INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Shortest Route Optimization for Emergency Service: Case Study in Kumasi, Ghana

Edward Obeng Amoako
Lecturer, Department of Mathematics,
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

Abstract:
It is becoming difficult for emergence services to find the best route especially in Kumasi to any destination in order to save lives in real time. This study deals with the problem of finding shortest paths in traversing some locations within the Kumasi Metropolis in the Ashanti Region of Ghana. Dijkstra's Algorithm was selected to determine the shortest distances from any location to any destination within the Kumasi metropolis. The objective of thesis is to use Dijkstra's algorithm in constructing the minimum spanning tree considering the dual carriage ways in the road network of Kumasi metropolis within the shortest possible time for emergence services. The distance between 51 locations of the towns with the major roads was measured and a legend and a matrix were formulated. A visual basic program was prepared using Dijkstra's algorithm. The distances were used to prepare an input deck for the visual program. The methodology employed included review of relevant literature of the types of Dijkstra's algorithm and methods employed in the solution of the Dijkstra's algorithm and to develop computer solutions – ArcGIS and VB.net for faster computation of Dijkstra's algorithm. The result shows a remarkable reduction in the actual distance as compared with the ordinary routing. These results indicate, clearly the importance of this type of algorithms in the optimization of network flows. Hence the shortest distance from any area in Kumasi metropolis to another can easily be calculated using this thesis so as to minimize the average loss of lives in case emergences.

Keyword: *Dijkstra's Algorithm, Floyd's algorithm, shortest path, geographic information system, transportation*

1. Introduction

Travelling is a part of daily life. The majority of people (especially in large cities or developing countries) rely heavily on emergences services in the case of accident such as road accident, fire and any disaster event people will rely on these emergence services instead of their own vehicles. In a metropolis with a complicated transport network, people often do not know how to reach their destination except where they often visit. In addition, people may want to plan for the fastest or the most economical method to their destinations. Such tasks require a sophisticated knowledge about public transport network. Further, we need a multi-modal route finding system, because a transport network comprises many modes of transportation, including railway, bus, mini-bus, and so on, within a large metropolis such as Kumasi. When a user asks for a path from one place to another, the system can generate routes, in multi-modal or single modal mode, according to input criteria, such as cost, time, or transportation mode.

Transportation model is but one of the many problems that can be represented and solved as a network problem. To be specific consider the following situations:

(i). Determination of the minimum- cost flow schedule from oil fields to refineries and finally to distribution centre this can be transported through tracks, trains etc.

(ii). Determination of the shortest route joining two cities in an existing network of roads.

(iii). Collection, transporting and dumping of garbage.

(iv). Business, scheduling deliveries and installations while including time window restrictions, or a calculating drive time to determine customer base, taking into account rush hour versus midday traffic volumes.

(v). Education, generating school bus routes honouring curb approach and no U-turn rules. (vi). Environmental Health, determining effective routes for county health inspectors.

(vii).Public Safety, routing emergency response crews to incidents, or calculating drive time for first responder planning.

(viii).Public Works, determining the optimal route for point-to-point pickups of massive trash items or routing of repair crews.

(ix).Retail, finding the closest store based on a customer's location including the ability to return the closest ranked by distance.

(x).Transportation, calculating accessibility for mass transit systems by using a complex network data set.

A study of this representative list reviews that;

Situation a) is a minimal spanning tree model

Situation b) is a shortest route model

Situation c) is a minimum-cost capacitated network model

The examples cited above deal with the determination of distances and material flow in a literal sense, the network models listed can be represented, and in principle, solve as linear programs. However the tremendous number of variables and constraints that normally accompanies a typical network model makes it inadvisable to solve network problems directly by the simplex method. The nature and/or structure of these problems allow the development of highly efficient algorithms, which in most cases are based on linear programming theory.

1.1. Objectives and Material of the Study

The objective of this thesis is to create a formation movement shortest path finding algorithm for emergences services vehicles to implement tactical movement within a large metropolis such as Kumasi and optimization scheme for transportation planning and analysis to provide a major advantage in its ability to take into account a range of different, often unrelated criteria, even if these criteria cannot be directly related to quantitative outcome measures. Generally, methodology consists of the study major routes, sampling procedure, sample size and how the data is analyzed. Kumasi Metropolis, however, has been selected as the reference region The methodology employed included review of relevant literature of the types of Dijkstra's algorithm and methods employed in the solution of the Dijkstra's algorithm and to develop computer solutions – ArcGIS and VB.net for faster computation of Dijkstra's algorithm. With the development of geographic information systems (GIS) technology, network and transportation analyses within a GIS environment have become a common practice in many application areas. A key problem in network and transportation analyses is the computation of shortest paths between different locations on a network. Sometimes this computation has to be done in real time. For the sake of illustration, let us have a look at the case of a 911 call requesting an ambulance to rush a patient to a hospital. Today it is possible to determine the fastest route and dispatch an ambulance with the assistance of GIS. Because a link on a real road network in a city tends to possess different levels of congestion during different time periods of a day, and because a patient's location cannot be expected to be known in advance, it is practically impossible to determine the fastest route before a 191 call is received. Hence, the fastest route can only be determined in real time. In some cases the fastest route has to be determined in a few seconds in order to ensure the safety of a patient. Moreover, when large real road networks are involved in an application, the determination of shortest paths on a large network can be computationally very intensive. Because many applications involve real road networks and because the computation of a fastest route (shortest path) requires an answer in real time, a natural question to ask is: Which shortest path algorithm runs fastest on real road networks? Although considerable empirical studies on the performance of shortest path algorithms have been reported in the literature (Dijkstra 1959; Dial et al,. 1979; Glover et al., 1985; Gallo and Pallottino 1988; Hung and Divoky 1988; Ahuja et al., 1990; Mondou et al.,1991; Cherkassky et al., 1993; Goldberg and Radzik 1993), there is no clear answer as to which algorithm, or a set of algorithms runs fastest on real road networks. In a recent study conducted by Zhan and Noon (1996), a set of three shortest path algorithms that run fastest on real road networks has been identified. These three algorithms are:

a. the graph growth algorithm implemented with two queues,
b. the Dijkstra's algorithm implemented with approximate buckets, and
c. the Dijkstra's algorithm implemented with double buckets.

Dijkstra's algorithm was then used on the node graph, but modified to run faster using this extra data. However this optimization changes Dijkstra's Algorithm so that it only finds a path, rather than the shortest path. Other applications of Dijkstra are

a. Routing of postal workers
b. Routing robots through a warehouse
c. Drilling holes on printed circuit board

A network consists of a set of points and a set of lines connecting certain pair of the points.

These points are called nodes and are linked by arcs, edges or branches. Associated with each arc is the flow of some type. In a transportation network, cities represent nodes and highways represent edges or arc, with traffic representing arc flow. The standard notation for describing network G= (N, A) where N is the set of nodes and A is the set of edges or arcs.

Today it is possible to determine the faster route and dispatch the immediate assistance or with the help of the assistance of Geological Information System {G.I.S). With the advance development in technology, the analyses of networking and transportation within the technological environment have become a common practice in many applicable areas. The key problem in network and transportation is the computation of the SHORTEST PATHS between different locations on a network Sometimes this computation has to be done in real times. For the sake of illustration, let us have a look at the case of an emergency call, requesting an ambulance to rush a patient from a very remote area to a hospital. Because a link on a real road network in the city tends to posse different levels of congestion during different time period of a day and because a Patient's location cannot be expected to be known in advance, it is practically impossible to determine the fastest route before a call is received. Hence the fastest route can only be determined in real time.

In some cases the fastest route has to be determined in a few second in order to ensure the safety of a patient. Moreover when large real road network are involved in an application, the determination of SHORTEST PATHS on a large network can be computationally very intensively. The collection, transport and disposal of solid waste, which is a highly visible and important municipal service, involves a large expenditure but receives, scant attention. This problem is even more crucial for large cities in developing countries due to the hot weather. A constructive heuristic, which takes into account the environmental aspect as well as the cost, is proposed to solve the routing aspect of garbage collection. This is based on a look-ahead strategy, which is enhanced by this additional mechanism:

The problem and its impact on the environment collection of household refuse/industrial waste is one of the most difficult operational problems faced by local authorities in any large city. The collection problem is especially crucial for cities in developing countries. Solid wastes generated from urban and industrial sources also contain a large number of ingredients, some of which are toxic.

2. Methodology

The methodology employed included review of relevant literature of the types of Dijkstra algorithm and methods employed in the solution of the Dijkstra algorithm and to develop computer solutions – ArcGIS and VB.net for faster computation of Dijkstra algorithm

2.1. Background of Graph Theory

In this chapter, some fundamental concepts of graph theory are introduced and will be referred to in subsequent discussions.

2.2. Definition of a Graph

In mathematics and computer science, graph theory deals with the properties of graphs. Informally, a graph is a set of objects, known as nodes or vertices, connected by links, known as edges or arcs, which can be undirected or directed (assigned a direction). It is often depicted as a set of points (nodes, vertices) joined by links (the edges). Precisely, a graph is a pair, G = (V; E), of sets satisfying E∈ [V]; thus, the elements of E are 2-element subsets of V. The elements of V are the nodes (or vertices) of the graph G, the elements of E are its links (or edges). In this case, E is a subset of the cross product V *V which is denoted by E ∈ [V]. To avoid notational ambiguities, we shall always assume that V ∩E =∅.

A connected graph is a non-empty graph G with paths from all nodes to all other nodes in the graph. The order of a graph G is determined by the number of nodes. Graphs are finite or infinite according to their order. In this paper, the graphs are all finite and connected. Furthermore, a graph having a weight, or number, associated with each link is called a weighted graph, denoted by G = (V; E; W).

2.3. Degree of a Vertex (Node)

A node *v* is incident with a link *e* if *v* ∈*e*; then *e* is a link at *v*. The two nodes incidents with a link are its end nodes. The set of neighbours of a node *v* in *G* is denoted by *N* (*v*).The degree *d* (*v*) of a node *v* is the number |*E* (*v*)| of links incident on *v*. This is equal to the number of neighbours of *v*. A node of degree 0 is isolated. The number δ (*G*) = min {*d* (*v*) | *v*∈*V*} is the minimum degree of *G*, while the number Δ (*G*) = max {*d* (*v*) | *v*∈*V*} is the maximum degree. The average degree of *G* is given by the number

$$d(G) = \frac{1}{|V|}\sum_{v\in V} d(V)$$

Clearly,

$$\delta(G) \leq d(G) \leq \Delta(G)$$

The average degree globally quantifies what is measured locally by the node degrees: the number of links of *G* per node. Sometimes it is convenient to express this ratio directly, as ε (*G*) = |*E*|/|*V*|. The quantities *d* and ε are intimately related. Indeed, if we sum up all of the node degrees in *G*, we count every link exactly twice: once from each of its ends.

Thus,

$$|E| = \frac{1}{2}\sum_{v\in V} d(v) = \frac{1}{2} d(G).|V|$$

and therefore

$$\in (G) = \tfrac{1}{2} d(G)$$

Graphs with a number of links that are roughly quadratic in their order are usually called dense graphs. Graphs with a number of links that are approximately linear in their order are called sparse graphs. Obviously, the average degree *d* (*G*) for a dense graph will be much greater than that of a sparse graph.

In a graph, a path, from a source node *s* to a destination node *d*, is defined as a sequence of nodes (v0, v1, v2, ..., vk) where s = v0, d = vk, and the links (v0, v1), (v1, v2), ..., (vk−1, vk) are present in E. The cardinality of a path is determined by the number of links. The cost of a path is the sum of the link costs that make up the path.

An optimal path from node *u* to node *v* is the path with minimum cost, denoted by *(u, v)*. The cost can take many forms including travel time, travel distance, or total toll. In my research, the cost or weight of a path stands for the travel time which is needed to go through the path.

2.4. Transportation Network Data Model

A transportation network is a type of directed, weighted graph. The use of GIS for transportation applications is widespread and a fundamental requirement for most transportation GIS is a structured road network. In developing a transportation network model, the street system is represented by a series of nodes and links with associated weights. This representation is an attempt to quantify the street system for use in a mathematical model. Inherent in the modelling effort is a simplification of the actual street system. The network nodes represent the intersections within the street system and the network links represent the streets. The weights represent travel time between the nodes.

As a specialized type of graph, a transportation network has characteristics that differ from the general graph. A suitable data structure is required to represent the transportation network. Comparing the three data structures, an adjacency list representation of the graph occupies less space because it does not require space to represent links which are not present. The space complexity of an adjacency list is $O(|E|+|V|)$, where $|E|$ and $|V|$ are the number of links and nodes respectively. In contrast, incidence matrix and adjacency matrix representations contain too many 0s which are useless and redundant in storage. The space complexity of incidence matrices and adjacency matrices are $O(|E| \times |V|)$ and $O(|V|2)$ respectively. In the following discussion, we shall take a more detailed look at the three data models in terms of storage space and suitable operations. Using a naive linked list implementation on a 32-bit computer, an adjacency list for an undirected graph requires approximately $16 \times (|E|+|V|)$ bytes of storage space. On the other hand, because each entry in the adjacency matrix requires only one bit, they can be represented in a very compact way, occupying only $|V|2/8$ bytes of contiguous space. First, we assume that the adjacency list occupies more memory space than that of an adjacency matrix.

Then

$$16X(|E| + |V|) \geq \frac{|v|^2}{8}$$

Based on equation (3.1.2) in section 3.1, we have,

$$16X(\frac{1}{2}\, d(G)X\, (|V| + |V|) \geq \frac{|V|^2}{8}$$

where $d(G)$ is the average degree of G.

$$d(G) \geq \frac{|V| - 128}{64}$$

This means that the adjacency list representation occupies more space when equation (3.5) holds. In reality, most transportation networks are large scale sparse graphs with many nodes but relatively few links as compared with the maximum number possible ($|V| \times (|V| -1)$ for maximum).That That is, there are no more than 5 links ($\Delta(G) \approx 5$) connected to each node. In most cases there are usually 2, 3 or 4 ($\delta(G) = 2$) links, although the maximum links is $|V|$-1 for each node. Also, road networks often have regular network structures and a normal layout, especially for well-planned modern cities. Again most transportation networks are near connected graphs, in which any pair of points is traversable through a route. Assuming the average degree of a road network is 5, equation 3.5 holds only if $|V| \leq 448$.

However, most road networks contains thousands of nodes where $|V| >> 448$. As a result, equation 3.2 cannot hold. Thus, the adjacency list representation occupies less storage space than that of an adjacency matrix. For example, consider a road network containing 10000 nodes. If an adjacency matrix is employed to store the network, at least 10 megabytes of memory space is required. It will most likely take more computational power and time to manipulate such a large array, and then it is impossible to conduct routing searches in some mobile data terminals, such as smart phones and Personal Digital Assistance (PDAs).The comparison between the adjacency matrix and incidence matrix can give the same result. Assuming an adjacency matrix occupies more storage space than that of an incidence matrix,

then

$|V|^2 \geq |E| \times |V|$

From equation 3.2, we obtain, $d(G) \leq 2$ (3.6)

This means that the adjacency matrix representation occupies more space if and only if equation 3.6 holds. Since the minimum degree of transportation network is 2 ($\delta(G) = 2$), then equation 3.6 is invalid. As a result, the adjacency matrix occupies less storage space than that of the incidence matrix. Since the adjacency matrix cannot compete with the adjacency list in terms of storage space (i.e., requires more space), it follows that the incidence matrix will also not be able to compete.

Other than the space trade off, the different data structures also facilitate different operations. It is easy to find all nodes adjacent to a given node in an adjacency list representation by simply reading its adjacency list. With an adjacency matrix, we must scan over an entire row, taking $O(|V|)$ time, since all $|V|$ entries in row v of the matrix must be examined in order to see which links exist. This is inefficient for sparse graphs since the number of outgoing links j may be much less than $|V|$. Although the adjacency matrix is inefficient for sparse graphs, it does have an advantage when checking for the existence of a link $u \rightarrow v$, since this can be completed in $O(1)$ time by simply looking up the array entry $[u; v]$. In contrast, the same operation using an adjacency list data structure requires $O(j)$ time since each of the j links in the node list for u must be examined to see if the target is node v. However, the main operation in a route search is to find the successors of a given node and the main concern is to determine all of its adjacent nodes. The adjacency list is more feasible for this operation. The above discussions demonstrate that the adjacency list is most suitable for representing a transportation network since it not only reduces the storage space in the main memory, but it also facilitates the routing computation.

Since transportation networks are a specialized type of graph, some fundamental knowledge of graph theory is required. Some basic concepts, such as the definition of a graph, degree of a graph, and the definition of a path, were introduced at the beginning of this chapter. In the discussion of the degree of a graph, the dense graph and sparse graph have been defined and used in data model discussion. In the data model discussion, three types of data models for graph representation were given: the incidence matrix, adjacency matrix and adjacency list.

The discussion includes a description of each model, an analysis of the space complexity, storage space requirements and an examination of suitable operations for each model. Based on the discussion, an adjacency list is regarded as the best representation of the transportation network considering its own characteristics. My research, will utilize an adjacency list to construct topology of the experimental road network in order to implement my routing computations.

3. Typical Routing Queries

There are various types of routing queries that may be submitted to the centralized GIS server. To answer the queries, many algorithms have been developed to satisfy the conditions and requirements of these queries. The research for generalizing this document is focused on two typical routing queries. The first query deals with finding the optimal route from the current location to a known destination. The other query allows users to locate the closest facility of a certain category (hotel, hospital, gas station, etc.), in terms of travel distance (time), without knowing the destination explicitly.

3.1. Routing Query for Known Destination

For this query, the mobile client or driver has a definite destination in mind and desires to acquire the optimal route leading to the destination. Since the traffic condition changes continually over time, the optimal route will change during travel whenever up-to-date traffic conditions are provided. For example, when we want to drive from the airport to the KMA office, we can plan the entire optimal route prior to departure according to the current condition of the transportation network. However, it may not be the final optimal route due to frequent changes in the traffic conditions. So, we have to modify our route midway and plan a new path from the current location to the destination based on real-time traffic conditions. This case is more complicated than the conventional dynamic concept because both the traffic conditions and the query point (location of the driver) are dynamic. This type of query is also defined as an en route query since it is submitted while the client is moving.

3.2. Routing Query for Unknown Destination

For this query, drivers may inquire about the location of the closest facility, such as the nearest hotel, hospital or gas station, without knowing the destination in advance. In this case, the closest facility is defined in terms of travel distance (time) within the road network as opposed to travel distance. This query can be classified as the Nearest Neighbour problem.

Both the closest destination and an associated optimal route need to be found based on travel time within the road network. Similarly, the optimal route also has to be recalculated whenever up-to-date traffic conditions are provided. In extreme circumstances, the closest destination may also change. For example, in an unknown city, we may want to find the location of the closest post office after we check into a hotel. From the query result, we are aware of the position and optimal route to the closest post office. In this case, we expect the navigation service not only to provide the adaptive route leading to it, but also to confirm the validity of the closest post office while traveling. If the traffic conditions do not change significantly, the optimal route may only need to be slightly modified. If the traffic conditions change considerably or there are serious traffic congestions around the anticipated post office destination, this post office may no longer be the closest one in terms of traveling time. A new post office location and optimal route must then be determined dynamically based on the current location and traffic conditions. In this scenario, the query is an en route query. To solve this problem, a dynamic nearest neighbour and route searching algorithm is required.

3.3. Introduction to the Shortest Path Algorithms

The shortest path (SP) algorithms are among fundamental network analysis problems. Since 1957 a considerable progress has been made in the SP algorithms after Minty published his paper (1957). Minty succinctly described the basic SP problem for symmetrical networks (a network is symmetrical if for every pair of nodes the cost of a link between the two nodes is independent of their starting node). To state the problem beyond doubt, he suggested constructing a model of the given network. The model is made of strings, each string of the length proportional to the costs of the modelled link.

Finally, to find the links of the SP one has to pull the source node and the destination node of the journey as far away as possible. The tight strings are the links of the SP. Since 1957 there has been a number of major papers published, the most important were published by Bellman (1959), Dijkstra (1959) and Moore (1959). These articles were formative and most of the traffic research has used their results (for example Clercq (1972) and Cooke and Halsey (1966)). These articles are now included in references by most other publications.

There are a number of review papers. One of the utmost importance has been published by Dreyfus (1969). The review gives a comprehensive summary of the research, which has been carried out up to 1969. The article surveys over ten years of research, discussing the most crucial stages and pointing out the wrong and inefficient solutions. The paper also gives a brief solution of the SP problem for time varying costs of links, which is the basis of this report. The shortest path algorithms are currently widely used. They are the basis of the network flow problems, tree problems and many related other problems. They determine the smallest cost of travel, of a production cycle, the shortest path in an electric circuit or the most reliable path. In the book by Ahuja ,(1993) one can realize that the SP problem is an underlying problem of the network optimization and that it is closely related to network flows or tree building issues. Internet is a large field where the shortest path algorithms can be applied. The Internet problems involve data packages transmission with the minimal time or by the most reliable path. An example of the SP algorithms in the Internet is given by Cai, (1997). This paper proposes three SP algorithms. The devised algorithms are well explained. The article is closely related to the problem. The same algorithms can be used without fundamental changes to the urban traffic issues. The use of the proposed algorithms for public transportation networks will be studied in the section _Shortest path and the environment issues‘. Algorithms to be discussed here have a thirty-year old history and solutions to the fundamental problems are well known. The contemporary research is directed toward parallel computing as the method for further lowering of the time complexity bound of the shortest path algorithms. The report is not interested in the parallel approach. The article by Klein and Subramanian (1997) is an example of the shortest path parallel algorithm.

4. Some Network Definitions and Related Work

4.1. Types of Networks

There are several types of networks of special interest to the project: sparse, planar and road networks. Other types of networks (as grid or dense) are not taken into account.

4.1.1. Sparse Networks

Sparse networks are those which have the number of links only a few times bigger than the number of nodes. A network of one hundred (100) nodes and four hundred (400) links would be considered sparse but a network with one hundred (100) nodes and five thousand (5000) links would be classified as dense.

Public transportation networks are sparse. From node approximately four links leave. If a sparse network was presented in a matrix form, then in each row of the matrix about only four places would be used, the rest would be left idle. For matrix network representation there are algorithms, which handle efficiently the sparse networks. However, it is recommended not to use matrix-based algorithms since the matrix representation of a sparse network is highly inefficient. Instead of the matrix algorithms the tree building algorithms can be used as they store the sparse network information in an efficient way (usually using lists). In this thesis, the Dijkstra algorithm which is a basic tree building algorithm has been used. The matrix in figure 1 is an example of the inefficient matrix representation of a sparse network since there are more places unused than used.

4.1.2. Planar Networks

There are a number of SP algorithms for planar graphs. Methods characteristic to planar graphs (as separators) lower the computational bound of the SP algorithms. Since the road network and transport network are mostly planar, application of the algorithms from this group could bring more efficient solutions to our problem. However, not all road networks are planar, there are viaducts and bridges, which can destroy planarity and thus unable, limit or complicate the application of these algorithms. For this reason the methods for the planar graphs will not be considered. The article by Monika R. Henzinger etal., (1997) proposes three new algorithms for planar graphs.

4.1.3. Road Networks

In the representation of a road network a link represents a road and a node represents a crossroad. The ratio of the number of links to the number of nodes is approximately 3. (Steenbrink, year 1958), gives an example of a road network with about 2000 nodes and 6000 links). The link costs are always non-negative. The road networks are usually planar and sparse. The number of nodes is big, usually expressed in thousands. Road networks contain loops, which are allowed since they may be only of a non-negative cost (the link costs are only non-negative). The road networks are of a special interest in this thesis. The characteristic feature of the road networks is their nonnegative link lengths property. Dijkstra year made a good use of nonnegative lengths to design his algorithm. Because of this close relation between Dijkstra, algorithm and road characteristic, it should not be surprising that this report suffers constant _Dijkstra' referring. The project's road network of the Kumasi City had about seven hundred and eighteen (718) nodes and one thousand one hundred and eighty – seven (1187) links. The network represents the link connections between nodes and the distances between them.

4.2. All – Pairs Shortest Path Problem

The shortest path between two nodes might not be a direct edge between them, but instead involve a detour through other nodes. The all- pairs shortest path problem requires that we determine shortest path distances between every pair of nodes in a network.

Shortest path problems are the most fundamental and the most commonly encountered problem in the study of transportation and communication networks (syslo et al., 1983). There are many types of shortest-path problem. For example, we may be interested in determining the shortest path (i.e., the most economical path or fastest path, or minimum fuel-consumption path) from one specified node in the network to another specified node; or we may need to find shortest paths from a specified node to all other nodes. Shortest paths between all pairs of nodes in a network are required in some problems. Sometimes, one wishes to find the shortest path from one given node to another given node that passes through certain specified intermediate nodes.

In some application, one requires not only the shortest path but also the second and third shortest path. There are instances when the actual shortest path is not required, but only the shortest distance is required.

Next, we shall confine ourselves to two most important shortest-path problems; How to determine shortest distance (a short path) from a specified node to another specified node *t*, and ~how to determine shortest distances (all paths) from every node to every other in the network. Shortest path route problem deals with determining the connected arcs in a transportation network that collectively comprise the shortest distance. Between a source and a destination. The shortest path problem involves a weighted, possibly directed graph described by the set of edges and vertices shortest path deals with two algorithms for finding the shortest route.

5. Shortest Path Problems

The computation of shortest paths has been extensively researched since it is a fundamental issue in the analysis of transportation networks. There are many factors associated with shortest path algorithms. First, there is the type of graph on which an algorithm works - directed or undirected, real-valued or integer link costs, and possibly negative or nonnegative link-costs. Furthermore, there is

the family of graphs on which an algorithm works - acyclic, planar, and connected. All of the shortest path algorithms presented in this thesis assume directed graphs with non-negative real-valued link costs.

5.1. Dijkstra's Algorithm

Dijkstra's algorithm, named after its inventor, has been influential in path computation research. It works by visiting nodes in the network starting with the object's start node and then iteratively examining the closest not-yet-examined node. It adds its successors to the set of nodes to be examined and thus divides the graph into two sets: *S*, the nodes whose shortest path to the start node is known and *S'*, the nodes whose shortest path to the start node is unknown. Initially, *S'* contains all of the nodes. Nodes are then moved from *S'* to *S* after examination and thus the node set, S, ―grows‖. At each step of the algorithm, the next node added to *S* is determined by a priority queue. The queue contains the nodes *S'*, prioritized by their distance label, which is the cost of the current shortest path to the start node. This distance is also known as the start distance. The node, *u*, at the top of the priority queue is then examined, added to *S*, and its out- links are relaxed. If the distance label of *u* plus the cost of the out- link *(u, v)* is less than the distance label for *v*, the estimated distance for node *v* is updated with this value. The algorithm then loops back and processes the next node at the top of the priority queue. The algorithm terminates when the goal is reached or the priority queue is empty. Dijkstra's algorithm can solve single source SP problems by computing the one-to-all shortest path trees from a source node to all other nodes.

The pseudo-code of Dijkstra's algorithm is described below.

Function Dijkstra (*G, start*)

1) d [*start*] = 0
2) $S = \emptyset$
3) $S' = V \in G$
4) while $S' \neq \emptyset$
5) do u = Min (S')
6) $S = S \cup \{u\}$
7) for each link (u, v) outgoing from u
8) do if $d[v] > d[u] + w(u, v)$ // Relax (u, v)
9) then $d[v] = d[u] + w(u, v)$
10) Previous[v] = u

5.2. A Algorithm*

It is not feasible to use Dijkstra's algorithm to compute the shortest path from a single start node to a single destination since this algorithm does not apply any heuristics. It searches by expanding out equally in every direction and exploring a too large and unnecessary search area before the goal is found. Dijkstra's algorithm is a version of a BFS and although this algorithm is guaranteed to find the optimal path., it is not extensively applied due to its relatively high computing cost. This has led to the development of heuristic searches. In terms of heuristic searches, the A* algorithm is widely regarded as the most efficient method. The A* algorithm is a heuristic variant of Dijkstra's algorithm, which applies the principle of artificial intelligence. Like Dijkstra's algorithm, the search space is divided into two sets: *S*, the nodes whose shortest path to the start node is known and *S'*, the nodes whose shortest path to the start node is unknown. It differs from Dijkstra's algorithm in that it does not only consider the distance between the examined node and the start node, but it also considers the distance between the examined node and the goal node.

In the A* algorithm, g (n) is called the start distance, which represents the cost of the path from the start node to any node n, and h(n) is estimated as the goal distance, which represents the heuristic estimated cost from node n to the goal. Because the path is not yet complete, we cannot actually know this value, and h (n) has to be ―guessed‖. This is where the heuristic method is applied. In general, a search algorithm is called admissible if it is guaranteed to always find the shortest path from a start node to a goal node. If the heuristic employed by the A* algorithm never overestimates the cost, or distance, to the goal, it can be shown that the A* algorithm is admissible. The heuristic is called an admissible heuristic since it makes the A* search admissible. If the heuristic estimate is given as zero, this algorithm will perform the same as Dijkstra's algorithm. Although it is often impractical to compute, the best possible heuristic is the actual minimal distance to the goal. An example of a practical admissible heuristic is the straight-line distance from the examined node to the goal in order to estimate how close it is to the goal. The A* algorithm estimates two distances g(n) and h(n) in the search, ranks each node with the equation: f(n) = g(n) + h(n), and always expands the node n that has the lowest f(n). Therefore, A* avoids considering directions with non-favourable results and the search direction can efficiently lead to the goal. In this way, the computation time is reduced. Thus, the A* algorithm is faster than Dijkstra's algorithm for finding the shortest path between single pair nodes. The algorithm is an example of a best first search

5.3. Shortest Path and the Environment Issues

Suppose there is a need to find a path, which implies the smallest usage of fuel. This case is similar as the money cost of the travel, but it differs since the bus money cost (the money for a bus ticket, for example) is higher than (and not linear to) the fuel used. The shortest path in terms of used fuel needs to be evaluated from the environment point of view. The simplest approach to the problem is to ascribe a cost to every link costs that express the impact on the environment. A higher cost will be attached to a car link, and a smaller cost will be attached to the bus link. The cost of a link should be dependent on the length of s link. According to the criteria of cost, the algorithm searches for the shortest path and at the same time computes the time cost. The time cost of a shortest path generated with the help of such an algorithm will not be optimised. We can conceive the case where there is the shortest path found in

terms of the lowest fuel cost, but the time cost is not acceptable. This may happen is we waited a long time to save not a significant amount of fuel.

A number of constraining criteria for such a shortest route finding can be given. First, we can fix a certain amount of time which can be taken at most for waiting at a bus stop. Among the links that fulfil this condition, the link of the smallest fuel cost is chosen. Another constraint can be that the overall travel time cannot be greater than a fixed amount T. In the article by Cai at el., (1997) one can find three algorithms for the internet data packages routing among, which one algorithm is very well suited to our needs. The algorithm searches a specific type of networks. Each link in the network has two numbers ascribed: cost and time. For us the cost can be the fuel cost and time is the time cost. The proposed algorithm is going to find the shortest route according to fuel cost and with the overall time cost not exceeding a specific amount of time T. The main ideas of the project are involved in the adaptation of the algorithm proposed by Dreyfus (1969) for bus networks which is based on the Dijkstra‘s algorithm. The main ideas have been used to adopt the algorithm described by Dreyfus (1969) to the public transportation networks, to describe it mathematically.

6. Introduction to the Bus Routing Algorithm Description

The algorithm can be used for any public transportation network based on timetables. In any public transports means that the project takes into account, the root for the public transport which must be based on timetables in which the driver reports the bus routing algorithm.

6.1. The Bus Routing Algorithm

To understand the problem clearly, it is useful to visualise a traveller who wants to get from one bus stop to another in a city using buses only. The input data to the algorithm consists of a description of the bus transportation network (timetables, description of connections between bus stops), the bus stop where the journey begins (the source node) and the bus stop at which the journey ends (the destination node). The objective is to find the shortest path between the two specified nodes, namely the path that requires the minimal amount of time. The algorithm presented here was designed to solve the problem described above. The new algorithm had to be designed in order to meet the special needs of bus transportation networks such as timetables and the possibility of waiting at bus stops. The main difference between the standard shortest path problem and this one is that links vary with distance (time) and it is allowed to wait at the nodes as long as it is necessary to obtain the minimal time cost. The problem can be classified as the shortest path problem with time dependent costs of links and the allowance of waiting at nodes. A time cost of every link may differ in any desired way. The solution to the problem is based on Dijkstra‘s algorithm, which is the best known algorithm for directed networks with nonnegative link costs. There are several principles (as the use of a priority queue or the use of buckets) underlying an efficient implementation of the Dijkstra algorithm which can also be applied to implement the bus algorithm (the article by Cherkassy et al is a good paper discussing principles for Dijkstra algorithm implementation).The Dijkstra algorithm has to be modified because of two problems:

(i) The first modification deals with a problem of fixed times at which a bus leaves the bus stop. This new attribute of a link is going to be named the departure time of a link. Dijkstra‘s algorithm is not concerned with a departure time of a link; the algorithm was designed to work with links, which can be used at any time. In our problem the main constraint is that links cannot be used at any time, the time at which a bus link can be used is fixed according to a timetable.

(ii) The second problem is the actual cost of a bus connection between two nodes. In our case the cost of a link is not the criterion to judge the optimality of the link choice anymore. In the present problem the actual criterion is the sum of the waiting time and the link cost, or, in other words, the time of arrival at the finishing node of a link. In effect, we are also concerned with the waiting times at nodes. The need to wait at bus stops is a consequence of the departure time attribute (if a link cannot be used right now then it is necessary to wait for the departure). Suppose there are buses leaving a specific bus stop at 1, 2… 10 time units and arriving at the other specific bus stop after the time cost. The time costs of the buses differ considerably since the buses may be of different companies and they may take different routes. Having the data, the task is to find the cheapest connection between two nodes. The task then is to find the link that has the minimum sum of the time cost of a link and the waiting time (that is necessary to wait for this link). We can phrase the solution to the problem in this way: the sought link is the link of which the arrival time is minimal. This formulation of the solution, i.e. finding the minimal arrival time, is going to be used as opposed to the summation of a waiting time and the time cost (which is the same but complicates the coming formulas).

7. The Shortest Path

Let P = (s = x1, x2, …, x = xr) be the shortest path from the source node s to the destination node *x*. The nodes of the shortest path are such that they result in the minimal arrival time to the *xr* node. The time of arrival is given by *T(xr).* The subsequent nodes of the shortest path are found by the use of *T(xi)* function. To find *xr-1* we have to find the link which led to *xr* with minimal arrival time. Having the link, we have the starting node of the link. This starting node is the *xr-1* node of the shortest path. This method has to be repeated until the source node is reached.

7.1. Step Description of the Algorithm

The aim of the algorithm is to minimise *T(x)* (*x* is the destination node). To minimise it we first have to minimise *T(xi)* for nodes *xi* which are at the shortest path from s to x. Before we get to the algorithm description, there are some definitions to be introduced. We classify all the nodes of the graph into three sets, every node can be a member of only one of the following sets:

i. SPN: the Set of Permanent Nodes is the set of nodes which have been completely processed; the time of reaching these nodes has been computed and will not change. ii. SSN: the Set of Scanned Nodes is the set of nodes which have been reached, but have not been completely processed; the time cost of getting to them is known but may change. iii. SNRN: the Set of Not Reached Nodes is the set of nodes which have not been reached
at all.

7.2. Dijkstra's Algorithm

Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstrain 1959, is a graph search algorithmthat solves the single-source shortest path problem for a graph with non-negative edge path costs, outputting a shortest path tree. This algorithm is often used in routing. For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.
For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

7.3. Algorithm

Let's call the node we are starting with an initial node. Let a distance of a node X be the distance from the initial node to it. Our algorithm will assign some initial distance values and will try to improve them step-by-step:

i. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
ii. Mark all nodes as unvisited. Set initial node as current.
iii. For current node, consider all its unvisited neighbours and calculate their distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6 + 2 = 8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
iv. When we are done considering all neighbours of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
v. Set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.
vi. When all nodes are visited, algorithm ends.

7.4. Description of the Algorithm

Suppose you create a knotted web of strings, with each knot corresponding to a node, and the strings corresponding to the edges of the web: the length of each string is proportional to the weight of each edge. Now you compress the web into a small pile without making any knots or tangles in it. You then grab your starting knot and pull straight up. As new knots start to come up with the original, you can measure the straight up-down distance to these knots: this must be the shortest distance from the starting node to the destination node. The acts of "pulling up" and "measuring" must be abstracted for the computer, but the general idea of the algorithm is the same: you have two sets, one of knots that are on the table, and another of knots that are in the air. Every step of the algorithm, you take the closest knot from the table and pull it into the air, and mark it with its length. If any knots are left on the table when you're done, you mark them with the distance infinity. Or, using a street map, suppose you're marking over the streets (tracing the street with a marker) in a certain order, until you have a route marked in from the starting point to the destination. The order is conceptually simple: from all the street intersections of the already marked routes, find the closest unmarked intersection - closest to the starting point (the "greedy" part). It's the whole marked route to the intersection, plus the street to the new, unmarked intersection. Mark that street to that intersection, draw an arrow with the direction, then repeat. Never mark to any intersection twice. When you get to the destination, follow the arrows backwards. There will be only one path back against the arrows, the shortest one.
The Dijkstra's algorithm uses two types of labels: temporary and permanent. Both labels utilized the same format used with the cycles algorithm: namely, [*d, n*], where d is the shortest distance so far available for a current node, and n is the immediate predecessor node responsible for realizing the distance d. The algorithm starts with the source node carrying the permanent label [0-]. Next we consider all the nodes that can be reached directly from source node and then determine their associated labels. The newly created labels are designated as temporary.
The permanent label is selected from among all current temporary labels as the one having the smallest distance d in the label [*d n*] (ties are broken arbitrarily). The process is now repeated for the last node that has been designated permanent. In such a case, a temporary label of a node may be changed only if the new label yields a smaller distance d.
Let us apply the procedure to the network in figure below. a basic assumption of the algorithm is that all the distances in the network are non-negative.

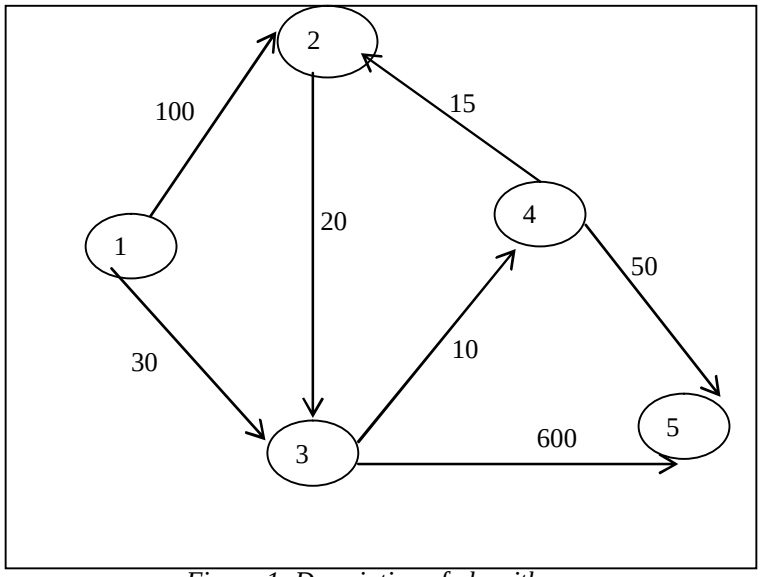

Figure 1: Description of algorithm

Iteration 0: Node 1 carries the permanent label (0)
Iteration 1: Nodes 2 and 3, which can be reached directly from node 1 (the last permanently labelled node), now carry the temporary labels (0 +100, 1) and (0 + 30, 1) or (100, 1) and (30, 1), respectively. Among the current temporary labels, node 3 has the smallest distance d +30 (+ min {100, 30}) thus node 3 is permanently labelled.
Iteration 3: node 4 and 5 can be reached from the last permanently labelled node (node 3) respectively. At this point, we have the three temporary labels [30 + 10, 3] and [30 +60, 3] (or [40, 3] and [90, 3] associated with nodes 2,4, and 5, respectively. Temporarily labelled node 4 has the smallest d = 40 (+ min {100, 40, 90}) and hence its label [40, 3] is converted to the permanently status.
Iteration 3: from the node 4, we now label node 2, with the now temporary label
[40 +15, 4] = [55, 4], which replace the old temporary label [100, 1]. Next, node include [55, 4] and [90, 4] associated with nodes 2 and 5, respectively. We thus label node 2 permanently with [55, 4]
The only remaining node is the sink node 5, which converts its [90, 4] into a permanent label, thus completing the procedure.

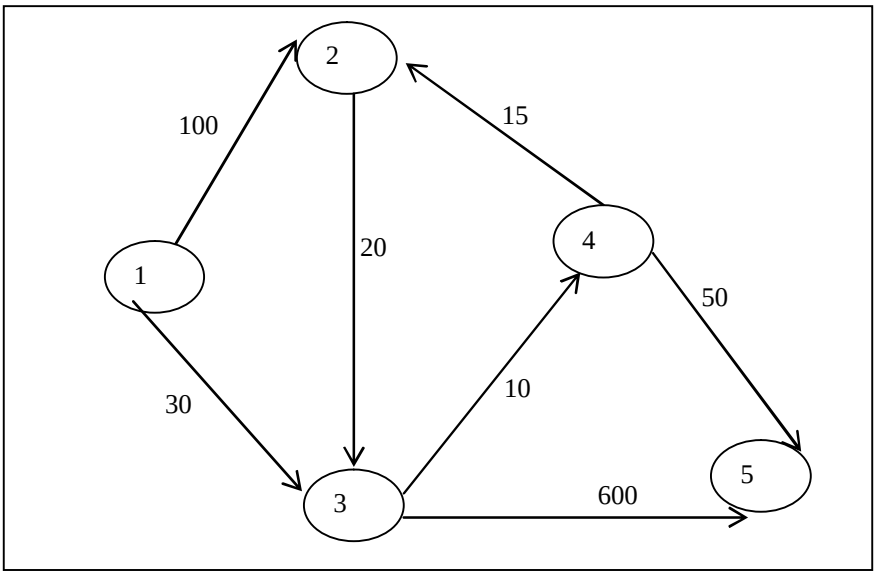

Figure 2: Description of the algorithm

Therefore, one would realize that the shortest distance in moving from node 1 to node 2 will be from node 1(which carries the permanent label (0,1) through node 3 [30,2] through node 4 [40,3] then finally to node 2 which will have the permanent label [55,4]
The solution in Figure 2 provides the shortest distance to each node in the network together with it route. In summary, Dijkstra's algorithm finds the shortest paths from a source node *s* to all other nodes in a network with non-negative arc lengths. Dijkstra's algorithm maintains a distance label *d(i)* with each node *i*, which is upper bound on the shortest path length from the source node to each node *i*. At any immediate step, the algorithm divides the nodes of the network under consideration into two groups: those, which it designates as permanently labelled (or permanent), and those, which it designates as temporarily labelled (or temporary). The

distance label to any permanent node represents the shortest distance from the source node to that node. The basic idea of the algorithm is to find out from the source node *s* and permanently label node in the order of their distances from the node *s*. Initially, node *s* is assigned permanent label of zero, and each other node *j* a temporary label equal to infinity. At each iteration, the label of a node *i* is the shortest from the source node along a path whose internal node (i.e. node other than *s* or the node *i* itself) are all permanently label. The algorithm selects a node *i* with the minimum temporary label (breaking ties arbitrary), makes it permanent, and reaches out from that node-that, seems all the edges/arcs emanating from the node *i* to update the distance labels of adjacent nodes. The algorithm terminates when it has designated all nodes permanent (Ahuja et al, 1993).

8. The Step Description

8.1. Step 1

The source node *s* is initialised as *scanned* (*s* ε *SSN*) and every other node *xi.s* of the graph is initialised as *not reached* (*xi* ε *SNRN*). Furthermore, the arrival time of the source node is set to *t0* (*t0* is the time at which the journey starts), i.e. *T(s) = t0*, and the arrival time of every other node *xi* sof the graph is set to infinity, i.e. *T (xi)* = ∞

8.2. Step 2

We process only one node during this step. We choose the node to be processed from the SSN (Set of Scanned Nodes). If the SSN is empty, this means there is no path between the source node and the destination nodes and the algorithm quits. If the SSN is not empty, we choose an*xi* node from the SSN which has minimal *T(xi)*. If there is more than one node with the minimal *T(xi)* then we choose one of them arbitrarily. In formula:

Xi SSN∈*T (xi)* = min *T(xj)* Xi *SSN* ∈

Once the *xi* node is chosen, we proceed to process it. First the node *xi* is excluded from the SSN and becomes a member of SPN (Set of Permanent Nodes). At this stage, it is certain that the arrival time *T(xi)* is minimal (it may only get larger since taking another link will increase the cost; link costs are always positive) and this is the reason for moving the *xi* node to SPN.

Next the links leaving the *xi* node are handled. From the set E (the set of links of the graph) every link which has the *xi* node as a starting one is selected. The retrieved set is further constrained to links of the departure time greater or equal to *T(xi)* (only buses that will arrive to a bus stop can be taken, not those which have left).

9. Data Collection and Analysis

Kumasi is the capital city of the Ashanti Region, a very important and historical Centre for Ghana. It is located about 250 km (by road) northwest of Accra. Kumasi is approximately 300 miles north of the equator and 100miles north of the Gulf of Guinea. It is the second largest city of Ghana with a population of 1,517,000. The metropolis is made up of 119 sub metros. Currently the emergence service for Kumasi Metropolis can all be located in Adum. Whereas the Ambulance service in the metropolis is located at the Komfo Anokye Teaching Hospital (KATH) and the Fire Service can also be located at Adum near the Kumasi metropolitan office. Cases handled by the Metropolis Ambulance Service (MAS) and Metropolitan Fire Services (MFS) range from Gynaecology, fire and to road accidents.

The MAS and MFS are both housed in a separate building at the KATH polyclinic and KMA and both runs two shifts systems; day and night. Communication is the key to running of these services.

This thesis offers an application solution for dynamic routing of vehicles in Kumasi. It proposes a routing system that users employs historical traffic data to model recurring congestion and compute initial shortest path. As unpredicted (nonrecurring) congestion occurs and is reported from some FM station or traffic control centre, the system analyses the real time data to determine if the planned route needs to be change (modified). It can changed the planned route as a function of the current position, destination location, and real time traffic condition The proposed routing system has been composed of three subsystems including ArcGIS Network Analyst (for the digitized map), Dijkstra's algorithm and VB.Net for the software development. The routing optimization problem in traffic management has been already explored with a number of algorithms. Routing algorithms use a standard of measurement called a metric (i.e. path length) to determine the optimal route or path to a specified destination. Optimal routes are determined by comparing metrics, and these metrics can differ depending on the design of the routing algorithm used (Parker, 2001).

Different kinds of algorithms have been proposed to finding the optimal routes, such as:

i. Simulated Annealing is a related global optimization technique which traverses the search space by generating neighbouring solutions of the current solution (Kirkpatrick et al., 1983).
ii. Tabu Search is similar to Simulated Annealing, in that both traverse the solution space by testing mutations of an individual solution. While simulated annealing generates only one mutated solution, tabu search generates many mutated solutions and moves to the solution with the lowest fitness of those generated (Glover et al., 1997).
iii. Genetic Algorithms (Holland, 1975) use biological methods such as reproduction, crossover, and mutation to quickly search for solutions to complex problems. Genetic algorithm begins with a random set of possible solutions. In each step, a fixed number of the better current solutions are saved and they are used to the next step to generate new solutions using genetic operators.
iv. The ant colony optimization algorithm which has been used to produce near optimal solutions to the travelling salesman problem. They have an advantage over simulated annealing and genetic algorithm approaches when the graph may change dynamically (Dorigo et al., 1999).

v. Dijkstra‘s algorithm, used by Network Analyst, is a greedy algorithm that solves the single-source shortest path problem for a directed graph with nonnegative edge weights (Dijkstra, 1959).

However, Network Analyst is still relatively new software, so there is not much published material concerning its application traffic management. Miller (2005) compares the RouteSmart 4.40, the ArcLogistics Route and the ArcMap Network Analyst extension on the ability of either software package to create routes usable by the Drivers in Adum, efficient manner for the city of Kumasi in Ashanti Region.

Dijkstra's Algorithm, introduced in 1959 provides one the most efficient algorithms for solving the shortest-path problem. In a network, it is frequently desired to find the shortest path between two nodes. The weights attached to the edges can be used to represent quantities such as distances, costs or times. In general, if we wish to find the minimum distance from one given node of a network, called the source node or start node, to all the nodes of the network, Dijkstra's algorithm is one of the most efficient techniques to implement. In general, the distance along a path is the sum of the weights of that path. The minimum distance from node ***a*** to ***b*** is the minimum of the distance of any path from node ***a*** to ***b***.

9.1. Network Data Analysis and Results

ArcGIS Network Analyst is a powerful extension that provides network-based spatial analysis including routing, travel directions, closest facility, and service area analysis. ArcGIS Network Analyst enables users to dynamically model realistic network conditions, including turn restrictions, speed limits, height restrictions, and traffic conditions at different times of the day (ESRI 2006). The users with Network Analyst extension are able to:

i. Find efficient travel routes,
ii. Determine which facility or vehicle is closest,
iii. Generate travel directions, and
iv. Find a service area around a site.

In the current work, using Network Analyst, an optimum route for the routine find in particular area is generated in the area under study. Network Analyst uses the Dijkstra‘s Algorithm (Dijkstra 1959) in order to solve the Routing Problem and it can be generated based on two criteria (Lakshumi et al 2006):

(i).**Distance criteria:** The route is generated taking only into consideration the location of the waste large items. The volume of traffic in the roads is not considered in this case.

(ii).**Time criteria:** The total travel time in each road segment should be considered as the: Total travel time in the route = runtime of the vehicle + distance time. The runtime of the vehicle is calculated by considering the length of the road and the speed of the vehicle in each road. The Network Analyst extension allows the user to perform ‖Find Best Route‖, which solves a network problem by finding the least cost impedance path on the network from one stop to one or more stops. Network modelling gives the opportunity to the user to include the rules relating to the objects, arcs and events in association with solving transportation problems (Stewart 2004).

9.2. The Path Finding Algorithm

Network Analyst software determines the best route by using an algorithm which finds the shortest path, developed by Edgar Dijkstra (1959). Dijkstra‘s algorithm is the simplest path finding algorithm, even though these days a lot of other algorithms have been developed.

Dijkstra‘s algorithm reduces the amount of computational time and power needed to find the optimal path. The algorithm strikes a balance by calculating a path which is close to the optimal path that is computationally manageable (Olivera, 2002). The algorithm breaks the network into nodes (where lines join, start or end) and the paths between such nodes are represented by lines. In addition, each line has an associated cost representing the cost (length) of each line in order to reach a node. There are many possible paths between the origin and destination, but the path calculated depends on which nodes are visited and in which order. The idea is that, each time the node, to be visited next, is selected after a sequence of comparative iterations, during which, each candidate-node is compared with others in terms of cost (Stewart, 2004).

The following comprehensible example, which is an application of the algorithm on a case of 6 nodes connected by directed lines with assigned costs, explains the number of steps between each of the iteration of the algorithm (Figure 3). The shortest path from node 1 to the other nodes can be found by tracing back predecessors (bold arrows), while the path‘s cost is noted above the node.

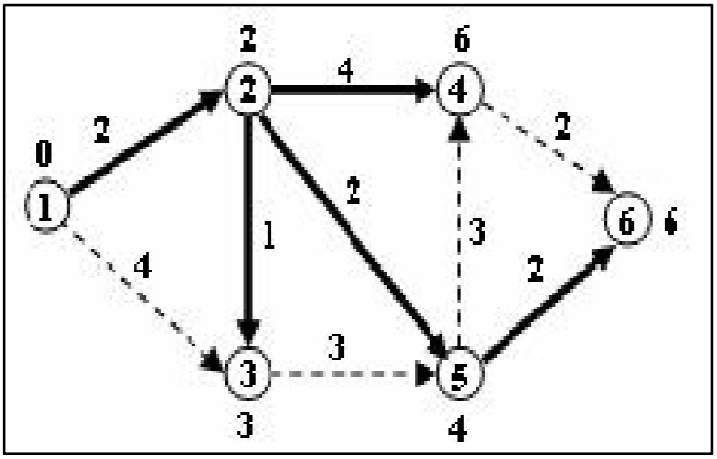

Figure 3: An example of Dijkstra's algorithm (Orlin 2003).

.

Each node is processed exactly once according to an order that is being specified below. Node 1 (i.e. origin node) is processed first. A record of the nodes that were processed is kept; call it Queue (Table 1). So initially Queue = {1}. When node k is processed the following task is performed: If the path's cost from the origin node to j could be improved including the vertex (k, j) in the path then, an update follows both of Distance[j] with the new cost and Predecessors[j] with k, where j is any of the unprocessed nodes and Distance [] is the path's cost from the origin node to j. The next node to be processed is the one with the minimum Distance [j], in other words is the nearest to the origin node among all the nodes that are yet to be processed. The shortest route is found by tracing back predecessors.

Queue	Next node	Distance						Predecessors				
		1	2	3	4	5	6	2	3	4	5	6
1	2	-	2	4	∞	∞	∞					
1,2	3	-	-	3	6	4	∞		2	2	2	
1,2,3	4	-	-	-	6	4	∞					
1,2,3,5	5	-	-	-	6	-	6					5
1,2,3,5,4	6	-	-	-	-	-	6					
1,2,3,5,4,6	-	-	-	-	-	-	-					

Table 1: A record, called Queue, with all processed nodes

Network Analyst can be very useful in a variety of sections (ESRI 2006) in our daily life, such as in:

i. Business, scheduling deliveries and installations while including time window restrictions, or calculating drive time to determine customer base, taking into account rush hour versus midday traffic volumes.
ii. Education, generating school bus routes honouring curb approach and no U-turn rules.
iii. Environmental Health, determining effective routes for county health inspectors.
iv. Public Safety, routing emergency response crews to incidents, or calculating drive time for first responder planning.
v. Public Works, determining the optimal route for point-to-point pickups of massive trash items or routing of repair crews.
vi. Retail, finding the closest store based on a customer's location including the ability to return the closest ranked by distance.
vii. Transportation, calculating accessibility for mass transit systems by using a complex network dataset.

9.3. Case Study

A digital road network in small area of Kumasi (Adum), capital of Ashanti Region, was used within the GIS map at a scale of about 1:2000. The road network was represented as connections of the nodes and links. Geometric networks are built in the ArcGIS model to construct and maintain topological connectivity for the road data in order to allow the path finding analysis to be possible. To plan the initial shortest path, use historical data of average traffic volume at surface streets or freeway segments within the area under study. The segment lengths have been extracted using ESRI's ArcGIS software. The average volume of each link in the network has been from obtained from KMA Traffic Unit. Summation of the travel distance (times) for all the segment of a particular path between origin and destination provides the total distance (time), which is minimized by the shortest path algorithm. The routing macro uses Dijkstra's routing algorithm.

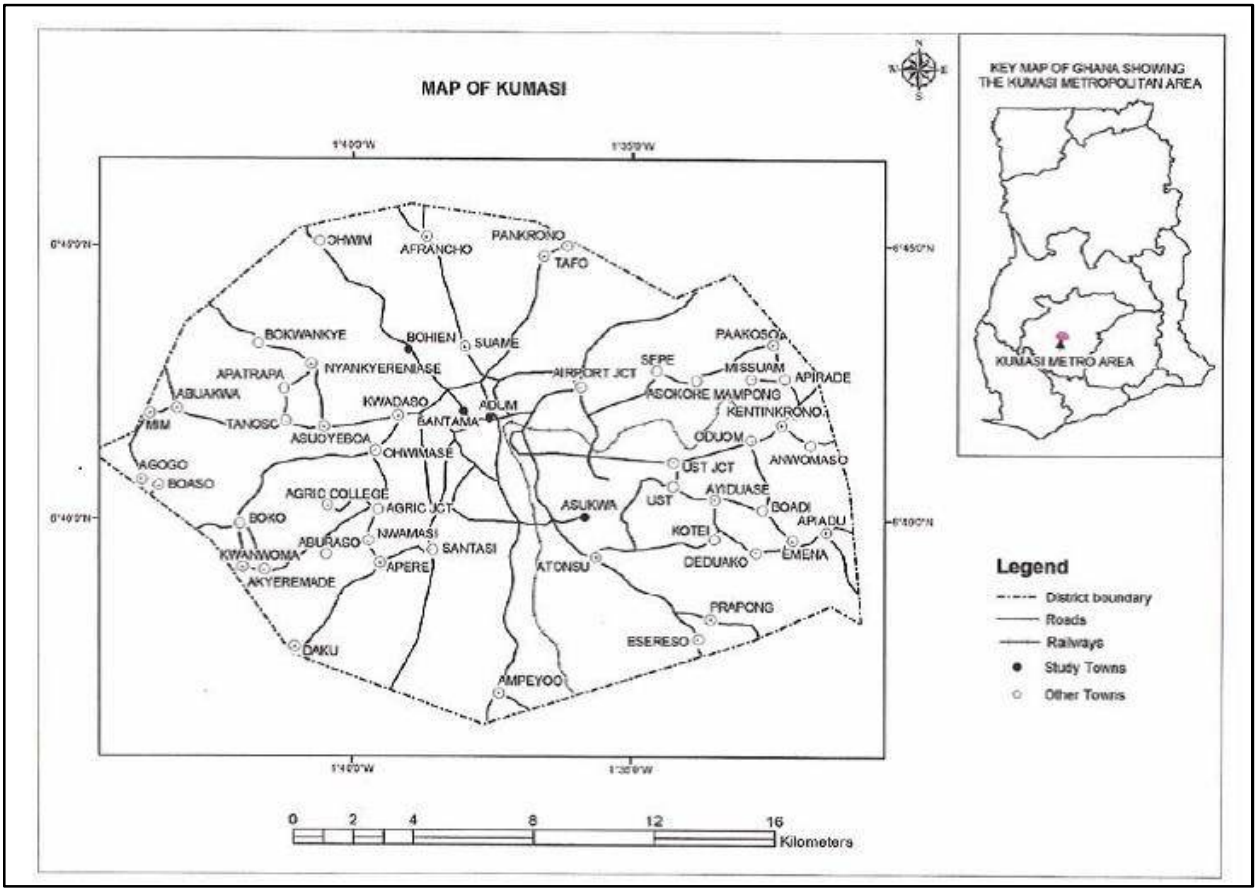

Figure 4: map of Kumasi metropolis

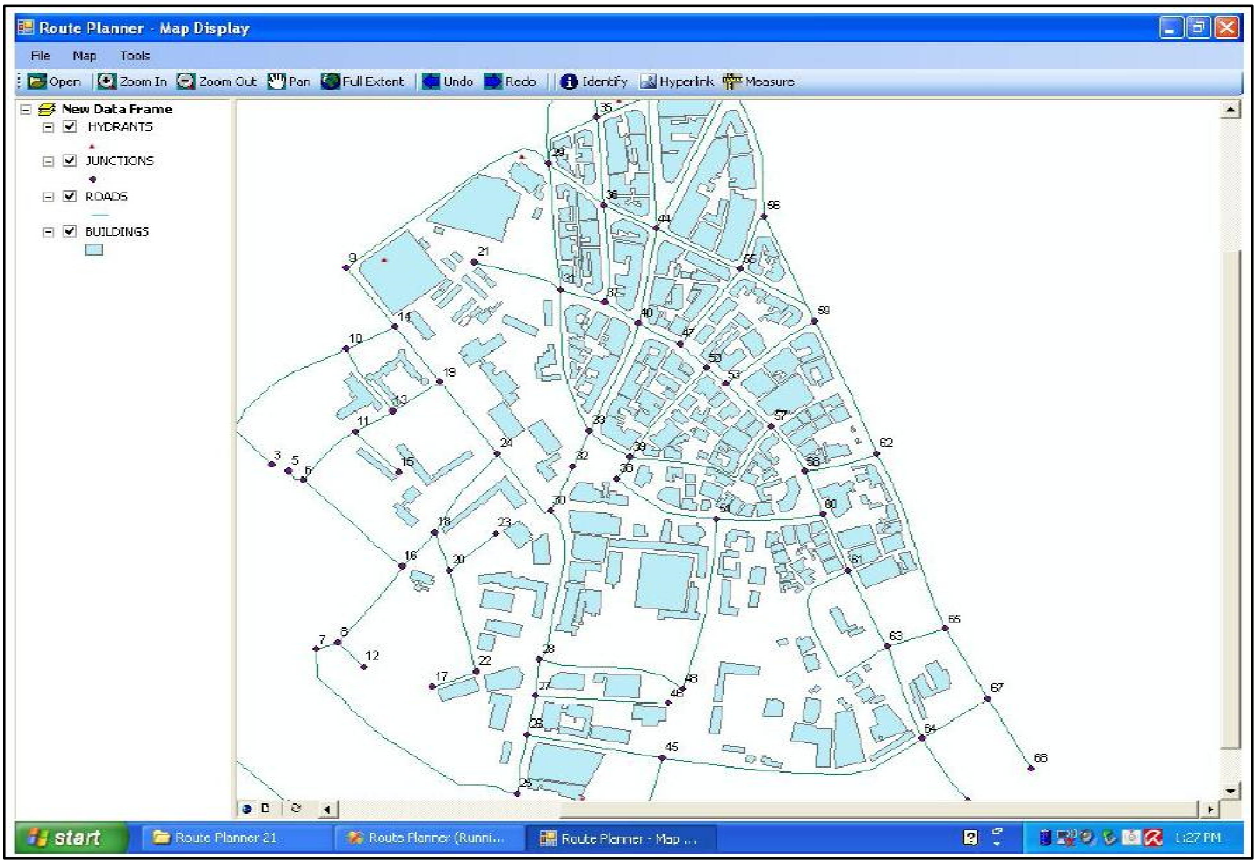

Figure 5

9.4. Proposed Solution

This thesis describes a study of planning vehicle routes for the shortest path in a district of Adum using Network Analyst - a user-friendly extension of ArcGIS and Visual Basis Dot Net with Dijkstra's algorithm, which provides efficient routing solutions in a simple and straightforward manner. In order to simulate the situation in ArcGIS, all the relevant information was acquired from KMA. More precisely, when creating a network routing solution, specific spatial data are needed for the accurate completion of the network. For example, a complete road network, where all the roads within the network are connected, is significant because it allows connection throughout the system.

9.5. Model Assumptions

- Traffic congestion not considered
- Calculation based on road distance
- State of the road not considered

Adum map was taken from the Town and Country Planning Department of KMA. Digitized by the Geodetic Department (KNUST) to convert the map into a road network

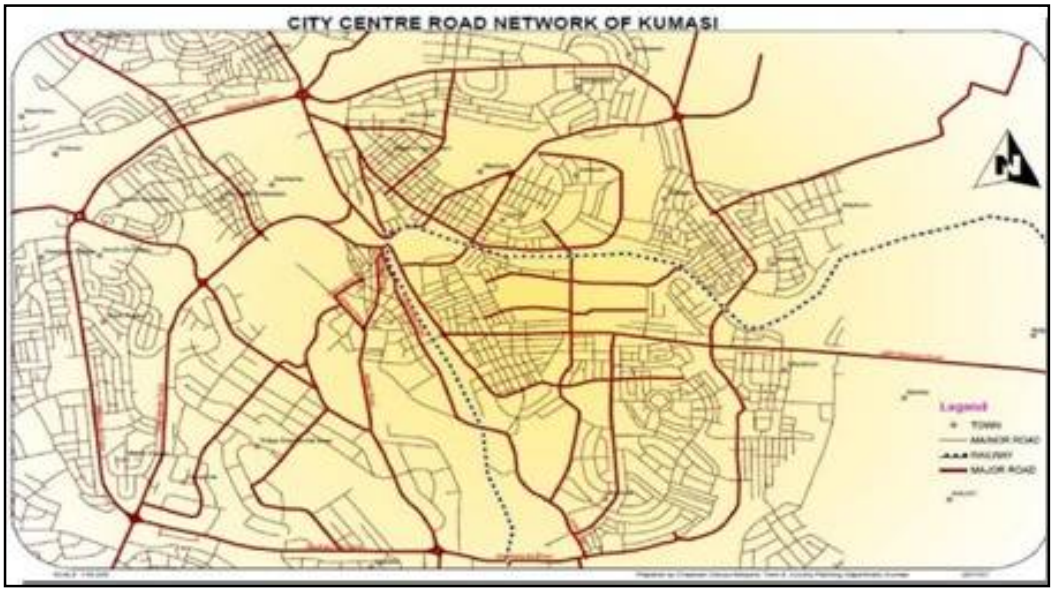

Figure 6: Shows the City Centre Road Network of Kumasi

9.6. Extract Map of Adum Network

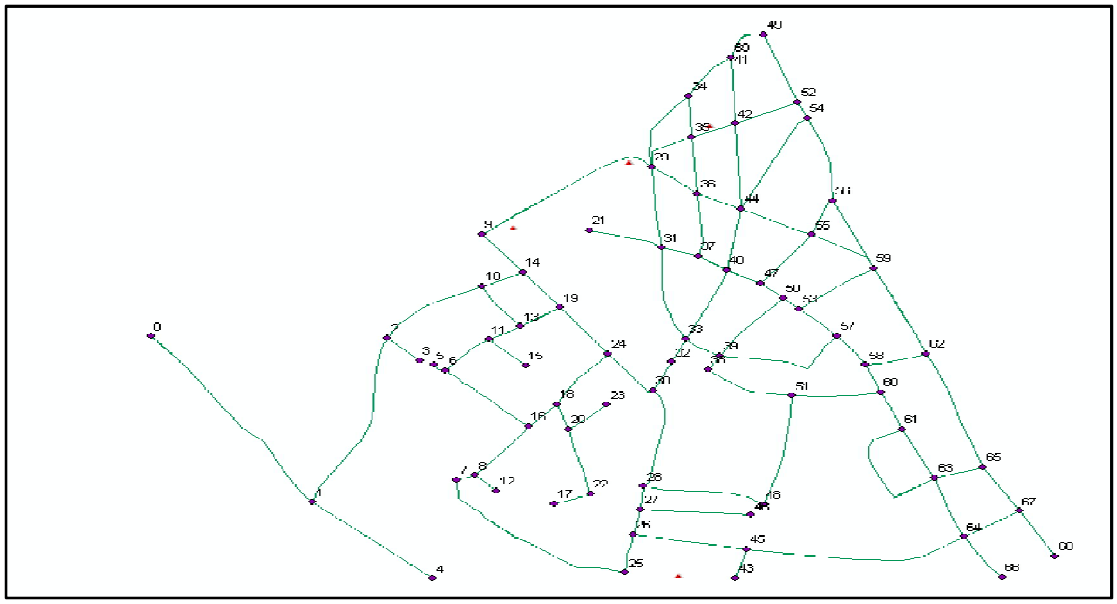

Figure 7: Shows the Extract Map of Adum Network

9.7. Software Development

The proposed routing system has been of three subsystems including:

- ArcGIS Network Analyst
- Dijkstra's Algorithm
- VB.Net

For the software development

9.8. Features of the Interface

- Step one: The first interface of the program.

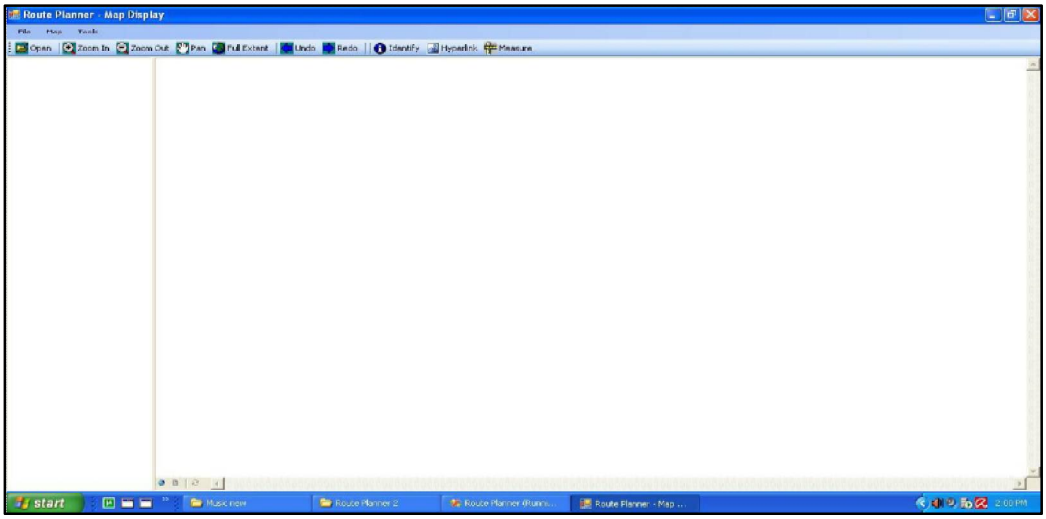

Figure 8: Shows the first interface of the program

- Step two: The user open to select to select the map needed.

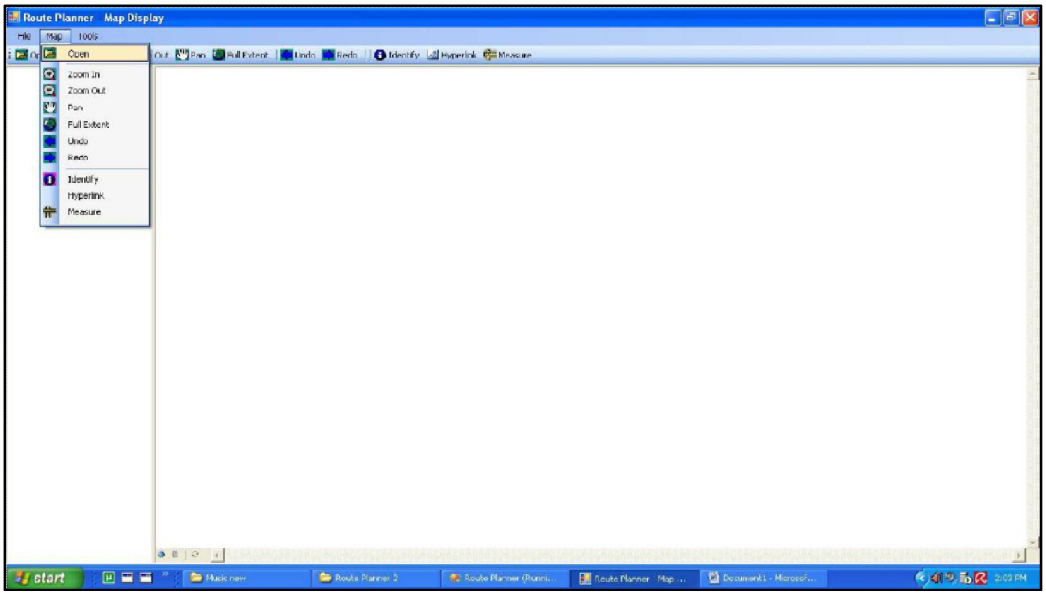

Figure 9: Shows the how users select a map

- Step Three: The user selects the needed map from the dialogue box to be open.

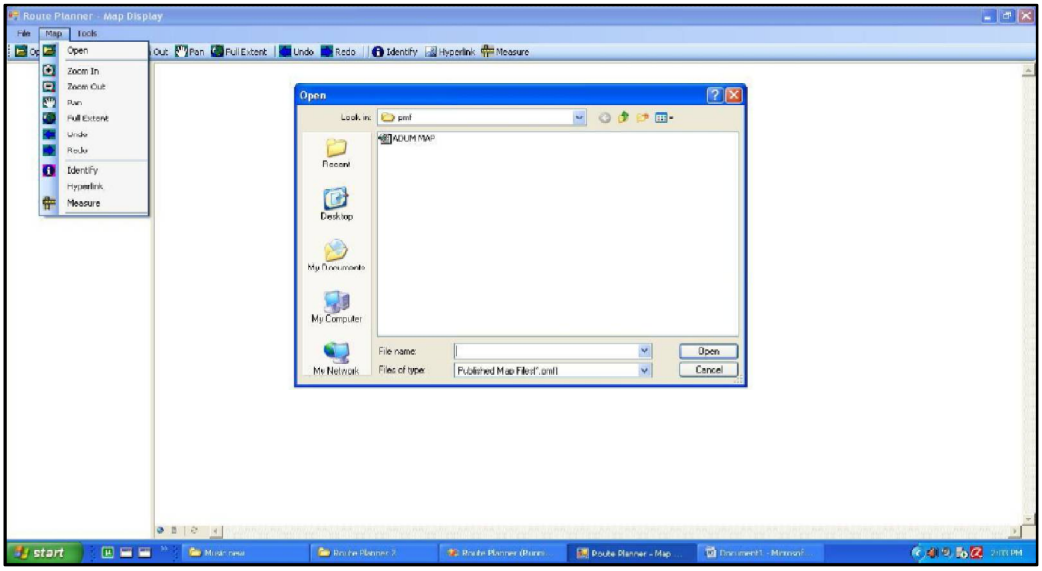

Figure 10: Shows how the maps are the display and selected

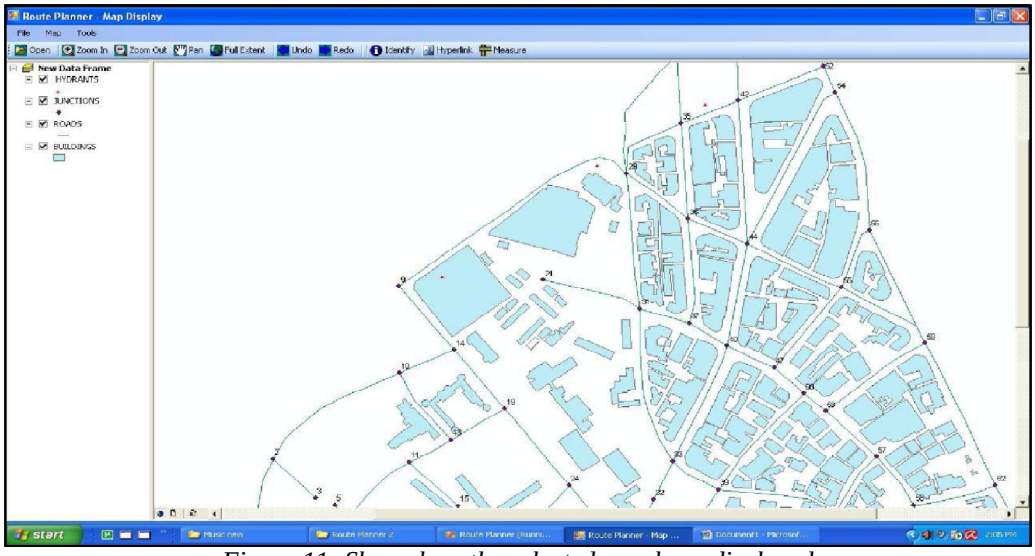

Figure 11: Shows how the selected map been displayed

- Step Four: The user uses the tool menu to select the Shortest Path Navigator where the user selects the Source Street and the Destination. The flash bottom flashes the selected street and the Flicker also flicks the selected street. The Go button uses to calculate the shortest distance from the Source Street to the Destination Street on the map, and then display the distance on the blank space. The Flash Features shows the shortest path on the map.

Figure 12: Shows how the user selects the Source Street and the Destination

- Step Five: How the streets are been selected.

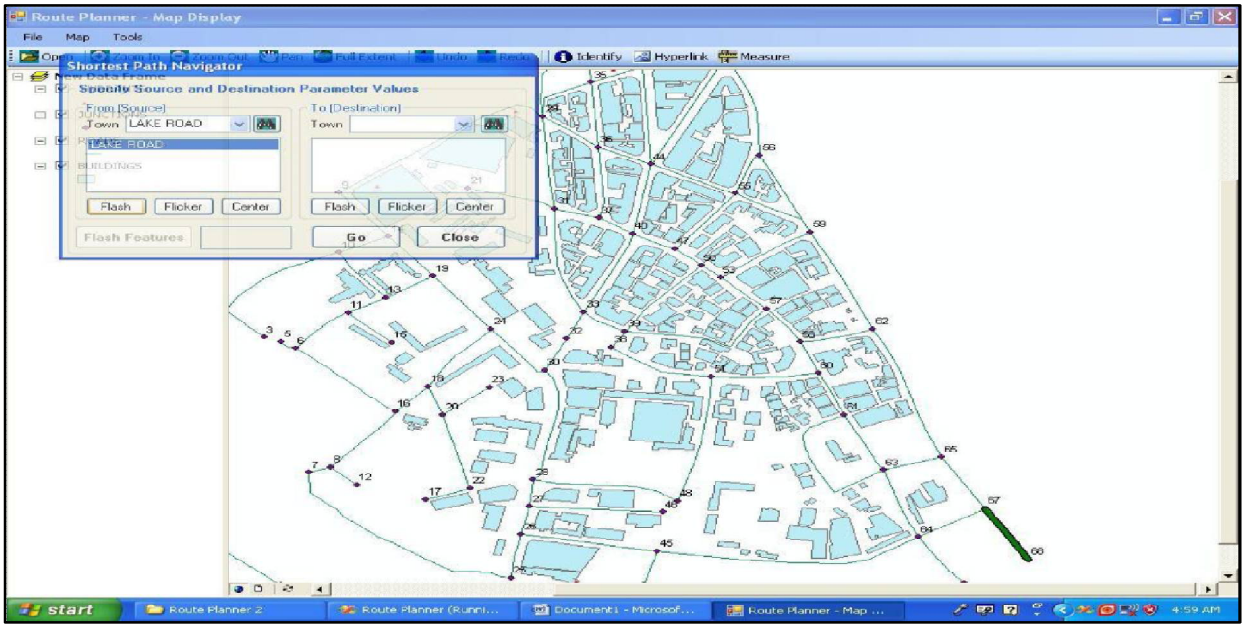

Figure 13: Shows how user selects the Source Street.

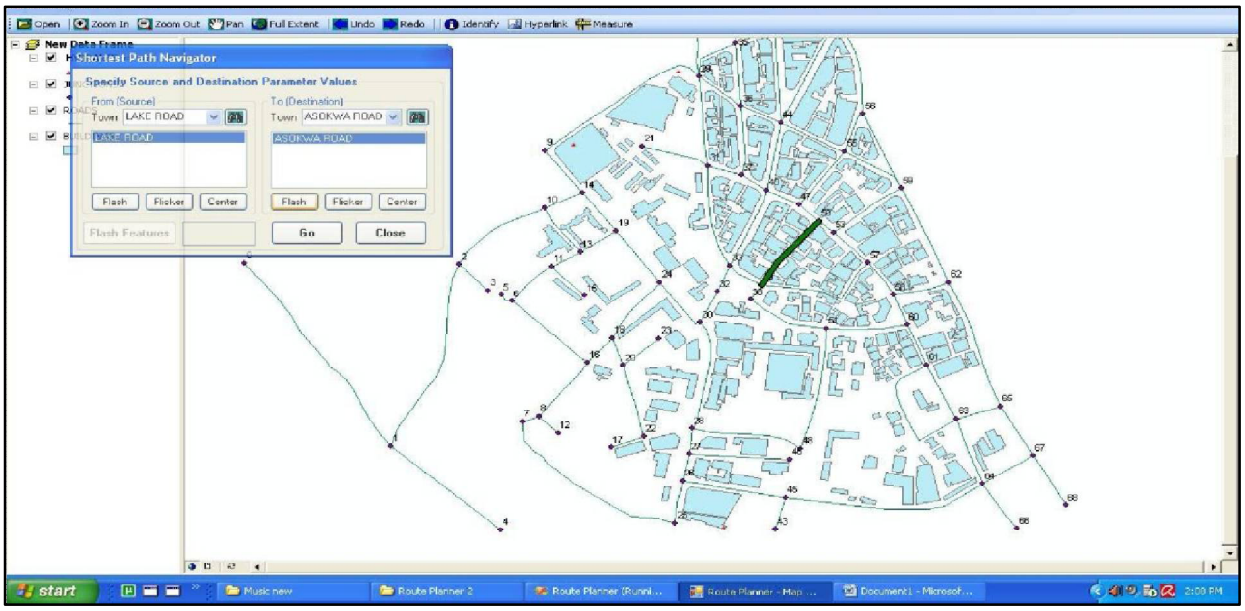

Figure 14: Shows how user selects the Destination Street.

9.9. Discussion and Future Work

These interfaces show how legend was loaded and displayed for the software obtained. The source node Adum (A) was made fixed and the destination node was to be selected. The program was run after loading the data .The computation was done by the program as: Adum chosen as the starting node (A) was assigned a permanent label of zero and each other node a temporal label. Each node is labelled with a distance from the start and a previous node. Each node is held in a queue to be evaluated later. The algorithm select a node with minimum temporal label to be evaluated from the queue, makes it permanent and reaches out from that node. All nodes adjacent to this vertex that have been visited were labelled and held in the queue.

The process continues until all the nodes in the queue had permanent label and algorithm terminated. The shortest path to a given node was labelled on that vertex. The path was found by tracking back through the network. The software displayed the source, destination, the shortest path and the optimal distance in kilometres. The result attained is able to provide the shortest distance from Adum to any location. It also provides the routes to obtain the shortest distances. Computation of shortest paths is a famous area of research in Computer Science, Operations Research and GIS. There is a great number of ways to calculate shortest paths depending on the type of network and problem specification. Network Analyst is not only capable of reproducing a satisfying number of scenarios, but also it has the ability to be easily adapted to new conditions.

10. Conclusions and Recommendations

10.1. Conclusion

In conclusion the shortest distance from any area on Kumasi to another can be calculated, let us have a look at the case of an emergence call, requesting an ambulance to rush a patient from any of the part of Kumasi to a KATH hospital. The shortest distance can easily be known using this project, because a link on a real road network in the city tends to posse different levels of congestion during different time period of a day and because a Patient‘s location cannot be expected to be known in advance, it is practically impossible to determine the fastest route before a call is received. The collection, transport and disposal of solid waste, which is a highly visible and important municipal service, involves a large expenditure but receives, scant attention. This problem is even more crucial for large cities in developing countries due to the hot weather once again the shortest distance can also be calculated using this project. This study addresses the problem of determining dynamic shortest path in traffic networks, where arc travel times vary over time. This study proposes a dynamic routing system which is based on the integration of GIS and real-time traffic conditions. It uses GIS for improving the visualization of the urban network map and analysis of car routing. GIS is used as a powerful functionality for planning optimal routes based on particular map travel time information. The results of this study illustrate that dynamic routing of emergency vehicle compared with static solution is much more efficient. This efficiency will be most important when unwanted incidents take place in roads and serious traffic congestion occur. In this study, the initial planned route is saved since when exist at any distance. The routing system analysis real distance data receives only portion of the planned path traffic data and vehicle location to determine if the direction may be a changed. This improves the computational planned route need to be modified.

- This study addresses the problem of determining shortest path in traffic networks, in Kumasi Metropolis
- The study proposes a routing system which is based on the integration of ArcGIS and road distance.
- It uses ArcGIS and VB.NET coding to obtain user friendly interface which allows the visualization of the Adum road map and traversal of shortest route between two selected junctions. The updated route is send via a dynamic routing system for all vehicles in urban (Adum) road communication system to vehicle driver to change his network has some special considerations which are the route. This process continues until the mission of subject of our future work.

10.2. Suggestion

Sometimes the given algorithms may produce output that is of no use even though it has been correctly generated. For example, there can be a path that will require an ambulance and one bus only to reach the destination after 30 minutes. However, the algorithm may advise you to take a car and three times to take a bus which will take 25 minutes, 5 minutes less than the previous path. From the point of view of the defined conditions the second path is better, but a more reasonable path is the first one, though 5 minutes shorter. The first path is actually better because it is less cumbersome (it is easier to take one bus instead of three), more reliable (three buses cause more risk than only one since each bus can break down, changing buses is risky as opposed to sitting in the bus) and is cheaper. This example proves the need to introduce different conditions for solving the shortest route problem. The future research can go into two directions. First, well known algorithms can be adapted into the public transport needs. For example, the algorithm for finding second shortest path, third etc, paths for buses can be developed. More can be proposed: finding the shortest path going through specific nodes, through specific number of nodes or by the most reliable path.

The other direction is more interesting: development of new algorithms for traffic issues and not just adaptation of existing algorithms. So far there has not been devised (as far as the author of this report knows) an algorithm for many public transport means: a train, an underground, buses and a car. There would not be anything interesting in this except that the buses and metro would be considered in parallel. A user could point out that the path should be build up in accordance with the following criteria:

i. The allowed types of changes (for example to change a bus to a train may be disallowed).
ii. Transportation, calculating accessibility for mass transit systems by using a complex network map.
iii. Public works, determining the optimal route for point- to – point pickups of trash items.
iv. Public Safety, routing emergency response crew to incidents, or calculating drive distance for first responder planning.

More than that, user can specify exactly how many changes he wants between different types of transportation. For example the user can say that only one change between car and bus is allowed but that changing between buses and an underground vehicle can be done as many times as necessary. Also, the number of changes can be named as *at* most or exactly. Therefore saying at most 3 changes of vehicle can ban choosing the best route with only one change. But still, this is should also be possible to find the path with exact number of changes. The flexibility of conditions seems to be very big.

11. References

i. Ahuja, R. K., Magnanti, T. L., Orlin, J. B., (1993). Network Flows: Theory, Algorithms and Applications, Prentice Hall, Englewood Cliffs, NJ
ii. (S. K. Amponsah, Gordon Amoako*, K. F. Darkwah and E. Agyeman, January, 2011)
iii. Arrival Time Dependent Shortest Path by On – Road Routing in Mobile Ad – Hoc Network (2005)
iv. Bellman, R., (1958) On a Routing Problem, Quart. Appl. Math. 16, 87-90
v. Cherkassky, B. V., Goldberg, A. V., Radzik, T., 1996, Shortest path algorithms: Theory and experimental evaluation, Mathematical Programming 73, 129-174 Cai, X., Klocks, T., Wong, C.K., (1997)
vi. CHISHAKI, Guoquan Li and Wen-Chih Huang (1994) The Developing Trend of Taxi Traffic in Beijing Metropolitan Region Zhongying Dong, Takeshi

vii. Dijkstra, E.W., 1959. ―A note on two problems in connexion with graphs‖. Numerische Mathematik, 1, 269-271.
viii. Dreyfus, S. E., An Appraisal of Some Shortest-Path Algorithms, Operations Research 1969, 395-412
ix. Hart, P. E., Nilsson, N. J., Raphael, B. (1972). Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", SIGART Newsletter, 37, pages. 28-29.
x. Husdal, J. (2000). Fastest Path Problems in Dynamic Transportation Networks, http://www.husdal.com/mscgis/research.htm, last accessed November 22, 2005.
xi. Network Flows: Theory, Algorithms and Applications, Prentice Hall, Englewood Cliffs, NJ
xii. Minty, G., A comment on the shortest route problem. Operations Research. 5, 724, 1957
xiii. Moore, E.F., The shortest path through a maze. Proceeding of an International Symposium on the theory of Switching, Part II, April 2-5, 1957, The Annals of the Computation Laboratory of Harvard University 30, Harvard University Press, and Cambridge Mass.
xiv. Open GIS - A Request for Technology - In Support of an Open Location Service (OpenLSTM) Testbed, 2000.
xv. Optimisation of an Existing Forest Road Network Using Network 2000
xvi. Optimisation of Yoshitaka AOYAMA & Yoshinobu HIROSE (1994)) The Impact of the Development of High Mobility Transportation Networks on Rural Cities, Related Problems and Countermeasures.
xvii. Peter W. Eklund, Steve Kirkby1, Simon Pollitt (2001) A Dynamic Multi-source Dijkstra‘s Algorithm for Vehicle Routing
xviii. Skiena, S. (1990). Implementing Discrete Mathematics: Combinatorics and Graph
xix. Steenbrink, P. A., Optimisation of transport networks, John Wiley & Sons Ltd, 1974
xx. Stewart, L.A., 2004. ―The Application of Route Network Analysis to Commercial ForestryTransportation‖. (Accessed on February 10, 2007).
xxi. Syslo ,M.M., Deo ,N., and Kowalk ,J.S. (1983). Optimization on Networks. In Greenblah , A., editor , Discrete Optimization Algorithms with Pascal Programs , Pages 221-392.Prentice _ Hall ,Inc., Eaglewood Cliffs, New Jessey.
xxii. Tadaomi SETOGUCHI (1994), Observation on the Characteristics of Converted Traffic,Viewed from the Available Forms of Urban Express ways.Hiroshi TANOUE, Takashi
xxiii. Vonderohe, A. P., Travis, L., Smith, R. L. and Tasai, V. (1993). NCHRP Report 359,Adoption of Geographic Information System for Transportation, Transport Research Board, National Research Council, Washington, DC.
xxiv. Young-Hwan Lee (1994) .A Study on the Development Plan of the New Yong-Jong Island International Airport