance.

4.5 Information Board Design

When calling the Google Direction API, the application can get a JSON object not only with routes information but also some useful information such as total route distance, time duration, and main street name. This useful information can be further used for notifying users. The JSON object will be parsed, and then the information will be retrieved and showed on the screen. Basically, the total driving distance which is measured in meter, the driving duration which is measured in minutes and the main street name will be displayed on the screen with a transparent background.

4.6 Summary

Basically, to implement the application, an algorithm is used to help search nearby places and a strategy is used to help get the real time optimal routes through multiple waypoints. The whole process can be described as follow:

1. Using Google Direction API to calculate the optimal route which goes through origin and destination.

2. Along with the route to the destination, we use Google Places API to search nearby places at each corner on the route.

3. Then now we can get several possible choices belong to users specified general types.

4. After removing the duplicated places, the places belong to same general type will be stored in an ArrayList. For example, if a user wants three general types, and they are gas station, bank and restaurant, in this situation, now there are three ArrayLists.

5. Every time selecting each place from an ArrayList, after searching all the ArrayLists, make them to a waypoints combination.

6. For every waypoint in the combination, connecting them with each other, and then using Kruskal algorithm to calculate the shortest path between them.

7. Then we can get the shortest path of all the waypoints combinations. Sorting them in an ascending order based on the distance.

8. Sending route combinations which have the most three shortest path of bird distance to call Google Direction API.

9. The routes return by Google Direction API will be saved and drawn on satellite maps.

Chapter 5

Experiments and Results

Birds-eye view is an elevated view of an object from above. The bird view distance can be regarded as the absolute distance between location A and location B. Suppose the latitude and longitude of two locations A and B are (lng A, lat B) and (lng B, lat B), then the bird view distance of A and B can be described as $\sqrt{(lngA - lngB)^2 + (latA - latB)^2}$.

In this thesis, bird-view distance is used as the reference to find the best three optimal routes. The strategy will be evaluated by several experiments. In real situations, many factors will have a great influence on the final result. Those factors could be the type of the waypoints, current traffic conditions, route conditions, typography and the density of road. So the experiments are much complex in real situations. There are two key questions to be answered in the experiments. The first question is why bird view distance can be used as a reference when selecting actual routes distance. Second, in which situation some points are valid while some are not when using the strategy proposed in this thesis.

The details of the experiments are described below:

5.1 Experiment to plot data on 2D graphs

In the first part, we select three most common general types as testing types, and they are gas station, restaurant and bank. Both start and end points are chosen from ten different states all over the nation which cover both cities and rural areas. When running the application, for each pair of start and end points, the bird view distance of all the possible intermediate waypoints combinations will be calculated. To do the experiment, actual driving distance and time duration of all the waypoints combinations also need to be calculated, and then sort them in an ascending order based on their bird view distances. The average number of possible combinations for all the pairs is 104.7.

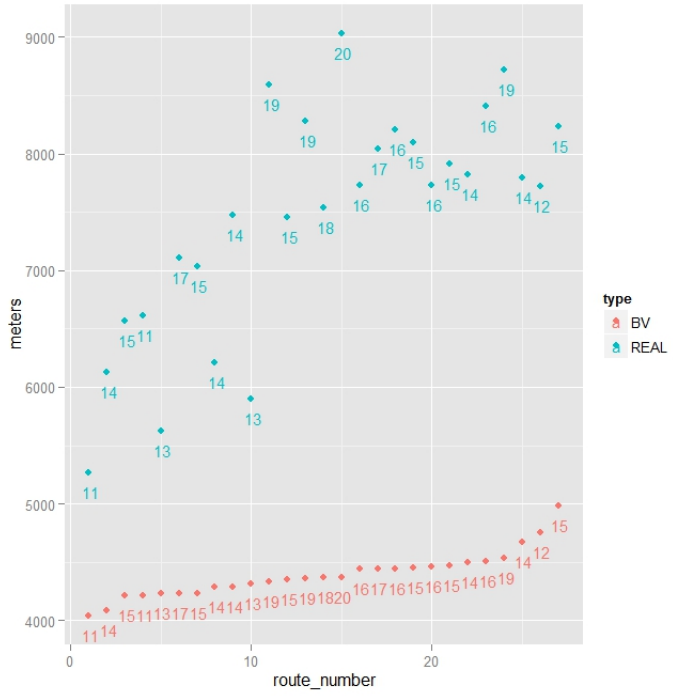

Figure 5.1: In this figure, x- axis stands for the number of waypoints combinations for one pair of start and end point, and these points are sorted in an ascending order based on their bird view distances. Y-axis represents the distance which is measured in meter. The orange points mean bird-view distance while the green points mean actual route distance for one route number. The labels on those points mean time duration which is measured in minutes.

To help better understand the result, we used R to plot the data. Figure 5.1 shows one of them, in figure 5.1, x- axis stands for the number of waypoints combinations for one pair of start and end points, and these points are sorted in an ascending order based on their bird view

distance. Y-axis represents the distance which is measured in meter. The orange points mean bird-view distance while the green points mean actual route distance for one route number. The labels on those points mean time duration which are measured in minutes. More results will be showed below and discussed in discussion chapter.

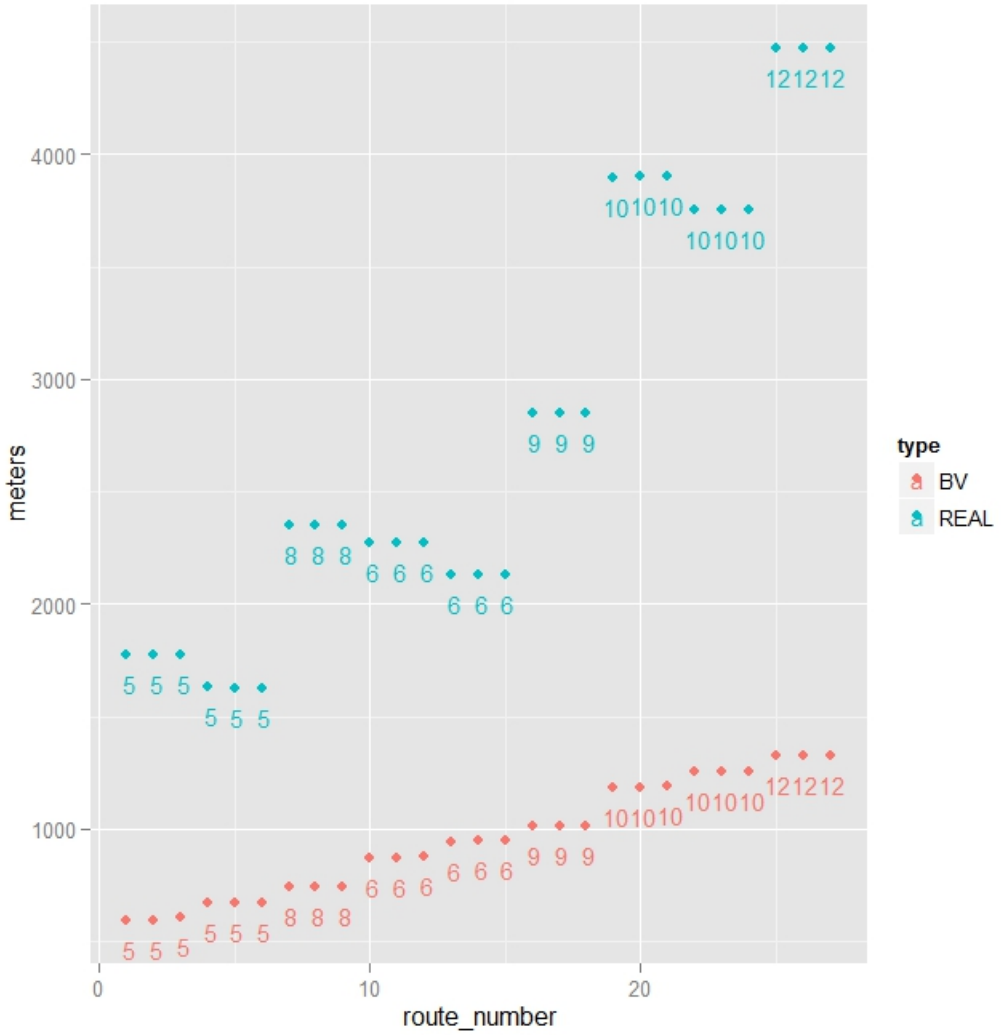

Figure 5.2: In this figure, x-axis stands for the number of waypoints combinations for one pair of start and end point, each number represents a possible waypoints combination, and those combinations are sorted in an ascending order based on their bird view distance. Y-axis represents the distance which is measured in meter. The orange points mark bird-view distance while the green points mark actual route distance for one route number. The label on those points means time duration which is measured in minutes. The data is tested in Washington D.C. and the addresses of start point and end point are 1101 New Hampshire Avenue Northwest Washington, DC 20037 and 2121 I st NW, Washington, DC 20052.

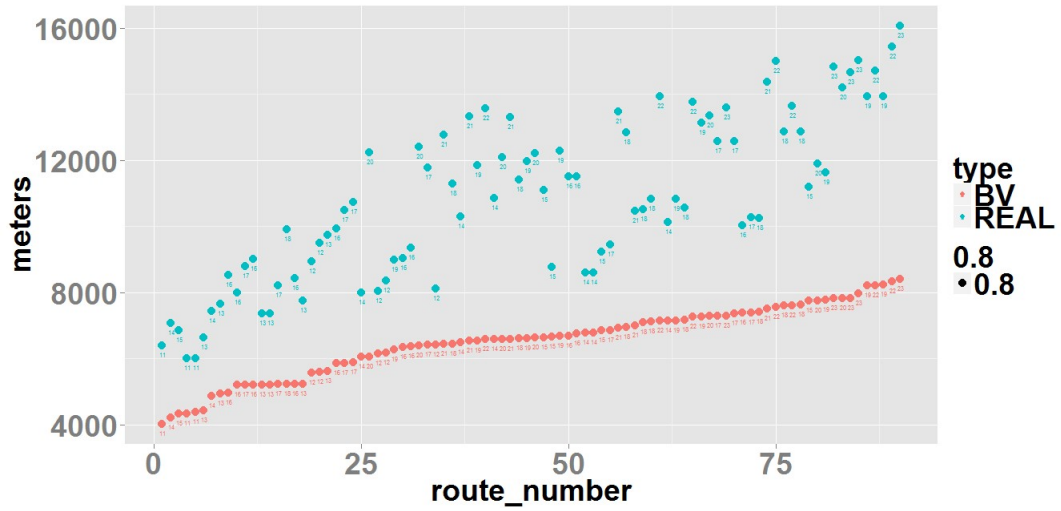

Figure 5.3: In this figure, x-axis stands for the number of waypoints combinations for one pair of start and end point, each number represents a possible waypoints combination, and those combinations are sorted in an ascending order based on their bird view distance. Y-axis represents the distance which is measured in meter. The orange points mark bird-view distance while the green points mark actual route distance for one route number. So in this figure, the number of waypoints combinations is 90. The addresses of start point and end point are 1714 Southwest 34th Street Gainesville, FL 3260 and 226 Tigert Hall, Gainesville, FL 32611

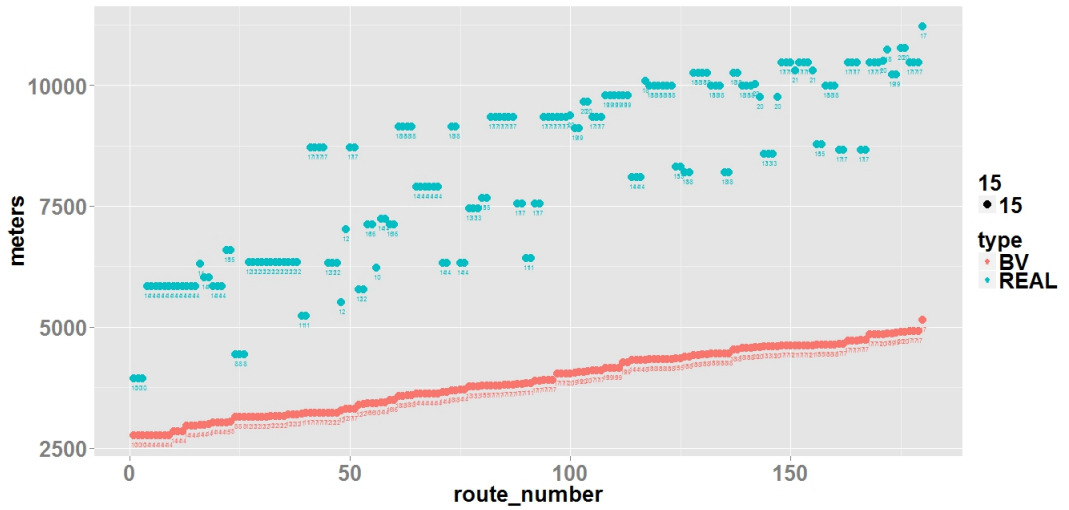

Figure 5.4: In this figure, x-axis stands for the number of waypoints combinations for one pair of start and end point, each number represents a possible waypoints combination, so in this graph it has 180 possible waypoints combinations and those combinations are sorted in an ascending order based on their bird view distance. Y-axis represents the distance which is measured in meter. The orange points mark bird-view distance while the green points mark actual route distance for one route number. The data is tested in Pennsylvania and the addresses of start point and end point are 1555 University Drive State College PA 16801 and DC 20037 and 201 Old Main University Park PA.

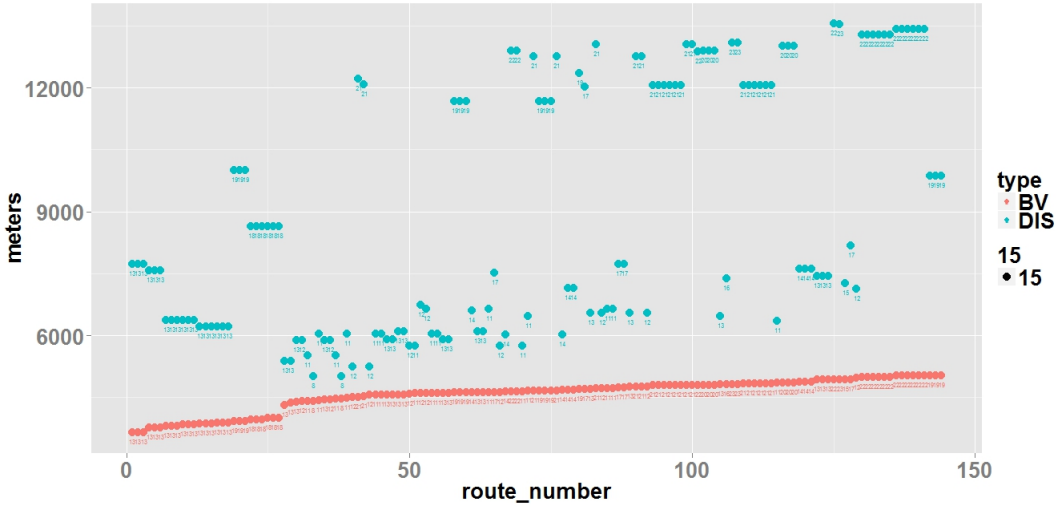

Figure 5.5: In this figure, x-axis stands for the number of waypoints combinations for one pair of start and end point, each number represents a possible waypoints combination, so in this graph it has 144 possible waypoints combinations and those combinations are sorted in an ascending order based on their bird view distance. Y-axis represents the distance which is measured in meter. The data is tested in Iowa and the addresses of start point and end point are 125 South Madison Street Iowa City IA 52242 and 300 East 9th Street Coralville, IA 5224.

5.2 Experiment to plot data on 3D graphs

In the second part, the experiment will be designed to answer the question why the bird view distance could be a criterion to select actual route distance. There are lots of factors that should be considered when answering this question, and those factors could be the route condition, popularity density, economic condition, and even topography. To reduce the influence of different factors, the experiments will be done in big cities where roads are criss-crossed and economic level is relatively high. In the following graphs, each point represents a possible waypoints combination for one pair of origin and destination, and is plotted on a 3D graph using R where x-axis represents the bird-view distance, y-axis represents the actual route distance and z axis represents the driving time duration. Distance is measured in meter and time cost is measured in minutes. In the following 3D graphs, points which are located in right-down side are actually what we needed, because those points have less bird view distance, less actual routes distance and less time cost, which mean when using bird view distance as a criteria to select actual routes, these points will be selected and should be effective for calling Google Direction API.

In figure 5.7, all points are distributed along the diagonal. If the points are uniformly distributed along the diagonal, it could be more accurate using bird view distance as the criteria, because the diagonal line means bird-view distance, actual route distance and time duration are proportional with each other.

Sometimes the waypoints combinations could be very small, which means when considering the strategy proposed in this thesis. The data size should be also taken into consideration. If the size is too small, the result will be like figure 5.8.

As showed in the figure 5.9, points are distributed into two parts. The situation usually happens, when there is no intuitive route between two places which belong to same general type. For instance, when searching nearby places, A, B are two places both belong to same general type -gas station, they are closed to each other but there is no direct route between them, which means you may drive around on the highway to find the exit or make a turn on a one-way route.

The points in different Graphs may have different distributions and patterns. But no matter

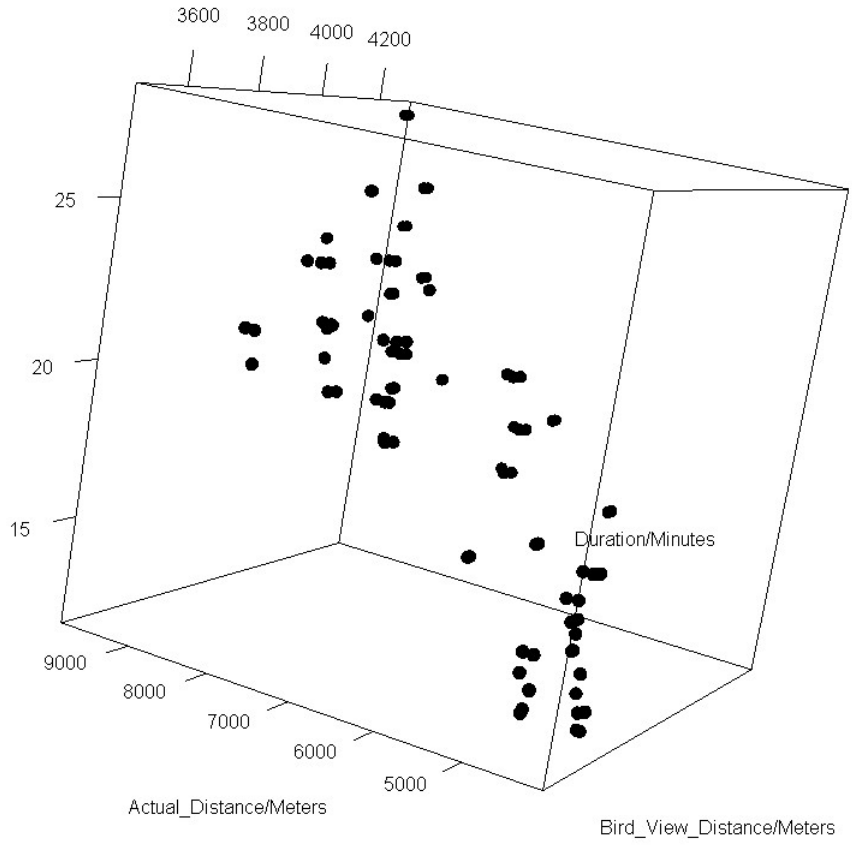

Figure 5.6: In the 3D graph, each point represents a possible waypoints combination for one pair of origin and destination, and is plotted on a 3D graph using R where x-axis represents the bird-view distance, y-axis represents the actual route distance and z axis represents the time duration. Distance is measured in meter and time cost is measured in minutes.

what patterns those graphs show, there always exist points which have less bird view distance, less actual distance and less time cost. Such points can be used for further calling Google API. In this situation, the best three bird view distances will be sent to call Google Direction API. To make the result more accurate, the first N (N<<total number of possible waypoints combination) of bird view distance can be also used for calling Google Direction API, however, deciding the value of N becomes another question, because the numbers of waypoints combinations vary depending on the pair of start and end points. If the number of the waypoints combinations is too small, the value of N should be small, and vice versa.

Bird view distance will be calculated by using Kruskal algorithm while actual route distance is calculated by calling Google Direction API. By selecting points having less bird view

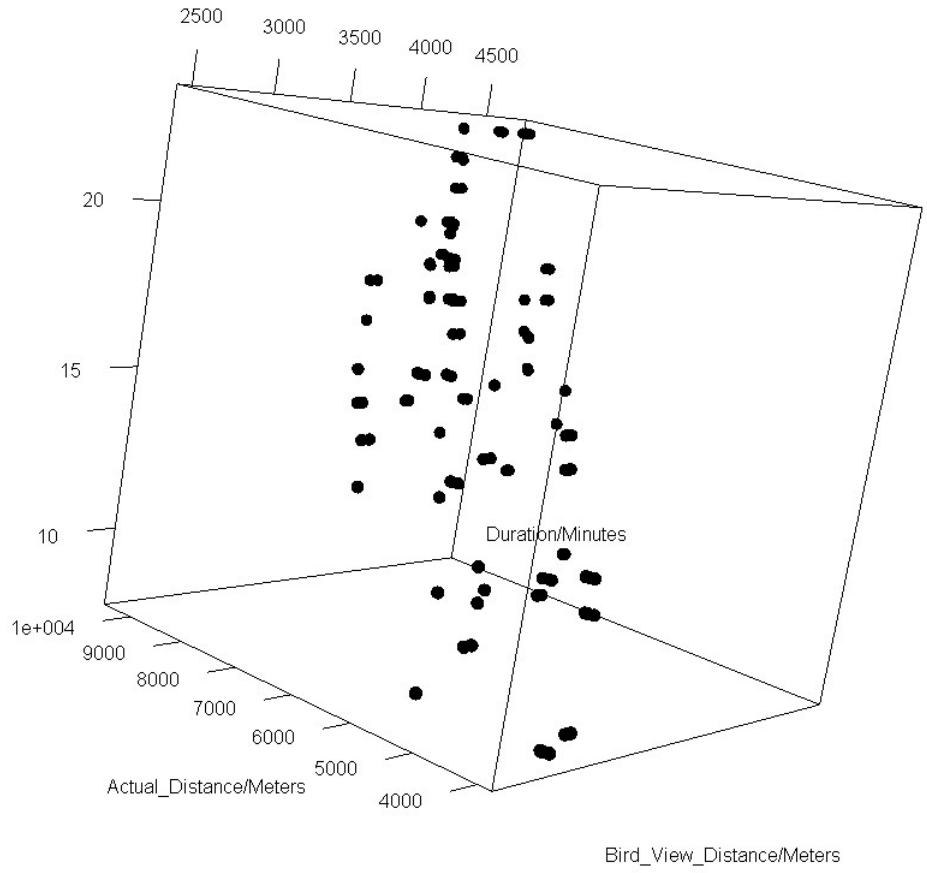

Figure 5.7: In the 3D graph, each point represents a possible waypoints combination for one pair of origin and destination, and is plotted on a 3D graph using R where x-axis represents the bird-view distance, y-axis represents the actual route distance and z axis represents the time duration. Distance is measured in meter and time cost is measured in minutes.

distance, since ideally bird view distance is directly proportional with actual route distance, points having shorter actual route distance will be also selected.

Selecting the start and end point is very important when doing the experiments. The start and end point should be reasonable; they should belong to a home area and work area. However, to identify which area people live in and in which area people work, more surveys need to be done. The current strategy in this thesis is for city area, searching the city on Google map, and then randomly choosing the points as a start point and an end point. For rural area, the addresses of state universities are searched as the start point and the nearby hotel as the end point.

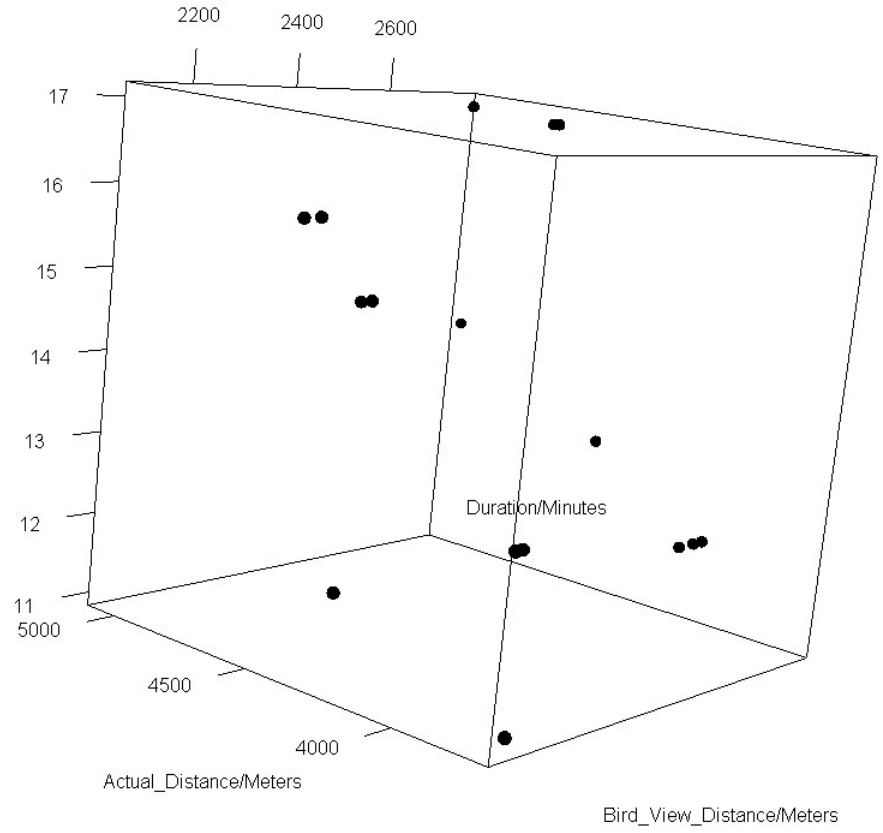

Figure 5.8: In the 3D graph, each point represents a possible waypoints combination for one pair of origin and destination, and is plotted on a 3D graph using R where x-axis represents the bird-view distance, y-axis represents the actual route distance and z axis represents the time duration. Distance is measured in meter and time cost is measured in minutes.

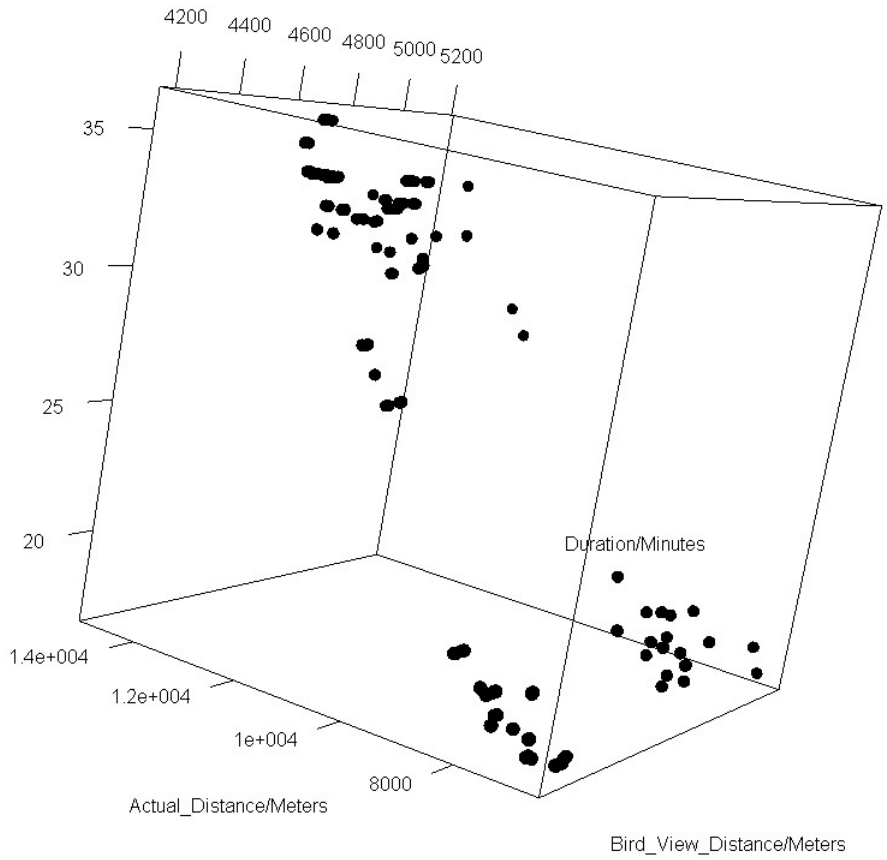

Figure 5.9: In the 3D graph, each point represents a possible waypoints combination for one pair of origin and destination, and is plotted on a 3D graph using R where x-axis represents the bird-view distance, y-axis represents the actual route distance and z axis represents the time duration. Distance is measured in meter and time cost is measured in minutes.

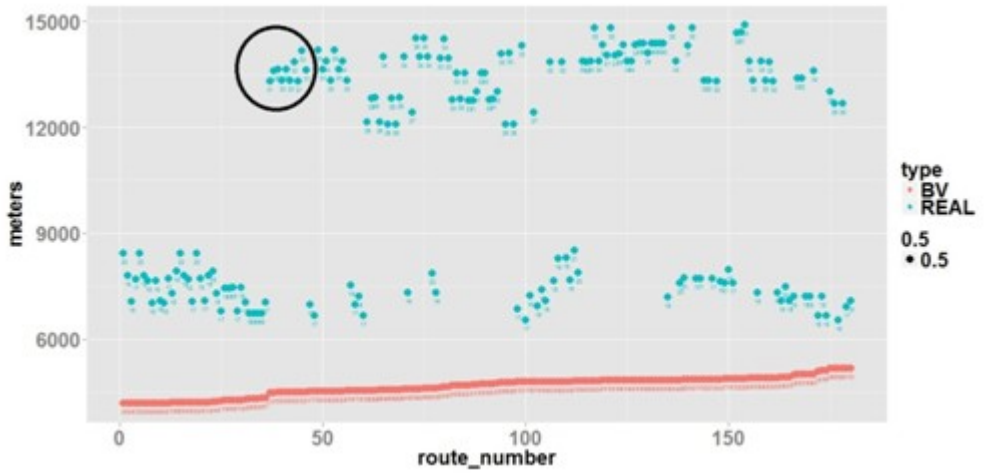

Figure 5.10: The invalid points is circled in the graph, which means it may be invalid by using the strategy.

5.3 Simulation on a JavaScript Application

Finding the invalid points may help to improve the strategy. So in the third part, another experiment will be implemented to answer question in what situation some points are valid while some are not.

Ideally, the actual route distance should be increased with the increasing of bird view distance. However, due to the complexity of the topography, many factors will have an influence on the actual route distance. The experiment will begin with finding the points which are invalid using the proposed strategy and figure out why it is invalid by comparing with valid point. The actual driving distances of some points are rapidly changed and then go back to normal level soon after several points. Such points can be regarded as the invalid points and the points which go back to the normal level can be regarded as valid points. For example, the point within the circle in graph 5.10 can be regarded as invalid points, because those points compared with the previous points, the distance has been rapidly changed and then after several points, the distance goes back to normal level. After selecting those invalid points, those points will be plotted and as it is mentioned before, each point represents for a possible waypoints combination. Hence, the optimal routes through the waypoints of this point will be drawn on the map which can show a more intuitive result.

Basically, the experiment will begin with setting the start and end point located in cities

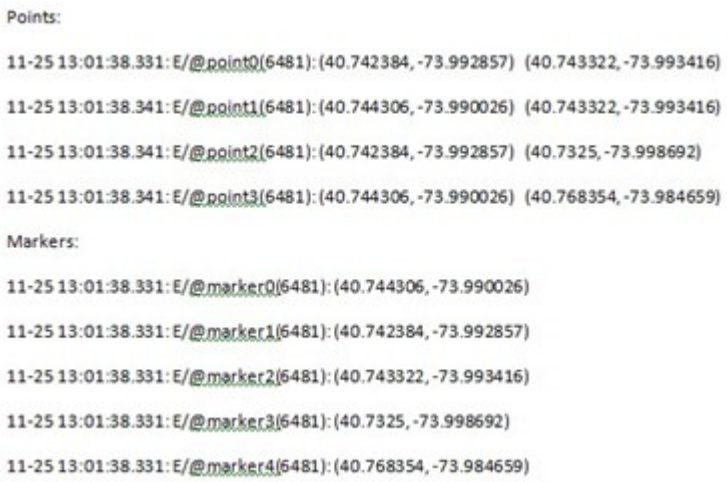

Figure 5.11: In order to analyze the route through multiple waypoints, the location of waypoints and the data used to draw the bird view lines between waypoints need to be retrieved and further used to do the simulation on a JavaScript application.

and rural areas. Both bird view distance and actual route distance will be calculated and then plotted using R showed in figure 5.10, based on the graph and our data validation rules, we can select the valid data and invalid data. Table 5.1 is a partial dataset of actual route distance when drawing figure 5.10, the row (33, 34, 35, 36, 47, and 48) can be regarded as the valid points. Other columns in this table can be marked with invalid data, because the distance of these columns is rapidly increased and goes back to normal after several points. Hence, a pair of invalid and valid points is to be selected. Moreover, the valid and valid point should be closed to each other. Each point stands for a waypoints combination, and then an Android application is designed to retrieve the waypoints location information in these two combinations in which one is for valid point, and the other one is invalid point.

To make it more intuitive and specify each situation, an HTML and JavaScript application is designed to show the maps and markers. The location information will be further used in this application. By using Google Maps JavaScript API v3 and Direction API, both actual route and bird view route can be plotted on the map, showed in figure 5.12.

As it is showed in figure 5.12, blue line is the actual driving route and red line is bird view route. Marker A stands for start point and marker E stays as end point. The rest of markers are used to label three waypoints which should belong to a restaurant, a gas station and a bank.

Table 5.1: Partially Dataset for Drawing the figure 5.10

Route Number	Meter	Time Duration
...	...	...
33	6721	16
34	6723	16
35	6721	16
36	7051	17
37	13297	31
38	13608	31
39	13646	33
40	13335	33
41	13646	33
42	13335	33
43	13842	32
44	13297	31
45	14156	33
46	13609	31
47	6983	17
48	6672	17
...	...	...

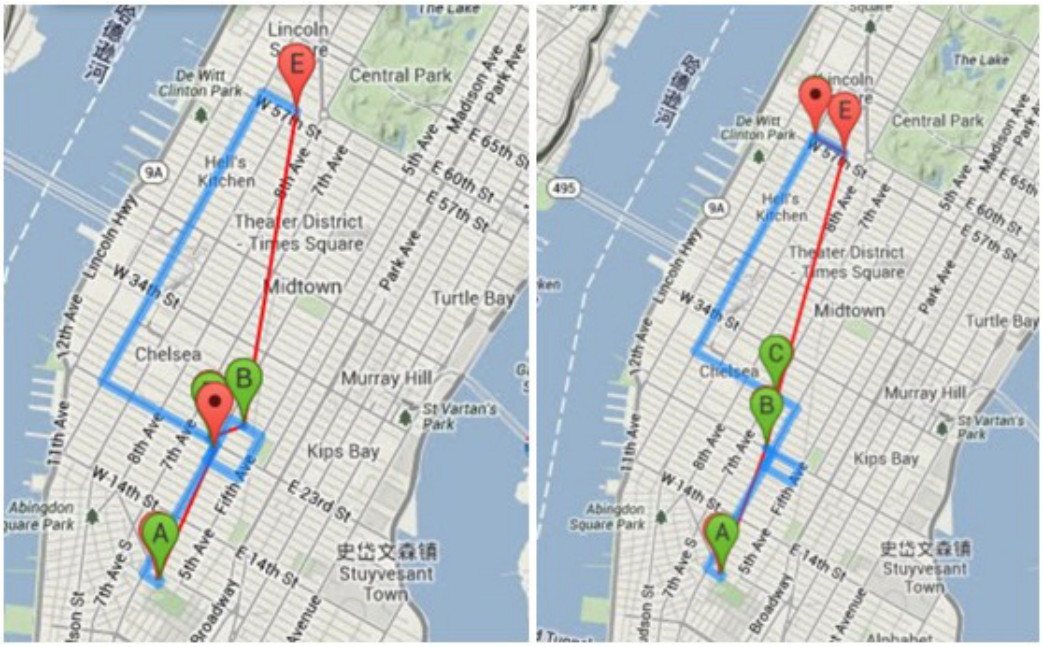

Figure 5.12: In the figure, marker A stands for start point and marker E stays as end point. The rest of markers are used to label three waypoints which should belong to a restaurant, a gas station and a bank. The blue line is the actual driving route and red line is bird view route.

5.4 Simulation Result

First, the experiment will be done at city areas, following are the addresses of the testing dataset:

```
Dataset1:
Start Point:
103 Waverly Place
New York, NY 10011
(212) 777-9515
End Point:
356 West 58th Street
New York, NY 10019
(212) 554-6000

Dataset2:
Start Point:
1401 Saint Joseph Parkway
Houston, TX 77002
End Point:
3204 Ennis Street
Houston, TX 77004
```

In figure 5.12, the actual distance of right graph is longer than left graph, since the waypoints of two pictures are different but closed to each other. To go through all the waypoints, in right picture, in order to get to marker B, one needs to take a turn which causes more covered distance. In figure 5.13, compared with the routes on the left graph, in right graph, one needs to drive on the highway “Gulf Fwy” in order to get to the waypoint located on up-right side which may produce more distance, because when calling Google Direction API, it is very likely to return a route has light traffic and high speed limit.

Then the experiment will be done at place where there is less development, following are the addresses of those places:

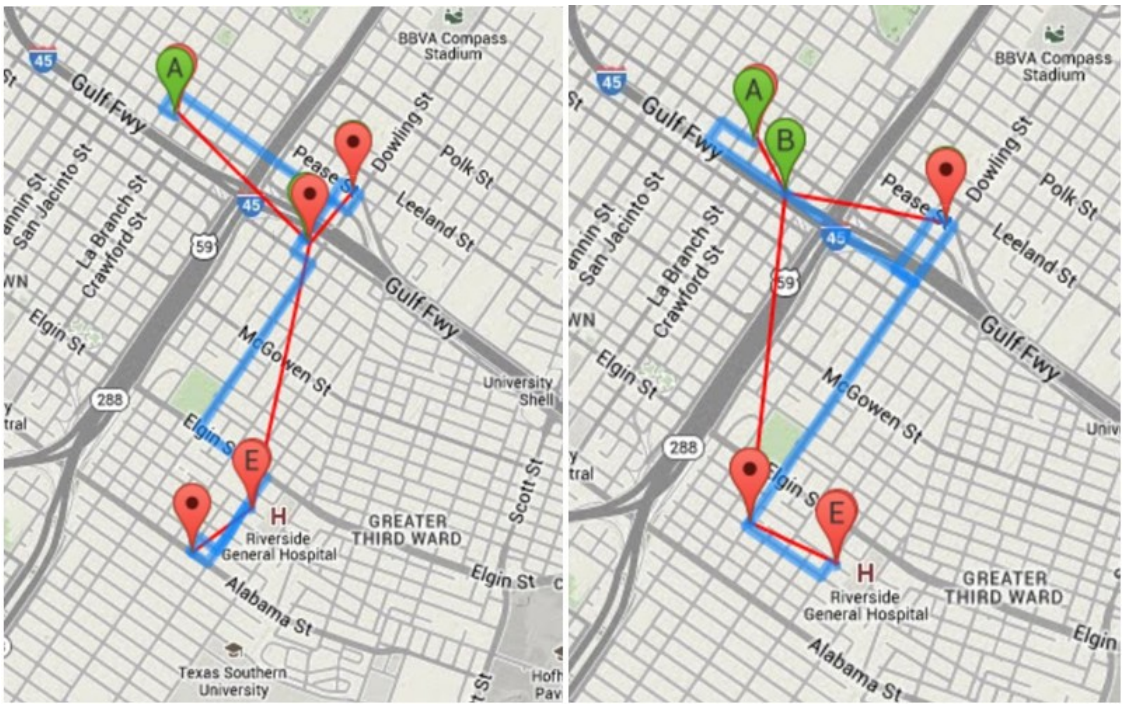

Figure 5.13: In the figure, marker A stands for start point and marker E stays as end point. The rest of markers are used to label three waypoints which should belong to a restaurant, a gas station and a bank. The blue line is the actual driving route and red line is bird view route

Dataset3:

Start Point:

300 East 9th Street

Coralville, IA 52241

(319) 688-4000

End Point:

125 South Madison Street

Iowa City, IA 52242

(319) 335-3513

Dataset4:

Start Point:

2500 Coolidge Road

East Lansing, MI 48823

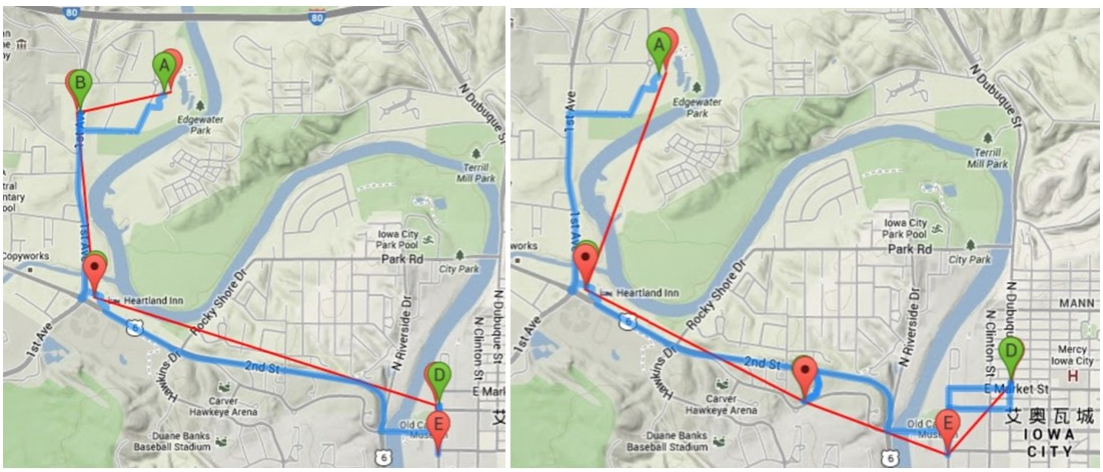

Figure 5.14: In the figure, marker A stands for start point and marker E stays as end point. The rest of markers are used to label three waypoints which should belong to a restaurant, a gas station and a bank. The blue line is the actual driving route and red line is bird view route.

```
(517) 324-2072
End Point:
426 Auditorium Rd,
East Lansing,
MI 48824

Dataset5:
Start Point:
201 Old Main
University Park, PA
End Point:
1555 University Drive
State College, PA 16801
(814) 235-6960
```

From the figure 5.14, the actual route of left graph is longer than right one. Because on the right graph, two waypoints need to be made a detour in order to reach them. Those waypoints produce more distance when calculating the actual route distance. In figure 5.15, on the right graph, there is a river between waypoint C and waypoint D, and waypoint D is on the other

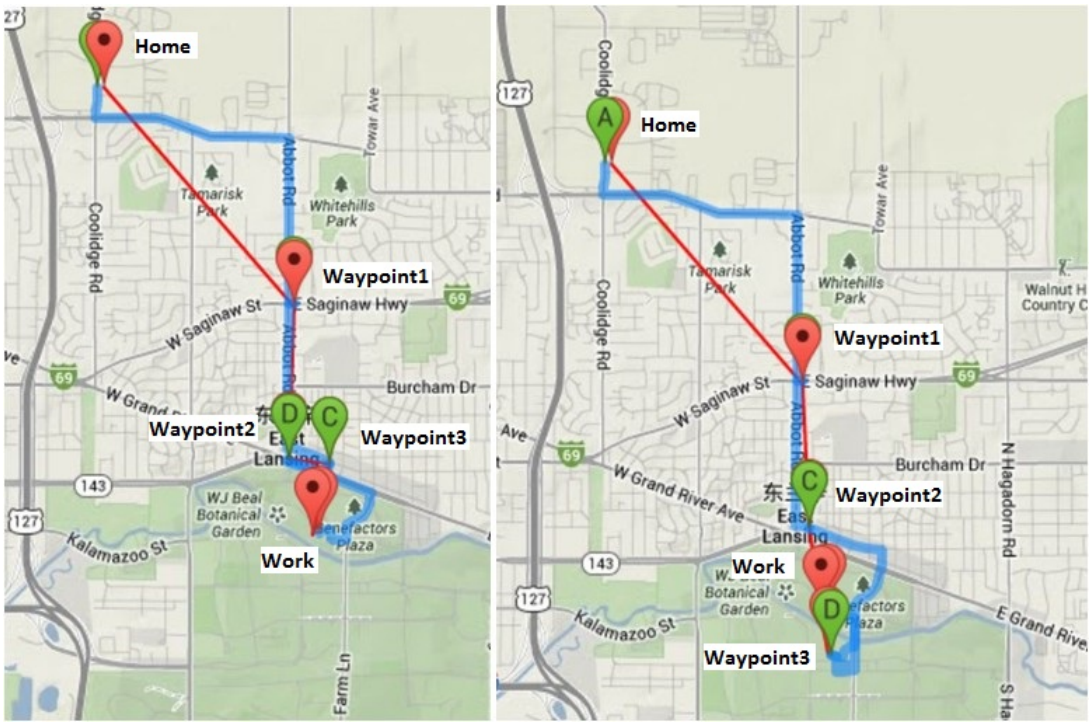

Figure 5.15: In this figure, Marker "home" stands for start point and Marker "work" stays as end point. Marker "Waypoint1", "Waypoint2", and "Waypoint3" are used to label three waypoints which should belong to a restaurant, a gas station and a bank. The blue line is the actual driving route and red line is bird view route.

side of the river, which produces more distance. In figure 5.16, by calling the Google Direction API, it will return the route which requires less driving time duration, however it is usually the case that calling Google Direction API will probably return you a route with a relatively high speed limit which may guide you to a highway and produce more distance.

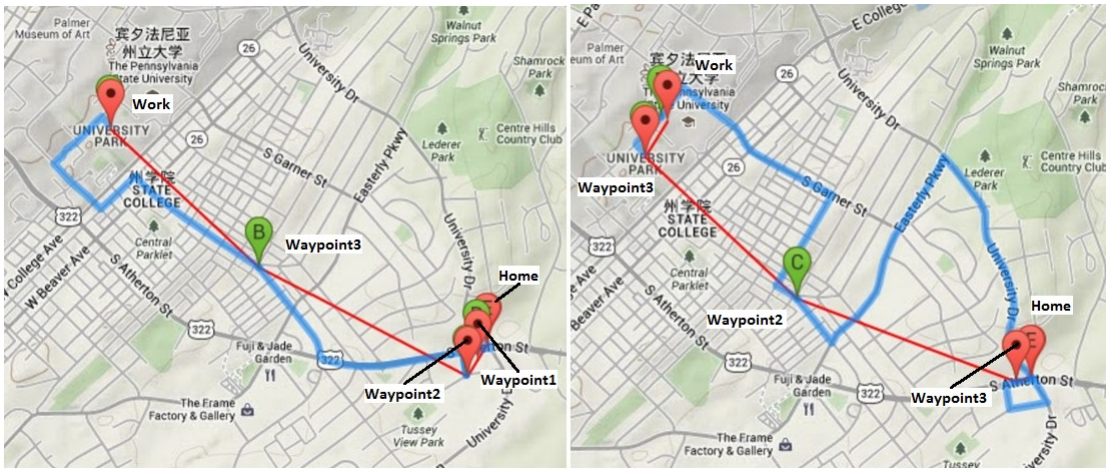

Figure 5.16: In this figure, Marker "home" stands for start point and Marker "work" stays as end point. Marker "Waypoint1", "Waypoint2", and "Waypoint3" are used to label three waypoints which should belong to a restaurant, a gas station and a bank. The blue line is the actual driving route and red line is bird view route.

Chapter 6

Discussion

The 2D graph results from experiment 5.1 show some interesting patterns, and these patterns will be discussed in this chapter.

In figure 5.2, points are grouped by three. Each point stands for a different waypoints combination, in order to better understand the waypoints in the combination, the URL used for API calling should be retrieved out. Figure 6.1 shows URL for requesting directions of such a group.

As the thesis discussed, original and destination equal to start and end points. "optimize=true" means the route returned by calling API is optimized. Different waypoints are divided by "—". Gas station, restaurant and bank are selected as the general types of waypoint, so the first and third waypoints in this group are same, and should belong to a gas station and a bank. Now let's consider the different point which is the second point. Based on the geography location, the point could be easily searched using Google Maps. Point (38.900568,-77.046363), Point (-77.046355, 38.901425) and Point (-77.046225, 38.901425) are "Lindy's Red Lion", "Lindy's Bon Airport" and "Italian Bertucci's Restaurant". These three places are all close to each other, so the actual driving route distances are pretty close and thats why the points seem to be grouped by three on the graph. Some types such as restaurants will return the waypoints which are closed to each other. This usually happens because they may be located in the same mall or same plaza.

Time duration has relationship with both bird-view distance and actual route distance, but the relationship between them is not intuitive.

In figure 5.3, some points which have longer actual route distances but may still cost less time. Different period in a day, different speed limits and other factors may have a great influence on the driving time duration. Because the time duration is based on current traffic

E/url@(5834):
http://maps.googleapis.com/maps/api/directions/json?origin=1101+new+Hampshire+avenue+northwe
st+Washington+DC&destination=2121+I+st+nw+Washington+DC&optimize=true&waypoints=38.905152
,-77.048567|38.900568,-77.046363|38.901425,-77.047515|&sensor=false

E/url@(5834):
http://maps.googleapis.com/maps/api/directions/json?origin=1101+new+Hampshire+avenue+northwe
st+Washington+DC&destination=2121+I+st+nw+Washington+DC&optimize=true&waypoints=38.905152
,-77.048567|38.900568,-77.046355|38.901425,-77.047515|&sensor=false

E/url@(5834):
http://maps.googleapis.com/maps/api/directions/json?origin=1101+new+Hampshire+avenue+northwe
st+Washington+DC&destination=2121+I+st+nw+Washington+DC&optimize=true&waypoints=38.905152
,-77.048567|38.900603,-77.046225|38.901425,-77.047515|&sensor=false

Figure 6.1: This figure shows the url when request the driving routes. Each url stands for a point in a group, so there are three points in the group

conditions, if the traffic is heavy on the route or there is a less speed limit on this route, it could take more time than the one with longer distance.

The best three points of bird-view distances could not always be the actual three shortest routes. Most results show only one or two best bird-view distances that exist in actual three shortest routes and sometimes even no one exists in it. What we want to propose here is that this method can be used as a reference to find the best three routes. Although sometimes the routes of bird view distance have only one or two even not one of them exists in actual best three routes, the method can still be used for selecting the good route (may not be best three). The graph can show bird-view distance is related with actual route distance, and the actual distance is related with driving duration. However, to find an equation or theory to prove it, it is more complex than we could handle.

In figure 5.4, with the increasing number of bird-view distance, the trend of actual distance is also rising. If drawing a trend line to show both of them, we can find that the slopes of these two trend lines are almost same. This usually happens in the city area. Since in the city area, roads are always highly structured and organized. Streets and avenues are crossed at most corners, which means, when you are driving in the city, you do not need to make detour to get to the destination. However, the route could not be always completely straight, so the actual driving route distance is still longer than bird-view route distance.

In figure 5.5, the bird-view distances of these points are divided into two parts. Some of them have a longer distance. This usually happens where there is river between start and

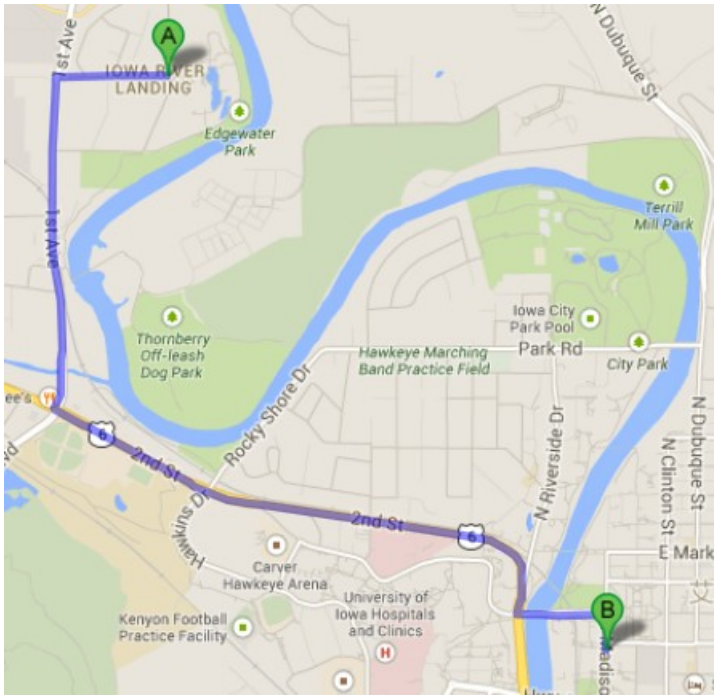

Figure 6.2: A JavaScript application is designed to draw the route on map. A marks the start point and B marks the end point.

destination which means in order to get to the destination you have to drive over the bridge.

Based on figure 5.5, the program plotted the both start (A) and destination (B) on the map view as showed in figure 5.6. There are 5 bridges that need to be covered before arriving at designation. In this case, some places are located on the other side of the river which means when you are trying to reach them, and they are no longer nearby places, because you probably have to make detour to find a bridge. As a result, when such places on the route, it will take longer distance.

The application is currently taking into account all of the places on the route to the user desired endpoint (work). The application works good while producing choices to go to different places. An application like this would help ease the normal hassle of figuring out where to go in order to not be late to work in the morning, find the optimal route or it could help the user discover their new favorite cafes and stores. The application can display the information of different routes which include a route summary, driving time duration and distance. This information can be used for navigation, route selection and further optimization. In current strategy, the best three optimal bird view routes will be sending to calling Google Directions

API, it may get better result if more bird view optimal results is used to call Directions API. Moreover, calculating directions is a time and resource intensive task. So the optimization of calculation process is pending to implement in the future work.

Chapter 7

Conclusion

In this thesis, an application on mobile platform to find the optimal route through multiple intermediate waypoints was implemented. An algorithm has been proposed in the application which helps find nearby places. A strategy has been proposed to give the recommendation routes. Several experiments were done to evaluate the strategy. By using R, 3D graphs are plotted to show the relationship between bird view distance, actual routes distance and driving time duration. Additionally, a JavaScript application was also designed to look at the route on the real maps and clarify what happened behind these graphs.

Basically, the application implemented all the functionality as we described above. It can place three recommended optimal routes while passing through multiple waypoints and display route information on the screen. A notification will be automatically sent when users are ready to go for work and by clicking the notification the application is automatically launched. Users could easily change the work address to any other address and then have the application function to guide them through their chosen stopovers on route to their destination. At the moment, the application only returns recommended three optimal routes, but it is easy to display more routes if a programmer chooses to do so. We doubt that more than three are needed for users to choose from. Studies have shown that when users have too many choices they become overwhelmed and unhappy. Hence, the process should be simplified.

Waypoints and efficient routes are used every day by companies such as FedEx, UPS, DHL, and many more. Those companies use high end equipment to calculate their efficient routes; getting similar functionality into a common user's phone is a great start. The application has many routes that it can take from here, but hopefully it will stay true to its beginnings helping others get to where they need to be quickly.

There could be optimizations made to get even more efficient intermediate waypoints, and

the number of API calls could be reduced if the directions and maps are cached.

The result shows that the best three recommended routes may not be always the top three shortest paths. However, the strategy still can be used to find good results. Based on the experiment in Chapter 5.1, 80% of results show at least one of the best three shortest driving routes will be selected using the application. 90% of results show at least one of the best three shortest time durations will be selected.

Finally, the app should work regardless of internet connection, location, or typos. As it was mentioned before, calculating directions is a time and resource intensive task. Whenever possible, calculate known addresses ahead of time and store the results in a temporary cache when designing the application.

References

[1] B. Hofmann-Wellenhof, H. Lichtenegger, J. Collins, Global Positioning System. Theory and Practice Springer, Wien, Austria, 1993.

[2] D. Kopitz and B. Marks RDS: The Radio Data System Artech House Books, UK, November 1988.

[3] Gartner Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. Retrieved February 13, 2014, from http://www.gartner.com

[4] D.B. Johnson A Note on Dijkstra's Shortest Path Algorithm *Proc. of ACM*, (1973), 385-388

[5] R. Dechter and J. Pearl Generalized Best-First Search Strategies and the Optimality of A* *Proc. of ACM*, (1985), 505-536

[6] B. Moses Sathyaraj, L. C. Jain, A. Finn, and S.Drake Multiple UAVs Path Planning Algorithms: a Comparative Study *Proc. of Fuzzy Optimization and Decision Making*, (2008), 257-267

[7] M. G.H. Bell Hyperstar: A Multi-Path Astar Algorithm for Risk Averse *Proc. of ACM*, (1973), 385-388

[8] Jaillet, Patrick A Priori Solution of a Traveling Salesman Problem in which a Random Subset of the Costomers are Visited In *Operations Research*, (1988)

[9] Manoj Kanta Mainali, Shingo Mabu, Xianneng Li and Kotaro Hirasawa. Optimal Route Planning with Restrictions for Car Navigation Systems. In *IEEE*, (2010), 393–397

[10] Winston Chang Line Graph In *R Graphics Cookbook*, (2007), 50–59.

[11] Dijkstra, E. W. A note on two problems in connexion with graphs In *Proc. of Numerische Mathematik*, (1959).

[12] Bellman, Richard On a Routing Problem In *Proc. of Quarterly of Applied Mathematics*, (1958).

[13] Ford Jr., Lestor R., D.R. Fulkerson Flows in Networks In *Proc. of Princeton University Press*, (1962).

[14] Johnson, Donald. Efficient Algorithms for Shortest Paths in Sparse Networks In *Proc. of Journal of the ACM*, (1977).

[15] Floyd, Robert Algorithm 97 (SHORTEST PATH) In *Proc. of Communications of ACM*, (1977).

[16] Ganti, R. K., Pham, N., Ahmadi, H., Nangia, S., and Abdelzaher, T. F. GreenGPS: a participatory sensing fuel-efficient maps application. In *Proc. of MobiSys'10*, ACM (2010), 151–164.

[17] Patel, K., Chen, M. Y., Smith, I., and Landay, J. A. Personalizing routes. In *Proc. of UIST'06*, ACM (2006), 187–190.

[18] Krumm, J. Where will they turn: predicting turn proportions at intersections. *Personal Ubiquitous Comput. 14*, 7 (Oct. 2010), 591–599.

[19] Ariely, D., Gneezy, U., Loewenstein, G., and Mazar, N. Large Stakes and Big Mistakes. *Review of Economic Studies 76*, 2 (2005), 451–469.

[20] K. Hirasawa, H. Miyazaki, J. Hu and K. Goto. "Discrete Random Search Method "RasID-D" for Optimization Problems" Journal of Signal Processing, 7 (2004), 351-358.

[21] Blevis, E. Sustainable interaction design: invention & disposal, renewal & reuse. In *Proc. of CHI'07*, ACM (2007), 503–512.

[22] Ing-Chau Chang, Hung-Ta Tai, Dung-Lin Hsieh, Feng-Han Yeh, Siao-Hui Chang Design and Implementation of the Travelling Time- and Energy-Efficient Android GPS Navigation App with the VANET-based A* Route Planning Algorithm *Proc. of IEEE*, (2013), 85–92.

[23] Chetty, M., Tran, D., and Grinter, R. E. Getting to green: understanding resource consumption in the home. In *Proc. of UbiComp'08*, ACM (2008), 242–251.

[24] David M. Mark Automated Route Selection For Navigation *Proc. of IEEE*, (1986)

[25] Lin Jun, Du Jumping, Wang Su Study on Travel Route Intelligent Navigation System Based on WEBGIS *Proc. of International Conference on Artificial Intelligence and Computational Intelligence*, (2009), 560–564

[26] Gonder, J., Earleywine, M., and Sparks, W. Final report on the fuel saving effectiveness of various driver feedback approaches. *National Renewable Energy Laboratory Milestone Report NREL/MP-5400-50836* (2011).

[27] A. Maruyama, N. Shibata, Y. Murata, K.Yasumoto and M. Ito. A Personal Tourism Navigation System to Support Traveling Multiple Destinations with Time Restrictions In *Proc. of the 18th International Conference on Advanced Information Networking and Application*, (2004), 18-21.

[28] M. Rudack, M. Meincke and M. Lotf On the Dynamics of Ad Hoc Networks for Inter Vehicle Communication In*Proc. of the ICWN*, (2002)

[29] H. Kanoh and N. Nakamura. Route Guidance with Unspecified Staging Posts Using Genetic Algorithm for Car Navigation Systems In *Proc. of the IEEE Conference on Intelligent Transportation Systems*, (2000), 119-124.

[30] Google Inc.. Introduction to the Google Maps Android API v2. *Google Maps Android API V2.* Retrieved September 20, 2013, from http://www.google.com

[31] Google Inc.. The Google Directions API. *Google Maps API Web Services.* Retrieved September 26, 2013, from http://www.google.com

[32] Google Inc.. Getting Started. *Google Places API.* Retrieved September 30, 2013, from http://www.google.com

[33] Google Inc.. Notification. *Developer Guide.* Retrieved Oct 11, 2013, from http://www.google.com