ROUTE FIRST-CLUSTER SECOND PROBLEM, A CASE STUDY OF COCA-COLA BOTTLING COMPANY OF GHANA LIMITED

BY

WIAFE DANIEL ASOMANI (B.ed Mathematics)

A Thesis submitted to the Department of Mathematics

Institute of Distance Learning

Kwame Nkrumah University of Science and Technology

in Partial Fulfillment of the Requirements for the Degree

of

MASTER OF SCIENCE

(INDUSTRIAL MATHEMATICS)

OCTOBER, 2012

DECLARATION

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

Wiafe Daniel Asomani, PG4071310	…………………	19/12/12
Student's Name & ID	Signature	Date

Certified By

Prof. S. K. Amponsah	………………....	…………………
Supervisor's Name	Signature	Date

Certified by

Mr. F. K. Darkwah	…………………	…………………
Head of Department's Name	Signature	Date

ACKNOWLEDGEMENT

My profound gratitude goes to the Almighty God for His grace to see me through this work successfully.

I wish to express my sincere appreciation to my honourable supervisor Prof. S.K. Amponsah for his valuable and concise contributions and corrections that had resulted in the successful completion of this research work.

Furthermore, an appreciation goes to the Distribution Manager of The Coca-Cola Bottling Company of Ghana Limited (TCCBCGL), Mr. Solomon Nii Kometey Ashong, who provided most of the data for this research, not forgetting Mr. Owusu at Koforidua Depot of TCCBCGL.

I also thank my Lecturers at the Department of Mathematics, IDL, especially Dr. Amponsah, Nana Kena-Frempong, the late Mr. Agyeman, and Dr. Osei Frimpong.

I am also grateful to my course mates from Eastern Region for their support.

My indebtedness could never run out without acknowledging Mr. Kyei Dompreh who assisted me in writing a programming language for the analysis of the data.

DEDICATION

I dedicate this research work to my dear wife Rubby Obedekah Wiafe and children (Andrew Asomani Wiafe, Edmund Darfour Wiafe and Marilyn Asabea Wiafe) and Millicent Obedekah for their understanding, love and support.

ABSTRACT

Routing of production company vehicle in developing countries poses a challenging task.

This study seeks to minimize the overall cost, which was essentially based on the distance travelled by the vehicles of the company. We proposed a heuristic method to find feasible solution to an extended capacitated arc routing problem on undirected network, inspired by minimization of distribution cost of The Coca-cola Bottling Company of Ghana Limited (TCCBCGL). The heuristic procedure consists of Route first – Cluster second method.

Route first-Cluster second method constructed in a first phase a giant Travelling Salesman Problem tour, disregarding the capacity constraints, and partitioning this tour into clusters with feasible vehicle routes, taking into consideration the vehicle capacity. The shortest path problem within the clusters was solved using Dijkstra's algorithm. The technique was compared with the existing schedule with respect to cost and distance travelled. The adoption of the proposed heuristic resulted in reduction of the number of existing vehicles, the distribution cost and reduction in vehicle distance travelled per supply. The study revealed a good performance of the proposed heuristic method, which would be useful in vehicle routing.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.0 INTRODUCTION

The distribution of goods from depots to their final destination plays essential role in the management of many distribution systems.

Typically, in a distribution system, vehicles provide pick-up, delivery or repair and maintenance services to customers who are geographically dispersed in a given area. In its numerous applications, the common objective of distribution is to find a set of routes for the vehicles to satisfy a variety of constraints so as to minimize the total fleet operation cost, while satisfying the demands of the various destinations.

The problem most manufacturing companies in the world and Ghana in particular, faces, is how to cut down transportation cost whilst satisfying route and available constraints to serve their customers with the demand for some goods and services.

Route first-Cluster second, is one of the numerous ways of solving vehicle routing problems.

1.0.1 PRODUCTION AND DISTRIBUTION

Today, the revolutions in transportation and communication technologies have increased the extent of the domestic market of many nations over the last two centuries (De Vries, 1994). Moreover, the expansion of markets is associated with major changes in the course of many nations economic history. In the late eighteenth and the early nineteenth centuries, the introduction of canals were credited with increasing the levels of inventive activities and triggering of industries (Sokoloff, 1998).Households became less self-sufficient and rather specialized consumer-labourers; firms that specialized in the production of various goods emerged in great numbers. Division of labour within firms led to a re-organisation of production and increased levels of productivity with its challenges (Sokoloff, 1984).

During those periods there were increased in industrialization, which gave birth to proliferation of specialized manufacturing firms (Kim, 2000). In this period, the industrial structure was composed of single unit firms who specialized in production of manufacturing goods and wholesale merchants and retail stores owners who distributed the goods. Since the manufacturing firms typically specialized in a narrow line of products, it was simply too costly for them to market their products directly to consumers. The wholesale merchants, who bought and sold sufficient quantity of products, in this setting, were able to lower the cost of transactions more efficiently. The wholesale merchants in addition to collecting information on various manufacturers, by locating major cities, also send sales agents to rural country stores to collect information on rural consumer demands. In this period, most consumers were able to judge the quality of most products upon usual inspection; however, for some of the goods, they relied on the local producers and retail merchants' reputation for honesty (Kim, 1995).

With advances in science and technology during the late nineteenth centuries, it became increasingly difficult for consumers to discern the quality of product which they consumed.

Kim (1995) indicated that, even the manufacturing processes of the most basic product such as food, became so sophisticated that consumers no longer have enough knowledge to discern whether a product is wholesome or poisonous.

1.0.2 CHANNELS OF DISTRIBUTION

There are three types of distribution; Intensive, Selective and Exclusive distribution. Intensive distribution is where the product is sold to as many appropriate retailers or wholesalers as possible. Selective distribution occurs when the producer restrict the number of retail outlets if the product which required specialized servicing or sales support. Then the Exclusive distribution is when a single outlet is given an exclusive franchise to sell the products in a geographic area (Hitchcock, 1941).

Distribution could be broadly classified into Direct and indirect distribution. There is direct distribution if the producer bypass the marketing intermediaries and supplies the product direct to the consumer. Indirect distribution involves the use of intermediaries or middlemen and retailers to make the product available to the consumer (Kim, 2000).

Channel of distribution is a path through which goods and services flow in one direction (from producer to wholesaler or from vendor to the consumer) and the payments generated by them that flow in the opposite direction (from consumer to the vendor).

A distribution channel can be as short as being direct from the vendor to the consumer or may include several interconnected intermediaries such as wholesalers, distributors, agents, retailers. According to Mahmoud (1996) there exist three main channels of distribution of goods in Ghana.

- Producer to consumer; where the producer sells directly to the consumer.
- Producer to the Retailer and from the Retailer to consumer; where the wholesaler is bypassed and the producer deal directly with the retailer.
- Producer to the wholesaler, from wholesaler to the Retailer and from the Retailer to the consumer; where the wholesaler buys in bulk from the producer and later resells to the retailers who in turn sell to the consumer.

When it comes to heavy industrial producers, there are four channels of distribution in Ghana.

- Producer to consumer; the producer market the products direct to the consumer. Example Tema steel works.
- Producer to customer. Some producers of industrial products use industrial distributors to market their product for them. Example Coca-Cola Bottling Company of Ghana limited, Guinness Ghana Breweries Limited, Ghacem Limited.
- Producer to an Agent, and from the Agent to the customer; is the most popular method foreign organizations use when entering the Ghanaian market.
- Producer to an Agent. From Agent to an Industrial distributor and from industrial distributor to customer.

1.0.3 DISTRIBUTION NETWORK AND MODEL

The entire chain of distribution intermediaries from the supplier to the consumer is called the distribution networks. A strong and efficient distribution network is one of the most important assets a manufacturer can have and is the biggest deterrent that faces the new competitors.

Distribution model concerns with the mathematical simulation of the key decision associated with a distribution channel to compute the optimal solutions regarding inventory, warehousing, routing, transportation and other such factors.

1.1 BACKGROUND TO THE STUDY

Ghana's economy is poised for considerable future growth primarily as a result of oil production, increased cocoa prices and the increased level of direct investment into the country. This along with a number of factors, are expected to result in increased consumer disposable income and resulting increase in demand for The Coca-Cola Bottling Company of Ghana Limited's premium products.

Therefore the company looks forward for future expansion of its customer base as well as piloting a way of distribution and selling that enhances the capability of its sales force. This will help the company out-execute the competition and remain agile enough to get ahead and stay ahead of its competitors.

1.1.2 PROFILE OF COCA-COLA BOTTLING COMPANY OF GHANA LIMITED

The Coca-Cola Bottling Company of Ghana Limited (TCCBCGL) was formed in March 1995 from the divestiture of the Bottling Division of GNTC and started operations on the 7th March 1995 at the GNTC plant at Adjabeng, Accra. It is a subsidiary of the Equatorial Coca-Cola Bottling Company which in turn is owned by The Coca-Cola Company, Atlanta, USA and the Cobega Group of Barcelona, Spain. The Company has two (2) production facilities or plants situated in Accra (main) and Kumasi.

As the effect of the global credit crunch lingers on and companies resort to laying employees off, The Coca-Cola Bottling Company of Ghana Limited (TCCBCGL) has been looking for ways of creating jobs for the youth. The company introduced what it calls the "Trolley Boys Project" during the latter part of 2007 to widen its distribution network as it finds ways of supporting the state's effort at creating job opportunities for the youth.

So far the project has created jobs for about 500 young people between the ages of 20 and 25. The young men who are recruited by the company through its dealers and distributors are given coca-cola branded trolleys and ice chests stuffed with the newly introduced 200ml coca-cola products and other canned as well as plastic bottled brands of The Coca-Cola Company.

The trolley boys are normally stationed at traffic-prone areas in Accra, Kumasi, Takoradi and Koforidua to refresh motorists and pedestrians. According to the Head of the Commercial Function of The Coca-Cola Bottling Company of Ghana, Anton

Van Zyl, the trolley boys make a personal profit of GH¢10 to GH¢15 daily, which amounts to between GH¢3,120.00 and GH¢4,680.00 per annum in a season- about five times more than the nominal GDP in Ghana.

The Coca-Cola Bottling Company of Ghana employs over 800 direct employees and 700 contract staff. The company has about 40,000 registered customers who rely mainly on selling its products for their income.

Apart from the 13 depots throughout the country(Regional Distributors at regional capitals), TCCBCGL has 63 Mini-Depot Operators (MDOs) that employ at least 10 people each and 128 Manual Distribution Centres (MDCs) which also employ 3 people each, across the country.

The company produces 10 main brands. These are Coca-Cola, Fanta, Sprite, Schweppes Malt and Schweppes Carbonated drinks, Krest, Burn Energy Drink, Minute Maid, Stoney Ginger Beer, Dasani and BonAqua. Over the years the company has grown their carbonated soft drinks market share from 60% in 1999 on divestiture to 94.4% in 2009.

DISTRIBUTION MODEL OF TCCBCGL

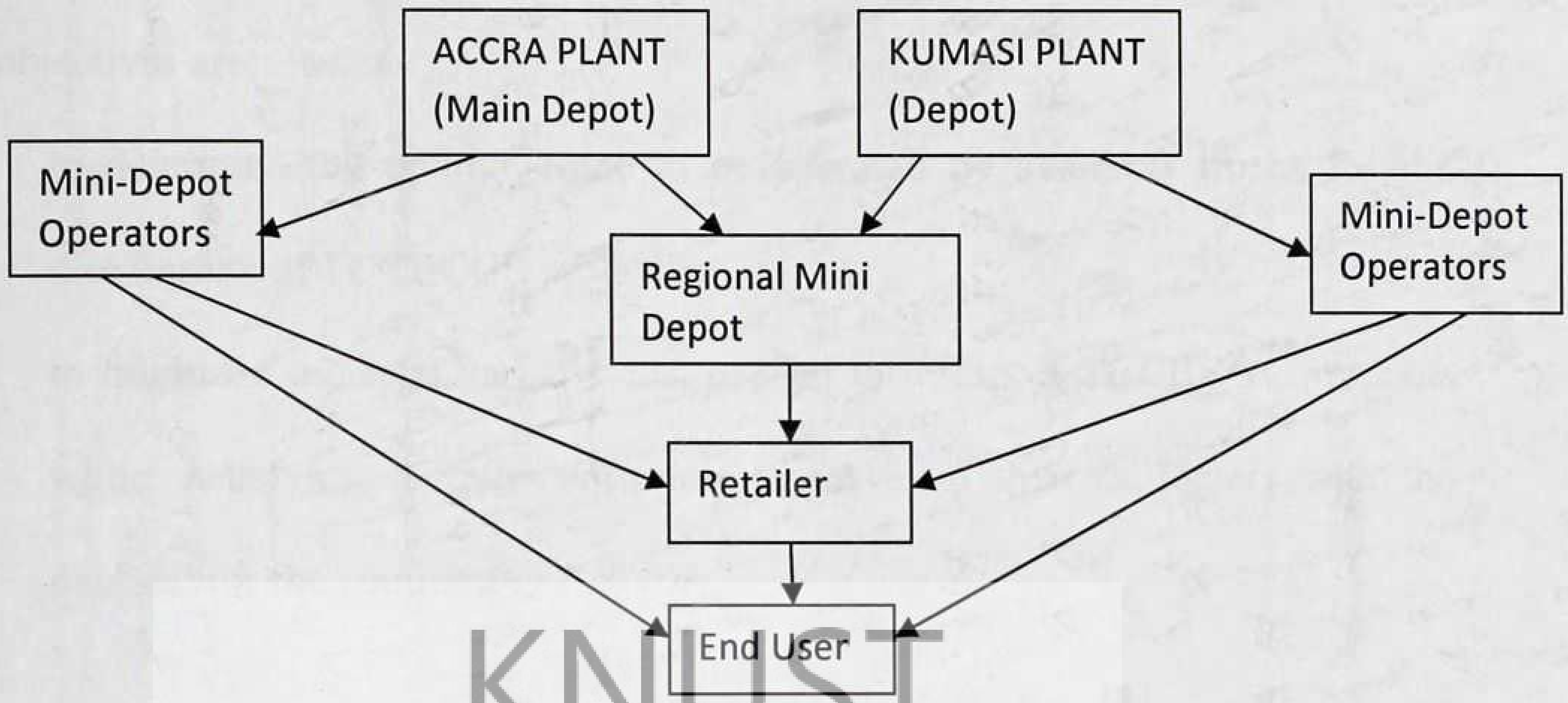


Figure 1.1: Schematic of TCCBGL distribution model.

1.2 PROBLEM STATEMENT

Every company seeks to optimize production efficiency and distribution control. Every industry has its distribution problems. One measure of the scale of these problems is the total distribution cost. Another measure is the proportion of total costs which is attributable to distribution. Clearly it is worth making some effort to increase efficiency in both cases. A figure for the cost of operating one vehicle in the distribution system is high. TCCBCG operate a fleet of about 21 long Haulage trucks. So a re-organization which would result in a saving of 1 vehicle in 5 would reduce the company's distribution cost.

This thesis seeks to use Route first-cluster second method of routing system to minimize transportation cost of TCCBCGL in order to achieve its target profit.

1.3 OBJECTIVE OF THE STUDY

This research is centered on Route first- cluster second method for vehicle routine. The objectives are:

(i) to determine the optimal route to be operated by available trucks for the distribution of TCCBCGL products.

(i) to minimize the total variable cost needed to transport TCCBCGL products while satisfying routing constraints to serve all their customers with the demand for the commodity.

1.4 JUSTIFICATION

The cost of transportation largely affects production companies which affect their operations. The recommendations and suggestions from this study when implemented will help Coca-Cola Bottling Company of Ghana achieve cost saving in distribution so as to improve upon their profit margin in order to remain competitive in the Ghanaian market. Coca-Cola Company and all other companies will be able to expand their operations when they are able to maximize profit. This will create more employment opportunities for people which will in effect have positive impact on the economy of Ghana

When organizations are able to reduce their transportation cost and save money, such monies could be used in enhancing corporate social responsibilities which will benefit the communities in which these organizations operate. The study will help The Coca-Cola Company of Ghana and other manufacturing and distribution companies to implement new transportation and distribution strategies to reduce cost and make more profit. Finally, the study will serve as a reference material for the academia for future research work.

1.5 METHODOLOGY

Data on the number of vehicles (trucks), vehicle capacity for all the vehicles, all their key distributors, their destination and distances between them and the depot, and, the cost of transportation for all the vehicles from the depot to their distribution points and back to the depot, would be obtained from the Distribution and Sales managers of the company.

Route first-cluster second method will be used to solve the transportation problem. The second phase of the method is a standard shortest-path problem on an acyclic graph and therefore Dijkstra's algorithm will be used. Computations will be done using VB.net for Dijkstra's algorithm.

1.6 SCOPE OF THE STUDY

The study will cover the Mini-Depot Operators (MDO) of The Coca-Cola Bottling Company of Ghana Limited within Western, Central, Eastern, and southern Volta Regions, their actual destination, the distances from Accra to these MDOs, the distances between the MDOs and the quantity of goods supplied to the MDOs at a time.

1.7 LIMITATION

There were a number of factors which could limit this piece of work to some extent. One of the constrains is access to information. Due to competitions among companies, there is reluctance by Managers in given out vital company information.

Also the time span (duration) for the research was limited. However sufficient time is needed for such a work. Lastly, funds and resources was also a constraint since one needed to move to the premises of Coca-Cola Bottling Company of Ghana Limited for data.

1.8ORGANISATION OF THE THESIS

The thesis is divided into five chapters. Chapter one provides information on the background of the study. It also throws light on the problem statement and objective of the study, methodology, justification, and organization. Chapter two review related works of some authors and summarize the route first-cluster second method of solving transportation problem. Chapter three presents the methodology, mathematical formulations and variants methods of solving transportation problem in order to minimize cost. Chapter four presents data collection and analysis of the study. Chapter five captures the summary of findings, recommendations and conclusions.

1.9 SUMMARY

In this chapter, we focused on production and distribution as it is today. We also looked at channels of distribution and distribution network, and its associated problems. Brief history of The Coca-Cola Bottling Company of Ghana Limited, background to the study, statement of the problem, objectives of the thesis, methodology, scope of the study, limitation and organization of the thesis. In the next chapter, we shall review some literature pertaining to Vehicle routing problem and Route first-Cluster second problem.

CHAPTER 2

LITERATURE REVIEW

2.1 THE VEHICLE ROUTING PROBLEM (VRP)

The vehicle routing problem lies at the heart of distribution management. It is faced each day by thousands of companies and organization engaged in the delivery and collection of goods or people. Conditions vary from one setting to the next, therefore the objectives and constraints encountered in practice are highly variable. Most algorithmic research and software development in this area focus on a limited number of prototype problems. By building enough flexibility in optimization systems one can adopt these to various practical contexts (Laport 1992).

The vehicle routine problem (VRP) can be defined as a problem of finding the optimal route of delivery collection from one or several depots to a number of cities or customers, while satisfying some constraints. Collection of household waste, gasoline delivery trucks, goods distribution, snow plough and mail delivery are the most used application of the VRP. The VRP plays a vital role in distribution and logistics.

Much progress has been done since the publication of the first article on the" truck dispatching" problem by Dantzig and Ramser (1959). The authors described the problem as a generalized problem of Travelling Salesman Problem (TSP). Several variants of the basic problem have been put forward. Strong formulations have been proposed, together with polyhedral studiers and exact decomposition algorithms. In particular the study of this class of problems has stimulated the emergence and the growth of several meta-heuristics whose performance is constantly improving.

In 2002, Toth and Vigo (Maffioli, 2003) have reported that the use of computerized methods in distribution processes often results in saving ranging from 5% to 20% in transportation costs. Barker (2002), described several case studies where the application of VRP algorithms has led to substantial cost savings.

2.1.1 CLASSICAL VEHICLE ROUTING PROBLEM

The Classical Vehicle Routing Problem (CVRP) is one of the most popular problems in combinatorial optimization and its study has given rise to several exact and heuristic solution techniques of general applicability it generalizes the travelling salesman problem (TSP) and is therefore NP-hard. A recent survey of the VRP can be found in the first six chapters of the book edited by Toth and Vigo (2002 a).

The VRP is often defined under capacity and route length restrictions. When only capacity constraints are present the problem is denoted by CVRP. Most exact algorithms have been developed with capacity constraints in mind but several apply *mutatis mutandis* to distance constrained problems. In contrast most heuristics explicitly consider both types of constraint.

In classical VRP, the customers are known in advance. Moreover, the driving time between the customer and the service times at each customer are used to be known (Madsen et al, 1995). The classical VRP can be defined as follows (Laport, 1992):

Let G = (V, A) be a graph where $V = \{1, \ldots, n\}$ is a set of vertices representing delivery points with the depot located at vertex 1, and A is the set of arcs. With every arc (i, j) $i \neq j$ is associated with a non-negative distance matrix $C = (C_{ij})$. In some contexts, C_{ij} can be interpreted as a travel cost or as a travel time. When C is symmetrical, it is often convenient to replace A by a set E of undirected edges.

THE SYMMETRIC VRP

The symmetric VRP is defined on an ample undirected graph G = (V, *E*). The set v = {0,…,*n*} is a vertex set. Each vertex $i \in V \backslash \{0\}$ represents a customer having a non-negative demand q_i, while vertex O responds to a depot. To each edge $e \in E = \{(i, j): i, j \in V, i<j\}$ is associated with a travel cost C_e or C_{ij}. A fixed fleet of m identical vehicles each of capacity *Q*, is available at the depot. The symmetric VRP calls for the determination of a set of m routes whose total travel cost is minimized and such that:

(i) each customer is visited exactly once by one route

(ii) each route starts and ends at the depot.

(iii) the total demand of the customers served by route does not exceed the vehicle capacity *Q*,

(iv) the length for each route does not exceed a preset limit L.

(v) it is common to assume constant speed so that distance travel times and travel cost are considered as synonymous).

A solution can be viewed as a set of m cycle sharing a common vertex at the depot. The asymmetric VRP is similarly defined on a directed graph G = (V, A), where $A = \{(i, j): i, j \in V, i \neq j\}$ is and arc set. In this case a circuit (directed cycle) is associated with a vehicle route.

2.1.2 CAPACITATED VRP

The capacitated vehicle routing problem can be described as follows: Let G = (V',E) an undirected graph is given where V' = (0, 1,…..,n) is the set of n + 1 vertices and E is the set of edges. Vertex **O** represents the depot and the vertex set V = V'\ {0} corresponds to n customers. A nonnegative cost d_{ij} is associated with each edge (i,j) $\in E$. The q_i units are supplied off from depot **O** (We assume q = o). A set of m identical vehicles of capacity Q is stationed at dept **O** and must be used to supply the customers. A route is defined as a least cost simple cycle of graph G passing through depot **O** and such that the total demand of the vertices visited does not exceed the vehicle capacity.

The practical importance of the capacitated vehicle routing problem provides the motivation for the effort involved in the development of heuristic algorithms (Baldacci et al., 2007). Survey covering exact algorithms was given by Laport (1992). The chapters of Toth and Vigo (Moffioli, 2003) have surveyed the most effective exact methods proposed in the literature up to 2002. A recent survey of the C-VRP, covering both exact and heuristic algorithms can be found in the chapter of Cordeau et al., (2001) in the book edited by Bernhart and Laporte (Baldacci et al., 2007). The most promising exact algorithms for the symmetric capacitated VRP, which have been published since then are due to Baldacci et al., (2004), Lysgaard et al., (2004) and Fukasawa et al., (2006).

2.1.3 VARIANTS OF VRP

Some of the variants are:

(i) *VRP with time windows*: In the VRP with time windows, a number of vehicles is located at a central depot and has to serve a set of geographically dispersed customers with a demand with a specific time window. The problem is to optimize the use of fleet of vehicle that must make a number of stops to serve a set of customers, and to specify which customers should be served by which vehicle and in what order to minimize the cost, subject to vehicle capacity and service time restrictions (Ellabib et al., 2002).

(ii) *VRP with Pick-up delivery*: One complication in real-life situations as far as VRP is concern, is the complication that arises in practice such that goods not only need to be brought from the depot to the customers, but also must be picked up at a number of customers and brought back to the depot. The VRP with Pick-up and delivery is known as VRP with Backhauls (Ropke and Pisinger 2006; Bianchessi and Rughini 2007). The problem can be divided into two independent (CVRP) (Rokpe and Pisinger 2006); one for the delivery (Linehaul) customers and one for pick-up (backhaul) customers, such that some vehicles would be designated to linehaul customers and other to backhaul customers.

(iii) *Stochastic VRP*: The stochastic VRP arise when considering demands and travel times as stochastic variable.

Other variants include the *heterogeneous vehicle feet*.

2.2 THE TRANSPORTATION PROBLEM

One of the subclasses of the linear programming problems for which simple and practical computational procedures have been developed, that take advantage of the special structure of the problem is the Transportation problem.

The first formulation of transportation problem was presented, along with a constructive solution by Hitchcock in 1941.Koopman (1947) single handedly spearheaded a research on potentialities of linear programs for the study of problems in Economics. Since Koopman's work was based on the work done earlier by Hitchcrock, the classical case of the transportation problem is often referred to as Hitchcock-Koopman's transportation problem. The problem may be expressed as minimization of transport cost for moving a single commodity from ***a*** origins (source) to ***b*** destinations (sink) while operating within supply and demand constraints.

Hammer (1969) introduced the time minimizing or bottleneck transportation problem, and the algorithm for solving the problem. The objective was to minimize the total time to transport all supply to the destination, rather than minimizing cost.

Williams (1963) and Szwarc (1964) discussed the stochastic transportation problems, with the objective of minimizing total transportation cost plus expected penalty cost, Wilson (1972, 1973, and 1975) showed that a linear approximation can be used in order to solve the stochastic transportation problem as a capacitated transportation problem.

Toth and Vigo (1997) examined the problem of determining and optimal schedule for a fleet of vehicles used to transport handicapped persons in an urban area, by using a Tabu Threshold procedure to the starting solution obtained by insertion algorithm.

2.3 REVIEW OF ALGORITHMS FOR VEHICLE ROUTING PROBLEMS

Gillet and Miller (1974) presented a sweep algorithm to solve the vehicle dispatch problem. The principle of this algorithm is cluster first-route second. A number of customers supplied by a single depot are clustered by their polar coordinate angle. All customers are re-ordered according to their polar coordinate angle and a cluster is created by sweeping consecutive customers as long as the capacity constraint is not violated. For each cluster, a travelling salesman problem must be solved to determine the minimum path in the route.

Another heuristic based on the cluster first-route second approach is the two-phase method of Christofides, Mingozzi and Toth (1979).According to Bodin and Golden (1981) there are many other approaches to solve the vehicle routing problem which includes cluster first-route second and savings/insertion. The cluster first-route second approach clusters all demand nodes into feasible groups and then efficient routes are proposed for each group. Route first-cluster second approach initially determines a single route through all demand nodes, and then partitions this single route into smaller and more feasible routes.

Bodin and Krush (1978) utilized the route first-cluster second approach for routing street sweepers. They pointed out that the route first-cluster second approach should be superior to the cluster first-route second for solving this problem though route first-cluster second may produce overlapping clusters.

Bodin and Berman (1979) used route first-cluster second for routing school buses to and from a single school.

Je Beasley (1983) studied route first-cluster second method for vehicle routing and compared the performance of route first-cluster second to the saving algorithm and 3-optimal algorithm. The computational result showed that the route first-cluster second gave solutions at least as good as the saving method and often as good as the 3-optimal method.

Bramel and Simihi-Levi (1997) pres ented a book titled" The logic of logistics: Theory, algorithms and application for logistics management". They pointed out a survey of a variety of results covering most of the logistics area. One part of this work is about vehicle routing problem, covering and analysis of the single depot capacitated vehicle routing problem with equal demands and unequal demands. They also addressed the VRP with time window constraints and solving the VRP using a column generation approach.

2.4 ROUTE FIRST –CLUSTER SECOND METHODS FOR VEHICLE ROUTING

The vehicle routing problem can be defined as the problem of designing routes for delivery vehicles of known capacities, operating from a single depot, to supply a set of customers with known locations and known demand for a certain commodity. Routes for vehicle are designed to minimize some objective such as the total distance travelled.

Recent surveys (Bodin et al.,1979), Mole, Watson et al., 1979) list many approaches (both heuristics and optimal) to the problem based upon a route first-cluster second heuristics. A similar approach has been successfully applied to bus routine problems (Bodin et al.,1979), the routine of electricity meter readers (Stern and Dror, 1987), the routine of street sweepers and vehicle fleet size and mix problems(Golden et al., 1984).

The basic route first-cluster second method is best illustrated by a diagram, where we have a central depot surrounded by a number of customers. We first form a "giant tour" from the depot around all the customers and back to the depot (i.e. a travelling salesman's tour around all the customers including the depot). This tour can be formed in a number of different ways.

D - Depot

C - Customer C

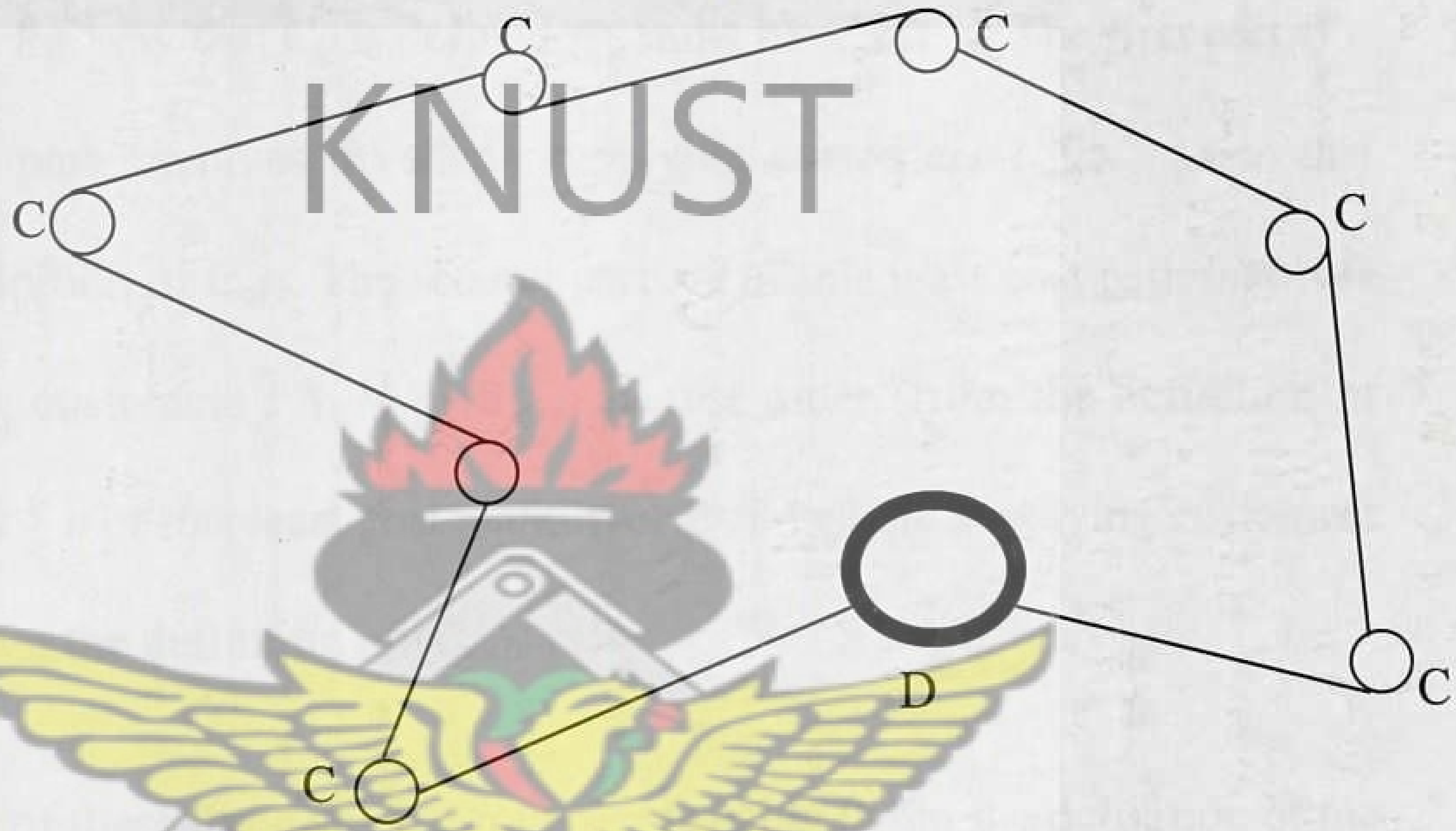


Figure 2.1: Example of a 'Giant' tour.

The key to the approach is that it is very easy to optimally partition such a tour into a set of feasible vehicle routes. Arbitrarily assign a direction to the giant tour and (without loss of generality) let 1 be the first customer on the directed tour after the depot (which we denote by 0), 2 be the second customer on the tour after the depot,…, n be the last customer on the tour after the depot. Let (d_{ij}) be the inter-customer distance matrix and defined a matrix (C_{ij}) by

C_{ij} = the distance travelled by a vehicle in supplying the customer $(i + 1, i + 2,..j)$ in that order if the vehicle route $(0, i + 1, i + 2, \ldots, j, 0)$ is feasible $(i<j)$

$= \infty$ otherwise

$$\text{i.e. } C_{ij} = d_{o(i+1)} + \sum_{k=i+1}^{j-i} d_{k(k+1)} + d_o \qquad or \quad C_{ij=\infty}$$

Note that we assume here that all vehicles are identical. If we then find the least cost path from O to n in the directed graph with arc cost (C_{ij}) we will have an optimal partition of the (directed) giant tour into feasible vehicle routes (Note that if no path from **O** to n exist then the problem is infeasible).

For example, suppose that the cost path from **O** to n is **O** $-s-t-n$ of total cost C_{os} + C_{st} + C_{tn}, then from the way that C_{ij} is defined we must have $s<t<c$. The first part **O** $-s$ of this least cost path involves a vehicle supplying customers 1, 2,….,s in that order (from the definition of C_{os}). The second part $s–t$ of this least cost path involves a vehicle supplying customers $s+1$, $s+2$,…t in that order (from the definition of C_{st}). The final part $t-n$ of this least cost path involves a vehicle supplying customers $t+1$, $t+2$,…,n (from the definition of (C_{tn}).

We know that each of these three vehicle routes is feasible (from the definition of the (C_{ij}) and together they supply all the customers. Hence we have a solution to the vehicle routing problem. Note that this partition of the giant tour into three feasible vehicle routes is optimal since we found the least cost path from **O** to n in the directed graph with arc cost (C_{ij}). (Any path from **O** to n corresponds to a partition of the giant tour into feasible vehicle routes and the least cost path from **O** to n corresponds to an optional partition.)

We can see from the above description why the method is called route first-cluster second. We first decided the order in which the customers are to be visited (the routing part of the process) and then partition the customers (cluster the customers) into sets that constitute feasible vehicle routes. The route first-cluster second heuristic is attractive for a number of reasons:

(i) The use of a giant tour ensures that customers who are near to each other are close together in the giant tour and hence likely to be together on the vehicle routes considered in the formation of the matrix (C_{ij}).

(ii) Via the partitioning approach we are able implicitly to consider a large number of feasible vehicle routes and from them pick an optimal set of routes.

(iii) The partitioning of the giant tour is relatively fast computationally (eg using the algorithm of Dijkstra involves only O (n^2) operations).

(iv) Because the partitioning procedure is fast (and the other parts of the method are also not particularly time consuming) one can start form a number of different giant tours and produce a feasible set of vehicle routes from each tour. This overcomes the problem that any single giant tour might lead to a bad set of vehicle routes.

Note here that since it is easily shown that an optimal travelling salesman (giant) tour followed by an optimal partitioning does not necessarily lead to an optimal set of vehicle routes, one would expect that a heuristic, rather than optimal, approach to the formation of the giant tour would be sufficient (e.g. an initial random tour followed by a 2–optimal procedure or other heuristic approaches to the travelling salesman problem).

Levy et al., (1984) made a number of similar points in their discussion of the use of the approach for vehicle fleet size and mix problems.

There are a number of extensions that can make the basic method described above both to improve it to cope with the practical constraints associated with vehicle routing problems.

CHAPTER 3

METHODOLOGY

3.0 INTRODUCTION

In this chapter, we shall put forward the research methodology of the study.

3.1 HOW THE ALGORITHM FOR VRP WORKS.

3.1.1 SWEEP ALGORITHM

The sweep algorithm applied to planar instances of the VRP. Feasible clusters are initially formed by rotating a ray centered at the depot. A vehicle route is then obtained for each cluster by solving a TSP. Some implementations include a post optimization phase in which vertices are exchanged between adjacent clusters and routes are re-optimized. The sweep algorithm is commonly attributed to Gillet and Miller (1974), who popularized it, but it was first found in a paper by Wren and Holiday (1972).

A simple implementation of this method is as follows;

Assume each vertex i is represented by its polar coordinates (O_i, P_i) where O_i is the angle and P_i is the ray length. Assign a value $O_i^* = 0$ to an arbitrary vertex i^* and compute the remaining angles from (O, i^*). Rank the vertices in increasing order of their O_i.

*Step*1 : (Route initialization). Choose an unused vehicle k.

*Step*2 : (Route construction). Starting from the unrouted vertex having the smallest angle, assign vertices to vehicle k as long as its capacity or the maximal route length is not exceeded. In tightly constrained DVRPs, 3 – opt may be applied after each insertion. If unrouted vertices remain, go to *step*1

Step 3: (Route optimization) optimize each vehicle route separately by solving the corresponding TSP (exactly or approximately).

3.1.2 FISHER AND JAIKUMAR ALGORITHM

The fisher and Jaikumar algorithm is also well known. Instead of using a geometric method to form the cluster, it solves a Generalized Assignment Problem (GAP). It can be described as follows;

Step 1: (Seed selection). Choose seed vertices j_k in V to initialize each cluster *k*.

Step 2: (Allocation of customers to seeds). Compute the cost d_{ik} of allocating each customer *i* to each cluster *k* as

$$d_{ik} = \min\{C_{oi} + C_{ijk} + C_{jko},\ C_{ojk} + C_{io}\} - (C_{ojk} + C_{jio})$$

Step 3: (Generalized assignment). Solve a GAP with costs d_{ij}, customer weights q_i and vehicle capacity Q.

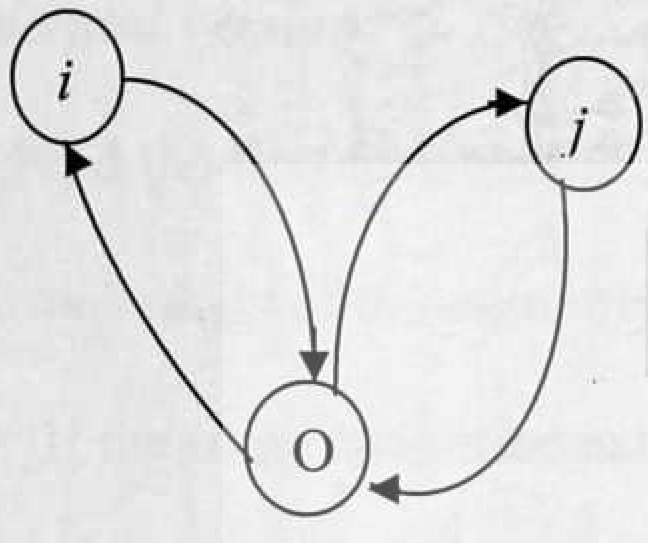
Step 4: (TSP Solution). Solve a TSP for each cluster corresponding to the GAP solution.

The number of vehicle routes *k* is fixed a priori in the fisher and Jaikumar heuristic. The authors proposed a geometric method based on the partition of the plane into K cones according to the customer weights. The seed vertices are dummy customers located along the rays bisecting the cones. Once the clusters have been determined, the TSP s are solved optimally using a constraint relaxation based approach. However Fisher and Jaikumar article does not specify how to handle distance restrictions, although some are present in the test problem.

3.1.3 CLARKE – WRIGHT SAVINGS ALGORITHM

The savings algorithm is a heuristic algorithm, and therefore it does not provide an optimal solution to the problem with certainty. The method does, however often yield a relatively good solution.

The basic savings concept expresses the cost savings obtained by joining two routes into one route as illustrated in figure 3.1, where point **O** represents the depot



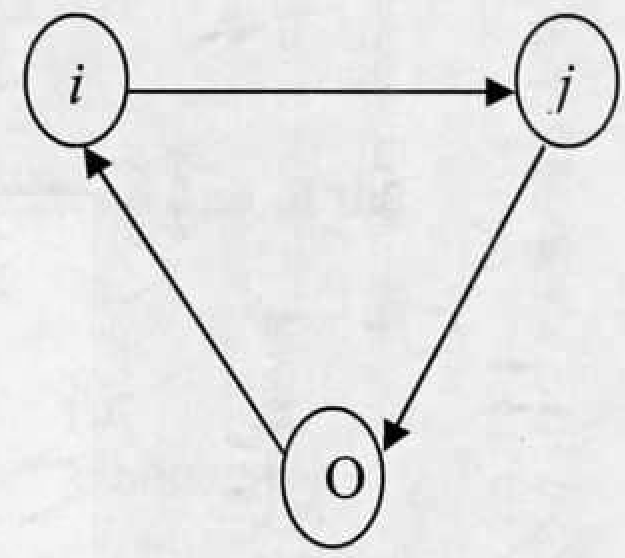


Figure 3.1: cost savings

Initially in fig. 3.1(a) customers *i* and *j* are visited on separate routes. An alternative to this is to visit the two customers on the same route, for example in the sequence *i*- *j* as illustrated in fig. 3.1(b). Because the transportation costs are given, the savings that result from driving the route in figure 3.1(b) instead of the two routes in figure 3.1(a) can be calculated. Denoting the transportation cost between two given points *i* and *j* by C_{ij}, The total transportation cost D_a in figure 1 (a) is : $D_a = C_{oi} + C_{io} + C_{oj} + C_{jo}$

Equivalently, the transportation cost D_b in fig. 1(b) is : $D_b = C_{oi} + C_{ij} + C_{jo}$

By combining the two routes one obtains the savings S_{ij}.

$$S_{ij} = D_a - D_b = C_{io} + C_{oj} - C_{ij} .$$

Step 1: Calculate the savings $S_{ij} = C_{oj} + C_{io} - C_{ij}$ for all pairs of customer s*i* and *j*

Step 2: Order the savings in descending order.

Step 3: Starting at the top of the list do the following;

For parallel version:

Step 4: If a given link results in a feasible route according to the constraints of the VRP, then append this link to the solution if not reject the link.

Step 5: Try the next link in the list and repeat step 4 until no more links can be chosen.

For sequential version;

Step 4: Find the first feasible link in the list which can be used to extend one of the two ends of the currently constructed route.

Step 5 : If the route cannot be extended further terminate the route. Choose the first feasible link in the list to start a new route.

Step 6 : Repeat step 4 and 5 until no more links can be chosen.

3.1.4 BRAMEL AND SIMCHI -LEVI ALGORITHM

Bramel and Simchi-Levi described a two-phase heuristic in which the seeds are determined by solving a capacitated location problem and these remaining vertices are gradually included into their allotted route in a second stage. The authors suggest first locating k seeds, called concentrators, among the n customer locations to minimize the total distance of customers to their closest seed while ensuring that the total demand assigned to any concentrator does not exceed Q. Vehicle routes are then constructed by inserting at each step the customer assigned to that route seed having the least insertion cost.

Bramel and Simchi – Levi describe the two-phase heuristic in another way, in which they first orders the customers according to their locations, disregarding demand size, and then partitions this ordering to produce feasible clusters that satisfy vehicle capacity. These clusters consist of set of customers that are consecutive in the initial order. Customers are then routed within their cluster depending on the specific heuristic. This heuristic is route first – cluster second. This model will therefore give the minimum number of trips. However, we do not allow splitting the orders to favour vehicle capacity constraint. Therefore, the delivery quantities will be as high as possible but not exceeding vehicle capacity. The route first – cluster second approach takes the following steps:

Step 1: Begin with the solutions from the customers in each replenishment interval.

Step 2: Route all customers into a single route by considering their traveling distance from one point to another point. Begin the route at the depot and consider the closest customer relative to the depot.

Step 3: The customers that are close to the routed customer will be routed in this single route.

Step 4: Repeat step 3 until all customers are routed.

Step 5: For clustering phase, begin with the sequence of a single route. Start with the customers in the first position, add the delivery quantity (demand) of the customer at the second position to the first position of the sequence.

Step 6: If the delivery quantity is greater than the vehicle capacity, move to the next Position in the sequence and repeat step 5 until all customers are considered.

Step 7: The next cluster will be created by considering the customers that are left in the sequence. Repeat step 5, 6 and 7 until all customers are clustered.

Step 8: Apply the shortest path algorithm such as nearest algorithm, Floyd's algorithm or Dijkstra's algorithm to each cluster in order to achieve minimum distance in each cluster.

3.2 GRAPH THEORY

Graphs are defined by a set of vertices and a set of edges where each edge connects two of its vertices. Graphs are further classified into directed and undirected graph depending on whether edges they are directed. An important subclass of directed graphs that arises in many applications such as precedence constrained scheduling problems is directed acyclic graphs (DAG). Other interesting subclasses of undirected graphs include trees and bipartite graphs. There are two main data structure for representing graph, the adjacency lists and the adjacency matrix.

Graph searching procedures such as depth – first search (DFS) and breadth – first search (BFS) [Cormen et al., 1989, Tarjan, 1983] form basic pre-processing steps for most graph algorithms. Algorithms based on DFS have been known for the problem of searching mazes. DFS on directed graphs can be used to classify its vertices into strongly connected components to detect cycle and to find a topological order of the vertices of directed acyclic graph

3.2.1 MINIMUM – COST PROBLEM

Graphs can be used to model networks where vertical model sites and edges model links between sites. Each link has an associated constructed cost. Finding a set of links of minimum total cost that connects all the sites is known as the minimum SPANNING TREE problem. It is known that a greedy algorithm solves the problem and a variety of algorithms have been developed for finding minimum spanning trees. A recent exciting development was a randomised linear time algorithm to solve the problem.

3.2.2 SHORTEST – PATH PROBLEM

A class of important problems in graphs is shortest-path problems, which play an important role in routing message efficiently in networks. The problems of finding the transitive closure of a graph and all pairs shortest path have been well studied and several algorithms have been proposed (Cormen et al., 1989).

3.2.3 MATCHINGS AND FLOWS

Maximum-flow problems is that of finding a maximum flow of a single commodity from a source vertex to a sink vertex that satisfies capacity constraints on the edges and flow conservation constrains at the vertices. Associating costs with edges yields the minimum-cost flow problem. Several graph problems, including the assignment problem and graph connectivity can be reduced to the minimum cost flow problem.

Graph is often depicted as a set of points (node, vertices) joined by links (edges). A graph is a pair, $G = (V, E)$, of sets satisfying $E \in [V]$; thus the elements of V are the nodes (or vertices) of the graph G, the elements of E are its links (edges). In this case, E is a subset of the cross product $V*V$ which is denoted by $E \in [V]$. We shall always assume that $V \cap E = \phi$ to avoid notational ambiguities.

A connected graph is a non-empty graph G with paths from all nodes to all other nodes in the graph. The order of a graph G is determined by the number of nodes.

Graphs are finite or infinite according to their order. A graph having a weight or number, associated with each link is called a weighed graph, denoted by G = (V,E,W).

An example is shown below.

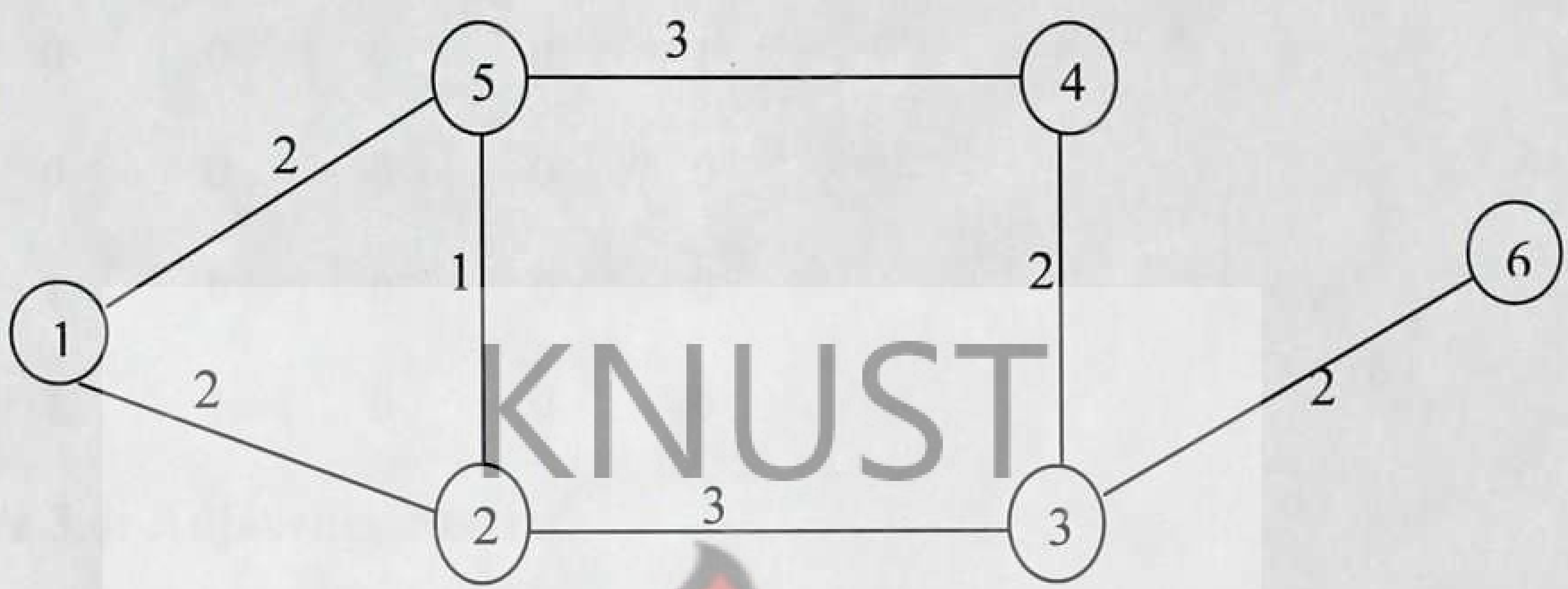


Figure 3.2: A diagram of weighted graph with 6 nodes and 7 edges/ links

Graph may be a dense graph or a sparse graph depending on the number of links. The numbers of links that are roughly quadratic in their order are usually called dense graphs. In the case where the numbers of links are approximately linear in their order, are called sparse graphs.

3.2.4 ADJACENCY MATRIX

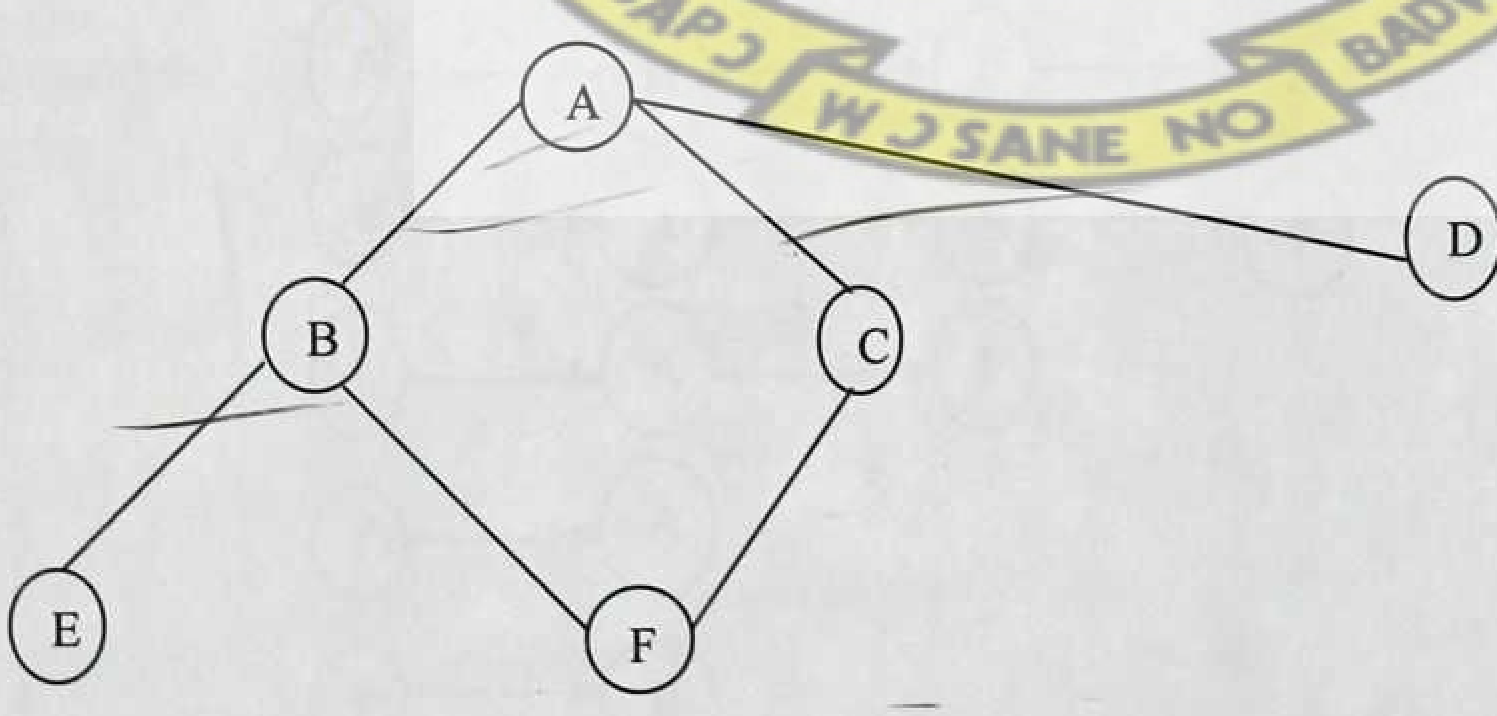


Figure 3.3: A diagram of unweighted graph

The adjacency matrix of a graph is an n by n matrix stored as a two-dimensional array with rows and columns labeled by graph nodes. A 1 or 0 is placed in position (*u*, *v*) according to whether *u* and *v* are adjacent or not. Node *u* and *v* are defined as adjacency if they are joined by a link.

	A	B	C	D	E	F
A	O	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	0	1
D	1	0	0	0	0	0
E	0	1	0	0	0	0
F	0	1	1	0	0	0

Figure 3.4: Adjacency Matrix

3.2.5 ADJACENCY LIST

The adjacency list is another form of graph representation in computer science. This structure consists of a list of all nodes in a given graph. Each node in the list is linked to its own list containing the names of all nodes that are adjacent to it. Adjacency list is most often used when the graph contains a small to moderate number of links.

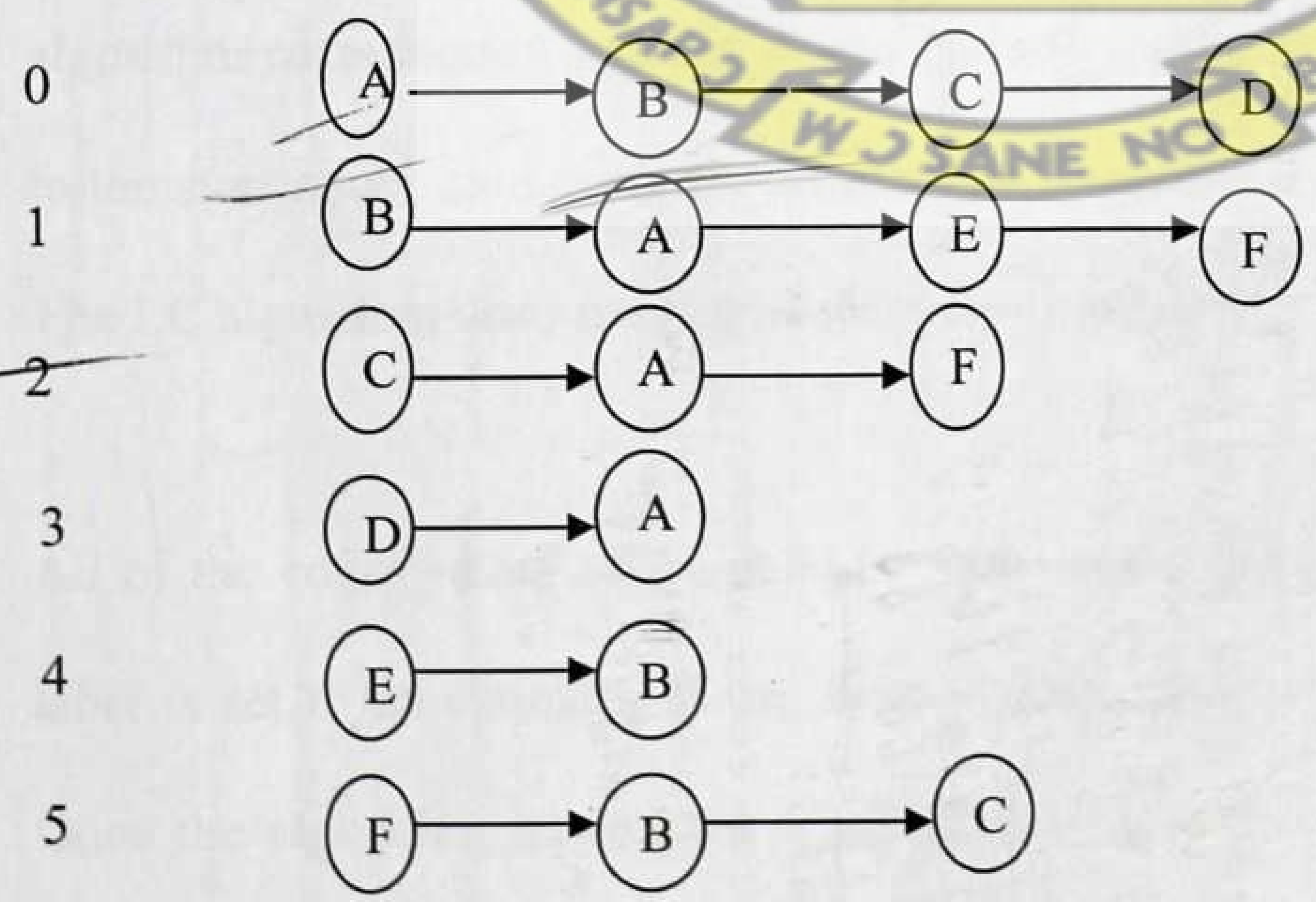


Figure 3.5: Adjacency list.

3.3 SHORTEST PATH ALGORITHM (SP ALGORITHM)

Path finding is applicable to many kinds of networks such as roads, utilities, water, electricity meter reading, telecommunication and computer networks. The total number of algorithms that have been developed over the years is immense depending only on the type of network involved.

After Minty (1957), considerable progress has been made in the shorts path algorithm. Numbers of major pages published were published by Bellman (1958), Dijkstra (1959) and Moore (1959). There are a number of review papers, such as the one by Dreyfus (1969).

Labeling algorithms are the most popular and efficient algorithms for solving the SP problem. These algorithms utilize a label for each node that corresponds to the tentative shortest path length P_k to that node. The algorithm proceeds in such a way that these labels are updated until the shortest path is found.

Labeling algorithm can be divided into two sets: The Label Setting (LS) algorithm and Label Correcting (LC) algorithm. For each number of iteration, the LS algorithm permanently sets the label of a node as the actual shortest path from itself to the start node, increasing the shortest path vector by one component at each step. The LC algorithm does not permanently set any labels.

All of the components of the shortest path vectors are obtained. Simultaneously, a label is set to an estimate of the shortest path from a given node at each iteration. Once the algorithm terminates, a predecessor label is stored for each node, which represent the previous node in the shortest path to the current node. As a result, it only determine the path set $P_t = \{P_1, ... ,P_t\}$, in the last step of the algorithm. Backtracking is then used to construct the shortest paths to each node. Typical label setting algorithms include Dijkstra's algorithm and A* algorithm. An example of label correcting algorithms is Floyd – warshall algorithm.

The shortest path algorithms are currently widely used. They are the basis for the network flow problems, tree problems and many other related problems. They determine the smallest cost of travel of a production cycle, the shortest path in an electric circuit or the most reliable path.

3.3.1 GENERAL CLASSIFICATION OF SHORTEST PATH ALGORITHM.

The shortest path algorithms are either matrix algorithm or tree building algorithms (tree algorithm are also called labeling algorithms).

Matrix Algorithm

Matrix algorithms store the network information in the matrix form and carry out the computations using basic matrix operations (as addition and multiplication of matrices or matrix's elements). The disadvantage of the matrix is the imposed inefficient matrix representation of a sparse network. The more significant disadvantage is that the matrix representation allows one directed link between two nodes (there can be at most two links between two nodes, but they have to be of distinct directions).

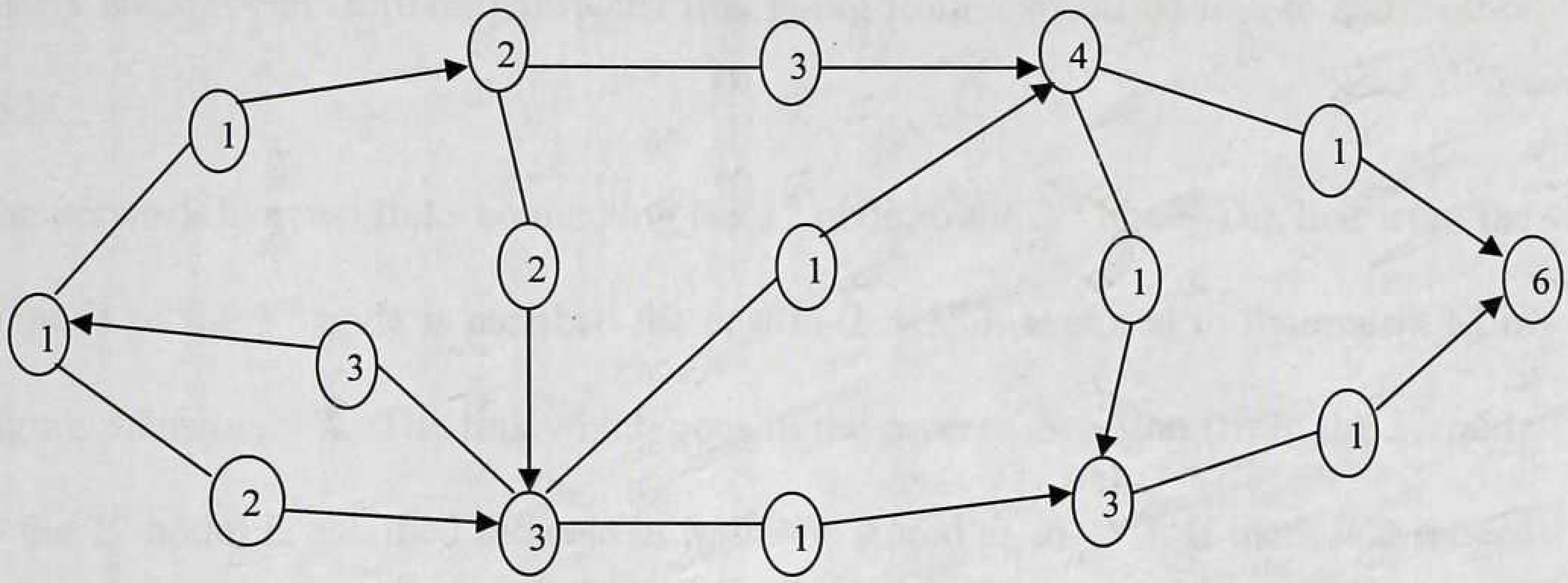


Figure 3.6: Sample Network that can be represented in a matrix form

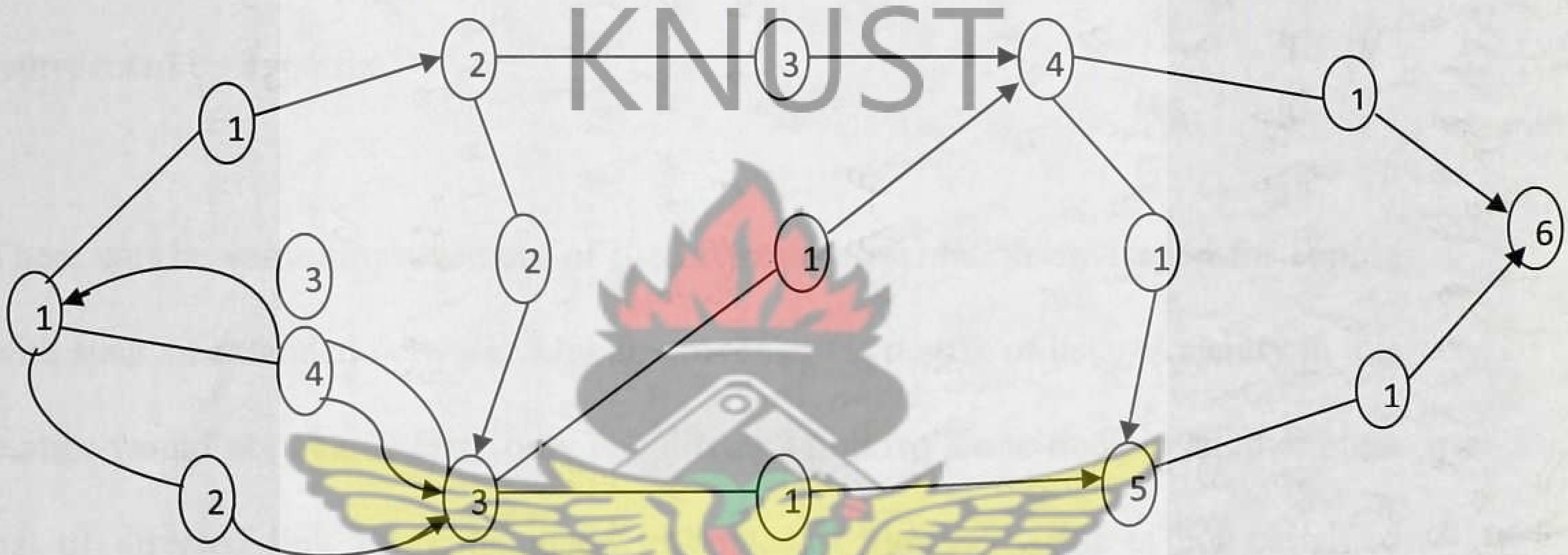


Figure 3.7: Sample Network that cannot be represented in a matrix form.

$$M = \begin{pmatrix} 0 & 1 & 2 & \infty & \infty & \infty \\ \infty & 0 & 2 & 3 & \infty & \infty \\ 3 & \infty & 0 & 1 & 1 & \infty \\ \infty & \infty & \infty & 0 & 1 & 1 \\ \infty & \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Figure 3.8: Matrix representation of the network of figure 3.6.

The network from Figure 3.6 is specified by a matrix in Figure 3.8. Not every network can be represented in such a way. If a network has more than one directed link from a single node to some other node, then it cannot be represented in a regular matrix since it can store only directed link going from a specified link to some other node.

The network has two links connecting the 1st node to the 3rd node. The link from the 1st node to the 3rd node is ascribed the cost of 2, which is stored in the matrix M in Figure 3.8as $a_{13} = 2$. The link which goes in the reverse direction (from the 3rd node to the 1st node) is ascribed the cost of 3, this is stored as $a_{31} = 3$. If there was a need to represent the three links between 1st and 3rd nodes from the Figure 3.6, then we realize we have run out of places in the matrix and the network cannot be fully represented by a matrix.

There can be some improvement of the matrix representation envisaged for coping with such an extended network. One improvement is matrix of lists. An entry in this matrix would not characterize only one directed link from one node to another but a list of directed link from this node to another. However, this is not classified anymore as the matrix approach to the SP problem.

The Tree Building Algorithm

A tree building algorithm builds a tree with the root in the source node of the trip. Each node of the network can be either a leaf or a fork of the tree. A fork leads to another forks or leave. These are certain true statement about the tree. The first is that, there are p leaves, then, these leaves are p nodes of the biggest cost to reach among all nodes. The second says that, each fork node (a node that is a fork in the tree) is of the cost, smaller than a cost of any leaf node (a node that is a leaf in the tree).

Building such a tree is dynamic programming task since the result of a node just reached can be used to calculate the cost of the node which can be reached immediately after this node.

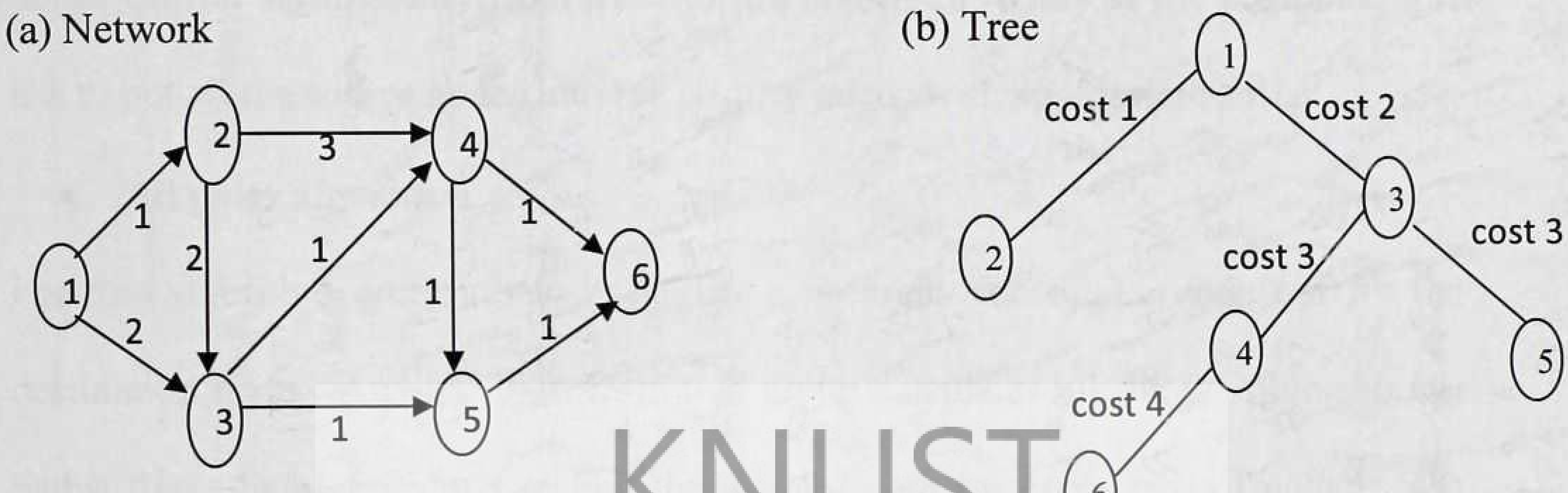


Figure 3.9: Example of Network and its tree.

3.3.2 INPUT AND OUTPUT TO THE SHORTEST PATH ALGORITHM

The shortest path algorithm can be divided into groups that differ by the given input and the desired output. The groups are; One pair algorithm, One – to –many, Many-to – one and All pairs algorithms.

- **One pair Algorithm**

There are two nodes given the source node and the destination node. A shortest algorithm finds only one shortest path (if it exist) from the given source node to the given destination node. The tree algorithms are going to build an incomplete tree with the root in the source node. The tree will be complete up to the moment the destination node has been reached. The Dijkstra's algorithm (1959) and Bellman algorithm (1958) are examples that are pair algorithms.

- **Many-to-one pair algorithms.**

This problem is given many source nodes and one destination node. To each source node there is time ascribed saying what time the journey starts from that node. The solution to the problem is to find the shortest path from any source node to the destination node that will result in reaching the destination node at the minimal time of arrival (not cost of the journey).

This type of a problem is easy to solve having the Dijkstra algorithm. The solution doesn't differ significantly from the Djikstra algorithm. Only at the beginning one has to put all the source nodes into the priority queue with appropriate costs.

- **All pairs algorithm**

For this algorithm group, there is neither a necessity for source node nor for the destination node. An algorithm from this group calculates all the possible shortest paths; thus, the algorithm is to find the shortest path for every pair of nodes. The number of path (the paths from one and the same node is 0 and doesn't require calculation) is therefore

$n^2 - n = n (n - 1)$. The computations are mostly done on matrices. The Floyd algorithm (1962) is an example from the all pairs algorithm group.

- **One-to-many algorithm**

Only the source node is specified. All shortest paths from this source node to all other nodes will be calculated. If there is a path from the source node to every other node, then there will be $(n - 1)$ shortest paths evaluated (n is the number of nodes in the network). A tree building algorithm will create a complete shortest path tree. The Dijkstra algorithm and the Bellman algorithm are also examples of one-to-many algorithms.

3.4 SOLUTIONS TO VEHICLE ROUTING PROBLEM.

The VRP is a Non- deterministic Polynomial time hard (NP – hard) problem, which is hard to solve in a polynomial time (Bodin et al., 1983). No optimal algorithm that can solve NP – hard problems in a polynomial time has been found (Falkenauer, 1996). Finding optimal solutions of NP – hard problem is usually very time consuming and sometimes even impossible. Due to this characteristic, it is not realistic to use optimal solution methods to solve large problems.

Branch and bound method has been applied to problems with small number of customers (Pereira et al., 2002), but not for large size, where computational limitation of memory buffers, and computing resources exists. Hence many other approaches based on heuristics approximation algorithm that aim at finding good feasible solutions quickly, have been introduced (Laporte et al., 2000; Prescott – Gagnon et al., 2009).

The heuristics can be classified into two kinds: Classical heuristics (1960 – 1990) and Modern heuristics (1990 -)

- Classical VRP heuristics can be broadly classified into three.

(i) ***Constructive Heuristics***

It gradually build a feasible solution while keeping an eye on solution cost, but do not contain an improvement phase per se. Examples of such heuristics are Clarke – Wright savings heuristics (1964), insertion heuristics and sequential verses parallel version.

(ii) ***In two – phase heuristics***, the problem is decomposed into its two natural components:

Clustering of vertices into feasible routes, and, actual route construction, with possible feedback loops between the two stages. Two – phase heuristics can be divided into two classes: ***Cluster first, route second methods*** and ***route first, cluster second methods***.

In the first case, vertices are first organized into feasible clusters, and a vehicle route is constructed for each of them.

In the second case, a tour is first built on all vertices and is then segmented into feasible vehicle routes.

(iii) ***Improvement methods*** attempt to upgrade any feasible solution by performing a sequence of edge or vertex exchanges within or between vehicle routes. The distinction between constructive and improvements methods is however often blurred since most constructive algorithms incorporate improvements steps (typically 2 or 3 – opt) at various stages.

Six main types of metaheuristics have been applied to the VRP.

- Simulated Annealing (SA) / Deterministic Annealing (DA)
- Tabu Search (TS) /Guided local search
- Variable Neighborhood search (VNS)
- Genetic Algorithm (GA) /Evolutionary methods
- Ant Colony Optimization (ACO)
- Artificial Neural Networks (ANN)

The SA, DA and TS, start from an initial solution x_1 and move at each iteration t from x_t to a solution x_{t+1} in the neighborhood $N(x)$ of x_t, until a stopping condition is satisfied. If $f(x)$ denotes the cost of x, the $f(x_t + 1)$ is not necessarily less than $f(x_t)$. As a result, care must be taken to avoid cycling. GA examines at each step a population of solutions. Each population is derived from the preceding one by combining its best elements and discarding the worst. ASA is a constructive approach in which several new solutions are created at each iteration using some of the information gathered at previous iterations. As was pointed out by Taillard et al (1995), TS,GA and SA are methods that record as the search proceeds, information on solutions encountered and use it to obtain improved solution. ANN is learning mechanism that gradually adjusts weights until an acceptable solution is reached. The rules governing the search differ in each case and these must also be tailored to the shape of the problem at hard. Also, a fair amount of creativity and experimentation is required.

3.5 IMPROVEMENT/ FEASIBILITY ROUTINE

There are improvement routines for VRP (Salhi and Rand, 1993). Most of them can be used both to decrease the amount of infeasibility for each route. Some of these routines are:

- Routine REVERSE: This is a new routine which is based on the observation that reversing the direction of a route may improve its length. The procedure simply chooses the route with the smaller maxload.
- Routine 2- OPT : This method, first described by Lin (1965) is based on interchanging arcs say ***ab*** and ***cd*** with ***ac*** and ***bd***. The direction of the routine will be reversed between customers, ***b*** and ***c***. We summarized that, infeasibilities in VRP occur due to the fact that the customers are in the wrong order on the vehicle route. Thus, it is reasonable to re-arrange the ordering of the customers on the route by reversing the direction of those parts of the vehicle route, where infeasibilities occur. We note that applying this transformation twice returns the route to its original state, thus 2- OPT can be thought of as its own dual. It may be used either to decrease route length or to

reduce the occurrence of infeasibilities. This observation is true for most of the subsequent routines.

- Routine 3- OPT : This is a slight modification of the original routine of Lin (1965) which is based on the exchange of any three arcs with three other arcs. The modified method considers only three consecutive arcs and hence is much faster.
- Routine SHIFT : This routine involves two routes, but is otherwise very similar to our version of 3- OPT. It involves the deletion of a customer from a route and its insertion into an arc on another route.
- Routine EXCHANGE : This routine is an extension to shift in that two customers are inserted simultaneously into each other's former routes, but not necessarily into the former locations of each other.
- Routine REINSERT: This method originates from a theorem by Mosheior (1994). The author proved that a weakly feasible travelling salesman tour with pickups and deliveries can always be made strongly feasible by re-inserting the depot. Thus routine REINSERT considers all arcs on a tour for possible depot re-insertion. Mosheior (1994) showed just how useful this routine can be in eliminating infeasibilities; however it is noted that, it is also possible that re-inserting the depot would decrease the route length.

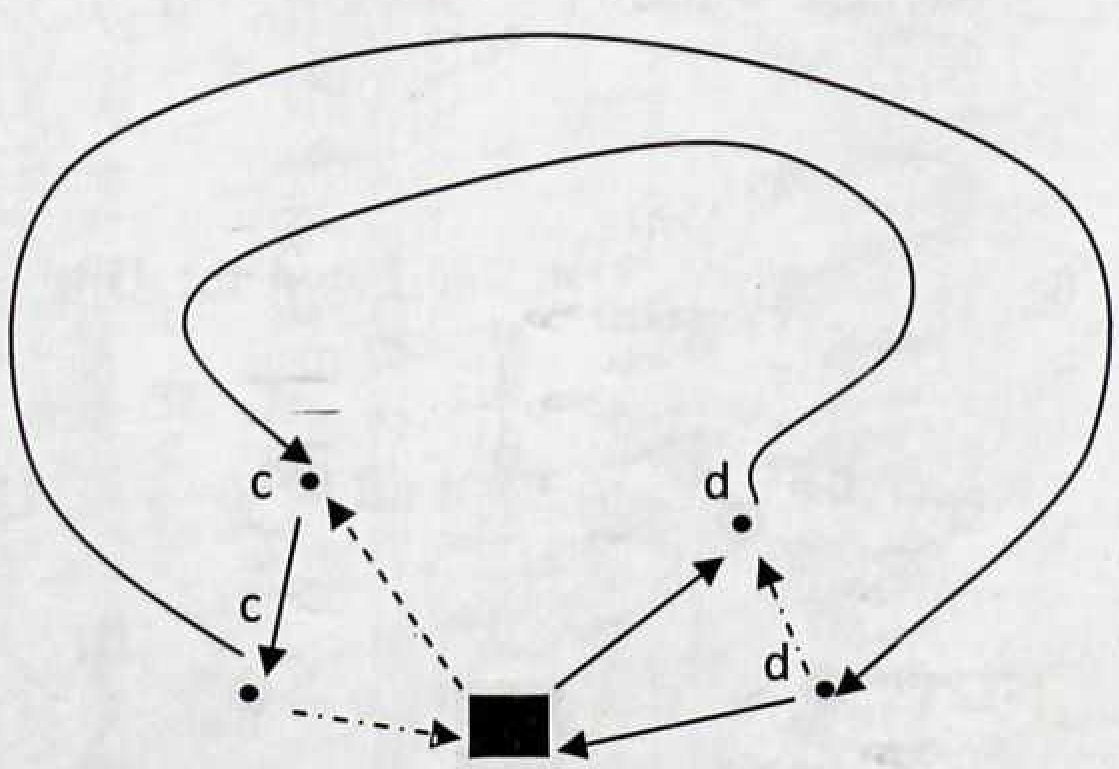


Figure 3.10: Routine reinsert

3.6 FORMULATIONS OF VRP's

3.6.1 MODEL FORMULATION FOR SINGLE DEPOT VRP.

An integer programming formulation of the single-depot vehicle routing problem.

The data of the problem are as follows: There are m vehicles. The capacity of vehicle h is equal to Q_h ($h = 1, \ldots, m$). The depot is indexed by $i = 1$ and the customers by $i = 2, \ldots, n$: the demand of customer i is equal to q_i ($i = 2, \ldots, n$). Finally, there is a matrix $(C_{ij})_{i,j=1}^{m}$ of travel times.

As to the decision variables let,

$$y_{hi} = \begin{cases} 1, & \text{if vehicle } h \text{ visits customer } i \\ 0, & \text{otherwise} \end{cases}$$

$$X_{hij} = \begin{cases} 1, & \text{if vehicle } h \text{ visits customer } i \text{ and } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}$$

This problem is

$$\text{Minimize} \sum_{h=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{hij}$$

subject to

$$\sum_{h=1}^{m} y_{hi} = \begin{cases} m, & for \ i = 1 \\ 1, & for \ i = 2, \ldots, n \end{cases} \quad (3.1)$$

$$\sum_{h=1}^{m} q_i y_{hi} \leq Q_h \text{for } h = 1, \ldots, m \quad (3.2)$$

$$y_{hi} \in \{0, 1\} \qquad \text{for } h = 1, \ldots, m, i = 1, \ldots, n \quad (3.3)$$

$$\sum_{j=1}^{n} X_{hij} = \sum_{j=1}^{n} X_{hji} = y_{hi} \text{ for } h = 1, \ldots, m, \ i = 1, \ldots, n \quad (3.4)$$

$$\sum_{i,j \in u} X_{hij} \leq |U| - 1 \qquad \text{for } h = 1, \dots, m, \qquad U \subset \{2, \dots, n\} \tag{3.5}$$

$$X_{hij} \in \{0, 1\} \quad \text{for } h = 1, \dots, m, \qquad i, j = 1, \dots, n \tag{3.6}$$

The conditions (3.1) ensure that each customer is allocated to one vehicle and that the depot is allocated to each vehicle. The conditions (3.2) are the vehicle capacity constraints. The conditions (3.4) ensure that a vehicle which arrives at a customer also leaves that customer. The conditions (3.5) are the subtour elimination constraints. This formulation is due to Fisher and Jaikumer (1981).

3.6.2 THE CLASSICAL VEHICLE ROUTING PROBLEM FORMULATION

The formulation of the VRP can be presented as follows (Laporte, 1992);

Let x_{ij} be an integer variable which may take value $\{0, 1\}$, $\forall \{i, j\} \in E \backslash \{\{0, j\} : j \in V$ and value $\{0, 1, 2\}$, $\forall \{0, j\} \in E, j \in V$. Note that $x_{oj} = 2$ when a route including the single customer j is selected in the solution.

The VRP can be formulated as the following integer programming:

$$\text{Minimize} \sum_{i \neq j} d_{ij} X_{ij} \tag{3.7}$$

subject to

$$\sum_{j} X_{ij} = 1, \qquad \forall\, i \in V, \tag{3.8}$$

$$\sum_{i} X_{ij} = 1 \qquad \forall\, j \in V, \tag{3.9}$$

$$\sum_{i} X_{ij} \geq |s| - V(s), \qquad \{s : s \subseteq V\, \{1\}, |s| \geq 2\} \tag{3.10}$$

$$X_{ij} \in \{0, 1\}, \forall\, \{i, j\} \in E;\ i \neq j \tag{3.11}$$

In this formulation (3.7), (3.8), (3.9) and (3.11) defined a modified assignment problem (i.e assignments on the main diagonal are prohibited).Constraints (3.10) are sub-tour elimination constraints: V(s) is an appropriate lower bound on the number of vehicles required to visit all vertices in the optimal solution.

3.6.3 CAPACITATED VEHICLE ROUTING PROBLEM (CVRP) FORMULATION

The CVRP formulation proposed by Laporte et al., (1985) is then,

$$\text{Minimize} \sum_{e \in E} d_e X_e \tag{3.12}$$

subject to

$$\sum_{e \in \delta(i)} X_e = 2 \tag{3.13}$$

$$\sum_{e \in \delta(0)} X_e = 2m, \tag{3.14}$$

$$\sum_{e \in \delta(s)} X_e \geq 2r(s), \qquad s \subseteq V\,\{0\}, s \neq \emptyset \tag{3.15}$$

$$X_e \in \{0, 1\}, \qquad e \notin \delta(0) \tag{3.16}$$

$$X_e \in \{0, 1, 2\}, \quad e \in \delta(0) \tag{3.17}$$

The degree constraints (3.13) states that each customer is visited exactly once, whereas the depot degree constraints (3.14) means that m routes are created. Capacity constraints (3.15) impose both the connectivity of the solution and the vehicle capacity requirements by forcing a sufficient number of edges to enter each subset of vertices. We note that since the Bin – Packing problem (BPP) is NP- hard in the strong sense, $r(s)$ may be approximated from below by any BPP lower bound such as $\left[\sum_{i\in s} q_i/Q\right]$.

Finally, constraints (3.16) and (3.17) impose that each edge between two customers is traversed at most once and each edge incident to the depot is traversed at most twice.

A widely used alternative formulation is based on the set partitioning or set covering models. The formulation was originally proposed by Balinski and Quandt (1964) and contains a potentially exponential number of binary variables.

Let $R = \{R1, \ldots, Rs\}$ denote the collection of all feasible routes, with $s = |R|$. Each route R_j has an associated cost γ_j and a_{ij} is a binary coefficient equal to 1 if and only if vertex I is visited (i.e. covered) by route R_j. The binary variable X_j, $j = 1, \ldots, s$ is equal to1 if and only if route R_j is selected in the solution.

The model is;

Minimize $\sum_{j=1}^{s} \gamma_j X_j$(3.18)

Subject to

$$\sum_{j=1}^{s} a_{ij} X_j = 1, \quad i = V\{0\} \tag{3.19}$$

$$\sum_{j=1}^{s} X_j = m \tag{3.20}$$

$$X_j \in \{0, 1\}, j = 1, \ldots, s \tag{3.21}$$

Constraints (3.19) imposed that each customer i is covered by exactly one route, and (3.20) requires that m r outes be selected. Because route feasibility is implicitly considered in the definition of R, this is a general model which may easily take additional constraints into account. Moreover, when the cost matrix satisfies the triangle inequality (ie $C_{ij} \leq C_{ik} + C_{kj}$ for all $i, j, k, \in V$), the set partitioning model may be transformed into an equivalent set covering model by replacing equality sign with "≥ " in (3.19). Any feasible solution to the set partitioning model may be transformed into feasible solution of smaller or equal cost.

Fukasawa et al., (2004) presented the capacitated vehicle Routing problem formulation as follows,

Let $H = (N, A)$, d, q and Q defined a CVRP instance having vertex **O** as the depot and the remaining vertices in N as clients/ customers.

$$\text{Minimize} \sum_{e=(u,v)\in A} d(e)X_e \tag{3.22}$$

subject to

$$\sum_{e=\delta(\{u\})} X_e = 2 \qquad \forall u \in N\{0\} \tag{3.23}$$

$$\sum_{e=\delta(\{0\})} X_e \geq 2k^* \tag{3.24}$$

$$\sum_{e=\delta(s)} X_e \geq 2k(s), \quad \forall s \in N\{0\} \tag{3.25}$$

$$X_e \leq 1, \qquad \forall e \in A \setminus \delta(\{0\}), \tag{3.26}$$

$$\sum_{l=1}^{p} q_1^e \lambda_1 - X_e = 0, \quad \forall e \in A \tag{3.27}$$

$$X_e \in \{0, 1, 2\}, \qquad \forall e \in A, \tag{3.28}$$

$$\lambda_1 \geq 0, \qquad \forall l = \{1, \ldots\ldots, p\} \tag{3.29}$$

X_e represents the number of times that edge e is traversed by a vehicle. This variable can assume value 2 if e is adjacent to the depot, corresponding to a route with a single customer/client. $\lambda_l$variables would ideally be associated with valid routes. This would imply having a strongly NP –hard column generation problem. $\lambda_l$variables are associated to the set of all q – routes satisfying the vehicle capacity constraint. A q - route is a walk that starts at the depot, traverses a sequence of clients/customers with total demand at most Q, and returns to the depot. Clients may appear more than once in a q - route and its demand considered for each time. Each variable λ_l is therefore associated to one of the P possible q - routes.

Degree constraint (3.23) states that each client/customer vertex is served by exactly one vehicle. Constraint (3.24) requires that at least k^* vehicles leave and return to the depot. This number, representing the minimum number of vehicles to service all clients/customers is calculated by solving a Bin-Packing Problem (BPP). The rounded capacity constraints stated in (3.25) use

$k(s)$

$= \left[\sum_{u \in s} q(u)/Q \right]$ as a lower bound on minimum number of vehicles necessary to service the clients/customers in set $s \subset N$. Constraints (3.27) oblige X to be a linear combination of q – routes. The total constraints complete the formulation.

3.7 FLOYD'S ALGORITHM

Floyd's Algorithm (also known as Floyd-Warshall or Roy-Warshall algorithm) is a graph analysis algorithm for finding shortest paths in a weighted graph (with positive or negative edge weight) and also for finding transitive closure of a relation R. A single executive of the algorithm will find the lengths of the shortest paths between all pairs of vertices. The algorithm is an example of dynamic programming.

Floyd's algorithm compares all possible paths through the graph between each pairs of vertices. It is able to do this by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal. The Floyd's algorithm assumes that there are no negative cycles. Nevertheless, if there are negative cycles the Floyd's algorithm can be used to detect them in the following ways:

- The Floyd's algorithm iteratively revises path lengths between all pairs of vertices (i, j), including where $i = j$;
- Initially, the length of the path (i, j) is zero;
- A path $\{(i, k)\ (k, j)\}$ can only improve upon this if it has length less than zero, i.e. denites a negative cycle.
- Thus, after the algorithm, (i, j) will be negative if there exist a negative-length path from i back to j.

<u>Pseodocode</u>

```
/* Assume a function edgeCost (i, j) which returns the cost of the edge from
i to j
(infinity if there is none)
Also assume that n is the number of vertices and edgeCost (i, j) = 0
*/
intpath[ ][ ]
/* A 2-dimensional matrix. At each step in the algorithm, path[i][j] is the
shortest
```

```
path from i to j using intermediate vertices (1, k - 1). Each path
[i][j] is initialized to edgeCost (i, j).
*/
procedure FloydWarshall ( )
for k: 1 to n
for i: 1 to n
for j: 1 to n
path [i][j] = min (path [i][j], path [i][k]  + path [k][j]);
```


The Floyd algorithm typically only provides the length of the paths between all pairs of vertices. With simple modifications, it is possible to create a method to reconstruct the actual path between any two endpoint vertices. While one may be inclined to store the actual path from each other vertex to other vertex, this is not necessary and in fact, is very costly in terms of memory.

For each vertex, one need only store the information about the highest index intermediate vertex one has to go through if one wishes to end up at any given vertex. Therefore, information to reconstruct all paths can be stored in a single N×N matrix 'next' where next [i][j] represents the highest index vertex one must travel through if one intends to take the shortest path from *i* to *j*. The modified algorithm is as below.

```
procedure FloydWarshallWithPatReconstruction ( )
  for k: 1 to n
      for i: 1 to n
          for j: 1 to n
              if path [i][k]  +  path [k][j] < path [i][j] then
```

```
                    path [i][j] : = path [i][k] + path [k][j];
                    next [i][j] : = k;
    procedure GetPath (i, j)
            if path [i][j] equals infinity then
            return   "no path ";
    int intermediate: = next [i][j]
        if intermediate equals 'null 'then
        return " " ;  /* there is an edge from i to j, with no vertices between  */
    else
        return GetPath (i, intermediate) + intermediate +GetPath (intermediate, j)
```



```
Floyd-Warshall(W, n)
{fori = 1 to n  do
        for j = 1 to n  do
                { d[i, j] = W[i, j];
                        Pred [i, j] = nil;
                }
fork = 1 to n  do
        for i = 1 to n   do
                for j = 1 to n   do
                if(d[i, k] + d[k, j] < d[i, j])
                { d[i, j] = d[i, k] + d[k, j]
                pred[i, j] = k;
                }
returnd[1…. n, 1…. n];
}
```

3.8 BELLMAN-FORD ALGORITHM

Bellman-Ford algorithm computes single source shortest path problem in the general case in which edges of a given digraph can have negative cycles. This algorithm , like Dijkstra's algorithm uses the notation of edge relaxation but does not use with greedy method. Again it uses d[u] as an upper bound on the distance d[u, v] from u to v.

The algorithm progressively decreases an estimate d[v] on the weight of the shortest path from the source vertex *s* to each vertex v in V until it achieve the actual shortest path. The algorithm returns Boolean TRUE if the given digraph contains no negative cycles that are reachable from source vertex *s* otherwise it returns Boolean FALSE.

Pseodocode of Bellman-Ford Algorithm

```
Procedure Bellmanford (list vertices, list edges, vertex source)
// This implementation takes in a graph, represented as lists of vertices and edges,
// and modifies the vertices so that their distance and predecessor attributes store
// the shortest paths.
// Step 1: initialize graph
        for each vertex v in vertices :
                if v is source then v. distance : = 0
                        elsev. distance : = infinity
                                v. predecessor: = null
// Step 2: relax edges repeatedly
        for i from 1 to size (vertices) - :
                for each egde uv in edges : // uv is edge from u to v
```

```
            u: = uv. source
            v: uv. destination
        if u. distance + uv. Weight < v. distance
        v. distance: = u. distance + uv. weight
        v. predecessor : = u
// Step 3:  check for negative – weight cycles
        For each edge uv in edges
        u: = uv. source
        v: = uv. destination
        if u. distance + uv. Weight < v. distance :
        error " Graph contains a negative – weight cycle "
```

Bellman-ford algorithm

BELLMAN-FORD (G, *w*, *s*)

```
INITIALIZE-SINGLE-SOURCE (G, s)
for each vertex i = 1 to V[G] - 1 do
    for each edge (u, v) in E[G] do
        RELAX (u, v, w)
For each edge (u, v) in E[G] do
    if d[u] + w(u, v) < d[v] then
        return FALSE
return TRUE
```

3.9 DIJKSTRA'S ALGORITHM

Dijkstra's algorithm reduces the amount of computational time and power needed to find the optimal path. The algorithm strikes a balance by calculating a path which is close to the optimal path that is computationally manageable (Olivera, 2000). The algorithm breaks the network into nodes (where lines join, start or end) and the paths between such nodes are represented by lines.

In addition, each line has an associated cost representing the cost (length) of each line in order to reach a node. There are many possible paths between the origin and destination, but the path calculated depends on which nodes are visited and in which order. The idea is that, each time the node, to be visited next is selected after a sequence of comparative iterations during which, each customer node is compare with others in terms of cost (Stewart, 2004).

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

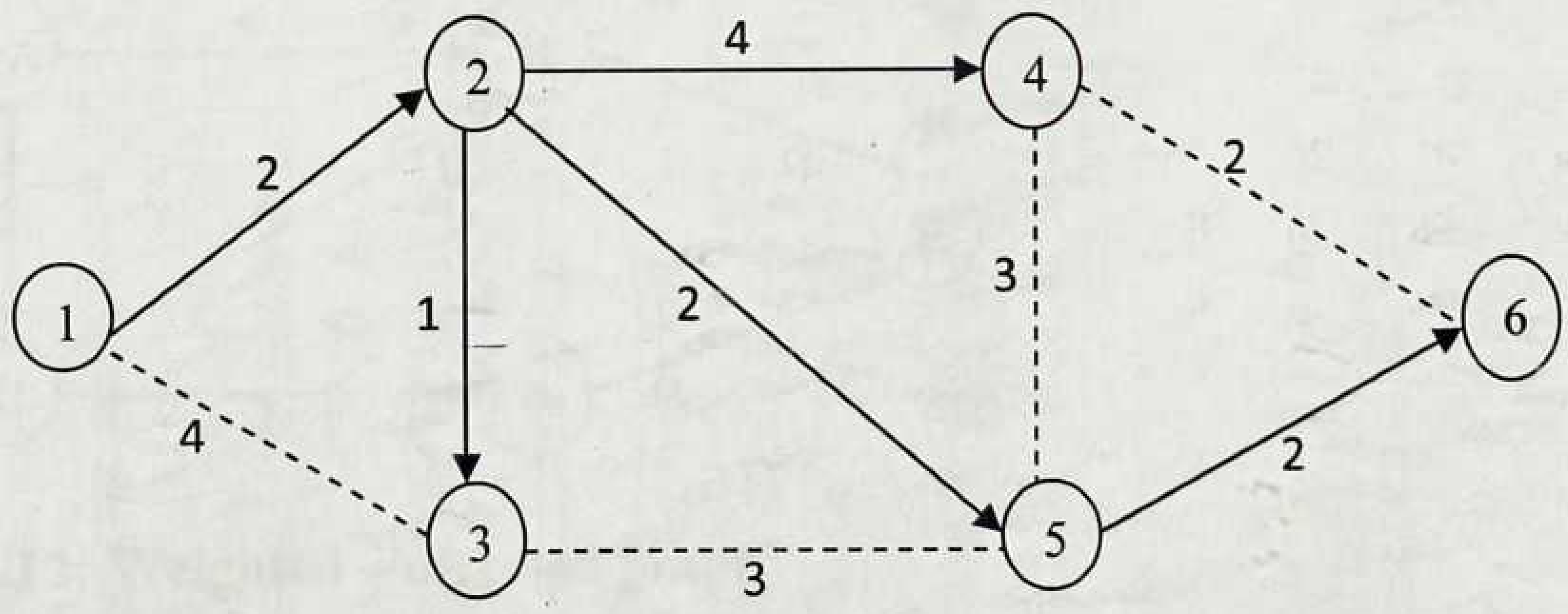


Figure 3.11: An example of Dijkstra's algorithm **(Orlin, 2003)**

Dijkstra's algorithm works by visiting nodes in the network starting with the objects start node and iteratively examining the closest not- yet- examined node. It adds its successors to the set of nodes to be examined and thus divides the graph into two sets: S, the nodes whose shortest path to the start node is known and S^1, the nodes whose shortest path to the start node is unknown. Initially, S^1 contains all of the nodes. Nodes are then moved from S^1 to S after examining and thus the node set, S, "grows". At each step of the algorithm, the next node added to S is determined by a priority queue. The queue contains the nodes S^1, prioritized by their distance label, which is the cost of the current shortest path to the start node. This distance is also known as the start distance. The node *u*, at the top of the priority queue is then examined, added to S, and its out- links are relaxed. If the distance label of *u* plus the cost of the out- link (*u*, *v*) is less than the distance label for *v*, the estimated distance for node *v* is updated with this value. The algorithm then loops back and processes the next node at the top of the priority queue. The algorithm terminates when the goal is reached or the priority queue is empty. Dijkstra's algorithm can solve single source SP problems by computing the one- to- all shortest path trees from a source node to all other nodes.

Consider the example below:

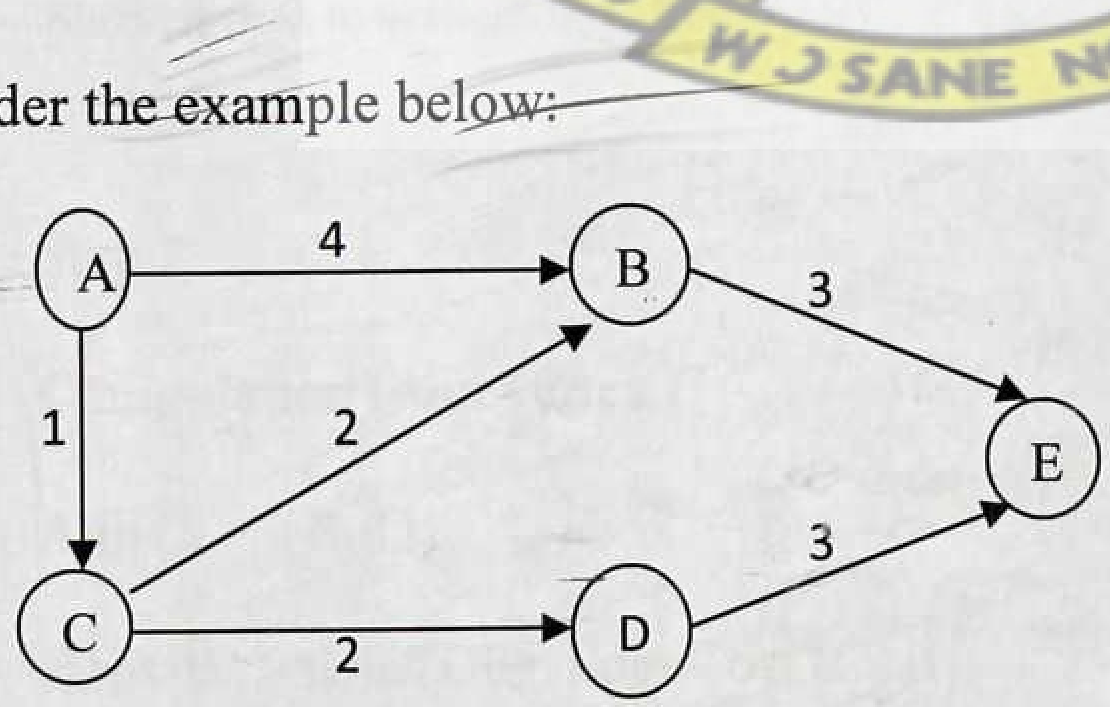


Figure 3.12: Weighted – directed graph

The above weighted graph has 5 vertices from A – E. The value between the two vertices is known as the edge cost between two vertices. For example the edge cost between A and C is 1. Using the above graph the Dijkstra's algorithm is used to determine the shortest path from the source A to the remaining vertices in the graph. This is solved as follows

- Initial step

  sDist[A] = 0; the value to the source itself

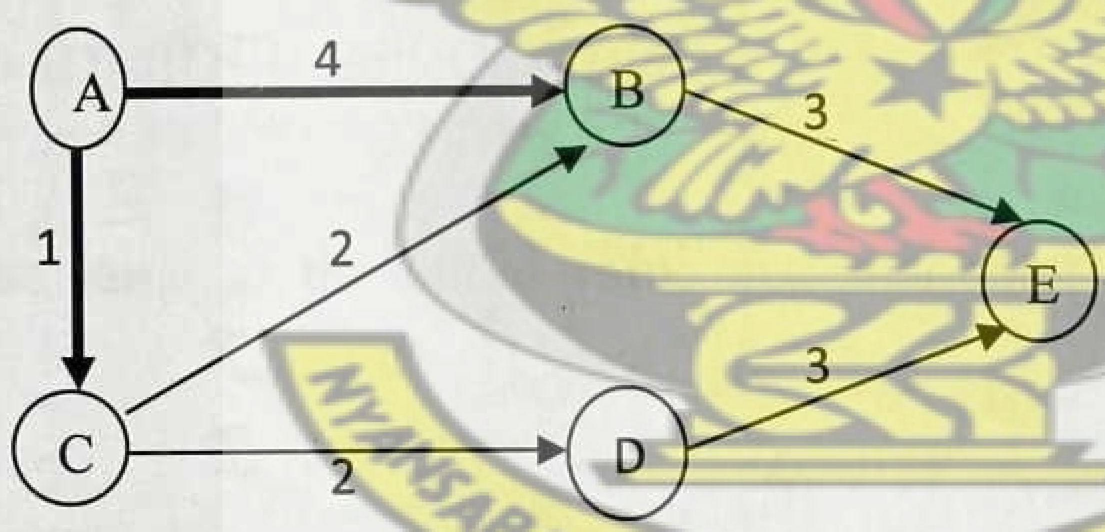
  sDist[B] = ∞, sDist[C] = ∞, sDist[D] = ∞, sDist[E] = ∞; the nodes not processed yet

- Step 1

  Adj[A] = {B, C}; computing the value of the adjacent vertices of the graph

  sDist[B] = 4;

  sDist[C] = 2;



Shortest path to vertices B, C from A

- Step 2

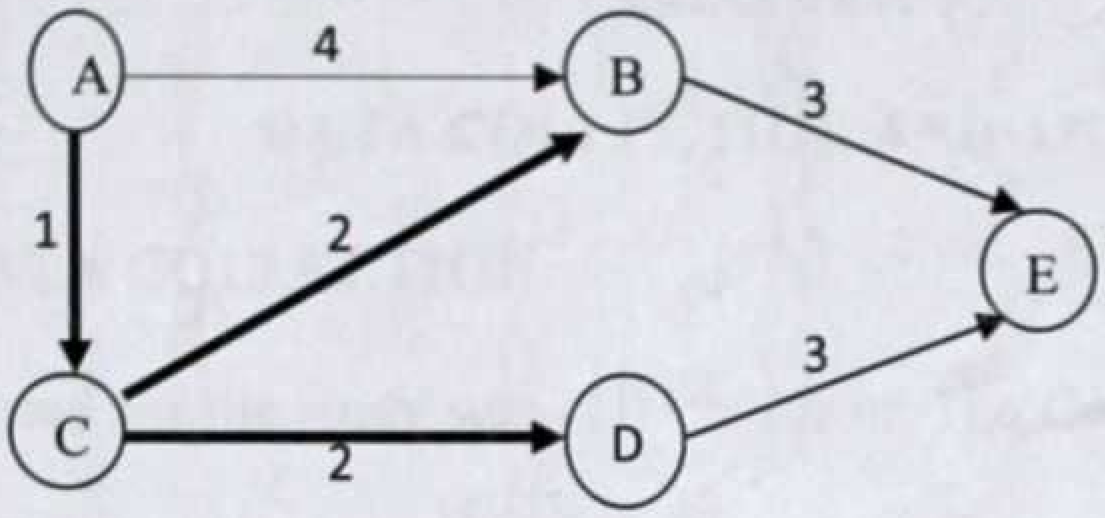
  Computation from vertex C

  Adj[C] = {B, D}

  sDist[B] >sDist[C] + EdgeCost [C, B]
  4 > 1 + 2 (True)

  Therefore, sDist[B] = 3;

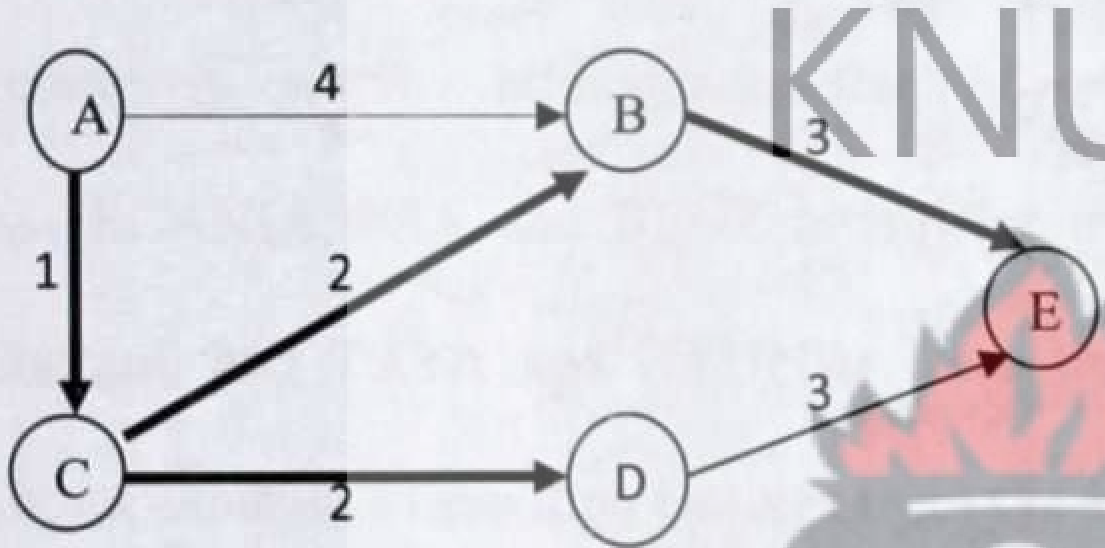
  sDist[D] = 2;



Shortest path from B, D using C as intermediate vertex

Adj[B] = {E};

sDist[E] = sDist[B] + EdgeCost[B, E]
= 3 + 3 = 6



Shortest path to E using B as intermediate vertex

Adj[D] = {E}

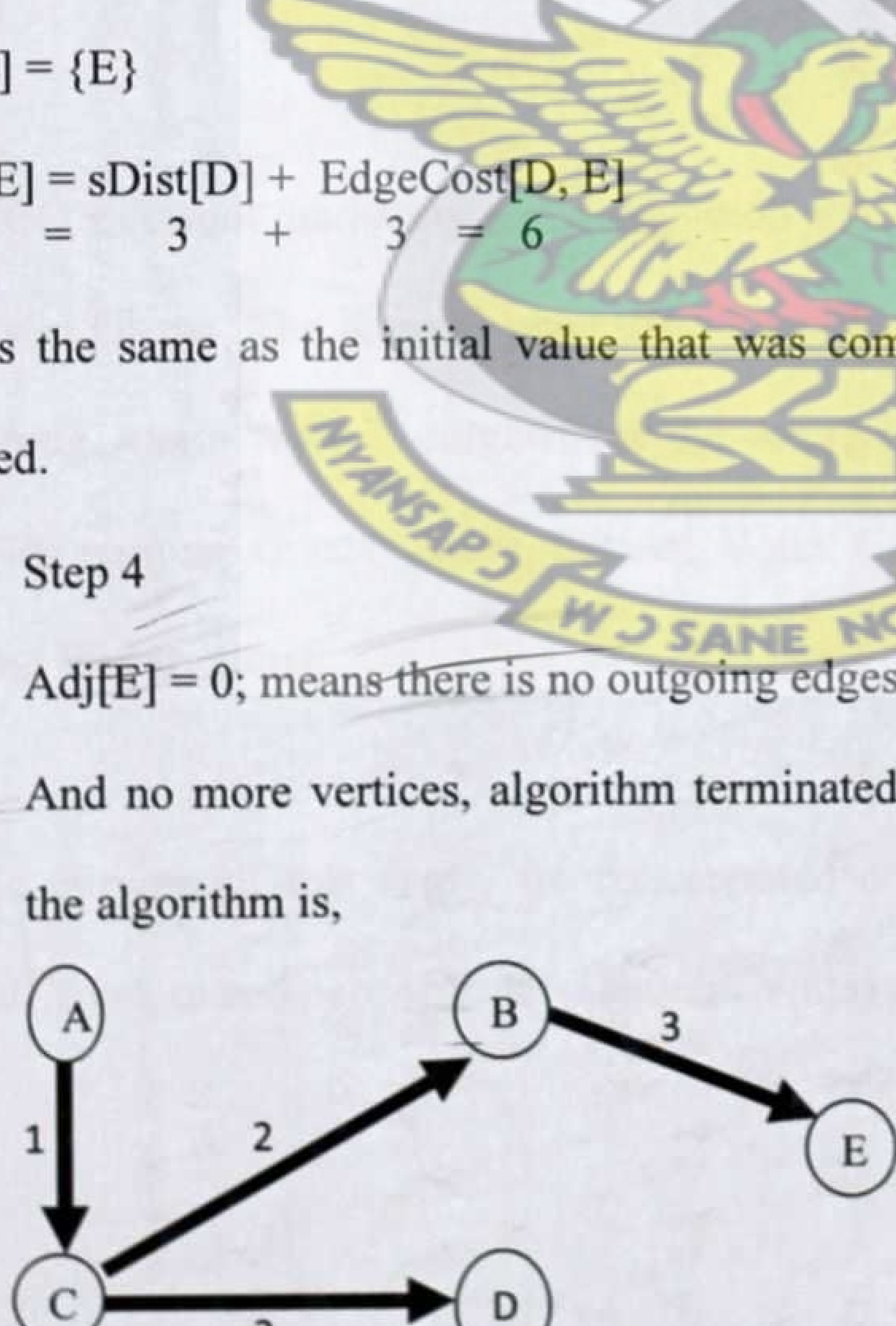
sDist[E] = sDist[D] + EdgeCost[D, E]
= 3 + 3 = 6

This is the same as the initial value that was computed so sDist[E] value is not changed.

- Step 4

  Adj[E] = 0; means there is no outgoing edges from E

  And no more vertices, algorithm terminated. Hence the path which follows the algorithm is,



The path obtained using Dijkstra's algorithm

CHAPTER 4

DATA COLLECTION AND ANALYSIS

4.1 DATA COLLECTION

The data used for the study was collected from The Coca-cola Bottling Company of Ghana Limited; Accra main Depot. Distances between the key distributors, thus Mini-Depot operators (MDO's) and Depot were collected from the sales session of TCCBCG. Distances between the MDO's was found using Google maps.

The company operates homogenous fleet of vehicles. The types of vehicle used include SCANIA, KIA and BENZ articulator trucks (Long Haulage) for external distribution, and TATA and HYUNDAI ace trucks for internal distribution. The maximum capacity of the long haulage trucks is 1512 and that of the Tata ace trucks is 450.

TCCBCG's current marketing and distribution activities are organized in all the ten regions in Ghana. The Kumasi plant distributes the company's products in Ashanti and Brong Ahafo regions only whiles the Accra plant distributes products to the following regions; Greater Accra, Eastern, Volta, Central, Western, Northern, Upper East and West regions.

For the purpose of this study, we concentrated on the customers within Eastern, Central, Western and part of Volta (southern Volta).

4.2 DATA ANALYSIS

The table below indicates the destination of the MDO's, their distances from the depot and the quantities of crates of soft drinks supplied per delivery. The total capacity of a full loaded truck is 1512crates.

Table 4.1: Destination of Customers and their distance in kilometres from the Accra Depot

CODE	LOCATION OF DISTRIBUTOR (CUSTOMER)	DISTANCE OF DISTRIBUTOR FROM DEPOT(km)	AVERAGE CRATES OF DRINKS SUPPLIED/ DEMANDED
D0	Accra	0	0
D1	Winneba	66	270
D2	Mankessim	112	260
D3	Cape Coast	149	500
D4	Takoradi	228	600
D5	Esiama	300	260
D6	Tarkwa	293	305
D7	Bogoso	313	280
D8	Samreboi	385	260
D9	Enchi	429	420
D10	Asankragua	416	305
D11	Bawdie	338	265
D12	WassaEkuropon	323	230
D13	TwifoPraso	224	245
D14	AssinFoso	190	260
D15	Kade	116	200
D16	Oda	130	400
D17	Swedru	85	465
D18	Asamankese	79	350
D19	Suhum	67	300
D20	Nkawkaw	147	370
D21	AkimTafo	103	180
D22	Koforidua	91	650
D23	Somanya	77	240
D24	Atimpoku	82	270
D25	Ho	165	700
D26	Aflao	182	850

Table 4.2: Cost matrix for fuel (in gallons) consumption throughout the locations of twenty six key distributors

c_{ij}	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26
C0	0	12	19	29	40	58	56	54	72	74	72	58	62	37	32	20	25	15	14	12	28	18	15	13	15	32	35
C1	12	0	9	15	29	41	40	42	56	64	52	43	45	28	22	18	16	4	12	18	34	24	22	24	26	37	42
C2	19	9	0	6	25	32	31	32	47	55	43	34	36	19	13	22	17	10	19	25	35	34	29	30	32	44	50
C3	29	15	6	0	14	26	25	27	41	49	37	27	30	13	16	28	23	17	26	32	42	38	36	36	39	51	56
C4	44	29	25	14	0	12	11	17	32	37	28	19	21	19	30	42	36	30	39	45	55	31	49	50	52	64	70
C5	58	41	32	26	12	0	11	17	14	24	29	19	21	32	35	54	49	44	52	57	61	66	62	63	65	77	82
C6	55	40	31	25	11	11	0	5	21	29	17	8	12	18	24	40	35	41	41	47	50	56	52	64	64	76	81
C7	54	42	32	27	17	17	5	0	15	20	11	2	4	13	20	36	30	37	37	43	44	49	47	65	65	77	82
C8	72	56	47	41	32	14	21	15	0	8	5	32	14	28	31	50	45	52	53	59	54	64	64	78	80	92	97
C9	82	64	55	49	37	24	29	20	8	0	9	20	22	36	42	58	53	59	61	67	55	73	71	85	87	99	104
C10	72	52	43	37	28	29	17	11	5	9	0	28	11	24	31	46	41	47	49	55	47	58	59	74	76	88	93
C11	65	43	34	27	19	19	8	2	32	20	28	0	4	15	21	37	32	38	38	44	42	52	48	64	67	79	84
C12	62	45	36	30	21	21	12	4	14	22	11	4	0	17	24	39	38	40	46	52	39	58	56	69	69	81	86
C13	43	28	19	13	19	32	18	13	28	36	24	15	17	0	6	22	17	23	25	31	36	37	35	50	52	64	69
C14	36	22	13	16	30	35	24	20	31	42	31	21	24	6	0	16	10	17	19	24	32	30	29	47	44	58	64
C15	22	18	22	28	42	54	40	36	50	58	46	37	39	22	16	0	5	16	6	12	14	18	16	24	24	45	50
C16	25	16	17	23	36	49	35	30	45	53	41	32	38	17	10	5	0	11	8	14	18	20	18	26	29	47	52
C17	15	4	10	17	30	44	41	37	52	59	47	38	40	23	17	16	11	0	8	14	28	20	18	24	26	38	44
C18	14	12	19	26	39	52	41	37	53	61	49	38	46	25	19	6	8	8	0	6	20	12	10	18	19	39	44
C19	12	18	25	32	45	57	47	43	59	67	55	44	52	31	24	12	14	14	6	0	14	6	4	12	13	36	41
C20	28	34	35	42	55	61	50	44	54	55	47	42	39	36	32	14	18	28	20	14	0	11	16	23	26	36	55
C21	19	24	34	38	31	66	56	49	64	73	58	52	58	37	30	18	20	20	12	6	11	0	4	12	15	27	42
C22	17	22	29	36	49	62	52	47	64	71	59	48	56	35	29	16	18	18	10	4	16	4	0	7	9	21	36
C23	14	24	30	36	50	63	64	65	78	85	74	64	69	50	47	24	26	24	18	12	23	12	7	0	3	15	31
C24	16	26	32	39	52	65	64	65	80	87	76	67	69	52	44	24	29	26	19	13	26	15	9	3	0	12	28
C25	32	37	44	51	64	77	76	77	92	99	88	79	81	64	58	45	47	38	39	36	36	27	21	15	12	0	22
C26	35	42	50	56	70	82	81	82	97	104	93	84	86	69	64	50	52	44	44	41	55	42	36	31	28	22	0

The transportation cost of TCCBCG represents about 23% of the total production cost. The company has registered seven transporters who operate within the area understudy with 21 haulage trucks and 78 mini (5tons) trucks.

An average cost of GH¢6.02 is incurred in transporting product per km. The ratio of this amount to the truck load of 1512 crates was 3.98×10^{-3}. Therefore the unit cost in transporting products from the Depot to the MDO's is 3.98×10^{-3} multiply by the distances from the Depot to the MDO's.

4.3 MODEL FORMULATION

The problem was treated as VRP with Pick-up and Delivery since products need to be brought from the depot to the customers and crates of empty bottles be picked up from the customers and brought back to the depot.

THE MODEL ASSUMPTIONS

- Each customer has deterministic and constant demand
- There is no allocation between customers
- The system involves capacitated and identical vehicles
- The customers are located dispersedly
- The costs associated with this system are fixed order cost and transportation cost.

The mathematical notation and formulation are as follows; Let

v be set of vehicles, where $v \in \{1, 2, \ldots, V\}$

n be set of clusters, where $n \in \{1, 2, \ldots, N\}$

C_n be cost of assigning a vehicle to cluster n; $\forall n \in N$

U_n be maximum load that will have to be carried in cluster n

t_v be remaining capacity of each partially loaded vehicle v

$$X_{vn} = \begin{cases} 1, & \text{if vehicle } v \text{ assigned to cluster } n \\ 0, & \text{otherwise} \end{cases}$$

The objective function Z is given by

$$\text{Minimize} \quad Z = \sum_{v \in V} \sum_{n \in N} C_n X_{vn} \quad \ldots (1)$$

Subject to

$$\sum_{v \in V} X_{vn} = 1, \quad \text{for } n = 1, 2, \ldots, N \quad \ldots (2)$$

$$\sum_{n \in N} U_n X_{vn} \le t_v \quad \text{for } v = 1, 2, \ldots, V \quad \ldots (3)$$

$$X_{vn} \in \{0, 1\} \quad \text{for } n = 1, 2, \ldots N \text{ and } v = 1, 2, \ldots, V \quad \ldots (4)$$

Constraint (2) ensures that each cluster is assigned to exactly one vehicle whiles the constraint (3) ensures that the maximum load in a cluster does not exceed the capacity of the vehicle assigned to that cluster.

ROUTING PHASE

The first phase of the problem was a routing phase which consisted of a "giant" TSP tour. The tour began from the depot and considered the closest customer relative to the depot. Thus nearest neighbor algorithm was employed here. Therefore the next customer was then routed in this single route.

Though the nearest neighbor algorithm was used, at some instances, the tour was directed towards certain customers to avoid traversing a customer twice.

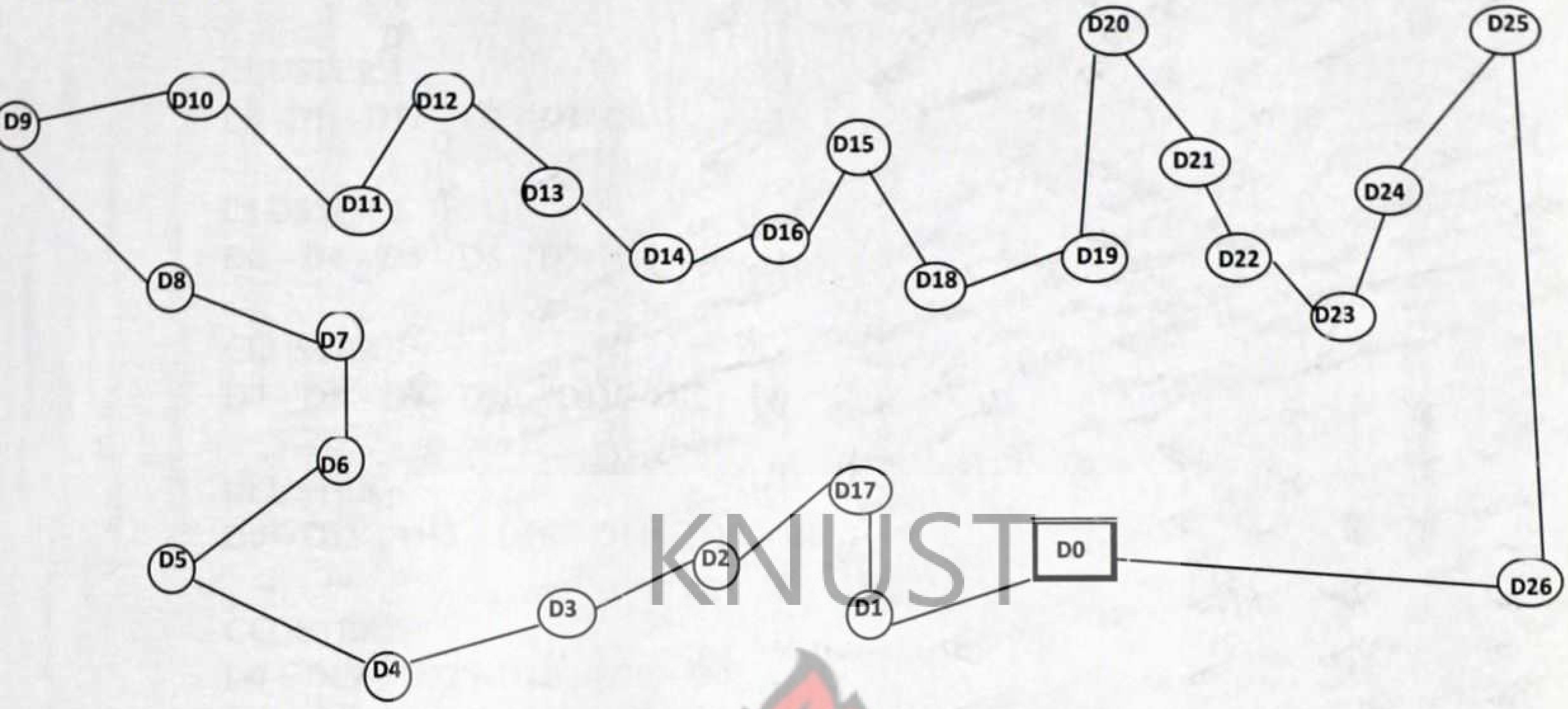


Figure 4.1: The resulted 'GIANT' TSP tour

The tour follows as shown below.

Accra → Winneba → Agona Swedru → Mankesim → Cape Coast → Takoradi → Esiama → Tarkwa → Bogoso → Samreboi → Enchi → Asankragua → Bawdie → Wassa Ekuropon → Twifo Praso → Assin Foso → Oda → Kade → Asamankese → Suhum → Nkawkaw → Akim Tafo → Koforidua → Somanya → Atimpoku → Ho → Aflao → Accra.

CLUSTERING PHASE

The clustering phase began with the customer in the first position. The delivery/ demand quantity of the second customer was added to that of the first and the next added until the quantity was less than or equal to the vehicle capacity. Those customers formed cluster. If the quantity was greater than the quantity of the vehicle, the next customer was considered.

This was repeated until all the customers were considered.

The resulted clusters

CLUSTER 1
D0 -D1 – D17 – D2 – D3- D0

CLUSTER 2
D0 – D4 – D5 – D6 – D7 – D0

CLUSTER 3
D0 – D8 – D9 – D10 – D11 – D12 – D0

CLUSTER 4
D0 – D18 – D15 – D16 – D14 – D13 – D0

CLUSTER 5
D0 – D19 – D22 – D21 – D20 – D0

CLUSTER 6
D0 – D23 – D24 – D25 – D0

CLUSTER 7
D0 – D26 – D0

4.4 COMPUTATIONAL PROCEDURE

The computations was done using Visual Basic dot net (VB.Net) for Dijkstra's algorithm version 1.0.0.0 (2012) which performed the traditional Dijkstra's algorithm to find the shortest distance between two nodes (Destinations of MDOs). Visual Basics dot Net with Dijkstra's algorithm provides efficient routing solution in a simple and straight forward manner.

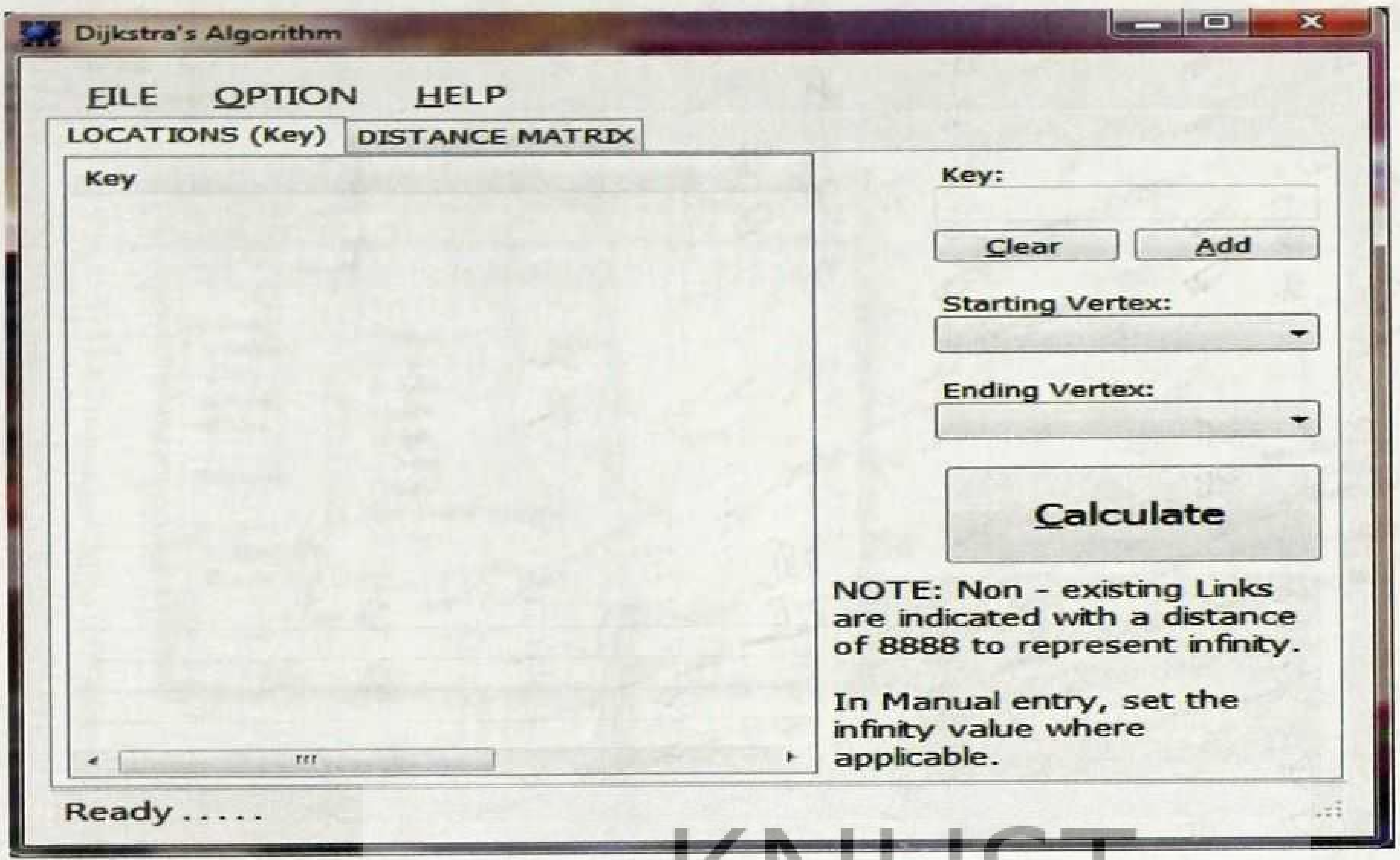


Figure 4.2: The first interface of the program.

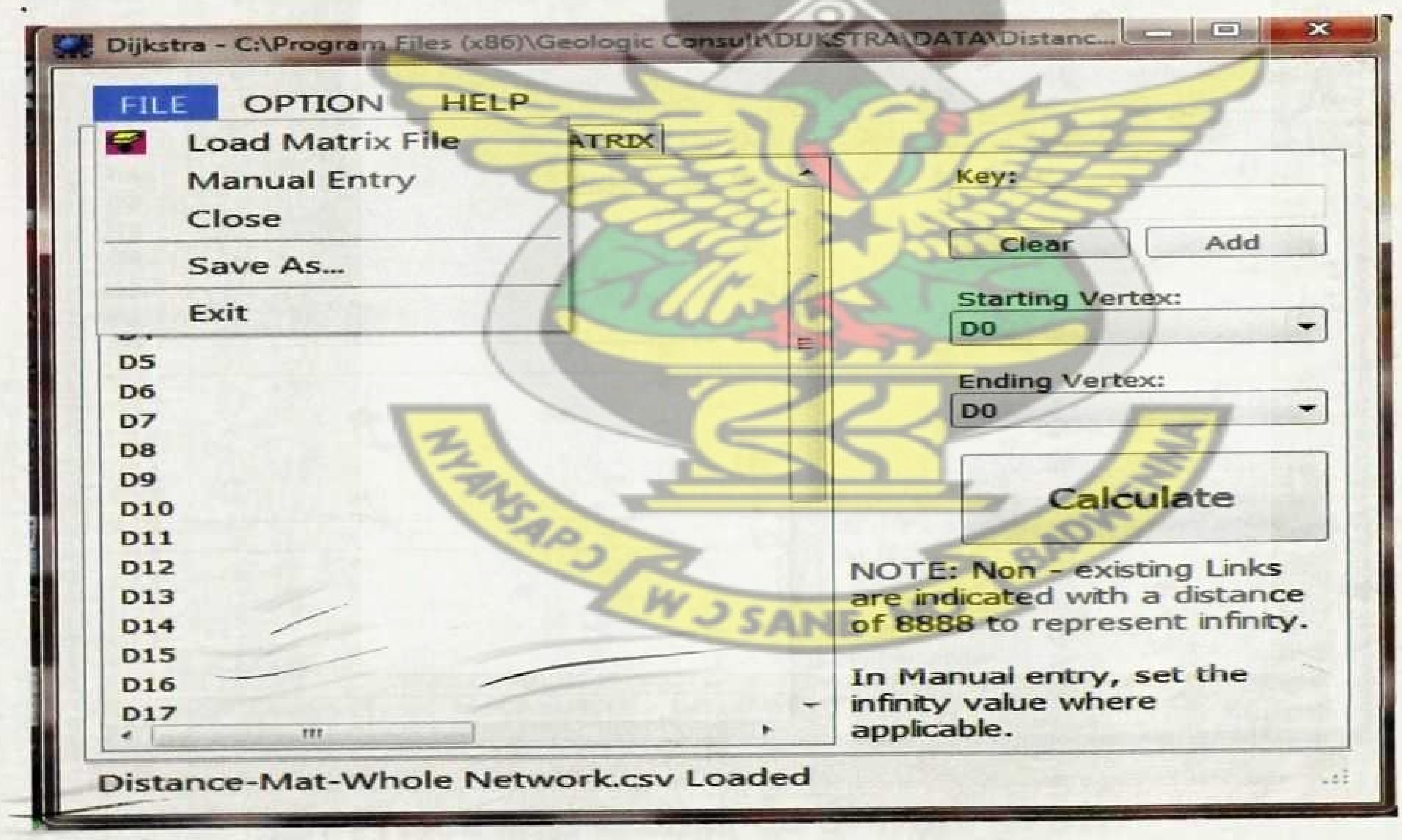


Figure 4.3: Shows how matrix files are loaded in the program

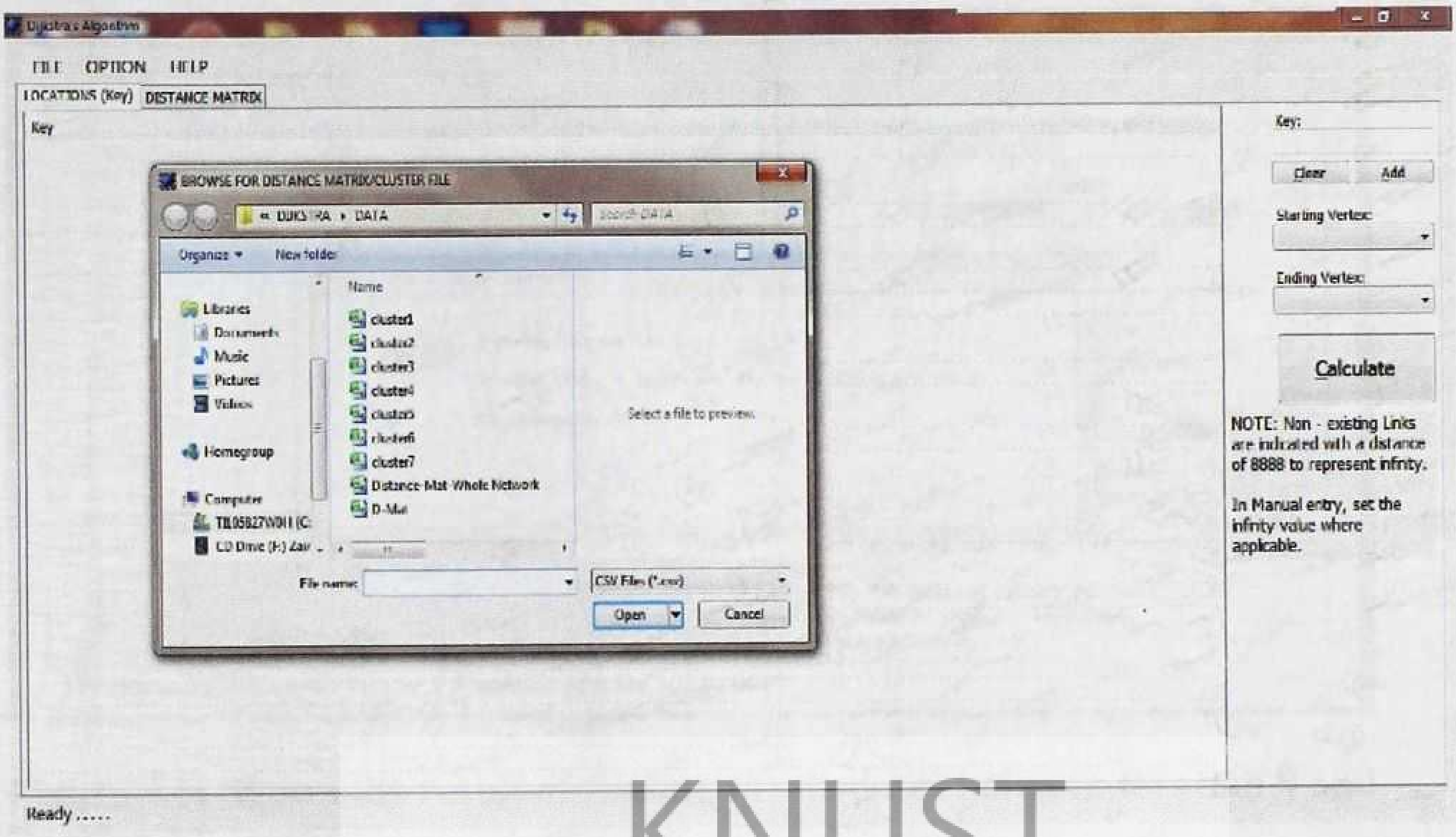


Figure 4.4: Shows how the files are selected

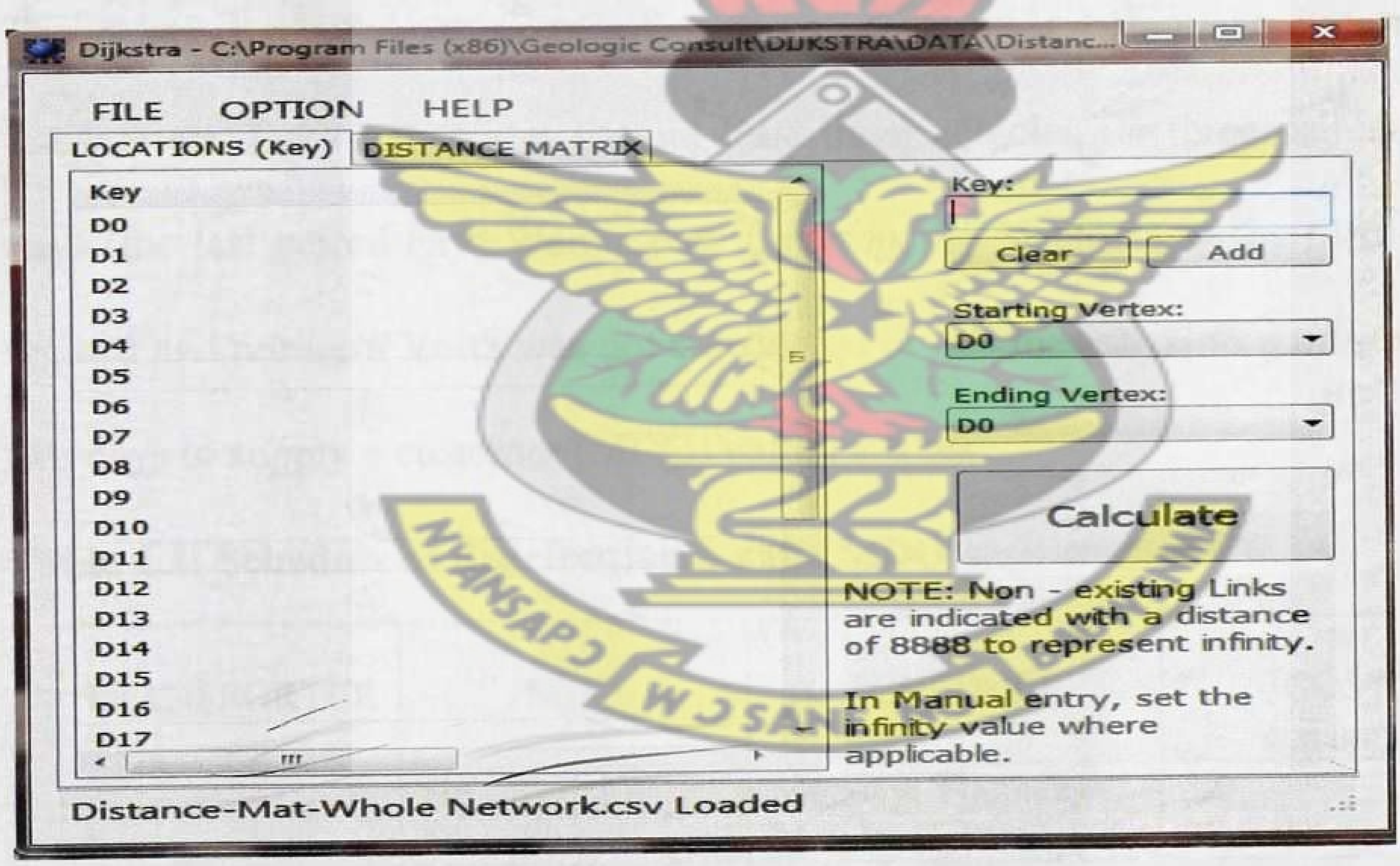


Figure 4.5: Shows result after loading the distance matrix.

The distance between locations which are represented by D0, D1, D2,... can be calculated using the starting vertex and the ending vertex. The result is display as shown in figure 4.8 below. The shortest possible route and the total distance are displayed.

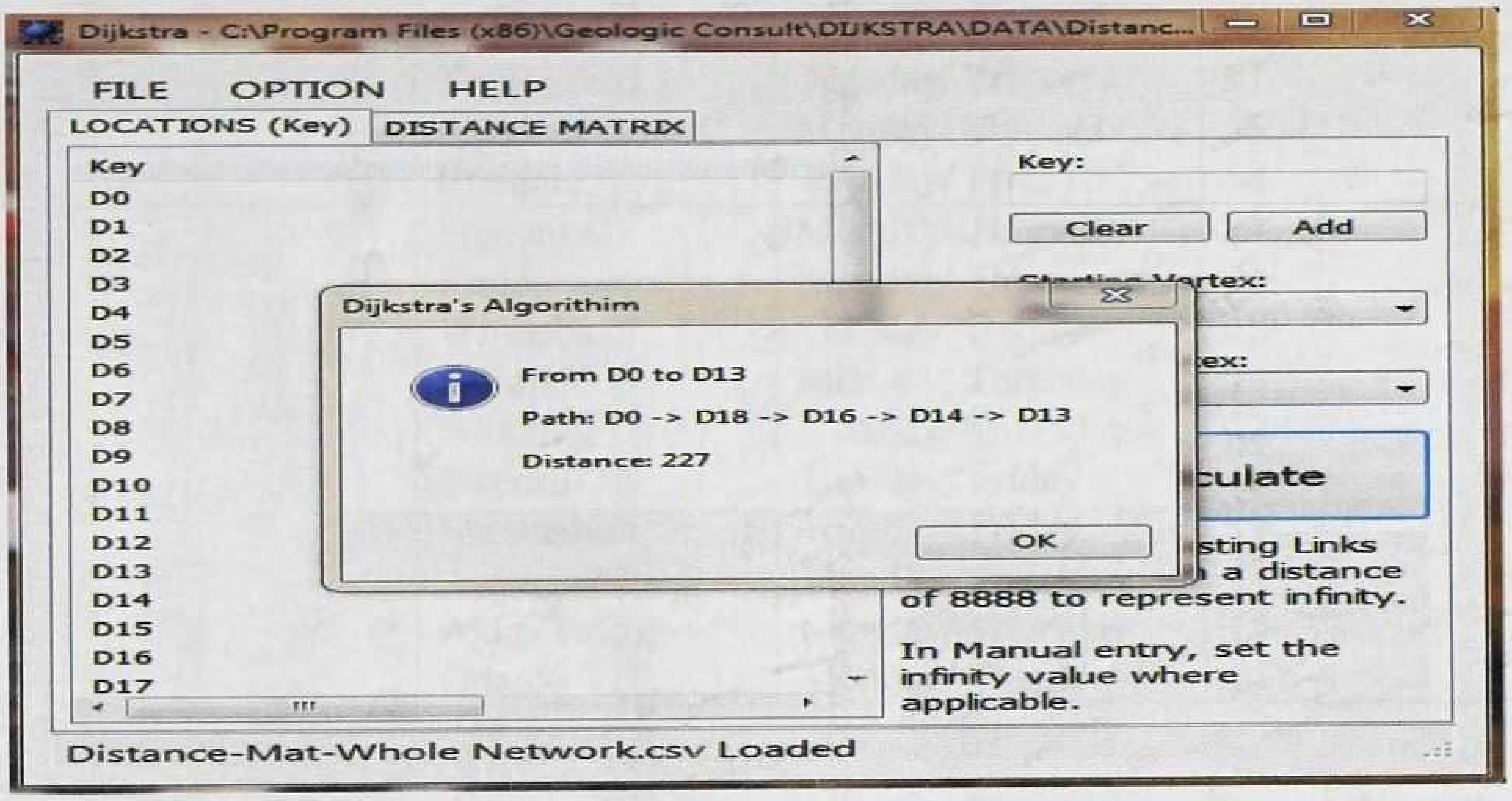


Figure 4.6: Shows the result of finding the distance between location 0 and location 13.

4.5 RESULTS AND DISCUSSION

The existing schedule of distribution is as in the table below. There were seven transporters. Each transporter uses at least three vehicles for three paired schedule days (the last paired day; Wednesday/ Saturday for locations in the three northern regions and northern Volta was not considered in this thesis). Thus a transporter had two days to supply a customer (MDO) within a week.

Table 4.3: Schedule of distribution for the MDO's.

TRANSPORTER	MDO	SCHEDULE	COST(fuel consumption)
	Ho	Monday/ Thursday	32
1	Nkawkaw	Monday/ Thursday	28
2	Enchi	Tuesday/ Friday	74
	Cape Coast	Tuesday/ Friday	25
3	Bawdie	Monday/ Thursday	58
	Takoradi	Tuesday/ Friday	40
	Bogoso	Tuesday/ Friday	54
4	Tarkwa	Tuesday/ Friday	56
	Wassa Ekuropon	Tuesday/ Friday	62
5	Koforidua	Monday/ Thursday	15
	Asankragua	Tuesday/Friday	72
	Assin Foso	Tuesday/ Friday	32

6	Samreboi TwifoPraso Aflao Essiama	Tuesday/ Friday Tuesday/ Friday Monday/ Thursday Tuesday/ Friday	72 37 35 58
7	Somanya Kade Winneba Atimpoku Suhum Swedru Mankessim Asamankese Akim Tafo Oda	Monday/ Thursday Monday/ Thursday Tuesday/ Friday Monday/ Thursday Monday/ Thursday Tuesday/ Friday Tuesday/ Friday Monday/ Thursday Monday/ Thursday Monday/ Thursday	13 20 12 15 12 15 19 14 18 25

With the new model, we have seven clusters operated by seven transporters. Each transporter was assigned to a cluster and uses at least one vehicle. A transporter had at most seven days to supply a customer (MDO).

Table 4.4: Clusters of MDO's and their shortest route.

CLUSTER	ROUTE	COST (fuel consumption)
1	D0 - D1 – D17 – D2 – D3- D0	32
2	D0 – D4 – D5 – D6 – D7 – D0	68
3	D0 – D8 – D9 – D10 – D11 – D12 – D0	121
4	D0 – D18 – D15 – D16 – D14 – D13 – D0	41
5	D0 – D19 – D22 – D21 – D20 – D0	31
6	D0 – D23 – D24 – D25 – D0	28
7	D0 – D26 – D0	35

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.0 INTRODUCTION

This chapter presents conclusion and recommendation of the study.

5.1 CONCLUSIONS

This study addresses the problem of minimizing the cost of transportation for the distribution of Coca-cola Bottling Company of Ghana's product. The study proposed Route first-Cluster second approach.

From the table 4.3 above, total fuel consumption for a trip in a week is at least 873gallons (5238litres).From table 4.4, the total fuel consumption for a trip in a week was 356 gallons (2136litres). During off-peak season, there would be one vehicle for a cluster for distribution of the products at a time, which is less than three vehicles for each group of the MDOs. The total distance covered within the areas under study was reduced and therefore the cost of transportation was reduced, such that, if a transporter uses two days to satisfy the demand of each customer in a cluster within a week, the total fuel consumption would be 712 (i.e. 356×2) which is still less than the total fuel consumption for the existing distribution model. That is about 19% reduction in fuel consumption.

It is therefore evident that the total fuel consumption is reduced due to reduced distance covered.

5.2 RECOMMENDATIONS

Based on the findings of this research work, and the reduction in the cost of transportation on the distribution of Coca-cola Bottling Company products, from the depot to the areas under study, we recommend that the company adopt this Route first-Cluster second approach model in their distribution, so as to minimize cost of distribution and at the same time satisfy the demand of their customers.

We also recommend that further study be conducted which will cover the entire key distributors in the nation.

REFERENCES

1. Ahuja, R. K., Magnanti, T. L., Orlin, J. B., (1993). Network Flows: Theory, Algorithms and Applications, Prentice Hall, Englewood Cliffs, NJ

2. Aminu A Ibrahim (2007) Graph Algorithms and shortest path problem: A case of Dijkstra's Algorithm and the Dual Carriage Ways in Sokoto Metropolis 382.

3. Amponsah, S.K. and Darkwah, F.K., —Operations Research II, IDL, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, 2010.

4. Amponsah, S.K., —Optimization Technique II, IDL, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana, 2011.

5. Baker, E. K. (1983). An exact algorithm for the time-constrained traveling salesman problem. Operation Research, 31, 938-945.

6. Baker, E. K. and Schaffer, J. R. (1998). —Solution Improvement Heuristics for the Vehicle Routing Scheduling Problem with Time Window Constraints, American Journal of Mathematics and Management Sciences, 6:261-300.

7. Beasley, J. E. (1983). Route first- Cluster second method for vehicle routing. Omega, 11,4, 403 - 408

8. Bellman, R. (1958). "On a routing problem", Quarterly of Applied Mathematics 16, page. 87-90.

9. Bellman, R. (1960). "Combinatorial Processes and Dynamic Programming". In: Combinatorial Analysis. American Mathematical Society, Providence, Rhode Island, U.S.A. pp 217-249.

10. Bellman, R. (1962). —Dynamic Programming Treatment of the TSP II. Journal of Association of Computing Machinery.9:66.

11. Bellmore, M. and Nemhauser, G.L. (1968). "The Traveling Salesman Problem: A Survey". Operations Research. 16:538-558.

12. Bodin, L. D., Golden, B. L., and Bender, A. (1983). The state of the art in the routing and scheduling of vehicles and crews. Computers and Operations Research, 10(2), 79- 116.

13. Bodin, L. D., Golden, B. L., Assad, A. A., and Ball, M. (1983). Routing and scheduling of vehicles and crews, the state of the art. Computers and Operations Research, 10(2), 63-212.

14. Christofides, N. (1985). Vehicle routing. In The traveling salesman problem: A guided tour of combinatorial optimization (pp. 431-448). Chichester, UK: John Wiley & Sons.

15. Christofides, N., Mingozzi, A. and Toth, P. (1981). An algorithm for the time constrained traveling salesman problem. Report IC-OR, Imperial College of Science and Technology, London.

16. Christofides, N., Mingozzi, A., and Toth, P. (1979). The vehicle routing problem. In Combinatorial optimization (pp. 315-338).Chichester, UK: John Wiley & Sons. 110

17. Chu, P. C., and Beasley, J.E (1998). Constraint handling in genetic algorithms: the set partitioning problem. Journal of Heuristics, 4(4), 323-357.

18. Clarke, G. & Wright, J.W.: "Scheduling of Vehicles from a Central Depot to a Number of Delivery points. Operational Research, July/ August, 12: 568-581

19. Cordeau, J-F., Gendreau, M., Laporte, G., Potvin, J-Y., and Semet, F. (2002). A guide to vehicle routing heuristics. Journal of the Operational Research Society, 53, 512-522.

20. Cormen, T.H., Leiserson, C.E., Rivest, R.R.(2009). Introduction to Algorithms.

21. Dantzig, G. B., and Ramser, J. H. (1959). The truck dispatching problem. Management Science, 6, 80-91.

22. David Joyner, Minh Van Ngugen & Nathann Cohen (2011), Algorithmic Graph, Version 0.7-r1843

23. Davis, L. (1987). Genetic algorithms and simulated annealing. Pitman, London 55: 89-103

24. De Vries (1994).The Industrial Revolution and Industrious Revolution: The journal of Economic History 54: (2) 249-270

25. Desrochers, M., Desrosiers, J. and Solomon, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operation Research, 40, 342-354.

26. Desrochers, M., Lenstra, J. K., and Savelsbergh, M. W. P. (1990). A classification scheme for vehicle routing and scheduling problem. Journal of the Operational Research Society, 46, 322-332.

27. Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In Network routing. Handbooks in Operations Research and Management Science, 8, North Holland.

28. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs Numerische Mathematik 1, pages. 269-271.

29. Dreyfus, S. E., (1969). An Appraisal of Some Shortest-Path Algorithms, Operations Research 17, 395-412

30. Dumas, Y., Desrosiers, J., Gelinas, E., and Solomon, M. (1995). An optimal algorithm for the traveling salesman problem with time windows. Operation Research, 43, 367- 371.

31. Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. Computers and Industrial Engineering, 57, 1472-1483. 111

32. Ellabib I., Otman A.B., & Calamai P. 2002. An Experimental Study of a Simple Ant Colony System for the Vehicle Routing Problem with Time Windows. ANTS, LNCS **2463**: 53-64.

33. ESRI, GIS and Mapping Software Support Group, 2006. "ArcGIS Network Analyst: Routing, Closest Facility, and Service Area Analysis". http://www.esri.com/networkanalyst (Accessed on

34. Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. Journal of Heuristics, 2, 5-30.

35. Fisher, M. L. (1994). Optimal solution of vehicle routing problems using minimum K-trees. Operation Research, 42(4), 626-642.

36. Fisher, M. L. (1995). Vehicle routing. In Network routing. Handbooks in Operations Research and Management Science, 8, North Holland.

37. Fisher, M.L. (1973). Optimal Solution of Scheduling Problems using Lagrange Multipliers: Part I. Operations Research: Vol. 21, pp 1114-1127.

38. Foster, B.A. and Ryan, D.M. (1976). —An Integer programming Approach for the Vehicle Scheduling Problem, Operations Research, 27: 367-384.

39. Freisleben, B. and Merz, P. (1996). Genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. Proceedings of the IEEE Conference on Evolutionary Computation, 616-621.

40. Fukasawa R., Longo H., Lysgaard J., Poggi D. M., Reis M., Uchoa E. & Werneck R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Math. Program. Ser. **106**: 491–511.

41. Garey, M. R., and Johnson, D. S. (1979). Computers and intractability: a guide to the theory of NP-completeness, W.H. Freeman.

42. Gillet, B.C. and Miller, L.R. (1974). A Heuristic Algorithm for the Vehicle Dispatch Problem, Operation Research, 22(2):340-349.

43. Golden, B. L., and Addad, A. (1995). The vehicle routing problem. In Network routing. Handbooks in Operations Research and Management Science, 8, North Holland.

44. Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I. M. (1998). Metaheuristics in vehicle routing. In Fleet Management and Logistics (pp. 33-56). Boston, MA: Kluwer.

45. Held, M. and Karp, R. M. (1970).The Travelling Salesman Problem and Minimum Spanning Trees: Part I. Operations Research: Vol. **18**, pp 1138-116

46. Hitchcock, F.L. (1941). The Distribution of a Product from Several Sources to Numerous Locations, J. Math Phys.20, pp 224-230

47. Homberger, J., and Gehring, H. (2005). A two–phase hybrid meta-heuristic for the vehicle routing problem with time windows. E uropean Journal of Operation Research, 162, 220-238.

48. http://en.wikipedia.org/wiki/Industrial_revolution(2011,11,25)

49 http://forums.esri.com/Thread.asp?c=93&f=1944&t=187632&mc =21#msgid565389 (Accessed on November 25, 2011).

50. Joubert, J.W. (2007). The Vehicle Routing problem: Origins and Variants, University of Pretoria edt., pp 4 Journal of Economics 110, 4: 881-908.

51. Kim, S. (1995). Expansion of Markets and the Geographic Distribution of Economic Activities: Trends in U.S Regional manufacturing structure 1860-1987, Quarterly Journal of Economics, 110, 4: 881-908.

52. Kim, S. (2001). Markets and Multiunit Firms from an American Historical Perspective Washington University, St. Louis.

53. Koopman, T.C. (1947). Optimum Utilization of Transportation System, Proc. Intern. Statis. Conf., Washington, D.C

54. Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 23, 631-640.

55. Laporte, G., and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. Annals of Discrete Mathematics, 31, 147-184.

56. Laporte, G., and Semet, F. (1999). Classical heuristics for the vehicle routing problem. Technical Report G-98-54, GERAD.

57. Laporte, G., Gendreau, M., Potvin, J-Y., and Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research, 7, 285-300.

58. Lenstra, J. K., Desrochers, M., Savelsbergh, M.W.P. and Soumis, F. (1988) —Vehicle Routing with Time Windows: Optimization and Approximation, Elsevier Science Publishers, North-Holland, 1988, (pp.65-83).

59. Lin Sh. (1965), "Computer Solutions of the Traveling Salesman Problem," Bell Systems Technical Journal, 44 pp 2245-2269.

60. Lysgaard J., Letchford A.N. &Eglese R.W. 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math. Program. Ser. **100**: 423–445.

61. Maffioli F. 2003. The vehicle routing problem: A book review. 4OR **1**: 149–153.

62. Magnanti, T. L. (1981). Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. Networks, 11, 179-214.

63. Mahmoud, T. A. (1996). A -Level Business Management (4th edition), Folie Books Ent., Tema.

64. Minty, G., (1990). A comment on the shortest route problem. Operations Research.5, 724, 1957, MIT Press, Cambridge, Massachusetts.

65. Mole, R. H. (1979). A survey of local delivery vehicle routing methodology. Journal of the Operational Research Society, 30, 245-252.

66. Nagy G. & Salhi S. (2007). Location-routing: Issues, models and methods. European Journal of Operational Research **177** : 649–672.

67. Olivera, F., (2002). "Map Analysis with Networks". http://ceprofs.tamu.edu/folivera/GISCE/ Spring2002/Presentations/NetworksIntro.ppt (Accessed on February 10, 2007).

68. Orlin, J., 2003. "Dijkstra's Algorithm Animation". MIT Open Course Ware, Network Optimization, Spring 2003.

69. Reeves, C. R. (1993). Modern heuristic techniques for combinatorial problems. Oxford, England: Blackwell.

70. Repoussis, P. P., Tarantilis, C. D., and Ioannou, G. (2007). The open vehicle routing problem with time windows. Journal of the Operational Research Society, 58, 355-367. 114

71. Rice, M., 2006. "ArcGIS Desktop - Extension – Network Analyst forum".124

72. Rochat, Y., and Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics, 1, 147-167.

73. Ropke S. & Pisinger D. 2006.A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. European Journal of Operational Research **171**: 750–775.

74. Ryan, D.M. and Foster, B.A., —A Integer programming Approach to Scheduling North-Holland Publishing Company, Harwell, UK, 1981, (p.269-280).

75. Sokoloff, K. L. (1984a). Investment in Fixed and Working Capital during Early Industrialization: Evidence from U.S. Manufacturing Firms, Journal of Economic History 44: 545-556.

76. Sokoloff, K. L. (1984b). Was the Transition from the Artisanal Shop to the Non-Mechanized Factory Associated with Gains in Efficiency? Evidence from the U.S. Manufacturing Censuses of 1820 and 1850, Explorations in Economic History 21:351- 382.

77. Sokoloff, K. L. (1988). "Inventive Activity in Early Industrial America: Evidence From Patent Records, 1790-1846," Journal of Economic History 48: 813-849

78. Solomon, M. M., and Desrosiers, J. F. (1988). Time window constrained routing and scheduling problems. Transportation Science, 22, 1-13.

79. Stewart, L.A., 2004. "The Application of Route Network Analysis to Commercial Forestry Transportation". http://gis.esri.com/library/userconf/proc05/papers/pap1309.pdf (Accessed on November 25, 2011).

80. Taillard, E.P., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.-Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows, Journal of Transportation Science, 31:170-186.

81. Toth, P., and Vigo, D. (1998). Exact algorithms for vehicle routing. In Fleet Management and Logistics (pp. 1-31). Boston: Kluwer.

82. Toth, P., and Vigo, D. (2002). The vehicle routing problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA: SIAM.

APPENDIX A

Pseodocode of Dijkstra's Algorithm

```
functionDijkstra(Graph, source):
for each vertex v in Graph:            // Initializations
    dist[v]:= infinity;// Unknown distance function from source to
v
    previous[v]:= undefined; // Previous node in optimal path from
source
end for;
dist [source]:= 0;              // Distance from source to source
Q;= the set of all nodes in Graph ;
// All nodes in the graph are unoptimized - thus are in Q
   whileQis not empty:               // The main loop
      u:= vertex in Q with smallest distance in dist[] ;
          if dist[u] = infinity:
   break;   // all remaining vertices are inaccessible from source
      end if;
      remove u from Q;
for each neighbor v of u:   // where v has not yet been removed
from Q.
            alt:=dist[u] + dist_between(u, v) ;
            if alt< dist[v]:            // Relax (u,v,a)
                dist[v]:= alt;
                previous[v]:= u;
            decrease-key v in Q;        // Reorder v in the Queue
            end if;
        end for;
    end while;
    return dist [];
endDijkstra.
```

APPENDIX B

Visual basics.net code for Dijkstra's algorithm

```
Namespace Dijkstra
    ''' <summary>
''' Represents an object that specifies the distance between two vertices.
    ''' </summary>
    ''' <remarks></remarks>
    Public Class Edge

        '//properties
        Private _graph As Graph
        ''' <summary>
''' Gets the Graph that contains this Edge.
        ''' </summary>
        Public Read Only Property Graph() As Graph
            Get
                Return Me._graph
            End Get
        End Property

        Private _first As Vertex
        ''' <summary>
''' Gets the first Vertex of this Edge.
        ''' </summary>
        Public ReadOnly Property First() As Vertex
            Get
                Return Me._first
            End Get
        End Property

        Private _second As Vertex
        ''' <summary>
''' Gets the second Vertex of this Edge.
        ''' </summary>
        Public ReadOnly Property Second() As Vertex
            Get
                Return Me._second
            End Get
        End Property

        ''' <summary>
''' Gets the zero-based index of the Edge in the Graph.
        ''' </summary>
        Public ReadOnly Property Index() As Integer
            Get
                Return Me.Graph.Edges.IndexOf(Me)
            End Get
        End Property

        Private _distance As Double
        ''' <summary>
```

```
''' Gets or sets the distance between the two verticies.
        ''' </summary>
        Public Property Distance() As Double
            Get
                Return Me._distance
            End Get
            Set(ByVal value As Double)
                If Not Object.Equals(value, Me._distance) Then
Me._distance = value
Me.Graph.NotifyRecalculate()
                End If
            End Set
        End Property

        '//constructors
        Friend Sub New(ByVal graph As Graph, ByVal first As Vertex, ByVal second As
Vertex)
Me._graph = graph
Me._first = first
Me._second = second
        End Sub

        Friend Sub New(ByVal graph As Graph, ByVal first As Vertex, ByVal second As
Vertex, ByVal distance As Double)
Me.New(graph, first, second)
Me._distance = distance
        End Sub

    End Class
End Namespace


Namespace Dijkstra

    ''' <summary>
''' Represents an object that specifies a point in a graph.
    ''' </summary>
    Public Class Vertex

        '//properties
        Private _graph As Graph
        ''' <summary>
''' Gets the Graph that contains the Vertex.
        ''' </summary>
        Public ReadOnly Property Graph() As Graph
            Get
                Return Me._graph
            End Get
        End Property

        ''' <summary>
''' Gets an array of verticies that are neighbored with the Vertex.
        ''' </summary>
```

```
        Public ReadOnly Property Neighbors() As Vertex()
            Get
                Return Me.Graph.Edges.GetNeighbors(Me)
            End Get
        End Property

        ''' <summary>
''' Gets the zero-based index of the Vertex in the Graph.
        ''' </summary>
        Public ReadOnly Property Index() As Integer
            Get
                Return Me.Graph.Verticies.IndexOf(Me)
            End Get
        End Property

        Private _key As String
        ''' <summary>
        ''' Gets or sets an identifier for the Vertex.
        ''' </summary>
        Public Property Key() As String
            Get
                Return Me._key

            End Get
            Set(ByVal value As String)
                If Not Object.Equals(value, Me._key) Then
Me._key = value
Me.Graph.NotifyRecalculate()
                End If
            End Set
        End Property

        Private _visited As Boolean
        ''' <summary>
''' Gets a value indicating whether this Vertex has been visited.
        ''' </summary>
        Public ReadOnly Property Visited() As Boolean
            Get
                Return Me._visited
            End Get
        End Property

        Private _distance As Double
        ''' <summary>
''' Gets the distance that this Vertex is from the initial Vertex.
        ''' </summary>
        Public ReadOnly Property Distance() As Double
            Get
                Return Me._distance
            End Get
```

```
        End Property

        Private _tag As Object
        ''' <summary>
        ''' Gets or sets an object that contains data about the current instance.
        ''' </summary>
        Public Property Tag() As Object
            Get
                Return Me._tag
            End Get
            Set(ByVal value As Object)
Me._tag = value
            End Set
        End Property

        Private _previousVertex As Vertex
        Friend Property PreviousVertex() As Vertex
            Get
                Return Me._previousVertex
            End Get
            Set(ByVal value As Vertex)
                Me._previousVertex = value
            End Set
        End Property

        '//constructors
        Friend Sub News(ByVal graph As Graph, ByVal key As String)
Me._graph = graph
Me._key = key
Me._distance = Double.PositiveInfinity
        End Sub

        '//methods
        Friend Sub SetDistance(ByVal distance As Double)
Me._distance = distance
        End Sub

        Friend Sub SetVisited(ByVal visited As Boolean)
Me._visited = visited
        End Sub

    End Class

End Namespace
```