KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

MODELLING THE SHORTEST PATH PROBLEM BY USING LINEAR PROGRAMMING APPROACH. A CASE OF NKAWIE FIRE STATION.

BY

VIVIAN OSEI-BUABENG

A THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS, KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF M SC. INDUSTRIAL MATHEMATICS

JUNE, 2015.

DECLARATION

I hereby declare that this submission is my own work towards the award of the M Sc. degree and that, to the best of my knowledge, it contains no material previously published by another person nor material which had been accepted for the award of any other degree of the university, except where due acknowledgment had been made in the text.

Vivian Osei-Buabeng	.....	.....
Student	Signature	Date

Certified by:

Prof. S. K . Amponsah	.....	.....
Supervisor	Signature	Date

Certified by:

Prof. S. K. Amponsah	.....	.....
Head of Department	Signature	Date

DEDICATION

...To the Lord of Host, Alpha and Omega God Almighty for the strength and insight. Also to my family, Osei-Buabeng family

ABSTRACT

There has been a number of fire outbreak cases recorded in the Atwima Nwabiagya District that has brought about loss of lives of inhabitants and loss of properties. Some routes within the district can be reconstructed into bitumen roads so that fire attackers can traverse through the district in order to prevent fire incidents. The main objective in this study is finding the minimum travel distances and shortest paths from the Nkawie Fire Station to all other towns in the district of Atwima Nwabiagya district in the Ashanti Region of Ghana.

Shortest path algorithms of various variants have been discussed with examples in this study as well as review of abstracts of other related books and articles. The linear programming approach was the method used. The primal and dual models have been explained and dual model in which we maximize the source node minus the destination node subject to an inequality constraints, was the appropriate model used to write the constraints of the problem.

It was found out that as a destination is used in the objective function, the dual algorithm proceeds to obtaining minimum distances to every other destination. The Lindo 6.1 software was used to solve the maximization problem and the results are found in the appendices of this study. There was an optimal distances from Nkawie (NK) to all the towns in the district. The distance from Nkawie (NK) to Mfensi (MF) was found to be 14.3km, that of Akropong(AK) was 21.1km etc.

ACKNOWLEDGMENT

I would like to express my special appreciation and thanks to my supervisor Professor S. K. Amponsah, you have been a tremendous mentor to me. I would like to thank you for encouraging my research and for allowing me to grow as a researcher.

I would also like to thank the Lecturers of the Department of Mathematics who spent time with me during the synopses of the mock presentations. Their brilliant comments and suggestions enlightened my ideas of this study.

I would especially like to thank Dr. Gabriel Okyere and the Statistical Consulting and Quantitative Skill Development Unit of the Maths Department of the University, for introducing me to LaTeX software without it, my thesis would not have been presented orderly. Indeed, I have grown keen interest in this powerful software.

A special thanks to my family. Words cannot express how grateful I am to my parents Martin and Alice Osei-Buabeng, two sisters, Huldah and Eunice and two brothers Peter and Daniel for all of the sacrifices that you made on my behalf. Your support in prayer was what sustained me this far.

Last but not least, I would also like to thank my good friends and course mates who supported me in writing, and incited me to strive towards my goal even through thick and thin. Thank you all.

CONTENTS

LIST OF ABBREVIATION

SPP ..... Shortest Path Problem

SSSPP ..... Single-Source Shortest Path Problem

MST ..... Minimum Spanning Tree

LP ..... Linear Programming

GNFS ..... Ghana National Fire Service

TSP ..... Traveling Salesman Problem

DPCU ..... District Planning and Co-ordinating Unit

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Imagine you wanted to travel from Kumasi in Ashanti Region to visit the place Mankessim,in the Central Region, what is the fastest way to get there?. Imagine you wanted to communicate with a friend abroad, what is the fastest medium to use? Imagine you wanted to access a web page on the internet, which routers should be used such that the necessary information is downloaded to your computer faster?

These questions have something in common in that the optimal solution is the shortest path between two points of a network: a transformation network, a social network and a router network respectively. All roads may lead to Rome but we wish to arrive as soon as possible.

In this study, the researcher is interested and wish to find the shortest path from one given town (Nkawie) to another that passes through certain specified intermediate communities in the same road network within the Atwima Nwabiagya district. In some applications, one requires not only the shortest path but also the second and third shortest paths. Three important shortest path problems are elaborated as follows.

(i) How to determine the shortest path or distance from a single source or point to a single destination. For example, from an ambulance station to an accident scene.

(ii) How to determine the shortest path or distance from any source or point to many destinations. An example can be sited here, supply of goods from the warehouse to shops.

(iii) How to determine the shortest path or distance from many sources or points to a single destination. One example here is students coming from various houses to a school.

An optimal solution can therefore be found to these shortest path problems using some algorithms.

1.1 Background to the study

In establishing fire emergency response database, emergency preparedness planning is an important issue that can impact people's lives. If planned properly and implemented quickly, it can save hundreds or thousands of human lives and mitigate some of the economic losses in affected areas. However if planned poorly or not implemented in a timely manner, the consequence can be dire and could cost human live. Fokuo and Quaye-Ballard (2013). Thus an optimal solution to the shortest path problem can help curtailing the loss of lives and contribute immensely to the emergency preparedness planning.

Terminologies

1. Cost, distance, and weight are the capacities on each arc and are used interchangeably in this study.

2. Graph Map: Let $G_1 and G_2$ be graphs and $f : G_1 \rightarrow G_2$ be a continuous function. Then f is called a graph map, if

(i) for each vertex $v \epsilon V(G_1)$, $f(v)$ *is a vertex in* $V(G_2)$

(ii) for each edge $e \epsilon E(G_1)$, $dim(f(e)) \leq dim(e)$

3. Arc and Edges as well as Nodes and Vertices are also used interchangeably and mean the same.

1.1.1 Shortest Path Problems

A shortest path problem is to find a path with minimum travel cost or distance from one or more origins to one or more destinations through a network. Lim and Kim (2005). Shortest path analysis is important because of its wide range of applications in transportation. Naqi et al. (2010) stated that the shortest path helps calculate the most optimal route, and optimal routing is the process of defining the best route to get from one location to another. The best route could be the shortest or fastest depending on how it is defined.

There are a wide variety of problems that go under shortest path problems. These problems are classified according to the structure, assumptions made, the type of data and the nature of solutions desired. There is also a classification of the methods used to solve these problems. The structures are:

1. between a specified pair of nodes
2. between all pairs of nodes
3. from one origin to all other nodes or from all other nodes to a single destination.

1.1.2 Shortest Path Algorithms

An algorithm is a procedure or formula for solving a problem. Shortest path algorithm on the other hand is an algorithm designed essentially to find a path of minimum lengths between two specified connected weighted graph. El-Zohny and El-Morsy (2012).

However, this study discusses the fundamental algorithms designed to find an optimal solutions to shortest path problems. The study talks of Dijkstra's, Bellman-Ford, Floyd-Warshall algorithms and the linear programming procedure that are mostly applied in the study of shortest path problems

1.1.3 The District Fire Station

The district fire station has four departments. These are: Administration, Safety, Operations and Rural Fires. The administration has the stores and accounts. The role of the safety department is to give education on fire prevention to institutions such as schools, churches, banks etc. Staff in the operations department are the rescuers of fire accidents. Also known as the attackers. Rural fires department deliver information and educate the public on how to avoid bush fires in the rural areas.

1.2 Scope of the Study

The Atwima Nwabiagya District formerly the Atwima District is one of the largest among the twenty-seven(27) political and administrative districts in the Ashanti Region of Ghana. It is situated in the western part of the region and share common boundaries with Ahafo Ano South and Atwima Mponua Districts(to the west), Offinso municipal to the north, Amansie-West and Bosomtwe-Atwima Kwanwoma Distircts to the south, Kumasi Metropolis and Kwabre Districts to the East. There are about ninty (90) towns in the district.

Nkawie is the capital town of the district. It is about 20 minutes drive or 15 kilometers from Kumasi, the region's capital city. Some of the desirable (non-obnoxious) facilities that serve the entire district are hospitals, fire station and police stations. Nkawie Fire Station is the only fire emergency facility serving about 90 towns in the district.

For the purpose of this study, towns, roads and distances(km) are used to denote the nodes, arcs and cost in the network respectively.

1.3 Problem Statement

Since the year 2008, there has been thousands of fire incidents recorded across the country which has caused untimely deaths and destruction of properties. Recently, there has been number of fire outbreak cases recorded in the district. Many of these towns in the district are farther away from the only fire station in Nkawie and the main road is actually the commonest road known.

Fire Fighters/Officers need shortest paths and optimal routes to traverse in order to arrive at a fire outbreak scene on time. Thus this study seeks to finding an optimal solution using the Linear Programming Model that will minimize travel cost and time to fire incidents to curtail loss of lives and properties in the Atwima Nwabiagya District in the Ashanti Region.

1.4 Objectives of the study

The sole objectives of the study are:

- To determine the shortest paths from Nkawie fire station to all other towns in the district in order to minimize travel-distance cost to fire incidents.
- To obtain minimum travel distance for each town in the district using Linear Programming Approach.

1.5 Methodology

This section explains the methods and algorithms used in finding solutions to the problem with illustrated examples. It also discusses other related algorithms such as Prim's and Kruska's algorithms that discuss minimum connector.

Some are used when there are negative weights example of such algorithms used are the Floyd Warshall and Bellman-Ford. However Dijkstra's algorithm

solves the shortest path problem for a non-negative weight graph. The model used is the linear programming dual model. The distances between towns used in this study were collected from the Town and Country Plan Department in the district assembly. Also, related papers that were summarized in the literature review were cited from the internet. The Lindo 6.1 software was used to analyze the data.

1.6 Significance of the study

This study would be helpful for a town to provide a properly planned and organized optimal paths and the shorter distances for operation of the district's fire station as well as conducting further research regarding shortest path problems. In Atwima Nwabiagya district where the District's Fire Station is situated, cases of fire outbreaks, and deplorable long distance and single-lane routes are observed. After this study, the district will benefit greatly since more people will patronize the usage of the optimal routes and the shortest path created for the fire station through the district minimizing travel distance cost.

1.7 Limitations of the Study

The main limitations to this study are:

(i) Difficulties in accessing relevant information from the offices of the Atwima Nwabiagya District Assembly-Nkawie.

(ii) Less availability of literature relating to shortest path problems on single source to many destinations. Most of the related abstracts on shortest paths were from a single source to a single destination.

(iii) Financial constraints since the study was mainly financed by the researcher.

1.8 Justification

As a developing country, the problem of poor road networks and safety and emergency facilities like fire stations should be minimal. Thus, we can get as numerous diverse roads which are traffic free to use other than the few main streets in the regions of Ghana and Ashanti Region to be precise. The department of Road Commission must adhere to creating similar shorter paths by transforming narrow routes and some foot paths into feeder and tarred roads for easy accessibility to destinations.

Fire outbreak is a major problem that has many hazardous and tragic effects not only in the developing countries but also developed countries as well. In Ghana, fire outbreak has become rampant since 2008 to date. It has caused many untimely deaths and loss of properties. One sole reason to this effect is the deplorable and congestion in our road routings especially in the urban areas.

In this study, we wish to take the opportunity to address the need for the stakeholders to take into consideration, the reconstruction and conversion of routes or even foot paths that could serve as an easy access to carrying out services and humans into better tarred ones. In this case, travel time and cost will be minimal.

1.9 Organization of the Study

The study is organized into five chapters of which each has been elaborated. Each chapter explains a particular function to the study. Chapter one which is the introduction talks about the general introduction to the thesis topic. It identifies the background to the study in which the key phrases relating to the thesis topic are defined.

The scope and methodology of the study which give the boundary or extent of the research and brief history of the study area respectively. The problem statement which defines the main problem at hand and what the study seeks to find. Talking about the Objectives it seeks to achieve a an optimal solution to the problem. Also explains the rationale and purpose of the study.

Chapter two is Literature Review which talks about the summary of the abstracts of related books, articles, papers and journals. The next chapter is Methodology. It explains the mathematical treatment, logical presentation, formulation, models and variants

Analysis and Data Collection also follows and it explains how data was collected, analyzed and organized and discussion of results. The last item in this study is Summary, Conclusion and Recommendation.

CHAPTER 2

LITERATURE REVIEW

In this chapter, review of shortest path problems, algorithms and route network planning from related papers,journals,articles and books written by different authors are elaborated and their abstracts summarized.

2.1 Review of Shortest Path Problems

There are several impedance factors that can affect emergency services and vehicle response times. They include distance, travel time, and traffic congestion as a result of variations in traffic flow related to the time of day. Traffic congestion is a major problem in urban areas and can disrupt emergency response. Panahi and Delavar (2009).

According to the report of Szczesniak (2000), the report is concerned with the shortest path problem, the theory behind it, implementation and application Szczesniak (2000). The reports concentrates on two Shortest Path (SP) algorithms where one algorithm finds one shortest path in a network with time dependent costs of links in O(n+m) time, where n is the number of nodes, m is the number of links in the network Szczesniak (2000). The second algorithm is a combination of the first algorithm and the Dijkstra's algorithm which also runs in O(n+m)time Both algorithms are presented in the context of public transport networks and implemented in C++ using Standard Template Library.

Also, according to Wilson and Zwick (2013), they describe a new forward-backward variant of Dijkstra's and Spira's single-source shortest paths (SSSP) algorithms. While essentially all SSSP algorithm only scan edges forward and the

new algorithm scans some edges backward. The running time of the algorithm on a complete directed graph on n vertices with independent exponential edge weights is O(n), with very high probability. They also found the expected running time of the algorithm on complete graphs with independent exponential edge weights in $O\ (n^2)$. Here they took the time windows into consideration.

In the book 'Fully Dynamic Output Bounded Single Source Shortest Path Problem', the authors consider the problem of maintaining the distances and the shortest paths from a single source in either a directed or an undirected graph with positive real edge weights, handling insertions, deletions and cost updates of edges. They proposed fully dynamic algorithms with optimal requirements and query time. They based their solution on a dynamization of Dijkstra's algorithm. Frigioni et al. (1996)

According to Sumit et al. (2011), a modified algorithm of single source shortest path problem was presented in Graphics Processing Units (GPUs) using CUDA. In their paper, they first modified the standard Bellman-Ford algorithm to remove its drawbacks and make it suitable for parallel implementation, and then implement it using CUDA.

Moreover, in the abstract of the paper Risk-averse shortest path problems, they investigate routing policies for shortest path problems with uncertain arc lengths. The objective of their book was to minimize a risk measure of the total travel time. They use the conditional value-at-risk (CVaR) for where the arc lengths (durations) have known distributions and the worst-case CVaR for where these distributions are only partially described. Gavriel et al. (2012).

2.2 Review of Modified Shortest Path Algorithms

A shortest path algorithm applied to a routing problem in a transportation network can calculate the path with minimal travel cost or least impedance from

an origin to a destination. Depending on the type of cost, the shortest path can be referred to as the shortest, fastest, or most optimal path or route. Stroetmann (1997) addresses the correctness problem of an algorithm solving the constrained shortest path problem. He defines a non deterministic form of the algorithm and prove its correctness from a few simple axioms. He also shows that the abstract algorithm can be regarded as a natural extension of Moore's algorithm (Moore 1957) for solving the shortest path problem.

According to Eilam-Tzoreff (1998), the abstract discusses about disjoint shortest paths problem. In which a graph G and K pairs of distinct vertices (s_i, t_i), when $1 \leq i \leq k$ is given, they find whether there exist k pairwise disjoint shortest paths p_i, between s_i and t_i for all $1 \leq i \leq k$. They consider both directed and undirected graph.

In Porazilova (2005)'s paper, she describes the shortest path problem, its classification and the best known algorithm. She introduces a new algorithms for the Shortest Path problem and its acceleration suggested. Let ε be a real number in (0,1), the path is called an $(1+\varepsilon)$-approximation of the exactly shortest path between two points if its cost is at most $1+\varepsilon$ times the cost of the shortest path. In this paper the relative error R of the approximative shortest path P is the ratio between the shortest distance of the path from the final point and the length of the path:

$R = \frac{d(p,t)}{length(p)}$

This paper shows an application of Least-Square Primal-Dual(LSPD) algorithm to shortest path problems with non negative arc length is equivalent to the Dijkstra's algorithm. They also compare the LSPD algorithm with the conventional primal-dual algorithm is solving shortest path problems and show their difference due to degeneracy in solving the shortest path problems. Wang (2008).

According to Sinha (2004), they found the formulation of the shortest path problem for graphs without a negative-weight cycle. This method is based on the Bellman-Ford algorithm for computing shortest path. Given a graph G(V,E) with edge weights w(u,v) and two vertices-source s and t destination t; The Linear Programming(LP) formulation is as follows:

max x_t

subject to

$$\begin{cases} {}_{\forall}(u,v) \in \text{E } x_v \leq x_u + w(u,v), \\ x_s = 0 \end{cases} \tag{2.1}$$

It can be proved that, the constraints are satisfied if for all vertices v, x_v is substituted by the length of a shortest path from s to v. So, $x_t \geq d$ when d is the shortest distance of t from s. Conversely, considering any path $(x, v_1, v_2, ..., v_k, t)$ of minimal length d from s to t. Then $xv_1 \leq w(s, v_1), xv_2 \leq xv_1 + w(v_1, v_2) \leq w(x, v_1) + w(v_1, v_2)$ and so on. So, $x_t \leq w(x, v_1) + w(v_1, v_2) + ... + w(v_k, t) = d$. Hence the optimal value of x_t is equal to positive length of a shortest path from s to t.

Again, in this abstract, the author discusses the shortest path problem as an application of the theory of Metzler functions. This Metzler function is a function which arises in some system of optimization problems. The paper seeks to derive the Moore-Bellman-Ford algorithm for the shortest path problem. Masuzawa (2013). This paper also presents the Saddle Vertex Graph (SVG) a novel solution to the discrete geodesic problem. A geodesic path on the mesh is equivalent to a shortest path on the SVG, which can be solved efficiently using the shortest path algorithm eg. Dijkstra's algorithm. Ying et al. (2013).

According to Guerrero et al. (2013), they introduce the Generalized Elementary Shortest Path Problem (GESPP) where in addition to the features of the shortest path problem, nodes belong to predefined non-disjoint clusters. The clusters could be interpreted as groups of nodes with linking features.

Moreover, another study considers the 2-Way Multi Modal Shortest Path Problem (2WMMSPP). The goal of this paper is to find two multi modal paths with total minimal cost, an outgoing path and also a return path Huguet et al. (2013). This relates to this study in that, they all aim to achieving optimal shortest paths that will minimize total distance cost.

In this paper, the authors propose shortest path algorithms that use A^* search in combination with a new graph-theoretic lower-bounding technique based on landmarks and triangle inequality. The algorithms they use can compute optimal shortest paths and work on any directed graph. They also give experimental results and showing efficient of their new algorithms that outperform previous algorithms. Peng (2013). Their algorithm was actually meant to work efficiently on directed graphs.

In the abstract of "Dynamic Shortest paths minimizing travel times and costs" by Ahuja et al. (2002), they studied how to determine a shortest path from a specified node to all every other node in the network where arc travel times change dynamically. The problems and objectives of their study was similar to Dijkstra's single source shortest path problems. They considered two problems and these are: the minimum travel time walk problem which is to find a walk with the minimum travel-time and the minimum-cost walk problem which is to find a walk with the minimum weighted sum of the travel and excess travel time. Their study showed that, the minimum-time walk problem is solvable for a network known as FIFO while the minimum-cost walk problem is an NP-hard problem. They also developed a polynomial time algorithm for the minimum-time walk problem that arise in road networks with traffic lights.

In this paper, the author discusses the shortest path problem as an application of the theory of Metzler functions. Metzler functions arises in various kinds of system optimization problems such as the stable matching of two-sided markets

and the α cores with games with punishment-dominance relation they derive the Moore-Bellman Ford algorithm for the shortest path problem. Masuzawa (2013).

There has been several modifications of the shortest path algorithms for solving shortest path problems in the above abstracts. In using LP model for such problem, we find the primal dual,feasible dual and the restricted dual solutions. The primal-dual model which was modified by Barnes et al. (2002) as "The Least Square Primal-Dual Algorithm" was used in this study. The algorithm is primal-dual algorithm for solving LP problems. Instead of minimizing the sum of the Restricted Primal Problem(RPP) as does the original primal-dual algorithm PD, LSPD tries to minimize the sum of the square of the infeasibilities. It can be concluded that both LSPD and PD give the same solutions to an LP problem. Wang (2008).

Morever, this paper concerns the shortest path problem for a network in which arc costs can vary with time. They studied a corresponding linear program in space measures and prove the existence of an optimal solution. They also defined a dual problem and established strong duality results that show that, the value of the linear program equals that of the dual problem and both solutions are attained. Koch and Nasrabadi (2011)

2.3 Road Network Route Planning

Route Planning in a road networks is one of the many applications of shortest-path computations. When an approximate graph model is defined, many problems turn out to profit from shortest-path computations. The most general formulations of the shortest path problem looks at a directed graph $G = (V, E)$ and a cost function c that maps edges to arbitrary real number costs. It turns out that the most general problem is fairly expensive to solve. Thus we are interested in various restrictions that allow simpler and more efficient algorithm:

non-negativity edge costs and acyclic graphs. A route-planning algorithm requires a city map and a lot of dexterity but no computer. Lay threads along the roads on the city map. Make a knot wherever roads meet, and at your starting position. Now lift the starting knot until the entire net dangles below it. When any tangles and the threads have been successfully avoided nd the knots are thin enough so that only tight threads hinders a knot from moving down, then the tight threads define the shortest paths.

In this article, an algorithm to aid travelers' route choice decisions in road network with travel time uncertainty was proposed. The travel time of link is assumed to be spatially correlated only to neighbouring links. The spatially dependent reliable shortest path problem(SD-RSPP) was formulated as a multi criteria shortest path-finding problem. Their results demonstrated that, the size of the impact area would have a significant impact on both accuracy and computational performance of the proposal solution algorithm. Chen et al. (2012).

According to Delling et al. (2009), the authors developed methods that are faster than that of Dijkstra's. They gave an overview about the techniques that enable ongoing research on more challenging variants such as dynamic networks, time dependent and flexible objective functions of the problem.
The main objective of this thesis was to enhance and implement a principal module in TRANSIMS, called the Route Planner Module. The purpose of the Route Planner Module was to find time-dependent label-constrained shortest paths for transportation activities performed by travelers in the system. There are several variations of shortest path problems and algorithms that vary by application, contexts, complexity, required data, and computer implementation techniques. In general, these variants require some combination of a network consisting of nodes and links. The problem then seeks a shortest path between one or more origin-destination pairs.In this study, they also used the dual algorithm

of LP.Kangwalklai (2001).

2.4 Summary

In this chapter, there were numerous algorithms and methods used in the research of the papers, journals, articles and books written by theorists and authors. It can be seen that, most of the algorithms and methods for instance Dijkstra and linear programming were modified to suit the problem of their study. Some of the abstracts discussed in the literature review have similarities in that, they describe vertices as nodes, weight or cost as distances, used nonnegative weights and weighted digraphs. The abstracts also have the common problem of the single source shortest path problems in networks that is related to this study.

CHAPTER 3

METHODOLOGY

This chapter discusses the shortest path problem in graph theory, the shortest path variations and algorithms solving the shortest path problems with illustrations. It also explains the linear programming approach to shortest path problems.

3.1 Graph Theory

In graph theory, the shortest path problem is the problem of finding a path between two nodes in a graph such that the sum of the weights of its constituent edges is minimized. This is analogous to the problem of finding the shortest path between two intersections on a road map. The graph's nodes represent intersections while the edges correspond to road segments, each weighted by the length of its road segment.

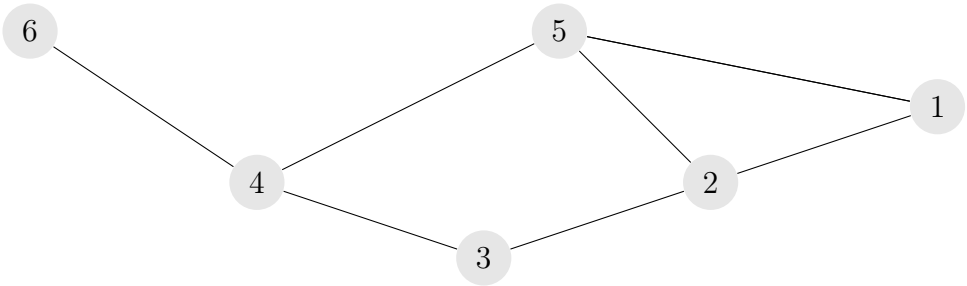


Figure 3.1: An undirected graph

Figure 3.1 is an undirected graph where all edges are bidirectional with no weights. Now determining paths from node 6 to node 1,there are two paths in between them and these are: (6,4,5,1) and (6,4,3,2,1).

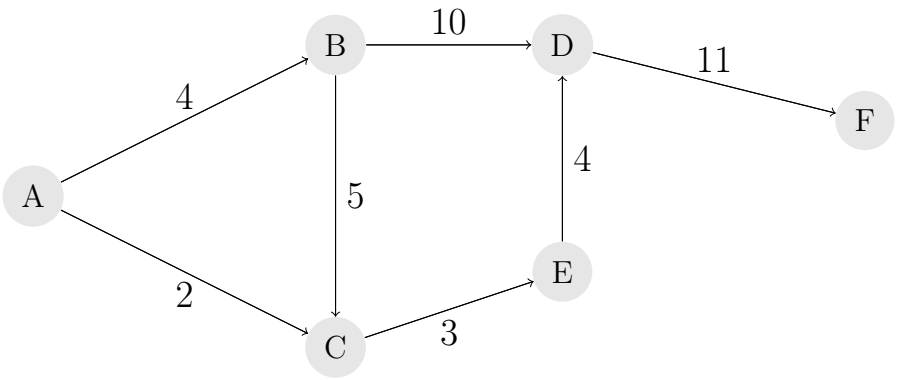


Figure 3.2: A directed graph

A directed graph (digraph), is a graph where the edges point in a direction. The shortest path between a pair vertex A and F in this graph is (A→C→E→D→F).

The shortest path problem can be defined for these types of graphs, directed, undirected or even mixed. For directed graphs, when consecutive vertices are connected by an appropriate directed edge, it is required to be a path.

Thus for an undirected graph, a path is a sequence of vertices. $P = (V_1, V_2, ..., V_n) \in V \times V \times ... \times V$ such that V_i is adjacent to V_{i+1} for $1 \leq i < n$ and a path P is called a path of length n from V_1 to V_n. Note that V_i and V_j are variables; their numbering here relates to their position in the sequence and needs not to relate to any canonical labeling of the veritices in a graph.

Now let $e_{i,j}$ be the edge incident to both V_i and V_j. Given a real-valued weight function $f : E \longrightarrow R$ and an undirected (simple) graph G, the shortest path from V to V' is the path $P = (V_1, V_2, ..., V_n)$ where $(v_1 = V and V_n = V')$ that over all possible n minimize the sum $\sum_{i=1}^{n-1} f(e_i, i+1)$ where each in the graph has a unit weight or $f : E \longrightarrow 1$, this is equivalent to finding the path with fewest edges. Some applications of this in a networking or telecommunications, this P is sometimes called the min-delay path problem and usually tied with a widest path problem.

3.2 The Shortest Path Problem

It is essentially the problem of finding the shortest path between a pair of vertices in a given graph and normally it is assumed the graph is connected so that there is a path and the shortest one can be found. And often this problem is stated in terms of edge weighted graph.

Edge weighted graph: It is basically a graph G together with a function called α the weighted function which maps the edges of the graph to some numbers called natural numbers. In this particular application we will think of them as positive integers where N is the set of Natural numbers including 0. Any of these numbers get assigned to each of the edges.

α : E(G)$\rightarrow$ N.

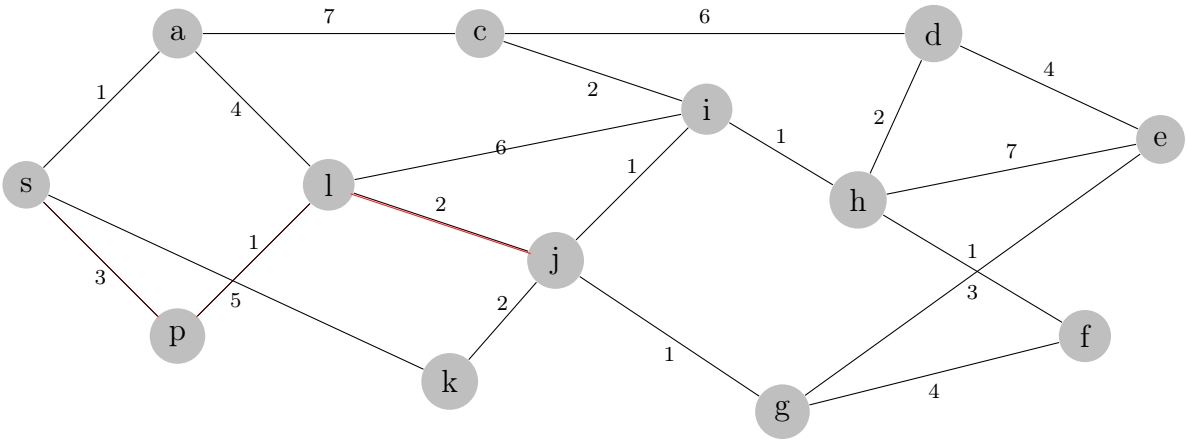


Figure 3.3: A graph network

This is a graph on quite a few number of vertices every one of the edges on the graph has been assigned with one of these natural numbers. For example, edge ac has a weight of 7. The practical aspect of having a weighted edge graph like this is that it can be thought of it as a transport network where maybe goods are delivered along roads, these weights could represent either the distance or the travel expenses of a particular road or the rate flow the amount of traffic one will encounter.

Now if the network of this graph actually represents communication network where each of these vertices is either a tower or a computer and these edges are given weights maybe based on the unreliability of the ability to pass on information between nodes.

The weight of a particular edge is going to be called the length having the weight function of a particular edge uv then $\alpha(uv)$ is called the length of edge uv. Now considering a certain path on the graph, called p, as shown in the network, the weight of this path is obtained by summing up the weight of individual edges. Thus α(p) is the edge weight of the path p defined as $\alpha(p) = \sum_{e \in p} \alpha(e)$ i.e. (sum of all of the weights of the edges where e is an edge in the path p). This can also be defined for any subgraph so notice that for any H of G, then the weight could be defined as

$$\left\{ \begin{array}{c} for\ any\ H \subseteq G \\ \\ \alpha(H) = \sum_{e \in H} \alpha(e) \end{array} \right\} in\ the\ general\ form. \tag{3.1}$$

In this particular example, it can be seen that α(p) = 3+1+2 = 6; and then α(p)is known as the distance of the path p. The point is that, for a practical application, the interest is in finding a path through a graph that has a minimum weight over the entire path.

Given a connected graph G with weight function α,

$\alpha : E(G) \longrightarrow N$, find for given u,v$\in$ V(G) find the minimum edge weight denoted as $d_G^{\alpha}(u, v) = min(\alpha(p)|p : u^* \longrightarrow V \quad)$

3.3 Variations of shortest path problem

3.3.1 One-to-all shortest path problem

A weighted network (V, E, C) with node set V, edge set E and the weight set C specifying weight C_{ij} for the edges $(i, j) \in E$. We are also given a starting node

$s \in V$. The one-to-all shortest path problem is the problem of determining the shortest path from say node s to all the other nodes in the network.

The diagram below is a weighted digraph which is an example of a demonstration.

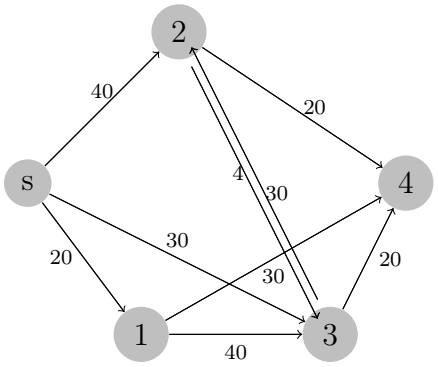


Figure 3.4: Illustration graph of one-to-all

The set of vertices are s,1,2,3 and 4 where s is the starting node. The set of edges are: (s,1),(s,2),(s,3),(1,3),(1,4)(2,3),(2,4),(3,2),(3,4). From node s to node 1 the weight is 20 units, s to 2 is 40 etc.

3.3.2 One-pair shortest path problem

This is a problem of finding the shortest path for a pair of vertices in a graph. Consider the graph below, the problem is to find the shortest path from vertex A to vertex D.

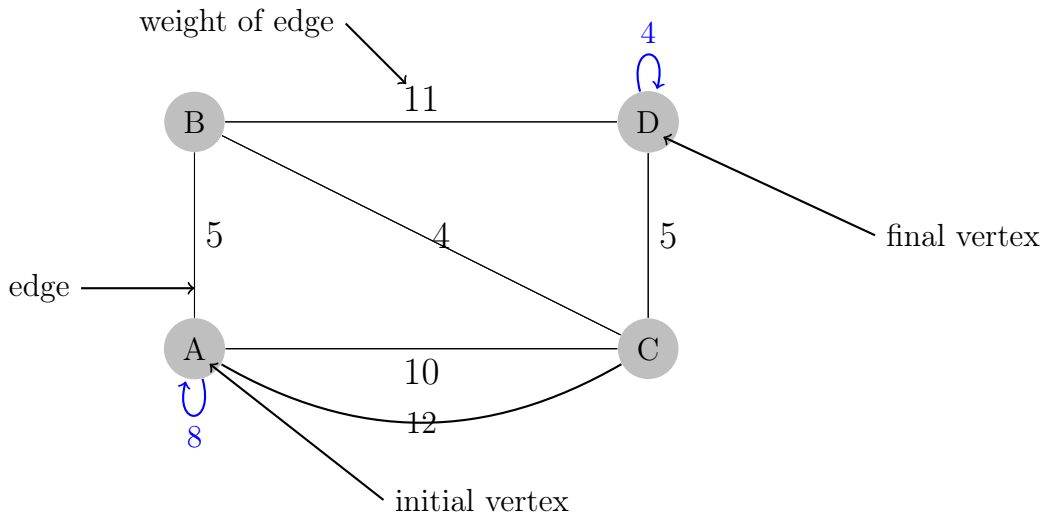
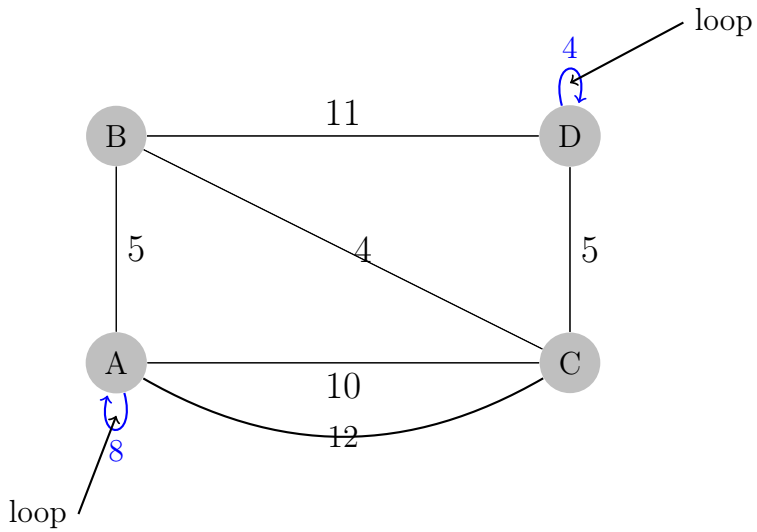


Figure 3.5: Graph with loops and parallel arcs

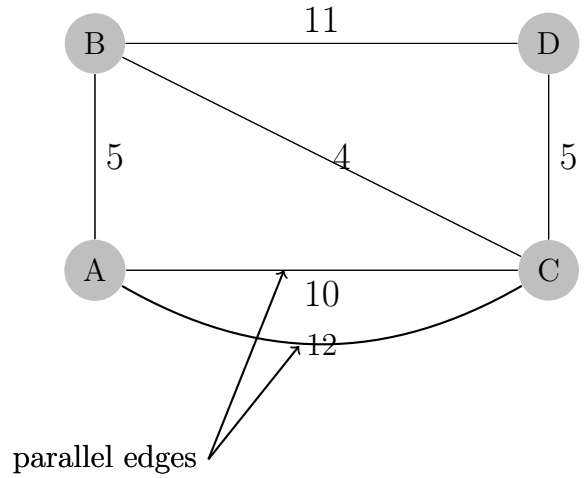
Fig.3.5 is a weighted undirected graph with loops and parallel edges. Before the shortest path can be found from A to D, these loops and parallel edges must be removed.

Step 1: Remove all loops. Any edge that starts and end at the same vertex is a loop.



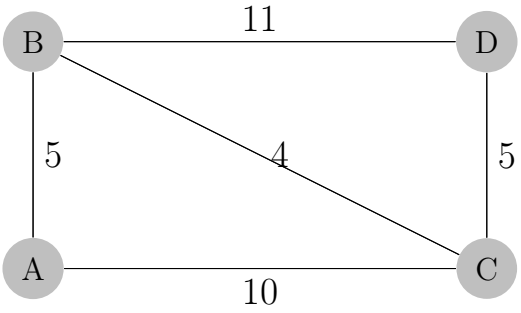
Vertices A and D have loops that have weights of 8 and 4 respectively. These loops must be gotten rid of as stated by the procedure

Step 2: Remove all parallel edges between two vertices except the one with least weight.



In this graph, vertex A and C are connected by two parallel edges having weight 10 and 12 respectively, so 12 being the largest is removed and 10 is kept.

Step 3: Now below is the required graph to work with.



Finding the shortest path from A to D, a shortest path table is created. As the graph has four vertices, hence the table will have four columns. Column name is the same as vertex name. Infinity has been assigned to the nodes B, C and D in the first row. The minimum value formula would be used in the calculation.

Consider two vertices X(Source Vertex)and Y(Destination Vertex)and an edge that directly connects them. Then the following formula is obtained: Min(DestValue, MarkedValue + EdgeWeight) DestValue = The value in the destination vertex (i.e., Y)column. MarkedValue = The value in the source vertex (i.e., X) column. EdgeWeight = The weight of the edge that connects the source (i.e., X) and the destination(i.e., Y) vertex.

Min(DestValue, MarkedValue + EdgeWeight); for e.g. If DestValue = 10, Marked value = 5 and Edge weight = 4 then Min(10, 5+4) = Min(10, 9) = 9. As 9 is smaller than 10.

Marked	A	B	C	D
A✓	0	∞	∞	∞
B✓	0	Min(∞, 0+5) 5	Min(∞, 0+10) 10	∞
C✓	0	5	Min(10, 5+4) 9	Min(∞, 5+11) 16
D✓	0	5	9	Min(16, 9+5) 14

Table 3.1: Summary of illustration of one-pair shortest path

Iteration 1: Now from the graph and table, Considering an edge between A and B, source vertex = A, Destination vertex = B. DestValue = ∞, Marked value = 0 and edge weight is 5.

Min(∞, 0+5) is written in column B. Solving Min(∞, 0+5) we get 5 as 5 is the smallest. Put 5 in column B.

Iteration 2: Now find the edge that directly connects vertex A and vertex C. The edge of weight is 10 that directly connects A and C.

Considering an edge between A and C so, source vertex = A Destination vertex = C. DestValue = ∞, MarkedValue = 0 and edgeweight is 10. Thus Min(∞, 0+10)is written in column C. Solving, we get 10. From the graph, we find no such edge that directly connects vertex A and D so we write ∞ in the D column.

Iteration 3: Find the edge that directly connects vertex B and vertex C and that is 4. Condering an edge between B and C so, source vertex = B, destination vertex = C. Destvalue = 10, Markedvalue = 5, Edgeweight = 5. Solving,

Min(10, 5+4) so write min(10, 5+4) in the column of vertex C. Solving, we get 9

Iteration 4: Now find the edge that directly connects vertex B and vertex D. Considering an edge between B and D so, source vertex = B, destination vertex = D. DestValue = ∞, marked value = 5 and edgeweight = 11. We therefore write Min(∞, 5+11)in column D. Solving, we get 16. Now that third row is completely filled, we now find the smallest unmarked value in the third row. It can be seen from the table that, the smallest unmarked value in the third row is 9, so we mark it with square box.

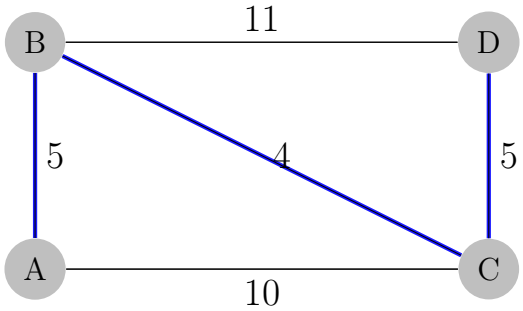
Iteration 5: Find the edge that directly connects C and D. Considering an edge between C and D, source vertex is C, destination vertex is D. Destination Value = 16, Marked Value = 9, edgeweight = 5 so we write Min (16, 9+5) in column D. Solving we get 14.

Now that the fourth row is filled we find the smallest unmarked value in the fourth row and mark it with a square box.

This implies that there are two or more columns with the same smallest unmarked value, then any can be chosen. This only denotes that there will be more than one shortest path in the graph.

As final vertex D is marked so we will stop here. Since column D has the marked value 14, it follows that the shortest path has the minimum weight of **14**.

By backtracking, the required shortest path which is **A** $\longrightarrow$ **B** $\longrightarrow$ **C** $\longrightarrow$ **D** has been marked with blue in the figure below.



3.3.3 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 the shortest path distances between every pair of nodes in the network is determined. Amponsah and Darkwah (2007).

3.4 Algorithms solving the shortest path problems

3.4.1 Floyd Warshall Algorithm

Floyd Warshall's algorithm is a graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights. The algorithm in its recognized form was propounded by the three computer scientists Robert Floyd, Bernard Roy and Stephen Warshall in 1962. It is an all-pairs shortest path algorithm for weighted graphs. The algorithm compares all possible paths through the graph between each pair of vertices.

Consider the graph G with vertices V numbered 1 through N; then consider a function shortest Path(i,j,k) that returns the shortest possible path from i to j. If w(i,j) is the weight of the edge between vertices i and j, the shortest path (i,j,k+1) can be defined in terms of the following formula.

Shortest Path(i,j,k+1)=min(shortest path (i,j,k)),shortest path(i,k+1,k)+shortest path(k+1,j,k). It works by first computing shortest path (i,j,k) for all pairs for k=1, then k=2 e.t.c. This process continues until the shortest path for all (i,j) pairs have been found.
The following illustrations depict Floyd-Warshall's Algorithm. Consider the weighted graph below

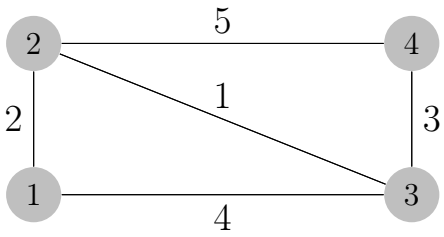


Figure 3.6: Illustration graph of Floyd Warshall's

Now two tables Distance Table(D) and Sequence Table(S) are drawn. They can also be referred to as matrix. The D will hold distances between any two vertices, whereas the S holds the name of the vertices that helps in finding the shortest path between any two vertices.

If there are N vertices in a graph then N-1 iterations are observed. Also, for a graph having N nodes, the Distance and Sequence tables will also have N of rows and columns. In the Fig:3.6, the nodes are 4 hence 3 iterations are required.

Notations k = Iteration number D_k = Distance table in kth iteration S_k = Sequence table in kth iteration d_{ij} = The distance between vertices i and j. i = row number j= column number and k= iteration number.

Put '-' in cells having the same row and column name.
Now Fill the empty cells with the weight of the edges that directly connects each pair of vertices.

Fill in the cells with the following	**Fill in the cells in the columns with the following**
c_{12} *and* c_{21} *with* 2	*column* 2 *with* 2
c_{13} *and* c_{31} *with* 4	*column* 3 *with* 3
c_{14} *and* c_{41} *with* ∞	*column* 4 *with* 4
c_{23} *and* c_{32} *with* 1	
c_{24} *and* c_{42} *with* 5	
c_{32} *and* c_{23} *with* 1	
c_{43} *and* c_{34} *with* 3	

Iteration 0: k=0

D_0	1	2	3	4
1	–	2	4	∞
2	2	–	1	5
3	4	1	–	3
4	∞	5	3	–

Table 3.2: Distance Table

S_0	1	2	3	4
1	–	2	4	∞
2	1	–	3	4
3	1	2	–	4
4	1	2	3	–

Table 3.3: Sequence Table

Put '-' in cells having the same row and column name. Now Fill the empty cells with the weights of the edges that directly connect each pair of vertices.

Copy the contents of row 1 and column 1 of table D_0 in D_1 and S_0 in S_1

Now fill the cells c_{ij} in distance table D_k using the following conditions. Is $d_{ij} > d_{ik} + d_{kj}$ (in distance table D_{k-1})? If Yes, then fill the cell c_{ij} in D_k table with the value $d_{ik} + d_{kj}$ of D_{k-1} table. If No then fill the cell c_{ij} in D_k table with the value d_{ij} of D_{k-1} table where Note that, we always fill

the cell c_{ij} in D_k table with the smallest number.

For c_{23},	*For* c_{24},	*For* c_{34}
$i = 2, j = 3, k = 1$	$i = 2, j = 4, k = 1$	$i = 3, j = 4, k = 1$
So from D_0 *table*	*So from* D_0 *table*	*So from* D_0 *table*
$d_{ij} = d_{23} = 1$	$d_{ij} = d_{24} = 5$	$d_{ij} = d_{34} = 3$
$d_{ik} = d_{21} = 2$	$d_{ik} = d_{21} = 2$	$d_{ik} = d_{31} = 4$
$d_{kj} = d_{13} = 4$	$d_{kj} = d_{14} = \infty$	$d_{kj} = d_{14} = \infty$
Is $d_{23} > d_{21} + d_{13}$	*Is* $d_{24} > d_{21} + d_{14}$	*Is* $d_{34} > d_{31} + d_{14}$
Is $1 > 2 + 4$	*Is* $5 > 2 + \infty$	*Is* $3 > 4 + \infty$
No	*No*	*No*
$\Rightarrow c_{23} = d_{23} = 1$	$\Rightarrow c_{24} = d_{24} = 5$	$\Rightarrow c_{34} = d_{34} = 3$
Hence fill c_{23} *and*	*Hence fill* c_{24} *and*	*Hence fill* c_{34} *and*
c_{32} *with* 1 *in* D_1	c_{42} *with* 5 *in* D_1	c_{43} *with* 3 D_1.

Iteration 1:k=1

D_1	1	2	3	4
1	–	2	4	∞
2	2	–	1	5
3	4	1	–	3
4	∞	5	3	–

S_1	1	2	3	4
1	–	2	4	∞
2	1	–	3	4
3	1	2	–	4
4	1	2	3	–

As k=2, Copy the contents of row-2 and column-2 of table D_1 in D_2 and S_1 in S_2.

For c_{13},	*For* c_{14},	*For* c_{43}
$i = 1, j = 3, k = 2$	$i = 1, j = 4, k = 2$	$i = 4, j = 3, k = 2$
So from D_1 *table*	*So from* D_1 *table*	*So from* D_0 *table*
$d_{ij} = d_{13} = 4$	$d_{ij} = d_{14} = \infty$	$d_{ij} = d_{43} = 3$
$d_{ik} = d_{12} = 2$	$d_{ik} = d_{12} = 2$	$d_{ik} = d_{42} = 5$
$d_{kj} = d_{23} = 1$	$d_{kj} = d_{24} = 5$	$d_{kj} = d_{23} = 1$
Is $d_{13} > d_{12} + d_{23}$	*Is* $d_{14} > d_{12} + d_{24}$	*Is* $d_{43} > d_{42} + d_{23}$
Is $4 > 2 + 1$	*Is* $5 > 2 + \infty$	*Is* $3 > 5 + 1$
Yes	*Yes*	*No*
$\Rightarrow c_{13} = 2 + 1 = 3$	$\Rightarrow c_{14} = 2 + 5$	$\Rightarrow c_{34} = d_{34} = 3$
Hence fill c_{13} *and*	*Hence fill* c_{14} *and*	*Hence fill* c_{34} *and*
c_{31} *with* 3 *in* D_2	c_{41} *with* 7 *in* D_2	c_{43} *with* 3 *in* D_2.

Iteration 2:k=2

D_2	1	2	3	4
1	–	2	3	7
2	2	–	1	5
3	3	1	–	3
4	7	5	3	–

S_2	1	2	3	4
1	–	2	2	2
2	1	–	3	4
3	2	2	–	4
4	2	2	3	–

As k=3, Copy the contents of row-3 and column-3 of table D_2 in D_3 and S_2 in S_3,

For c_{12},	*For* c_{14},	*For* c_{24}
$i = 1, j = 2, k = 3$	$i = 1, j = 4, k = 3$	$i = 2, j = 4, k = 3$
So from D_2 *table*,	*So from* D_2 *table*,	*So from* D_0 *table*
$d_{ij} = d_{12} = 2$	$d_{ij} = d_{14} = 7$	$d_{ij} = d_{24} = 5$
$d_{ik} = d_{13} = 3$	$d_{ik} = d_{13} = 3$	$d_{ik} = d_{23} = 1$
$d_{kj} = d_{32} = 1$	$d_{kj} = d_{43} = 3$	$d_{kj} = d_{34} = 3$
Is $d_{12} > d_{13} + d_{32}$?	*Is* $d_{14} > d_{13} + d_{43}$?	*Is* $d_{24} > d_{23} + d_{34}$?
Is $2 > 3 + 1$?	*Is* $5 > 2 + \infty$?	*Is* $5 > 1 + 3$?
No	*Yes*	*Yes*
$\Rightarrow c_{12} = d_{12} = 2$	$\Rightarrow c_{14} = d_{14} = 3 + 3$	$\Rightarrow c_{24} = d_{42} = 1 + 3$
Hence fill c_{12} *and*	*Hence fill* c_{14} *and*	*Hence fill* c_{24} *and*
c_{21} *with* 2 *in* D_3	c_{14} *with* 6 *in* D_3	c_{42} *with* 4 *in* D_3.

Iteration 3: k=3

D_3	1	2	3	4
1	–	2	3	6
2	2	–	1	4
3	3	1	–	3
4	6	4	3	–

S_3	1	2	3	4
1	–	2	2	3
2	1	–	3	3
3	2	2	–	4
4	3	3	3	–

To find the distance and shortest path between every two nodes, follow the summary table below.

Source Node(i)	Destination Node(j)	Distance(d_{ij}) in D_3	Sequence Path(S_{ij}) in S_3
1	2	2	1⟶2
1	3	3	$1 \longrightarrow 2 \longrightarrow 3$
1	4	6	$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$
2	3	1	$2 \longrightarrow 3$
2	4	4	$2 \longrightarrow 3 \longrightarrow 4$
3	4	3	$3 \longrightarrow 4$

Table 3.4: Summary of illustrations of Floyd Warshall's

3.4.2 Dijkstra's Algorithm

Dijkstra's algorithm, conceived by the Computer Scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs. (that is $C_{ij} \geq 0 \forall (i,j) \in E$)

In formal notation, intersection means vertex, road means edge and map means graph. For a given source vertex in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex. Although, Dijkstra originally only considered the shortest path between a given pair of nodes, it can also be used for finding costs of shortest path 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 vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, this shortest path algorithm is widely used in network routing protocols.

- Construct the network (or graph).
- Label the starting node with a 0 and put a box/circle around it.
- Look at each of the arcs connecting to the starting node and choose the one of least value. Write the value next to the node it is connecting to and draw a box around it.
- Temporarily label (number without boxes) all nodes connecting to the permanent labelled (boxed) nodes with their distances from the starting point.

- Choose the temporary label of least value and box it.
- Repeat steps 4 and 5 until the destination node has a permanent label that is the destination you are trying to reach.
- Retrace the shortest route backwards through the network back to your start node.

Consider the undirected graph of Figure 3.7 below.

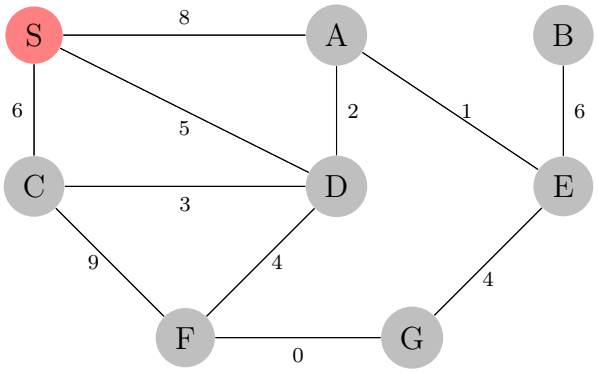
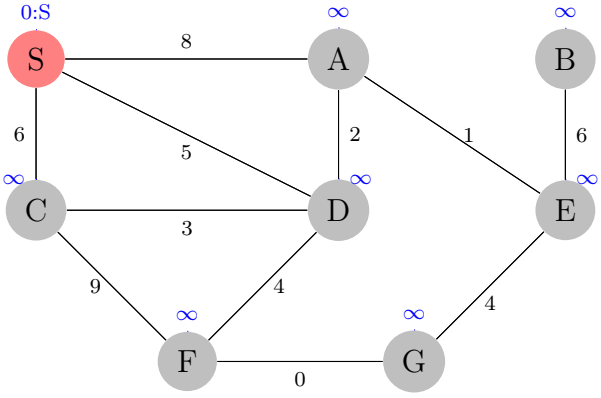
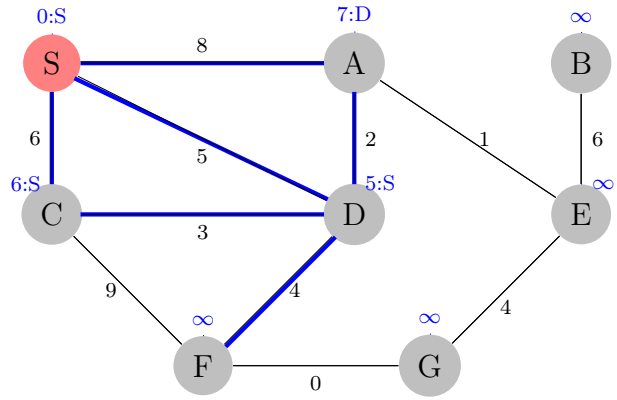


Figure 3.7: Illustration graph of Dijkstra's

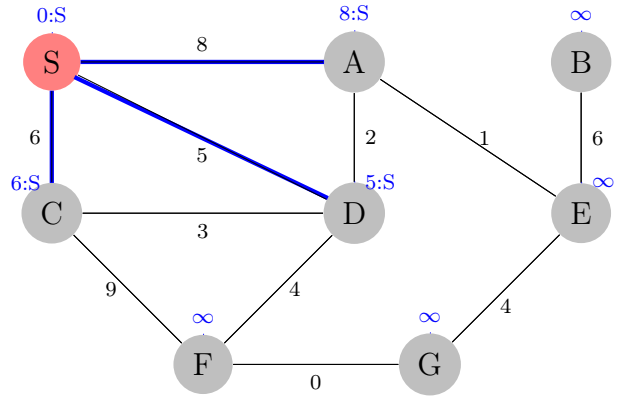
Here the problem is to find the shortest path from the source node S to every other vertex in the graph. Initialize the distance of the source to 0 and the distance to all other vertices to ∞.



Iteration 1: Starting with the source vertex S, relax edge SA, SC and SD. Relax edge SA a path to A with a distance of 8 coming from the source vertex S. Hence update A's weight from ∞ to 8. Relax edge SC a path to C with a distance of 6 coming from the source vertex S. Hence update C'S weight



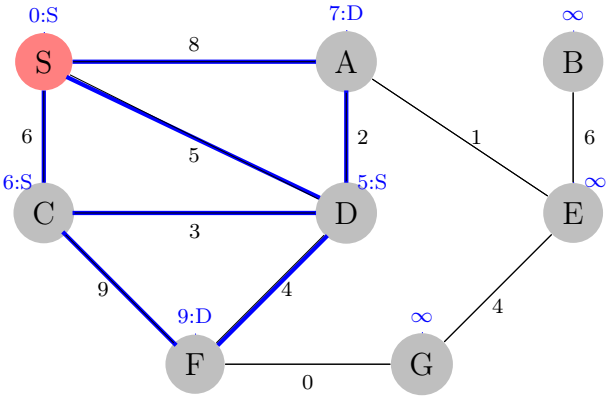
from ∞ to 6. Relax edge SD a path to D with a distance of 5 coming from the source vertex S. Hence update D's weight from ∞ to 5 and mark them in blue as shown.



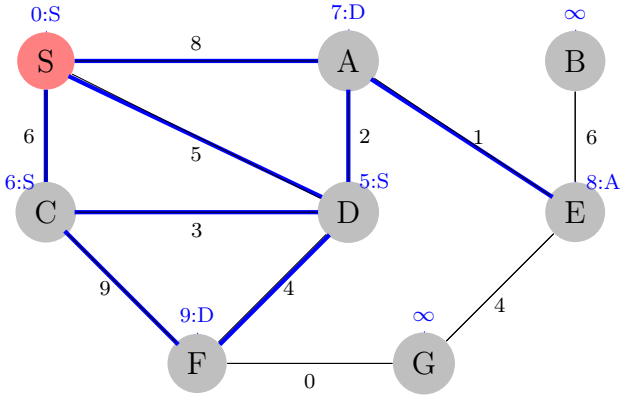
Iteration 2: Choosing the closest vertex D, relax edges DA, DC and DF. Relax edge DA a path to A with a distance of 2 from vertex D. Adding 2 to the current distance of D from the source of 5, there is a shorter distance 7 to A than the one existing one by passing through Vertex D. Hence update A's weight from 8 to 7. Relax edge DC a path to C with a distance of 3 coming from Vertex D. Hence update C's weight from ∞ to 3. Relax edge DF a path to F with a distance of 4 coming from Vertex D. Adding 4 to the current distance of D from the source of 5, we have 9. Hence update F's weight from ∞ to 9 and mark them with blue as shown.

Iteration 3 Choosing the next newest vertex C to the source, relax only edge CF since edges CA and CD have been relaxed already. Vertex C has a distance of 6 from the source and the remaining edge CF which gives a distance of 15 to F. This distance 15 is worse than the distance 9 vertex F already has

so we maintain the 9 and mark C as done.

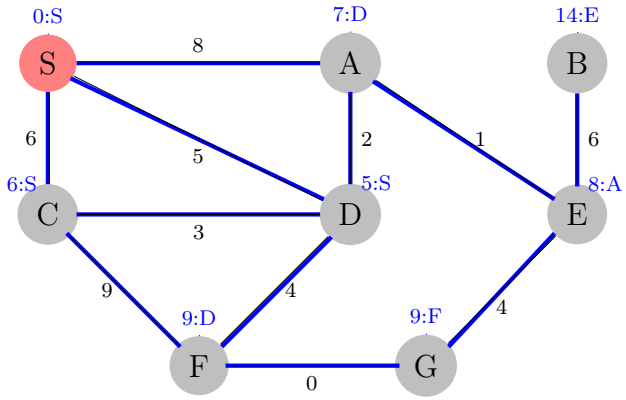


Iteration 4 Choosing the next vertex A to the source, relax only edge AE. Relax edge AE a path to E with a distance of 1 from vertex A. Adding 1 to the current distance of A of 7, find a shorter distance to E than the infinity. Hence update E's weight in the preceding vertex A from ∞ to 8.

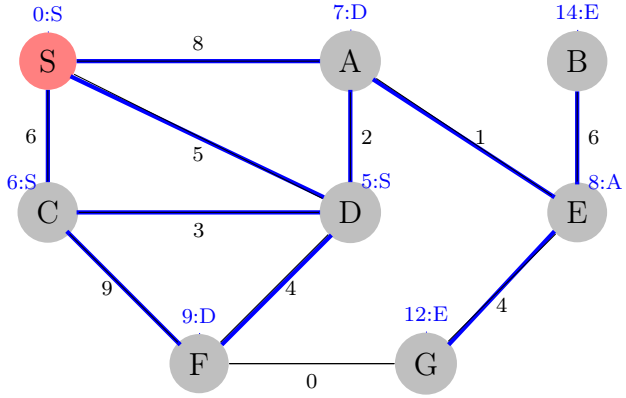


Iteration 5 Choosing the next newest vertex E, we relax edges EB and EG. Relax EB a path to B with a distance of 6 from vertex E. Adding 6 to the current distance of E from the source of 8 produces a shorter distance to A than the infinity. Hence update B's weight from ∞ to 14.

Now, it can be seen that, these predecessors' values can be used to trace the shortest path to any vertex. This is because B's predecessor is E as E's predecessor is A, A's predecessor is D and D's predecessor is S. Hence the backtracking gives: S $\longrightarrow$ D $\longrightarrow$ A $\longrightarrow$ E $\longrightarrow$ B.



Relax EG a path to G with a weight of 4 from vetex E. Adding 4 to the current distance of E from the source of 8 produces 12. Hence update G's weight from ∞ to 12, a shorter distance than ∞.



Iteration 6 Choosing the next newest vertex F, relax only edge FG. Relaxing FG gives a path to G that is three (3) less than the existing one. Hence update vertex G's weight from 12 to 9 coming from vertex F.

Iteration 7 Next, move to vertex G and B which both have no edges to relax. Thus the iterations end here and the shortest distances from the single-source S to every other vertex in the illustration graph are obtained and summarized in Table 3.5.

Source vertex	Destination vertex	Path	Total Distance
S	A	S→D→A	7
S	B	S→D→A→E→B	14
S	C	S→C	6
S	D	S→D	5
S	E	S→D→A→E	8
S	F	S→F	9
S	G	S→G	9

Table 3.5: Summary of illustrations of Dijkstra's

3.4.3 Bellman-Ford Algorithm

Bellman-Ford algorithm is a single source shortest path algorithm which also works on negative edge weights. While Dijkstra's algorithm assumes all edges have nonnegative weights, Bellman-Ford algorithm solves the single-source shortest path problem in general case in which edge weights can be negative.
Given (G,w,s) a function of the directed weighted graph below, we initialize all the vertices to ∞ and set the source vertex to zero and relax the edges accordingly.

Bellman-Ford (G,w,s)

$Initialize\ single-source(G,s)$	$Initialize\ Single-Source\ (G,s)$
$fori=1to$	$for\ each\ vertex\ v \in G.V$
$for\ each\ edge\ (u,v) \in G.E$	$v.d=\infty$
$Relax\ (u,v,w)$	$v.\pi = nil$
$for\ each\ edge\ (u,v) \in G.E$	$source.d=0$
$ifv.d > u.d + w(u,v)$	$Relaxation\ for\ each\ adjacent\ vertex$
$return\ false$	$Relax(u,v,w)for$
$(report\ that\ a\ negative-weight\ cycle\ exists)$	$each\ vertex\ v \in G.Adj[u]$
$return\ True$	$ifv.d > u.d + w(u,v)$
$v.d = u.d + w(u,v)$	$v.\pi = u$

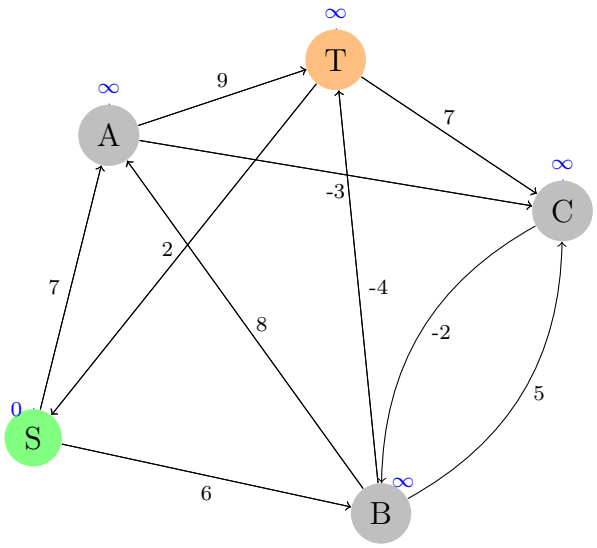


Figure 3.8: Illustration graph with negative weights

Iteration 1: Starting from node S, relax edges SA and SB and update A from ∞ to 7 and B from ∞ to 6 since $v.d > u.d + w(u,v)$ and mark A and B with their predecessor node S.

Iteration 2: Choosing node B, relax edges BT and BC and update C to 5+6=11

and T to 6-4= 2. Mark C and T with their new predecessor node B.

Iteration 3: Choosing node A, relax edge AC and update C from 11 to 7-3=4 and mark C with its new predecessor node A. Node T is not updated here since 7+9=16 is more than 2 it already obtained from B.

Iteration 4: Choosing node C, relax again edge CB because, the idea is to obtain the minimal value. Hence update B from 6 to 4-2=2 and write down B's newest predecessor C.

Iteration 5: Now need to relax and update edge BT from 2 to 2-4=-2.

Therefore, the shortest path and distance from S through all other nodes in the graph has been obtained. **S⟶A⟶C⟶B⟶T** and the shortest distance is **-2**.

	S	A	B	C	T
$d[v]$	0	∞,7	∞,6,2	∞,11,4	∞,2,-2
$\pi[v]$		S	S , C	B , A	B, B

Table 3.6: Summary of illustrations of Bellman-Ford's

3.5 Linear Programming (LP) Approach

Linear Programming is mathematical programming where the equations are linear equations in a set of variables that model a problem. It is the ultimate practical problem-solving model that encompasses shortest paths, network flows, minimum spanning trees(MST), assignments, transportation etc.

Also, it depicts theoretically and practically efficient techniques for solving large-scale problem and because of its efficiency was chosen to be the appropriate model for this study. Linear Programming model has three main components and they are:

1. Objective Function: These also describes a criterion that we wish to either minimize (e.g. cost or distance) or maximize (e.g. profit or gain)

2. Decision Variables: These describe the choices under the researcher's control.

3. Constraints: They describe the limitations that restrict the choices for decision variables.

3.5.1 Linear Programming in Single-Source Shortest Path Problem

The formulation is similar to that of the linear programming of shortest path problems. The objective is to find the shortest path from node s to all other nodes $t \in V\ s$.

$$\begin{aligned} &\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij}, \\ &\text{subject to} \\ &\sum_{(i,k)\in A} x_{i,k} - \sum_{(j,i)\in A} x_{j,i} = -1\ \forall\ i \neq s \\ &\sum_{(s,i)\in A} x_{si} = n-1, \\ &0 \leq x_{ij} \leq n-1,\ \forall\ (i,j) \in A \end{aligned} \tag{3.2}$$

It can be seen from the linear model below that, a flow is sent from s, which has supply n-1, to every other node, each with demand one (1) Hochbaum (2001).

3.5.2 Linear Programming Formulation

A linear programming formulation finds the shortest path from say node s to node t in a given graph.

Given a directed graph (V,E) with source node s, target node t, and cost or weight or distance c_{ij} for each edge (i,j) in E, consider the program with variable x_{ij}. $G = (V, E), c_{ij} : (i, j) \in E, c_{ij} \geq 0 \forall (i, j)$ where s and t are the distinguished nodes and V is the set of vertices or nodes and E the set of edges.

Let for each edge x_{ij} be the flow on edge (i,j). Flow balance in each node (Nonnegative weights)

$x_{ij} = 1$ if arc (i,j)$\in P$ or is in the shortest path and

$x_{ij} = 0$ otherwise.

$$\begin{aligned} &\text{Min} \sum_{(i,j)\in E} c_{ij}x_{ij}, \\ &\text{subject to} \sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij} = \begin{cases} -1, i = s \\ 0, i \neq s,t \ \forall \ i \in V \\ 1, i = t \end{cases} \end{aligned} \tag{3.3}$$

Primal Problem:

$$\begin{aligned} &\text{Min} \sum_{(i,j)\in A} c_{ij}x_{ij}, \\ &\text{subject to} \\ &\sum_j x_{sj} = 1 \\ &\sum_j x_{ij} - \sum_j x_{ji} = 0, i \neq s,t, \\ &-\sum_j x_{jt} = -1, \\ &x_{ij} \geq 0 \end{aligned} \tag{3.4}$$

A is the node arc incidence matrix

$$A = \begin{bmatrix} (i,j) \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

The column in the matrix has only two entries 1 and -1 and the dual constraints become the transpose of the matrix A. The i for 1 and j for -1 implies that, as

node i is enters node j leaves.

Dual Constraints:

$$\begin{aligned} &\text{Max } \pi_s - \pi_t, \\ &\text{subject to} \\ &\qquad \pi_i - \pi_j \leq c_{ij}, \forall (i,j) \in E,\ i,j \neq t \\ &\qquad \pi_i \geq 0,\ \pi_i \leq 0,\ \pi_t = 0 \end{aligned} \tag{3.5}$$

Since $c_{ij} \geq 0$, we can choose $\pi_i = 0\ \forall$ i as a starting dual-feasible solution, choose J as

$J = (i,j) : \pi_i - \pi_j = d_{ij}$, where J is the set of admissible arcs. Add up the primal constraints except the last one to obtain

$-\sum_j x_{jt} = 1$. This implies that the last constraint will always be satisfied by a feasible solution for (P).

In discussing the dual constraints for an LP, we can also elaborate some properties of the duality.

- LP dual has variables u_i for each node $i \in V$ and it can be interpreted as the length of path from node s to node i.
- LP dual constraints are difference constraints.
- Linear Programming is bounded below if and only if there is no negative length cycle.
- LP optimality conditions corresponds to the usual path optimality condition. In that, given

$$d(j) = \textit{label on node } j,$$

optimality means

$$d(j) - d(i) \leq \ length\ of\ (i,j)$$

and the solution

$$x_j = d(j).$$

$$\Rightarrow x_j - x_i = d(j) - d(i) \leq \ length\ of\ (i,j) = b_{ij}.$$

- We can also deduce that the LP formulation has the integralty property, i.e. 0-1 solutions naturally **?**.

Restricted Primal(RP):

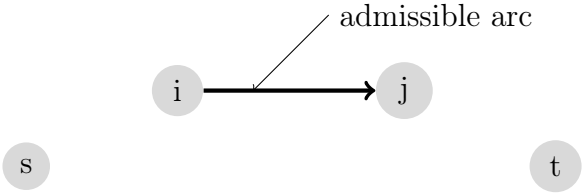
$$\begin{aligned} &\text{Min} \sum_{i \neq t} x_{ai} \\ &\text{subject to} \\ &\sum_j x_{sj} + x_{as} = 1 \\ &\sum_j x_{ij} - \sum_j x_{ji} + x_{ai} = 0, i \neq s, t \\ &x_{ij} \geq o, (i,j) \in J. \\ &x_{ij} = 0 (i,j) \notin J. \end{aligned} \tag{3.6}$$

Restricted Dual(DR):

$$\begin{aligned} &\text{Max } \bar{\pi}_s \\ &\text{subject to} \\ &\bar{\pi}_i - \bar{\pi}_j \leq 0 (i,j) \in J, \\ &\bar{\pi}_t = 0, \\ &\bar{\pi}_i \geq 0, \ \bar{\pi}_i \leq 0 \\ &\bar{\pi}_i \leq 1 \ \forall \ i, i \neq t \end{aligned} \tag{3.7}$$

One can find an optimal solution for DR without solving the RP. The idea is that all the restricted dual variables for DR must be less or equal to 1. So the maximum value can only be 1. Start with $\bar{\pi}_s = 1$ and all other $\bar{\pi}_i = 1$ if (s,i) is an admissible arc. But $\bar{\pi}_t = o, \bar{\pi}_i = 0$ if node t is reachable from node i via admissible arcs for all other nodes $\bar{\pi}_i = 1$. This is the feasible solution for DR. If t is reachable from j, $(i,j) \in J. \Rightarrow \bar{\pi}_i = 0 = \bar{\pi}_j \Rightarrow \bar{\pi}_i - \bar{\pi}_j = 0$ $(i,j) \in J$ and i is not reachable from s and t is not reachable from j.

The dual variable π_i for each node i corresponds to a distance label from node i to node t (thus $\pi_t = 0$) and the reduced cost c_{ij}^{π} for each edge (i,j) is $c_{ij} - \pi_i + \pi_j$. Given an initial feasible dual solution π, (we can use π because $c_{ij} \geq 0$ for each edge (i,j)) for all-1-dual, we first identify a set of admissible arcs, denoted by $\hat{A} = (i,j) \in E : c_{ij} \geq =$. Let $\hat{G} = (N, \hat{A})$ denote the admissible graph, which contains all the nodes in N but only arcs in $\hat{A}$. Wang (2008)



Here the admissible arcs are not enough to give a path from s to t.

Now modifying the dual solution gives,

$$
\begin{aligned}
&\pi + \theta\bar{\pi} \leq c, \theta > 0 \\
&(i,j) \in E, \pi_i - \pi_j + \theta(\bar{\pi}_i - \bar{\pi}_j) \leq c_{ij} \\
&\textit{As long as } \bar{\pi}_i - \bar{\pi}_j \leq 0, \textit{no need to worry since } \theta \textit{ has to be positive.} \\
&\textit{Now to compute } \theta \textit{ when } \bar{\pi}_i - \bar{\pi}_j > 0, \\
&\theta \leq \frac{c_{ij} - (\pi_i - \pi j)}{\bar{\pi}_i - \bar{\pi}_j}, \\
&\bar{\pi}_i - \bar{\pi}_j > 0
\end{aligned}
\tag{3.8}
$$

$$\theta = Min\frac{c_{ij} - (\pi_i - \pi j)}{\bar{\pi}_i - \bar{\pi}_j} = c_{lk} - (\pi_l - \pi_k)$$

$$\begin{aligned} &where \\ &\bar{\pi}_i - \bar{\pi}_j > 0 \\ &whenever\ this\ is\ positive\ it\ implies\ \bar{\pi}_i - \bar{\pi}_j\ is\ 1 \\ &\pi_l - \pi_k + \theta(\bar{\pi}_l - \bar{\pi}_k) = c_{lk} \\ &\Rightarrow (l,k) \in J.\ if\ an\ (i,j) \in J, \end{aligned} \tag{3.9}$$

then (i,j) will always continue on J. Then there is the need to update $\pi + \theta\bar{\pi}$.

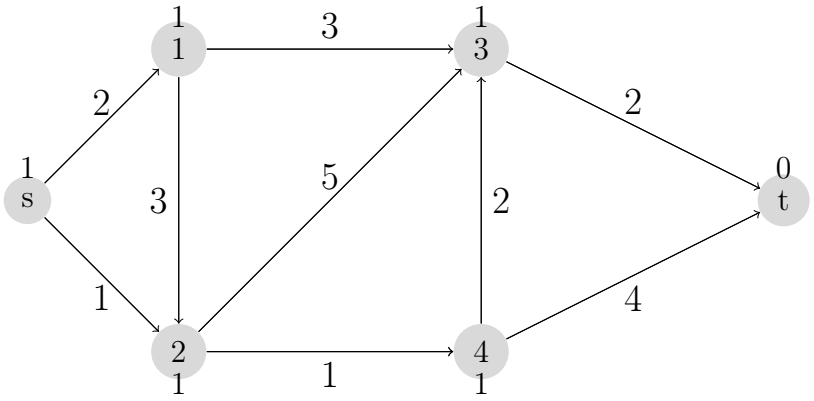


Figure 3.9: Illustration graph of LP method

Specifically, the problem can be viewed as if every node other than t sends one unit of flow to satisfy the demand (n-1) of node t in a way that minimizes the total transportation cost Wang (2008).

Restricted Dual solutions:

Let J be the set of admissible arcs, $\hat{V}$ be the set of admissible nodes, π be the initial feasible dual solution, $\bar{\pi}$ be the restricted dual solution and θ is the positive parameter that modifies the feasible dual solution. Also $\tilde{J}$ be the set of all possible arcs connecting the chosen node. Then the following iterations will be needed to find the shortest path of the above graph using linear programming.

Maximize π_s

subject to

$$\pi_s - \pi_1 \leq 2 \qquad \pi_2 - \pi_4 \leq 1$$

$$\pi_s - \pi_2 \leq 1 \qquad \pi_3 - \pi_t \leq 2$$

$$\pi_1 - \pi_3 \leq 3 \qquad \pi_2 - \pi_3 \leq 5$$

$$\pi_1 - \pi_2 \leq 3 \qquad \pi_4 - \pi_3 \leq 2$$

$$\pi_1 - \pi_3 \leq 3 \qquad \pi_4 - \pi_t \leq 4$$

$$\pi_t = 0,\ \pi_i \geq 0,\ \pi_i \leq 0.$$

Iteration 1: Dual: π =(0, 0, 0, 0, 0) J = $\{\emptyset\}$ $\hat{V} = \{\text{t}\}$

$\bar{\pi}$=(1,1,1,1,1) $\tilde{J}$=$\{(3,\text{t}), (4,\text{t})\}$ $\theta = \min(c_{3t}, c_{4t})$

$\Rightarrow\ \theta = \min(2, 4) = 2$

Iteration 2: Dual: $\pi = \pi + \theta\bar{\pi}$ (i.e. modified π), $\pi = (2, 2, 2, 2, 2)$ J = {3,t}

$\hat{V}$=$\{\text{t},3\}$ $\bar{\pi} = (1,\ 1,\ 1,\ 0,\ 1)$ $\tilde{J} = \{(1, 3), (2, 3), (4, 3)\}$

$\theta = min(c_{13}, c_{23}, c_{43}) \Rightarrow \theta = min(3, 5, 2) = 2$

Iteration 3: Dual: $\pi = \pi+\theta\bar{\pi}$ (i.e. modified π), $\pi = (4, 4, 4, 2, 4)$ J = $\{(3,\text{t}),(4,3)\}$

$\hat{V}$=$\{\text{t},3,4\}$ $\bar{\pi} = (1,\ 1,\ 1,\ 0,\ 0)$ $\tilde{J} = \{(1, 3), (2, 3), (2, 4)\}$

$\theta = min(c_{13}, c_{23}, c_{2,4}) \Rightarrow \theta = min(3, 5, -1) =$ -1

Iteration 4: Dual: $\pi = \pi + \theta\bar{\pi}$ (i.e. modified π), $\pi = (5, 5, 5, 2, 4)$ J =$\{(3,\text{t}),(4,3),(2,4)\}$

$\hat{V}$=$\{\text{t},3,4,2\}$ $\bar{\pi} = (1,\ 1,\ 0,\ 0,\ 0)$ $\tilde{J} = \{(s, 1), (s, 2), (1, 2)\}$

$\theta = min(c_{s2}, c_{12}) \Rightarrow \theta = min(1, 3) = 1$

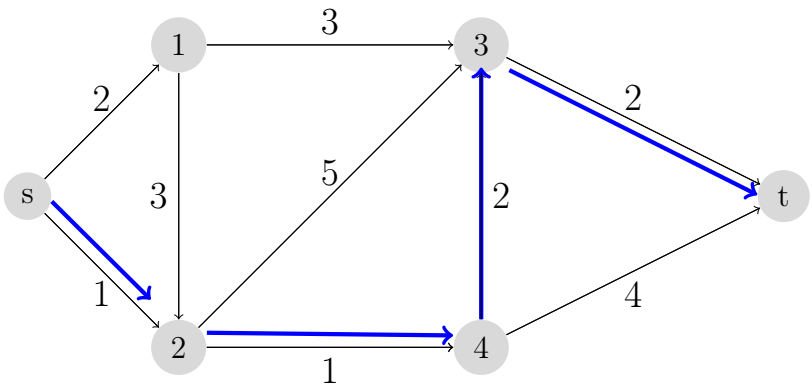


Figure 3.10: Summary graph of LP method

The iteration ends here since the new restricted dual solution becomes 0, and the last admissible arc is from node s to node 2. The optimal solution for the primal solution is obtained. The optimal path from s to t consists of $s \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow t$ and the total length will be $\mathbf{1} + \mathbf{1} + \mathbf{2} + \mathbf{2} = \mathbf{6}$. The restricted primal and restricted dual focus on building a path between a node and the destination node t.

3.6 Other Network Problems

3.6.1 Minimum Connector

The aim of minimum connector problems is to find the spanning tree of minimum weight. One practical problem could be that, suppose there is a collection of cities which need to be connected by a system of railway lines.

The question is how must the system be constructed in such a way that, the cost is minimal regardless of the passengers' inconveniences? The aim of minimum connector problems is to find the spanning tree of minimum weight. Two methods for solving this problem are Prim's algorithm and Kruskal's algorithm which both produce minimum spanning trees.

Lemma

Given a network (G,ϕ), the weight of a subgraph $H \subset G$ can be defined to be

$$\phi(H) = \sum_{e \in E(H)} \phi(e) \tag{3.10}$$

$$\phi(e) > 0.$$

The problem is that, given two vertices $u_0, v_0 \in V(G)$, find a $u_o v_0$-path of smallest weight. Here, a network of minimal cost is constructed.

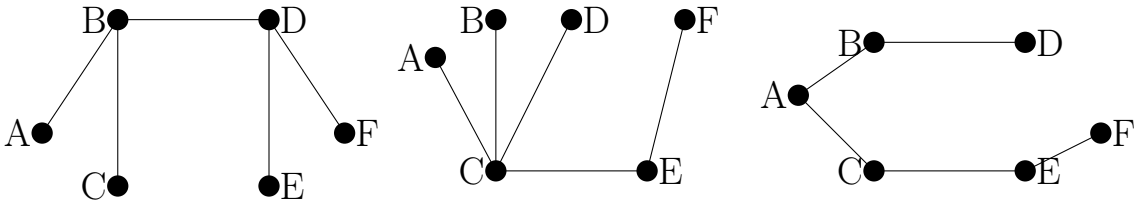
Observations

- The solution must be connected
- The solution should contain no cycles, for if there were any cycles, one of the edges could be removed to get a smaller cost. Thus every edge must be a bridge. Glickenstein (2012).

A tree is a connected undirected graph with no simple circuits or cycles. An undirected graph is a tree if and only there is a unique simple path between any of its vertices. A spanning tree is a subgraph that contains all the vertices in the original graph and is also a tree. A graph may have many spanning trees. On connected graph G=(V,E), a spanning tree must have the following properties;

1. it must be a connected subgraph
2. it must be acyclic(i.e. no cycles formed)
3. it is a tree in which number of its edges is one less than the number of its vertices (i.e. $|E| = |V| - 1$) and last but not least,
4. it must contain all vertices of G

A graph can have several trees for example,



However, a tree that encloses at any edge forming a loop or cycle is not a spanning tree. Examples are:

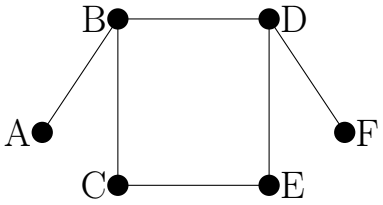


Figure 3.11: A tree with a loop

Figure 3.11 is not a spanning tree since the nodes BCED form a cycle or loop. Amponsah and Darkwah (2007).

3.6.2 Minimum Spanning Tree

This is the spanning tree of minimum cost for a given graph. Then it can be induced that, a minimum connector is also a spanning tree with minimum length or weight or cost. The number of edges in a minimum spanning tree is always one less than the number of vertices

Below is an example of a connected graph and one of its minimum spanning trees.

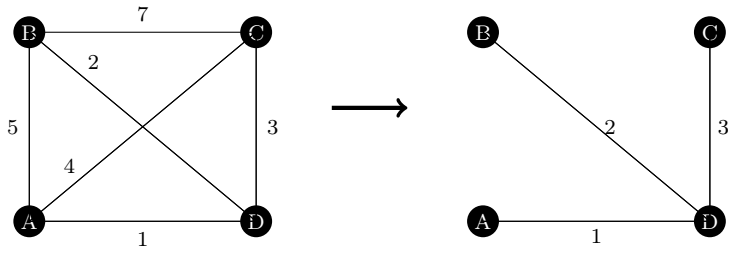


Figure 3.12: Complete graph and Minimum spanning tree

It can be seen that if we add the costs of the minimum spanning tree, we get 1+2+3=6 which will definitely be lesser than the overall total of the original graph.

Prim's Algorithm

Prim's algorithm finds the minimum spanning tree for a connected weighted undirected graph. This algorithm was independently developed by the computer scientist Robert C. Prim in 1957.

In Prim's algorithm, we are not concerned with distance from say node s to node t, the focus is on selecting the next smallest cost edge that connects to the current set of reachable nodes.

Matrix formulation of Prim's algorithm

Step 1: Label the column corresponding to the start vertex with a 1. Delete the row corresponding to that vertex.

Step 2: Ring the smallest available value in any labeled column.

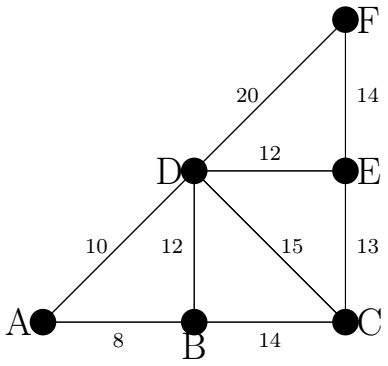
Step 3: Label the column corresponding to ringed vertex with a 2, etc. Delete the row corresponding to that vertex.

Step 4: Repeat steps 2 and 3 until all rows have been deleted.

Step 5: Write down the order in which edges were selected and the length of the minimum spanning tree.

Suppose a company is installing a system of cables to connect all towns in a region. The weights in the network below show the distances in kilometers. The question is, what is the least amount of cable needed? Amponsah and Darkwah (2007).

Iteration 1: Choosing a starting vertex say A. Delete the row A. Look for the smallest entry in column A.



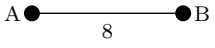
1

	A	B	C	D	E	F
~~A~~	~~∞~~	~~8~~	~~∞~~	~~10~~	~~∞~~	~~∞~~
B	(8)	∞	14	12	∞	∞
C	∞	14	∞	15	13	∞
D	10	12	15	∞	12	20
E	∞	∞	13	12	∞	14
F	∞	∞	∞	20	14	∞

Iteration 2: AB is the smallest edge joining A to the other vertices. Put edge AB into the solution. Detlete row B and look for the smallest entry in columns A and B.

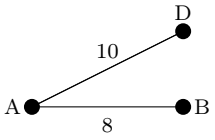
1 2

	A	B	C	D	E	F
~~B~~	~~8~~	~~∞~~	~~14~~	~~12~~	~~∞~~	~~∞~~
C	∞	14	∞	15	13	∞
D	(10)	12	15	∞	12	20
E	∞	∞	13	12	∞	14
F	∞	∞	∞	20	14	∞



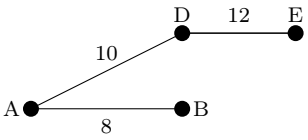
Iteration 3: AD is the smallest edge joining A and B to the other vertices. Put edge AD into the solution. Delete row D. Look for the smallest entry in columns A, B and D

	1 A	2 B	C	3 D	E	F
C	∞	14	∞	15	13	∞
~~D~~	~~10~~	~~12~~	~~15~~	~~∞~~	~~12~~	~~20~~
E	∞	∞	13	(12)	∞	14
F	∞	∞	∞	20	14	∞



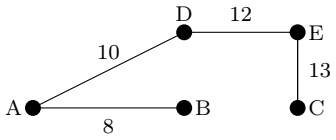
Iteration 4: DE is the smallest edge joining A, B and D to the other vertices. Put edge DE into the solution. Delete row E. Look for the smallest entry in the columns A, B, D and E.

	1 A	2 B	C	3 D	4 E	F
C	∞	14	∞	15	(13)	∞
~~E~~	~~∞~~	~~∞~~	~~13~~	~~12~~	~~∞~~	~~14~~
F	∞	∞	∞	20	14	∞



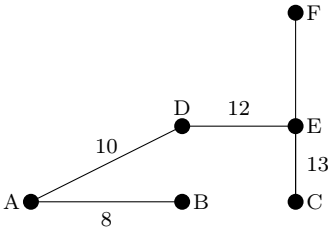
Iteration 5: EC is the smallest edge joining A, B, D and E to the other vertices. Put edge EC into the solution. Delete row C. Look for the smallest entry in the columns A, B, D, E and C.

	1	2	5	3	4	
	A	B	C	D	E	F
~~C~~	~~∞~~	~~14~~	~~∞~~	~~15~~	~~13~~	~~∞~~
F	∞	∞	∞	20	(14)	∞



Iteration 6: EF is the smallest edge joining A, B, D, E and C to the other vertices. Put EF into the solution.

	1	2	5	3	4	6
	A	B	C	D	E	F
~~F~~	~~∞~~	~~∞~~	~~∞~~	~~20~~	~~14~~	~~∞~~



Now all vertices have been connected into the spanning tree. The number of edges is five in the spanning tree which is one less than the number of vertices in the original diagram. The minimum length needed for the work would be 8+10+12+13+14 = 57km.

Kruskal algorithm

This algorithm seeks to find an edge of the least possible weight that connects any two trees in the forest. It was propounded by another computer scientist Joseph Kruskal in 1956.

Unlike Prim's algorithm which only grows one tree, Kruskal algorithm grows a collection of trees (a forest). And it continues until the forest merges to a single tree. The following steps illustrate Kruskal algorithm on a graph.

Step 1: Arrange all edges in a list(L) in a non decreasing order.

Step 2: Select edges from L, and include that in set T, avoid cycle.

Step 3: Repeat step 3. until T becomes a tree that covers all vertices

The solution to the cable problem that has already been stated is determined by using Kruskal's algorithm.

Edge(L)	Weight	MST Edge(T)	Weight
AB	8	AB	8
AD	10	AD	10
BC	14	–	–
CE	13	CE	13
DC	15	–	–
DE	12	DE	12
DF	20	–	–
EF	14	EF	14

Table 3.7: Summary of illustration of Kruskal's

$A \longrightarrow B = 8$ $\qquad$ $D \longrightarrow E = 12$

$A \longrightarrow D = 10$ $\qquad$ $C \longrightarrow E = 13$

$* B \longrightarrow D = 12$ $\qquad$ $* B \longrightarrow C = 14$

$D \longrightarrow C = 15$ $\qquad$ $E \longrightarrow F = 14$

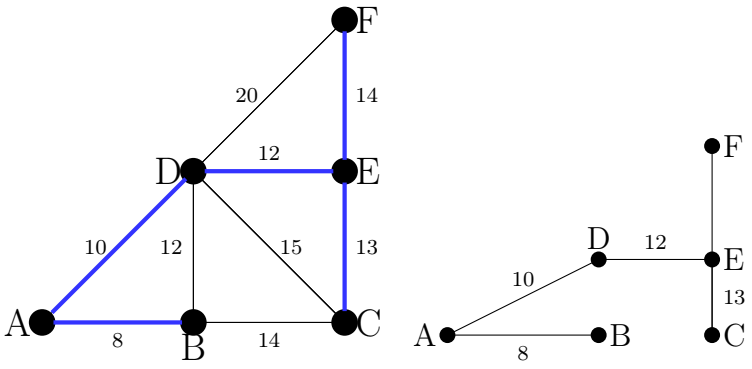
Note

BD will not be chosen since if forms a cycle with A and B.

BC will not be chosen since it also forms a cycle with A, B, D and E.

The weights in the last column are added, 8+10+12+13+14 = 57. Therefore the least amount of cable needed for the towns would be 57km.

It can be concluded that although the two algorithms gave the same solution, Kruskal's algorithm is faster than Prim's algorithm. Thus, the solution is formulated into the spanning tree below .



The edges that are in the solution are marked with blue.

3.7 Summary

This chapter discussed shortest path problems, algorithms, methods and other related network problems that seek to minimize travel cost. Each of the algorithms stated was demonstrated with graphs and iterations and through all that, the optimal solutions were obtained.

CHAPTER 4

DATA COLLECTION, ANALYSIS AND RESULTS

This chapter describes the collection, methods and presentation of analysis of data followed by a discussion of the research findings. The findings relate to research objectives that guided the study.

4.1 Data Collection

The distances between towns and the topographical map of the district were collected from the District Planning and Co-ordinating Unit Office of the Atwima Nwabiagya District Assembly.

Table 4.1: Summary of Lengths and condition of Road Types in the District

Road Type	Length (km)	Good		Fair		Bad	
		km	%	km	%	km	%
Bitumen Roads	109.50	56.90	94.52	24.50	3.66	28.10	1.83
Gravelled Roads	157.00	23.70	41.80	37.20	29.14	96.10	29.10
Total	266.50	80.60	67.29	61.70	16.80	124.20	17.56

Source: Survey by DPCU, 2014

Table 4.1 is the summary of the road type and conditions and the total length of each type in the district. There are two types and these are: bitumen(tarred roads) and gravelled(un tarred roads).

The total length(in km) of bitumen roads is 109.50 whereas that of gravelled roads is 157.0 and the total distance of both road types is 266.50 km. However, the percentage of roads that are good is about 67.29% while that of the bad roads

is about 17.56%.

Figure 4.1: Atwima Nwabiagya District Map

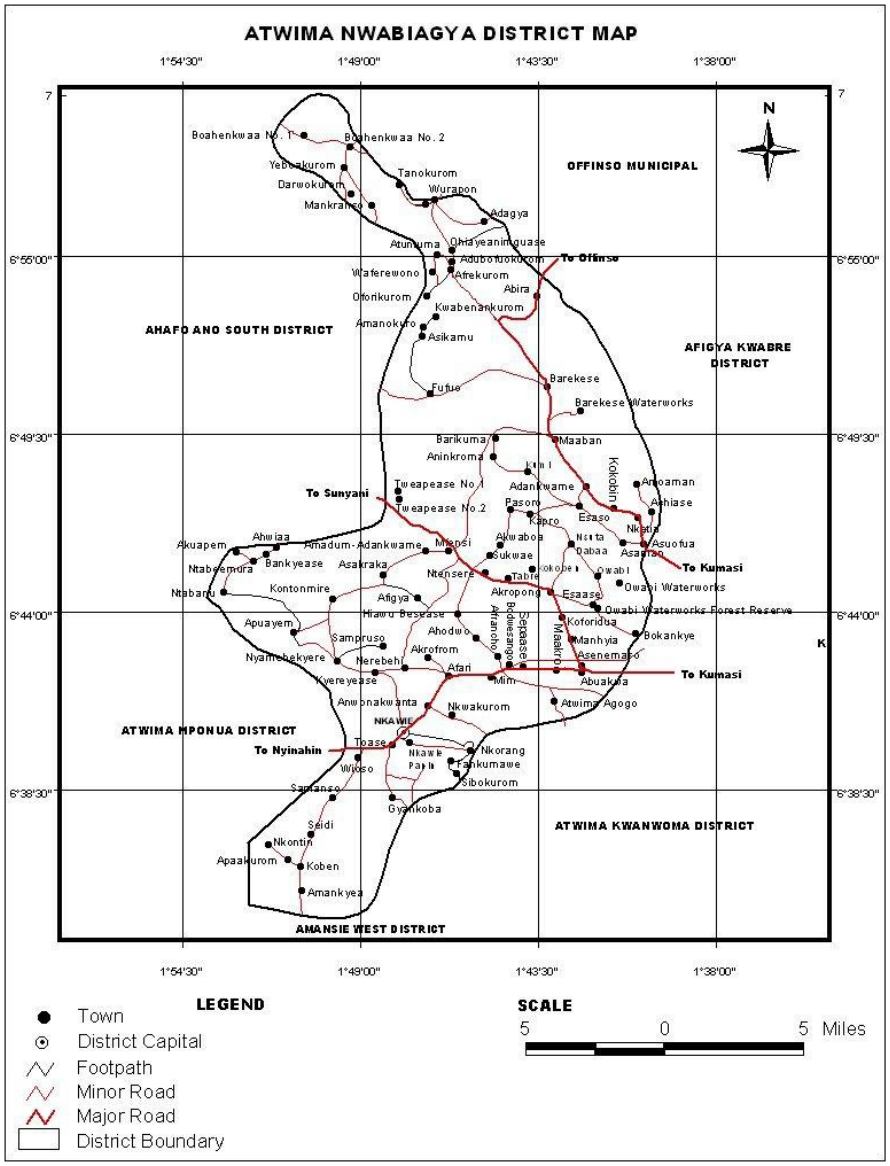


Figure 4.1 shows the topographical map of about 94 towns or communities.

4.1.1 Network System and Codes of Communities of the district

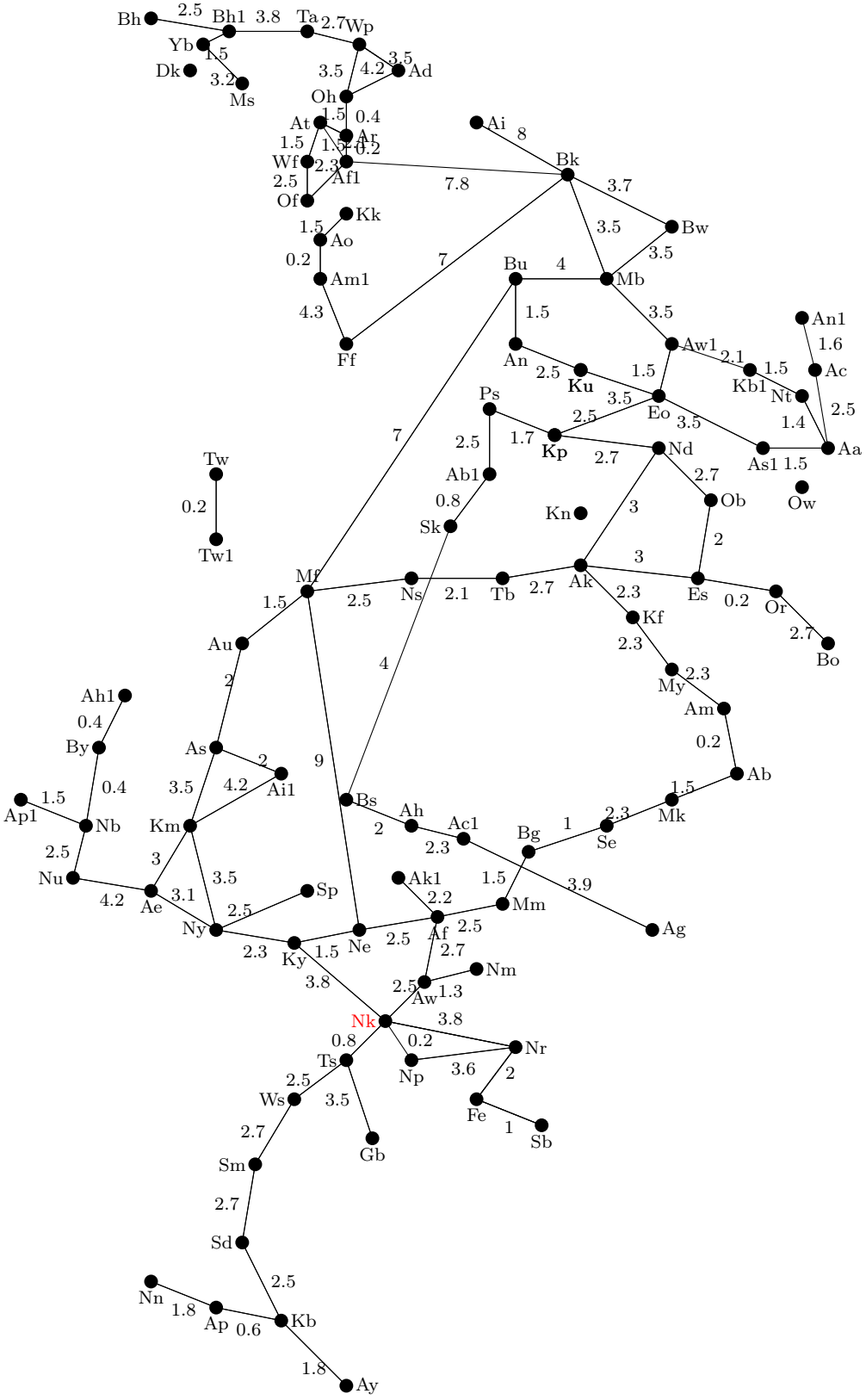


Figure 4.2: Network representation of Atwima Nwabiagya District

This network system was created from the district map. It is made up of nodes, arcs and distances; the nodes are the towns, the arcs are the paths between any two towns and the weight on them are the distances. The red node denotes the starting node that is Nkawie(Nk).

Table 4.2: Towns and their codes

(a) Table 1

Community	Code	Community	Code	Community	Code	Community	Code	Community	Code
Abira	Ai	Afigya	Ai1	Darwokurom	Dk	Maakro	Mk	Nyamebekyere	Ny
Abuakwa	Ab	Afrancho	Ac1	Esaase	Es	Manhyia	My	Oforikurom	Of
Achiase	Ac	Afrekurom	Af1	Esaso	Eo	Mankranso	Ms	Ohiayeanimguase	Oh
Adagya	Ad	Ahodwo	Ah	Fankumawe	Fe	Mfensi	Mf	Owabi	Ob
Adankwame	Aw1	Ahwiaa	Ah1	Fufuo	Ff	Mim	Mm	Owabi Forest Reserve	Or
Adankwame Amadum	Au	Akrofrom	Ak1	Gyankobaa	Gb	Nerebehi	Ne	Owabi Waterworks	Ow
Adubofourkrom	Ar	Akropong	Ak	Kapro	Kp	Nkawie	Nk	Pasoro	Ps
Afari	Af	Akuapem	Ap1	Koben	Kb	Nkawie Panin	Np	Samanso	Sm
Sukwae	Sk	Sibokrom	Sb	Tanokurom	Ta	Tabre	Tb	Tweapease No.1	Tw
Wurapon	Wp	Wioso	Ws						

(b) Table 1

Community	Code	Community	Code	Community	Code	Community	Code	Community	Code
Akwaboa	Ab1	Asakraka	As	Barekese	Bk	Koforidua	Kf	Nketia	Nt
Amankyea	Ay	Asaman	As1	Barekese Waterworks	Bw	Kokoben	Kn	Nkontin	Nn
Amanokuro	Ao	Asenemaso	Am	Barikuma	Bu	Kokobin	Kb1	Nkoran	Nr
Amoaman	An1	Asikamu	Am1	Hiawu Besease	Bs	Kontomire	Km	Nkwakom	Nm
Aninkroma	An	Asuofua	Aa	Boahenkwa No.1	Bh	Kumi	Ku	Nsuta Daabaa	Nd
Anwonankwanta	Aw	Atuntuma	At	Boahenkwa No.2	Bh1	Kwabenankurom	Kk	Ntabaemura	Nb
Apaakurom	Ap	Atwima Agogo	Ag	Bodwesango	Bg	Kyereyease	Ky	Ntabanu	Nu
Apuayem	Ae	Bankyease	By	Bokankye	Bo	Maaban	Mb	Ntensere	Ns
Sampruso	Sp	Seidi	Sd	Sepaase	Se	Toase	Ts	Tweapease No.2	Tw1
Waferewano	Wf	Yeboakrom	Yb						

Table 4:2 shows the several communities within the district which has been put in chronological order. Each community has been assigned a code for easy

identification.

4.1.2 Data for the problem

Table 4.3: Distances between towns

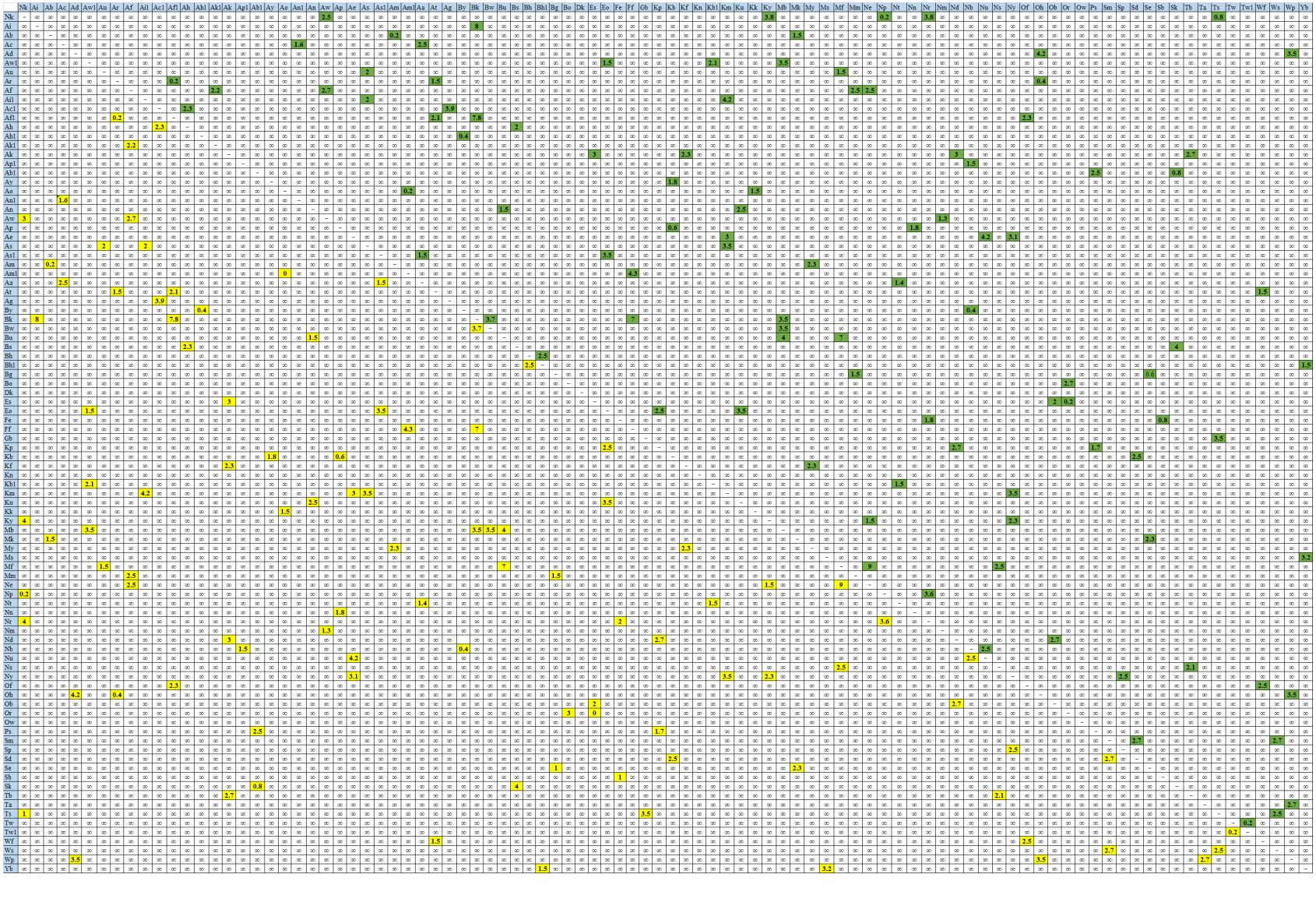
FROM	NK	NK	NK	NK	NK	NR	NR	AW	AW	TS	TS	WS
TO	NP	NR	AW	TS	KY	FE	NP	NM	AF	WS	GB	SM
DISTANCE(km)	0.2	3.8	2.5	0.8	3.8	2	3.6	1.3	2.7	2.5	3.5	2.7
FROM	BG	SE	AB	AB	MY	MY	AB1	AB1	KP	KP	KP	OB
TO	SE	MK	MK	AM	AM	KF	SK	PS	PS	ND	EO	ND
DISTANCE(km)	1	2.3	1.5	0.2	2.3	2.3	0.8	2.5	1.7	2.7	2.5	2.7
FROM	SM	SD	KB	KB	AP	FE	AF	AF	NE	NE	KY	NY
TO	SD	KB	AY	AP	NN	SB	MM	NE	KY	MF	NY	SP
DISTANCE(km)	2.7	2.5	1.8	0.6	1.8	1	2.5	2.5	1.5	9	2.3	2.5
FROM	OB	OR	EO	EO	EO	AA	AA	KB1	KB1	AC	AC	BU
TO	ES	BO	AS1	AW1	KU	AS1	NT	NT	AW1	AA	AN1	AN
DISTANCE(km)	2	2.7	3.5	1.5	3.5	1.5	1.4	1.5	2.1	2.5	1.6	1.5
FROM	NY	NY	AE	AE	NB	NB	NB	BY	KM	KM	AS	AS
TO	KM	AE	KM	NU	NU	AP1	BY	AH1	AS	AI1	AI1	AU
DISTANCE(km)	3.5	3.1	3	4.2	2.5	1.5	0.4	0.4	3.5	4.2	2	2
FROM	BU	MB	MB	MB	BK	BK	BK	BK	AF1	AF1	AF1	WF
TO	MB	BK	BW	AW1	BW	AI	AF1	FF	OF	AR	AT	OF
DISTANCE(km)	4	3.5	3.5	3.5	3.7	8	7.8	7	2.3	0.2	1.5	2.5
FROM	MF	MF	NS	NS	AK	AK	AK	AK	AH	BS	AH	
TO	AU	BU	MF	TB	TB	ND	KF	ES	BS	SK	AC1	
DISTANCE(km)	1.5	7	2.5	2.1	2.7	3	2.3	3	2	4	2.3	
FROM	WF	AT	OH	OH	OH	WP	BH	YB	YB	BG	AC1	
TO	AT	AR	AR	WP	AD	TA	BH1	BH1	MS	MM	AG	
DISTANCE(km)	1.5	1.5	0.4	3.5	4.2	2.7	2.5	1.5	3.2	1.5	3.9	

Source: Survey by DPCU, 2013

Table 4.3 shows the distances in kilometers from one town to another town.

Figure 4.3 shows the matrix representation of the from-to matrix with distances (in km) of the towns It is a (94x94) matrix of 94 rows and 94 columns starting from the source node. From the chart, the distance from Nkawie(Nk) to Kyereyaase (Ky) was recorded to be 3.8 km and Ahodwo(Ah) to Afrancho(Ac1), 2.3 km etc.

Figure 4.3: Matrix of towns and distances



4.1.3 Assumptions

1. All roads are assumed to be good.

2. The approach to the data given in this study is not by hypothesis testing but by observations and research questions.

3. Time bounds method is not used.

4.2 Methods and Presentation of Analysis

The LP Dual model in equation(3.4) discussed for SPP in chapter three would be applied here. The objective function of the SPP will be to minimize the distance traveled from the starting node to all other nodes.

4.2.1 Model for the Problem

$$\text{Max } \pi_t - \pi_s,$$

subject to

$$\pi_j - \pi_i \leq c_{ij}, \forall\ (i,j) \in E,\ i,j \neq t$$

$$\pi_i \geq 0,\ \pi_i \leq 0,\ \pi_s = 0$$

where,

π_s *is the dual variable for the starting node* s,

π_t *is the dual variable for the destination node* t,

π_i *is the dual variable corresponding to the constraint for node* i *in equation*(3.2),

π_j *is the dual variable corresponding to the constraint for node* j *in equation*(3.2) *and*

c_{ij} *is the cost ie the distance between node* i *and node* j

Thus in finding the shortest path from the single source s to the single destination t, the dual model will proceed to determine the shortest path and distances to

all the other destinations in the graph. Kangwalklai (2001).

4.2.2 Formulation of the Problem

4.2.3 Dual objective function and constraints

Max $TA - NK$

subject to

$$
\begin{array}{lll}
AW - NK \leq 2.5 & AK - KF \leq 2.3 & KP - PS \leq 1.7 \\
NR - NK \leq 3.8 & TB - AK \leq 2.7 & ND - KP \leq 2.7 \\
NP - NK \leq 0.2 & NS - TB \leq 2.1 & OB - ND \leq 2.7 \\
TS - NK \leq 0.8 & MF - NS \leq 2.5 & ND - AK \leq 3.0 \\
WS - TS \leq 2.5 & BU - MF \leq 7.0 & ES - OB \leq 2.0 \\
GB - TS \leq 3.5 & NY - KY \leq 2.3 & ES - AK \leq 2.3 \\
SM - WS \leq 2.7 & AE - NY \leq 3.1 & OR - ES \leq 0.2 \\
SD - SM \leq 2.7 & NU - AE \leq 4.2 & BO - OR \leq 2.7 \\
KB - SD \leq 2.5 & NB - NU \leq 2.5 & EO - KP \leq 2.5 \\
AY - KB \leq 1.8 & AP1 - NB \leq 1.5 & EO - KU \leq 3.5 \\
AP - KB \leq 0.6 & BY - NB \leq 0.4 & AN - KU \leq 2.5 \\
NN - AP \leq 1.8 & AH1 - BY \leq 0.4 & BU - AN \leq 1.5 \\
NR - NP \leq 3.6 & SP - NY \leq 2.5 & BU - MB \leq 4.0 \\
FE - NR \leq 2.0 & KM - NY \leq 3.5 & MB - BW \leq 3.5
\end{array}
$$

$SB - FE \leq 1.0$ $KM - AE \leq 3.0$ $AS1 - EO \leq 3.5$

$NM - AW \leq 1.3$ $AI1 - KM \leq 4.2$ $AA - AS1 \leq 1.5$

$KY - NK \leq 3.8$ $AS - KM \leq 3.5$ $NT - AA \leq 1.4$

$AF - AW \leq 2.7$ $AS - AI1 \leq 2.0$ $AC - AA \leq 2.5$

$NE - KY \leq 1.5$ $AU - AS \leq 2.0$ $AN1 - AC \leq 1.6$

$AF - NE \leq 2.5$ $MF - AU \leq 1.5$ $KB1 - NT \leq 1.5$

$MM - AF \leq 2.5$ $MF - NE \leq 9.0$ $AW1 - KB1 \leq 2.1$

$BG - MM \leq 1.5$ $EO - KP \leq 2.5$ $AW1 - EO \leq 1.5$

$SE - BG \leq 1.0$ $BS - AH \leq 2.0$ $MB - AW1 \leq 3.5$

$MK - SE \leq 2.3$ $AC1 - AH \leq 2.3$ $BK - MB \leq 3.5$

$AB - MK \leq 1.5$ $AG - AC1 \leq 3.9$ $BK - BW \leq 3.7$

$AM - AB \leq 0.2$ $SK - BS \leq 4.0$ $FF - BK \leq 7.0$

$MY - AM \leq 2.3$ $AB1 - SK \leq 0.8$ $AI - BK \leq 8.0$

$KF - MY \leq 2.3$ $PS - AB1 \leq 2.5$ $AF1 - BK \leq 7.8$

$AM1 - FF \leq 4.3$ $AT - AR \leq 2.1$ $WP - AD \leq 3.5$

$AO - AM1 \leq 0.2$ $WF - OF \leq 2.5$ $TA - WP \leq 2.5$

$KK - AO \leq 1.5$ $AT - WF \leq 1.5$ $BH1 - TA \leq 3.8$

$OF - AF1 \leq 2.3$ $OH - AR \leq 0.4$ $BH - BH1 \leq 2.5$

$AT - AF1 \leq 1.5$ $AD - OH \leq 4.2$ $YB - BH1 \leq 1.5$

$AR - AF1 \leq 0.2$ $WP - OH \leq 3.5$ $MS - YB \leq 3.2$

4.2.4 Computational procedure

The objective function and constraints were formulated using the dual LP model. The computer brand used for running the model was Toshiba that has 2.20 GHz, 750GB hard disk drive capacity and installed memory (RAM)of 4.00 GB(3.88 GB). Lindo 6.1 software was used to obtain the o.

4.3 Analysis of Findings

4.3.1 Interpretation of Results

The "LP optimum found at step 1" means that the optimal solution was found in iteration 1 of the initial tableau. Since the objective is to minimize costs from Nkawie(NK) to all other towns, iterations were done for quite a number of towns before the last town Tanokrom(TA).

Thus the optimum value of 51.70 km is the minimized distance from NK to TA while that of the other towns are all found in second column of the results in appendix A. The constraints are written based on the graph network of the district map.

4.3.2 Results and Findings

Table 4.4: Summary table of results

Source	Destination	Optimal Value/ Shortest Distance(km)	Shortest Path
NK	SB	6.8	NK→NR→FE→SB
NK	NR	6.1	NK→NR
NK	TS	0.8	NK→TS
NK	AB	14	NK→AW→AF→MM→BG→SE→MK→AB
NK	WS	3.3	NK→TS→WS
NK	GB	4.3	NK→TS→GB
NK	SM	6.0	NK→TS→WS→SM
NK	AK	21.1	NK→AW→AF→MM→BG→SE →MK→AB→AM→MY→KF→AK
NK	AN	22.3	NK→KY→NE→MF→BU→AN
NK	AY	13.0	NK→TS→WS→SM→SD→KB→AY
NK	FE	5.8	NK→NR→FE
NK	MF	14.3	NK→KY→NE→MF

Table 4.5: Summary table of results

Source	Destination	Optimal Value/ Shortest Distance(km)	Shortest Path
NK	SD	8.7	NK→TS→WS→SM→SD
NK	BU	21.3	NK→KY→NE→MF→BU
NK	MM	7.7	NK→AW→AF→MM
NK	BO	27.0	NK→AW→AF→MM→BG→SE→MK →AB→AM →MY→KF→AK→ES→OR→BO
NK	KM	9.6	NK→KY→NY→AE→KM
NK	AU	15.1	NK→KY→NY→KM→AS→AU
NK	NN	13.6	NK→TS→WS→SM→SD→KB→AP→NN
NK	KF	18.8	NK→AW→AF→MM→BG→SE →MK→AB→AM→MY→KF
NK	FF	35.8	NK→KY→NE→MF→BU→MB→BK→FF
NK	AP	11.8	NK→TS→WS→SM→SD→KB→AP
NK	KB	11.2	NK→TS→WS→SM→SD→KB
NK	AW1	28.8	NK→KY→NE→MF→BU→MB→AW1

Sample of the solutions in Appendix A have been selected and tabulated in Table 4.5 and 4.3. It was found out that, the distances from the source town NK to the immediate destinations such as NR and TS remained the same and were not minimized, since there are no intermediate nodes.

The shortest distance or optimal value from Nkawie(NK) to Akropong(AK) was found out to be 21.1km and the shortest or sequence path was NK→AW→AF→MM→BG→SE→MK→AB→AM→MY→KF→AK. Instead of taking the path NK→KY→NY→AE→KM→AS →AU→MF→NS→TB→AK that would yield a bigger value of 26.5km.

4.4 Summary

The optimal distances and paths have been determined. Unfortunately, there is no established work in the field of shortest distances and paths within the district and therefore there cannot be any form of comparison with the results obtained in Appendix A. The next chapter of the research discusses the conclusion and recommendations made to all stakeholders.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

This chapter discusses the conclusions and recommendations made based on the analysis in chapter four.

5.1 Conclusion

As discussed earlier the purpose for this study was to find an optimal distance and shortest path from Nkawie Fire Station to all towns of Atwima Nwabiagya District. In order to achieve this objective, a connected graph with distances between pairs of nodes was created out of the district map.

The LP dual model was the appropriate method applied to minimize the distances and to determine the shortest paths.The findings in this study provide a minimized travel-cost shown in the Appendices.

5.2 Recommendations

1. The office of the district fire service should employ the findings of this study to avert the shortest path problems they encounter in their operations.

2. Critical examination of this study is allowed for the purpose of further studies in the area of shortest path problems.

REFERENCES

Ahuja, R. K., Orlin, J. B., Pallottino, S., and Scutella, M. G. (2002). Dynamic shortest paths minimizing travel times and costs. *MIT Sloan working paper*.

Amponsah, S. and Darkwah, F. (2007). *Operations Research*.

Barnes, F., Chen, V., Gopalakrishnan, B., and Johnson, E. L. (2002). A least square primal-dual algorithm for solving linear programming problem. *Operations Research Letter*, 30.

Chen, B. Y., Lau, W. H., Sumalee, A., and lin Li, Z. (2012). Reliable shortest path finding in stochastic network with spatial correlated link travel time. *International Journal of Geographical Information Science*, 26(2).

Delling, D., Sanders, P., Schultes, D., and Wagner, D. (2009). Engineering route planning algorithm. *Algorithmic of large complex networks*.

Eilam-Tzoreff, T. (1998). The disjoint shortest paths problem. *Discrete applied mathematics*, 85(4):113–138.

El-Zohny, H. and El-Morsy, H. (2012). Shortest path algorithm for some graphs before and after folding. 75(2):141–148.

Fokuo, E. K. and Quaye-Ballard, J. A. (2013). Gis based fire emergency response system. vol. 2(issue. 1):32–40.

Frigioni, D., Marchetti-Spaccamela, A., and Nanni, U. (1996). Fully dynamic output bounded single-source shortest path problem. pages 212–221.

Gavriel, C., Hanasusanto, G., and Kuhn, D. (2012). Dept. of comut; imperial coll. london, london, uk. pages 2533–2538.

Glickenstein, D. (2012). Connector problem.

Guerrero, W. J., Velasco, N., Prodhon, C., and Amaya, C. (2013). On the generalized elementary shortest path problem: A heuristic approach. *International Network Optimization Conference INOC 2013*, 41.

Hochbaum, D. S. (2001). A new-old algorithm for minimum-cost and maximum-flow in closure graphs. *Networks*, 37(4).

Huguet, M.-J., Kirchler, D., Parent, P., and Calvo, R. W. (2013). Efficient algorithm for the 2-way multi modal shortest path problem. *Electronic Notes in Discrete Mathematics*, 41:431–437.

Kangwalklai, S. (2001). *Time dynamic label-constrained shortest path problem with application to TRANSIMS: A transportation planning system.* Virginia Tech.

Koch, R. and Nasrabadi, E. (2011). A duality theory for continuous time dynamic shortest paths with negative transit times.

Lim, Y. and Kim, H. (2005). A shortest path algorithm for real network based on path overlap. *Journal of the Eastern Asia Society for Transportation Studies*, 6:1426–1438.

Masuzawa, T. (2013). *Metzler Functions and the shortest path problem.* Number 2012-030. Keio/Kyoto Joint Global COE Program.

Naqi, A., Akhter, N., and Ali, N. (2010). Developing components of web gis for shortest path analysis "find shortest route": A geographical explanation for ssgc, pakistan. *Sindh University Research Journal*, 42.

Panahi, S. and Delavar, M. (2009). Dynamic shortest path in ambulance routing based on g i s. *International journal of Geoinformatics*, 5(1):13.

Peng, R. (2013). *Algorithm design using spectral graph theory.* Number 277.

Porazilova, A. (2005). The shortest path. *The proceedings on the XX. conference Geometry and Computer Graphics.*

Sinha, S. N. (2004). Graph cut algorithm in vision, graphics and machine learning. *UNC Chapel Hill*.

Stroetmann, K. (1997). *The constrained shortest path problem: A case study in using ASM's*, volume 3.

Sumit, K., Alok, M., and Raghvendra Singh, T. (2011). *A modified parallel Approach to single-source shortest path problem for massively dense graphs using CUDA*.

Szczesniak, I. (2000). Shortest path algorithms for public transport.

Wang, I.-L. (2008). *On solving shortest paths with Least -Square Primal-Dual Algorithm*, volume 25.

Wilson, D. B. and Zwick, U. (2013). A forward -backward single-source shortest paths algorithm. *Foundations of Computer Science (FOCS)*, pages 707–716.

Ying, X., Wang, X., and He, Y. (2013). Saddle vertex graph (svg): A novel solution to the discrete geodesic problem. *ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2013*, 32(170).

APPENDIX A

RESULTS

LP OPTIMUM FOUND AT STEP 1

OBJECTIVE FUNCTION VALUE

1) 51.70000

VARIABLE	VALUE	REDUCED COST
TA	51.700001	0.000000
NK	0.000000	0.000000
AW	2.500000	0.000000
NR	3.800000	0.000000
NP	0.200000	0.000000
TS	0.800000	0.000000
WS	3.300000	0.000000
GB	4.300000	0.000000
SM	6.000000	0.000000
SD	8.700000	0.000000
KB	11.200000	0.000000
AY	13.000000	0.000000
AP	11.800000	0.000000
NN	13.600000	0.000000
FE	5.800000	0.000000
SB	6.800000	0.000000
NM	3.800000	0.000000
KY	3.800000	0.000000
AF	5.200000	0.000000
NE	5.300000	0.000000
MM	7.700000	0.000000
BG	9.200000	0.000000
SE	10.200000	0.000000
MK	12.500000	0.000000
AB	14.000000	0.000000
AM	14.200000	0.000000
MY	16.500000	0.000000
KF	18.799999	0.000000
AK	21.100000	0.000000
TB	9.700000	0.000000
NS	11.800000	0.000000
MF	14.300000	0.000000
BU	21.299999	0.000000
NY	6.100000	0.000000
AE	6.600000	0.000000
NU	10.800000	0.000000
NB	13.300000	0.000000
AP1	14.800000	0.000000
BY	13.700000	0.000000
AH1	14.100000	0.000000
SP	8.600000	0.000000
KM	9.600000	0.000000
AI1	8.800000	0.000000
AS	10.800000	0.000000
AU	12.800000	0.000000
BS	17.299999	0.000000
AH	15.300000	0.000000
AC1	13.000000	0.000000
AG	16.900000	0.000000
SK	21.299999	0.000000
AB1	22.100000	0.000000
PS	24.600000	0.000000
KP	26.299999	0.000000
ND	24.100000	0.000000
OB	21.400000	0.000000
ES	23.400000	0.000000
OR	23.600000	0.000000
BO	26.299999	0.000000
EO	28.799999	0.000000
KU	25.299999	0.000000
AN	22.799999	0.000000
MB	33.799999	0.000000
BW	33.599998	0.000000
AS1	23.799999	0.000000
AA	25.299999	0.000000
NT	26.700001	0.000000
AC	27.799999	0.000000
AN1	29.400000	0.000000
KB1	28.200001	0.000000
AW1	30.299999	0.000000
BK	37.299999	0.000000
FF	44.299999	0.000000
AI	45.299999	0.000000
AF1	45.099998	0.000000

APPENDIX A

AM1	48.599998	0.000000
AO	48.799999	0.000000
KK	50.299999	0.000000
OF	47.400002	0.000000
AT	46.599998	0.000000
AR	45.299999	0.000000
WF	49.900002	0.000000
OH	45.700001	0.000000
AD	49.900002	0.000000
WP	49.200001	0.000000
BH1	55.500000	0.000000
BH	58.000000	0.000000
YB	57.000000	0.000000
MS	60.200001	0.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	0.000000	0.000000
3)	0.000000	0.000000
4)	0.000000	0.000000
5)	0.000000	0.000000
6)	0.000000	0.000000
7)	0.000000	0.000000
8)	0.000000	0.000000
9)	0.000000	0.000000
10)	0.000000	0.000000
11)	0.000000	0.000000
12)	0.000000	0.000000
13)	0.000000	0.000000
14)	7.200000	0.000000
15)	0.000000	0.000000
16)	0.000000	0.000000
17)	0.000000	0.000000
18)	0.000000	1.000000
19)	0.000000	0.000000
20)	0.000000	1.000000
21)	2.600000	0.000000
22)	0.000000	0.000000
23)	0.000000	0.000000
24)	0.000000	0.000000
25)	0.000000	0.000000
26)	0.000000	0.000000
27)	0.000000	0.000000
28)	0.000000	0.000000
29)	0.000000	0.000000
30)	0.000000	0.000000
31)	14.100000	0.000000
32)	0.000000	0.000000
33)	0.000000	0.000000
34)	0.000000	1.000000
35)	0.000000	0.000000
36)	2.600000	0.000000
37)	0.000000	0.000000
38)	0.000000	0.000000
39)	0.000000	0.000000
40)	0.000000	0.000000
41)	0.000000	0.000000
42)	0.000000	0.000000
43)	0.000000	0.000000
44)	0.000000	0.000000
45)	5.000000	0.000000
46)	2.300000	0.000000
47)	0.000000	0.000000
48)	0.000000	0.000000
49)	0.000000	0.000000
50)	0.000000	1.000000
51)	0.000000	0.000000
52)	0.000000	0.000000
53)	0.000000	0.000000
54)	0.000000	0.000000
55)	0.000000	0.000000
56)	0.000000	0.000000
57)	0.000000	0.000000
58)	4.900000	0.000000
59)	5.400000	0.000000
60)	0.000000	0.000000
61)	0.000000	0.000000
62)	0.000000	0.000000
63)	0.000000	0.000000
64)	0.000000	0.000000
65)	0.000000	0.000000
66)	0.000000	1.000000
67)	0.000000	1.000000
68)	0.000000	1.000000

APPENDIX B

69)	16.500000	0.000000
70)	3.300000	0.000000
71)	8.500000	0.000000
72)	0.000000	0.000000
73)	0.000000	0.000000
74)	0.000000	0.000000
75)	0.000000	0.000000
76)	0.000000	0.000000
77)	0.000000	0.000000
78)	0.000000	1.000000
79)	0.000000	1.000000
80)	0.000000	1.000000
81)	0.000000	0.000000
82)	0.000000	0.000000
83)	0.000000	0.000000
84)	0.000000	1.000000
85)	0.000000	0.000000
86)	0.000000	0.000000
87)	0.000000	0.000000
88)	0.000000	0.000000
89)	0.000000	0.000000
90)	0.000000	1.000000
91)	0.800000	0.000000
92)	0.000000	0.000000
93)	4.800000	0.000000
94)	0.000000	1.000000
95)	0.000000	0.000000
96)	0.000000	1.000000
97)	4.200000	0.000000
98)	0.000000	1.000000
99)	0.000000	0.000000
100)	0.000000	0.000000
101)	0.000000	0.000000
102)	0.000000	0.000000

NO. ITERATIONS= 1

STATISTICS AND RANGE(SENSITIVITY ANALYSIS)

STATIISTICS

ROWS= 102 VARS= 88 INTEGER VARS= 0(0 = 0/1) QCP= 0
NONZEROS= 305 CONSTRAINT NONZ= 202(202 = +-1) DENSITY=0.034
SMALLEST AND LARGEST ELEMENTS IN ABSOLUTE VALUE= 0.200000 9.00000
OBJ=MAX, NO. <,=,>: 101 0 0, GUBS <= 51 VUBS >= 0
SINGLE COLS= 15 REDUNDANT COLS= 0

SENSITIVITY ANALYSIS

RANGES IN WHICH THE BASIS IS UNCHANGED:

	OBJ COEFFICIENT RANGES		
VARIABLE	CURRENT COEF	ALLOWABLE INCREASE	ALLOWABLE DECREASE
TA	1.000000	0.000000	1.000000
NK	-1.000000	0.000000	INFINITY
AW	0.000000	0.000000	0.000000
NR	0.000000	0.000000	0.000000
NP	0.000000	0.000000	0.000000
TS	0.000000	0.000000	0.000000
WS	0.000000	0.000000	0.000000
GB	0.000000	0.000000	0.000000
SM	0.000000	0.000000	0.000000
SD	0.000000	0.000000	0.000000
KB	0.000000	0.000000	0.000000
AY	0.000000	0.000000	0.000000
AP	0.000000	0.000000	0.000000
NN	0.000000	0.000000	0.000000
FE	0.000000	0.000000	0.000000
SB	0.000000	0.000000	0.000000
NM	0.000000	0.000000	0.000000
KY	0.000000	0.000000	1.000000
AF	0.000000	0.000000	0.000000
NE	0.000000	0.000000	1.000000
MM	0.000000	0.000000	0.000000
BG	0.000000	0.000000	0.000000
SE	0.000000	0.000000	0.000000
MK	0.000000	0.000000	0.000000
AB	0.000000	0.000000	0.000000
AM	0.000000	0.000000	0.000000
MY	0.000000	0.000000	0.000000

APPENDIX B

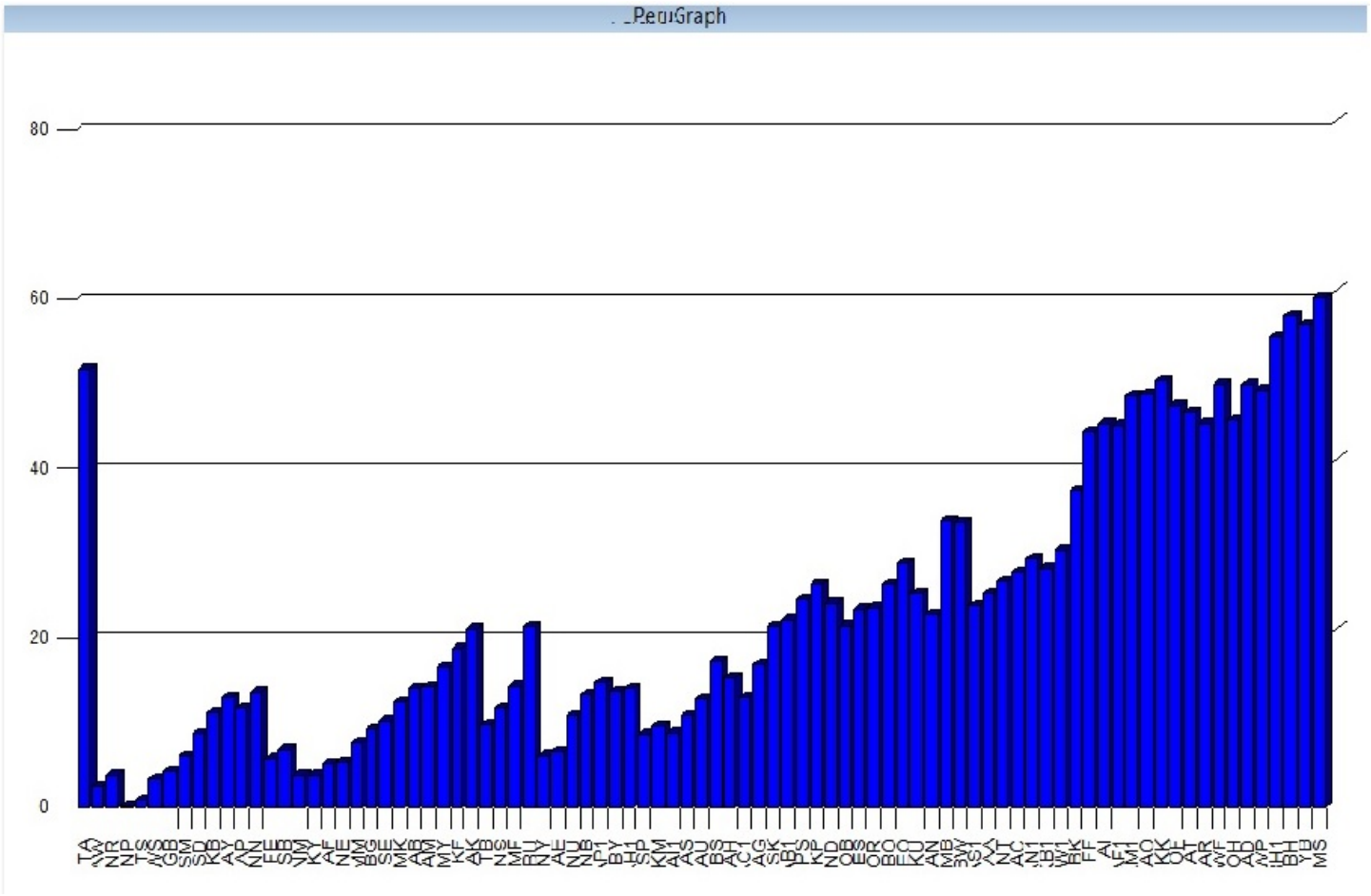
KF	0.000000	0.000000	0.000000
AK	0.000000	0.000000	0.000000
TB	0.000000	0.000000	1.000000
NS	0.000000	0.000000	1.000000
MF	0.000000	0.000000	1.000000
BU	0.000000	0.000000	1.000000
NY	0.000000	0.000000	0.000000
AE	0.000000	0.000000	0.000000
NU	0.000000	0.000000	0.000000
NB	0.000000	0.000000	0.000000
AP1	0.000000	0.000000	0.000000
BY	0.000000	0.000000	0.000000
AH1	0.000000	0.000000	0.000000
SP	0.000000	0.000000	0.000000
KM	0.000000	0.000000	0.000000
AI1	0.000000	0.000000	1.000000
AS	0.000000	0.000000	1.000000
AU	0.000000	0.000000	1.000000
BS	0.000000	0.000000	1.000000
AH	0.000000	0.000000	1.000000
AC1	0.000000	0.000000	1.000000
AG	0.000000	0.000000	0.000000
SK	0.000000	0.000000	1.000000
AB1	0.000000	0.000000	1.000000
PS	0.000000	0.000000	1.000000
KP	0.000000	0.000000	1.000000
ND	0.000000	0.000000	0.000000
OB	0.000000	0.000000	0.000000
ES	0.000000	0.000000	0.000000
OR	0.000000	0.000000	0.000000
BO	0.000000	0.000000	0.000000
EO	0.000000	0.000000	1.000000
KU	0.000000	0.000000	1.000000
AN	0.000000	0.000000	1.000000
MB	0.000000	0.000000	1.000000
BW	0.000000	0.000000	1.000000
AS1	0.000000	0.000000	1.000000
AA	0.000000	0.000000	1.000000
NT	0.000000	0.000000	1.000000
AC	0.000000	0.000000	0.000000
AN1	0.000000	0.000000	0.000000
KB1	0.000000	0.000000	1.000000
AW1	0.000000	0.000000	1.000000
BK	0.000000	0.000000	1.000000
FF	0.000000	0.000000	0.000000
AI	0.000000	0.000000	0.000000
AF1	0.000000	0.000000	1.000000
AM1	0.000000	0.000000	0.000000
AO	0.000000	0.000000	0.000000
KK	0.000000	0.000000	0.000000
OF	0.000000	0.000000	0.000000
AT	0.000000	0.000000	0.000000
AR	0.000000	0.000000	1.000000
WF	0.000000	0.000000	0.000000
OH	0.000000	0.000000	1.000000
AD	0.000000	0.000000	0.000000
WP	0.000000	0.000000	1.000000
BH1	0.000000	0.000000	0.000000
BH	0.000000	0.000000	0.000000
YB	0.000000	0.000000	0.000000
MS	0.000000	0.000000	0.000000

RIGHTHAND SIDE RANGES

ROW	CURRENT RHS	ALLOWABLE INCREASE	ALLOWABLE DECREASE
2	2.500000	2.600000	2.500000
3	3.800000	INFINITY	3.800000
4	0.200000	7.200000	0.200000
5	0.800000	INFINITY	0.800000
6	2.500000	INFINITY	3.300000
7	3.500000	INFINITY	4.300000
8	2.700000	INFINITY	6.000000
9	2.700000	INFINITY	8.700000
10	2.500000	INFINITY	11.200000
11	1.800000	INFINITY	13.000000
12	0.600000	INFINITY	11.800000
13	1.800000	INFINITY	13.600000
14	3.600000	INFINITY	7.200000
15	2.000000	INFINITY	5.800000
16	1.000000	INFINITY	6.800000
17	1.300000	INFINITY	3.800000
18	3.800000	14.100000	2.600000
19	2.700000	2.600000	5.200000
20	1.500000	2.300000	2.600000

APPENDIX C

Figure 5.1: From



APPENDIX C

Figure 5.2: From