Addis Ababa University
College of Natural Science
Department of Mathematics

LOCATION PROBLEMS ON WEIGHTED GRAPHS

Stream:- Optimization

By: - Abrha Tahir
Advisor: - Berhanu Guta (Ph.D)

A Project Submitted to The Department of Mathematics as Partial Fulfillment of the Requirements for the Degree of Masters of Science in Mathematics

16,October 2015
Addis Ababa, Ethiopia

Contents

Addis Ababa University
Department of Mathematics

The undersigned hereby certify that they have read and recommend to the school of graduate studies for acceptance of a project entitled *Location Problems on weighted graphs* by *Abrha Tahir* in partial fulfillment of the requirements for the degree of master of Science.

Dated: 16,October 2015

Advisor: _____
Dr.Berhanu Guta

Examining committee: 1. Dr. Semu Mitku _____

2. Dr. Yirgalem Tsegaye _____

16,October 2015

Acknowledgement

First of all I would like to thank the Lord our God for from Him through Him and to Him are all things in such a way that I believe without Him nothing is possible and also with Him nothing is impossible-with Him all things are possible.

Secondly, I would like to express my heartfelt-thank which goes to my advisor Dr. Berhanu Guta for his knowledgeable comment and suggestion enhance my effort to the next level. Moreover, during the lecture in the class he inspire me to exploit my maximum potential as much as possible. and I would like to say thanks to the Department of mathematics, Addis Ababa University for providing the necessary materials throughout the preparation of this paper.

Finally, I would like to express gratitude to my family and those who support directly or indirectly through not only ideas but also material assistance till the completion of this paper.

THANKS

Notation

G = Graph/Network.
V = $\{v_1, v_2, ..., v_n\}$ is a finite set of vertices.
E = $\{e_1, e_2, ..., e_n\}$ is a finite set of edges.
n = the number of vertices/nodes.
m = the number of edges.
I = the set of demand nodes.
i = demand nodes
J = The set of candidate facility .
j = candidate facility
h_i = the demand at node i or the weight at vertex i.
a_{ij} = edge distance(or arc length)
d_{ij} = the shortest path distance between demand node i and node j.
X_i = the location variables(facilities)
Y_{ij} = the allocation(assignment) variables
p = number of facilities to be located.
P = Path from node i to node j in the graph G.
$S(i)$ = Star distance.
$r(i)$ = radial distance.
r = coverage distance that satisfies customers' demand.
$q(r)$ = Optimal value of the set covering problem
B_L = Lower bound of the p-center objective function value.
B_U = Upper bound of the p-center objective function value.
x_{p-1} = the set of locations of $(p-1)$ facilities.
S_j^k = the sum of the entries in each column of the matrix.
APSP = All pairs shortest path.
SSSP = single-source shortest path.

Abstract

Network location problems occur when new facilities are to be located on a network. The facility location problems locate a set of facilities (resources) to minimize the cost of satisfying some set of demands (of customers) with respect to some set of constraints.

Locating a facility to the best place is a decision making problem. The best place depends on criteria like the optimal distance, the capacity of the facility, population density, and optimal cost etc. so the goal of solving location problems is to find the best location or locations to fit one or more facility which will make the highest utility value.

In this project,two sets of location problems such as the median and the center location problems are presented to locate facilities on weighted graphs(or networks).

To solve the optimal solutions that identify the median and center location of the weighted graph, we first compute the all pairs shortest path distance of the weighted graph. thus,the all pair - label correcting algorithm implemented by the Floyd-Warshals algorithm was applied to compute the distance matrix. We also examine some of the heuristic methods and binary search techniques developed to solve the problems.

Chapter 1

Preliminaries

1.1 Introduction

Network location problems occur when new facilities are to be located on a network. The network of interest may be a road network, an air transport network, a computer network, or a network of shipping lines. For a given network location problem, the new facilities are often idealized as points, and may be located anywhere on the network; constraints may be imposed upon the problem so that new facilities are not too far from existing facilities[3].
Network location problems are concerned with finding the right locations to place one or more facilities in a network of demand points[10], i.e., customers represented by nodes in the network, that optimize a certain objective function related to the distance between the facilities and the demand points. Usually, the facilities to be located are desirable, i.e., customers prefer to have the facilities located as close to them as possible. For example, services such as police and fire stations, hospitals, schools, and shopping centers are typical desirable facilities.
A network is a system of points with distances between them. A network can represent roads, pipelines, cables etc. Typical problems with networks involve finding the shortest path between one point in the network and another. In many real occasions, various attributes (various costs and profits) are usually considered in a shortest path problem. Because of the frequent occurrence of such network structured problems, there is a need to develop an efficient procedure for handling these problems[1].
The shortest path algorithm is among a small group of efficient algorithms that exist for this class of problems. In a network, any node may be connected by edges to any number of other nodes. In most representations, a link also has a cost / weight / distance / length that gives the cost for traveling across the link. Finding all pairs shortest path distances can give us the best route(path) from one point to another in order to locate facilities in the network[1].
This project encompasses solution methodologies for two well-known basic location problems; the median location problem and the center location problem. Hence, in the first unit we will see some preliminaries like basic definitions and notations. in the second unit we will see location problem models,basically the median and center location models.Finally we discuss some solution proceeders with illustrative examples.

1.2 Problem definitions

Let $G = (V, E)$ be a weighted graph (directed or undirected) which contains no negative cycle.

- Define a weight function $w_1 : E \longrightarrow \mathbb{R}^+ \cup \{0\}$, where $\mathbb{R}^+$ is the set of positive real numbers, that determine the length of a path between all pairs of vertices, which associates a length $w_1(e) = a_{ij} \geq 0$ to each arc $e = (i, j) \in E$. Where, $V = \{v_i : i = 1, 2, 3, ..., n\}$ is a set of vertices and $E = \{e_i : i = 1, 2, 3, ..., m\}$ is a set of edges in G.

- Define a weight function $w_2 : V \longrightarrow \mathbb{R}^+$, that determine the weight of the nodes in the graph G,which is the demand at each node i, That is $w_2(i) = h_i$.

- For any two vertices $i, j \in V$, let $d(i, j)$ be the length of the shortest path from i to j in G. If P is the path in G,then the weighted distance from a vertex i to P is defined as the product of the demand associated with node i,(h_i) and the distance $d(i, j)$ from a vertex i to that vertex j in P that is closet to i (i.e, weighted distance $= h_i d(i, j)$). Thus, the sum of the weighted distance from all the vertices in G to the path P is given by;
$$\sum h_i d(i, j), \ for\ each\ j \in p\ and\ i \in G.$$

- The path P which minimizes the sum is the median path of the graph G and the node j that identifies the path P with minimum sum is the median point of the graph G.

- The path P which minimizes the weighted distance $(h_i d(i, j))$ from node i to a node farthest from node i,is the central path of the graph G and the node that identifies the central path is the center of the graph G.

- Let D and D^* be the matrices whose (i, j) elements are a_{ij} and $d(i, j)$ respectively. The problem of computing D^* is known to be the all pairs shortest path (APSP) problem. Once we have computed the APSPP using Floyd-Warshals algorithm, two facility location problems are taken in to consideration . The center and median location problems.

In this paper, we need to solve the optimal locations (solutions) that identify the median and center of the graph G.

1.3 Definitions,Notations and Theorems

- A graph $G = (V, E)$ consists of none empty finite set $V = \{v_1, v_2, v_3, ..., v_n\}$ of vertices and finite set $E = \{e_1, e_2, e_3, ..., e_n\}$ of edges, where each edge links a vertex to vertex. Each arc $(i, j) \in E$ has an associated cost (or length) a_{ij} [2].

- A graph $G = (V, E)$ is said to be a weighted graph (Network) if each edge e is assigned a number, the weight of the edge e given by $w(e) = a_{ij}$[2].
Depending upon the problem being solved, sometimes weights are assigned to the edges. The weights could represent the distance between two locations, the travel time, or the

travel cost. It is important to note that the distance between vertices in a graph does not necessarily correspond to the weight of an edge.

- A graph $G = (V, E)$ is said to be a digraph (directed graph) if each of its edge is a directed edge (or arcs). Where as undirected graph is a graph where no direction is associated with the edges [2].

- Directed network is a directed graph whose nodes and/or arcs have associated numerical values (typically costs, capacities, and/or supplies and demands). In this paper, we often make no distinction between weighted graphs and networks, so we use the terms "weighted graph" and "network" synonymously[2].

- Path in a graph $G = (V, E)$ is a sequence of adjacent edges $(v_1, v_2), (v_2, v_3), .., (v_{t-1}, v_t)$ where no edge and no intermediate vertex is repeated. v_1 is initial and v_t is terminal vertex of the path. All other vertices are intermediate vertices [2].

- Cycle is a closed path,$(v_1, v_2), (v_2, v_3), .., (v_{t-1}, v_t), (v_t, v_1)$, where the initial and terminal vertices of the path are the same [2].

- A negative cycle in a graph G is a cycle $v_0 - v_1 - ... - v_t - v_0$ in which $w(v_0, v_1) + w(v_1, v_2) + ... + w(v_t, v_0) < 0$.
  Which mean a cycle whose sum of the weight of edges is a negative value.

- Connectivity: Two nodes i and j are connected if the graph contains at least one path from node i to node j. A graph is connected if there is a path between every pair of its vertices. Otherwise, the graph is disconnected [2].

- Strong Connectivity: A connected graph is strongly connected if it contains at least one directed path from every node to every other node [2].

- Trees is a connected graph that contains no cycle. A tree is a very important graph theoretic concept that arises in a variety of network flow algorithms [2].

- Distance label: an estimate (in particular, an upper bound) on the shortest path distance from the source node to each network node [2].

- A demand node is considered to be covered if the length of the shortest path (d_{ij}) is less than or equal to the coverage distance r $(i, e.d_{ij} \leq r)$. Where coverage distance (r) is a pre specified distance.

- A shortest path is the path between specified nodes having minimum weighted total cost (or length),and is described as the minimum path cost.

$$\min \sum h_i d_{ij}, where \;\; i, j = 1, 2, \ldots, n$$

- **Definition**
  Let G be an n-vertex weighted graph. The point $y_0 \in G$ is defined as the absolute

median of G [4], if the sum of the weighted shortest distances between y_0 and every other point y on G satisfies the inequality:

$$\sum_{i=1}^{n} h_i d(v_i, y_0) \leq \sum_{i=1}^{n} h_i d(v_i, y) \tag{1.1}$$

- **Theorem 1. (Hakimi 1964)**
An absolute median of a graph is always at a vertex of the graph.
Proof
let x_0 be an arbitrary point on the graph G, not on a vertex of G, there always exist a vertex v_m of G such that

$$\sum_{i=1}^{n} h_i d(v_i, v_m) \leq \sum_{i=1}^{n} h_i d(v_i, x_0), \qquad (by \ \ definition \ \ 1.1).$$

Note that this result does not exclude other absolute medians existing on the network. Let the point x_0 be located on an edge (v_a, v_b). Nodes are re-indexed in such a way that the point x_0 is located on the edge (v_p, v_{p+1}). For all nodes with indices smaller than or equal to p, the shortest path connecting the node and the point x_0 goes through node v_p, (i.e. connected through the left of x_0) while for all nodes with indices larger than p, the shortest path between the node and the point x_0 goes through node v_{p+1} (i.e. through the right of x_0).

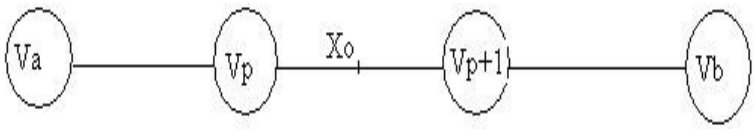

Figure 1. Simple path.

The total weighted distance can then be expressed as the sum of two terms, representing the sum of the weighted distances to the left-side nodes, and to the right-side nodes, respectively:

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = \sum_{i=1}^{p} h_i d(v_i, x_0) + \sum_{i=p+1}^{n} h_i d(v_i, x_0)$$

In turn, each distance can be decomposed in to two as follows:

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = (\sum_{i=1}^{p} h_i d(v_i, v_p) + \sum_{i=1}^{p} h_i d(v_p, x_0)) + (\sum_{i=p+1}^{n} h_i d(v_i, v_{p+1}) + \sum_{i=p+1}^{n} h_i d(v_{p+1}, x_0))$$

$$since, \sum_{i=p+1}^{n} h_i d(v_{p+1}, x_0) = \sum_{i=p+1}^{n} h_i d(v_{p+1}, v_p) - \sum_{i=p+1}^{n} h_i d(v_p, x_0)$$

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = (\sum_{i=1}^{p} h_i d(v_i, v_p) + \sum_{i=1}^{p} h_i d(v_p, x_0)) +$$
$$(\sum_{i=p+1}^{n} h_i d(v_i, v_{p+1}) + \sum_{i=p+1}^{n} h_i d(v_{p+1}, v_p) - \sum_{i=p+1}^{n} h_i d(v_p, x_0))$$

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = (\sum_{i=1}^{p} h_i d(v_i, v_p) + \sum_{i=p+1}^{n} h_i d(v_i, v_{p+1}) + \sum_{i=p+1}^{n} h_i d(v_{p+1}, v_p)) +$$
$$[\sum_{i=1}^{p} h_i - \sum_{i=p+1}^{n} h_i] d(v_p, x_0)$$

The term in square brackets is, the sum of node weights on the left, minus the sum of the node weights on the right of the point x_0. Without loss of generality, suppose that the sum of node weights on the left is larger than or equal to the sum on the right. Then, the term in square brackets is non-negative, and by reducing the distance $d(v_p, x_0)$ that multiplies the square bracketed term, i.e. moving the point x_0 to the left, the total sum is reduced or, at most, stays the same. The minimum value for this distance is zero, which happens when the median point x_0 is located on the node v_p. The same argument can be repeated when the sum of the node weights on the right of x_0 is strictly larger than the sum of node weights on the left. In that case, the term in square brackets is strictly negative, and moving the point x_0 to the right strictly reduces the value of the total sum. The best value is obtained when x_0 is located on top of v_{p+1}. This proves that there is always a median point on a vertex of the graph, either on the left or the right of a point x_0 on an edge, i.e., for any point x_0,

$$\sum_{i=1}^{n} h_i d(v_i, v_m) \leq \sum_{i=1}^{n} h_i d(v_i, x_0)$$

Under this generalization, the absolute median is the 1-median of the graph. the term p-median refers to the set of vertices X_p. the vertices in X_p are called p-median vertices.

- **Definition**
  Let G be a graph. If X_p is a set of p points $\{x_1, x_2, \ldots, x_p\}$ in G, and the distance of a node v_i to X_p is,

  $$d(v_i, X_p) = \min\{d(v_i, x_1), d(v_i, x_2), \ldots, d(v_i, x_p)\}$$

  i.e. the distance between the node v_i and its closest point $x_k \in X_p$, then the set X_p^* is a p-median of the graph G, if for every X_p on G, X_p^* satisfies:

  $$\sum_{i=1}^{n} h_i d(v_i, X_p^*) \leq \sum_{i=1}^{n} h_i d(v_i, X_p) \tag{1.2}$$

  In other words,X_p^* is the set of p points on the graph such that, if these points were facilities of some sort, the total weighted distance between the demands and their closest facility would be minimized[5].

- **Theorem 2. (Hakimi 1965)**
  There exists a subset V_p^* of the set of vertices, containing p vertices such that for every set of p points of X on G

$$\sum_{i=1}^{n} h_i d(v_i, V_p^*) \leq \sum_{i=1}^{n} h_i d(v_i, X) \tag{1.3}$$

  Proof
  Let p be clusters of demand points, each cluster j consisting of a point x_j in X and a set of demands for which x_j is the closest point in X. Then, if the point x_j is on an edge, by the theorem (Hakimi 1964), there is always a vertex median v_j^* such that

$$\sum_{i \in clusterj}^{n} h_i d(v_i, v_j^*) \leq \sum_{i \in clusterj}^{n} h_i d(v_i, x_j)$$

  The same inequality can be derived for each cluster. Note that the allocation of demands has not changed; the demands that were in cluster j are still in the same cluster. Adding up all these Inequalities.

$$\sum_{j} \sum_{i \in clusterj}^{n} h_i d(v_i, v_j^*) \leq \sum_{i=1}^{n} h_i d(v_i, X)$$

  The left hand side of this inequality consists of the sum of p terms, one for each one of the original clusters. However, as the median point in each cluster moves toward a vertex, it might happen that a demand node becomes reassigned to a different node in V_j^* . This only happens if the re-allocation contributes to decrease still more the total sum, so

$$\sum_{i=1}^{n} h_i d(v_i, V_j^*) \leq \sum_{j} \sum_{i \in clusterj}^{n} h_i d(v_i, v_j^*) \quad \Rightarrow \quad \sum_{i=1}^{n} h_i d(v_i, V_j^*) \leq \sum_{i=1}^{n} h_i d(v_i, X) \quad \blacksquare$$

1.4 Shortest path problems

In graph theory, the shortest path problem is the problem of finding a path between two vertices(or nodes) in a weighted graph, such that the sum of the weights of its constituent edges is minimized[1].
Consider a road map; in this case, the vertices represent locations and the edges represent segments of road and are weighted by the time needed to travel that segment(or path). Researchers have studied several different types of (directed) shortest path problems. The most common problems are:

- The single-source shortest path problem(SSSPP),which is used to find the shortest paths from one node to all other nodes for networks with non-negative arc lengths using label-setting (or Dijkstra's) algorithm. It designate one label as permanent (optimal) at each iteration, after it assign tentative distance labels to nodes at each step. The distance labels are estimates of (i.e., upper bounds on) the shortest path distances [2].

- The all-pairs shortest path problem(APSPP) determines shortest path distances between every pairs of nodes in a network. it assign tentative distance labels to nodes at each step and considers all labels as temporary until the final step, when they all become permanent. The distance labels are estimates of (i.e., upper bounds on) the shortest path distances [2].

1.4.1 Optimality conditions

In this section we develop necessary and sufficient conditions for a set of distance labels to represent shortest path distances.
Let $d(j)$ for $j \neq s$, denote the length of a shortest path from the source node (s) to the node j [we set $d(s) = 0$].

- If the distance labels are shortest path distances, they must satisfy the following necessary optimality conditions:

$$d(j) \leq d(i) + a_{ij}, for \;\; all(i,j) \in E. \tag{1.4}$$

  These inequalities state that for every arc (i, j) in the network, the length of the shortest path to node j is not greater than the length of the shortest path to node i plus the length of the arc (i, j). For, if not, some arc $(i, j) \in E$ must satisfy the condition $d(j) > d(i) + a_{ij}$; in this case, we could improve the length of the shortest path to node j by passing through node i, thereby contradicting the optimality of distance labels $d(j)$.

- If the distance labels $d(j)$ satisfies the necessary optimality conditions above , they also satisfies the sufficient condition for optimality.
  Let $s = i_l - i_2 - ... - i_k = j$ be any directed path P from the source node s to node j.
  The necessary optimality conditions imply that;

$$\begin{gathered} d(j) = d(i_1, i_k) \leq a_{i_1 i_2} + d(i_2, i_k) \\ d(i_2, i_k) \leq a_{i_2 i_3} + d(i_3, i_k) \\ d(i_3, i_k) \leq a_{i_3 i_4} + d(i_4, i_k) \\ \vdots \\ d[i_{k-1}, i_k] \leq a_{i_{k-1} i_k} + d(i_k, i_k). \\ \text{these inequalities, in turn, imply that} \\ d(j) \leq a_{i_1 i_2} + a_{i_2 i_3} + ... + a_{i_{k-l} i_k} = \sum a_{ij}, \text{ where } (i,j) \in P. \end{gathered}$$

Thus $d(j)$ is a lower bound on the length of any directed path from the source to node j. Since $d(j)$ is the length of some directed path from the source to node j, it also is an upper bound on the shortest path distance. Therefore, $d(j)$ is the shortest path length, and we have established the following result.

Theorem 3. Shortest Path Optimality Conditions
For every node $j \in G$, let $d(j)$ denote the length of some directed path from the source node to node j. Then the numbers $d(j)$ represent shortest path distances if and only if they satisfy the following shortest path optimality conditions [2]:

$$d(j) \leq d(i) + a_{ij}, \quad for\ all\ (i,j) \in E. \tag{1.5}$$

1.4.2 Label-correcting algorithm

The Label-correcting algorithm maintain a set of distance labels $d(j)$ for every node $j \in G$ at every stage. At intermediate stages of computation, the distance label $d(j)$ is an estimate of (an upper bound on) the shortest path distance from the source node s to node j, and at termination it is the shortest path distance [2].
The label $d(j)$ is either ∞, indicating that we have yet to discover a directed path from the source to node j, or it is the length of some directed path from the source to node j [2].

$$d(j) = \begin{cases} 0, & if\ i = j. \\ \infty, & if\ (i,j) \ni E\ for\ i \neq j. \\ d(i,j), & if\ (i,j) \in E\ for\ i \neq j. \end{cases}$$

For each node j we also maintain a predecessor index, $pred(j)$, which records the node prior to node j in the current directed path of length $d(j)$.

$$pred(j) = \begin{cases} Nill, & if\ (i,j) \ni E\ and\ i = j. \\ i, & if\ (i,j) \in E\ for\ i \neq j. \end{cases}$$

At termination, the predecessor indices allow us to trace the shortest path from the source node back to node j. The label-correcting algorithm is a general procedure for successively updating the distance labels until they satisfy the shortest path optimality conditions.
Formal description of the generic label-correcting algorithms [2].
Algorithm:- label-correcting;
begin
$d(s) := 0$ and $pred(s) := Nill$;
$d(j) := \infty$ for each $j \in G - \{s\}$;
while some arc (i,j) satisfies $d(j) > d(i) + a_{ij}$ do
begin
$d(j) := d(i) + a_{ij}$, where $(i,j) \in E$.
$pred(j) := i$;
end;
end;
This algorithm Selects arcs violating their optimality conditions and updates distance labels.

1.5 All-pairs shortest path problem

The all-pairs shortest path problem requires that, we determine shortest path distances between every pairs of nodes in a network. Hence we suggest two approaches for solving the

all-pairs shortest path problem [2].

1. **The repeated shortest path algorithm**, which is well suited for sparse networks. It preprocesses the network so that all (reduced) arc lengths are nonnegative. Then applies Dijkstra's algorithm n times with each node $i \in G$ as the source node. that mean,
   If the network has non-negative arc lengths, we can solve the all-pairs shortest path problem by applying any single-source shortest path algorithm (SSSPA)(or Dijkstra's algorithm) n times, considering each node as the source node once [2].
   If the network contains some negative arcs, we first transform the network to one with non-negative arc lengths.Then we compute the shortest path distances in the original network from the shortest path distances in the transformed network using SSSPA.

2. **The all-pairs label-correcting algorithm**, which is the generalization of the label-correcting algorithm,and is especially well suited for dense networks. It determines shortest path distances between every pairs of nodes in a network only if the network contains no negative cycle by assigning tentative distance labels to nodes at each step and considers all labels as temporary until they all become permanent[2].

The generic all-pairs label-correcting algorithm develop a special implementation of this generic algorithm, known as the Floyd-Warshall algorithm, that runs in $O(n^3)$ time [2]. The all-pairs label-correcting algorithm relies on all-pairs shortest path optimality conditions.

1.5.1 All-Pairs Shortest Path Optimality Conditions

The all-pairs label-correcting algorithm maintains a distance label $d(i, j)$ for every pair of nodes in the network; this distance label represents the length of some directed path from node i to node j and hence will be an upper bound on the shortest path length from node i to node j.
The algorithm updates the matrix of distance labels until they represent shortest path distances [2]. It uses the following generalization.
Theorem 4. All-Pairs Shortest Path Optimality Conditions.
For every pair of nodes $(i, j) \in G \times G$, let $d(i, j)$ represent the length of some directed path from node i to node j. These distances represent the all pairs shortest path distances if and only if they satisfy[2]:

$$d(i, j) \leq d(i, k) + d(k, j), \quad for \ \ all \ \ nodes \ i, j, and \ k \in G. \tag{1.6}$$

Proof.
We use a contradiction argument to establish that the shortest path distances $d(i, j)$ must satisfy the conditions. Suppose that $d(i, k) + d(k, j) < d(i, j)$ for nodes i, j, and k. The union of the shortest paths from node i to node k and node k to node j is a directed path of length $d(i, k) + d(k, j)$ from node i to node j. This directed path decomposes into a directed path, say P, from node i to node j and some directed cycles. Since each directed cycle in the network has non-negative length, the length of the path P is at most,$d(i, k)+d(k, j) < d(i, j)$, which is contradicting the optimality of $d(i, j)$.

We now show that if the distance labels $d(i,j)$ satisfy the conditions and they represent shortest path distances.
Let P be a directed path of length $d(i,j)$ consisting of the sequence of nodes $i = i_1 - i_2 - i_3 - ... - i_k = j$. By optimality condition we have:

$$\begin{aligned} d(i,j) = d(i_1,i_k) &\leq a_{i_1 i_2} + d(i_2,i_k) \\ d(i_2,i_k) &\leq a_{i_2 i_3} + d(i_3,i_k) \\ d(i_3,i_k) &\leq a_{i_3 i_4} + d(i_4,i_k) \\ &\vdots \\ d[i_{k-1},i_k] &\leq a_{i_{k-1} i_k} + d(i_k,i_k). \end{aligned}$$

These inequalities, in turn, imply that

$$d(i,j) \leq a_{i_1 i_2} + a_{i_2 i_3} + ... + a_{i_{k-l} i_k} = \sum a_{ij}, \text{ for } (i,j) \in P$$

Which contradicts our assumption that $d(i,k) + d(k,j) < d(i,j)$.
Therefore, $d(i,j)$ is a lower bound on the length of any directed path from node i to node j. By assumption, $d(i,j)$ is also an upper bound on the shortest path length from node i to node j. Consequently, $d(i,j)$ must be the shortest path length between these nodes which is the derived conclusion of the theorem.
The all-pairs shortest path optimality conditions immediately yield the following generic all-pairs label-correcting algorithm: Start with some distance labels $d(i,j)$ and successively update these until they satisfy the optimality conditions [2]. If the operation checking whether $d(i,j) > d(i,k) + d(k,j)$, then the algorithm set $d(i,j) = d(i,k) + d(k,j)$ as a triple operation.
Algorithm:- all-pairs label-correcting;
begin
set $d(i,j) := \infty$ for all $(i,j) \in G \times G$;
set $d(i,i) := 0$ for all $i \in G$;
for each $(i,j) \in E$, do $d(i,j) := a_{ij}$;
while the network contains three nodes i, j, and k
satisfying $d(i,j) > d(i,k) + d(k,j)$ do $d(i,j) := d(i,k) + d(k,j)$;
end;
To establish the finiteness and correctness of the generic all-pairs label correcting algorithm, we assume that the data are integral and that the network contains no negative cycle.

1.5.2 Floyd-Warshall's Algorithm

Floyd-Warshall algorithm is a procedure, which is used to find the shortest paths among all pairs of nodes in a graph, which does not contain any cycles of negative length (for then the shortest path is undefined. But,the graph may have negative weight edges. The main advantage of this algorithm is its simplicity.
The Floyd-Warshall algorithm is an efficient matrix method algorithm to find all- pair shortest paths on a graph. That is, it is guaranteed to find the shortest path between every pair of vertices in a graph. This algorithm can also be used to detect the presence of negative cycles; the graph has a negative cycles if at the end of the algorithm, the distance from a

vertex i to itself is negative (*i.e.* $a_{ii} < 0$).

The Floyd-Warshall algorithm obtains a matrix of shortest path distances within $O(n^3)$ computations. The algorithm achieves this bound by applying the triple operations cleverly. The algorithm is based on inductive arguments developed by an application of a dynamic programming technique[2].
Let $d^n(i,j)$ be the length of the shortest path from node i to node j that uses only the vertices $1, 2, 3, ..., n-1$, as intermediate vertices (nodes) in $G \times G$.
Clearly, $d^{k+1}(i,j)$ represents the actual shortest path distance from node i to node j.
The Floyd-Warshall algorithm first computes:
$\Rightarrow$ $d^1(i,j)$ for all node pairs i and j.
$\Rightarrow$ Using $d^1(i,j)$, it then computes $d^2(i,j)$ for all node pairs i and j.
$\Rightarrow$ Using $d^2(i,j)$, it then computes $d^3(i,j)$ for all node pairs i and j.
$\Rightarrow$ It repeats this process until it obtains $d^{k+1}(i,j)$ for all node pairs i and j, when it terminates.
$\Rightarrow$ Given $d^k(i,j)$, the algorithm computes $d^{k+1}(i,j)$ using the following property.

Property:- $d^{k+1}(i,j) = min\{d^k(i,k) + d^k(k,j), \quad d^k(i,j)\}$
This property is valid for the following reason. A shortest path that uses only the nodes $1, 2, ..., k$ as intermediate nodes either:

- does not pass through node k, in which case $d^{k+1}(i,j) = d^k(i,j)$, or
- does pass through node k, in which case $d^{k+l}(i,j) = d^k(i,k) + d^k(k,j)$.
  Therefore, $d^{k+l}(i,j) = min\{d^k(i,j), d^k(i,k) + d^k(k,j)\}$.

After n iterations, there is no need any to go through any more intermediate vertices, so the distance $d^n(i,j)$ represents the shortest distance between i and j.

Algorithm:- Floyd-Warshall;
begin (Initialisation:)
Set $d(i,j) := \infty$ and $pred(i,j := Nill$; for all node pairs $(i,j) \in G \times G$.
Set $d(i,i) := 0$; for all nodes $i \in G$.
Set $d(i,j) := a_{ij}$ and $pred(i,j) := i$; for each arc $(i,j) \in E$.
for each $k := 1 - to - n$ do
for each $(i,j) \in G \times G$ do
if $d(i,j) > d(i,k) + d(k,j)$ then,
begin
$d(i,j) := d(i,k) + d(k,j)$;
$pred(i,j) := pred(k,j)$;
end;
end;
This algorithm updates or Corrects distance labels in a systematic way until they represent

the shortest path distances and gives the following Output.

$$\textbf{Distance Matrix} \quad D = \begin{pmatrix} d_{11} \dots d_{1n} \\ \vdots \ddots \vdots \\ d_{n1} \dots d_{nn} \end{pmatrix}, \quad \textbf{Predecessor matrix} \quad P = \begin{pmatrix} p_{11} \dots p_{1n} \\ \vdots \ddots \vdots \\ p_{n1} \dots p_{nn} \end{pmatrix}$$

where d_{ij} is the length of a shortest path from i to j and (p_{ij}, j) is the final edge of a shortest path from i to j that is p_{ij} is the predecessor of the node j on a shortest i - j - path (if such a path exists)[2].

The Floyd-Warshall algorithm uses predecessor indices, $pred(i, j)$, for each node pair (i, j). the index $pred(i, j)$ denotes the last node prior to node j in the tentative shortest path from node i to node j. The algorithm maintains the invariant property that when $d(i, j)$ is finite, the network contains a path from node i to node j of length $d(i, j)$. Using the Predecessor indices, we can obtain this path, say P, from node i to node j as follows. We backtrack along the path P starting at node j. Let $k = pred(i, j)$. Then k is the node prior to node j in P. Similarly, $l = pred(i, k)$ is the node prior to node k in P,$h = pred(i, l)$ is the node prior to node l in P and so on. We repeat this process until we reach node i. Thus,by backtracking the path p is $i - A - .... - h - l - k - j$.

The Floyd-Warshall algorithm clearly performs n major iterations, one for each i, and within each major iteration, it performs $O(1)$ computations for each node pair. Consequently, it runs in $O(n^3)$ time. We thus have established the following result.

Theorem5.

The Floyd-Warshall algorithm computes shortest path distances between all pairs of nodes in $O(n^3)$ time [2].

Proof:-

One may observe that all pair shortest paths can be computed applying Dijkstras algorithm n times every time from a different source node. if n is the number of vertices in a graph or network, the complexity of Dijkstras algorithm runs in $O(n^2)$ times.

In turn,the Floyd-Warshall algorithm runs Dijkstras algorithm (or $O(n^2)$), n times. So, we can compute the running time for Floyd-Warshall algorithm as: $O(n \times n^2) = O(n^3)$.

Hence,in this way one would get an overall complexity of Floyd-Warshall's algorithm $O(n^3)$ times.

Example.1:- Consider the directed graph(or Network)below.

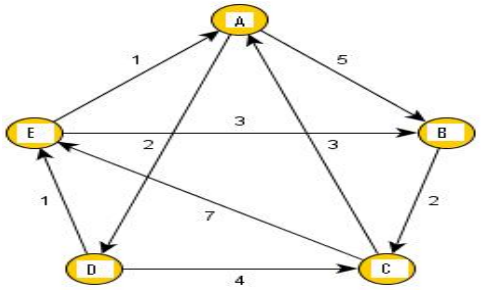

Figure 2: Simple Graph(or network)

Computing all pairs shortest path distances using the Floyd-Warshall algorithm, one gets the following matrices in which at each step the entries corresponding to a new path are

replaced by the length of the shortest path. (i.e., whenever $d_{ij}^{(k+1)} = d_{i,k+1}^{(k)} + d_{k+1,j}^{(k)}$).
For $k = 0$. Implies no intermediate node is used.

$$D^0(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & \infty & 2 & \infty \\ B & \infty & 0 & 2 & \infty & \infty \\ C & 3 & \infty & 0 & \infty & 7 \\ D & \infty & \infty & 4 & 0 & 1 \\ E & 1 & 3 & \infty & \infty & 0 \end{array}\right), \qquad P^0(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & Nill & A & Nill \\ B & Nill & Nill & B & Nill & Nill \\ C & C & Nill & Nill & Nill & C \\ D & Nill & Nill & D & Nill & D \\ E & E & E & Nill & Nill & Nill \end{array}\right)$$

$d(A,A) = 0 \;\; d(A,B) = 5 \;\; d(A,C) = \infty.$ $\qquad P(A,A) = Nill, \;\; P(A,B) = A, \;\; P(A,C) = Nill$

For $k = A$. Implies node A is used as intermediate node.

$$D^A(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & \infty & 2 & \infty \\ B & \infty & 0 & 2 & \infty & \infty \\ C & 3 & 8 & 0 & 5 & 7 \\ D & \infty & \infty & 4 & 0 & 1 \\ E & 1 & 3 & \infty & 3 & 0 \end{array}\right), \qquad P^A(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & Nill & A & Nill \\ B & Nill & Nill & B & Nill & Nill \\ C & C & A & Nill & A & C \\ D & Nill & Nill & D & Nill & D \\ E & E & E & Nill & A & Nill \end{array}\right)$$

$d(C,B) = min\{d^0(C,B), d^0(C,A) + d^0(A,B)\}.$ $\qquad P(C,B) = P(A,B) = A$
$d(C,B) = min\{\infty, 3+5\} = 8$

For $k = \{A, B\}$. Implies node A and B are used as intermediate nodes.

$$D^B(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & 7 & 2 & \infty \\ B & \infty & 0 & 2 & \infty & \infty \\ C & 3 & 8 & 0 & 5 & 7 \\ D & \infty & \infty & 4 & 0 & 1 \\ E & 1 & 3 & 5 & 3 & 0 \end{array}\right), \qquad P^B(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & B & A & Nill \\ B & Nill & Nill & B & Nill & A \\ C & C & A & Nill & A & C \\ D & Nill & Nill & D & Nill & D \\ E & E & E & B & A & Nill \end{array}\right)$$

$d(A,C) = min\{d^A(A,C), d^A(A,B) + d^A(B,C)\}.$ $\qquad P(A,C) = P(B,C) = B$
$d(C,B) = min\{\infty, 5+2\} = 7$

For $k = \{A, B, C\}$. Implies node A, B and C are used as intermediate nodes.

$$D^C(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & 7 & 2 & 14 \\ B & 5 & 0 & 2 & 7 & 9 \\ C & 3 & 8 & 0 & 5 & 7 \\ D & 7 & 12 & 4 & 0 & 1 \\ E & 1 & 3 & 5 & 3 & 0 \end{array}\right), \qquad P^C(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & B & A & C \\ B & C & Nill & B & A & C \\ C & C & A & Nill & A & C \\ D & C & A & D & Nill & D \\ E & E & E & B & A & Nill \end{array}\right)$$

$d(A,E) = min\{d^B(A,E), d^B(A,B) + d^B(B,C) + d^B(C,E)\}.$ $\;\; P(A,E) = P(C,E) = C$
$d(C,B) = min\{\infty, 5+2+7\} = 14$

For $k = \{A, B, C, D\}$. Implies node A, B C and D are used as intermediate nodes.

$$D^D(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & 6 & 2 & 3 \\ B & 5 & 0 & 2 & 7 & 8 \\ C & 3 & 8 & 0 & 5 & 6 \\ D & 7 & 12 & 4 & 0 & 1 \\ E & 1 & 3 & 5 & 3 & 0 \end{array}\right), \qquad P^D(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & D & A & D \\ B & C & Nill & B & A & D \\ C & C & A & Nill & A & D \\ D & C & A & D & Nill & D \\ E & E & E & B & A & Nill \end{array}\right)$$

$d(B, E) = min\{d^C(B, E), d^C(B, C) + d^C(C, A) + d^C(A, D) + d^C(D, E)\}, \qquad P(A, E) = C$
$d(C, B) = min\{9, 2 + 3 + 2 + 1\} = 8$

For $k = \{A, B, C, D, E\}$. Implies node A, B C,D and E are used as intermediate nodes.

$$D^E(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & 6 & 2 & 3 \\ B & 5 & 0 & 2 & 7 & 8 \\ C & 3 & 8 & 0 & 5 & 6 \\ D & 2 & 4 & 4 & 0 & 1 \\ E & 1 & 3 & 5 & 3 & 0 \end{array}\right), \qquad P^E(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & D & A & D \\ B & C & Nill & B & A & D \\ C & C & A & Nill & A & D \\ D & E & E & D & Nill & D \\ E & E & E & B & A & Nill \end{array}\right)$$

$d(D, B) = min\{d^D(D, B), d^D(D, E) + d^D(E, B)\}. \qquad P(D, B) = P(E, B) = E$
$d(C, B) = min\{12, 1 + 3\} = 4$
The final matrices contain the length of all pairs shortest paths and all the information allowing their reconstruction.

Example.2:- Consider the undirected graph(or Network)below.

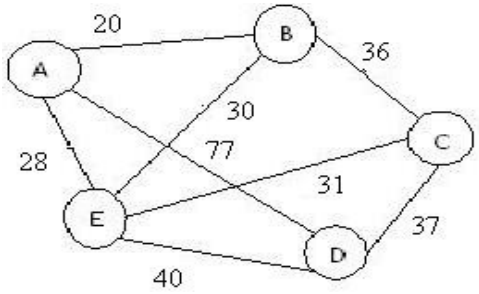

Figure 3: Simple Graph(or network)

Computing all pairs shortest path distances using the Floyd-Warshall algorithm, one gets the following matrices, in which at each step the entries corresponding to a new path are replaced by the length of the shortest path. (i.e., whenever $d_{ij}^{(k+1)} = d_{i,k+1}^{(k)} + d_{k+1,j}^{(k)}$).
For $k = 0$. Implies no intermediate node is used.

$$D^0(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & \infty & 77 & 28 \\ B & 20 & 0 & 36 & \infty & 30 \\ C & \infty & 36 & 0 & 37 & 31 \\ D & 77 & \infty & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^0(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & Nill & A & A \\ B & B & Nill & B & Nill & B \\ C & Nill & C & Nill & C & C \\ D & D & Nill & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

$d(A,A)=0 \quad d(A,C)=\infty.$ $\qquad P(A,A)=Nill, \quad P(A,B)=A, \quad P(A,C)=Nill$
$d(A,B)=20 \quad d(A,D)=77$

For $k=A$. Implies, A is used as intermediate node.

$$D^A(d_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & \infty & 77 & 28 \\ B & 20 & 0 & 36 & 97 & 30 \\ C & \infty & 36 & 0 & 37 & 31 \\ D & 77 & 97 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^A(p_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & Nill & A & A \\ B & B & Nill & B & \mathbf{A} & B \\ C & Nill & C & Nill & C & C \\ D & D & \mathbf{A} & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

$d^A(B,D)=min\{d^0(B,D), d^0(B,A)+d^0(A,D)\}$ $\qquad P(B,D)=P(B,A)+P(A,D)$
$\quad =min\{\infty, 20+77\}=97=d(D,B).$ $\qquad\qquad =P(A,D)=A$

For $k=\{A,B\}$. Implies, A and B are used as intermediate nodes.

$$D^B(d_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & 56 & 77 & 28 \\ B & 20 & 0 & 36 & 97 & 30 \\ C & 56 & 36 & 0 & 37 & 31 \\ D & 77 & 97 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^B(p_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & \mathbf{B} & A & A \\ B & B & Nill & B & A & B \\ C & \mathbf{B} & C & Nill & C & C \\ D & D & A & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

$d^B(A,C)=min\{d^A(A,C), d^A(A,B)+d^A(B,C)\}.$ $\qquad P(A,C)=P(A,B)+P(B,C).$
$\quad =min\{\infty, 20+36\}=56=d(C,A).$ $\qquad\qquad =P(B,C)=B$

For $k=\{A,B,C\}$. Implies, A,B and C are used as intermediate nodes.

$$D^C(d_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & 56 & 77 & 28 \\ B & 20 & 0 & 36 & 73 & 30 \\ C & 56 & 36 & 0 & 37 & 31 \\ D & 77 & 73 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^C(p_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & B & A & A \\ B & B & Nill & B & \mathbf{C} & B \\ C & B & C & Nill & C & C \\ D & D & \mathbf{C} & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

$d^C(B,D)=min\{d^B(B,D), d^B(B,C)+d^B(C,D)\}.$ $\qquad P(B,D)=P(B,C)+P(C,D).$
$\quad =min\{97, 36+37\}=73=d(D,B).$ $\qquad\qquad =P(C,D)=C$

For $k=\{A,B,C,D\}$. Implies, A,B,C and D are used as intermediate nodes.

$$D^D(d_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & 56 & 77 & 28 \\ B & 20 & 0 & 36 & 73 & 30 \\ C & 56 & 36 & 0 & 37 & 31 \\ D & 77 & 73 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^D(p_{ij})=\left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & B & A & A \\ B & B & Nill & B & \mathbf{C} & B \\ C & B & C & Nill & C & C \\ D & D & \mathbf{C} & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

For $k = \{A, B, C, D, E\}$. Implies, A,B,C D,and E are used as intermediate nodes.

$$D^E(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & 56 & 68 & 28 \\ B & 20 & 0 & 36 & 70 & 30 \\ C & 56 & 36 & 0 & 37 & 31 \\ D & 68 & 70 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \end{array}\right), \qquad P^E(p_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & Nill & A & B & A & A \\ B & B & Nill & B & \mathbf{E} & B \\ C & B & C & Nill & C & C \\ D & D & \mathbf{E} & D & Nill & D \\ E & E & E & E & E & Nill \end{array}\right)$$

$$\begin{aligned} d^E(A,D) &= min\{d^D(A,D), d^D(A,E) + d^D(E,D)\}. \\ &= min\{77, 28+40\} = 68 = d(D,A). \end{aligned} \qquad \begin{aligned} P(A,D) &= P(A,E) + P(E,D). \\ &= P(E,D) = E \end{aligned}$$

$$\begin{aligned} d^E(B,D) &= min\{d^D(B,D), d^D(B,E) + d^D(E,D)\}. \\ &= min\{73, 30+40\} = 70 = d(D,B). \end{aligned} \qquad \begin{aligned} P(B,D) &= P(B,E) + P(E,D). \\ &= P(E,D) = E \end{aligned}$$

Where, $D^E(d_{ij}) = D^*(d_{ij})$

The final matrices D^E and P^E contain the length of all pairs of shortest paths and all the information allowing their reconstruction respectively. Therefore,from the final matrices, we can see that the shortest path from node D to B has length $d(D,B) = 70$ and movies from D to B via $E = Pred(D,B)$.
To determine the associated path,we recall that the path (i,j) represents a direct link only if $p_{ij} = i$. Otherwise, i and j are linked through at least one other intermediate node. This is because, $p_{DB} = E \neq B$,implies that the path from D to B is not direct. Hence the path is initially given as $D \to E \to B$. Now since $p_{DE} = D$ and $p_{EB} = B$,implies the path P is direct from node D to E and from E to B respectively. Therefore, no further dissecting is needed,and the path $D \to E \to B$ defines the shortest path from D to B.

Effectiveness and Finiteness of the algorithm for shortest path problems

To establish the effectiveness and finiteness of the generic all-pairs label correcting algorithm, we assume that the data are integral and that the network contains no negative cycle.

We first consider the effectiveness of the algorithm. At every step the algorithm maintains the invariant property that whenever $d(i,j) < \infty$, the network contains a directed walk of length $d(i,j)$ from node i to node j. We can use induction on the number of iterations to show that this property holds at every step. Now consider the directed walk of length $d(i,j)$ from node i to node j at the point when the algorithm terminates. This directed walk decomposes into a directed path, say P, from node i to node j, and possibly some directed cycles. None of these cycles could have a positive length, for otherwise we would contradict the optimality of $d(i,j)$. Therefore, all of these cycles must have length zero. Consequently, the path P must have length $d(i,j)$. The distance labels $d(i,j)$ also satisfy the optimality conditions (1.4), for these conditions are the termination criteria of the algorithm. This conclusion establishes the fact that when the algorithm terminates, the distance labels represent shortest path distances.
Now consider the finiteness of the algorithm. Since all arc lengths are integer and C is the

largest magnitude of any arc length, the maximum (finite) distance label is bounded from above by nC and the minimum distance label is bounded from below by $-nC$ Each iteration of the generic all-pairs label-correcting algorithm decreases some $d(i, j)$. Consequently, the algorithm terminates within finite number of iterations.
Hence, the algorithms that we have stated to solve the shortest path problems are more effective and efficient because they are simple and easy to implement.

Chapter 2

Location Problem on Weighted Graphs

2.1 Median location problem model

The median location problem locate median facilities on the network G in order to minimize the sum of the weighted distances from each node to its nearest facility. This particular technique is very useful in problems where we are interested in finding the location of p-facilities to serve demand nodes so that the transportation cost is minimized [7].
In median problems, the relationship between the distance between facilities and demand nodes and the cost associated with the facility/demand pair is usually linear. there are no capacity constraints at the facilities and therefore it is optimal to satisfy the demand at a demand node from a single facility. an optimal solution can be found by restricting the search to the demand nodes [7].
The p-median problem is to find the location of p facilities on a network so that the total cost is minimized. Therefore, the cost of serving demands at node i is given by the product of the demand at node $i, (h_i)$ and the distance between demand node i and the nearest facility to node i, (d_{ij}) [7].
That is, transportation cost/weighted cost/length $= h_i d_{ij}$.
Then,the total cost of serving demands at node i (or total transportation cost)is defined as the sum of the weighted distance from node i to all other nodes in the graph G.

$$\sum h_i d_{ij}, \text{ for all } i, j \in G.$$

Definition
Let $G = (V, E)$ be a weighted graph and $d(i, j)$ be the length of the shortest path between two nodes i and j in G. The star distance $S(i)$ of a graph G is defined as the sum of the weighted distance from node i to all other nodes in the graph G[2].

$$S(i) = \sum h_i d_{ij}, \quad \text{for all } i, j \in G.$$

Definition
Let $G = (V, E)$ be a weighted graph. The median of the graph(or Network) is the minimum of the star distance $S(i)$ of the graph G[2].

$$Median\ of\ the\ graph\ G = \min\{S(i) = \sum h_i d_{ij} \mid i \in N(G)\},\ for\ all\ i, j \in G.$$

To formulate the median problem model we define the following inputs

- I = the set of demand nodes indexed by i.
- J = The set of candidate facility locations, indexed by j.
- d_{ij} = distance between demand node i and site j.
- h_i = demand at node i.
- p = the number of facilities to be located.

Decision variables are defined as follows:-
Let X_{ij} be location variable and Y_{ij} the allocation variable.such that,

$$X_{ij} = \begin{cases} 1,\ if\ facility\ i\ is\ located\ at\ site\ j \\ 0,\ Otherwise. \end{cases}$$

$$Y_{ij} = \begin{cases} 1,\ if\ demand\ node\ i\ is\ assigned\ to\ a facility\ at\ \ site\ j. \\ 0,\ Otherwise. \end{cases}$$

Since, $\sum Y_{ij}$, is the total demands served by(or assigned to)a facility at node j,the total demand - weighted distance is the product of the total weighted distance and the total demands served by the facility at node j.
Total demand - weighted distance is computed as follows:-

$$(\textstyle\sum_i h_i d_{ij})(\sum_i Y_{ij}) = \sum_i \sum_j h_i d_{ij} Y_{ij}.$$

Which is the objective function to be minimized.Thus, the p-median problem model can be formulated as follows:-

$$Minimize \quad \sum_{i \in I} \sum_{j \in J} h_i d_{ij} Y_{ij} \tag{2.1}$$

$$Subject\ to: \sum_{j \in J} X_j = p \tag{2.2}$$

$$\sum_{j \in J} Y_{ij} = 1, \quad \forall i \in I \tag{2.3}$$

$$Y_{ij} - X_j \leq 0, \quad \forall i \in I, j \in J \tag{2.4}$$

$$X_j \in \{0, 1\}, \quad \forall i \in I, j \in J \tag{2.5}$$

$$Y_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in J \tag{2.6}$$

The objective function (2.1):- minimizes the total demand-weighted distance between each demand node and the nearest facility.Constraint set (2.2):- states that exactly p facilities are to be located. Constraint set (2.3):- requires that each demand node i is assigned to

exactly one facility j. Constraint set (2.4):- restricts demand node assignments only to open facilities. Constraint (2.5):- established the sitting decision variable as binary. Constraint set (2.6):- requires the demand at a node to be assigned to one facility only.
Constraints (2.4) link the location variables (X_{ij}) and the allocation variables (Y_{ij}). They state that demands at node i can only be assigned to a facility at location j ($Y_{ij} = 1$) if a facility is located at node $j(X_{ij} = 1)$. This constraint is a strong version of the constraint linking the location and allocation variables. If no facility is located at node j ($X_{ij} = 0$), then all of the allocation variables using this facility must be 0 ($i.e,\ \sum Y_{ij} = 0$). That is, vertices cannot be allocated to non p-median vertices.

2.2 Center location problem

The centre location problem considers that a demand point is served by its nearest facility and therefore gives full coverage in the sense of set covering models.

2.2.1 Set covering problem model(SCP)

The set covering problem is to find a set of facilities with minimum cost from among a finite set of candidate facilities so that every demand node is covered by at least one facility. The set covering model minimize the number of facilities needed to cover all demand nodes[7]. A demand node is considered to be covered if the shortest distance (d_{ij}) is less than or equal to the coverage distance r, $(i, e.d_{ij} \leq r)$. Daskin (1995) formulated the set covering model as follows;

- f_j = Cost of locating a facility at candidate site j.

$$c_{ij} = \begin{cases} 1,\ if\ demand\ at\ node\ \text{i}\ can\ cover\ candidate\ site\ \text{j}\ (i, e\ d_{ij} \leq r). \\ 0,\ if\ not\ (i, e\ d_{ij} \geq r). \end{cases}$$

let the decision variables be

$$X_j = \begin{cases} 1,\ if\ we\ locate\ facility\ at\ candidate\ site\ \text{j} \\ 0,\ if\ not \end{cases}$$

With these notations, the set covering problem is formulated as follows;

$$Minimize \quad \sum_{j \in J} f_j X_j = q(r) \tag{2.7}$$

$$\text{subject to}: \quad \sum_{j \in J} c_{ij} X_j \geq 1 \quad \forall i. \tag{2.8}$$

$$X_j = \{0, 1\}, \quad \forall j.$$

The objective function (2.7) minimizes the total cost of the facilities that are selected. Constraints (2.8) stipulate that each demand node i, must be covered by at least one facility. Note that the left-hand side of (2.8) gives the number of located facilities that can cover

demand node i.
If all the costs are identical($f_j = 1$),then the objective function $\sum_{j\in J} f_j X_j = \sum_{j\in J} X_j$. We stipulate a coverage distance r, such that $d_{ij} \leq r$ implies demand node i can be covered by facility j. This affects constraints (2.8), because the relationship between d_{ij} and r will determine whether c_{ij} is 1 or 0.

2.2.2 Center location model

The center location problem locate facilities on the network G in order to minimize the maximum weighted distances from each node to its nearest facility.
In the center location problem, each demand point has a weight. these weights may have different interpretations such as time per unit distance, cost per unit distance or loss per unit distance [7]. So the problem would be seeking a center to minimize a maximum time, cost or loss. The p-center problem [4][5] addresses the problem of minimizing the maximum distance that demand is from its closest facility given that we are siting a pre-determined number of facilities.
There are several possible variations of the center location problem. among these,the vertex p-center problem restricts the facility to be on the nodes of the network; while the absolute p-center problem permits the facilities to be anywhere on the network [7].
Both versions focus on the weighted graphs or networks.In the weighted model, the distances between demand nodes and facilities are multiplied by a weight associated with the demand node.
Definition
Let $G = (V, E)$ be a weighted graph and $d(i, j)$ be the length of the shortest path between two nodes i and j in G. The radial distance $r(i)$ of a graph G is defined as the distance from node i to a vertex(or node) farthest from node i in the graph G.
Definition
Let $G = (V, E)$ be a weighted graph. The center of the graph G is defined as the minimum of the radial distance $r(i)$ of a graph G.
To formulate the center problem we define the following inputs.

- I = the set of demand nodes indexed by i.
- j = The set of candidate facility locations, indexed by j.
- d_{ij} = distance between demand node i and candidate site j.
- h_i = demand at node i.
- p = the number of facilities to locate.

Decision variables

$$X_{ij} = \begin{cases} 1, \ if \ facility \ \text{i} \ is \ located \ at \ candidate \ node \ \text{j} \\ 0, \ Otherwise \end{cases}$$

$$Y_{ij} = \begin{cases} 1, \ if \ demand \ node \ \text{i} \ is \ assigned \ to \ facility \ at \ candidate \ node \ \text{j} \\ 0, \ Otherwise \end{cases}$$

Objective function

Let $r(i)$ = the maximum(or farthest) distance between a demand node i and the nearest facility j to which it is assigned. With this notation, the p-center problem can be formulated as follows:

$$Minimize \quad r(i) \tag{2.9}$$

$$Subject\ to: \quad \sum_{j \in J} X_j = p \tag{2.10}$$

$$\sum_{j \in J} Y_{ij} = 1, \quad \forall i \in I \tag{2.11}$$

$$Y_{ij} - X_j \leq 0, \quad \forall i \in I, j \in J \tag{2.12}$$

$$r(i) - \sum h_i d_{ij} Y_{ij} \geq 0, \quad \forall i \in I \tag{2.13}$$

$$X_j \in \{0, 1\}, \quad \forall j \in J \tag{2.14}$$

$$Y_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in J \tag{2.15}$$

The objective function (2.9):- minimizes the maximum demand-weighted distance between each demand node and its closest facility. Constraint (2.10):- stipulates that p-facilities are to be located. Constraint set (2.11):- requires that each demand node is assigned to exactly one facility. Constraint set (2.12):- restricts demand node assignments only to open facilities. it also state that demands at node i cannot be assigned to a facility at node j unless a facility is located at node j. Constraint set (2.13):- defines the lower bound on the maximum demand-weighted distance, which is being minimized. Constraint set (2.14):- established the sitting decision variable as binary. Constraint set (2.15):- requires the demand at a node to be assigned to one facility only.

Chapter 3

Solution methods

In this section we present algorithms and computational procedures to solve the median and center location problems based on the all pairs shortest path distance computed by the Floyd-Warshal's algorithm in chapter one.

3.1 Median location problem

3.1.1 Median location problem on weighted graph

Let $G = (V, E)$ be a weighted (directed or undirected) graph with edge costs of non-negative real numbers.
Let $a(i, j)$ be the arc length (edge cost) and $d(i, j)$ be the length of the shortest path from i to j. The shortest path from node i to node j is the path with the minimum sum of edge costs over all possible paths connecting nodes i and j.
Let the matrices D and D^* be the matrices whose (i, j) elements are $a(i, j)$ and $d(i, j)$ respectively. The problem of computing D^* is known to be the all-pairs shortest path (APSP)problem.
Let $d(i, j)$ be the length of the shortest path between two nodes i and j in G. The star distance $S(i)$ of a graph G is defined as the sum of the weighted distance from node i to all other nodes in the graph G.
The median point which is the optimal location of the graph(or Network) is the minimum of the star distance $S(i)$ of the graph G.

$$Median\ of\ G = \min\{S(i) = \sum h_i d(i, j)\}, \quad \text{for all } i, j \in G.$$

Once we have constructed the matrix D^* of the all-pairs shortest path distance from the graph (or network) using Floyd-Warshal's algorithm, we compute the sum of the entries in each column of the matrix.
The smallest of such a sum identifies the optimal location(solution) which is the median point of the graph (or Netwok).
Example 3. Now consider the matrices D^* computed from the graph of figure 2 and 3 in chapter one of pages 13 and 14 respectively.

$$D^*(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 5 & 6 & 2 & 3 \\ B & 5 & 0 & 2 & 7 & 8 \\ C & 3 & 8 & 0 & 5 & 6 \\ D & 2 & 4 & 4 & 0 & 1 \\ E & 1 & 3 & 5 & 3 & 0 \\ \hline S(i) & 11 & 20 & 17 & 17 & 18 \end{array}\right), \quad D^*(d_{ij}) = \left(\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 20 & 56 & 68 & 28 \\ B & 20 & 0 & 36 & 70 & 30 \\ C & 56 & 36 & 0 & 37 & 31 \\ D & 68 & 70 & 37 & 0 & 40 \\ E & 28 & 30 & 31 & 40 & 0 \\ \hline S(i) & 172 & 156 & 160 & 215 & 129 \end{array}\right)$$

The last raw shows the sum of the entries in each column of both matrices.
Median of $G_2 = min\{S(i) = \sum_{i=A}^{E} d(i,j)\} = min\{11, 20, 17, 18\} = 11$
This shows that the optimal location is at the column with the minimum sum. Hence, the median of the graph G is at vertex A.
Median of $G_3 = min\{S(i) = \sum_{i=A}^{E} d(i,j)\} = min\{172, 156, 160, 215, 129\} = 129$
This implies that the optimal location is at the column with the minimum sum. Hence, the median of the graph G is at vertex E.

3.1.2 Median location problem on a General weighted Graph

The p-median problem can be solved on a general graph as well as a tree using a number of heuristic algorithms.

- **Heuristic algorithms**

A heuristic algorithms is a procedure that determine the optimal solution to an optimization problems. These heuristics fall into two broad classes of heuristics: construction algorithms and improvement algorithms. Daskin (1995)discus three heuristics: a myopic algorithm,an exchange heuristic and a neighborhood search algorithm.
The myopic algorithm is characterized as a heuristic algorithms, in such a way that, it is a construction algorithm in which we attempt to build a good solution for p-median problems. The myopic algorithm constructs a solution by locating the first facility at the one location that minimizes demand weighted total distance. This objective is calculated through total enumeration of the possible solutions. Subsequent facilities are located in a similar fashion, while holding the previously located facilities constant. The myopic heuristic algorithm that we outline here is, often gives results that are either provably optimal or very close to optimal. This algorithm is called myopic because it is a shortsighted approach for decision-making. It is also simple and easy to understand and apply.
If we were to locate only a single facility on the network, we could easily find the optimal location by enumerating all possible locations and choosing the best (i.e., by total enumeration). Specifically, since we know that at least one optimal solution to any p-median problem consists of locating only on the demand nodes, we could evaluate the 1-median objective function,

$$S_j = \sum_i h_i d_{ij}$$

that would result if we locate at demand node j , for each demand node. We would then choose the location that result in the smallest value of S_j . If we only want to locate a single

facility, it is clear that this approach would give an optimal solution, since we would have tested each possible location.
Suppose now that we are given the location of $(p-1)$ facilities. Let X_{p-1} denote the set of locations of these $(p-1)$ facilities. Also, let $d(i, X_{p-1})$ be the shortest distance between demand node i and the closest node in the set X_{p-1}. Similarly, we let $d(i, j \cup X_{p-1})$ be the shortest distance between demand node i and the closest node in the set X_{p-1} augmented by candidate location j. The best place to locate a single new facility, given that the first $(p-1)$ facilities are located at the sites given in the set X_{p-1}, is at the location j that minimizes;

$$S_j = \sum_i h_i d(i, j \cup X_{P-1}).$$

This approach leads to the myopic algorithm for constructing a solution to the p-median problem on general weighted graph.

Myopic Algorithm for the p-Median Problem

- **Step 1**:-Initialize $k = 0$ and $X_k = \emptyset$,the empty set,
  where(k is the number of facilities we have located so far) and (X_k will give the location of the k facilities that we have located at each stage of the algorithm).

- **Step 2**:- Increment k, the counter on the number of facilities located.

- **Step 3**:- Compute $S_j^k = \sum_i h_i d(i, j \cup X_{k-1})$ for each node j which is not in the set X_{k-1}.
  Note that S_j^k gives the value of the p-median objective function if we locate the k^{th} facility at node j, given that the first $(k-1)$ facility are at the locations given in the set X_{k-1} (and node j is not part of that set).

- **Step 4**:-Find the node $j^*(k)$ that minimizes S_j^k that is, $j^*(k) = argmin\{S_j^k\}$.
  Note that $j^*(k)$ gives the best location for the k^{th} facility, given the location of the first $(k-1)$ facility. Add node $j^*(k)$ to the set X_{k-1}, to obtain the set X_k; that is, set$X_k = X_{k-1} \cup j^*(k)$.

- **Step 5**:-If $k = p$ (i.e., we have located p facilities), **Stop**.
  The set X_p is the solution to the myopic algorithm. If $k < p$, go to step 2.

Analysis and convergent property of myopic Algorithm

The myopic heuristic algorithm works in the following way. Firstly, a facility is located in such a way as to minimize the total cost (the demand-weighted total distance) for all customers. Facilities are then added one by one until p is reached. For this heuristic, the p-facility location that gives the minimum cost is selected, Hence,the algorithm adds as the p- facility locations reduces the total demand weight distance for p-median problem as much as possible holding the locations of the first $p-1$ facilities fixed. the facility sites continue to be added until p- facility sites have been included in to the solution. All candidate facility sites are examined and the one whose addition to the current solution reduces the demand-weighted total distance the most is added to the incumbent(current) solution. The

process continues until the solution includes p- facilities. If the algorithm does not include p-facilities, it forms a loop infinitely many times until it includes p-facilities. If the algorithm includes the p-facilities then it satisfy the termination condition, hence, we say the algorithm converges and have a solution,which is unique solution. Therefore, the set of solution X_p obtained by the myopic algorithm above is unique solution. which is to mean that, we can not find another set different from the set X_p simultaneously whose elements are optimal solution for the p- median problem.

Example:-Consider the general weighted graph shown below.

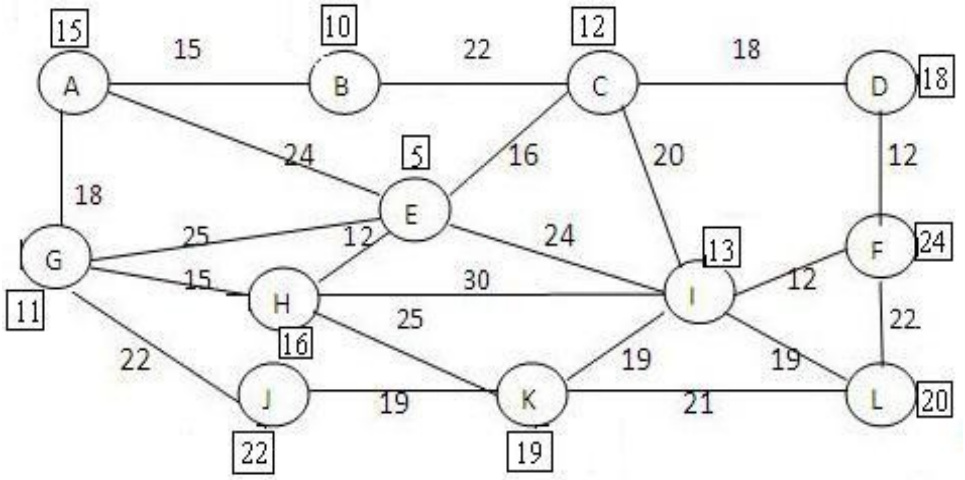

Figure 4:simple general network for p-median

Numbers in boxes next to nodes are the demands(h_i) which are the weight associated with nodes of the network or graph . We need to locate 5-median points. Thus,$p = 5$.

Computing the All pair shortest paths weighted distance Matrix $D(h_i d_{ij})$ for the above Network using Floyd - Warshal's algorithm gives.

$\mathbf{D(h_i d_{ij})} =$

	A	B	C	D	E	F	G	H	**I**	J	K	L
A	0	225	555	825	360	900	270	495	720	600	870	1005
B	150	0	220	400	380	520	330	480	420	550	610	610
C	444	260	0	216	192	360	492	336	240	696	468	468
D	990	720	324	0	612	216	1062	828	432	1116	774	612
E	120	190	80	170	0	180	125	60	120	235	185	215
F	1440	1248	720	288	864	0	1368	1008	288	1200	744	528
G	198	363	451	649	275	627	0	165	495	242	440	671
H	528	768	448	736	192	672	240	0	480	592	400	736
I	624	546	260	312	312	156	585	390	0	494	247	247
J	880	1210	1276	1364	1034	1100	484	814	836	0	418	880
K	1102	1159	741	817	703	589	760	475	361	361	0	399
L	1340	1220	780	680	860	440	1220	920	380	800	420	0
$S(i)$	7816	7913	5855	6457	5784	5760	6936	5971	**4772**	6886	5576	6371

Since S_j^k is the sum of the entries in each column of the matrix, the minimum sum is 4772. that is $j^*(k) = argmin\{S_j^k\} = 4772$. this implies that the first facility ($for\ k = 1$) is located at node I.
The value of the 1-median objective function is the smallest S_j^k value corresponds to $j = I$, that is $S_I^1 = 4772$.
If we locate only one median (node I) for the network (or weighted graph)G, then the optimal total demand-weighted distance can be calculated as follows:

$$\sum_{i=A}^{L}\sum_{j=I} h_i d_{ij} Y_{iI} = h_A d(A,I)Y_{AI} + h_B d(B,I)Y_{BI} + h_C d(C,I)Y_{CI} + h_D d(D,I)Y_{DI}$$
$$+h_E d(E,I)Y_{EI} + h_F d(F,I)Y_{FI} + h_G d(G,I)Y_{GI} + h_H d(H,I)Y_{HI}$$
$$+h_I d(I,I)Y_{II} + h_J d(J,I)Y_{JI} + h_K d(K,I)Y_{KI} + h_L d(L,I)Y_{LI} = 4772.$$

To locate a second median, we need to compute $h_i \times \min\{d(i,I), d(i,j) : j = A, B, C, ..., L\}$ for each node/candidate location pair (i, j).

- $h_A \times \min\{d(A,I), d(A,A)\} = 15 \times \min\{48, 0\} = 15 \times 0 = 0$
  $h_A \times \min\{d(A,I), d(A,B)\} = 15 \times \min\{48, 15\} = 15 \times 15 = 225$
  $h_A \times \min\{d(A,I), d(A,C)\} = 15 \times \min\{48, 37\} = 15 \times 37 = 555$
  $\vdots$
  $h_A \times \min\{d(A,I), d(A,L)\} = 15 \times \min\{48, 67\} = 15 \times 48 = 720$

- $h_B \times \min\{d(B,I), d(B,A)\} = 10 \times \min\{42, 15\} = 10 \times 15 = 150$
  $h_B \times \min\{d(B,I), d(B,B)\} = 10 \times \min\{42, 0\} = 10 \times 0 = 0$

$h_B \times \min\{d(B,I), d(B,C)\} = 10 \times \min\{42, 22\} = 10 \times 22 = 220$
$\vdots$
$h_B \times \min\{d(B,I), d(B,L)\} = 10 \times \min\{42, 61\} = 10 \times 42 = 420$

- $h_C \times \min\{d(C,I), d(C,A)\} = 12 \times \min\{20, 37\} = 12 \times 20 = 240$
  $h_C \times \min\{d(C,I), d(C,B)\} = 12 \times \min\{20, 22\} = 12 \times 20 = 240$
  $\vdots$
  $h_C \times \min\{d(C,I), d(C,L)\} = 12 \times \min\{20, 39\} = 12 \times 20 = 240$

Computing all $h_i \times \min\{d(i,I), d(i,j) : j = A, B, C, ..., L\}$, resulting for second myopic median as follows.

$\mathbf{D(h_i d_{ij})} =$

	A	B	C	D	E	F	**G**	H	I	J	K	L
A	0	225	555	720	360	720	270	495	720	600	720	720
B	150	0	220	400	380	420	330	420	420	420	420	420
C	240	240	0	216	192	240	240	240	240	240	240	240
D	432	432	324	0	432	216	432	432	432	432	432	432
E	120	120	80	120	0	120	120	60	120	120	120	120
F	288	288	288	288	288	0	288	288	288	288	288	288
G	198	363	451	495	275	495	0	165	495	242	440	495
H	480	480	448	480	192	480	240	0	480	480	400	480
I	0	0	0	0	0	0	0	0	0	0	0	0
J	836	836	836	836	836	836	484	814	836	0	418	836
K	361	361	361	361	361	361	361	361	361	361	0	361
L	380	380	380	380	380	380	380	380	380	380	380	0
$S(i)$	3485	3725	3943	4296	3696	4268	**3145**	3655	4772	3563	3858	4392

Computation for second myopic median

The smallest total demand-weighted distance is the minimum of the sum of the entries in each column, which is column G. that is $j^*(k) = argmin\{S_j^k\} = 3145$. this implies that the second facility $(k = 2)$ is located at node G.
Then,the optimal distance when the median at node G is calculated as follows;

$$\sum_{i=A}^{L}\sum_{j=G} h_i d_{iG} Y_{iG} = h_A d(A,G)Y_{AG} + h_B d(B,G)Y_{BG} + h_C d(C,G)Y_{CG} + h_D d(D,G)Y_{DG}$$
$$+h_E d(E,G)Y_{EG} + h_F d(F,G)Y_{FG} + h_G d(G,G)Y_{GG} + h_H d(H,G)Y_{HG}$$
$$+h_I d(I,G)Y_{IG} + h_J d(J,G)Y_{JG} + h_K d(K,G)Y_{KG} + h_L d(L,G)Y_{LG} = 3145$$

The column totals correspond to node G is $S_j^2 = 3145$. To locate a third median facility, we would compute $h_i \times \min\{d(i,G), d(i,I), d(i,j) : j = A, B, C, ..., L\}$ for each node/candidate location pair (i,j). Then, find the column totals corresponding to S_j^3 and the column with

the smallest total, and then locate at the corresponding node. Proceeding in this manner, we obtain the results shown in Table 1 for the first five myopic medians.

medians	location	total demand-weighted distance
1	I	4772
2	G	3145
3	F	2641
4	J	2157
5	A	1707

Table 1:- *Results for the first five myopic medians*

Once we have located some facilities(5-median)whether they are optimal or not, each demand node should be assigned to the nearest facility. This creates sets of nodes such that all nodes in the same set are assigned to the same facility. We refer to the nodes within the set as being in the neighborhood of the facility to which they are assigned.
Note that in assigning demand nodes to facilities, we can break ties arbitrarily. Thus, demand node E is assigned to the neighborhood of the facility at node A even though it could also have been assigned to the neighborhood of the facility located at node I. Within each neighborhood, we would expect that the median would be located optimally. In other words, we expect that the facility serving each neighborhood would be located at the optimal 1-median site for the nodes within the neighborhood. Since finding the optimal 1-median may be done simply by enumeration, we can readily ensure that this condition is satisfied. This leads to the neighborhood search improvement algorithm.

Neighborhood search algorithm

The neighborhood search algorithm can begin with any set of p facility sites. For example, we could begin with the 5 facilities identified by the myopic search algorithm. For each facility site, the algorithm identifies the set of demand nodes that constitute the neighborhood around the facility site. within each neighborhood, the optimal 1-median is found. If any sites have changed, the algorithm reallocates demands to the nearest facility and forms new neighborhoods. If any of the neighborhoods change, the algorithm again finds the 1-median within each neighborhood, and so on.
Figure 5 below is a flowchart of the neighborhood search algorithm. Maranzana (1964) was the first to propose such an algorithm.

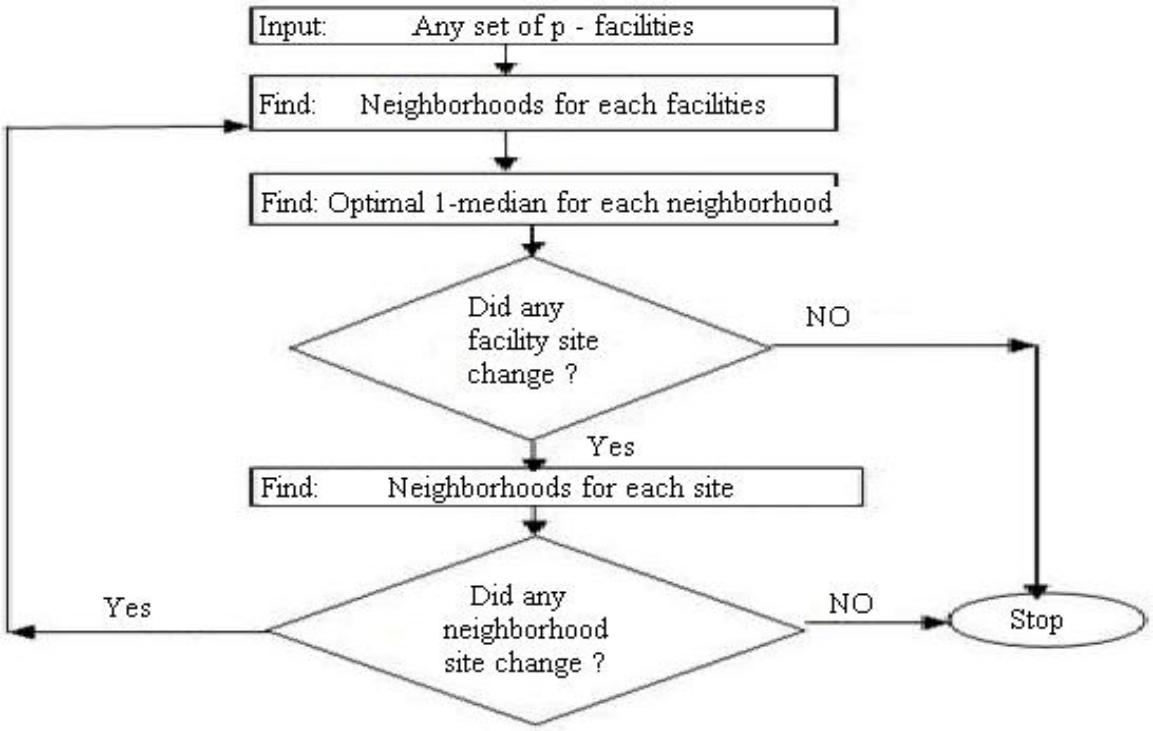

Figure 5:*Flowchart of neighborhood search algorithm.*

Returning to the neighborhoods identified for the myopic 5-median solution shown in Figure 6, the only neighborhood in which the location of the 1-median is suboptimal is that associated with node G. specifically, we move the facility from node G to node H. This reduces the total demand-weighted distance from 1707 to 1632.
Figure 6 displays the neighborhoods associated with the five myopic medians of the network shown in Figure 4 above.

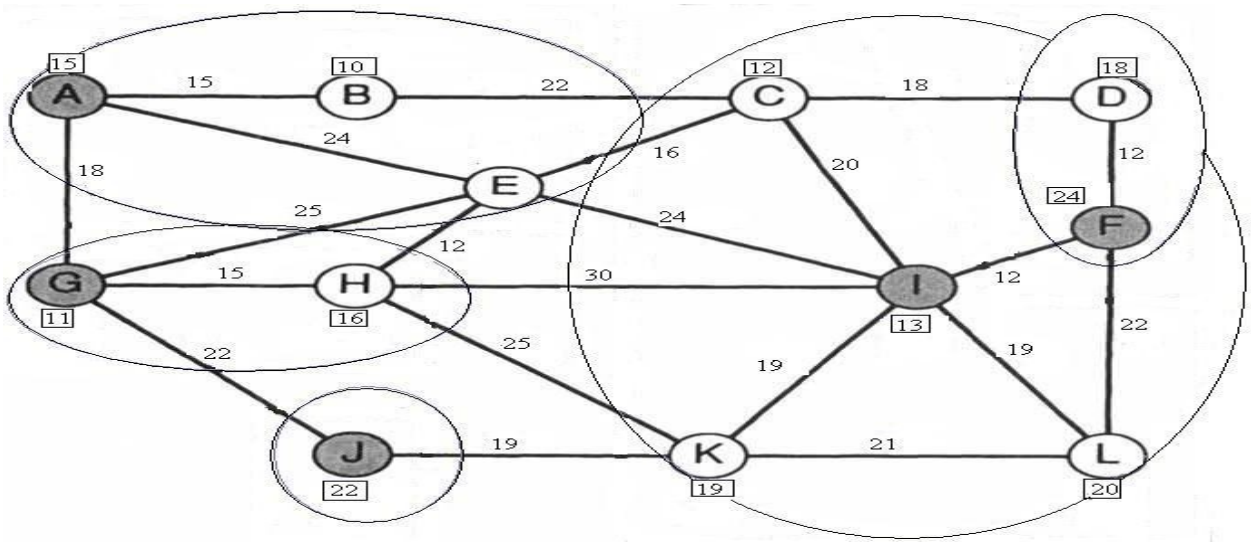

Figure 6:-*Neighborhoods associated with myopic 5-median solution.*

We now attempt to reassign demands to the new set of facilities and find that node E should now be assigned to the facility at node H. This further reduces the total demand-weighted distance to 1572. The resulting neighborhoods are shown in Figure 7.

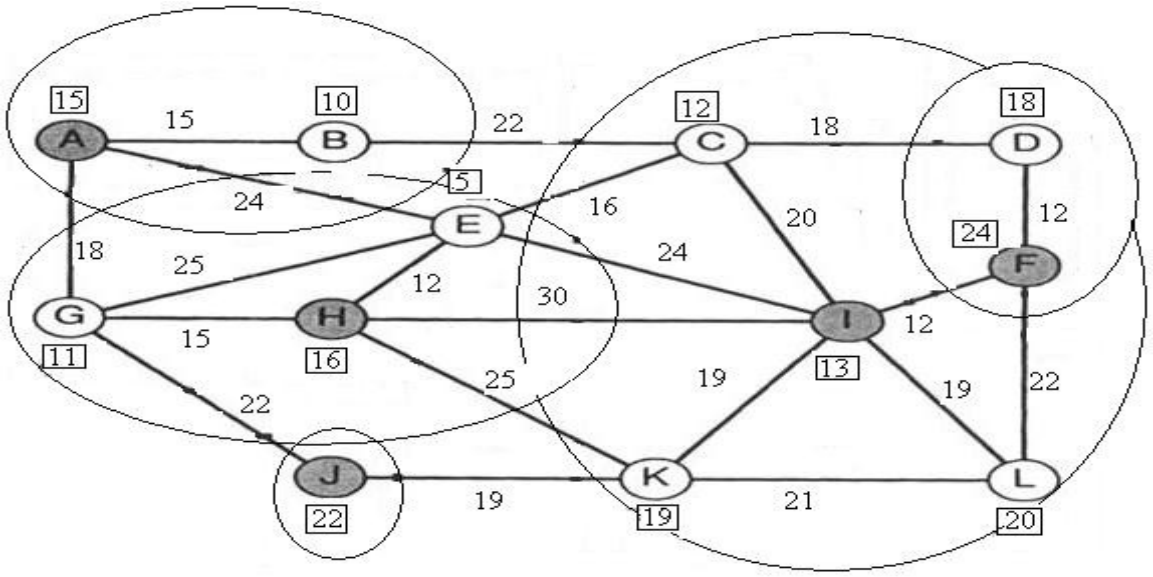

Figure 7:-*New neighborhoods after facility relocation and demand node reassignment.*

Within each neighborhood, we again find the optimal 1-median. Now the 1-median locations do not change and the algorithm stops.
Hence, we retain the solution shown in figure 7. Therefore, nodes A,F,H,I and J are the median of the weighted graph of figure 4 above.
Solution is not always optimal for the p-median problems (except 1-median), therefore this algorithm is used only as first step for other algorithms. The aim of the improvement algorithms is to find an optimal or a sub-optimal solution for the facility location. By integrating the myopic algorithm into one of the improvement algorithms, we can find an optimal solution for p-median problem that minimizes the total demand-weighted distance between the demand nodes and the facilities.

3.2 Center location problem

3.2.1 Center location problem on weighted graph

The p-center problem seeks the location of p-facilities. Each demand point receives its service from the closest facility. The objective is to minimize the maximal(furthest) distance for all demand points. The p-center problem consists of choosing p facilities among a set of possible locations and assigning demand nodes to them in order to minimize the maximum distance between a demand node and the facility to which it is allocated.
Let $G = (V, E)$ be a weighted (directed or undirected) graph with edge costs(or length) of non-negative real numbers. the vertex set $V = \{v_1, v_2, v_3, ..., v_n\}$ and edges are pairs of vertices. Let $a(i, j)$ be the edge cost(or length) from vertex i to vertex j, and $d(i, j)$ be the length of the shortest path from i to j. The shortest path from i to j is the path with the

minimum sum of costs of edges over all possible paths.
Let the matrices D and D^* be the matrices whose (i,j) elements are $a(i,j)$ and $d(i,j)$ respectively. The problem of computing D^* is known to be the all-pairs shortest path (APSP)distance matrix.
The radial distance $r(i)$ of a graph G is defined as the distance from node i to a vertex(or node) farthest from node i in the graph G. Hence,the optimal location which is the center of the graph G is the minimum of the radial distance $r(i)$ of a graph G.

$$Center = \min\{r(i); i = 1 : n\}.$$

Once we have constructed the all-pairs shortest path distance matrix D^* of figure 21 using Floyd-Warshal's algorithm,then we compute the maximum entries in each raw of the matrix. The smallest of such a maximum identifies the optimal location(solution) which is the center of the graph (or Netwok).
Now consider the matrices D^* computed from the graph of figure 2 and 3 in chapter one respectively.

$$D^*(d_{ij}) = \left(\begin{array}{c|cccccc} & A & B & C & D & E & r(i) \\ \hline A & 0 & 5 & 6 & 2 & 3 & 6 \\ B & 5 & 0 & 2 & 7 & 8 & 8 \\ C & 3 & 8 & 0 & 5 & 6 & 8 \\ D & 2 & 4 & 4 & 0 & 1 & 4 \\ E & 1 & 3 & 5 & 3 & 0 & 5 \end{array}\right), \quad D^*(d_{ij}) = \left(\begin{array}{c|cccccc} & A & B & C & D & E & r(i) \\ \hline A & 0 & 20 & 56 & 68 & 28 & 68 \\ B & 20 & 0 & 36 & 70 & 30 & 70 \\ C & 56 & 36 & 0 & 37 & 31 & 56 \\ D & 68 & 70 & 37 & 0 & 40 & 70 \\ E & 28 & 30 & 31 & 40 & 0 & 40 \end{array}\right)$$

The last column of the matrices shows the maximum entries in each raw, which mean the farthest node j in the column from each node i in the raw. Thus the minimum of these maximum entries (or farthest node) identifies the optimal center of the graph.
Center of fig 2= $min\{r(i)\} = min\{6, 8, 4, 5\} = 4$
Therefore,the center of the graph G of figure 2 is node D.
Center of fig 3= $min\{r(i)\} = min\{68, 70, 56, 70, 40\} = 40$
Therefore,the center of the graph G of figure 3 is node E.

Example:-Vertex 1-Center on a weighted tree Network (Daskin 1995)
Consider the weighted tree network in the Fig below.

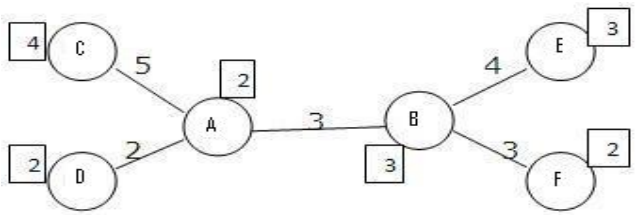

Figure 8:*weighted tree network*

First we need to compute the all pairs shortest path weighted distance matrix $D(h_i d_{ij})$ of the graph using Floyd-Warshal's algorithm.

$$D(h_i d_{ij}) = \left(\begin{array}{c|cccccc|c} & A & B & C & D & E & F & r(i) \\ \hline A & 0 & 6 & 10 & 4 & 14 & 12 & 14 \\ B & 9 & 0 & 24 & 15 & 12 & 9 & 24 \\ C & 20 & 32 & 0 & 28 & 48 & 44 & 48 \\ D & 4 & 10 & 14 & 0 & 18 & 16 & 18 \\ E & 21 & 12 & 36 & 27 & 0 & 21 & 36 \\ F & 12 & 6 & 22 & 16 & 14 & 0 & 22 \end{array}\right)$$

The maximum entry $(r(i))$ in the i^{th} row of the matrix D are 14,24,48,18,36 and 22. then taking minimum of the maximum entry, that is;

$$Center = min(r(i)) = \min\{14, 24, 48, 18, 36, 22\} = 14.$$

Any row with the smallest value the maximum entry identifies a vertex center.that is raw A, which implies v_A is a vertex 1-center on this weighted tree graph.

Set covering problem on weighted graphs

The set covering problem is to find a minimum number of p facilities such that no user is farther than a maximum service-distance away. We first solve the optimal value of the set covering problem when the $d_{ij} \leq r$.

Example. To illustrate the formulation of the set covering problem, we consider the network shown below.

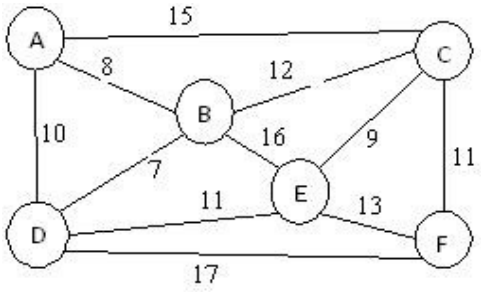

Figure 9:*General weighted graph.*

the matrix of all pairs shortest path distance is computed using Floyd-Warshal's algorithm as follows.

$$D(d_{ij}) = \left(\begin{array}{c|cccccc|c} & A & B & C & D & E & F & r(i) \\ \hline A & 0 & 8 & 15 & 10 & 21 & 26 & 26 \\ B & 8 & 0 & 12 & 7 & 16 & 23 & 23 \\ C & 15 & 12 & 0 & 19 & 9 & 11 & 19 \\ D & 10 & 7 & 19 & 0 & 11 & 17 & 19 \\ E & 21 & 16 & 9 & 11 & 0 & 13 & 21 \\ F & 26 & 23 & 11 & 17 & 13 & 0 & 26 \end{array}\right)$$

The minimum of the maximum entry along the rows gives the the optimal center, that are the 3^{rd} and 4^{th} rows of the matrix. Thus, nodes C and D are the facilities to be located as

center in network or graph.
In the sense of set covering problem,demand is considered to be covered if the distance (d_{ij}) is less than or equal to r,that is $d_{ij} \leq r$. But, if $d_{ij} > r$,then it will be eliminated.
Let the pre-specified distance to be used in this example be $r = 11$.

$$D(d_{ij}) = \left(\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 8 & - & 10 & - & - \\ B & 8 & 0 & - & 7 & - & - \\ C & - & - & 0 & - & 9 & 11 \\ D & 10 & 7 & - & 0 & 11 & - \\ E & - & - & 9 & 11 & 0 & - \\ F & - & - & 11 & - & - & 0 \end{array}\right)$$

Hence,set covering problem is formulated as follows.

$$\begin{aligned} Minimize \quad & X_A + X_B + X_C + X_D + X_E + X_F \\ Subject\ to: \ & X_A + X_B + X_D \geq 1. && Node\ A\ covered \\ : \ & X_A + X_B + X_D \geq 1. && Node\ B\ covered \\ : \ & X_C + X_E + X_F \geq 1. && Node\ C\ covered \\ : \ & X_A + X_B + X_D + X_E \geq 1. && Node\ D\ covered \\ : \ & X_C + X_D + X_E \geq 1. && Node\ E\ covered \\ : \ & X_C + X_F \geq 1. && Node\ F\ covered \\ Integrality\ constraints: \ & X_A, X_B, X_C, X_D, X_E, X_F = \{0, 1\}. \end{aligned}$$

When all of the costs are equal ($i.e. f_j = 1$), we can often reduce the size of the problem using a variety of reduction rules.Hence we use this reduction technique to reduce some rows and columns of the problem.

- **Column reduction**:- For any columns j and k, if $c_{ij} \leq c_{ik}$ for all demand nodes i and $c_{ij} \leq c_{ik}$ for at least one demand node i, then location k covers all demands covered by location j. Location k is said to dominate j and hence column j is eliminated.then we set $X_j = 0$.
- **Row reduction**:- For any row i , if $\sum c_{ij} = 1$ then, there is only one facility site that can cover node i. In such case, we find location j such that $c_{ij} = 1$ and set $X_j = 1$. We then eliminate rows containing X_j. since those constraints with $X_j = 1$ will be satisfied.

In the problem above,candidate site D dominates nodes A and B (since a facility located at D will cover nodes A, B, D, and E, while a facility located at A will only cover nodes A, B, and D, and a facility at B will only cover nodes A, B, and D).Thus, we eliminate column A and B and set $X_A = X_B = 0$. Similarly, candidate site C dominates site F (since a facility located at C covers demand nodes C, E, and F, while a facility at F covers only nodes C

and F). We therefore we eliminate column F and set $X_F = 0$. This leads to:

$$\begin{aligned} Minimize \qquad & X_C + X_D + X_E \\ Subject\ to: \quad & X_D \geq 1 \\ : \quad & X_D \geq 1 \\ : \quad & X_C + X_E \geq 1 \\ : \quad & X_D + X_E \geq 1 \\ : \quad & X_C + X_D + X_E \geq 1 \\ : \quad & X_C \geq 1 \\ Integrality\ constraints: \quad & X_C, X_D, X_E = \{0, 1\}. \end{aligned}$$

Using the row reduction, since $\sum c_{ij} = 1$, holds for the first,second and the last constraints, we set $X_D = 1$ and $X_C = 1$ and eliminate all the rows containing X_D and X_C. The solution therefore is $X_C = X_D = 1$ and $X_A = X_B = X_E = X_F = 0$.

$$\sum\{X_C + X_D\} = 2$$

The objective function equals 2;that is $q(r) = q(11) = 2$ the facility will be located at X_C and X_D.
From the above matrix, Node C covers itself, node E and node F Whereas Node D covers itself, node A, node B and node E.

3.2.2 Center location Problem on a General weighted Graph.

A solution approach to solve the vertex p-center on a general weighted graph is outlined in this section. A general graph is a network $G = (V, E)$ which has at least one cycle. A path in a network is called a cycle if the initial and final vertices in the path are identical.
Example
Consider vertex 2-Center on a General weighted Graph (Francis 1992)

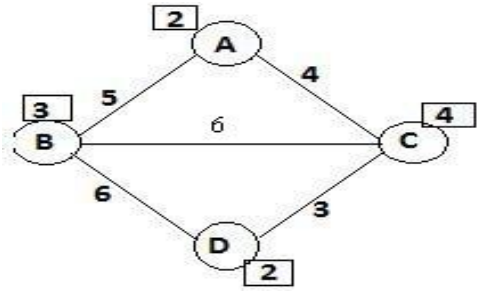

Figure 10:*General weighted graph.*

To find a vertex center,compute the Matrix $D = (h_i d_{ij})$ using Floyd-Warshal's algorithm.

$$D(h_i d_{ij}) = \left(\begin{array}{c|cccc|c} & A & B & C & D & r(i) \\ \hline A & 0 & 10 & 8 & 14 & 14 \\ B & 15 & 0 & 18 & 18 & 18 \\ C & 16 & 24 & 0 & 12 & 24 \\ D & 14 & 12 & 6 & 0 & 14 \end{array} \right), then$$

The maximum entry for each row of the matrix are 14,18,24 and 14. Any row with the smallest value of the maximum entry identifies a vertex center.that is;

$$Center = min(r(i)) = \min\{14, 18, 24, 14\} = 14$$

This implies that nodes A and D are centers on this simple general weighted graph.
For a simple graph, computing the vertex 1-center or 2-center is easy, but when the graph is a more complicated, it becomes more difficult. However we can generalize the approach in the following manner.
The p-centre model considers that a demand point is served by its nearest facility and therefore gives full coverage in the sense of set covering models.
A solution approach to solve the p-center on a general weighted graph is outlined here. We assume that all link distances are integer values. (Since all rational values can be converted to integer values by multiplying them by a sufficiently large number, this is not a restrictive assumption).
In this approach, we search over the range of coverage distances to find the smallest one that allows all nodes to be covered by a vertex center. The search procedure is called binary search.
In this procedure, we define initial lower and upper bounds for the objective function value of the p-center problem. Using the average of the lower and upper bounds, we solve the set covering problem. If the number of facilities needed to cover all nodes is less than or equal to p, we will find out that the objective function of the p-center problem cannot be larger than this coverage distance. So we replace the upper bound to the current coverage distance. If the number of facilities needed to cover all nodes is greater than p, then similarly we will find out that the objective function value of the p-center problem must be larger than the current value, because we have to cover all nodes with a smaller number of centers and hence a larger coverage distance. In this case we would replace the lower bound with the current coverage distance plus 1.
Let $q(r)$ denote the optimal value of the set covering problem when the coverage distance is r. also let B_L and B_U be as lower and upper bounds on the p-center objective function value. The initial values of lower and upper bounds are defined as follows:

$$B_L = 0 \qquad \text{and} \qquad B_U = (n-1)(\max_{ij}(a_{ij}))(\max(h_i))$$

Where n is the number of nodes in the graph and d_{ij} is the length of link (i, j).

Binary search algorithm for the vertex p-center on a general weighted graph:

- **Step 1**:-Set B_L and B_U. Using the above equations. Note that by setting these values in this manner, we make sure that every possible value for the coverage distance is considered.
- **Step 2**:-Set $r = \lfloor \frac{B_U + B_L}{2} \rfloor, Where \lfloor x \rfloor$,denotes the largest integer less than or equal to x.

- **Step 3**:-Solve a set covering problem with a coverage distance of r and let the solution be $q(r)$.
- **Step 4**:-If $q(r) \leq p$, reset B_U to r; else reset B_L to $r+1$.
- **Step 5**:-If $B_U \neq B_L$, go to step 2; otherwise stop.

The lower bound B_L is the optimal value of the objective function and the locations corresponding to the set covering solution for this coverage distance are the optimal locations for the p-centre problem. That is, we stop when $B_L = B_U$ .When the iterations are completed with $B_L = B_U$, we consider the r values and check for the r value that equals the stopping B_L value. The solution is found at the iteration that gives this r value. r is the optimal value for the objective function of p-center problem.

Example:-

To illustrate this algorithm, consider the vertex 2-centre weighted graph below.

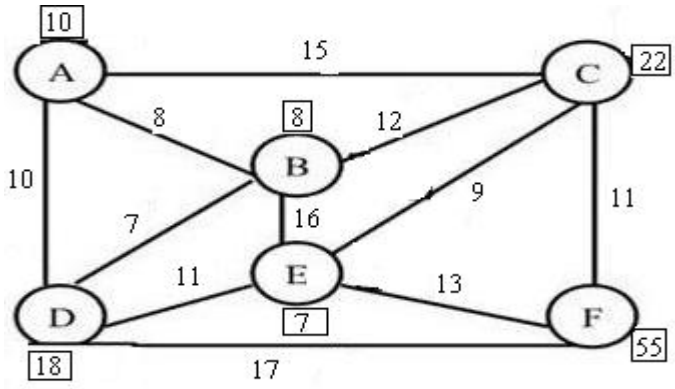

Figure 11:*vertex 2- centre on a weighted graph.*

We solve the vertex 2- centre on a weighted graph as follows. Since the number of nodes are $(n = 6)$, Then

$\Rightarrow$ the lower bound (B_L)=0

$\Rightarrow$ the upper bound $(B_U) = (n-1)(\max_{ij}(a_{ij}))(\max(h_i)) = (6-1)(17)(15) = 4675$

$\Rightarrow$ the coverage distance $(r) = \lfloor \frac{B_U+B_L}{2} \rfloor = \lfloor \frac{4675+0}{2} = \lfloor 2337.5 \rfloor = 2337$

Then,Solve a set covering problem with a coverage distance of r=2337. Note that in solving the set covering problem, demand node i is covered by facility site j only if $h_i d_{ij} \leq r$. So the iteration of vertex 2- centre algorithm is summarized as follows:

iteration	B_L	B_U	r	$q(r)$	location
1	0	4675	2337	2	X_A, X_E
2	0	2337	1168	2	X_A, X_E
3	0	1168	584	2	X_A, X_E
4	0	584	292	2	X_B, X_E
5	0	292	146	3	X_B, X_C, X_F
6	147	292	219	3	X_A, X_E, X_F
7	220	292	255	2	X_A, X_F
8	220	255	237	3	X_A, X_E, X_F
9	238	255	246	2	X_A, X_F
10	238	246	242	2	X_A, X_E
11	238	242	240	3	X_A, X_E, X_F
12	241	242	241	3	X_A, X_E, X_F
13	242	242	stop		

Table 6:- Iteration vertex 2- centre algorithm

We stop at iteration 13, because we have $B_L = B_U$. The solution is at iteration 10 where B_L of iteration 13 equals the coverage distance (r) of iteration 10. The optimal value of the objective function is 242 and the facilities should be located at X_A and X_E.
X_A covers $X_A, X_D and X_B$. Where as X_E covers $X_E, X_C and X_F$.

Summary

In this project we have discussed the location problems on weighted graphs, such as center location problem and median location problems.where,the graph contains no negative cycle. The all-pairs shortest path problem solved by label-correcting algorithms were applied and the Floyd-Warshal's algorithm was used to compute the shortest distance between every pairs of vertices.
Therefore,we use the matrix computed by Floyd-Warshal's algorithm to locate the optimal center and median as facilities. In center location problem we locate p number of facilities on either of the nodes or links for a vertex center and an absolute center problems to minimize the maximum demand-weighted distance between a demand node and the nearest facility to the node. Here,we outlined a binary search technique to solve the center location problem. In median location problems we locate facilities so that the total demand-weighted distance between each demand node and the nearest facility is minimized. To solve median location problems a number of heuristic algorithms, such as simple myopic algorithms,neighborhood search and an exchange algorithm were presented.

References

1. A. B. Sadavare, Dr. R. V. Kulkarni and Maharashtra, A Review of Application of Graph Theory for Network. Vol. 3 (6), 2012.

2. Ahuja, Ravindra K: Network flows: theory, algorithms, and applications, Ravindra K. Ahuja, Thomas L. Magnantl. James B. Orlin. New Jersey,(1993).

3. Barbaros C. Tansel, Richard L. Francis and Timothy J. Lowe: Location on Networks: A Survey. Part I: The p-Center and p-Median Problems. http://www.jstor.org/stable/2630870, Management Science, Vol. 29, No. 4 (Apr., 1983), pp. 482-497.

4. Hakimi, S. L: Optimum location of switching centres and the absolute centres and medians of a graph. Operations Research, Vol. 12, pp 450- 459,(1964).

5. Hakimi S L: Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Operations Research 13, 462475,(1965).

6. Jalali, O. V: Overview of the Algorithms for Solving the P-Median Facility Location Problems. Advanced Studies in Biology (2012).

7. MARK S. DASKIN : Network and Discrete Location Models, Algorithms, and Applications. John Wiley and Sons. New York, Northwestern University (1995).

8. R.Z. Farahani and M. Hekmatfar (eds.), Facility Location: Concepts, Models, Algorithms and Case Studies, Contributions to Management Science,DOI 10.1007/978-3-7908-2151-2 1,©Physica-Verlag Heidelberg 2009.

9. Someah-Addae Ebenezer: Location of Ambulance at Students Residence. Thesis, A Vertex two-Center Problem.2009.

10. Teku Emmanuel; Location of Additional Library Facility. Thesis, Case Study: Berekum Municipality.Kwame Nkrumah University of Science and Technology 2013.