



**USING DIJKSTRA ALGORITHM IN CALCULATING ALTERNATIVE
SHORTEST PATHS FOR PUBLIC TRANSPORTATION WITH TRANSFERS
AND WALKING**

CASE STUDY: ANKARA

HAITHAM LATIF HASSAN AL-TAMEEMI

JUNE 2014

**USING DIJKSTRA ALGORITHM IN CALCULATING ALTERNATIVE
SHORTEST PATHS FOR PUBLIC TRANSPORTATION WITH TRANSFERS
AND WALKING
CASE STUDY: ANKARA**

**A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES OF
ÇANKAYA UNIVERSITY**

**BY
HAITHAM LATIF HASSAN AL-TAMEEMI**

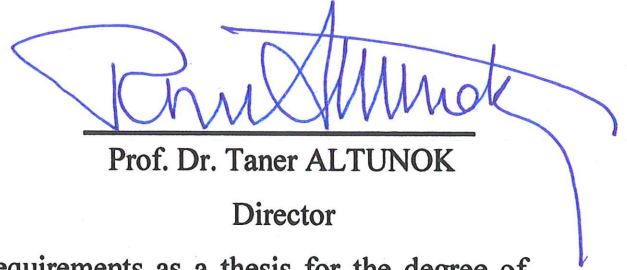
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
MATHEMATICS AND COMPUTER SCIENCE
INFORMATION TECHNOLOGY PROGRAM**

JUNE 2014

Title of the thesis: Using Dijkstra Algorithm in Calculating Alternative Shortest Paths for Public Transportation with Transfers and Walking Case Study: Ankara

Submitted by: Haitham Latif Hassan AL-TAMEEMI

Approval of the Graduate School of Natural and Applied Science, Çankaya University.



Prof. Dr. Taner ALTUNOK

Director

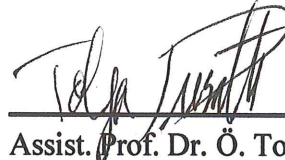
I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.



Prof. Dr. Billur KAYMAKCALAN

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.



Assist. Prof. Dr. Ö. Tolga PUSATLI

Supervisor

Examination date: 27/06/2014

Examination Committee Members:

Assist. Prof. Dr. Ö. Tolga PUSATLI (Çankaya Univ.)

Assoc. Prof. Dr. Hadi Hakan MARAŞ (Çankaya Univ.)

Ahmet DABANLI (M.S.) (Başarsoft)



STATEMENT OF NON PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct, I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Haitham Latif Hassan AL-TAMEEMI

Signature : 

Date : 27/06/2014

ABSTRACT

USING DIJKSTRA ALGORITHM IN CALCULATING ALTERNATIVE SHORTEST PATHS FOR PUBLIC TRANSPORTATION WITH TRANSFERS AND WALKING CASE STUDY: ANKARA

AL-TAMEEMI, Haitham Latif Hassan

M.Sc., Department of Mathematics and Computer Science
Information Technology Program

Supervisor: Assist. Prof. Dr. Ö. Tolga PUSATLI

June 2014, 72 pages

In this study, the possibility of using the Dijkstra algorithm in calculate shortest paths and using GIS to connect transport lines for increasing efficiency in the accessibility to shortest paths was studied. The results are presented as spatial maps. Basically, the research question this thesis addresses is: "Can Iraqi authorities use graph theory and shortest path algorithm to build a system to develop a public transit system including exchanging lines and walking, based on the experiences of transport systems in Turkey?"

As a research methodology, Ankara's mass transportation has been examined as considerable similarities are found between Ankara and Baghdad in terms of population in number and in density. Hence, an introduction is given on both cities. Additionally, information technologies to investigate and promote public transport systems in the literature are visited. As gathering spatial data has required considerable time and resources, sample datasets of Ankara's mass transport system are used to run demonstrative applications to highlight potential problems in the technical side.

The surveyed literature also includes graph theory, transportation network, shortest path problems and GIS usage in the public transportation.

The used MapInfo Professional and MapBasic software tools are available on the market. Specific application programs are developed in MapBasic to ease analyses and implementation in MapInfo. Additionally, programs are coded in C++ to apply Dijkstra algorithm to feed data to GIS software. Limitations are also reported to open further research avenues including other algorithms for path optimization.

The results show that Dijkstra algorithm can be successfully applied to reform and design mass transportation systems. An unfortunate finding underlined that it is too early to jump to application design as the primary problem is to collect reliable and up to date spatial data of Baghdad.

Keywords: Ankara, Baghdad, Dijkstra Algorithm, GIS, Graph Theory, Shortest Path, Public Transportation System.

ÖZ

YÜRÜME DAHİL AKTARMALI TOPLU TAŞIMADA ALTERNATİF KISAYOL HESAPLAMASI İÇİN DİJKSTRA ALGORİTMASI KULLANIMI ÇALIŞMA KONUSU: ANKARA

AL-TAMEEMI, Haitham Latif Hassan
Matematik-Bilgisayar Anabilim Dalı
Bilgi Teknolojileri Yüksek Lisans Programı
Yrd. Doç. Dr. Ö. Tolga PUSATLI

Haziran 2014, 72 Sayfa

Bu çalışmada, Dijkstra algoritması kullanılarak kısa yol hesaplanmasının ve ulaşım hatlarının bağlanmasında CBS kullanımının etkinliği arttırması işlenmiştir. Sonuçlar mekansal veri haritaları şeklinde sunulmuştur. Temel olarak, tezin hedef aldığı araştırma sorusu: "Irak, çizge kuramı ve kısa yol algoritması ve Türkiye'deki yürüme dahil aktarmalı hat değiştirme deneyimlerini kullanarak toplu taşımacılık sistemi kurabilir mi?"

Bağdat'la Ankara arasında nüfus yoğunluğu ve çokluğu bakımından önemli benzerlikler görüldüğünden, araştırma yöntemi olarak, Ankara'nın toplu taşınması incelenmiştir. Bu sebepten, her iki şehir hakkında tanıtıcı bilgi verilmiştir. Ek olarak,

literatürde toplu taşıma sistemlerini incelemek ve iyileştirmek için kullanılan bilgi teknolojileri taranmıştır. Bağdat'ın mekansal verisini almak önemli miktarda zaman ve kaynak gerektirdiğinden, Ankara toplu taşıma sistemin örnek verisi toplanmış ve bu veriler, olası teknik sorunları açığa çıkarmak için örnek uygulamalarda kullanılmıştır.

Taranan literatür çizge kuramı, taşıma ağları, kısa yol ve toplu taşımacılıkta CBS kullanımını da içermektedir.

Kullanılan MapInfo Professional ve MapBasic programları piyasada olan uygulamalardır. MapInfo'da uygulamaları kolaylaştırabilmek için, MapBasic'de konuya yönelik uygulamalar geliştirilmiştir. Ek olarak, Dijkstra algoritmasını uygulamak ve CBS yazılımına girdi oluşturmak için C++ programlama dilinde uygulama yazılmıştır. Tezdeki sınırlamalar, örneğin başka algoritmalar, olası devam projeleri için rapor edilmiştir.

Sonuçlar gösteriyor ki Dijkstra algoritması, toplu taşıma sistemlerinin iyileştirilmesinde ve tasarımında kullanılabilir. Bir talihsiz bulgu da gösteriyor ki doğrudan uygulama tasarımına geçmek için çok erken çünkü birincil sorun Bağdat'ın doğru ve güncel mekansal verisinin toplanmasıdır.

Anahtar Sözcükler: Ankara, Bağdat, Dijkstra Algoritması, CBS, Çizge Kuramı, Kısayol, Toplu Taşımacılık Sistemi.

ACKNOWLEDGEMENT

First of all, praise to GOD "ALLAH" on all the blessings, one of these blessings was the help in achieving this research to its end.

I would like to express my sincere gratitude to my thesis advisor Assist. Prof. Dr. Ö. Tolga PUSATLI for his continuous support of my Master of Science studies and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me all the time of my research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master of Science studies.

Sincere thanks are also extended to Başarsoft staff especially Mr. Ahmet DABANLI for his valuable support throughout my thesis and project.

Finally, my special thanks to my family for their endless and continuous encourage and support throughout the years.

TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT.....	iv
ÖZ.....	vi
ACKNOWLEDGEMENTS.....	viii
TABLE OF CONTENTS.....	ix
LIST OF FIGURES.....	xi
LIST OF TABLES.....	xiv
LIST OF ABBREVIATIONS.....	xv

CHAPTERS:

1. INTRODUCTION.....	1
1.1. Baghdad - Iraq.....	1
1.2. Ankara-Turkey.....	4
1.3. Purpose and Scope.....	8
1.4. Motivation.....	8
1.5. Research Question.....	9
1.6. Software Used.....	9
1.7. Obtaining Data.....	10
1.8. Research Methods.....	14
2. BACKGROUND AND LITERATURE REVIEW.....	15
2.1. Graph Theory.....	15

2.2.	Transportation Networks.....	16
2.3.	Shortest Path.....	16
2.4.	Dijkstra Algorithm.....	17
2.5.	Geographic Information System (GIS).....	21
2.6.	Literature Review.....	24
2.7.	Problems.....	32
3.	SAMPLE APPLICATION OF MASS TRANSPORT SYSTEM IN ANKARA.....	37
3.1.	Data Analyses.....	37
3.2.	Implementation and Addressing the Problems.....	49
4.	CONCLUSION.....	67
4.1.	Findings and Results.....	67
4.2.	Limitations.....	68
4.3.	Future Studies.....	70
4.4.	Conclusion.....	72
	REFERENCES.....	R1
	APPENDICES.....	A1
	A. MapBasic Code.....	A1
	B. C++ Code.....	B1
	C. CURRICULUM VITAE.....	C1

LIST OF FIGURES

FIGURES

Figure 1	Iraq borders [1]	1
Figure 2	Cary in Baghdad	2
Figure 3	Baghdad roads.....	4
Figure 4	Turkey.....	5
Figure 5	Ankara.....	5
Figure 6	Underground rail system lines	6
Figure 7	Construction in progress rail system lines	7
Figure 8	MapInfo, MapBasic, and Bloodshed websites.....	9
Figure 9	Transport lines of Ankara	10
Figure 10	Sample bus stops data	11
Figure 11	MapInfo and select datasets to open	11
Figure 12	Ankara bus stops.....	12
Figure 13	Connected among bus stops (bus line)	13
Figure 14	ank_stops and ank_lines layers.....	13
Figure 15	Dijkstra algorithm example	18
Figure 16	Routes to covered all nodes	20
Figure 17	Geometry types.....	22
Figure 18	Non-spatial data for bus sops.....	23
Figure 19	An example on accessibility method implementation steps [23]	25

FIGURES

Figure 20	Homonymous stops and different types of public transport.....	30
Figure 21	Source and destination points as bus stops.....	32
Figure 22	Destination and source are free points.....	33
Figure 23	Destination, source points and the path.....	34
Figure 24	Connections among bus stops.....	35
Figure 25	Bus stops relationship by walking.....	37
Figure 26	Create new table and add to the current mapper.....	38
Figure 27	Modify Table Structure text box.....	39
Figure 28	Open MapBasic window from MapInfo.....	40
Figure 29	Query implementation to achieve the relationship by walking.....	40
Figure 30	Choose Update Column.....	41
Figure 31	Requirement to update.....	41
Figure 32	Update Column text box.....	41
Figure 33	Relationship table.....	42
Figure 34	Represent relationship as a layer.....	42
Figure 35	Another relationship as a layer.....	43
Figure 36	Choose Buffer from Table menu.....	43
Figure 37	Detect buffer object and store the result.....	44
Figure 38	Creating new table.....	44
Figure 39	Complete creating the new table.....	45

FIGURES

Figure 40	Detect the buffer criteria	46
Figure 41	Data aggregation	46
Figure 42	SQL Select text box	47
Figure 43	Buffer table layer	48
Figure 44	Buffer table result	49
Figure 45	SetBusStops button pad	50
Figure 46	Several nearest bus stop	51
Figure 47	Choose another nearest bus stop	51
Figure 48	Example for choose a bus stop	52
Figure 49	Push first button	53
Figure 50	NearestStationStart text file	54
Figure 51	Content of NearestStationStart text file	55
Figure 52	Push second button	56
Figure 53	NearestStationFinal text file	57
Figure 54	Content of NearestStationFinal text file	57
Figure 55	Create new project and choose windows application	59
Figure 56	FinalResult text file.....	63
Figure 57	Push Third button in SetBuStops pad	64
Figure 58	Shortest path	66
Figure 59	Longest path.....	66

LIST OF TABLE

TABLES

Table 1	Implementation Dijkstra algorithm	19
Table 2	Shortest path and path from node A to all nodes	20

LIST OF ABBREVIATIONS

APSP	All Pair's Shortest Path
ASCII	American Standard Code for Information Interchange
CE	Cross-Entropy
CPM	Critical Path Method
CPU	Central Processing Unit
CSP	Circular Shortest Path
DBMS	Database Management System
EGO	The Electricity Gas Bus General Directorate of Ankara
ESRI	Environmental Systems Research Institute
GIS	Geographic Information System
GPS	Global Positioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
ISM	Interpretive Structural Modeling
LTCM	Least Transfer Cost Model
MBTA	Multiple Backtracking Algorithms
MILP	Mixed Integer Linear Programming
RAM	Random Access Memory
SQL	Structured Query Language
SSSP	Single Source Shortest Path
TNS	Transfer Between a Pair of Neighbor Stops
TSS	Transfer at the Same Stop
TUS	Transfer Between a Pair of Up-down Stop

CHAPTER I

INTRODUCTION

In this thesis, the technical possibilities of optimizing mass transport systems in Iraq are investigated. First, a short background information regarding Iraq is given to make the reader familiar to the country.

1.1 Baghdad - Iraq



Figure 1 Iraq borders [1]

Iraq is one of the countries in the West of Asia, and overlooks the Arabian Gulf, and has borders with both Turkey, Iran, Kuwait, Saudi Arabia, Jordan and Syria [1] (Figure 1).

Baghdad is the capital of Iraq, and the official language in Iraq is Arabic.

The one of the important feature in Iraq are rivers, and it has two rivers Euphrates and the Tigris. In addition, we can see mountains distribution in the north, as well as the desert can find it in the west [1].

The first attempts to create public transport system in Iraq were in 1889 by the Ottoman governor Midhat Pasha. However, this system depends on the establishment of a railway called Tram, and used the double deck wooden carts pulled by horses called Cary (Figure 2) [2].



Figure 2 Cary in Baghdad

Following this attempt, the real milestone to build public transport was at the beginning of 1938 when Al-Amana was established as a mass transport company by the decree of the Baghdad mayor; the route was called "Cary" or as known publicly "Al-Amana". Today, this company is called "State Company for Travellers and Delegates Transportation" [3].

After many consecutive wars, the infrastructure of this company including bus stops, and most buses were destroyed in 2003 during American invasion. These series of

devastation have made Iraq one of the countries in the need to rebuild its infrastructure of mass transport [4].

The population of the capital Baghdad, as of 2009, statistics is approximately 7,180,889, making it the largest city in Iraq [5].

When we have a look at the geography of Baghdad, we see flat lands not so problematic for a variety of mass transport. In addition, Tigris river divides the city into two parts, making it easier to establish water transportation system as well (Figure 3).

Unfortunately, we could not find any reliable spatial/nonspatial data regarding the transportation system of Baghdad hence we choose Ankara as a possible case study for this thesis.

Briefly, we have used Dijkstra algorithm to optimize routes in public transportation to promote the system. Before we get into the detail, it is beneficial to mention the following requirements for any viable transport project in Iraq.

The important infrastructures needed in Iraq are [6]:

1. Rehabilitation of the main roads and building new bus stations.
2. Acquiring modern buses suitable to the Iraqi streets.
3. Creating a modern and easy network of routes covering all the areas of Baghdad.
4. Scheduling the time of buses based on the people's needs, especially at peak time of the day.
5. Study the traffic congestion in Iraqi streets which is useful to plan optimal routes in addition to the usage for transport scheduling.
6. Use information technology to monitor/promote the public transport system.
7. Allocating resources for all of the above.

When we want to develop the public transport, we must think of redistributing the bus stops in a way that fit to the needs of passengers. One of the most important parts in the infrastructure construction is to focus on the use of information technology to develop this field.

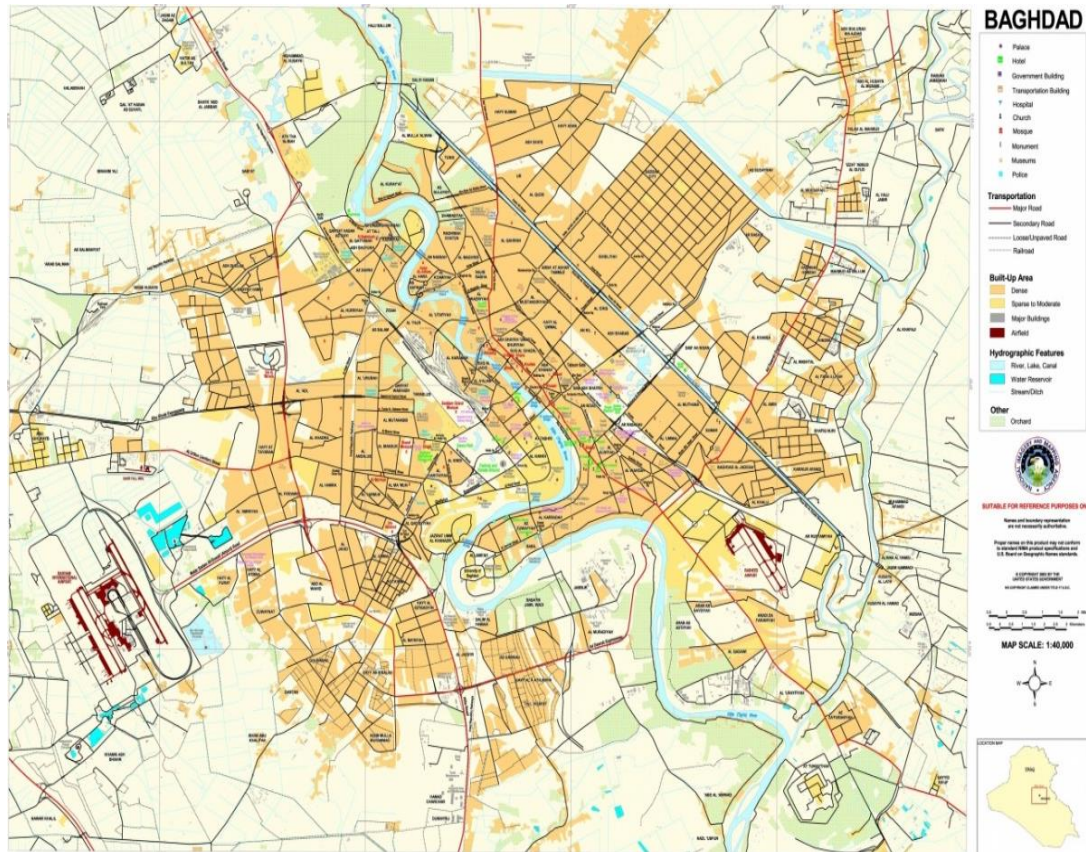


Figure 3 Baghdad roads

1.2 Ankara-Turkey

Ankara one of the largest cities located it in the central of Anatolia, and it is a capital of Turkey (Figure 4, 5). The modern planning of Ankara was produced by a German city planner named Carl Christoph Lörcher in 1924. Ankara is a center of Turkish government, as well as it considered one of the most important commercial and industrial city. In addition, the foreign embassies and diplomatic staff are located in the city. The population of Ankara in 2013 was about 5,045,083 [7].



Figure 4 Turkey

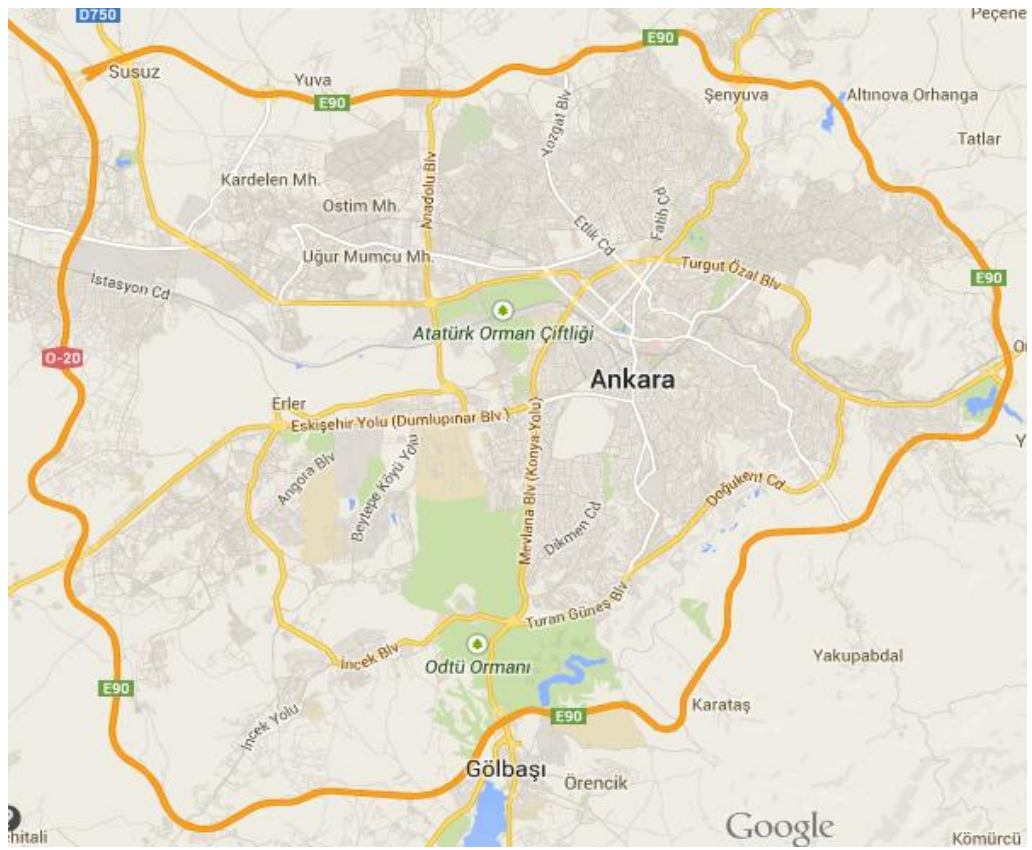


Figure 5 Ankara

The population growth and increasing demands for transportation lead to the development of the modern transportation system to cover all points in the city. For this reason we can see in Ankara several transportation system used to meet all people's needs. The city's public transportation system is inexpensive and easy to use. Municipality buses, public buses, Dolmuş (mini buses), and subway are among Ankara's public transportation alternatives.

The Electricity Gas Bus General Directorate (EGO) [8] is the company that directs the public transportation in Turkey such as subway system and public buses. The Ankara Metro is the rapid transit system and consists of several lines.

The current operating rail system lines (Figure 6) are:

- Ankara (A1): from Dikimevi to Aşti; it is about 8.52 km line, and contains 11 stations.
- Ankara Metro (M1): from Kizilay to Batıkent; it is about 14.66 km line, and contains 12 stations.
- Ankara Metro (M2): from Kizilay to Çayyolu; it is about 16.59 km line, and contains 11 stations.
- Ankara Metro (M3): from Batıkent to Sincan/Torekent; it is about 15.36 km line, and contains 11 stations.

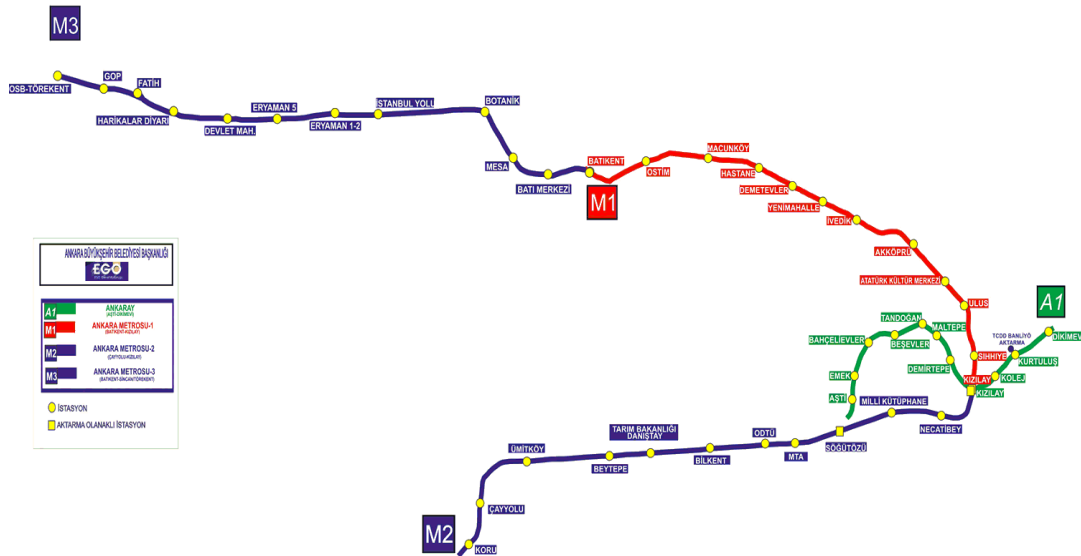


Figure 6 Underground rail system lines

Another rail system line is currently being constructed (Figure 7):

- Ankara Metro (M4): from Tandogan to Kecioren; it is about 10.58 km line, and contains 11 stations.

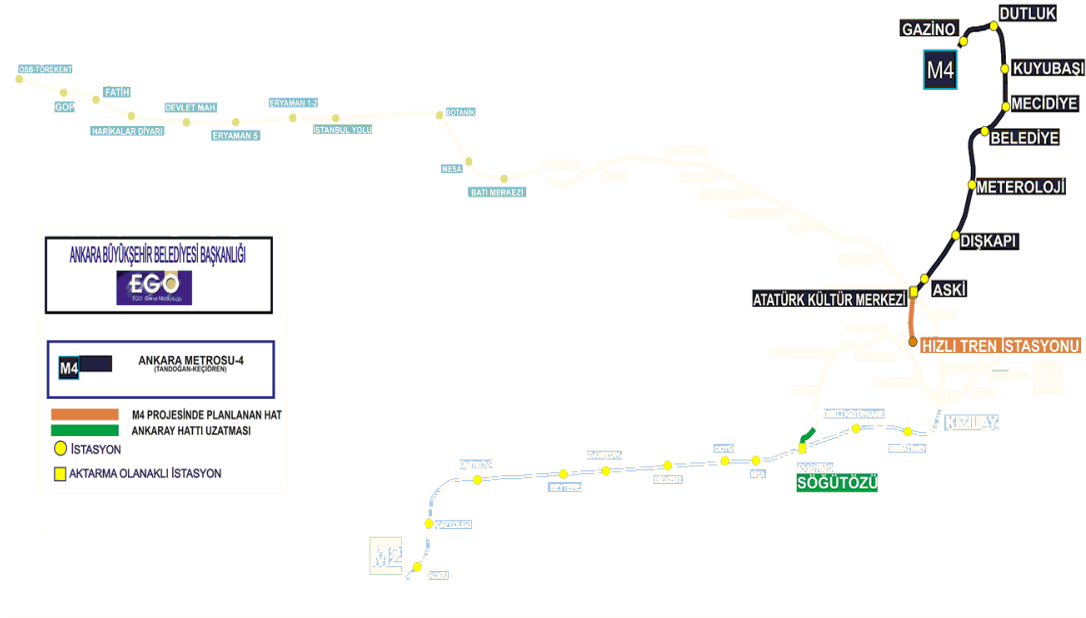


Figure 7 Construction in progress rail system lines

In addition there is planned a new rail system line from Kizilay to Esenboga called Ankara Metro (M5).

In addition to the subway, the buses working to serve the transportation system in Ankara are operated by EGO company. The buses first started to be used in Ankara in 1930, when 100 buses were purchased from the Soviet Union. In 2001, the subway and city bus systems were integrated to use a single type of fare ticket.

In 2005, 50 diesel and 400 natural gas buses were purchased. In 2006, vehicle tracking and automation system was activated. In the present, EGO company contains 1.829 buses with several vehicle types [8].

1.3 Purpose and Scope

This dissertation is about the possibility of the use of geographical information systems to develop a system for finding alternative routes from one location to the target destination with the shortest alternative path in the public transportation systems.

In addition to MapInfo, an application developed in C++ language was used to find the shortest path between any two points in the transport network, to extract more than one path between these points, and to recognize the difference between the shortest and longest paths. Related to the content, we have employed the graph theory, more specifically the algorithm of shortest path to detect the shortest path, so we can use it to reduce the time and the cost to reach the destination point.

In this thesis, firstly we provided basic information of GIS, graph theory, shortest path problem and algorithms in chapter 2. The survey of some of the reports covering the use of geographic information, graph theories and the development the public transport systems is provided in section 2.6.

The case study of this dissertation is about 11 public lines in Ankara (section 1.7). We used the Dijkstra algorithm to find the longest and shortest paths between any specific points (section 3.2), and the problems faced are listed in section 2.7.

1.4 Motivation

The researcher is a teacher in Iraq, always interested about student's lives and their problems. One of the most important and biggest problems is student's access to school in short time with minimal effort and cost.

As seen above, Iraq and Turkey have more than one type of transportation system in addition to increasing demand for transport in both of them. Additionally, the researcher attends Cankaya University in Ankara which gives the opportunity to study local transportation system. Given these, the researcher decided to choose Ankara as a case study.

1.5 Research Question

As a purpose of this thesis, the researchers try to answer this question:

Can Iraq authorities use graph theory and shortest path algorithm to build a system to develop the public transit system including exchanging lines and walking, based on the experiences of transport system in Turkey?

1.6 Software Used

MapInfo Professional Version 11.5 and MapBasic version 11.5 were used to represent the bus stops and lines and to write codes respectively [9]. In addition, C++ IDE by Bloodshed Software [10] was used to calculate the shortest paths based on Dijkstra Algorithm, and to create text files which used throughout the thesis (Figure 8).

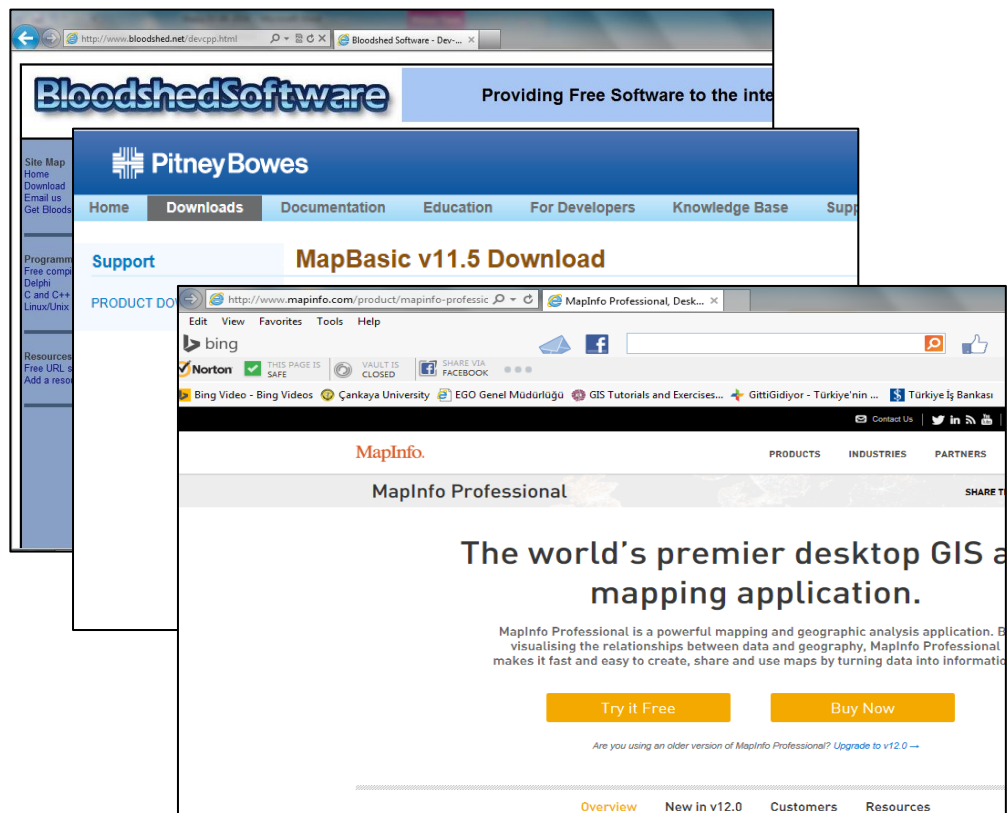
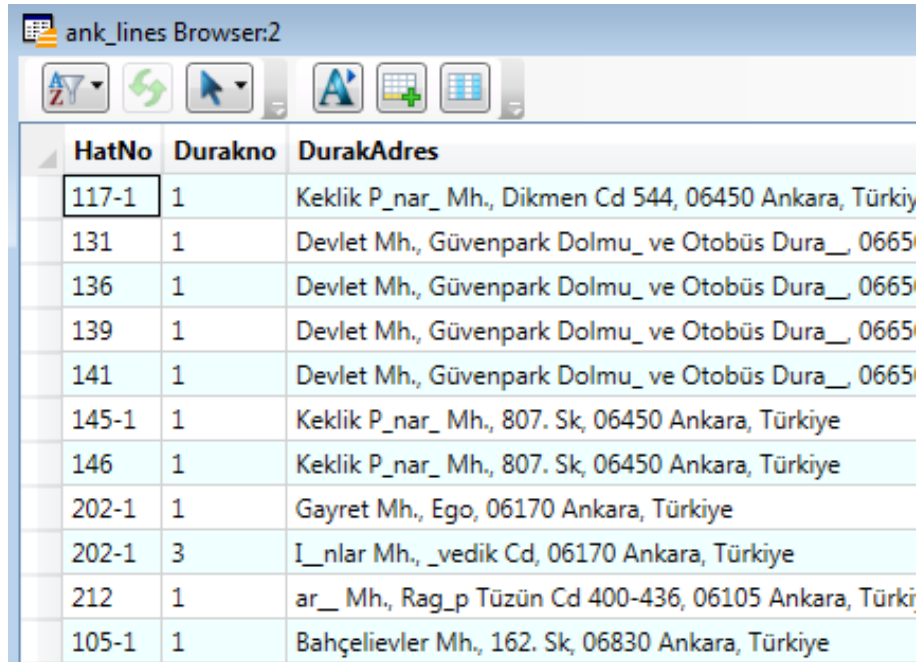


Figure 8 MapInfo, MapBasic, and Bloodshed websites

1.7 Obtaining Data

The sample data were obtained from Başarsoft company; these datasets may not be up-to-date as they are used only as sample. These datasets present:

- 1- Transport lines of Ankara (ank_lines): 11 routes were used in this research, and they are (Figure 9):
 - a. 117-1
 - b. 131
 - c. 136
 - d. 139
 - e. 141
 - f. 145-1
 - g. 146
 - h. 202-1
 - i. 202-1
 - j. 212
 - k. 105-1



HatNo	Durakno	DurakAdres
117-1	1	Keklik P_nar_ Mh., Dikmen Cd 544, 06450 Ankara, Türkiy
131	1	Devlet Mh., Güvenpark Dolmu_ ve Otobüs Dura__, 0665
136	1	Devlet Mh., Güvenpark Dolmu_ ve Otobüs Dura__, 0665
139	1	Devlet Mh., Güvenpark Dolmu_ ve Otobüs Dura__, 0665
141	1	Devlet Mh., Güvenpark Dolmu_ ve Otobüs Dura__, 0665
145-1	1	Keklik P_nar_ Mh., 807. Sk, 06450 Ankara, Türkiye
146	1	Keklik P_nar_ Mh., 807. Sk, 06450 Ankara, Türkiye
202-1	1	Gayret Mh., Ego, 06170 Ankara, Türkiye
202-1	3	I_nlar Mh., _vedik Cd, 06170 Ankara, Türkiye
212	1	ar_ Mh., Rag_p Tüzün Cd 400-436, 06105 Ankara, Türki
105-1	1	Bahçelievler Mh., 162. Sk, 06830 Ankara, Türkiye

Figure 9 Transport lines of Ankara

2- As a sample set of Ankara bus stops (ank_stops) we have selected 608 stops in this thesis (Figure 10).

StationNo	HatNo	Durakno	DurakAdres
133	117-1	46	Harbiye Mh., Dikmen Cd 232-242, 06450 Ankara, Türk
134	117-1	47	Ayd_nlar Mh., Dikmen Cd 262-278, 06450 Ankara, Tür
135	117-1	48	Ayd_nlar Mh., Dikmen Cd 294, 06450 Ankara, Türkiye
136	117-1	49	Osman Temiz Mh., Dikmen Cd 336, 06450 Ankara, Tür
137	117-1	50	Malazgirt Mh., Dikmen Cd 357-359, 06520 Ankara, Tü
138	117-1	51	Huzur Mh., Dikmen Cd 386, 06520 Ankara, Türkiye
139	117-1	52	Huzur Mh., Dikmen Cd 394-398, 06520 Ankara, Türkiy
140	117-1	53	Karap_nar Mh., Dikmen Cd 420-426, 06450 Ankara, Tü
141	117-1	54	Karap_nar Mh., Dikmen Cd, 06800 Ankara, Türkiye
142	117-1	55	Akp_nar Mh., Dikmen Cd 498-502, 06800 Ankara, Türk
143	117-1	56	Keklik P_nar_ Mh., Dikmen Cd 524, 06800 Ankara, Türk
600	117-1	57	Keklik P_nar_ Mh., Dikmen Cd 544, 06450 Ankara, Türk
144	131	1	Devlet Mh., Güvenpark Dolmu_ ve Otobüs Dura_, 066
145	131	2	Devlet Mh., Dikmen Cd, Ankara, Türkiye

Figure 10 Sample bus stops data

The previous data are the main data obtained from Başarsoft company, and they are presented as layers by using MapInfo software. Figure 11 shows how to open MapInfo and select the data.

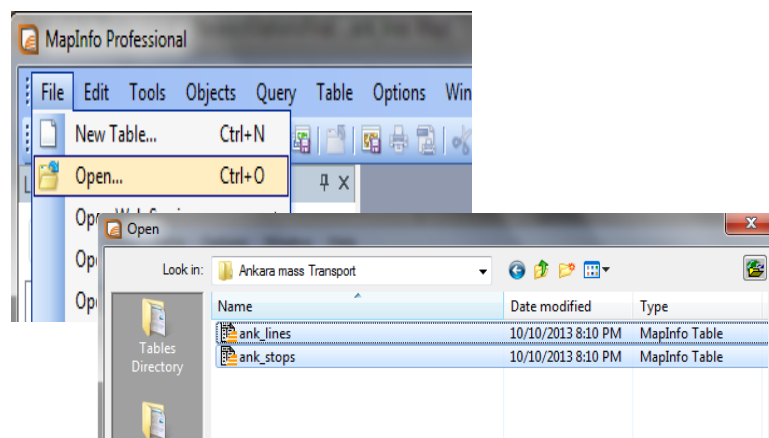


Figure 11 MapInfo and select datasets to open

Figures 12 and 13 show two layers. First layer represents bus stops; second layer represents connection among the bus stops in the same route (bus line). Figure 14 represents these layers as overlaid.

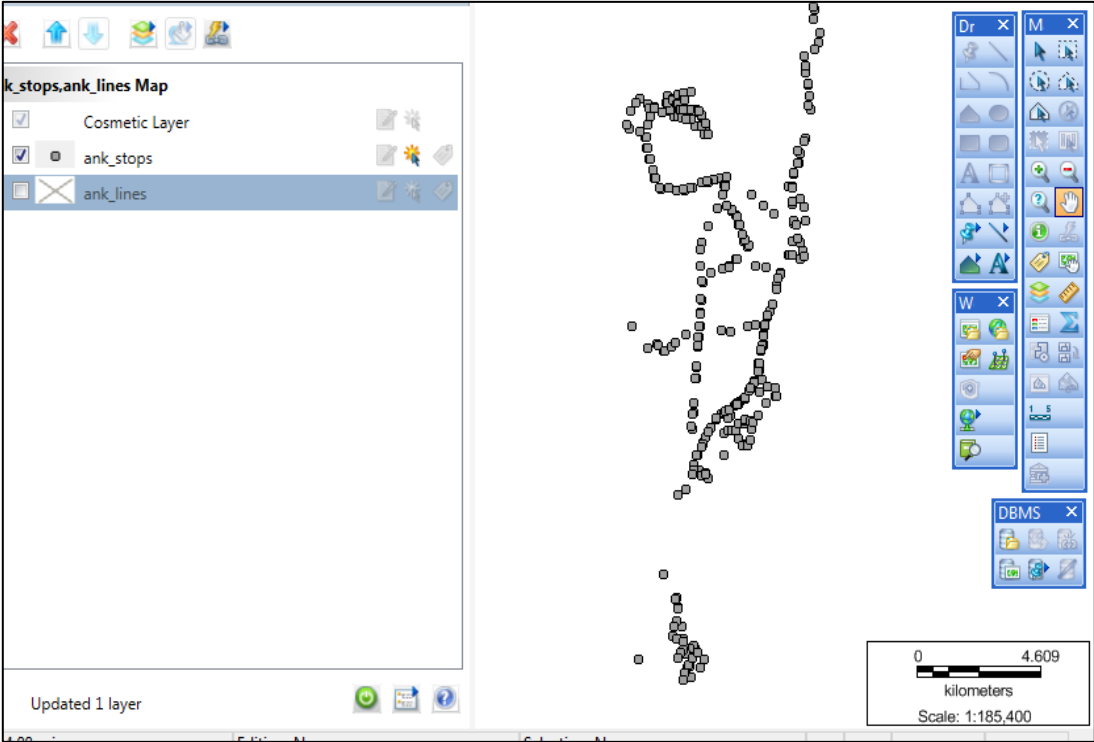


Figure 12 Ankara bus stops

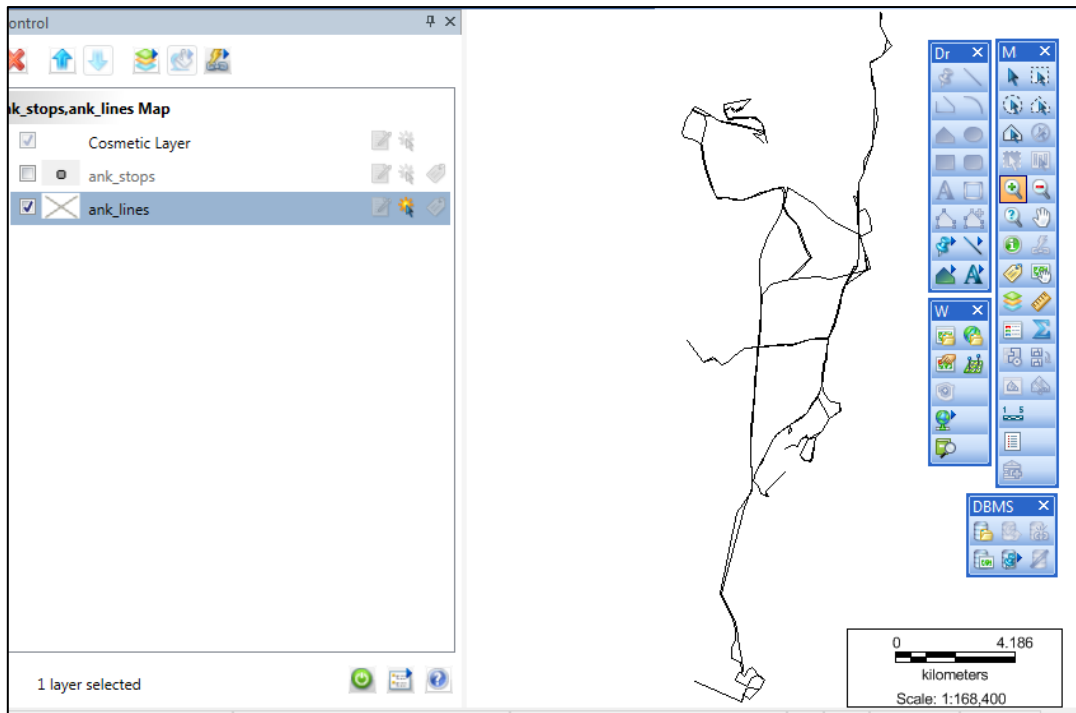


Figure 13 Connected among bus stops (bus line)

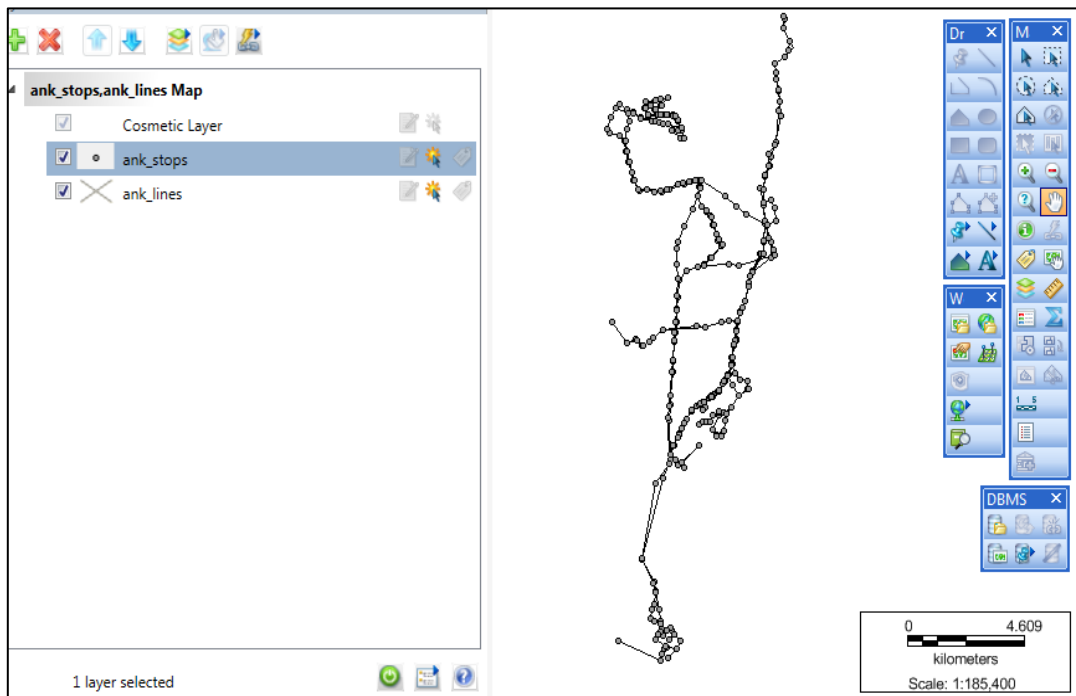


Figure 14 ank_stops and ank_lines layers

1.8 Research Methods

In this thesis, the researcher has spent extra care to avoid plagiarism from the literature. However, there may be some unwanted similarities just by coincidence; to minimize such cases, similarity check online software, iThenticate has been used before the final submission of the text. The organization of the thesis is as follows:

Firstly, the researcher obtained the sample datasets, then studied these dataset to understand how to use them in this thesis.

Secondly, we studied the graph theory especially the shortest path problem and algorithms to solve this problem, and decided to choose Dijkstra algorithm.

Data analyze is an important part in this thesis, and that is the third stage. The researcher by using MapBasic analyzes datasets for implementation.

After analyzing the data, the researcher used C++ application to find the paths among origin and destination points, and sent this result as a text file.

Finally, by using MapInfo the researcher can represent the result in a map to find the difference between shortest and longest paths.

CHAPTER II

BACKGROUND AND LITERATURE REVIEW

As was explained in chapter 1, this dissertation is about the GIS and graph theory, for that reason we will provide a basic concept for them, in addition to the literature review and the problems addressed.

2.1 Graph Theory

Before explaining the graph theory, we will introduce some definitions:

Vertices (V) or Nodes: are a set of points where two or more straight lines meet [11]. We always use it to represent a specific point in the map such as bus stop. Note that vertex is not exactly same as node however, we use these terms interchangeably in this thesis as the difference does not affect our analyses. This distinction is underlined in section 4.2 while discussing limitations.

Edge (E): is a line segment which connects two vertices [11]. In this thesis the edges represented a distance between the bus stops.

Path (P): in a graph, the path is a traversing sequence of vertices connected by edges. A simple path is one in which all vertices are different from one another. A cycle is a path with at least one edge whose first and last vertices are the same. The length of a path or a cycle is its number of edges [11].

Graph theory: is a mathematical theory which concerned with to analyze and measure he networks. it consist of edges and vertices [12].

For example the bus network can be represented as a graph where the bus stations are the graph vertices and links between stations are the graph edges.

Another important characteristic of a graph for the transportation network is that a graph can be weighted. Liu defines the weighted graph as the graph whose numerical label (value) or weight is assigned to each of its edges. In case of transport modeling, this weight is the way of representing the cost of travelling from one point (vertex) to another [12].

2.2 Transportation Networks

Is a set of lines to represent network of roads, streets, or any connections type between nodes, used to provide the flexibility to movement of passengers and vehicular movement or flow of some commodity. it usually use a graph theory to analyze it, for example, in this thesis use the graph theory to analyze the network roads to solve the shortest path problem and find alternative shortest path [11].

2.3 Shortest Path

Shortest path is a fundamental problem in graph theory [13]; then we can define the shortest path problem as the problem of finding a path between two vertices (source and destination).

For example to find the shortest way from capital city to another city, in this case the cities are represented as vertices in the map [14].

Another definition of Liu [12], "*The path(s) through the network from a known starting point to an optional ending point that minimizes distance, or some other measure based on distance, such as travel time*".

This problem solved by using different algorithms [15] depending on the structure of the graph. In this section we classify the shortest path algorithms into:

- Single source shortest path (SSSP) algorithms: in this type of problem, we want to find the shortest path from the specific source node to specific destination node or all nodes. Dijkstra algorithm is more efficient to solve this type of shortest path problem especially when we do not have negative edge weight, however if we have it, the Bellman-Ford's algorithm is more efficient than Dijkstra [12] [13].
- All pair's shortest path (APSP) algorithms: if we want to find the shortest path from all nodes to all nodes, then the Floyd-Warshall algorithm is more efficient to solve this type of shortest path problem. Another algorithm to solve this type of problem is called Johnson's algorithm, and this type may be faster than Floyd–Warshall on sparse graphs [12] [13].

Another algorithms are available to solve shortest path algorithms, such as A* search algorithms, Viterbi algorithm, and others. In this thesis we are focusing on Dijkstra algorithm.

2.4 Dijkstra Algorithm

Dijkstra algorithm was designed in 1956 by Dutch computer scientist Edsger Dijkstra [16], and it is one of the most popular algorithms to find the shortest paths to our knowledge.

When the nonnegative edge path is considered, then we can use this algorithm to solve the single-source shortest path problem for a graph. This algorithm is often used in routing and as a subroutine in other graph algorithms. By using this algorithm, we can find the shortest path between the specific vertex to all vertices in the graph [17] [18].

For example, in Iraq we can represent all cities as vertices, and the edge path costs represent driving distances between all pair cities connected by a direct road. The Dijkstra's algorithm can be used to find the shortest route between the capital city, Baghdad, and all other cities.

Methodology: We can define the algorithm methodology as follow [13] [17]:

1. Choose the source node as a first permanent node, and assign it 0 costs.
2. Check all neighbor nodes from the previous permanent node.
3. Calculate the cumulative cost of each neighbor nodes and make them temporary.
4. Check the temporary nodes as follows:
 - a. Choose a node with the smallest cumulative cost, and make it as a permanent node. Keep in mind; do not check the permanent again because this is a final cost for this node.
 - b. If we have more path to reach to the nodes, then the shortest cumulative cost paths was chosen.
5. Repeat steps 2 to 4 to make all nodes as a permanent.

Example: In Figure 15 below, we have 8 nodes connecting together through a weights path, and we want to calculate the shortest path between node A to all other nodes.

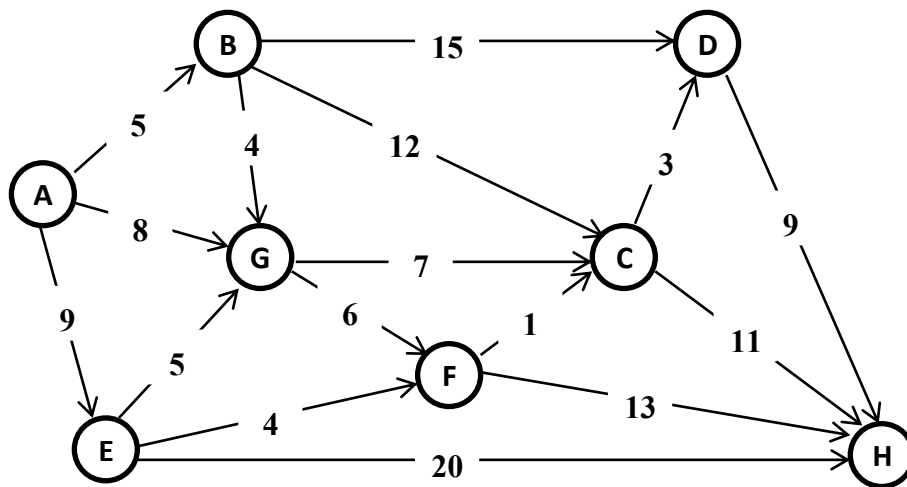


Figure 15 Dijkstra algorithm example

Notes:

- From node to itself we refer (-----), because in road networks we do not have a weight from point to itself.
- Register the weight between nodes according to others.
- Use (∞) sign if we do not have a direct path.
- Use black block to refer this node as visited.

We can use Table 1 to implement the algorithm: so the visited node column refers to which nodes was visited, current node column refer which node we can use now, and other column refer to the nodes:

Table 1 Implementation Dijkstra algorithm

Visited node	Current node	A	B	C	D	E	F	G	H
A	A	-----	5,A	∞	∞	9,A	∞	8,A	∞
A,B	B		-----	17,B	20,B	9,A	∞	8,A	∞
A,B,G	G			15,G	20,B	9,A	14,G	-----	∞
A,B,G,E	E			15,G	20,B	-----	13,E		∞
A,B,G,E,F	F			14,F	20,B		-----		26,F
A,B,G,E,F,C	C			-----	17,C				25,C
A,B,G,E,F,C,D	D				-----				25,C
A,B,G,E,F,C,D,H	H								-----

Then by using the Table 1, we can find the shortest path by choosing the least value for the node column, and find the path by tracing the shortest of those values and the node related to it, Table 2 represent path and shortest path between A to all nodes.

Table 2 Shortest path and path from node A to all nodes

<p>From node A to node A:</p> <p>Shortest path = 0</p> <p>Path = A</p>	<p>From node A to node B:</p> <p>Shortest path = 5</p> <p>Path = A,B</p>
<p>From node A to node C:</p> <p>Shortest path = 14</p> <p>Path = A,E,F,C</p>	<p>From node A to node D:</p> <p>Shortest path = 17</p> <p>Path = A,E,F,C,D</p>
<p>From node A to node E:</p> <p>Shortest path = 9</p> <p>Path = A,E</p>	<p>From node A to node F:</p> <p>Shortest path = 13</p> <p>Path = A,E,F</p>
<p>From node A to node G:</p> <p>Shortest path = 8</p> <p>Path = A,G</p>	<p>From node A to node H:</p> <p>Shortest path = 25</p> <p>Path = A,E,F,C,H</p>

By using the previous results we can find the routes to cover all nodes as shown in Figure 16.

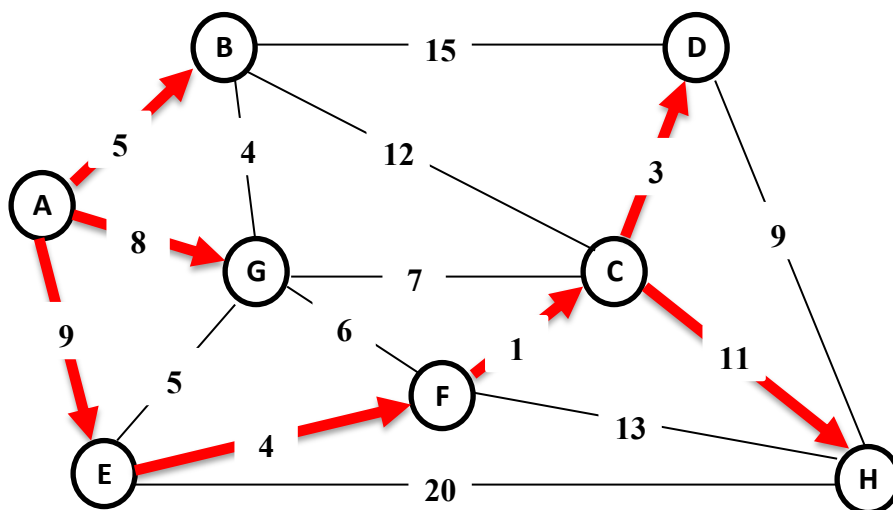


Figure 16 Routes to covered all nodes

2.5 Geographic Information System (GIS)

What is GIS?

Geographic Information System (GIS) is an integrated system of hardware, software and data which can be dealing with geographically referenced information such as capture, manage, analyze and display this information [19] [20].

We can use GIS to understand, analyze and visualize all information in several ways to discover the relationships, patterns, and trends in the maps. In addition, it is used to solve all problems related with geographically information.

Components of GIS

From the above definition we can divide the GIS components in [20]:

- ❖ **Hardware:** are all computer physical devices which connecting as a system to used operating the GIS such as computer servers.
- ❖ **Software:** Are all functions and operations in computer systems which used to store, analyze, and display geographic information. the following represent some key software components:
 - Database management system (DBMS).
 - Input and manipulation tools.
 - Tools that support geographic query, analysis, and visualization.
 - Graphical user interface (GUI) which used to easy to use software.
- ❖ **Data:** is one of the most important components in GIS. This data consist of geographical features and their attribute information.
Digitizing the data is an operation used to enter this data to the GIS system, by digitally encoding geographic features such as building, roads or country boundaries.
- ❖ **People:** people represent all technical specialists working in this system as well as end users who used this technology.
- ❖ **Method:** it is a unique system plans which consider to modeling and operating the GIS system.

GIS data: GIS Systems work with many different types of data [20]:

1. **Spatial data:** All data which related with geographic earth or spatial component called spatial data. We can dealing with these data by using GIS, We can classify the spatial data into:

- **Vector data:** in this type, the data are stored as a series of X, Y coordinate pairs inside the computer's memory. Vector data provide a way to represent real world features in GIS application; in the following Figure we can see houses, trees, rivers, and roads. All of them are features and we can represent it in GIS application. Vector features have attributes which describe them, for example we describe the weight of the roads. Vector data have geometry, which defines the position and the shape of this feature. In vector data we have three types of geometry called points, polylines, and polygons, (Figure 17).

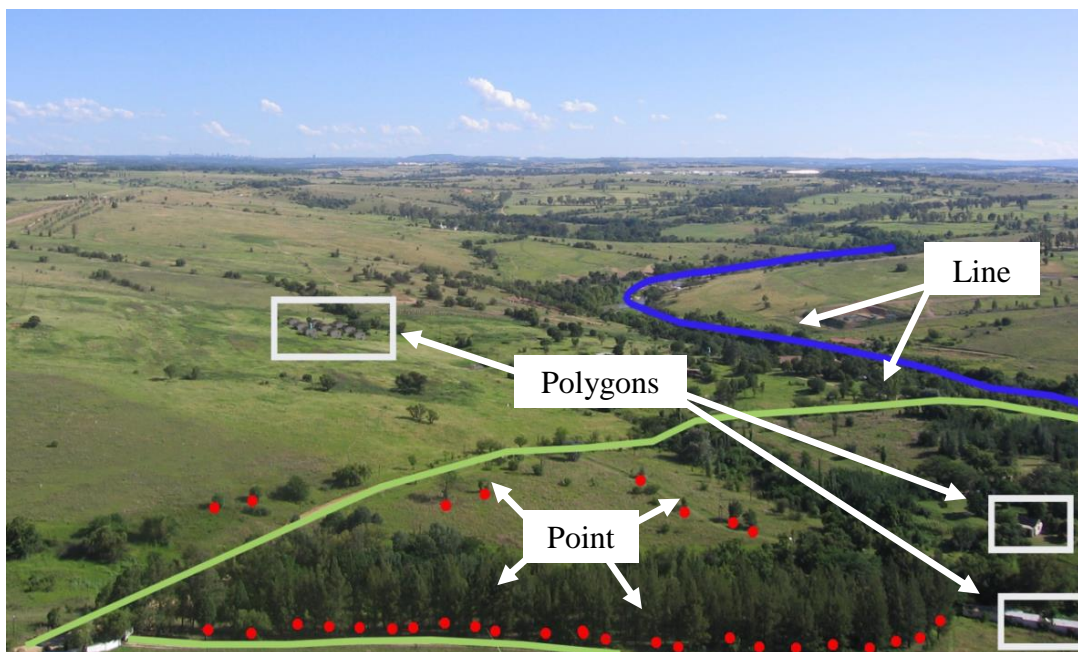


Figure 17 Geometry types

- Raster data represented as a grid matrix , i.e. consist of series of rows and columns, and we can store the information in matrix cells, such as road number.

2. **Non-spatial data:** are all data which not have any related to the location, these data can recognize as a simple form, and also called attribute data. However, we can use linked these data to the GIS features by using a unique identifier. Such as, attributes of bus stops include its station no., line no., line name, etc. (Figure 18).

StationNo	HatNo	Durakno	DurakAdres	DurakAdi
88	117-1	1	Keklik P_nar_ Mh, Dikmen Cd 544, 06450 Ankara, Türkiye	SOKULLU HAR.NOK.1.DURAK
89	117-1	2	Keklik P_nar_ Mh, Dikmen Cd 524, 06800 Ankara, Türkiye	D_KMEN CAD.4.DURAK
90	117-1	3	Akp_nar Mh, Dikmen Cd 498-502, 06800 Ankara, Türkiye	D_KMEN CAD.5.DURAK
91	117-1	4	Karap_nar Mh, Dikmen Cd, 06800 Ankara, Türkiye	D_KMEN CAD.6.DURAK
92	117-1	5	Karap_nar Mh, Dikmen Cd 420-426, 06450 Ankara, Türkiye	D_KMEN CAD.7.DURAK
93	117-1	6	Huzur Mh, Dikmen Cd 402, 06450 Ankara, Türkiye	D_KMEN CAD.8.DURAK
94	117-1	7	Keklik P_nar_ Mh, Dikmen Cd 403-405, 06520 Ankara, Türkiye	D_KMEN CADDES_ 7.DURAK
95	117-1	8	Huzur Mh, Dikmen Cd 386, 06520 Ankara, Türkiye	D_KMEN CAD.9.DURAK
96	117-1	9	Malazgirt Mh, Dikmen Cd 370, 06520 Ankara, Türkiye	D_KMEN CAD.10.DURAK
97	117-1	10	_ht. Cengiz Karaca Mh, Dikmen Cd, 06450 Ankara, Türkiye	D_KMEN CADDES_ 10.DURAK
98	117-1	11	Osman Temiz Mh, Dikmen Cd 336, 06450 Ankara, Türkiye	D_KMEN CAD.11.DURAK
99	117-1	12	Ayd_nlar Mh, Dikmen Cd 310-320, 06450 Ankara, Türkiye	D_KMEN CAD.12.DURAK
100	117-1	13	Ayd_nlar Mh, Dikmen Cd, 06450 Ankara, Türkiye	D_KMEN CAD.13.DURAK
101	117-1	14	Ayd_nlar Mh, Dikmen Cd, 06450 Ankara, Türkiye	D_KMEN CADDES_ 14.DURAK
102	117-1	15	_lkad_m Mh, Dikmen Cd 233-237, 06450 Ankara, Türkiye	D_KMEN CAD.14.DURAK
103	117-1	16	lkad_m Mh, Dikmen Cd 193-203, 06450 Ankara, Türkiye	D_KMEN CAD.15.DURAK

Figure 18 Non-spatial data for bus sops

2.6 Literature Review

In [21], the authors present their work in China. They report on the traffic consultation system of Guangzhou designed by an algorithm to find N shortest paths successfully and they analyze the complexity of the system.

In this paper the authors discuss the issue of consultation system. The users always need a good choice for decision-making (shortest path is one of the best choices in public transport). In addition, these users need an alternative choice to get more flexibility in decision-making (more paths to the destination). The authors found more than one path to reach to destination point which is called as "N shortest path problem". The researcher see the authors thinking to find more than one shortest path to support the consultation system, and based on this idea, we aims to find more than one bus stop locations to solve problem 3 which will be explain in section 2.7.

Fuhao and Jiping [22] present a new algorithm based on the topological relation of the vector data to meet the network analysis for huge data in practice. Firstly, the authors present the principle of Dijkstra algorithm, then propose an improved shortest path algorithm. Finally, they present the disadvantages of using Dijkstra algorithm. They present the problem that comes with huge data: this huge datasets of numerous are used to present the adjacent and distance matrices. It means that if we have a lot of nodes, then we will need a lot of memory capacity in the implementation.

This algorithm helps the authors to use the connections between the arcs to save memory and avoid the correlation matrix. in addition, by applying it to the network with huge nodes, helps them to support the network analysis of numerous maps after the pre-processing phase.

Conventional forecasting transit demand requires detailed data such as dwelling unit and census blocks [23]. Huang et al. studied transit demand forecasting using GIS-based accessibility modeling approach which is based on a distance decay concept. The process of accessibility method implementation is given in details in Figure 19.

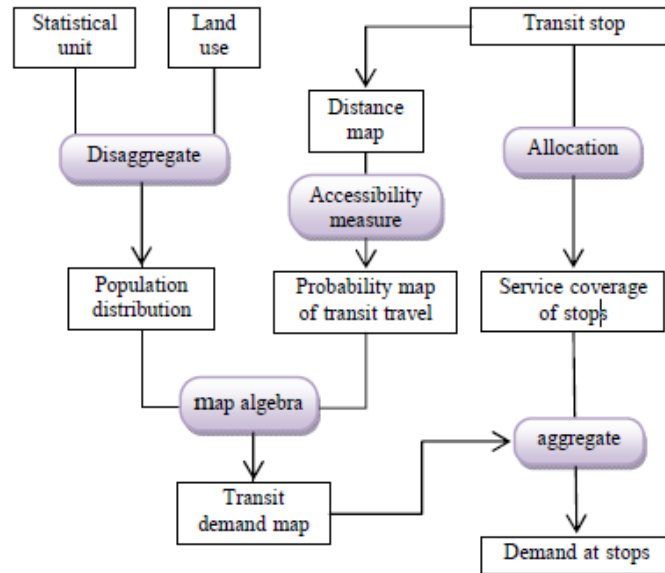


Figure 19 An example on accessibility method implementation steps [23]

This GIS process is tested on Wuhan in 2009, a city in China with over 4.5 million residents. The bus trip routes were drawn based on two models: negative exponential model and negative logistic model. The study showed that both models underestimated the bus transit share by comparing them with transit survey for Wuhan in 1993. In this paper, it was concluded that using one accessibility model to model the transit production is not suitable for large cities due to the population complexity.

Ma et al., [24], conducted a study to create an areal model for the city of Nanjing/China. In their study, MapInfo was used to create the topological road network of the city. The shortest paths were computed using Floyd algorithm. Accessibility factor planar distribution model is created for the entire region. This model was crucial to conduct a regional accessibility analysis that has an impact on the transportation systems and urban planning. The authors claim that if the accessibility factor planar distribution model of the whole region is established, then it would be suitable to use for any area to choose. Finally, they can make a further expand in their model by: taking into account the complexity of the city's transportation system, and using other indicators of the accessibility such as time index.

In an earlier work [25], Benjamin claim that shortest path can often be used as a benchmark or a starting point for solving more complicated problems in transportation analyses. About 10 years ago he submitted a useful paper for researchers and practitioners in transportation, GIS, operations research and management science. This paper identified three shortest path algorithms on a real road network. The transportation analysis concerning large geographic regions has done in real time, then it need a high performance shortest path algorithms that run faster in real time network.

Hala and Ossman use the GIS tools to develop a least-cost path to link three cities in Sinai peninsula in Egypt [26]. They model three visions: an engineering vision, environment vision and hybrid vision, and finally use the definite software multicriteria evaluation to compare these visions and recommend using the hybrid vision model to develop a least-cost path. They concluded in their study to use the GIS and shortest path algorithm in the early planning system. It proves to be well suited, economic and time-saving for a sustainable corridor location design. The advantage of their study is to avoid many location problems such as sand dunes, faults, high slopes and zones of ecological or cultural values, in addition, the ability of the technique to provide several alternatives and to compare such alternatives is a benefit for designers.

In [27], Alazab et al. studied the traffic system in Zarqa city in Jordan; they proposed an optimal transportation routing algorithm to cater the dynamic changes in the traffic flow, and solve a routing problem in a stochastic transportation network. The authors use a systematic approach to use it in their implement the proposed algorithm for an efficient road transportation routing system integrated within a GIS to provide real-time traffic flow information. In their study, they considered a stochastic shortest path problem on a road network, which is composed of links with nonstationary travel times where subsets of these links are observed for traffic flow in real-time with the aid of GIS. An assumption was made that each observed link can be in either in a congested or uncongested state based on the travel time distribution used in their algorithm. *“The authors concluded that real-time traffic information from GIS incorporated within an interactive Web application can significantly reduce expected*

total costs, vehicle usage and driver productivity during times of heavy congestion. In addition the use of good implementation attributes such as network representation, node labeling method, selection rules and data structures have also facilitated in developing a real-time performance algorithm within a web-based application” [27].

Al-Mumaiz [28] did a thorough study on the city of Al-Ramadi in Iraq to find the shortest time between predefined zones through a GIS network analysis. The work was done by creating 5 zones in the city of Al-Ramadi, then trying to find the shortest distances between the centers of these zones. An Origin-destination matrix was presented in this study which includes the trips within the city.

In [29], Sahar and Moosavi were to modify the cross-entropy (CE) algorithm to find high quality solutions and fairly quickly for shortest path problem the shortest path in static network.

In [30], Al-Joboory et al. employed GIS to select optimum road path between two cities depending on the facility of ArcView software with its extensions of spatial analysis and 3D analysis to solve the problem. The authors reported that GIS is used to give the geometric design that contains types of curves and the geographic coordinates necessary to set out the optimum road, as well as to select the optimum path.

Kadry, Abdallah, and Joumaa [31] modified Dijkstra algorithm to reduce the iterations used in the traditional algorithm. The main idea in this research was to deal with the second stage of the traditional version of this algorithm where the shortest routes from a node to its successors are many and to reduce the number of iterations. The results showed that iteration can be reduced by 37% of the traditional way.

Critical path method (CPM) is a project management technique containing a sequence of scheduled activities that analyze the projects and in addition determines the duration of a project plan or an iteration plan [32]. Shankar and Sireesha presented a new approach to determine a CPM network by using Dijkstra algorithms [33]. They showed the applicability to use Dijkstra to find a CPM, and easily applied to more complicated application in network analysis. They modified this algorithm by using a forward pass algorithm to calculate the earliest time (E_i 's), a backward pass

algorithm to calculate the latest time (L_j 's), the slack time (T_{ij} 's) by using the following formula.

$$T_{ij} = L_j - E_i - t_{ij} \text{ for each activity using the activity times } t_{ij}.$$

The researcher found the effectiveness the Dijkstra algorithm for several applications in network analysis, and this is the main reason to use that algorithm. This is an example to show the variety of Dijkstra algorithms.

In Beijing, the public transportation system faced multiple choice routes problem, because the bus lines become smooth and convenient, and reached more than 800 lines. Besides there are many complex algorithms used to solve a shortest path problem. On these bases, Yong and Hongping [34] used the relationship of shortest path algorithms, direct matrix, and transfer matrix to develop a mathematical model, for improvement. The new method reduces the complexity of the algorithm, then stores the sparse matrix used in the calculation in the nonzero elements in order to save storage space, reduce cycle times, and improve set operation thoughts in the line of the query operation. These speeds are up the operations and check out the path in less than 20 seconds. As can be seen, this paper main focus is to reduce the implementation time, and the memory space needed. Based on these, we aimed to compare all these features in terms of applicability in Iraq, as we explain more in section 4.3.

Tracking mass transport vehicles via mobile devices is not a new technology. Examples include Bus-Net [35], a mobile application open to public use in Japan. However, the system cannot bear the increasing expectations of the public, especially children of school age and senior citizens. For example, the system cannot update arrival times to the bus stops when there is a change in the service, such as a delay. To avoid such misinformation, a new update is proposed to be patched to Bus-Net; the vehicles are now equipped with GPS devices and send real-time spatial information to Bus-Net. Furthermore, the mathematical model has been updated. It demanded the smart phones to develop this system by using the GPS function and an algorithm to estimate the delay of the bus based on location information. This is instead of setting up telecommunication equipment on bus stops, which needs a

higher investment. The new system describes the bus location system, estimates the bus delay, and effectiveness of the bus location system.

From this paper we can observe the effectiveness of information technology and the modern techniques in developing the transportation system and assist in reducing the time and cost.

The travelling needs and human habits during travel, in addition to the features obtained from the public transport networks which differ from other non-public networks are important factors to success and promote public transport systems. The obstacles faced by public transport systems such as traffic congestion and the cost of unexpected transportation are the most important impediments to networks transportation. Yuewen and Zhong [36] combined factors of Shanghai residents on travel habits, and verified and analyzed the constraints of passengers travel decision-making factors. Through comparative analysis, based on the ISM (interpretive structural modeling) analysis method [37], they could effectively analyze the psychology and the principle of optimal choice of traveling passengers. They use the time cost for waiting and transportation by bus and walking, fee paid for the transportation, and the distention for all points as factors. Then, they represented the basic binary relations or the direct connection between these factors in a matrix called "adjacency matrix". They employed this matrix to create a new matrix called "reachability" to show the transitive binary relations between the systematic elements or any reachable path between two nodes. After that, they classified the reachability matrix elements into reachability set (collection of all the reachable elements), common set (common parts of both reachability set), and advanced set (intersection between reachability and common sets), and divided the structure obtained it into several layers of factors.

In [36], we observed that the travel cost was one of the important factors to verify and analyze the constraints of passengers travel decision-making factors. In addition, the complexities of public transport network in many cities (such as Qingdao, Wuhan, Hangzhou, etc.) have homonymous stops at different places, and some of them are far away from each other. In those years, Li et al. [38] presented a paper to find the Least Transfer Cost Model (LTCM) for optimizing public transport travel routes. They

classified the homonymous stops into up-down stops, neighbor stops, and different stops with same name. In addition, they classified the bus stops into transfer at the same stop (TSS - has the least cost as travelers do not need to walk when they change buses), transfer between a pair of up-down stop (TUS - has a little more cost as travelers and should have a short walk), and transfer between a pair of neighbor stops (TNS) based on the place where the transfer happens, public transport transfers between homonymous stops (Figure 20). “They proposed to measure the convenience of different types of transfers. LTCM is on the basis of least-transfer optimal travel route model, and transfer cost is supposed to be another standard to find out which route is better. When two or more routes have the same transfer time, choose the least transfer cost route to be the optimal travel route”. Compared with other least-transfer optimal travel route models, LTCM can not only choose the optimal travel route without GPS information however also can identify homonymous stops. Thus LTCM can be used in public transport network based on both geography and path-stop relationship” [38].

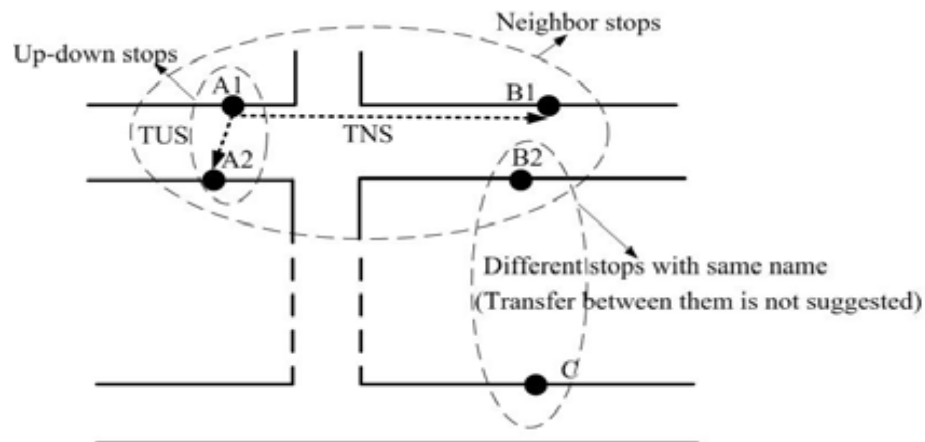


Figure 20 Homonymous stops and different types of public transport

In [39] Martinez et al. proposed and tested a new formulation to determine the time interval between subsequent busses for small and large scale transportation systems. The new method is called Mixed Integer Linear Programming (MILP). The method was tested on 13 line and 130 line transportation systems (real test case on Montevideo-Uruguay). The researchers have also proposed a metaheuristic approach,

which is a high level procedure to provide solution for optimization problems; this approach is based on Tabu Search for solving frequency optimization problems for large systems, with more than 100 lines.

Martinez et al showed that even for small size transportation systems, improvement is possible by reporting a 3% system improvement using their model for a case with 13 lines.

Shortest path algorithms used to find paths between physical locations, in addition have several applications such as crack detection, and road or linear feature extraction in images. There are applications where the starting and ending positions of the shortest path need to be constrained; then Sun and Pallottino [40] presented new modern algorithm for the extraction of a circular shortest path (CSP) in an image such that the starting and ending positions coincide. These algorithms are:

1. Multiple search algorithms.
2. Image patching algorithm.
3. Multiple backtracking algorithms (MBTA).
4. The combination of image patching and multiple back-tracking algorithms.
5. Approximate algorithm.

After that, the authors compared these algorithms, and concluded that the image patching algorithm is faster although a solution is not guaranteed. The MBTA is also fast and guaranteed to a circular path, however may not be the optimal one. The combination of image patching algorithm and the MBTA achieves a much higher probability and speed in finding the optimum CSP.

Hadas et al. [41] studied the public transit system with respect to attributes related to the connectivity and coordination. They developed a model to calculate transit connectivity measures based on time and transfer quality. They used three sources of information in their model:

- a) Google transit.
- b) Local-transport agency.
- c) GIS-based road network.

Their analysis was based on three main attributes; the type of transfers made, waiting time and walking time. The researchers applied their model in Auckland city and

North Shore city in New Zealand and compared the outputs from both cities. Using their model, they concluded that the Auckland's trips are shorter however the quality of the North Shore city transfer is better.

2.7 Problems

The problems covered in this thesis are as follows:

1. Calculating the shortest path between origin and destination.

In section 3.2 step 3, the researcher used an application of his development in C++ to calculate the shortest path between the origin and destination points based on Dijkstra algorithm.

2. Detecting the source and destination points.

For the path to be drawn, the user should determine the starting point (origin) and final point (destination) for routing. If the origin and destination points are bus stops, then we can directly use the bus stop information, as shown on Figure 21.

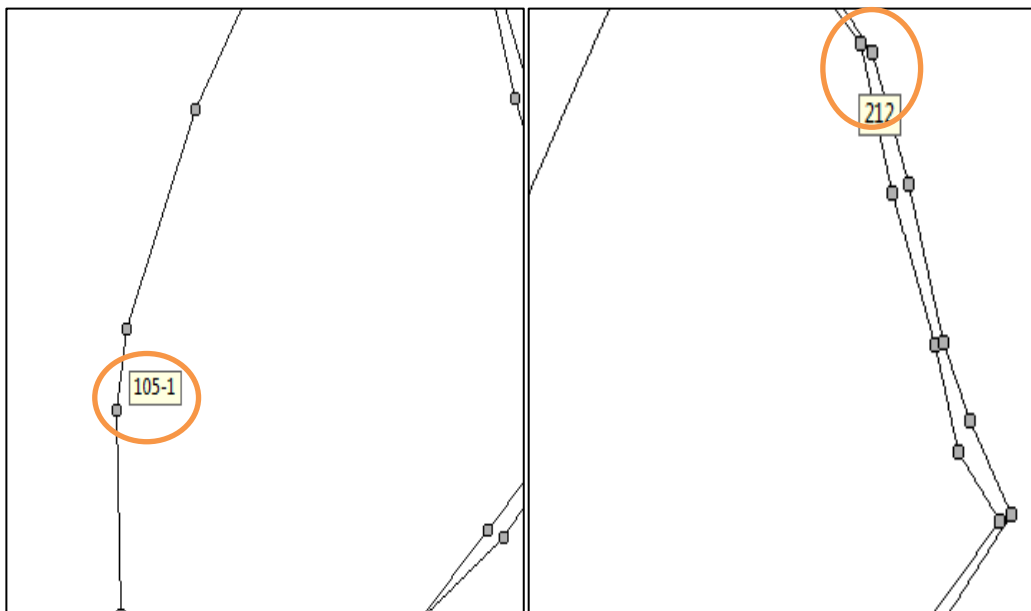


Figure 21 Source and destination points as bus stops

However, in general the origin and destination can be any point on the map; in such a case, the researcher should identify nearest bus stop to origin and destination points (Figure 22). Then the researcher used the MapBasic to write a code to detect the origin and destination points, section 3.2 steps 1 and 2.

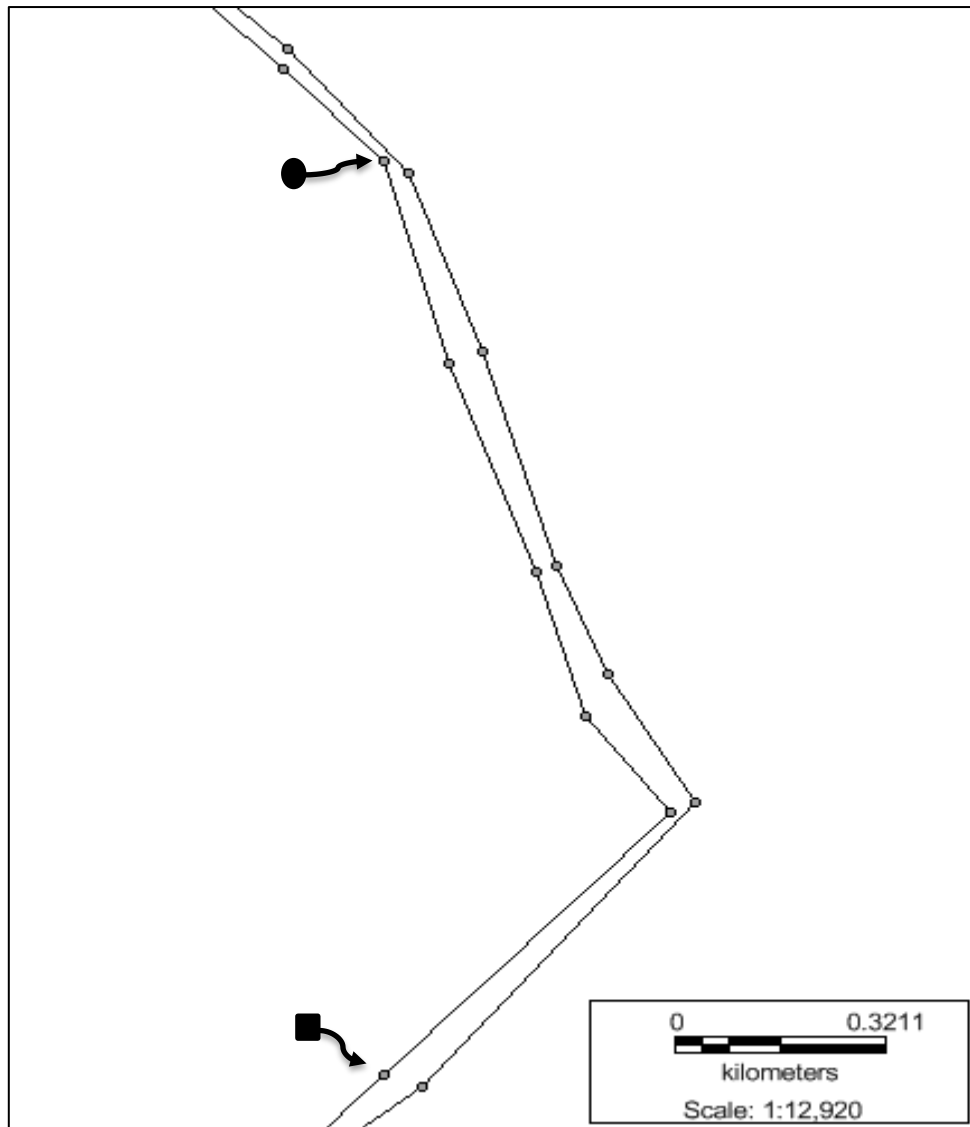


Figure 22 Destination and source are free points

3. There is no direct path from origin and destination points

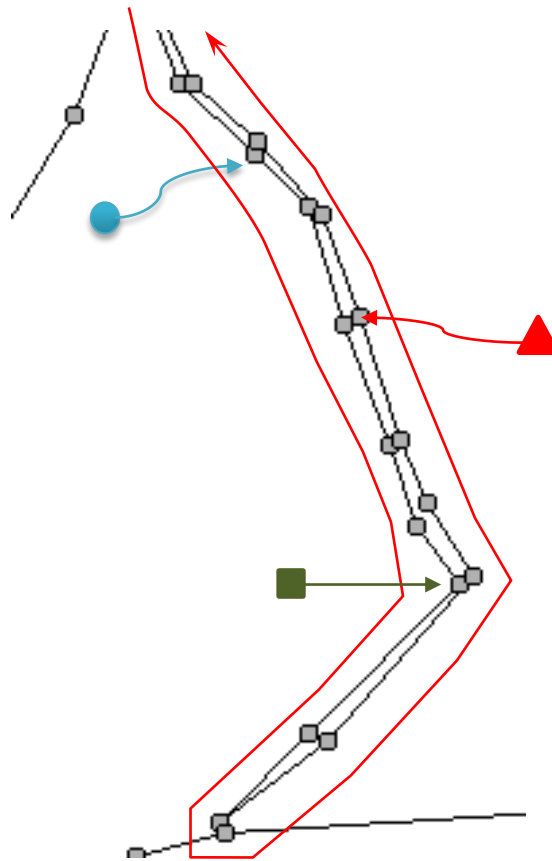


Figure 23 Destination, source points and the path

In Figure 23, the origin is shown as a square point and the destination as a circle point while the arrow shows the path. Here, the origin point does not have a direct connection to the destination point.

Another case is shown on Figure 23, where the origin is a square point and the destination is a triangle point, and arrow line is a path. In this case the distance between origin and destination is quite long.

These cases represent long paths to reach to the destination. The researcher solved these problems in section 3.2 steps 1 and 2.

4. Connections among bus stops.

An important issue in transport system is to make connections among the bus stops. Such connections are based on the type of transportation by bus or on foot as seen on Figure 24.

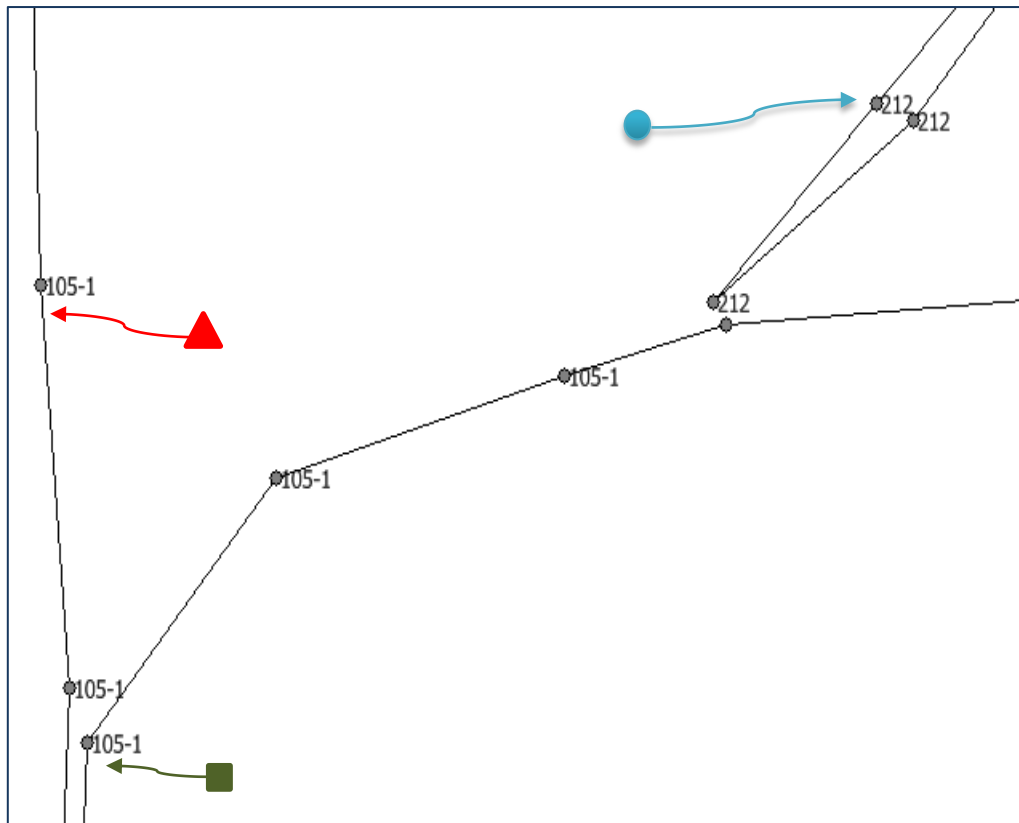


Figure 24 Connections among bus stops

In Figure 24, if we want to go from circle point to square point, we need to use line no 212 to the central lines stops, then take another line called 105-1. This caused the largest path and in addition the longest time. The same case appears when we want to go from square point to triangle point.

The researcher solved this problem in section 3.1.

5. Dealing with huge data

The transportation systems have huge datasets to represent bus stops, addresses, destinations, transmit type, etc.; hence manipulation such as calculating shortest path requires handling huge data; the researcher has faced this problem while reading and processing in C++. The problem is solved as shown in section 3.2 step 3.

6. Extract the result

In section 3.2 step 4, the researcher use MapBasic to extract the shortest path between origin and destinations.

CHAPTER III

SAMPLE APPLICATION OF MASS TRANSPORT SYSTEM IN ANKARA

This chapter addresses the problems presented in chapter 2.

3.1 Data Analyses

In this part, we present analyses of the data obtained from Bařarsoft company. This analysis helps to solve problem 4 given in section 2.7. The idea to analyze these datasets is to investigate potential relationships among all the bus stops based on their proximities by the type of transportation bus or walk (Figure 25).

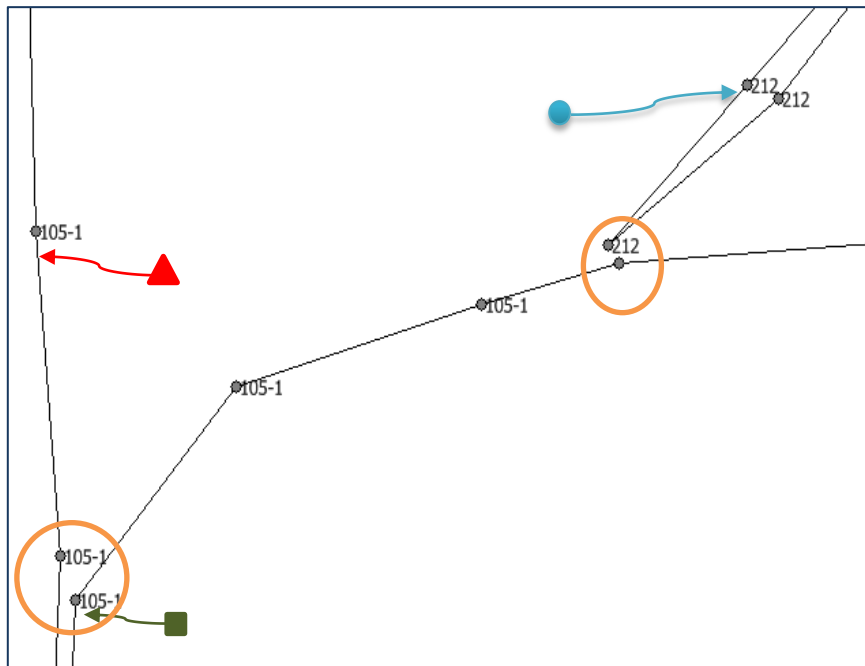


Figure 25 Bus stops relationship by walking

When we want to go from the circle point to the square point we have more than one option:

First option: to use only the bus route. In this cases, if we do not have a direct path from origin to destination, or the origin and destination are not on the same line then we approximately we have a long path to reach to the destination

Second option: we can create walk relations between bus stops with a specific length.

These relations help us to convert the bus line, and helps in the above example to convert the line 212 to line 105-1.

In the same time we can use these walk relationships if we want to go from the square point to the triangle point

The bus stop relationships by walking can be made in two ways:

1. By MapInfo Nearest Function

From File menu select New Table or Ctrl+N from keyboard. After that detect Add to Current Mapper from New Table box, and press Create, Figure 26.

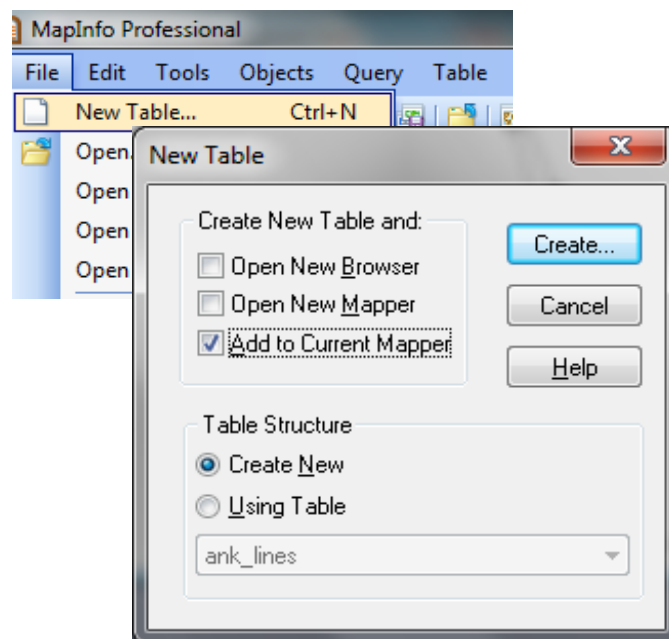


Figure 26 Create new table and add to the current mapper

After that, the Modify table Structure text box is shown. In this text box, we can add any field we want form the table. In addition we can give it a name and classify the type of field, Figure 27.

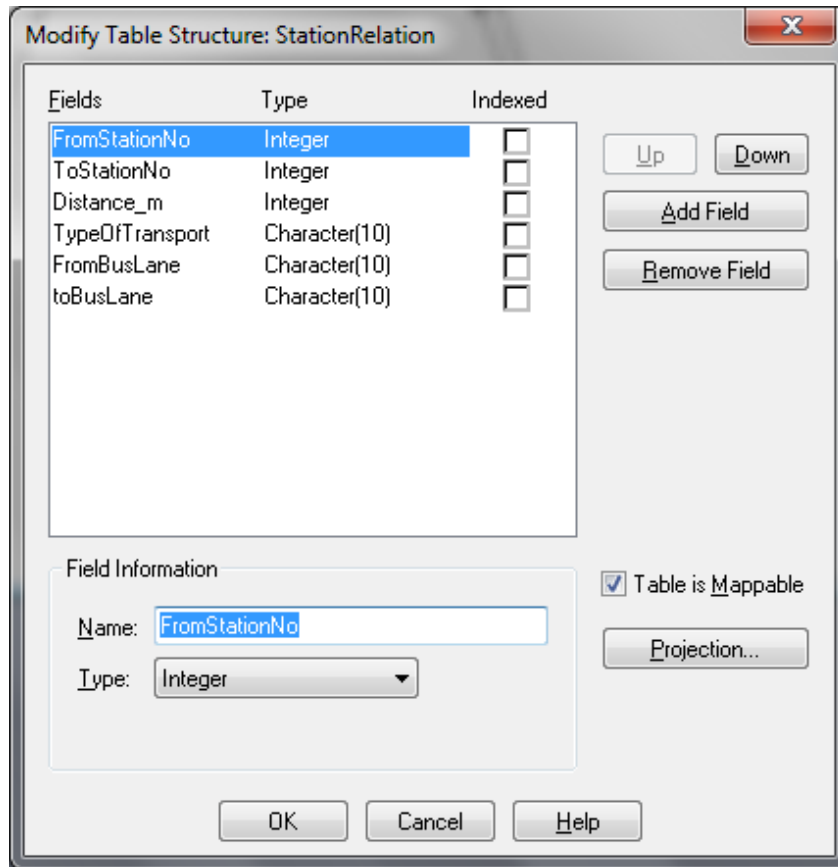


Figure 27 Modify Table Structure text box

After we create the table and fields, we use the Nearest statement function to create the relationship [42].

After that, we open the MapBasic window from Option menu (Figure 28), and write a code to make a query to achieve the relationship by walking. By running this code, we can detect the walking distance by meters.

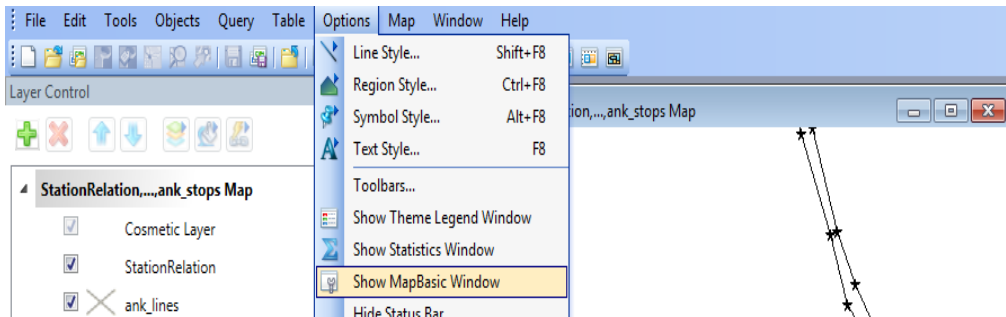


Figure 28 Open MapBasic window from MapInfo

We need a "FromTable" name and "To" table name, so we make a query to have a copy of the Table. So we saved ank_stops as AnkStopsCopy, respectively. Figure 29 shows the implementation query, it take 11 second to finish by using a personal computer with processor Intel(R) Core™ i3-2310 CPU @ 2.10 GHz, 8.00 RAM, and 64-bit Windows 7 Home Premium.

Then we write the code as follows:

```
Nearest 5 From table ank_stops To AnkStopsCopy into StationRelation
ignore max 500 units "m" Data FromStationNo=ank_stops.stationno,
ToStationNo=AnkStopsCopy.Stationno, typeofTransport="Walk",
Frombuslane=ank_stops.hatno,tobuslane=ankstopscopy.hatno
```

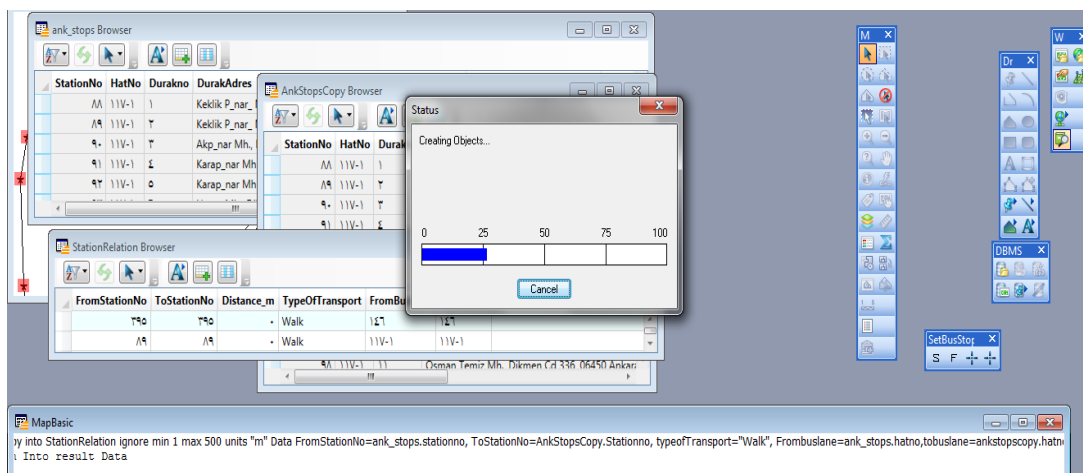


Figure 29 Query implementation to achieve the relationship by walking

A necessary step to update the distances is that we choose Update Column from Table menu, figures 30, 31 and 32 respectively.

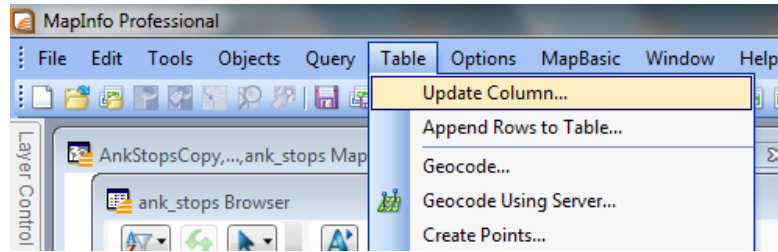


Figure 30 Choose Update Column structure

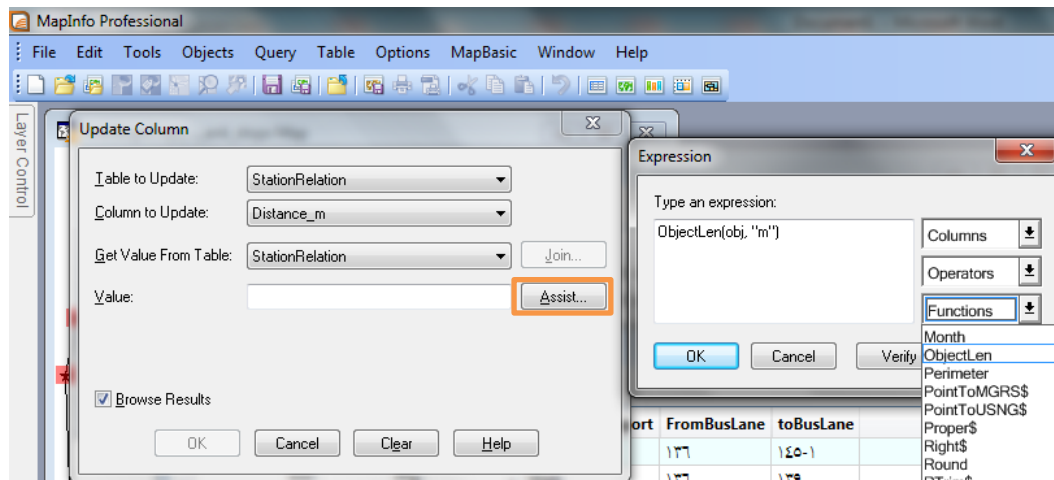


Figure 31 Requirement to update

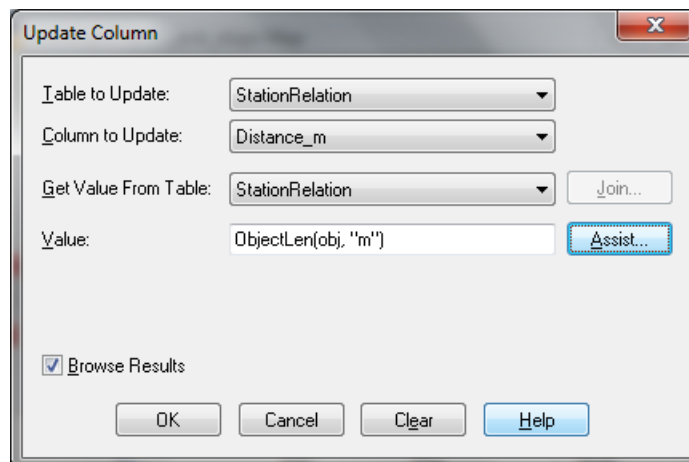


Figure 32 Update Column text box

In Figure 33 we can see the relationship table, and figures 34, 35 shows the relationships as layers.

FromStationNo	ToStationNo	Distance_m	TypeOfTransport	FromBusLane	toBusLan
88	88	0	Walk	117-1	117-1
88	395	2	Walk	117-1	146
88	600	4	Walk	117-1	117-1
88	381	5	Walk	117-1	145-1
88	330	7	Walk	117-1	145-1
89	89	0	Walk	117-1	117-1
89	331	1	Walk	117-1	145-1
89	251	2	Walk	117-1	139
89	396	3	Walk	117-1	146
89	255	3	Walk	117-1	139
90	90	0	Walk	117-1	117-1
90	332	3	Walk	117-1	145-1
90	397	3	Walk	117-1	146

Figure 33 Relationship table

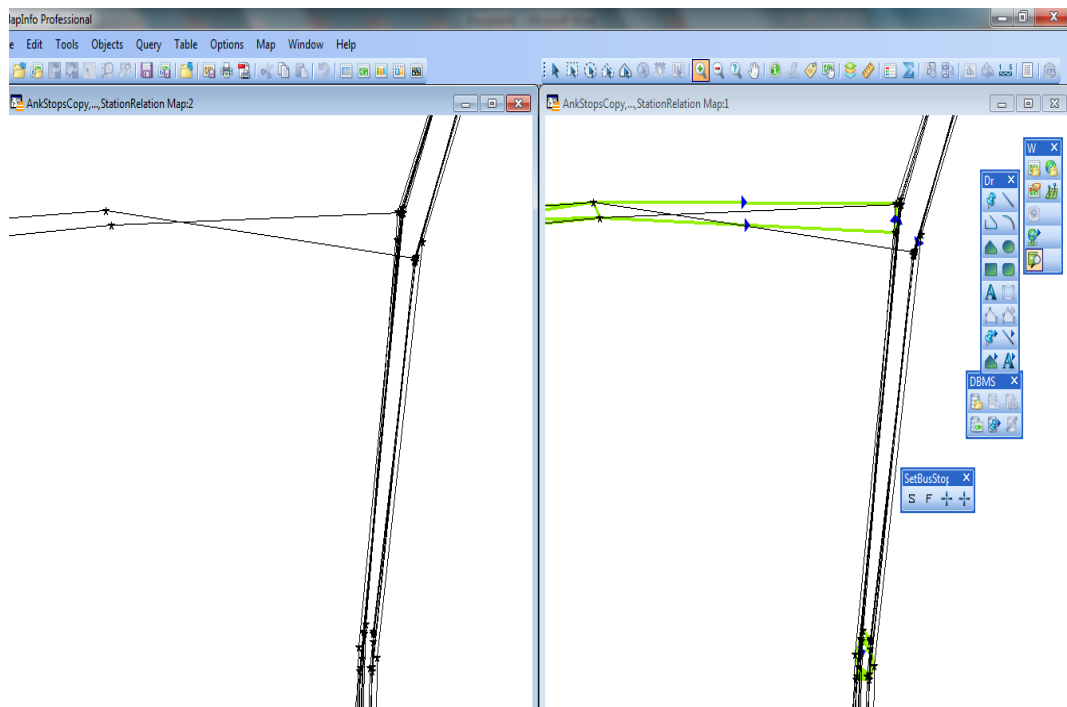


Figure 34 Represent relationship as a layer

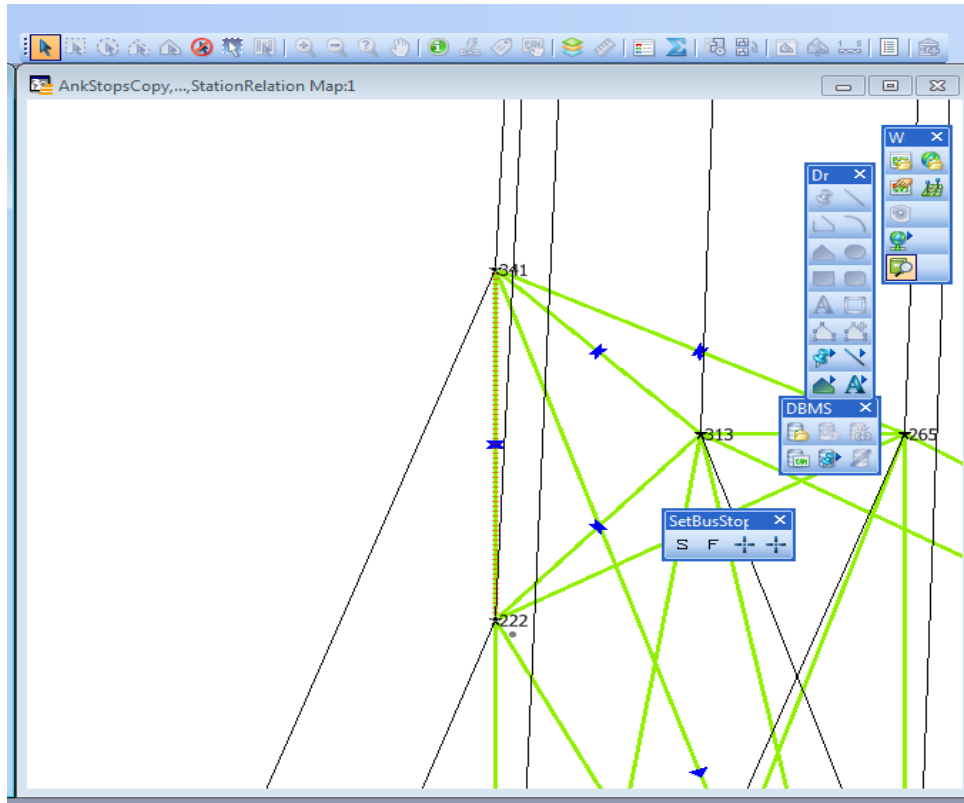


Figure 35 Another relationship as a layer

2. We create circles around BusStops and generate SQL intersection Query.

Choose Buffer from Table menu (Figure 36).

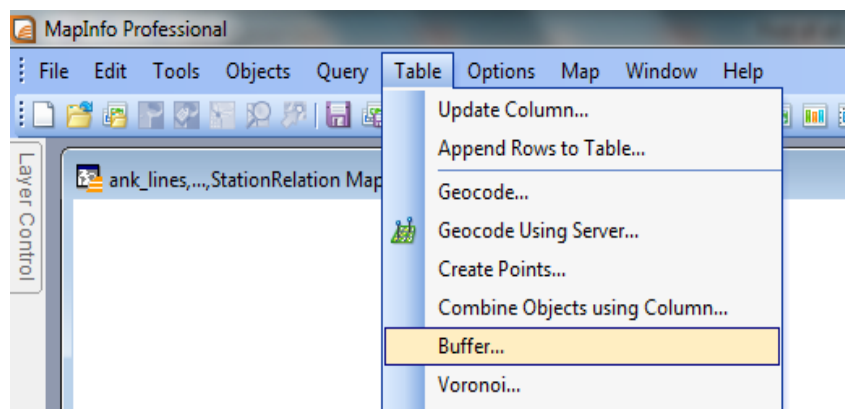


Figure 36 Choose Buffer from Table menu

Select the object and store the result in new table (Figure 37).

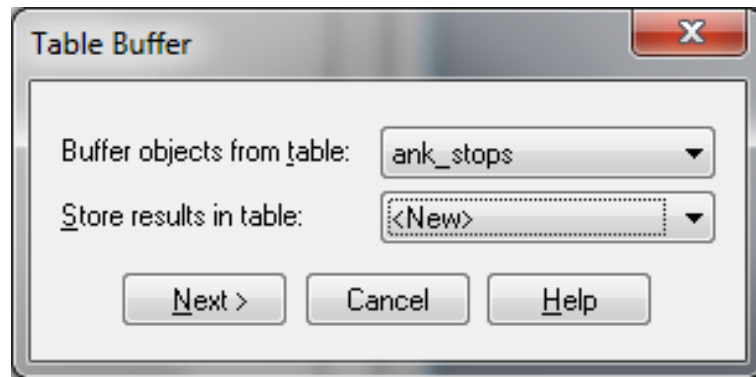


Figure 37 Detect buffer object and store the result

We complete the requirements by creating a new table and saving it as shown on figures 38, 39.

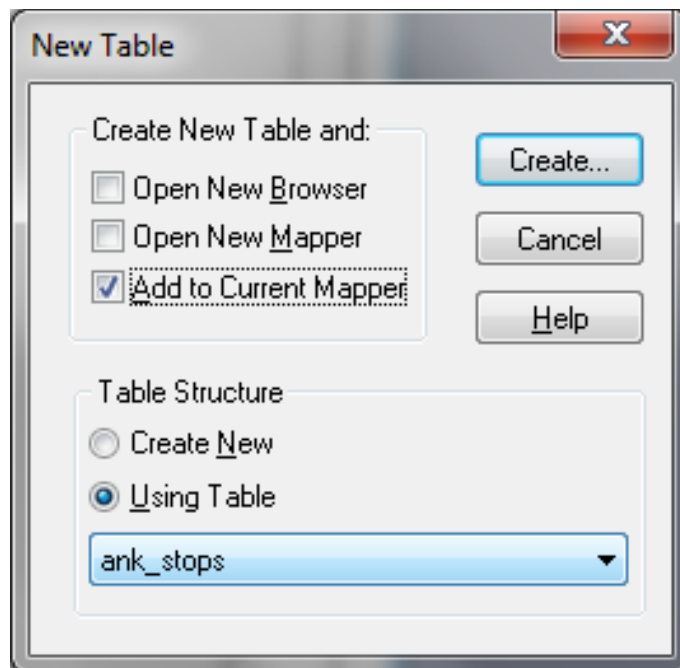


Figure 38 Creating new table

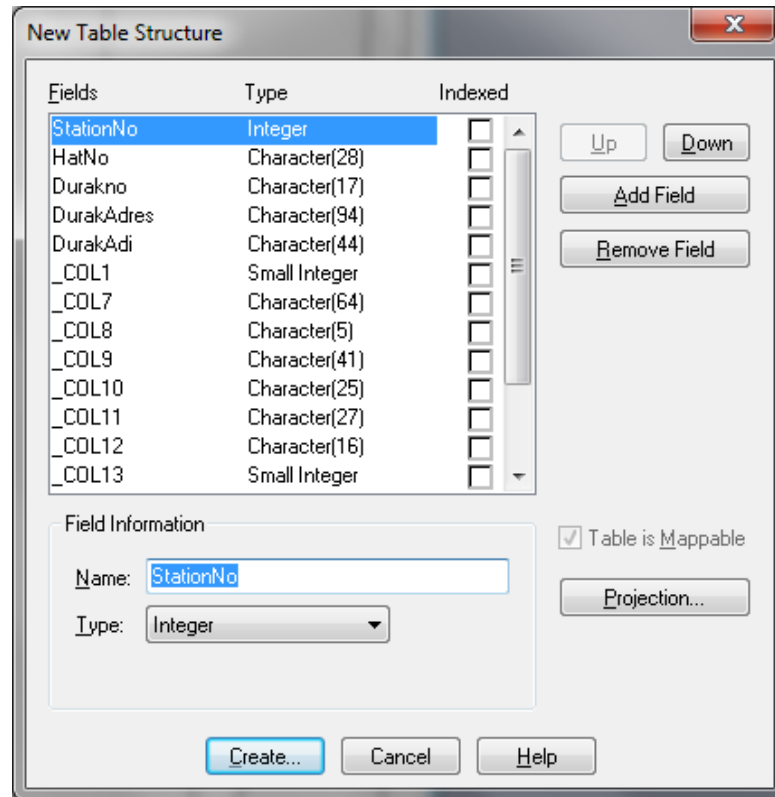


Figure 39 Complete creating the new table

After that, we must detect the buffer criteria to complete this query, such as the value to present the walking distance and the unit, in addition to choose one buffer for each object (Figure 40).

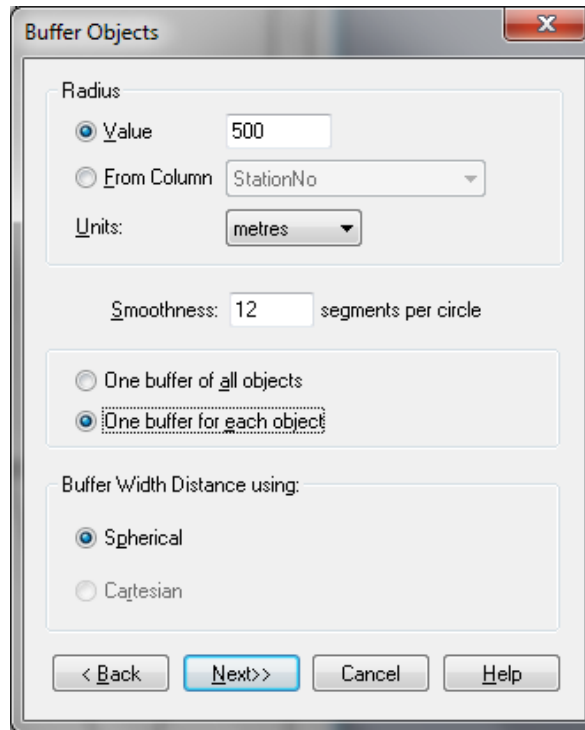


Figure 40 Detect the buffer criteria

Complete the data aggregation as follows (Figure 41).

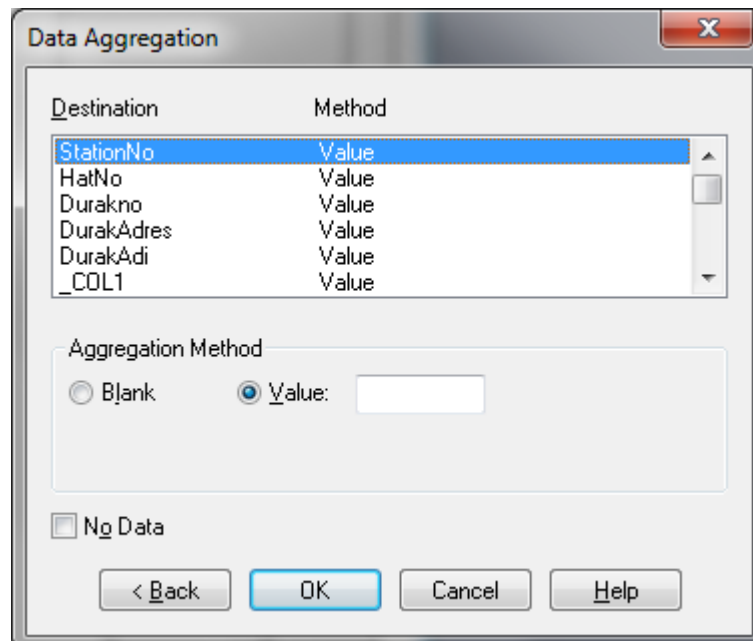


Figure 41 Data aggregation

Finally, we must to complete the SQL required to complete the query (Figure 42).

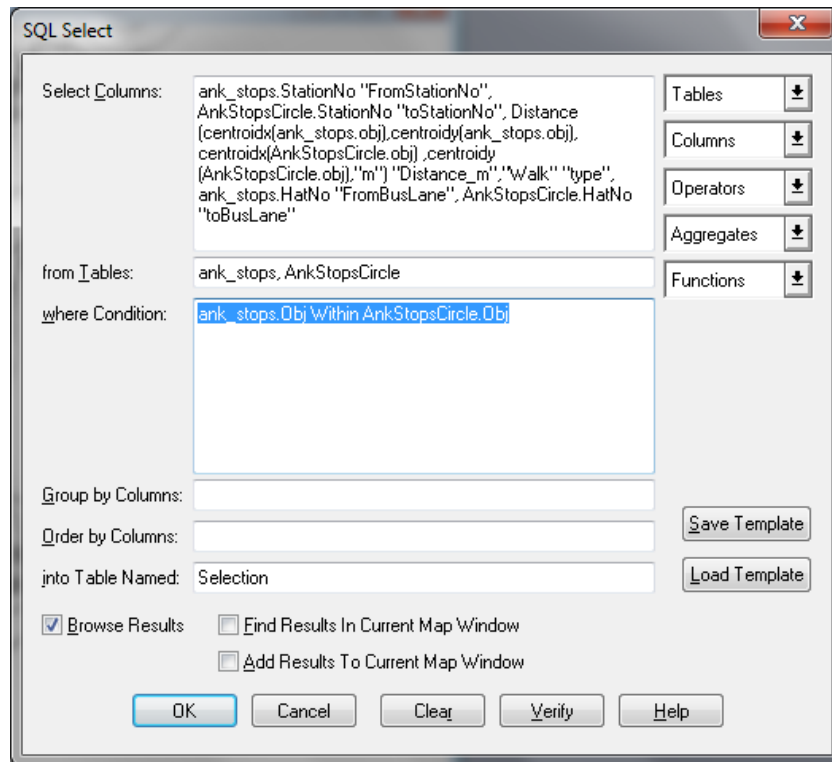


Figure 42 SQL Select text box

SELECT	ank_stops.StationNo "FromStationNo", AnkStopsCircle.StationNo "toStationNo", Distance(centroidx(ank_stops.obj),centroidy(ank_stops.obj), centroidx(AnkStopsCircle.obj) , centroidy(AnkStopsCircle.obj),'m') "Distance_m","Walk" "type", ank_stops.HatNo "FromBusLane", AnkStopsCircle.HatNo "toBusLane"
FROM	ank_stops, AnkStopsCircle
WHERE	ank_stops.Obj Within AnkStopsCircle.Obj

The result of buffering is represented as a layer on figures 43, 44 respectively.

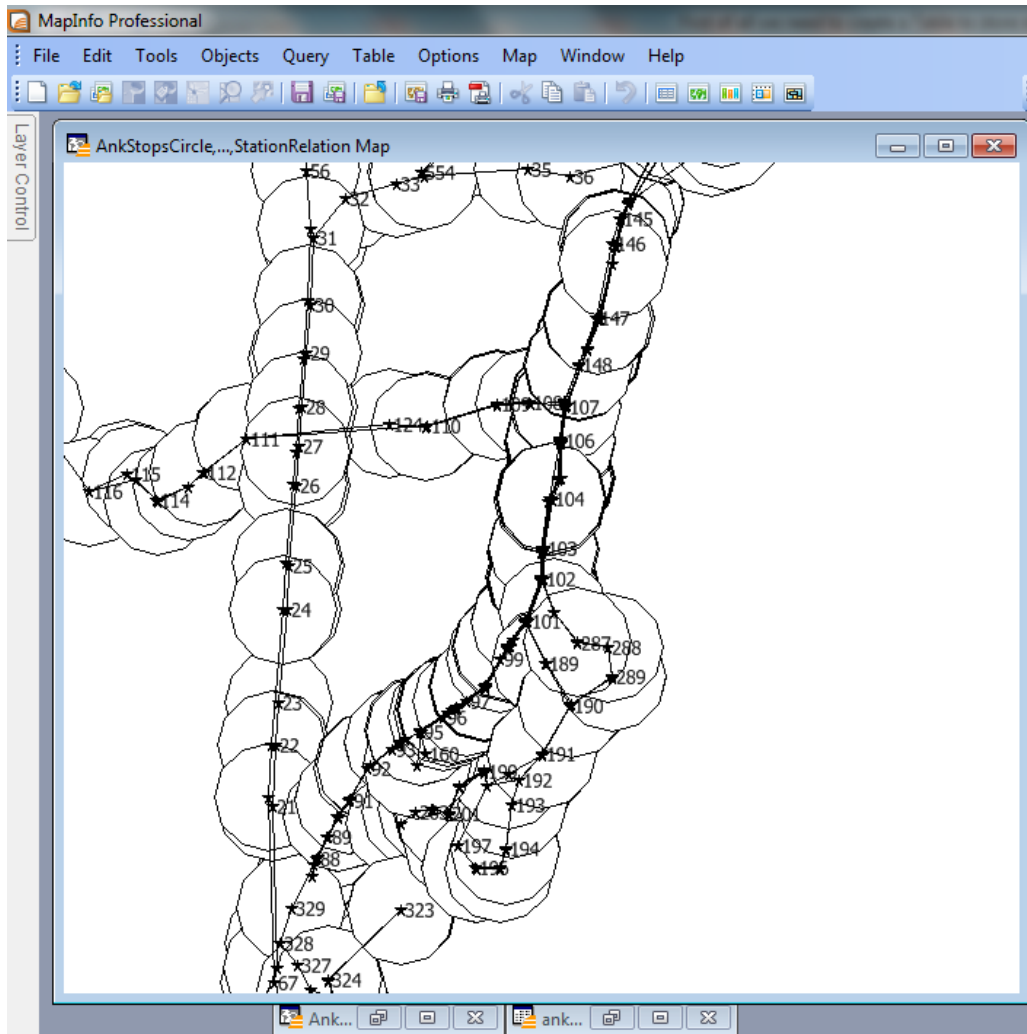


Figure 43 Buffer table layer

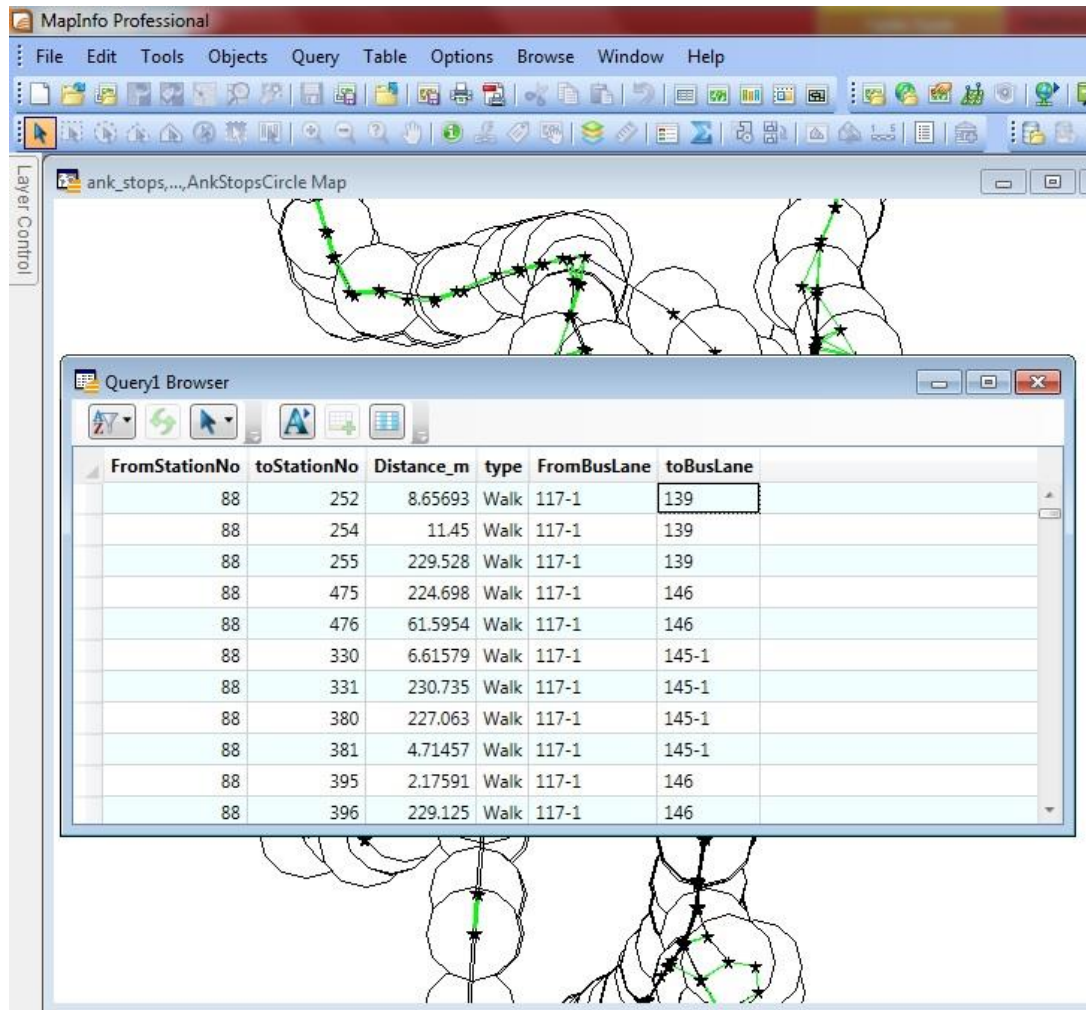


Figure 44 Buffer table result

This table can be saved as a text file to be used as main dataset in C++ application.

3.2 Implementation and Addressing the Problems

In this part, we divide the implementation into 4 steps to address the problems which listed in section 2.7. Firstly, we use the MapBasic to write a code to create a button pad called SetBusStops (Figure 45).

Create Buttonpad "SetBusStops" as

ToolButton Calling SetStartBusStop icon 116

ToolButton Calling SetendBusStop icon 103

PushButton Calling calcstations

PushButton Calling showfastestandlongest

n=10 'No Of Nearest Bus Stops





pa=ApplicationDirectory\$()

if right\$(pa,1) <> "\" then pa = pa + "\" end if



Figure 45 SetBusStops button pad

This pad contains 4 buttons appointed to the implementation steps:

- 1-  **SetStartBusStop** sub-routine.
- 2-  **SetEndBusStop** sub-routine.
- 3-  **calcstations** sub-routine.
- 4-  **showfastestandlongest** sub-routine

1- This step is used to detect the nearest bus stop to the start point, and used with step 2 to solve problems 2 and 3 in section 2.7.

Our approach to solve the problem 3 is to find several nearby bus stops to the start and end points when we implement the step 2 (Figure 46).

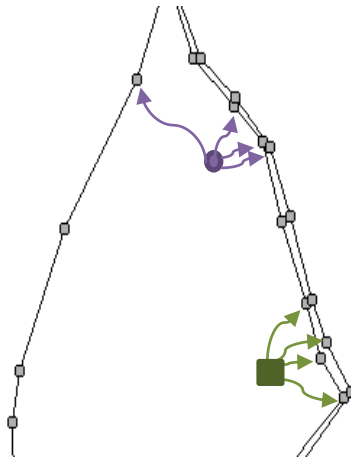


Figure 46 Several nearest bus stop

The support of this approach is as follows:

In Figure 23, if we want to go from square point to circle point, we spend a large path to reach, in addition spend long time, because we do not have a direct path to the destination. However, in case we have more than one nearest bus stops, we can choose another bus stop which has a direct path to the destination. In Figure 47, we do not use the nearest bus stop, we use bus stop on the opposite side, because it has a direct path to the destination.

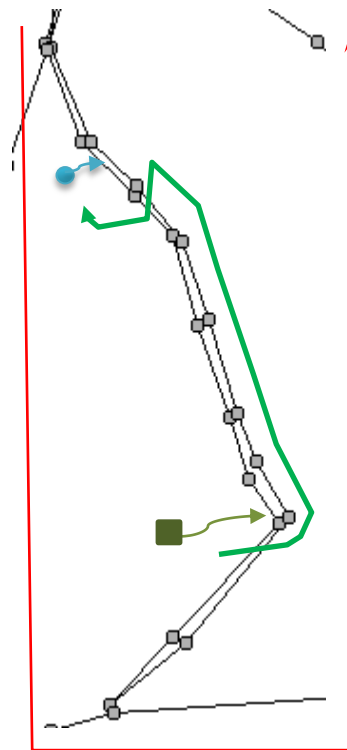


Figure 47 Choose another nearest bus stop

Example:

Figure 48 shows a captured aerial view from Google map [43]. If we want to go from the square point toward circle point, we do not use the nearest bus stop, we use bus stop on the opposite side of the road.

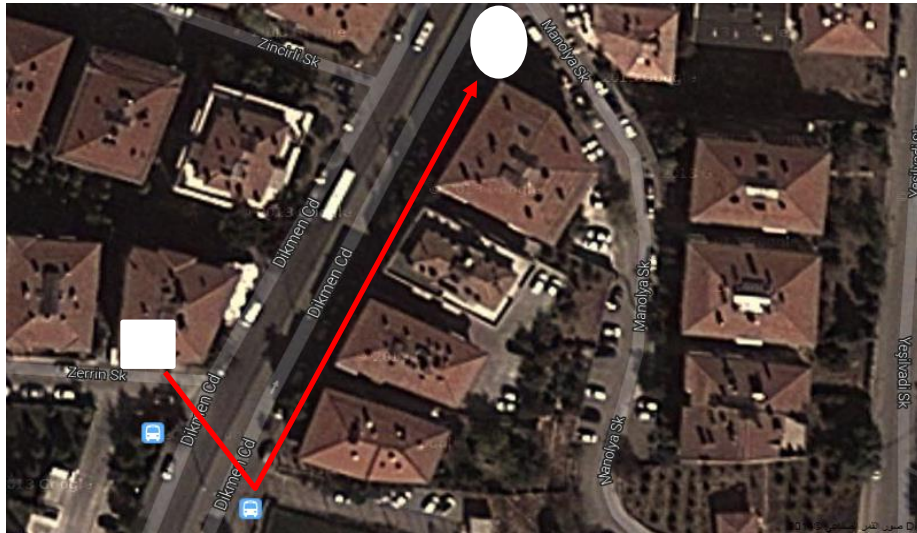



Figure 48 Example for choose a bus stop

The researcher wrote a MapBasic code to detect the source and destination point and to find 10 nearest bus stops for source and destination.

To create new MapBasic files choose new from the File menu.

We push on the first button  to call the **SetStartBusStop** sub routine, to detect the start point. The following code bellow shows **SetStartBusStop** sub routine.

```
Sub SetStartBusStop
  x= Commandinfo(1)
  y= Commandinfo(2)
  t = "NearestStationsStart"
  call FindNearestBusStop
End sub
```

As we see in the code above, we call **FindNearestBusStop** sub routine to find the nearest 10 bus stops to the start point. Figure 49 shows the icon pad.

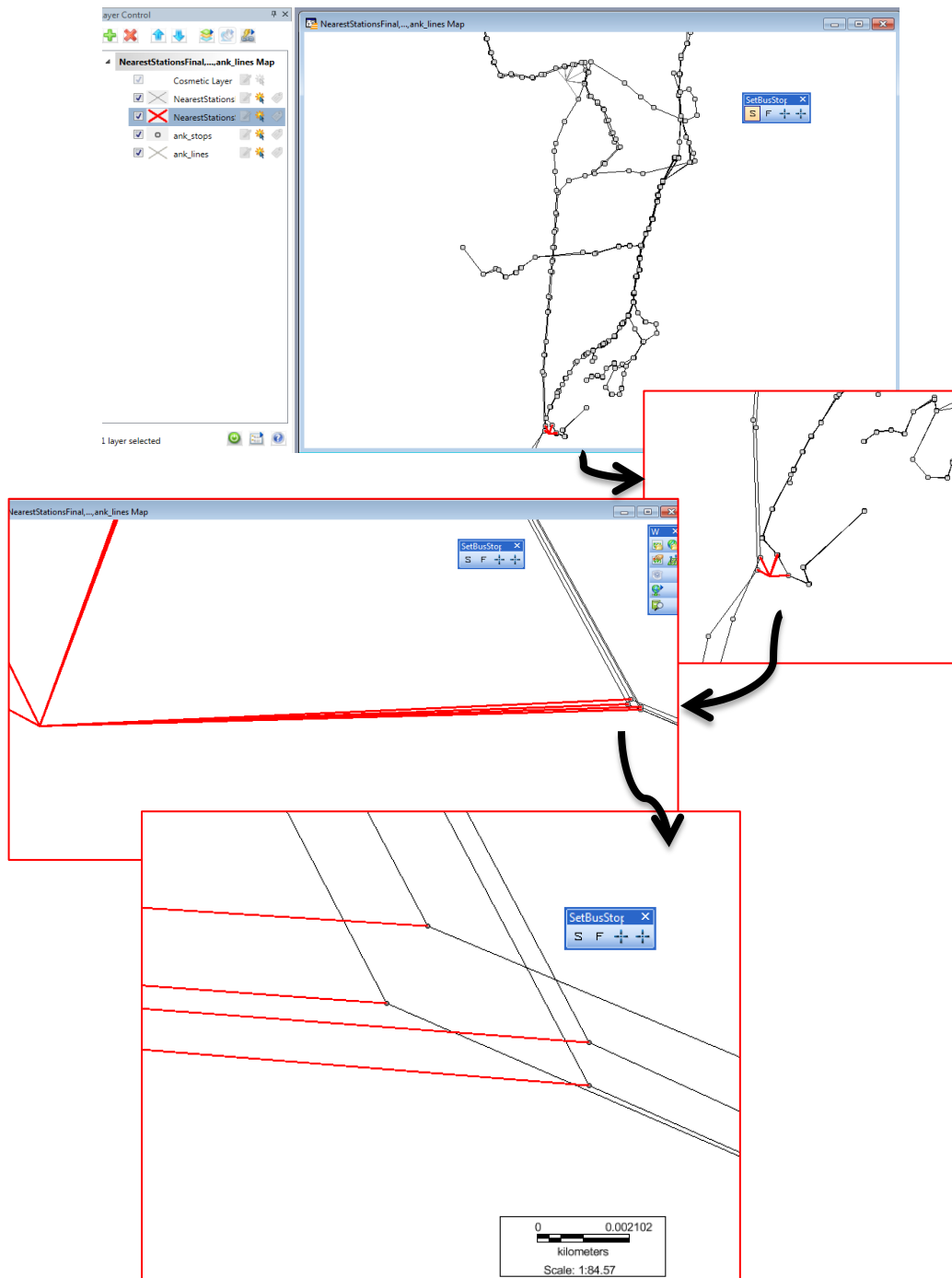


Figure 49 Push first button

The following code bellow shows **FindNearestBusStop** sub routine.

```
Sub FindNearestBusStop
  oobj= Createpoint(x,y)
  delete from t
  commit table t
  pack table t

  Nearest 10 From variable Oobj To ank_stops Into t Data col1=col1

  Update t set Distance_m = Objectlen(obj,"m")
  Distance(centroidx(obj),centroidy(obj),x,y,"m")

  p=Tableinfo(t,19)
  if right$(p,3) = "tab" then p = left$(p,len(p)-3) + "txt" end if

  print p + t + ".txt"
  Export t Into p Type "ASCII" Overwrite CharSet "WindowsTurkish"
end sub
```

As the result of this step, NearestStationsStart.txt is created (Figure 50). This text file stores the nearest bus stops in the first column and the distance to the start point in the second column (Figure 51).

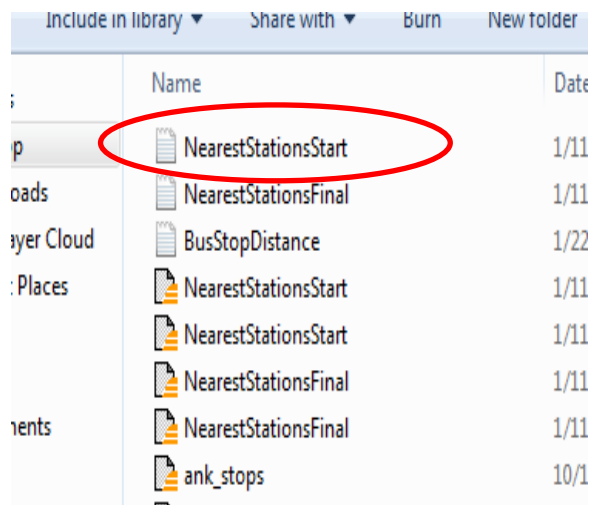


Figure 50 NearestStationStart text file

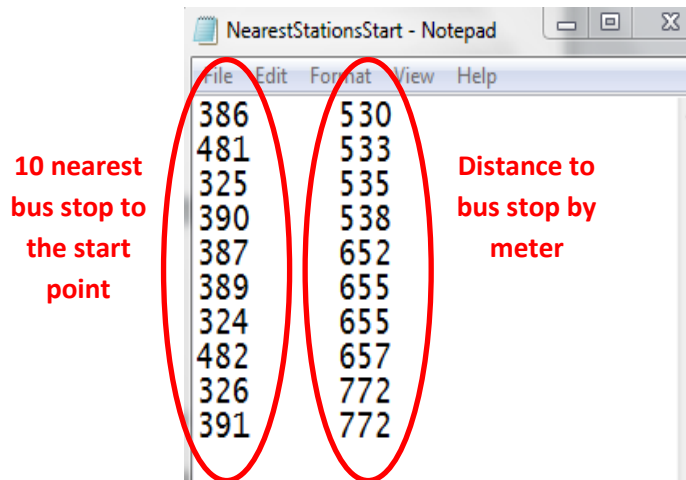


Figure 51 Content of NearestStationStart text file

- 2- The second step triggered with the F icon, is to detect the nearest bus stop to destination point (problem 2 in section 2.7), and to find several nearest bus stops to the destination point (problem 3 in section 2.7).

When we push on the second button  we call the **SetEndBusStop** sub routine to detect the end point.

The following code bellow shows **SetEndBusStop** sub routine.

```

Sub SetEndBusStop
    x= Commandinfo(1)
    y= Commandinfo(2)
    t = "NearestStationsFinal"
    call FindNearestBusStop
End sub

```

Again we can see the code invoking **FindNearestBusStop** sub routine, to find 10 nearest bus stops to the end point (destination). Figure 52 shows implement step 2.

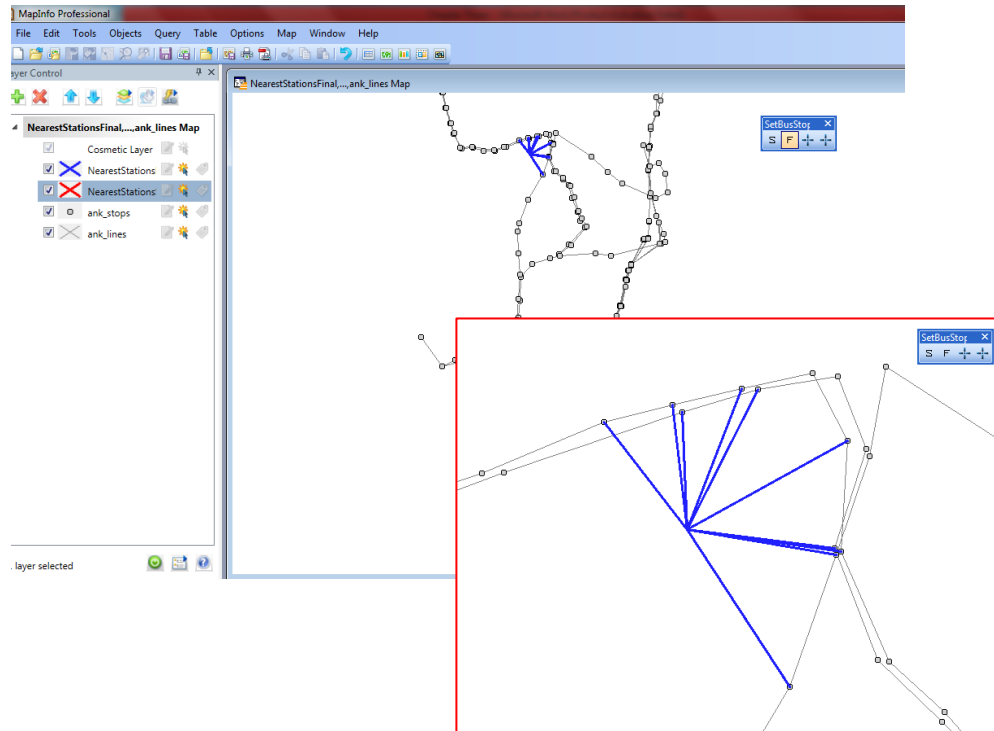


Figure 52 Push second button

The result for this step is a text file, NearestStationsFinal.txt shows on the Figure 53. This text file contains two columns, to represent the nearest bus stops to destination point and the distances to the final point (Figure 54).

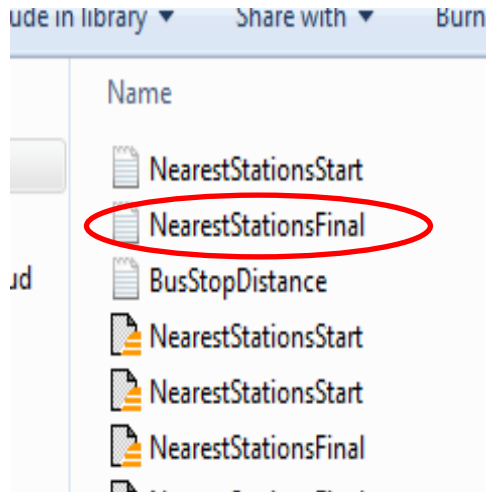


Figure 53 NearestStationFinal text file

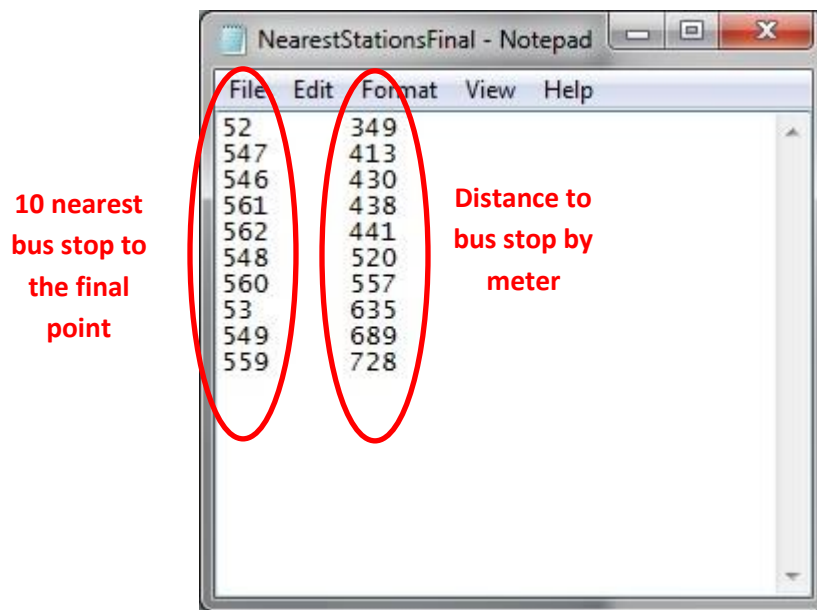


Figure 54 Content of NearestStationFinal text file

- 3- This step is to find the shortest paths between source and destination (problem 1 in section 2.7). In addition we solve the problem 5 in section 2.7 by using specific function. This step starts when we press the first start icon, and call the **calcstations** sub routine.

```
sub calcstations

call closefinalfile
if fileexists(pa+"FinalResult.txt") then kill pa + "FinalResult.txt" end if
if fileexists(pa+"NearestStationsStart.txt") and
fileexists(pa+"NearestStationsFinal.txt") then
run program pa + "dijkstra.exe"
end if

end sub
```

In MapBasic code above we can see invoking the dijkstra.exe file.

To create a new project in C++ go to file menu and choose new>project. After that choose windows application to create standard windows application, or we can choose Console Application as shown on Figure 55.

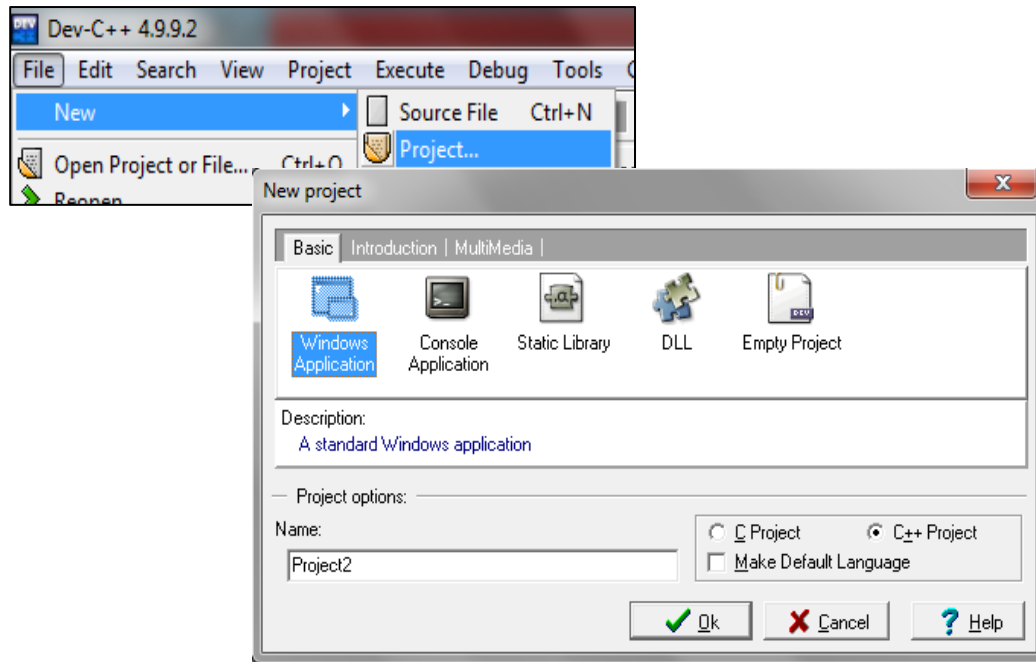


Figure 55 Create new project and choose windows application

Firstly, we approach the problem 5 given in section 2.7 by using `<sstream>` library and `(ifstream)` instructions to read data from text file.

We generate two dimensional arrays, called `MasterFile` to save all data read from the file `BusStopDistance.txt`. The `MasterFile` array contains 6 columns:

Column 1: Source bus stop.

Column 2: Destination bus stop

Column 3: Distance between source and destination bus stop.

Column 4: Transport type (Bus/Walk).

Column 5: Bus line.

Column 6: Time needed to reach from source to destination. The values in this column are calculated based on the transportation type.

The followed code below, shows how can be read from text file.

```
// Read From Text File (FromS=Source station, ToS=Destination station,
                        Dis=Distance, BR1=Bus line, Ty=Transit Type)
int FromS,ToS,Dis,BR1;
string Ty;
double MasterFile[n][6];
ifstream Read("BusStopDistance.txt");
if (Read.is_open()) {
    for(i=0;i<n;i++) {
        Read >> FromS;
        Read >> ToS;
        Read >> Dis;
        Read >> Ty;
        Read >> BR1;
        MasterFile[i][0]=FromS;
        MasterFile[i][1]=ToS;
        MasterFile[i][2]=Dis;
        if (Ty=="Walk") {
            MasterFile[i][3]=1;MasterFile[i][5]=MasterFile[i]
            [2]/4/1000*3600; }
        else {
            MasterFile[i][3]=2;MasterFile[i][5]=MasterFile[i]
            [2]/30/1000*3600;}
        if (MasterFile[i][5]==0) MasterFile[i][5]=1;
        MasterFile[i][4]=BR1;
    }
}

Read.close();
// End to Read From Text File
```

After we detect the number of bus stops, we store it in **BusStopDistance.txt** as a one dimensional array called **BusStops**. It is sorted in ascending order.

```
// Detect Number Of BusStop (BN=Number of bus stops)
short BusStop[n]; bool Flag;
for(i=0;i<n;i++) BusStop[i]=0;
int BN=0;
for (i=0;i<n;i++){
    Flag=true;
    for (j=0;j<n;j++){
        if (MasterFile[i][0]==BusStop[j]) { Flag=false; break; }
    }
    if (Flag) { BusStop[BN]=MasterFile[i][0]; BN=BN+1; }
}
for (i=0;i<n;i++){
    Flag=true;
    for (j=0;j<n;j++){
        if (MasterFile[i][1]==BusStop[j]) { Flag=false; break; }
    }
    if (Flag) { BusStop[BN]=MasterFile[i][1]; BN=BN+1; }
}

// Arrange BusStop Array
int R;
for(i=0;i<BN-1;i++)
for(j=i;j<BN;j++)
    if(BusStop[i]>BusStop[j]){
        R=BusStop[i];
        BusStop[i]=BusStop[j];
        BusStop[j]=R;
    }

// End Arrangement
// End Detect Number Of BusStop
```

After that we create a bus stop distance array (BusStopDis) to be computed through Dijkstra algorithm as explained in section 2.4.

Following is the C++ code to create BusStopDis array.

```
// Create BusStopDistance

short BusStopDis[BN][BN];

for(i=0;i<BN;i++)

    for(j=0;j<BN;j++) BusStopDis[i][j]=29999;

int Pi=0,Pj=0;

for(i=0;i<n;i++)

    {

        for(j=0;j<BN;j++)

            {

                BusStopDis[j][j]=20000;

                if (MasterFile[i][0]==BusStop[j]) Pi=j;

                if (MasterFile[i][1]==BusStop[j]) Pj=j ;

            }

        BusStopDis[Pi][Pj]=MasterFile[i][2];

    }

// End BusStop Distance
```

Use the **ofstream** instructions to create text file called **FinalResults.txt**. This file is used to save the implementation results.

As following as below shows the C++ code for creating **FinalResult.txt**.

```

ofstream Write("FinalResult.txt",ios::app);
    if(Write.is_open()){
        Write<<SA[Mi][0]<<"\t";
        Write<<FA[Mj][0]<<"\t";
        Write<<ExportArray[Mi][Mj]+SA[Mi][1]+FA[Mi][1]<<"\t";
        for (int jj=0;jj<BN;jj++){
            if(sta[jj]==9999) continue;
            Write<<sta[jj]<<" ";
        }
        Write<<"\n";
    }
Write.close();

```

Figure 56 shows **FinalResult.txt** file.

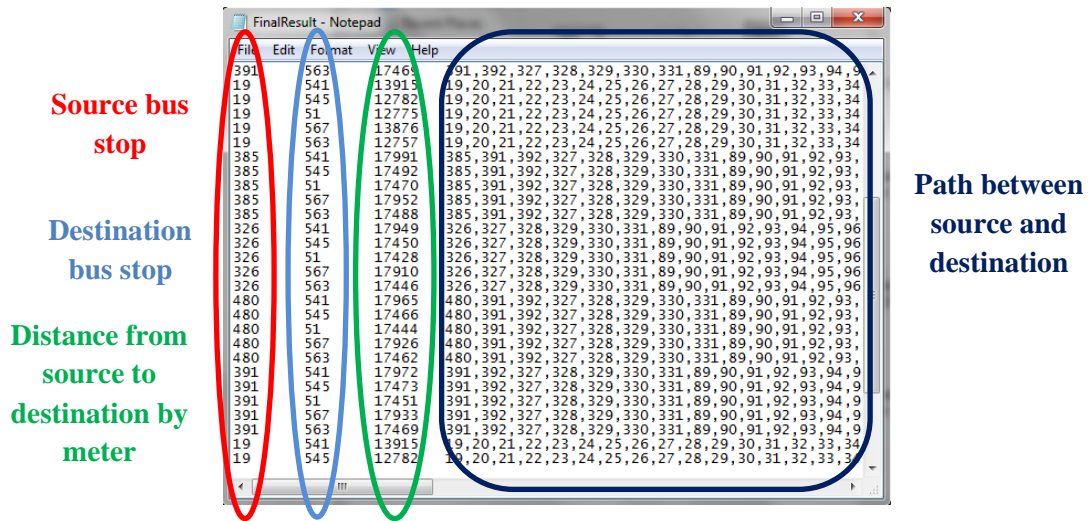


Figure 56 FinalResult text file

Figure 57 shows the third button to run the C++ project. By using a personal computer with processor Intel(R) Core™ i3-2310 CPU @ 2.10 GHz, 8.00 RAM, and 64-bit Windows 7 Home Premium. We can see the command screen appearing 3 seconds to run C++ project, and disappear.

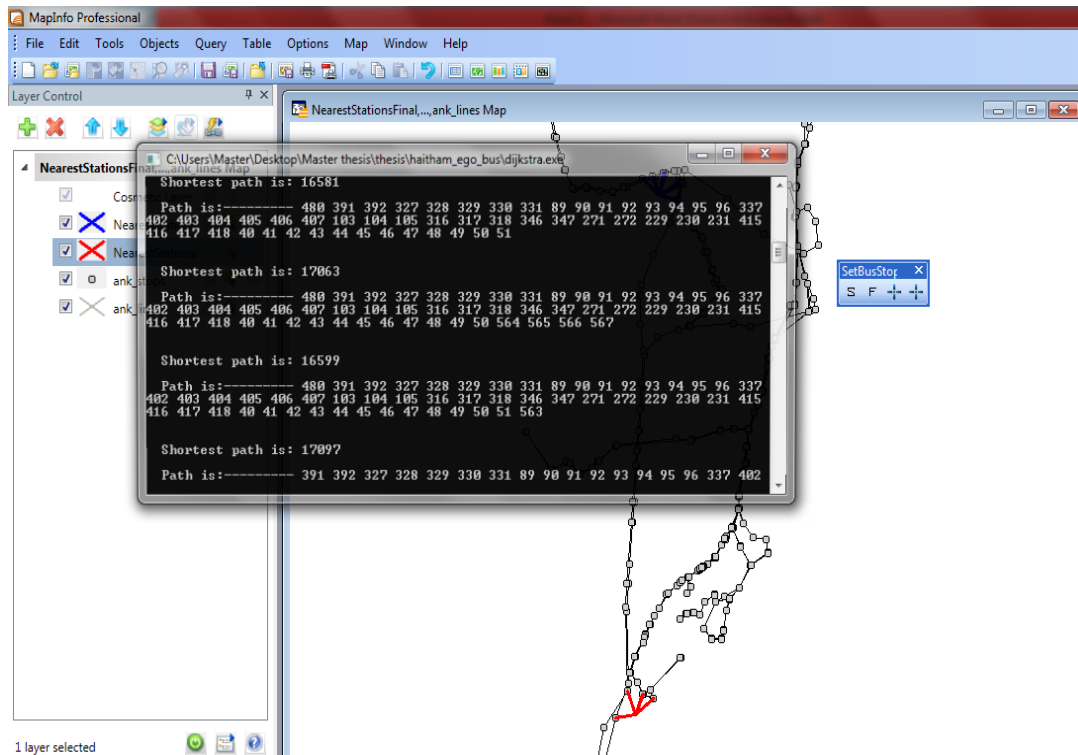



Figure 57 Push Third button in SetBuStops pad

- 4- The fourth step start when we press  icon. It helps to solve problem 6 which listed in section 2.7. We use the **FinalResult.txt** to find the shortest and longest paths by using a MapBasic sub-routine, **showfastestandlongest**. Basically, we choose the less value in column 3 to represent the shortest path between the source and destination bus stops, and choose the largest value to represent the long path.


```

sub showfastestandlongest
dim s as string
call closefinalfile
if not fileexists(pa+"FinalResult.txt") then note "File " + pa + "FinalResult.txt
not found" exit sub end if
Register Table pa+"FinalResult.txt" TYPE ASCII Delimiter 9 Charset
"WindowsArabic" Into pa+"FinalResult.TAB"
Open Table pa+"FinalResult.TAB" Interactive
select * from FinalResult order by _COL3 into selsort
fetch first from selsort
s=selsort.col4
run command "Select * from ank_stops where StationNo=any("+left$(s,len(s)-1)
+") into ShortestRoute noselect"
fetch last from selsort
s=selsort.col4
run command "Select * from ank_stops where StationNo=any("+left$(s,len(s)-1)
+") into LongestRoute noselect"

if numwindows() =0 then map from ank_stops end if
if windowinfo(frontwindow(),3) <>1 then map from ank_stops end if
Add Map Auto Layer LongestRoute
Set Map Layer 1 Display Global Global Symbol (33,16711680,12)

Add Map Auto Layer ShortestRoute
Set Map Layer 1 Display Global Global Symbol (34,65280,12)

end sub

```

Figure 58 below shows the shortest path calculated.

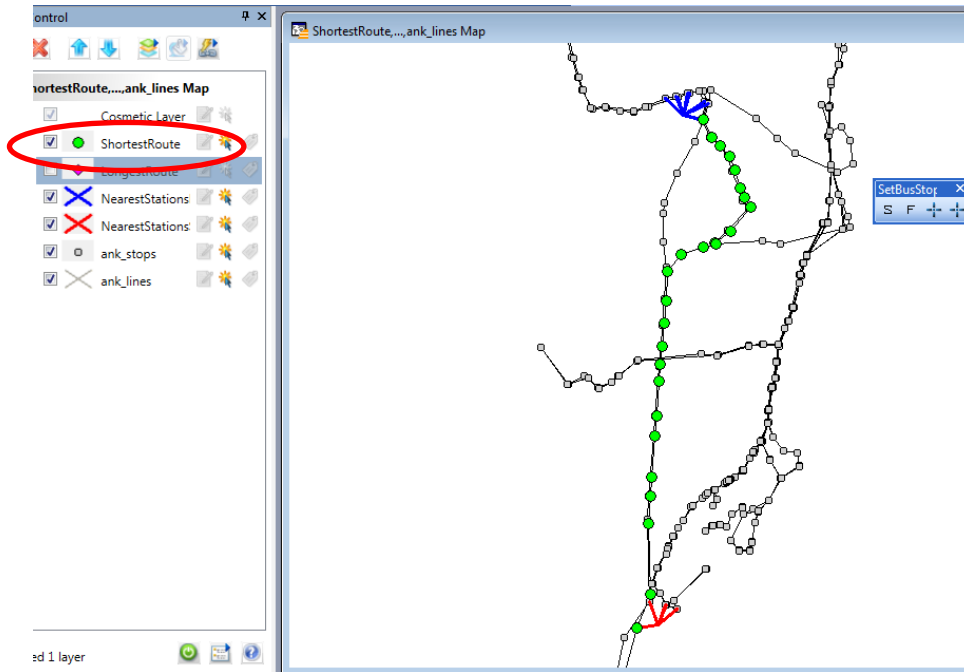


Figure 58 Shortest path

Figure 59 below shows the longest path calculated.

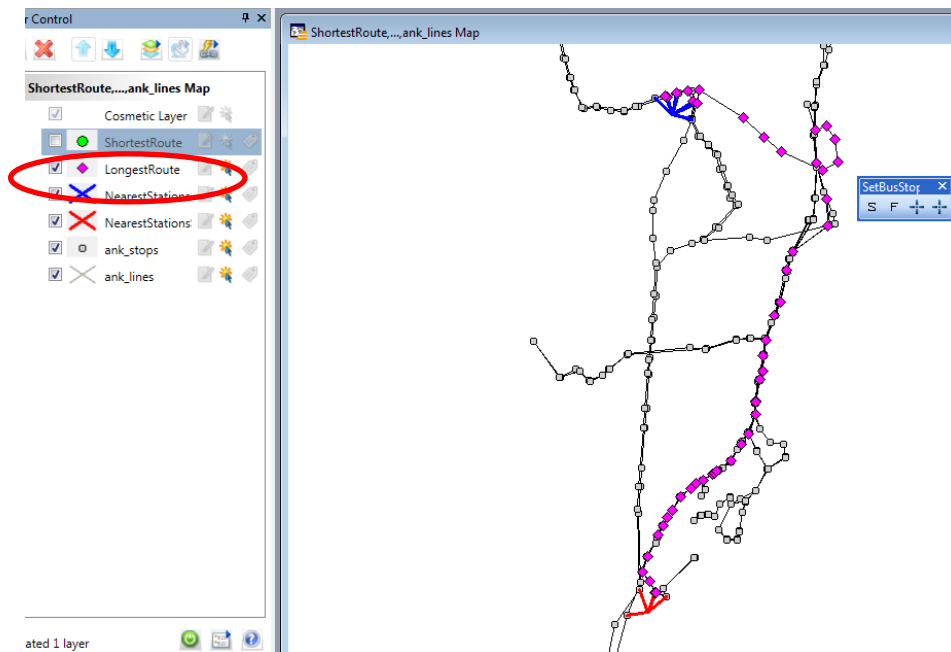


Figure 59 Longest path

CHAPTER IV

CONCLUSION

As we showed in our literature survey in section 2.6 and case study about Ankara transportation in chapter 3, there are underlined potential problems in the implementation, and we tried to address and proposed solution for them.

In this chapter, we present the findings and associated limitations of this study; the chapter also includes follow-up future studies.

Finally, we present the conclusion for this thesis to address the research question.

4.1 Findings and Results

Results that appeared in chapter 3 are not limited to the research questions listed in chapter 1; hence, there are additional results that appeared for future studies.

The following list contains what we can do with our datasets:

- 1.** Successful study of Ankara as a case. In chapter 1, we presented the transportation system in Baghdad and Ankara and saw the similarities that enable us to use Ankara's transportation system as a sample dataset.
- 2.** Clean and up to date datasets need to be collected in order to start planning mass transport system. Primary datasets should be spatial and nonspatial datasets of residential, commercial and industrial areas, roads, traffic limitation such as peak hours and demographic datasets including population density.

3. The literature review in section 2.6 showed that GIS can be employed in developing a transportation system in many countries and graph theory can be used to solve the major problem in transport field called “the shortest path problem” by using several algorithms. We used Dijkstra algorithm in this thesis to find more than one possible shortest path. In addition, literature shows several software such as ArcView as a GIS software and use other programming languages such as Java. In this thesis, we successfully used MapInfo with C++ program language for our data and obtained plausible results.
4. Addressing potential problems. In section 2.7, we show the potential problems in implementation. These problems are related to dealing with sample dataset. Thus, in chapter 3, we showed how to analyze these data and how to solve these problems.
5. Successful use of Dijkstra algorithm. We were successful in using the Dijkstra algorithm to find alternative shortest paths and showed the differences between shortest and longest paths.

4.2 Limitations

In this thesis we considered following limitations:

- 1- Choosing Ankara as a case study: there are several reasons for choosing Ankara. First, the researcher is currently studying and living in Ankara. Second, as we showed in section 1.2, Ankara has similarities with Baghdad such as topography of both cities’ central parts are quite flat, there is a variety of transport systems and both places are populated center. However, the most important reason is that The Electricity, Gas and Bus General Directorate (EGO) uses the modern techniques such as GIS to develop Ankara’s transport system. Additionally, Baghdad does not have any rail transport system and

Ankara does not have any water transport system. The sample data used in this thesis are just for academic purposes and they are datasets of a limited number of bus stations in Ankara; there is no guarantee they are up to date. This limitation is related with finding 1.

2- We used straight lines to connect bus stops instead of real roads; for this reason, the connections do not reflect real distances between bus stops which limits findings 4 and 5. With this limitation, we have used vertex and node terms interchangeably.

3- Findings 3 and 5 have the following limitations:

The software used: the availability of the software used by specialized experts working in this field is a major reason for choosing it. In this thesis, the researcher used MapInfo and MapBasic in the implementation assisted by the expertise of Başarsoft company. In addition, the researcher has a background in using C++ to write program codes for implementing the used algorithm.

The algorithms used: as can be seen in section 2.3, there are many algorithms that can be used to find the shortest path, however the researcher has chosen Dijkstra algorithm, because it used more frequently in the field, and the time limitation of the research did not allow the researcher to apply more algorithms.

4. While determining closest bus stops we have chosen to limit the alternatives as five in step 3 in section 3.2. This number is a practically chosen number to reduce running time of the application.

4.3 Future Studies

The importance of this thesis lies in the use of GIS as well as graph theory to solve the shortest path problem in developing the public transport system. For this reason, we think there are more research sectors to be done it in the future. Some of them are as follow:

- 1- The major help this study is offering is a way to develop the transport system in Iraq. In association with this, there are some required preparations in Iraq, such as:
 - Provide a comprehensive look for the world experiences in public transport, to help the Iraqi chiefs to establish a plan to prepare all requirements such as collecting the datasets for all old bus stations, and thinking for their re-distribution based on the population distribution.
 - Provide appropriate budget for buying the necessary hardware and software.
 - Establish programs for population readiness to use the modern technologies through the establishment of public seminars and university lectures, in addition to providing free workshops to use these applications. As for the data, we must use legal issues to collect and use the datasets which important for implementing any project in the government, and to guarantee the gaining of result; so, the researcher must contact and share with all government institutions in Iraq, which are working in the country's transport systems.
- 2- Finding 2 and limitation 2. When we want to build more realistic model and simulation, polylines that overlay roads should be used instead of straight lines between bus stops. In this case, the distinction between node and vertex should be considered.
- 3- Related with findings 3 and 5 and limitation 3, the researcher can use another GIS software such as gvSIG [44] and use Java language as well as another

algorithm such as Floyd-Warshall algorithm to solve the shortest path problem to see differences in the results obtained in this thesis. The researcher is also thinking to merge all shortest path algorithms to see the advantage of any algorithm, and create a hybrid algorithm that can be used to find the shortest path in public transportation system.

- 4- In section 2.6 and related finding 4 and limitation 3, there are several studies used to develop the public transport system, such as in [31] where the authors reduced the iterations used in the traditional algorithm, and in [36] the authors combined factors on travel habits to verify and analyze the constraining factors the passengers faced in traveling decision-making; hence, the researcher has to consider collecting the advantages of these studies and implement them in the public transport system of Iraq.
- 5- To address the limitation 4, a further analysis can be done to optimize number of alternative bus stops nearby. These analyses should include further programming optimization to reduce resource consumption.
- 6- Another idea we are thinking about is taking into consideration the time schedule for the buses and to find the earlier buses beside the shortest path.
- 7- Developing a web and mobile applications for end users to finding a shortest path and make a stress test with many online users inquiring.

4.4 Conclusion

Using graph theory and specifically shortest path algorithm to solve shortest path problems, requires clean datasets for reliable analysis results.

From this thesis, we learnt about mass transport systems not limited to the following items but primarily:

- Required data for analyses
- How to survey potential problems
- Use of GIS along with the application tools to address these problems

About the research question:

Can Iraq authorities use graph theory and shortest path algorithm to build a system to develop the public transit system including exchanging lines and walking, based on the experiences of transport system in Turkey?

We believe the results obtained in our implementation support our hypothesis that Iraq can restart designing transportation system by starting to address the issues pointed in this thesis.

REFERENCES

1. **Maps of World Web Site, (2014),** “*Iraq Road Map*”, <http://www.mapsofworld.com/iraq/road-map.html>, (Data Download Date: 15.05.2014).
2. **Al-mashhadani A. A., (2013),** "*Terminals in the History of Passenger Buses in Baghdad*", <http://www.algardenia.com/tarfiya/menouats/6599-2013-09-29-09-43-33.html>, (Data Download Date: 12.03.2014).
3. <http://www.pt-tm.gov.iq>, (Data Download Date: 18.05.2014).
4. <http://www.amanatbaghdad.gov.iq>, (Data Download Date: 18.05.2014).
5. **Central Statistical Organization-Ministry of Planning-Iraq, (2010),** “*Iraq's Governorates by Area & Their Relative Share of Area & Population: 1997, 2009*”, <http://www.cosit.gov.iq/en/population-manpower-staatistics/life>.
6. **Mirza H. A., (2013),** “*The Development of Public Transport and Use of Clean Fuel*”, Citizen Newspaper-Iraq, 1418, <http://www.almowatennews.com/index.php/2013-04-19-21-25-43/1418-2013-06-26-16-19-54.html>, pp. 1-4.
7. **Turkey Statistical Institute Web Site, (2014),** “*The Population of the Metropolitan Municipalities and the Municipalities - 2013*”, http://rapor.tuik.gov.tr/reports/rwservlet?adnksdb2&ENVID=adnksdb2Env&report=wa_buyukbelediye.RDF&p_kod=1&p_yil=2013&p_dil=1&desformat=html, (Data Download Date: 12.05.2014).

8. **EGO Web Site, (2014), "Rail Systems"**
<http://www.ego.gov.tr/EN/newsreadEN.asp?id=3224>, (Data Download Date: 14.05.2014).
9. <http://www.mapinfo.com/>, (Data Download Date: 15.10.2013)
10. <http://www.bloodshed.net>, (Data Download Date: 15.10.2012).
11. **Kleinberg J., Tardos É., (2006), "Algorithm Design"**, Pearson Education Inc., pp. 73-78.
12. **Liu L., (2011), "Data Model and Algorithms for Multimodal Route Planning with Transportation Networks"**, Technische Universität München, pp. 9-13.
13. **Cargal J. M., (1988), "Discrete Mathematics for Neophytes: Number Theory, Probability, Algorithms, and Other Stuff"**, chapter 9.
14. **Sedgewick R., Wayne K., (2011), "Algorithms"**, 4th Edition, Princeton University, Pearson Education, Inc., pp. 638-651
15. **Sanan S., Jain L., Kappor B., (2013), "Shortest Path Algorithm"**, International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 7, ISSN 2319-4847, pp.316-320,
<http://www.ijaiem.org/volume2issue7/IJAIEM-2013-07-23-079.pdf>, (Data Download Date: 25.12.2014).
16. **Hoare T., (2003), "Edsger Wybe Dijkstra"**, Physics Today, pp. 96-98,
<http://scitation.aip.org/content/aip/magazine/physicstoday/article/56/3/10.1063/1.1570789>, (Data Download Date: 31.12.2013).
17. **Jungnickel D., (2005), "Graphs, Networks and Algorithms"**, 2nd Edition, Springer-Verlag Berlin Heidelberg, pp. 75-79.

18. **Ruohonen K., (2013)**, "*Graph Theory*", Translation by Janne Tamminen, Kung-Chung Lee and Robert Piché, pp. 61-63.
19. **ESRI Web Site, (2012)**, "What is GIS?", pp. 6-30,
<http://www.esri.com/~media/Files/Pdfs/library/bestpractices/what-is-gis.pdf>,
(Data Download Date: 15.03.2013).
20. **Buckley D. J., (1990)**, "*The GIS Primer: An Introduction to Geographical Information Systems*", Forestry Canada, pp. 14-25.
21. **Chai D., Zhang D., (2001)**, "*Algorithm and Its Application of N Shortest Paths Problem*", Institution of Space and Information Technology, Zhejiang University, Hangzhou, Zhejiang, pp. 1-6.
22. **Zhang F., Liu J., (2009)**, "*An Algorithm of Shortest Path Based on Dijkstra for Huge Data*", Chinese Academy of Surveying and Mapping, Beijing, China, International Conference on Fuzzy Systems, pp. 1-4.
23. **Huang Z., Ding Y., Li J., (2009)**, "*A GIS-based Accessibility Modeling Process for Estimating Transit Travel Demand*", School of Urban Design Wuhan University Wuhan, China, pp. 1-4.
24. **Ma J., Yu X., Chen G., Wang J., Pi Y., (2010)**, "*Research on Urban Accessibility Distribution Areal Model by Floyd Algorithm and Kriging Interpolation*", Department of Geographical Information Science, Nanjing University, China, Geoinformatics, 18th International Conference, pp. 1-4.
25. **Zhan F. B., (1997)**, "*Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures*", Journal of Geographic Information and Decision Analysis, pp. 69-82,
http://publish.uwo.ca/~jmalczew/gida_1/Zhan/Zhan.htm, (Data Download Date: 22.03.2014).

26. **Effat H. A., Hassan O. A., (2013),** "*Designing and Evaluation of Three Alternatives Highway Routes Using the Analytical Hierarchy Process and the Least-cost Path Analysis, Application in Sinai Peninsula, Egypt*", The Egyptian Journal of Remote Sensing and Space Science, pp. 141-151, <http://www.sciencedirect.com/science/article/pii/S1110982313000264>, (Data Download Date: 14.03.2014).
27. **Alazab A., Venkatraman S., Abawajy J., Alazab M., (2011),** "*An Optimal Transportation Routing Approach Using GIS-based Dynamic Traffic Flows*", 3rd International Conference on Information and Financial Engineering, pp. 172-778.
28. **Al-mumaiz M. O., (2012),** "*Paths Planning For AL-Ramdi Intercity Road Using GIS Tool*", Engineering College, University of Al-Mustansiriya/Baghdad, Eng. &Tech. Journal, Vol.30, No. 2, pp. 265-282, <http://www.iasj.net/iasj?func=fulltext&aId=25843>, (Data Download Date: 18.03.2014).
29. **Abbasi S., Moosavi F., (2012),** "*Finding Shortest Path in Static Networks: Using a Modified Algorithm*", Department of Industrial Engineering, Najafabad Branch, Islamic Azad University, Esfahan, Iran, International Journal of Finance & Banking Studies IJFBS, Vol.1 No.1, ISSN: 2147-4486, pp.29-34, <http://www.ssbfnct.com/ojs/index.php/ijfbs/article/viewFile/12/12>, (Data Download Date: 18.03.2014).
30. **Al-joboory B. S., Al-bakry M. M., Al-hamadany O. Y., (2006),** "*The Selection of Optimum Road Path Using Geographic Information System (GIS)*", Journal of Engineering, Number 2 Volume 12, University of Baghdad, College of Engineering, Department of Surveying, pp. 295-303, <http://www.iasj.net/iasj?func=fulltext&aId=45065>, (Data Download Date: 18.03.2014).

- 31. Kadry S., Abdallah A., Joumaa C., (2012), "On the Optimization of Dijkstra's Algorithm", Informatics in Control, Automation and Robotics. Springer Berlin Heidelberg, Springer Berlin Heidelberg, pp. 393-397.**
- 32. Baker S. L., (2004), "Critical Path Method (CPM)", pp. 1-9, <http://hspm.sph.sc.edu/COURSES/J716/CPM/CPM.html>, (Data Download Date: 01.04.2014).**
- 33. Ravi N., Sireesha V., (2010), "Using Modified Dijkstra's Algorithm for Critical Path Method in a Project Network", International Journal of Computational and Applied Mathematics, Volume 5 number 2 pp. 217-225.**
- 34. Yong B., Hongping H., (2010), "Mathematical Model of Best-path Planning Algorithms for Public Transportation Systems", International Conference on Computer Application and System Modeling (ICCASM), vol. 13, pp. 345-348.**
- 35. Kanatani N., Sasama T., Kawamura T., Sugahara K., (2010), "Development of Bus Location System Using Smart Phones", SICE Annual Conference 2010, pp. 2432-2433, <http://www.keisana.ike.tottori-u.ac.jp/publications/papers/368.pdf>, (Data Download Date: 02.04.2014).**
- 36. Yüewen L., Zhong W., (2012), "Research of Intelligent Model of Optimal Route for the Urban Public Transport", Second International Conference on Business Computing and Global Informatization (BCGIN), pp. 695-698, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6382628>, (Data Download Date: 02.04.2014).**
- 37. Mandal A., Deshmukh S. G., (1994), "Vendor Selection Using Interpretive Structural Modelling (ISM)" International Journal of Operations & Production Management 14.6, pp. 52-59, <http://www.emeraldinsight.com/journals.htm?articleid=848794>, (Data Download Date: 25.03.2014).**

- 38. Li Y., Xing J., Huang G., Meng L., (2010),** "*Least Transfer Cost Model for Optimizing Public Transport Travel Routes*", 2nd International Conference on Signal Processing Systems (ICSPS), Vol. 2, pp. 828-831, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5555864>, (Data Download Date: 02.04.2014).
- 39. Martinwz H., Mauttone A., Urquhart M. E., (2013),** "*Frequency Optimization in Public Transportation Systems: Formulation and Metaheuristic Approach*", European Journal of Operational Research 236, pp. 27-36, <http://www.sciencedirect.com/science/article/pii/S0377221713009065>, (Data Download Date: 02.04.2014).
- 40. Sun C., Pallottino S., (2003),** "*Circular Shortest Path in Images*", Pattern Recognition 36, pp. 709-719, <http://www.sciencedirect.com/science/article/pii/S0031320302000857>, (Data Download Date: 18.03.2014).
- 41. Hadas Y., Ranjitkar P., (2012),** "*Modeling Public-transit Connectivity with Spatial Quality-of-transfer Measurements*", Journal of Transport Geography 22, pp. 137-147, <http://www.sciencedirect.com/science/article/pii/S0966692311002274>, (Data Download Date: 03.04.2014).
- 42. MapBasic 11.5 Reference, (2012),** Pitney Bowes Software Inc., <http://reference.mapinfo.com/software/mapbasic/english/11.5/MapBasicReference.pdf>, (Data Download Date: 17.01.2014)
- 43. <https://maps.google.com>,** (Data Download Date: 26.02.2014).
- 44. Steiniger S., Bocher E., (2009),** "*An Overview on Current Free and Open Source Desktop GIS Developments*" International Journal of Geographical Information Science, 23(10), pp. 1345-1370.

APPENDIX A

MapBasic Code

The following below show the MapBasic code for implement and solve all problems and analysis.

```
include "mapbasic.def"
```

```
Declare sub calcstations
```

```
Declare sub showfastestandlongest
```

```
Declare sub closefinalfile
```

```
Declare Sub FindNearestBusStop
```

```
Declare Sub SetStartBusStop
```

```
Declare Sub SetEndBusStop
```

```
dim t,p,pa as string
```

```
dim x,y as float
```

```
dim n,i,j as integer
```

```
dim oobj as object
```

```
Create Buttonpad "SetBusStops" as
```

```
    ToolButton Calling SetStartBusStop icon 116
```

```
    ToolButton Calling SetendBusStop icon 103
```

```
    PushButton Calling calcstations
```

```
    PushButton Calling showfastestandlongest
```

n=10 'No Of Nearest Bus Stops

pa=ApplicationDirectory\$()

if right\$(pa,1) <> "\" then pa = pa + "\" end if

Sub SetStartBusStop

x= Commandinfo(1)

y= Commandinfo(2)

t = "NearestStationsStart"

call FindNearestBusStop

End sub

Sub SetEndBusStop

x= Commandinfo(1)

y= Commandinfo(2)

t = "NearestStationsFinal"

call FindNearestBusStop

End sub

Sub FindNearestBusStop

oobj= Createpoint(x,y)

delete from t

commit table t

pack table t

Nearest 10 From variable Oobj To ank_stops Into t Data coll=coll

Update t set Distance_m = Objectlen(obj,"m")'

p=Tableinfo(t,19)

if right\$(p,3) = "tab" then p = left\$(p,len(p)-3) + "txt" end if

print p + t + ".txt"

Export t Into p Type "ASCII" Overwrite CharSet "WindowsTurkish"

end sub


```

sub calcstations
    call closefinalfile
    if fileexists(pa+"FinalResult.txt") then kill pa + "FinalResult.txt" end if
    if fileexists(pa+"NearestStationsStart.txt") and
fileexists(pa+"NearestStationsFinal.txt") then
        run program pa + "dijkstra.exe"
    end if
end sub

sub closefinalfile
    dim k as integer
    for k = numtables() to 1 step -1
    if tableinfo(k,1) = "FinalResult" then
        Close Table FinalResult
        exit for
    end if
    next
end sub

sub showfastestandlongest
    dim s as string
    call closefinalfile
    if not fileexists(pa+"FinalResult.txt") then note "File " + pa + "FinalResult.txt not
found" exit sub end if
    Register Table pa+"FinalResult.txt" TYPE ASCII Delimiter 9 Charset
"WindowsArabic" Into pa+"FinalResult.TAB"
    Open Table pa+"FinalResult.TAB" Interactive
    select * from FinalResult order by _COL3 into selsort
    fetch first from selsort
    s=selsort.col4
    s=selsort.col4

```

```
run command "Select * from ank_stops where StationNo=any("+left$(s,len(s)-1) +")
into ShortestRoute noselect"
fetch last from selsort
s=selsort.col4
run command "Select * from ank_stops where StationNo=any("+left$(s,len(s)-1) +")
into LongestRoute noselect"
if numwindows() =0 then map from ank_stops end if
if windowinfo(frontwindow(),3) <>1 then map from ank_stops end if
Add Map Auto Layer LongestRoute
Set Map Layer 1 Display Global   Global Symbol (33,16711680,12)
Add Map Auto Layer ShortestRoute
Set Map Layer 1 Display Global   Global Symbol (34,65280,12)

end sub
```

APPENDIX B

C++ Code

```
#include <iostream>
#include <fstream>
#include <sstream>
using namespace std;
int main()
{
    int i,j,n=3367;
    // Read From Text File (FromS=Source station, ToS=Destination station,
    // Dis=Distance, BR1=Bus line, Ty=Transit Type)
    int FromS,ToS,Dis,BR1;
    string Ty;
    double MasterFile[n][6];
    ifstream Read("BusStopDistance.txt");
    if (Read.is_open()) {
        for(i=0;i<n;i++) {
            Read >> FromS;
            Read >> ToS;
            Read >> Dis;
            Read >> Ty;
            Read >> BR1;
            MasterFile[i][0]=FromS;
            MasterFile[i][1]=ToS;
            MasterFile[i][2]=Dis;
```

```

        if (Ty=="Walk") {
            MasterFile[i][3]=1;MasterFile[i][5]=MasterFile[i][2]/4/1000*3600;
                }
        else {
            MasterFile[i][3]=2;MasterFile[i][5]=MasterFile[i][2]/30/1000*3600;
                }
        if (MasterFile[i][5]==0) MasterFile[i][5]=1;
        MasterFile[i][4]=BR1;
            }
        }

    Read.close();
// End Read From Text File
// Detect Number Of BusStop (BN=Number of bus stops)
    short BusStop[n];
    bool Flag;
    for(i=0;i<n;i++) BusStop[i]=0;
    int BN=0;
    for (i=0;i<n;i++){
        Flag=true;
        for (j=0;j<n;j++){
            if (MasterFile[i][0]==BusStop[j]) { Flag=false; break; }
                }
        if (Flag) { BusStop[BN]=MasterFile[i][0]; BN=BN+1; }
            }
    for (i=0;i<n;i++){
        Flag=true;
        for (j=0;j<n;j++){
            if (MasterFile[i][1]==BusStop[j]) { Flag=false; break; }
                }
        if (Flag) { BusStop[BN]=MasterFile[i][1]; BN=BN+1; }
            }

```

```

// Arrange BusStop Array
int R;
for(i=0;i<BN-1;i++)
for(j=i;j<BN;j++)
    if(BusStop[i]>BusStop[j]){
        R=BusStop[i];
        BusStop[i]=BusStop[j];
        BusStop[j]=R;
    }

// End Arrangement
// End Detect Numbr Of BusStop
// Create BusStop Distance
short BusStopDis[BN][BN];
for(i=0;i<BN;i++)
    for(j=0;j<BN;j++) BusStopDis[i][j]=29999;
int Pi=0,Pj=0;
for(i=0;i<n;i++)
    {
        for(j=0;j<BN;j++)
            {
                BusStopDis[j][j]=20000;
                if (MasterFile[i][0]==BusStop[j]) Pi=j;
                if (MasterFile[i][1]==BusStop[j]) Pj=j ;
            }
        BusStopDis[Pi][Pj]=MasterFile[i][2];
    }
// End BusStop Distance

```

```

// Create Path Matrix
// Create Distance Array
    short WE[BN][BN],O[BN];
    int SA[5][2],FA[5][2];
    int St,DS,Ft,DF;
// start & final point
ifstream StartPoints("NearestStationsStart.txt");
    if (StartPoints.is_open()) { for(i=0;i<5;i++) { StartPoints >> St;
                                                StartPoints >> DS;
                                                SA[i][0]=St;
                                                SA[i][1]=DS;
                                                }
                                }

    StartPoints.close();
ifstream FinalPoints("NearestStationsFinal.txt");
    if (FinalPoints.is_open()) { for(i=0;i<5;i++) { FinalPoints >> Ft;
                                                FinalPoints >> DF;
                                                FA[i][0]=Ft;
                                                FA[i][1]=DF;
                                                }
                                }

    FinalPoints.close();
cout<<" Start Points          Final Points\n\n";
for(i=0;i<5;i++) cout<<" "<<SA[i][0]<<" ("<<SA[i][1]<<" )
                    "<<FA[i][0]<<" ("<<FA[i][1]<<" )\n";
// final start & final point
int Mi,Mj,ExportArray[10][10];
for(Mi=0;Mi<5;Mi++)
for(Mj=0;Mj<5;Mj++)
{
    int s,m,f;

```

```

    s=SA[Mi][0];
// first row for b
    for(i=0;i<BN;i++) WE[0][i]=BusStopDis[s-1][i];
// other rows for b
    O[0]=s;
    for (i=1;i<BN;i++)
    {
        m=WE[i-1][0];
        f=0;
        for(j=1;j<BN;j++)
            if (WE[i-1][j]<m) { f=j;m=WE[i-1][j]; }
        O[i]=f+1;
        for (int t=0;t<BN;t++)
        {
            if(WE[i-1][t]==20000) { WE[i][t]=20000;continue;}
            if((WE[i-1][t]==29999)&&(BusStopDis[f][t]==29999))
                { WE[i][t]=BusStopDis[f][t];continue;}
            if(WE[i-1][t]==29999) { WE[i][t]=BusStopDis[f][t]+m;continue;}
            if(f==t)          { WE[i][t]=20000;continue;}
            if((m+BusStopDis[f][t])>WE[i-1][t])
                { WE[i][t]=WE[i-1][t];continue;}
            else WE[i][t]=m+BusStopDis[f][t];
        }
    }
// End Path Matrix
// find The Path
    int SP,min,d,sta[BN]; // sta array for save the path
    d=FA[Mj][0];
    for(i=0;i<BN;i++) sta[i]=9999;
    SP=WE[0][d-1];
    i=BN-1;

```

```

if (d==s) {SP=0;sta[i]=d;goto FindResult;}
if ((d<s)&&(d>BN)) {goto FindResult;}
for (j=1;j<BN;j++)
if((WE[j][d-1]<SP)&&(WE[j][d-1]!=0)) { SP=WE[j][d-1]; }
sta[i]=d;

```

```
Repeat: min=WE[0][d-1];
```

```
f=0;
```

```
for (j=1;j<BN;j++)
```

```
    if((WE[j][d-1]<min)&&(WE[j][d-1]!=0)) {
```

```
        min=WE[j][d-1];
```

```
        f=j;
```

```
    }
```

```
i=i-1;
```

```
d=O[f];
```

```
if(d==s) goto FindResult;
```

```
else {
```

```
    sta[i]=O[f];
```

```
    goto Repeat;
```

```
}
```

```
FindResult:
```

```
sta[i]=d;
```

```
cout<<"\n\n"<<" Shortest path is: "<<SP;
```

```
cout<<"\n\n"<<" Path is:----- ";
```

```
for (i=0;i<BN;i++){
```

```
    if(sta[i]==9999) continue;
```

```
    cout <<sta[i]<<" ";
```

```
}
```

```
cout <<"\n";
```

```
// End Finding Path
```



```

ExportArray[Mi][Mj]=SP;
//Export Text File
ofstream Write("FinalResult.txt",ios::app);
if(Write.is_open()){
    Write<<SA[Mi][0]<<"\t";
    Write<<FA[Mj][0]<<"\t";
    Write<<ExportArray[Mi][Mj]+SA[Mi][1]+FA[Mi][1]<<"\t";
    for (int jj=0;jj<BN;jj++){
        if(sta[jj]==9999) continue;
        Write<<sta[jj]<<" ";
        }
    Write<<"\n";
}
Write.close();
//End Exportation
}
return EXIT_SUCCESS;
}

```

APPENDIX C

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: AL-TAMEEMI Haitham, Latif Hassan

Nationality: Iraqi (IQ)

Date and Place of Birth: 11 October 1971, Iraq

Marital Status: Married

Phone: +90 538 010 24 08

Email: hai_mah2003@yahoo.com

EDUCATION

Degree	Institution	Year of Graduation
B.Sc.	Department of Computer Science, Al-Mustansiriya University	1993
High School	Baghdad College	1989

WORK EXPERIENCE

Year	Place	Position
1995-2001	Libyan Arab Jamahiriya	Teacher
2005-until now	Ministry of Education, Iraq	Teacher

FOREIGN LANGUAGES: Arabic, English

HOBBIES: Computer hardware, Football, Music