DISTANCE CONSTRAINTS ON CYCLIC NETWORKS: A NEW POLYNOMIALLY SOLVABLE CLASS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hülya Emir

July, 1997

DISTANCE CONSTRAINTS ON CYCLIC NETWORKS: A NEW POLYNOMIALLY SOLVABLE CLASS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL

ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Hülya Emir.

tarafından bağışlanmıştır

By

Hülya Emir

July, 1997

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

Assoc. Prof. Barbaros Tansel(Principal Advisor)

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

Assoc. Prof. Osman Oğuz

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

Assoc. Prof. Mustafa Akgül

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

DISTANCE CONSTRAINTS ON CYCLIC NETWORKS: A NEW POLYNOMIALLY SOLVABLE CLASS

Hülya Emir
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Barbaros Tansel
July, 1997

Distance Constraints Problem is to locate new facilities on a network so that the distances between new and existing facilities as well as between pairs of new facilities do not exceed given upper bounds. The problem is NP-Complete on cyclic networks. The only known polynomially solvable class of distance constraints on cyclic networks is the case when the linkage network, which is an auxiliary graph induced by the distance bounds between new facility pairs, is a tree. In this thesis, we identify a new polynomially solvable class where each new facility is restricted to an a priori specified feasible region which is confined to a single edge and where the linkage network is cyclic with the restriction that there exists a node whose deletion breaks all cycles. We then extend the above class to a more general class where the linkage network has a cut vertex whose blocks fulfill the above assumptions.

Key words: Distance constraints, network location.

ÖZET

GENEL SERİMLERDE UZAKLIK KISITLARI PROBLEMİ: POLİNOM ZAMANDA ÇÖZÜLEBİLİR YENİ BİR SINIF

Hülya Emir
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Doç. Dr. Barbaros Tansel
Temmuz, 1997

Uzaklık Kısıtları Problemi, bir serim üzerinde yeni tesisleri, yeni tesisler ve varolan tesisler arasındaki ve yeni tesis çiftleri arasındaki uzaklıklar belli üst değerleri geçmeyecek biçimde yerleştirme problemidir. Problem çevrimsel serimlerde $NP-zordur$. Çevrimsel serimlerde polinom zamanda çözünürlüğü bilinen durum link seriminin, yeni tesis çiftleri arasındaki uzaklık sınırlarının belirlediği yardımcı çizgenin , bir ağaç olduğu zamandır. Bu tez çalışmasında, polinom zamanda çözülebilir yeni bir sınıf tanımlıyoruz. Bu sınıf her yeni tesisin önceden belirlenmiş olurlu bölgesinin tek bir ayrıtta bulunduğu ve çevrimsel link seriminin, iptal edildiğinde bütün çevrimlerin kırılacağı bir düğüme sahip olduğu varsayımıyla sınırlandığı durumlardır. Daha sonra yukarıdaki sınıfı daha geniş bir sınıf olan öbekleri yukarıdaki varsayımı sağlayan eklem düğümü olan link serimlerine genişletiyoruz.

Anahtar sözcükler Uzaklık Kısıtları, Serim Yerleşimi.

To my mother and my father

ACKNOWLEDGEMENT

I am mostly indebted to Barbaros Tansel who has been supervising me with patience and everlasting interest and being helpful in any way during my graduate studies.

I am also grateful to Osman Oğuz and Mustafa Akgül for showing interest to the subject matter and accepting to read and review this thesis.

I have to express my gratitude to the technical and academical staff of Bilkent University. I am especially thankful to Feryal Erhun, Alev Kaya, Serkan Özkan, Nebahat Dönmez, Bahar Kara and Muhittin Demir for their friendship and encouragement.

Finally, I would like to thank to my parents for everything.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

We consider the problem of locating m new facilities on a transport network so as to satisfy upper bounds on distances between specified pairs of new and existing facilities and specified pairs of new facilities. For convenience we refer to the problem as Distance Constraints Problem (DC). The existing facilities are at nodes of the network. New facilities can be placed anywhere on the network including nodes and interiors of edges. If a distance bound is imposed on a pair of facilities, those facilities are said to interact. Not all facility pairs need to interact but those that do must be placed so as not to violate the imposed upper bounds.

Kolen [7] proved that the problem (DC) is NP-Complete in the strong sense. One polynomial time solvable class in the literature is the special case where the transport network is a tree. Francis, Lowe and Ratliff [9] solved this special case in $O(m(m+n))$ where m and n are the number of new and existing facilities, respectively. Tansel and Yesilkokcen [12] made the first direct attack on the general network version of the problem. They took the transport network to be arbitrary and gave a strongly polynomial algorithm under the assumption that the pairs of new facilities that interact induce an acyclic graph (called LN in the sequel). The time bound of the algorithm is $O(|E|mn(m+logn))$ where $|E|$ is the number of edges of the transport network. This work provided new theory for general networks at the expense

of an assumption on new facility interactions. The tree network theory does not make any assumptions on this part of the data.

In this thesis, we identify a new class of problem instances for cyclic transport networks. For these problem instances, we develop a polynomial time algorithm which constructs a feasible solution to the distance constraints, if there is any.

There appears to be a number of reasons for considering distance constraints. If the new facilities are service facilities of some sort, such as fire stations, then we may wish to require that the fire stations be within a specified driving time (distance) of any point in the region it serves, and then attempt to minimize some objective function. Alternatively, we can envision military scenarios where a number of units cannot be too far from their supply bases and also should not be far from one another, in order that one unit may reinforce another if necessary. With the latter scenario, if a meaningful single objective function is difficult or impossible to obtain, we may possibly be satisfied with one or more feasible solutions from which to make a choice. Also since distance constraints is the recognition form of the minimax multifacility location problem with mutual communication, it deserves the attention given.

Major studies that are most related with our work can be summarized in the following table. Because of its impact on the solvability of the problem, let us define LN and n_j here. Linkage Network, LN is an auxiliary graph that captures the relations between new facilities. The node set of LN is the new facilities set and there is an arc between new facility i and j if there is an upper bound on the distance between them.

There are two kinds of constraints for distance constraint problem, between new and existing facilities and between new facilities. For the former one it is possible to restrict the set of points at which each new facility can be located. The subset of G in which x_j can be located in order to satisfy the first constraint set is called S_j. S_j may consists of many disjoint subedges and n_j is the number of disjoint segments of S_j.

Authors	Year	Assumptions			Explanations
		Trans. Net.	LN	n_j	
Francis, Ratliff & Lowe	1978	TREE	-	-	Gave an algorithm with $O(m(n+m))$
Kolen	1986	-	-	-	Proved that the problem is NP-Complete
Tansel& Yesilkokcen	1993	-	TREE	-	Gave an algorithm with $O(\|E\|mn(m+(m+logn))$
Tansel& Emir	1997	-	a node + subtrees	1	Gave an algorithm with $O(n^3)$

Table 1.1: The studies on DC problem

After providing the algorithm that provides a solution for this restricted problem where each S_j is a convex set restricted to an edge of G and LN is cyclic with the restriction that there exists a node whose deletion breaks all cycles, we will provide some extensions. We will deal with LNs that has a cut vertex whose blocks fulfill the above assumptions. We then relax the assumption on n_j and extend the polynomial time solvability to the case where S_j may consists disjoint segments whose union is restricted on edge of G again.

Now we take a look at the related literature in more detail.

Since we work in a continuous space where new facilities can be located at node or interior points of some edges, we need to work with an embedding of a transportation graph. The following derivation is taken from Dearing, Francis and Lowe [3].

Embedding of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is derived as follows:

For each arc $(v_i, v_j) \in \mathcal{E}$ we assume there exists a one-to-one mapping T_{ij} from the unit interval $[0,1]$ into $\mathcal{S}$ with $T_{ij}(0) = v_i$, $T_{ij}(1) = v_j$, and we define the embedding of (v_i, v_j), denoted by $[v_i, v_j]$, as $T_{ij}([0,1])$. We assume any two embedded arcs intersect at most one common point, a vertex. We define G by $G = \bigcup\{[v_i, v_j] : (v_i, v_j) \in \mathcal{E}\}$ and refer to G as the embedding of $\mathcal{G}$

Denote by ω_{ij} the inverse function of $T_{ij}(\lambda)$, so that ω_{ij} is a one-to-one mapping from $[v_i, v_j]$ onto $[0, 1]$. For $x \in [v_i, v_j]$ we define $[v_i, x] = T_{ij}([0, \omega_{ij}(x)])$ and $[x, v_j] = T_{ij}([\omega_{ij}(x), 1])$ and define the lengths of $[v_i, v_j]$ and $[x, v_j]$ to be $\omega_{ij}(x)e_{ij}$ and $[1 - \omega_{ij}]e_{ij}$ respectively where e_{ij} is the length of edge (v_i, v_j) which is assumed to be positive.

In the network location literature, the most common assumption is that the network of interest is a tree. Dearing, Francis, and Lowe [3] specify some reasons that make tree network location problems tractable. They suggest that the reason has to do with convexity or lack of it. In that study, it is proven that the distance function (the shortest path length) is a convex function for all data choices if and only if the transport network is a tree. Then the multifacility minimax problem with mutual communication (MMMC) whose recognition form is the distance constraints, the subject of this thesis, has a convex objective function as well as a convex constraint set. The lack of convexity is one source of difficulty for cyclic networks. However, problems such as the p-center or p-median are not convex but they are polynomial time solvable when the network is a tree and NP-Complete when the network is cyclic. Hence, convexity provides a partial explanation for hardness of cyclic networks.

For Tree Network

Francis, Lowe, and Ratliff [9](FLR from now on) obtain necessary and sufficient conditions termed separation conditions, for the distance constraints to be consistent. They represent distance constraints by using an auxiliary network, *Network BC* whose node set consists of existing facilities EF_i, $i = 1, 2..n$ and new facilities, NF_j, $j = 1, 2..m$ and whose arc set consists of arcs (EF_i, NF_j) if an upper bound is imposed on the distance between existing facility i and new facility j, and arcs (NF_j, NF_k) if an upper bound is imposed on the distance between new facilities j and k.

It is proven that if the length of the shortest path between EF_i and EF_j on *Network BC*, $\mathcal{L}(EF_i, EF_j)$, is greater than or equal to $d(v_i, v_j)$, the distance between the locations of existing facilities i and j on the transport network

for every pair of existing facilities then the distance constraints are consistent. That is;

$$\text{DC is consistent iff } d(v_i, v_j) \leq \mathcal{L}(EF_i, EF_j) \text{ for } 1 \leq i < j \leq n$$

Moreover,in that study an algorithm is proposed which constructs a feasible solution to the distance constraints if one exists. Then, they used the separation conditions to solve the MMMC problem.

Separation conditions, SC, are necessary and sufficient conditions for tree network but they are only necessary conditions for cyclic networks. Consider the following case. Even though SC are satisfied there exists no feasible location for x that satisfies the DC.

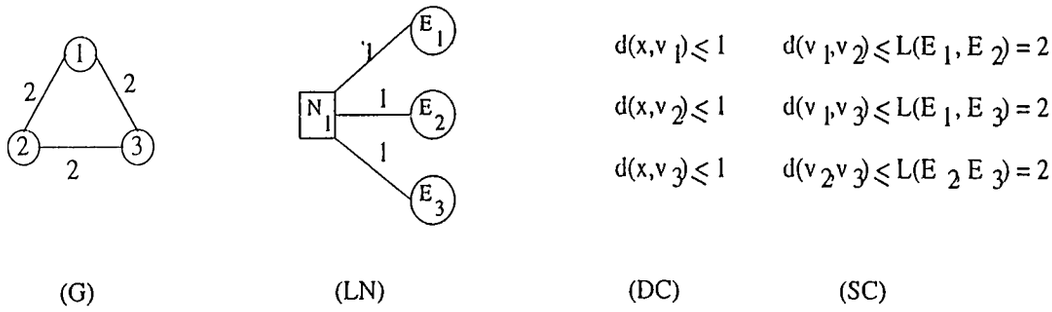


Figure 1.1: Failure of Sufficiency of SC on Cyclic Networks

Many papers followed the study of FRL (1978). These papers mainly use separation conditions and consider binding inequalities and multiobjective multifacility minimax location problems.

One of these works is Tansel, Francis, and Lowe [10]. In that study, tight paths on *Network BC* are defined as follows: Path $P(EF_p, EF_q)$ is a tight path if $\mathcal{L}(EF_p, EF_q) = d(v_p, v_q)$. It is proven that new facility k is uniquely located if and only if NF_k lies on at least one tight path $P(EF_p, EF_q)$ and moreover, if $P(EF_p, EF_q)$ is a tight path, then the nodes representing facilities in the path occur in the same order and spacing in the path as do the locations representing the facilities in $L(v_p, v_q)$, the unique path between v_p and v_q in the transport network.

In that paper, multiobjective optimization problems are also considered.

A test is given which identifies whether a location vector is efficient or not (a vector is efficient if a decrease in one component causes an increase in some other component.) If each NF_i lies in at least one tight path on *Network BC* whose arc lengths are given by the corresponding entries of the location vector, then that location vector is efficient.

In another study, Tansel, Francis, and Lowe [11] concentrated on biobjective multifacility minimax location problem on tree networks. A problem is defined which minimizes the vector whose first entry is the maximum of the distances between specified pairs of new and existing facilities and second entry is the maximum of the distances between specified pairs of new facilities. A necessary and sufficient condition for efficiency is provided. It is as follows: Vector Y is efficient if and only if at least one arc between new facilities is contained in a tight path of GBC_z where $z = f(Y)$ and f is the function to be minimized and GBC_z is the graph corresponding to distance constraint problem whose right hand side is a function of z. An efficient frontier of the biobjective multifacility minimax problem on a tree network is formed.

Erkut, Francis, and Tamir [6] consider $MMMC$ on tree networks in the presence of distance constraints. Their analysis relies on the results obtained in [9] for the distance constraints. The authors propose two algorithms to solve the constrained $MMMC$. The first one is a polynomial algorithm which performs a binary search over the objective value and requires the data to be rational numbers. It uses separation conditions to test feasibility at each step. The second algorithm is strongly polynomial and employs the general parametric approach suggested by Megiddo [8].

The first solution method is a composite algorithm with two main stages. In the first stage an interval of prespecified length that contains the objective function value is found. The second stage calculates the exact optimal value of objective function in that interval. The technique used in both stages is binary search over the objective function value. Sequential Location Procedure, the algorithm proposed by FLR, is used for checking the feasibility of DC at each iteration of the search. Stage two is concluded with one application of a shortest

path algorithm to find the exact value of the optimal objective function.

The second algorithm focused on $F(z)$, the minimum slack of *Network BC* of the problem as a function of the objective function value z. It is proven in that study that the real line can be decomposed into a finite set of intervals on which all $F(z)$ is linear in each interval. The algorithm finds critical values of z by comparing two linear functions and the sign of $F(z)$ is calculated for that point, SLP is used to check that, and the initial interval is reduced until $F(z)$ is linear in the remaining interval. Then the optimal objective function value is easily found.

Erkut, Francis, Lowe, and Tamir [5] consider the multifacility location problem on tree networks subject to distance constraints. All constraints and the onjective functions are arbitrary nondecreasing functions of any finite collection of tree distances between pairs of new and existing facilities and between distinct pairs of new facilities. It is shown in [5] that such problems which, when each function is expressed using only maximization and summation operations on nonnegatively weighted arguements, are linear programming problems of polynomial dimensions. This result may constitute another partial explanation to the question why tree networks are more tractable than cyclic network problems, since they have equivelent mathematical programming formulations while cyclic network versions of the same problems do not.

For Cyclic Networks

Dearing, Francis, and Lowe [3] observed that DC is not convex when the network is cyclic. Kolen [7] proved that DC is NP-Complete for cyclic netowrks. Erkut, ,Francis, Lowe and Tamir [5] stated that the problem posed on a spanning tree is a restriction of the problem on G so that it can be used as an approximation.

Node restricted version of MMMP is solved in polynomial time for the special case when LN is *series-parallel* or a *k-tree* by Chhajed and Lowe [2, 1]. The restriction of new facility locations to a finite set permits enumeration

and the additional assumption on LN allows to carry out the enumeration efficiently. This work is not related with ours, since we work with a continuous space formulation (new facilities can also be located at the interior points of the edges) so we need to follow a nonenumerative way.

The only work related to our study is by Tansel and Yesilkokcen [12, 13]. In these studies, G is taken to be an arbitrary network and a new polynomially solvable class is identified by assuming an acyclic structure for the constraints between new facilities.

First type of distance constraints for new facility j is handled by intersecting the neighbourhoods of existing facilities that j interacts with by the upper bound imposed on the distance between them. The resulting set of points for new facility j is called S_j.

They first define $N(S,r)$, expand of a nonempty set S by r units, where r is a positive constant. Then

$$N(S,r) = \bigcup_{y \in S} N(y,r)$$

Given a pair $(j,k) \in I$ if S_j is expanded by b_{jk} and this expansion is intersected with S_k, then every location x_k in this intersection allows the choice of some x_j in S_j. This operation, $S_k \leftarrow S_k \cap N(S_j, b_{jk})$, is called EXPAND/INTERSECT *Operation from* S_j *into* S_k.

They give an indexing convention for LN, which is assumed to be a tree. They root the tree at a new facility and relabel it as m and relabel the nodes so that the children of j has a smaller indices than j.

SEIP consists of two phases. In the first phase, nodes are processed beginning with leaves of LN and moving toward the root. A given node in any iteration is eligible for processing only if all children of that node are already processed. In processing an eligible node k in some iteartion, the aim is to find the intersection of S_k with expansion of all of its children. Then the new set is denoted by F_k.

If all F_k are nonempty, the second phase is initiated. In the second phase, actual locations of new facilities are found in post order, moving from the root, the last processed node in the first phase, to the childless nodes. A node is eligible for processing in some iteration of the second phase only if its unique parent is already processed. Actually the Expand/Intersect procedure is used in the reverse order and the sets that are obtained at the end of second phase are composite regions for new facilities. That is, if one point is chosen for some j, $\bar{x}_j \in F_j$, then it is possible to construct a feasible location vector whose jth component is $\bar{x}_j$.

The findings of Tansel and Yesilkokcen do not seem to be extendible in any obvious way to cyclic LN.

They first suggest replacing each arc in LN by two arcs with reverse directions, and each iteration starting from a node all the directed arcs are passed (an expand/intersect operation is applied) by a complete tour. The process is repeated until no set is changed from one iteration to other.

But they add that the difficulty with cyclic LN, even if the final sets have been computed somehow, their being nonempty does not imply consistency of DC. Here is an example that they provide to show the failure of cyclic LN.

Assume F_1, F_2, F_3 have been constructed somehow and $F_1 = \{x_1, x_4\}$, $F_2 = \{x_2, x_5\}$ and $F_3 = \{x_3, x_6\}$. Since the example is a small one, one can easily observe by trial and error that no matter how and in which order expand/intersect operations are applied, F_1, F_2, F_3 sets cannot be decreased further. However, DC is inconsistent.

The relation between our study and Tansel and Yesilkokcen [12, 13] will become clear in Section 2.2. Briefly, because of the additional restriction we put on the problem we can analyze the problem parametrically since we can apply expand/intersect operation in a very special way.

Consequently, our study is a relaxation of Tansel and Yesilkokcen in some respect (LN can be a node + subtrees which is more general than an acyclic

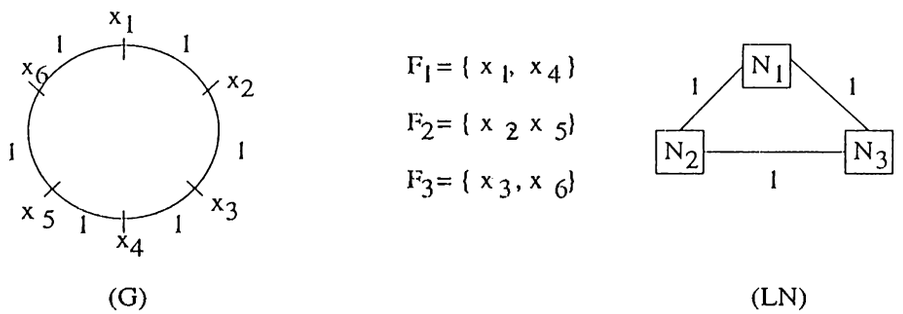


Figure 1.2: Example of Failure for Cyclic LN

structure) but restriction to that study in some other respect (number of disjoint segments of S_j is restricted with 1 in our algorithm while there is no restriction on this part of data in that study).

One superiority of this study over Tansel and Yesilkokcen is the following observation. That study cannot suggest a solution for cyclic LN case (not even an exponential way) but ours is extendible to this general case.

Here we will provide an overview of the rest of the thesis:

In Chapter 2, we will provide an algorithm to DC with three restrictions; in Section 2.2 we will describe our approach and introduce our tools that will be used, in Section 2.3, we will give the algorithm , discuss its complexity and provide an example of its application, in Section 2.4, we will give some methods that will improve the average performance.

In Chapter 3,we will relax the restrictions we put in Chapter 2 to some extent while remaining polynomial.

In Chapter 4, we will summarize our findings and give some future research directions as well.

Chapter 2

ALGORITHM

2.1 PROBLEM DEFINITION

We suppose a connected, undirected cyclic network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given with $\mathcal{V}$ denoting the finite set of nodes and $\mathcal{E}$ denoting the edge set of $\mathcal{G}$.Let $G = (V, E)$ be an embedding of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as defined in Dearing, Francis and Lowe [3]. Each edge $e \in E$ has a positive length. We take G to be the union of all embedded edges. A point x in G is either a node or an interior point of some edge. For any two points x, y in G, let $d(x, y)$ be the shortest path distance between x and y.

Existing facilities are at node locations $v_1, v_2, ...v_n$ in G and m new facilities will be located at points $x_1, x_2, ...x_m \in G$. Let I_C, I_B be given sets of pairs for which distance bounds are of interest. Note that $I_C \subseteq \{(j, i) : 1 \leq j \leq m, 1 \leq i \leq n\}$ and $I_B \subseteq \{(j, k) : 1 \leq j < k \leq m\}$. Given finite positive constants c_{ji} for $(j, i) \in I_C$ and b_{jk} for $(j, k) \in I_B$, the problem of interest is to find locations $x_1, ...x_m \in G$, if they exist, such that

$$d(x_j, v_i) \leq c_{ji} \quad \text{for } (j, i) \in I_C \qquad (DC.1)$$

$$d(x_j, x_k) \leq b_{jk} \quad \text{for } (j, k) \in I_B \qquad (DC.2)$$

We refer to the collection of constraints $(DC.1)$ and $(DC.2)$ as (DC). We

say (DC) is consistent if there is at least one location vector $(x_1, ...x_m) \in G^m$ that satisfies $(DC.1)$ and $(DC.2)$. Here G^m is the m-fold Cartesian product of G with itself.

An alternative formulation for DC is as follows. If we define $N(a,r) = \{x \in G : d(x,a) \leq r\}$ to be the neighborhood around a center a with radius r, then we have an equivalent formulation of distance constraints in terms of neighborhoods.

Let

$$S_j = \bigcap_{i \in I_j} N(v_i, c_{ji}) \text{ where } I_j = \{i : (j,i) \in I_C\}$$

Then DC can be rewritten as:

$$d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I_B$$
$$x_j \in S_j \text{ for } j \in J = \{1, 2..m\}$$

In the case of a tree network, each S_j is a subtree due to the convexity of neighborhoods (FLR [9]). When G is cyclic, each S_j, in general, is a disconnected set consisting of up to $n+1$ segments on a given edge and $O(|E|n)$ disjoint parts on the entire network (Tansel & Yesilkokcen [12]). It is stated in Tansel and Yesilkokcen [12], if n_j is the number of disjoint segments of S_j and S_j^k is the kth disjoint subset in S_j, finding a feasible solution to DC calls for two decisions:

(1) decide which S_j^k each x_j will be in among n_j possible choices.

(2) decide the actual locations of x_j's in their selected sets to satisfy (DC.2)

The resolution of the first decision alone is a major computational challenge. Any enumeration based scheme would have to select S_j^k's among $\prod_{j=1}^m n_j$ possible choices. In the worst case, the total number of selections is $O((|E|n)^m)$ which is computationally prohibitive for large m.

Suppose now the first decision is (somehow) made so that each x_j is restricted to a selected $\overline{S_j} = S_j^{k_j}$ for the jth new facility. We have the restricted problem

$$d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I_B \quad \text{(DC.2)}$$
$$x_j \in \overline{S_j} \text{ for } j \in J \quad (\overline{DC.1})$$

which is called $\overline{DC}$ .This restricted version of the problem is more closely related to the tree network problem, since each $\overline{S_i}$ is now a connected set. It is emphasized by Tansel and Yesilkokcen [12] that despite this resemblance, the restricted problem on G is nontrivial while the problem on a tree is efficiently solvable. There is no method from the existing literature that attempts to solve $\overline{DC}$ on general networks.

In this thesis, the algorithm we propose solves $\overline{DC}$ for some special I_B in polynomial time. If an efficient method is also found for the first decision then this strongly NP-Complete problem can be solved efficiently.

As it was defined in the introduction, Linkage Network, LN is an auxiliary network whose vertex set is the set of new facilities , $M = \{1, 2, ..m\}$, and whose undirected edge set is I_B (I in the sequel).

Let us define the term 'broken wheel' $BW_m = (M, E_m)$. This is a special type of linkage network where the edge set E_m consists of undirected edges $(1,j), j \in J = \{1, 2..m\} \backslash \{1\}$ and $(j, j+1), j \in J \backslash \{m\}$.

DC can be written as follows:

$$d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I$$
$$x_j \in S_j \text{ for } j \in J$$

where

$$I \subseteq \{(j,k) : 1 \leq j < k \leq m\}$$
$$S_j \subseteq G \text{ for } j \in J$$

We provide an algorithm at the end of this chapter that solves this problem with the following restrictions.

- Each S_j is a subedge$[a_j, b_j]$ of some edge $[v_{p_j}, v_{q_j}]$ and $d(a_j, b_j)$ = length of subedge$[a_j, b_j]$.

- $S_j \cap S_k = \emptyset$ for $(j,k) \in I$
- I is chosen so that LN is isomorphic to a subgraph of BW_m

2.2 GENERAL IDEA

Our starting point is the work of Tansel and Yesilkokcen (1993). From this study we know that DC is polynomially solvable when LN is a tree. We tried to modify the algorithm so that it solves DC when LN is a simple cycle. To do that, we fixed the location of x_1. When we fix x_1, facility 1 becomes an existing facility and LN becomes a tree which is solvable. Even though it is easy to solve DC when the location of x_1 is fixed, there are some difficulties in using this algorithm as a subroutine in the solution of DC when LN is a simple cycle.

First of all, there is no finite dominating set of points at which x_1 can be located within its feasible region. So, we may need to repeat this subroutine infinitely many times. This arises from the unpredictable structure of the feasible regions of the other facilities as x_1 moves in a segment of S_1 from one extreme point to the other. Let the solution set be the set of points in S_1 such that if x_1 is located at such a point, then it is possible to find locations for the other facilities that satisfy the constraints. As it will become clearer later, the solution set may consist of disjoint subsets of S_1. That is, the fact that u_1 and u_2 are in the solution set does not guarantee that any point in between is in the solution set. Also, there is no obvious way to characterize the common properties of the points in the solution set.

Therefore, even if it is easy to find solutions to DC if LN is a simple cycle, given that x_1 is located at a fixed point, it is *not* easy to use this information in the parametric study of x_1. But our algorithm still has a close relation with Tansel and Yesilkokcen's algorithm. We also use the expand and intersect operations of that algorithm but the restrictions we put on the problem allows us to use them in a very special way.

The following lemma will be frequently referred to in the following sections:

Lemma 1 *Let $x \in [u,v] \in E$ and $y \in G$ but $y \notin [u,v]$. Then*
$$d(x,y) = Min\{(d(u,x) + d(u,y)), (d(v,x) + d(v,y))\}$$

2.2.1 Reduction of Feasible Regions

Let S_1 be the subedge $[a_1, b_1]$ and S_2 be the subedge $[a_2, b_2]$ whose lengths are l_1 and l_2, respectively. Suppose $(1,2) \in I$ and x_1 is located at x, a point which is in S_1 and whose distance to a_1 is λ $(0 \leq \lambda \leq l_1)$. Define

$$S_2^1(\lambda) = \{y \in S_2 : d(x,y) \leq b_{12}\}$$

That is, $S_2^1(\lambda)$ is the set of points of S_2 which satisfies $d(x_1, x_2) \leq b_{12}$ given that x_1 is located at x.

To calculate $S_2^1(\lambda)$ we need the following definitions.

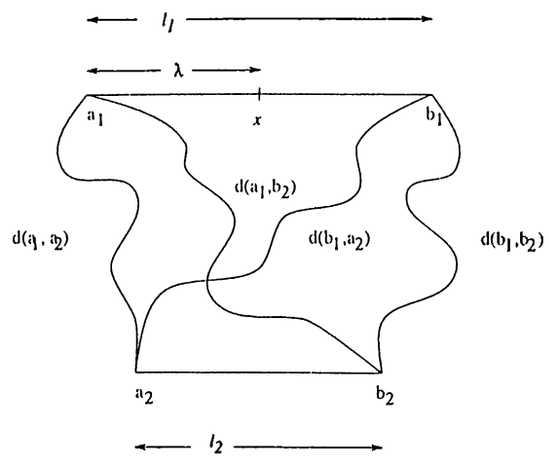


Figure 2.1: Calculation of $L_2^1(\lambda)$ and $R_2^1(\lambda)$

Let

$$L_2^1(\lambda) = Min\{(b_{12} - d(x,a_2)), l_2\}$$
$$R_2^1(\lambda) = Min\{(b_{12} - d(x,b_2)), l_2\}$$

These definitions can be better visualized by means of a string model. Suppose we fasten a string of length b_{12} at point x and pull it tight towards a_2 along the shortest path between x and a_2. If it does not reach, negative of the additional amount that is needed is $L_2^1(\lambda)$; if it reaches we attach the string at a_2. $L_2^1(\lambda)$ is the minimum of the length of the loose end of the string at a_2 and l_2. $R_2^1(\lambda)$ is similar except that the string is now pulled tight from x to b_2.

By using Lemma 1 we can say that if x_1 is located at $x \in S_1$ and there exists $y \in S_2$ that satisfies $d(x,y) \leq b_{12}$ then either a_2 or b_2 or both satisfies the same constraint $min\{d(x,a_2), d(x,b_2)\} \leq d(x,y) \leq b_{12}$ (Since $S_2 \subseteq [u_2, v_2] \in E$ and $S_1 \bigcap S_2 = \emptyset$).

Observation 1

(a) Assume $L_2^1(\lambda) \geq 0$ and let $\overline{L_2^1(\lambda)}$ be the unique point in S_2 which is $L_2^1(\lambda)$ units away from a_2. That is,

$$\overline{L_2^1(\lambda)} \in S_2 \text{ and } d(\overline{L_2^1(\lambda)}, a_2) = L_2(\lambda).$$

Then any point $y \in [a_2, \overline{L_2^1(\lambda)}]$ satisfies $d(x,y) \leq b_{12}$

(b) Assume $R_2^1(\lambda) \geq 0$ and let $\overline{R_2^1(\lambda)}$ be the unique point in S_2 which is $R_2^1(\lambda)$ units away from b_2. That is,

$$\overline{R_2^1(\lambda)} \in S_2 \text{ and } d(\overline{R_2^1(\lambda)}, b_2) = R_2(\lambda).$$

Then any point $y \in [\overline{R_2^1(\lambda)}, b_2]$ satisfies $d(x,y) \leq b_{12}$

Observation 2

$$S_2^1(\lambda) = \begin{cases} \emptyset & \text{if } L_2^1(\lambda) < 0, R_2^1(\lambda) < 0 \\ [a_2, \overline{L_2^1(\lambda)}] & \text{if } L_2^1(\lambda) \geq 0, R_2^1(\lambda) < 0 \\ [\overline{R_2^1(\lambda)}, b_2] & \text{if } L_2^1(\lambda) < 0, R_2^1(\lambda) \geq 0 \\ [a_2, \overline{L_2^1(\lambda)}] \cup [\overline{R_2^1(\lambda)}, b_2] & \text{if } L_2^1(\lambda) \geq 0, R_2^1(\lambda) \geq 0 \end{cases}$$

We will use $S_2(\lambda)$ for $S_2^1(\lambda)$ from now on. For a fixed λ, $S_2(\lambda)$ consists of at most two pieces. But as λ changes in $[0, l_1]$ these pieces may get smaller,

then disappear, then appear again and get larger. Consider the case in Figure 2.2. Given $S_1 = [a_1, b_1]$ and $S_2 = [a_2, b_2]$ with $b_{12} = 10$, we construct $S_2(\lambda)$ for various λ's.

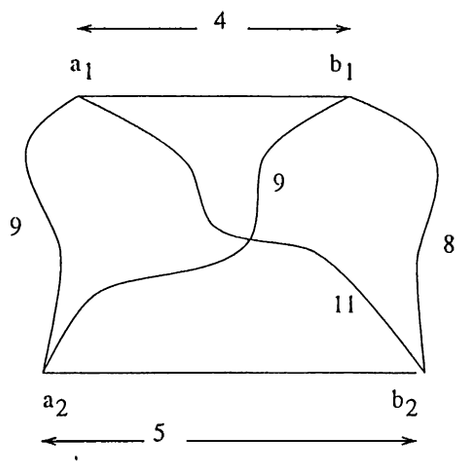


Figure 2.2: Data for example

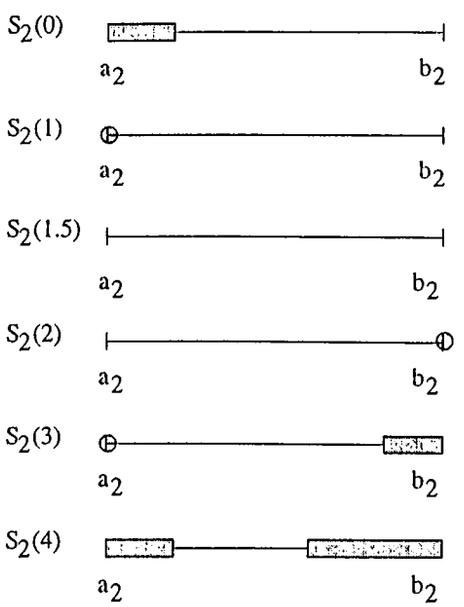


Figure 2.3: $S_2(\lambda)$ values for various λ

This unstructured behavior of $S_2(\lambda)$ somewhat complicates the analysis.

2.2.2 Feasible Region Determination Graph

In the remainder of the thesis, we use λ to mean the point on $[a_1, b_1]$ whose distance from a_1 is λ units, where $0 \leq \lambda \leq l_1$. Even though points on the embedded network are not numbers, the one-to-one correspondence between the points in $[a_1, b_1]$ and their distances from a_1 ensures that λ can be assigned both meanings with an abuse of notation.

Now we will present a graph in Figure 2.4 which is quite insightful. In this graph λ changes in $[0, l_1]$ (corresponding to point a_1 and b_1 in S_1 respectively) and y changes in $[0, l_2]$ (corresponding to points a_2 and b_2 respectively).

If point (λ, y) is in the shaded region, this means that $d(\lambda, y) \leq b_{12}$ (or equivalently $y \in S_2(\lambda)$). In order to partition (λ, y) points into feasible and infeasible regions, we will draw $L_2^1(\lambda)$ and $l_2 - R_2^1(\lambda)$.

The feasible region is the union of the region between $L_2^1(\lambda)$ and x axis where $L_2^1(\lambda) \geq 0$ and the region between $y = l_2$ line and $l_2 - R_2^1(\lambda)$ where $R_2^1(\lambda) \geq 0$. This gives the correct construction of the feasible region as a result of Observation 2.

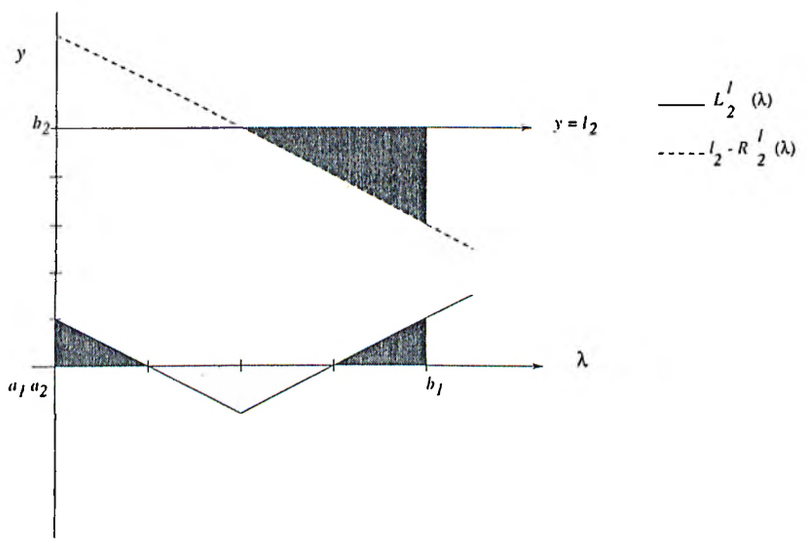


Figure 2.4: Feasible region determination graph

For each fixed $\lambda \in [a_1, b_1]$, if we draw a vertical line through λ, the portion(s)

of this line captured by the shaded region defines the set of all points y in $[a_2, b_2]$ for which $d(\lambda, y) \leq b_{12}$.

It is clear from the figure that for $1 < \lambda < 2$ we cannot find $y \in S_2$ that satisfies $d(x, y) \leq b_{12}$. So we do not need to consider $\lambda \in (1, 2)$ for the feasible region determination of the other facilities since there is no pair (x_1, x_2) that satisfies $d(x_1, x_2) \leq b_{12}$.

In order to convey the ideas of the algorithm, we chose a small example on which we illustrate some of the tools and explain the process of the algorithm.

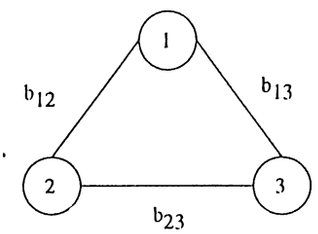


Figure 2.5: Linkage Network of the Example

The aim is to find a location vector $X = (x_1, x_2, x_3)$ which satisfies

$$d(x_1, x_2) \leq b_{12},$$
$$d(x_1, x_3) \leq b_{13},$$
$$d(x_2, x_3) \leq b_{23}$$
$$\text{and } x_j \in S_j \text{ for } j = 1, 2, 3$$

Given that x_1 is located at λ we determine the set of points x_2, x_3 that satisfy the constraints conditional on the fixed location of x_1. We then analyze the consequences parametrically as λ varies in S_1.

First determine

$$S_2(\lambda) = \{y \in S_2 : d(\lambda, y) \leq b_{12}\}$$

and then

$$S_3(\lambda) = \{z \in S_3 : d(\lambda, z) \leq b_{13} \text{ and } d(y, z) \leq b_{23} \text{ for some } y \in S_2(\lambda)\}$$

If $S_3(\bar{\lambda}) \neq \emptyset$ for some $\bar{\lambda}$ then locate x_3 at some point in $S_3(\bar{\lambda})$. Since $x_3 \in S_3(\bar{\lambda})$, $d(x, x_3) \leq b_{13}$ from the definition of $S_3(\lambda)$. Again since $x_3 \in S_3(\bar{\lambda})$

there exists a $y \in S_2(\bar{\lambda})$ such that $d(y, x_3) \leq b_{23}$. Locate x_2 at one such y. Since $x_2 \in S_2(\bar{\lambda})$, $d(x, x_2) \leq b_{12}$. So, after the construction of $S_2(\bar{\lambda})$ and $S_3(\bar{\lambda})$, it is possible to find a location vector that satisfies the constraints. If either of these sets is null, then there is no feasible solution to the constraints when x_1 is fixed at λ.

Let

$$\begin{aligned} S_3^1(\lambda) &= \{z \in S_3 : d(x, z) \leq b_{13}\} \\ S_3^2(\lambda) &= \{z \in S_3 : d(y, z) \leq b_{23} \; for some \; y \in S_2(\lambda)\} \end{aligned}$$

Then,

$$S_3(\lambda) = S_3^1(\lambda) \cap S_3^2(\lambda)$$

We know how to determine $S_3^1(\lambda)$ (same as the determination of $S_2^1(\lambda)$). The following observation provides a method to determine $S_3^2(\lambda)$. Now, for a fixed λ, instead of a unique point, we have a set of points at which y can be located. Let the extreme points of $S_2(\lambda)$ be the end points of the minimal subedge that covers $S_2(\lambda)$ (Note that $S_2(\lambda)$ may consists of disjoint segments).

Observation 3 *Let $y_1(\lambda)$ and $y_2(\lambda)$ be the extreme points of $S_2(\lambda)$. Then $S_3^2(\lambda) = [N(y_1(\lambda), b_{23}) \cup N(y_2(\lambda), b_{23})] \cap S_3$*

Let X and Y be points or set of points and r be a positive number. $N(X, r) \bigcap Y$ is called the expand of X by r and intersection with Y. This operation is called Expand/Intersect operation.

Observation 3 is true since there is no vertex in the interior of S_2 and the intersection of S_2 and S_3 is empty. Therefore, all the paths from $S_2(\lambda)$ need to use one of the extreme point of $S_2(\lambda)$ and consequently, the expansion of the set $S_2(\lambda)$ by b_{12} is achieved by expanding only at the extreme points.

The important conclusion of this observation is that the information of the extreme points of $S_i(\lambda)$ is sufficient to determine $S_j^i(\lambda)$.

Let $y_1(\lambda)$ be a point in $S_2(\lambda)$ which is farthest away from a_2 and $y_2(\lambda)$ be the point in $S_2(\lambda)$ which is farthest away from b_2. Let

$$L_2(\lambda) = d(y_1(\lambda), a_2)$$
$$R_2(\lambda) = d(y_2(\lambda), b_2)$$

It is direct to conclude that $L_2(.)$ ($R_2(.)$) is the upper (lower) envelope of the shaded region of feasible region determination graph of $S_2(\lambda)$ (refer to Figure 2.4). For values of λ where $S_2(\lambda) = \emptyset$, $L_2(\lambda)$ and $R_2(\lambda)$ values will be negative. We will provide a method for calculating those values.

Given $y_1(\lambda)$ and $y_2(\lambda)$ calculation of $S_3^2(\lambda)$ is as follows:

$$L_3^2(\lambda) = Min\{(b_{23} - Min\{d(y_1(\lambda), a_3), d(y_2(\lambda), a_3)\}), l_3\}$$

$$R_3^2(\lambda) = Min\{(b_{23} - Min\{d(y_1(\lambda), b_3), d(y_2(\lambda), b_3)\}), l_3\}$$

We compute these values by using the $L_2(\lambda)$ and $R_2(\lambda)$ functions in the following way:

$$\begin{aligned} L_3^2(\lambda) = Min\{(b_{23} - Min\{ & Min\{L_2(\lambda) + d(a_2, a_3), l_2 - L_2(\lambda) + d(b_2, a_3)\}, \\ & Min\{R_2(\lambda) + d(b_2, a_3), l_2 - R_2(\lambda) + d(a_2, a_3)\}\}, l_3\} \end{aligned}$$

$$\begin{aligned} R_3^2(\lambda) = Min\{(b_{23} - Min\{ & Min\{L_2(\lambda) + d(a_2, b_3), l_2 - L_2(\lambda) + d(b_2, b_3)\}, \\ & Min\{R_2(\lambda) + d(b_2, b_3), l_2 - R_2(\lambda) + d(a_2, b_3)\}\}, l_3\} \end{aligned}$$

Then from Observation 3,

$$S_3^2(\lambda) = \begin{cases} \emptyset & \text{if } L_3^2(\lambda) < 0, R_3^2(\lambda) < 0 \\ [a_3, \overline{L_3^2(\lambda)}] & \text{if } L_3^2(\lambda) \geq 0, R_3^2(\lambda) < 0 \\ [\overline{R_3^2(\lambda)}, b_3] & \text{if } L_3^2(\lambda) < 0, R_3^2(\lambda) \geq 0 \\ [a_3, \overline{L_3^2(\lambda)}] \cup [\overline{R_3^2(\lambda)}, b_3] & \text{if } L_3^2(\lambda) \geq 0, R_3^2(\lambda) \geq 0 \end{cases}$$

where

$$\overline{L_3^2(\lambda)} \in S_3 \text{ and } d(a_3, \overline{L_3^2(\lambda)}) = L_3^2(\lambda)$$
$$\overline{R_3^2(\lambda)} \in S_3 \text{ and } d(b_3, \overline{R_3^2(\lambda)}) = R_3^2(\lambda).$$

As can be seen, regardless of whether $S_i(\lambda)$ is a singleton (as in the case of $S_1(\lambda)$) or a set of points (consisting of a number of pieces) $S_j^i(\lambda)$ shows the same kind of structure.

For a specific λ, $S_i^j(\lambda)$ has at most two pieces, $[a_i, \overline{L_i^j(\lambda)}]$ and $[\overline{R_i^j(\lambda)}, b_i]$

When we intersect $S_3^1(\lambda)$ and $S_3^2(\lambda)$ we get the feasible region determination graph in Figure 2.6.

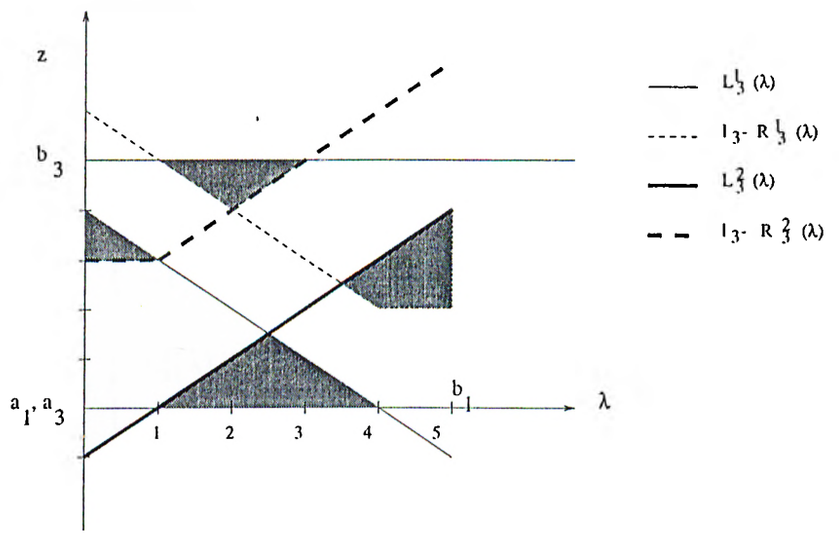


Figure 2.6: $S_3(\lambda)$

If $z \in S_3(\lambda)$, then at least one of the following is true.

$$z \in [a_3, \overline{L_3^1(\lambda)}] \quad and \quad z \in [a_3, \overline{L_3^2(\lambda)}]$$
$$z \in [a_3, \overline{L_3^1(\lambda)}] \quad and \quad z \in [\overline{R_3^2(\lambda)}, b_3]$$
$$z \in [\overline{R_3^1(\lambda)}, b_3] \quad and \quad z \in [a_3, \overline{L_3^2(\lambda)}]$$
$$z \in [\overline{R_3^1(\lambda)}, b_3] \quad and \quad z \in [\overline{R_3^2(\lambda)}, b_3]$$

It is possible to find answers to many questions by considering this graph. For example when we fix λ, the (λ, z) pairs that appear in the shaded region indicate that, there exists a feasible solution when x_1 is located at λ and x_3 is

located at z. When we fix z, we determine the set of λ values at which we can locate x_1. When we project the shaded region onto the x axis, we determine the set of values of λ, which permits a feasible solution. And lastly, when we project the shaded region onto the y axis, we determine the set of z values for which there is a feasible solution.

Suppose LN contains also node 4 and edge (3,4) and we want to determine the extreme points of $S_3(\lambda)$ in order to use them in the determination of $S_4^3(\lambda)$. Observe again that $L_3(\lambda)$ ($R_3(\lambda)$) is the upper (lower) envelope of the shaded region in Figure 2.6.

There are four candidates for $L_3(\lambda)$

$$\mathrm{K}_{\emptyset}(\lambda) = \begin{cases} l_3 & \text{if } R_3^1(\lambda) \geq 0 \text{ and } R_3^2(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{1\}}(\lambda) = \begin{cases} L_3^1(\lambda) & \text{if } L_3^1(\lambda) \geq l_3 - R_3^2(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{2\}}(\lambda) = \begin{cases} L_3^2(\lambda) & \text{if } L_3^2(\lambda) \geq l_3 - R_3^1(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{1,2\}}(\lambda) = Min\{L_3^1(\lambda), L_3^2(\lambda)\}$$

$$L_3(\lambda) = Max\{K_{\emptyset}, K_{\{1\}}(\lambda), K_{\{2\}}(\lambda), K_{\{1,2\}}(\lambda)\}$$

Similarly there are four candidates for $R_3(\lambda)$,

$$\mathrm{M}_{\emptyset}(\lambda) = Min\{R_3^1(\lambda), R_3^2(\lambda)\}$$

$$\mathrm{M}_{\{1\}}(\lambda) = \begin{cases} R_3^2(\lambda) & \text{if} L_3^1(\lambda) \geq l_3 - R_3^2(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_{\{2\}}(\lambda) = \begin{cases} R_3^1(\lambda) & \text{if } L_3^2(\lambda) \geq l_3 - R_3^1(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_{\{1,2\}}(\lambda) = \begin{cases} l_3 & \text{if } L_3^1(\lambda) \geq 0 \text{ and } L_3^2(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$R_3(\lambda) = Max\{M_\emptyset, M_{\{1\}}(\lambda), M_{\{2\}}(\lambda), M_{\{1,2\}}(\lambda)\}$$

Then in order to reduce S_4 to $S_4(\lambda)$ we need to calculate the left and right pieces that $S_3(\lambda)$ forms on S_4 by using $L_3(\lambda)$ and $R_3(\lambda)$.

We say facility j is an in-neighbor of i if $(i,j) \in I$ and S_j is reduced before S_i. Then as the number of in-neighbors of i increases, the number of comparisons needed to be made increases. A point $z \in S_i(\lambda)$ is either in the left piece or right piece of $S_i^j(\lambda)$ where $j \in \Gamma^{-1}(i)$ and $\Gamma^{-1}(i)$ is the set of in-neighbors of i. Since there are two possibilities for each in-neighbor, 2^k comparisons need to be performed in order to determine each of $L_i(\lambda)$ and $R_i(\lambda)$ where k is the in-degree of S_i.

We can formalize it for the general case as follows:

Let $P = \Gamma^{-1}(i)$ be the in-neighbors of i then

$$\mathrm{K}_\emptyset(\lambda) = \begin{cases} l_i & \text{if } Min_{j\in P} R_i^j(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

For $Q \subseteq P$ and $Q \neq \emptyset$

$$\mathrm{K}_Q(\lambda) = \begin{cases} Min_{j\in Q} L_i^j(\lambda) & \text{if } Min_{j\in Q} L_i^j(\lambda) \geq l_i - Min_{j\in P-Q} R_i^j(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$L_i(\lambda) = Max_Q\{K_Q(\lambda)\}$$

Similarly,

$$\mathrm{M}_\emptyset(\lambda) = Min_{j\in P}\{R_i^j(\lambda)\}$$

For $Q \subset P$ and $Q \neq \emptyset$

$$\mathrm{M}_Q(\lambda) = \begin{cases} Min_{j\in P-Q} R_i^j(\lambda) & \text{if } Min_{j\in Q} L_i^j(\lambda) \geq l_i - Min_{j\in P-Q} R_i^j(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_P(\lambda) = \begin{cases} l_i & \text{if } Min_{j\in P} L_i^j(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$R_i(\lambda) = Max_Q\{M_Q(\lambda)\}$$

2.2.3 Out-neighbors of i

Facility j is called an out-neighbor of facility i if $(i,j) \in I$ and j will be processed after i. Let

$$S_i(\lambda) = \{x_i : d(x_i, x_j) \leq b_{ij} \text{ for some } x_j \in S_j(\lambda) \text{ and } j \in \Gamma^{-1}(i)\}$$

The choice of the exact location of x_i does not depend on i's relation with its in-neighbors. As long as $x_i \in S_i(\lambda)$ we can find $x_j \in S_j(\lambda) for j \in \Gamma^{-1}(i)$ that satisfies $d(x_i, x_j) \leq b_{ij}$. Let $\Gamma(i)$ be the set of out-neighbors of i.

If $|\Gamma(i)| = 1$, then our selection will be one of the extreme points of $S_i(\lambda)$ since any other choice of x_i can be replaced by one of the extreme points of S_i with a larger $S_j^i(\lambda)$.

If $|\Gamma(i)| > 1$ (say $|\Gamma(i)| = 2, \Gamma(i) = \{p, q\}$) there are some easy cases that can be handled as well as some hard cases. Let $y_1(\lambda)$ and $y_2(\lambda)$ be the extreme points of $S_i(\lambda)$. $y_1(\lambda)$ reaches S_p means that $N(y_1(\lambda), b_{ip}) \cap S_p \neq \emptyset$

Easy Cases

1. If $y_1(\lambda)$ and $y_2(\lambda)$ do not reach S_p (or S_q) then the problem is infeasible for that λ since no other $x_i \in S_i(\lambda)$ can provide nonempty $S_p^i(\lambda)$ ($S_q^i(\lambda)$).

2. If $y_1(\lambda)$ ($y_2(\lambda)$) reaches both S_p and S_q and $y_2(\lambda)$ ($y_1(\lambda)$) reaches neither S_p nor S_q, then we can locate x_i at $y_1(\lambda)$ ($y_2(\lambda)$) since no other choice will provide a larger $S_p^i(\lambda)$ or $S_q^i(\lambda)$.

Before listing other easy cases, let us consider the following situation. $y_2(\lambda)$ reaches S_p but does not reach S_q and $y_1(\lambda)$ reaches S_q but does not reach S_p.

In the following example, for a fixed $\bar{\lambda}$, $S_i(\bar{\lambda})$ consists of three pieces.

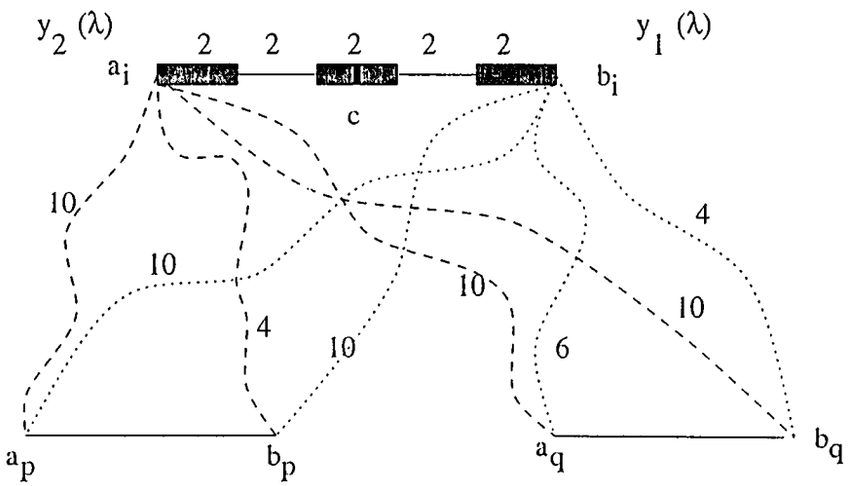


Figure 2.7: Data for the example

The distances on the dashed lines indicate the shortest path distances.

Let $b_{ip} = b_{iq} = 9$ $y_2(\bar{\lambda}) = a_i$ and $y_1(\bar{\lambda}) = b_i$
$y_1(\bar{\lambda})$ does not reach S_p but $y_2(\bar{\lambda})$ reaches S_p

Equivalently,

$$d(y_1(\bar{\lambda}), a_p) > b_{ip} \text{ and } d(y_1(\bar{\lambda}), b_p) > b_{ip}$$
$$\text{and } A_p(\bar{\lambda}) = Max\{b_{ip} - d(y_2(\bar{\lambda}), a_p), b_{ip} - d(y_2(\bar{\lambda}), b_p)\} \geq 0$$

Observation 4 *Suppose* $d(y_1(\bar{\lambda}), a_p) > b_{ip}$ *and* $d(y_1(\bar{\lambda}), b_p) > b_{ip}$ *and* $A_p(\bar{\lambda}) = Max\{b_{ip} - d(y_2(\bar{\lambda}), a_p), b_{ip} - d(y_2(\bar{\lambda}), a_p)\} \geq 0$

Then, point $z_2(\bar{\lambda})$ *is well defined and only points in* $[y_2(\bar{\lambda}), z_2(\bar{\lambda})]$ *provides nonempty* $S_p^i(\bar{\lambda})$ *where*

$$z_2(\bar{\lambda}) \in S_i \text{ and } d(y_2(\bar{\lambda}), z_2(\bar{\lambda})) = A_p(\bar{\lambda})$$

We know that $A_p(\bar{\lambda}) < d(y_1(\bar{\lambda}), y_2(\bar{\lambda}))$. Since otherwise $y_1(\bar{\lambda})$ can also reach S_p. So, $z_2(\bar{\lambda})$ is well defined. The claim is $x \in [y_2(\bar{\lambda}), z_2(\bar{\lambda})]$ provides nonempty $S_p^i(\bar{\lambda})$. When we consider the string model, the length of loose end of string that is attached to either a_p or b_p is $A_p(\bar{\lambda})$ and if we have started from a point

in $[y_2(\bar{\lambda}), z_2(\bar{\lambda})]$ by using $y_2(\bar{\lambda})$, it would reach S_p. Moreover, this set is the only set of points that provide nonempty $S_p^i(\bar{\lambda})$.

Similarly, only points in $[z_1(\bar{\lambda}), y_1(\bar{\lambda})]$ provides nonempty $S_q^i(\bar{\lambda})$ where $z_1(\bar{\lambda}) \in S_i$ and $d(y_1(\bar{\lambda}), z_1(\bar{\lambda})) = A_q(\bar{\lambda})$

Then, for

$$x_i \in M_i(\bar{\lambda}) = S_i(\bar{\lambda}) \cap [y_1(\bar{\lambda}), z_1(\bar{\lambda})] \cap [z_2(\bar{\lambda}), y_2(\bar{\lambda})]$$

we can find x_j such that

$$d(x_i, x_j) \leq b_{ij} \text{ where } j \in \Gamma(i) \cup \Gamma^{-1}(i) \text{ and } x_j \in S_j(\bar{\lambda}).$$

Going back to our example

$$L_p^{iL}(\bar{\lambda}) = -1 \text{ and } R_p^{iL}(\bar{\lambda}) = -1 \; A_p = Max\{L_p^{iR}(\bar{\lambda}), R_p^{iR}(\bar{\lambda})\} = 5 \geq 0$$
$$L_q^{iR}(\bar{\lambda}) = -1 \text{ and } R_q^{iR}(\bar{\lambda}) = -1 \; A_q = Max\{L_q^{iL}(\bar{\lambda}), R_q^{iL}(\bar{\lambda})\} = 5 \geq 0,$$

$$z_1(\bar{\lambda}) = z_2(\bar{\lambda}) = c$$

$$M_i(\bar{\lambda}) = S_i(\bar{\lambda}) \cap [y_1(\bar{\lambda}), c] \cap [c, y_2(\bar{\lambda})] = \{c\}$$

Only $x_i = c \in S_i(\bar{\lambda})$ gives nonempty $S_p^i(\bar{\lambda})$ and $S_q^i(\bar{\lambda})$.

The other easy cases are as follows:

3. When $M_i(\bar{\lambda}) = \emptyset$, there is no feasible solution to the problem.

4. When $M_i(\bar{\lambda})$ is a singleton, that point is the unique point which may be feasible for x_i (we cannot decide on the infeasibility by only considering a part of the data. So the problem may or may not be infeasible but the exact location of x_i is not problematic).

Hard Cases

1. When both $y_1(\bar{\lambda})$ and $y_2(\bar{\lambda})$ reaches both S_p and S_q

2. When $M_i(\bar{\lambda})$ is not a singleton.

In both of the cases, there is no finite dominating set in which x_i should be located on.Then while determining the order of reduction we allow unique out-neighbor for facility j except for the root facility, the facility whose location within its feasible region is parameterized for the analysis (facility 1 in the previous example). We allow multi out-neighbor for root facility since for any λ, $S_{root}(\lambda) = \lambda$, therefore it is a singleton.

2.3 ALGORITHM

Now we are ready to present the algorithm that gives a feasible location vector $(x_1, x_2, ...x_m)$, if it exists, to the constraints

$$d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I$$

$$x_i \in S_i \text{ for } i = 1, 2..m$$

where

$$S_i = [a_i, b_i] \subseteq [u_i, v_i] \in E \text{ and } d(a_i, b_i) = \text{length of } [a_i, b_i]$$

$$LN \text{ is isomorphic to a subgraph of } BW_m.$$

$$S_j \cap S_k = \emptyset \text{ for } (j,k) \in I$$

2.3.1 Main Steps of the Algorithm

1. ORIENTATION

   Input: LN

   Output: Directed LN (DLN), Array A

   In this step, we assign directions to the edges of LN to determine the order of reduction process. We keep this order information in array A.

2. DISTANCE CALCULATION

   Input: Transport Network $G = (E, V)$

Output: $d(a_j, a_k), d(a_j, b_k), d(b_j, a_k), d(b_j, b_k)$ for $(j,k) \in I$

In this step, we calculate distances between extreme points of S_j and S_k where $(j,k) \in I$

3. REDUCTION

Input : DLN,A, $d(a_j, a_k), d(a_j, b_k), d(b_j, a_k), d(b_j, b_k)$ for $(j,k) \in I$

Output: F, $S_j(\lambda)$ for $j \in J$

In this step, we reduce S_j's to $S_j(\lambda)$ where

$S_j(\lambda) = \{x_j \in S_j : d(x_i, x_j) \leq b_{ij}$ for some $x_i \in S_i(\lambda)$ and $i \in \Gamma^{-1}(j)\}$

Equivalently,

$$S_j(\lambda) = \bigcap_{i \in \Gamma^{-1}(j)} N(S_i(\lambda), b_{ij})$$

This step recursively determines the set of points of S_j that satisfy $d(x_i, x_j) \leq b_{ij}$ for facilities j that are processed before i and $(i,j) \in I$. It uses the order specified in the ORIENTATION step and according to this orientation it starts with the root node,r, (the one with zero in-degree in DLN) with $S_r(\lambda) = x$ with $d(x, a_r) = \lambda$. F at step k keeps the set of λ values for which there is a solution to the partial problem k (the set of constraints including only x_j where $j = \Lambda[i]$ $i \leq k$. At any stage, if F becomes empty the algorithm terminates infeasible.

4. CONSTRUCTION

Input: nonempty F,

Output: $(x_1, x_2, ...x_m)$

In this step, with a nonempty F, we construct a location vector $(x_1, x_2, ...x_m)$ that satisfies the distance constraints by moving in the reverse direction that is specified in the ORIENTATION phase.

2.3.2 Explanations of the Steps

1. ORIENTATION PHASE

In this step, we assign directions to the edges of LN to determine the order of reduction. Because of the difficulties that are listed before, we require every node other than the root, to have at most one out-neighbor.

The nodes and their orders are kept in array A. If node j is processed ith then $j = A[i]$.

The aim is to assign $order(i)$ to node i such that

1. $order(i) \in J$

2. $order(i) \neq order(j)$ if $i \neq j$

3. When we assign directions to the edges of $LN = (M, I)$ and obtain $DLN = (M, I')$ in the following way

$$[i,j] \in I \;\Rightarrow\; \begin{cases} (i,j) \in I' & \text{if } order(i) < order(j) \\ (j,i) \in I' & \text{otherwise} \end{cases}$$

then $|\Gamma(i)| \leq 1$ for $i \in J - \{j : order(j) = 1\}$

If for some $i \in J - \{j : order(j) = 1\}$, $\Gamma^{-1}(i) = \emptyset$ then we can add (j, i) to DLN with a large b_{ij}

We call LN 'feasible' if there is at least one order which satisfies the constraints. If the removal of one node from an LN leaves a collection of subtrees then that LN is feasible. Feasible and infeasible graph examples are given in Figure 2.8 and Figure 2.9 respectively.

While stating the problem we said that LN that I imposes should be isomorphic to a subgraph of BW_m. Each subgraph of BW_m is feasible but as you can easily observe from the figures, not every feasible graph is isomorphic to a subgraph of BW_m. Hence, the proposed method handles a more general

class that also includes BW_m's. Further extensions will be given in Chapter 3.

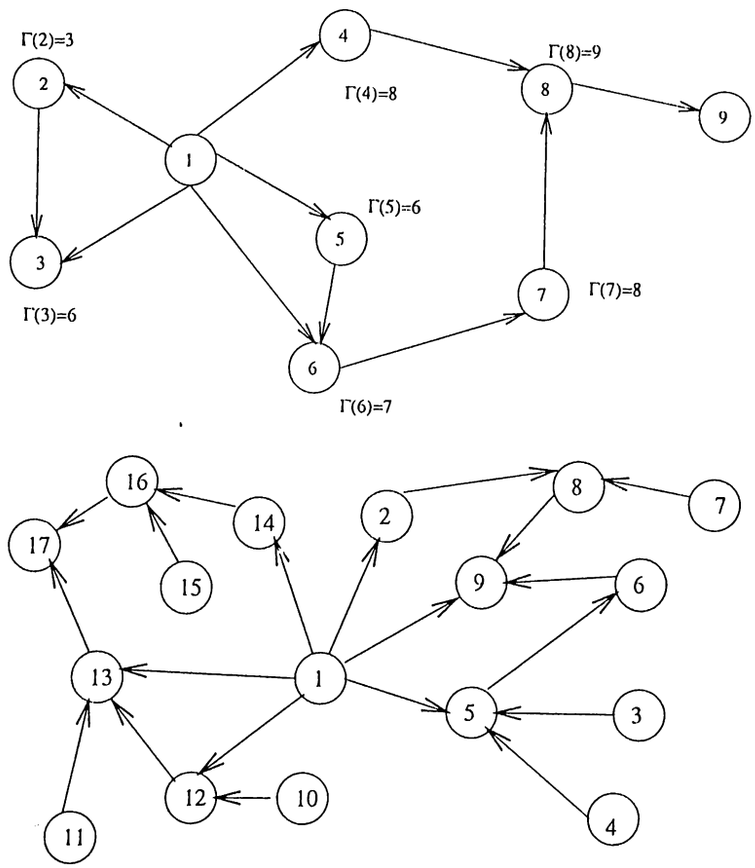


Figure 2.8: Examples of feasible graph

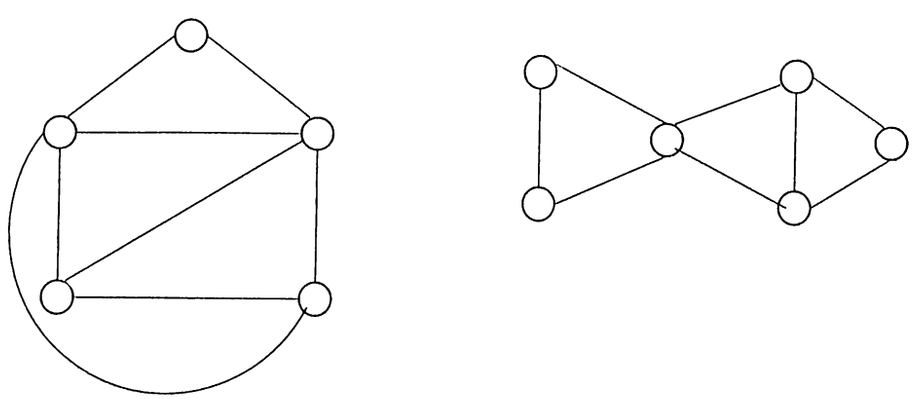


Figure 2.9: Infeasible Graphs

2.DISTANCE CALCULATION PHASE

$$S_j = [a_j, b_j] \subseteq [v_{p_j}, v_{q_j}] \in E$$
$$S_k = [a_k, b_k] \subseteq [v_{p_k}, v_{q_k}] \in E$$

In order to calculate $d(a_j, a_k), d(a_j, b_k), d(b_j, a_k), d(b_j, b_k)$ for $(j, k) \in I$, we first find the shortest path lengths between the nodes of the edges in which these subedges lie and then we calculate the distances of interest by making four comparisons for each distance.

We can use either Floyd's Algorithm $O(n^3)$ or apply Dijkstra's Algorithm $O(n^2)$ for n times in order to determine the shortest path lengths between the vertices of G (note that edge lengths are all nonnegative). But Floyd's Algorithm may produce some redundant information. Consider the following case:

Let $\overline{V} = \bigcup_{j=1}^{m} \{v_{p_j}, v_{q_j}\}, \overline{V} \subseteq V$ and let $|\overline{V}| = k << n$ then applying Dijkstra for each of $v_j \in \overline{V}$ to find $d(v_j, v_i)$ where $v_i \in V$ requires $O(kn^2)$ operations whereas applying Floyd's algorithm costs $O(n^3)$ which is not economical if k is much more smaller than n.

If $S_j, S_k \subseteq [v_p, v_q]$ with $p < q$ then the shortest path between the extreme points is the difference between their $\omega_{pq}(..)$ values.

If we rename the extreme points so that the one that is closer to the vertex with smaller index is a_j, we reduce the number of possible cases a lot.

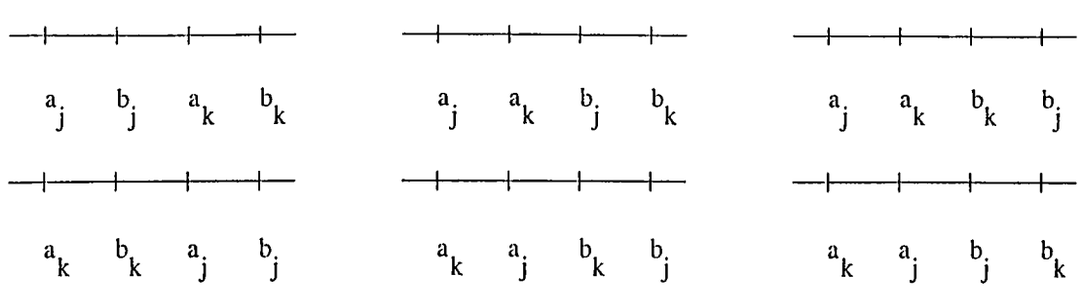


Figure 2.10: Possible orders of a_j, a_k, b_j, b_k on edge $[v_p, v_q]$

Let

$$a = Max\{\omega_{pq}(a_j), \omega_{pq}(a_k)\}$$
$$b = Min\{\omega_{pq}(b_j), \omega_{pq}(b_k)\}$$

If $a > b$ then $S_j \cap S_k = \emptyset$

$$d(u,v) = |\omega_{pq}(u) - \omega_{pq}(v)| \text{ where } u \in \{a_j, b_j\} \text{ and } u \in \{a_k, b_k\}$$

If $a \leq b$ then $S_j \cap S_k \neq \emptyset$

We assumed that $S_j \cap S_k = \emptyset$ at this moment. So whenever such a situation occurs, this algorithm gives a message that $S_j \cap S_k \neq \emptyset$ for $(j,k) \in I$.

3.REDUCTION PHASE

Given DLN and the distances between the extreme points of S_j and S_k for $(j,k) \in I$ we will narrow S_j to $S_j(\lambda)$, the set of points x_j in S_j such that there exists a $x_i \in S_i(\lambda)$ with $d(x_j, x_i) \leq b_{ij}$ $\forall i \in \Gamma^{-1}(j)$. To do that we calculate left and right pieces that each in-neighbor of i forms in S_i. These are calculated by expanding the extreme points of the in-neighbors and intersecting with S_i.

Then we determine the extreme points of $S_i(\lambda)$ in order to use in the determination of S_k where $\{k\} = \Gamma(i)$. While processing any i, the algorithm removes λ's from any further consideration if it causes some infeasibility up to that point. Whenever there is no λ left the algorithm states that the problem is infeasible.

4. CONSTRUCTION PHASE

If the reduction phase ends with nonempty F, this phase constructs a feasible solution for any $\lambda \in F$. It first chooses a $\bar{\lambda} \in F$, and locates $x_r = x$,where $x \in S_r$ and $d(x, a_r) = \bar{\lambda}$. It locates that x_j which has no out-neighbor at one of the extreme points of $S_j(\bar{\lambda})$, even if all points work, we choose the extreme points to provide easiness in the proof. Then trace back the in-neighbors of j, say i, to find out from which extreme point of $S_i(\bar{\lambda})$ x_j is reachable. From the reduction phase it is guaranteed that there exists at least one extreme point of $S_i(\bar{\lambda})$ that can reach the extreme point of $S_j(\bar{\lambda})$ at which

x_j is located, then locate x_i at that extreme point. Since each x_i has at most one out-neighbor, there cannot be any conflict since the information source is unique.

We can partition the node set of DLN into sets so that each set includes one node k with out-degree 0 and all other nodes are on exactly one path from r to k. The structure of DLN guarantees that no one node appear more than one set (Since each node's out-degree is at most one). Then for each set we will start with k and move backward to r, locate the in-neighbors of k first and then in-neighbors of the ones that are located until all nodes in that set are located.

Let us define the following sets. They will be used in the construction phase:

S = Set of nodes of DLN whose out-degrees are zero

Located = Set of nodes that are located

P = Subset of *located* whose in-neighbors are not located

W_i = Unlocated in-neighbors of i, where $i \in P$

For $i \in P$ $W_i = \Gamma^{-1}(i) - Located$

2.3.3 Algorithm

1. ORIENTATION PHASE

Step 0 $k = 2$ $A = [0, 0....0]$\{ an array with size m\}, $R = \emptyset$ $DLN = (M, I')$

$Root = \{i \in M : d(i) = Max_{j \in J}\{d(j)\}\}$, $D = \{i \in M : d(i) = 1\}$

Choose $i \in Root$, $R \Leftarrow R \cup \{i\}$, $A[1] \Leftarrow i$

if $[i, j] \in I$ then $(i, j) \in I'$

Step 1 $Q = (M - R) \cap \Gamma(R)$

Step 2 Choose a node $i \in Q$

Step 3 Check $N_i = \{j \in M - R : [i, j] \in I\}$

(a) If $|N_i| > 1$ then $Q \Leftarrow Q - \{i\}$ Go to 4

(b) If $|N_i| \leq 1$ then Go to 5

Step 4 (a) If $Q = \emptyset$ then

If $D - R = \emptyset$ STOP LN is not feasible.

Else Choose $j \in D - R$

Add $(A[1], j)$ to I' with $b_{A[1]j} = BIG$

$i \Leftarrow j$ Go to 5.

(b) If $Q \neq \emptyset$ then Go to 2

Step 5 $R \Leftarrow R \cup \{i\}$

If $|N_i| = 1$ and $j = N_i$ then add (i, j) to I'

$A[k] \Leftarrow i$

Step 6 (a) If $k = m$ STOP. LN is feasible.

(b) If $k < m$ then $k \Leftarrow k + 1$ Go to 1.

2. DISTANCE CALCULATION PHASE

STEP 1

$$S_j = [a_j, b_j] \subseteq [v_{p_j}, v_{q_j}] \in E \text{ for } j \in J$$

Let $\overline{V} = \bigcup_{j=1}^{m} \{v_{p_j}, v_{q_j}\}$

Apply Dijkstra for every $v \in \overline{V}$ to determine $d(v_p, v_q)$ for every $v_p, v_q \in \overline{V}$

STEP 2

For every $(j,k) \in I$

$$S_j = [a_j, b_j] \subseteq [v_{p_j}, v_{q_j}] \in E$$
$$S_k = [a_k, b_k] \subseteq [v_{p_k}, v_{q_k}] \in E$$

Let length of $[v_{p_j}, v_{q_j}]$ and $[v_{p_k}, v_{q_k}]$ be L_j and L_k respectively.

1. If $|\{v_{p_j}, v_{q_j}\} \cap \{v_{p_k}, v_{q_k}\}| \leq 1$

$$\begin{aligned} d(u,v) \quad = Min\{ \quad & \omega_{p_j q_j}(u).L_j + d(p_j, p_k) + \omega_{p_k q_k}(v).L_k, \\ & \omega_{p_j q_j}(u).L_j + d(p_j, q_k) + (1 - \omega_{p_k q_k}(v)).L_k, \\ & (1 - \omega_{p_j q_j}(u)).L_j + d(q_j, p_k) + \omega_{p_k q_k}(v).L_k, \\ & (1 - \omega_{p_j q_j}(u)).L_j + d(q_j, q_k) + (1 - \omega_{p_k q_k}(v)).L_k\} \end{aligned}$$

where $u \in \{a_j, b_j\}$ and $v \in \{a_k, b_k\}$

2. If $|\{v_{p_j}, v_{q_j}\} \cap \{v_{p_k}, v_{q_k}\}| = 2$

Let $\{v_{p_j}, v_{q_j}\} \cap \{v_{p_k}, v_{q_k}\} = \{v_p, v_q\}$ and Length of $[v_p, v_q] = L$

$$a = Max\{\omega_{pq}(a_j), \omega_{pq}(a_k)\}$$
$$b = Min\{\omega_{pq}(b_j), \omega_{pq}(b_k)\}$$

(a) If $a > b$ then $S_j \cap S_k = \emptyset$ and

$$d(u,v) = |\omega_{pq}(u) - \omega_{pq}(v)|.L$$

where $u \in \{a_j, b_j\}$ and $u \in \{a_k, b_k\}$

(b) If $a \leq b$ then $S_j \cap S_k \neq \emptyset$STOP.

Give a message that $S_j \cap S_k \neq \emptyset$ for $(j,k) \in I$

3.REDUCTION PHASE

Step 0 $r = A[1]$, $k = 2$, $F = [0, l_r]$

Step 1 $i \Leftarrow A[k]$

For $j \in \Gamma^{-1}(i)$ and $j \neq r$

$$L_i^{jL}(\lambda) = Min\{(b_{ij} - d(\overline{L_j(\lambda)}, a_i)), l_i\}$$
$$L_i^{jR}(\lambda) = Min\{(b_{ij} - d(\overline{R_j(\lambda)}, a_i)), l_i\}$$
$$R_i^{jL}(\lambda) = Min\{(b_{ij} - d(\overline{L_j(\lambda)}, b_i)), l_i\}$$
$$R_i^{jR}(\lambda) = Min\{(b_{ij} - d(\overline{R_j(\lambda)}, b_i)), l_i\}$$

Determine $\forall j \in \Gamma^{-1}(i)$

$$\mathrm{L}_i^j(\lambda) = \begin{cases} Min\{(b_{ir} - d(x, a_i)), l_i\} & \text{if } j = r \\ Max\{L_i^{jL}(\lambda), L_i^{jR}(\lambda)\} & \text{if } j \neq r \end{cases}$$
$$\mathrm{R}_i^j(\lambda) = \begin{cases} Min\{(b_{ir} - d(x, b_i)), l_i\} & \text{if } j = r \\ Max\{R_i^{jL}(\lambda), R_i^{jR}(\lambda)\} & \text{if } j \neq r \end{cases}$$

Step 2 $P = \Gamma^{-1}(i)$

$$\mathrm{K}_\emptyset(\lambda) = \begin{cases} l_i & \text{if } Min_{j \in P} R_i^j(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

For $Q \subseteq P$ and $Q \neq \emptyset$

$$\mathrm{K}_Q(\lambda) = \begin{cases} Min_{j \in Q} L_i^j(\lambda) & \text{if } Min_{j \in Q} L_i^j(\lambda) \geq l_i - Min_{j \in P-Q} R_i^j(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$L_i(\lambda) = K_{Q_i(\lambda)}(\lambda) = Max\{K_Q(\lambda)\}$$

Similarly,

$$\mathrm{M}_{\emptyset}(\lambda) = Min_{j \in P}\{R_i^j(\lambda)\}$$

For $Q \subset P$ and $Q \neq \emptyset$

$$\mathrm{M}_Q(\lambda) = \begin{cases} Min_{j \in P-Q} R_i^j(\lambda) & \text{if } Min_{j \in Q} L_i^j(\lambda) \geq l_i - Min_{j \in P-Q} R_i^j(\lambda) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_P(\lambda) = \begin{cases} l_i & \text{if } Min_{j \in P} L_i^j(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$R_i(\lambda) = M_{Q_i(\lambda)}(\lambda) = Max\{M_Q(\lambda)\}$$

Step 3 Determine $\Lambda = \{\lambda : L_i(\lambda) < 0 \text{ and } R_i(\lambda) < 0\}$

Step 4 $F \Leftarrow F - \Lambda$

Step 5 (a) If $F = \emptyset$ STOP INFEASIBLE

(b) Else if $k = m$ STOP FEASIBLE. Go to CONSTRUCTION step

(c) Else $k \Leftarrow k + 1$ Go to 1.

4.CONSTRUCTION PHASE

Step 0 $Located = P = \emptyset$

Choose a $\bar{\lambda} \in F$, locate $x_r = x$ where $x \in S_r$ and $d(x, a_r) = \bar{\lambda}$

$Located \Leftarrow Located \cup \{r\}$

Step 1 If $S \neq \emptyset$, Choose $k \in S$, $P = \{k\}$

Locate $x_k = \overline{L_k(\bar{\lambda})}$ or $\overline{R_k(\bar{\lambda})}$ arbitrarily

Step 1.1 If $P \neq \emptyset$ Choose $i \in P$

$W_i = \Gamma^{-1}(i) - Located$

If $x_i = \overline{L_i(\bar{\lambda})}$ Determine $K_{Q_i(\bar{\lambda})}(\bar{\lambda})$ that determines $L_i(\bar{\lambda})$

If $x_i = \overline{R_i(\bar{\lambda})}$ Determine $M_{Q_i(bar\lambda)}(\bar{\lambda})$ that determines $L_i(\bar{\lambda})$

Step 1.1.1 If $W_i \neq \emptyset$ Choose $j \in W_i$

Step 1.1.1.1 If $j \in Q_i(\bar{\lambda})$ and $L_i^j(\bar{\lambda}) = L_i^{jL}(\bar{\lambda})$
or $j \notin Q_i(\bar{\lambda})$ and $R_i^j(\bar{\lambda}) = R_i^{jL}(\bar{\lambda})$
Locate x_j at $\overline{L_j(\bar{\lambda})}$

Step 1.1.1.2 If $j \in Q_i(\bar{\lambda})$ and $L_i^j(\bar{\lambda}) = L_i^{jR}(\bar{\lambda})$
or $j \notin Q_i(\bar{\lambda})$ and $R_i^j(\bar{\lambda}) = R_i^{jR}(\bar{\lambda})$
Locate x_j at $\overline{R_j(\bar{\lambda})}$

$Located \Leftarrow Located \cup \{j\}$

$P \Leftarrow P \cup \{j\}$

$W_i \Leftarrow W_i - \{j\}$ Go back to Step 1.1.1

Step 1.1.2 If $W_i = \emptyset$

$P \Leftarrow P - \{i\}$ Go back to Step 1.1

Step 1.2 If $P = \emptyset$

$S \Leftarrow S - \{k\}$ Go back to Step 1

Step 2 If $S = \emptyset$ STOP . All facilities are located.

Theorem 1 *Let* $(\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_m)$ *be the location vector which is constructed by the algorithm. Then* $(\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_m)$ *is a feasible vector to problem P1. That is,*

$$d(\tilde{x}_j, \tilde{x}_k) \leq b_{jk} \text{ for } (j,k) \in I$$
$$\tilde{x}_i \in S_i \text{ for } i = 1, 2, ..m$$

Proof If $\tilde{X}$ is constructed by the algorithm then reduction phase ends with nonempty F and $\tilde{X}$ is constructed for some $\lambda \in F$

Suppose $\tilde{X}$ is constructed for $\bar{\lambda}$ $\tilde{x}_r = x$ where $x \in S_r$ and $d(\tilde{x}_r, a_r) = \bar{\lambda}$.

Other facilities are located at either $\overline{L_j(\bar{\lambda})}$ or $\overline{R_j(\bar{\lambda})}$ for $j \in J - \{r\}$.

where $\overline{L_j(\bar{\lambda})} \in S_j$ and $d(a_j, \overline{L_j(\bar{\lambda})}) = L_j(\bar{\lambda})$ and $\overline{R_j(\bar{\lambda})} \in S_j$ and $d(b_j, \overline{R_j(\bar{\lambda})}) = R_j(\bar{\lambda})$

By definition,

$$L_j(\bar{\lambda}) \leq l_j \text{ and } R_j(\bar{\lambda}) \leq l_j$$

Therefore, $\overline{L_j(\bar{\lambda})}$ and $\overline{R_j(\bar{\lambda})}$ are well defined points in S_j. So $x_j \in S_j$ for $j \in J$ and $(DC.1)$ constraints are satisfied.

For $(DC.2)$ constraints: Consider any facility j $(j \neq r)$

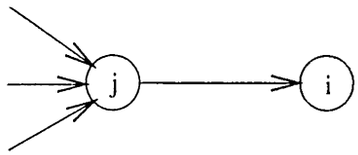
From the orientation phase we know that each node except r , has at most one out-neighbor. Let us call out-neighbor of j , facility i, if it exists.

From the construction phase we know that $\tilde{x}_j$ is located to $\overline{L_j(\bar{\lambda})}$ or $\overline{R_j(\bar{\lambda})}$

(i) arbitrarily (if j has no out-neighbor)

(ii) depending on the location of $\tilde{x}_i$

For (i), feasibility of $\tilde{x}_j$ depends only its in-neighbors. Since for those in-neighbors , except r, j is the unique out-neighbor. Therefore, feasibility of the distance constraint between j and its in-neighbors will be investigated when we consider these in-neighbors and their unique out-neighbor.



For (ii) $\tilde{x}_i$ is already located to $\overline{L_i(\bar{\lambda})}$ or $\overline{R_i(\bar{\lambda})}$ without considering the relationship between i and j.

Suppose $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ (where $L_i(\bar{\lambda}) = K_{Q_i(\bar{\lambda})}(\bar{\lambda})$)

There are four cases:

1. $j \in Q_i(\bar{\lambda})$
   (a) $L_i^j(\bar{\lambda}) = L_i^{jL}(\bar{\lambda})$ and $\tilde{x}_j$ is located to $\overline{L_j(\bar{\lambda})}$
   (b) $L_i^j(\bar{\lambda}) = L_i^{jR}(\bar{\lambda})$ and $\tilde{x}_j$ is located to $\overline{R_j(\bar{\lambda})}$
2. $j \notin Q_i(\bar{\lambda})$
   (a) $R_i^j(\bar{\lambda}) = R_i^{jL}(\bar{\lambda})$ and $\tilde{x}_j$ is located to $\overline{L_j(\bar{\lambda})}$
   (b) $R_i^j(\bar{\lambda}) = R_i^{jR}(\bar{\lambda})$ and $\tilde{x}_j$ is located to $\overline{R_j(\bar{\lambda})}$

Let us investigate each of the cases:

CASE 1-A $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ and $\tilde{x}_j = \overline{L_j(\bar{\lambda})}$, $j \in Q_i(\bar{\lambda})$ and $L_i^j(\bar{\lambda}) = L_i^{jL}(\bar{\lambda})$

Since $\bar{\lambda} \in F \Rightarrow L_i(\bar{\lambda}) \geq 0 \Rightarrow L_i(\bar{\lambda}) = Min_{k \in Q_i(\bar{\lambda})} L_i^k(\bar{\lambda})$

Since $j \in Q_i(\bar{\lambda}) \Rightarrow$

$$L_i(\bar{\lambda}) \leq L_i^j(\bar{\lambda}) = L_i^{jL}(\bar{\lambda}) \quad (1)$$

$$L_i^{jL}(\bar{\lambda}) = b_{ij} - d(\overline{L_j(\bar{\lambda})}, a_i) \quad (2)$$

$$L_i(\bar{\lambda}) = d(\overline{L_i(\bar{\lambda})}, a_i) \quad (3)$$

From (1), (2) and (3)

$$d(\overline{L_i(\bar{\lambda})}, \overline{L_j(\bar{\lambda})}) \leq d(\overline{L_i(\bar{\lambda})}, a_i) + d(\overline{L_j(\bar{\lambda})}, a_i) \leq b_{ij}$$

Therefore

$$d(\tilde{x}_i, \tilde{x}_j) \leq b_{ij}$$

CASE 1-B $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ and $\tilde{x}_j = \overline{R_j(\bar{\lambda})}$, $j \in Q_i(\bar{\lambda})$ and $L_i^j(\bar{\lambda}) = L_i^{jR}(\bar{\lambda})$

The reasoning is the same with CASE 1-A. But equations (1) and (2) should be changed with (4) and (5) respectively.

$$L_i(\bar{\lambda}) \leq L_i^j(\bar{\lambda}) = L_i^{jR}(\bar{\lambda}) \quad (4)$$

$$L_i^{jR}(\bar{\lambda}) = b_{ij} - d(\overline{R_j(\bar{\lambda})}, a_i) \quad (5)$$

From (3),(4) and (5)

$$d(\overline{L_i(\bar{\lambda})}, \overline{R_j(\bar{\lambda})}) \leq d(\overline{L_i(\bar{\lambda})}, a_i) + d(\overline{R_j(\bar{\lambda})}, a_i) \leq b_{ij}$$

Therefore,

$$d(\tilde{x}_i, \tilde{x}_j) \leq b_{ij}$$

CASE 2-A $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ and $\tilde{x}_j = \overline{L_j(\bar{\lambda})}$, $j \notin Q_i(\bar{\lambda})$ and $R_i^j(\bar{\lambda}) = R_i^{jL}(\bar{\lambda})$

Either $Q_i(\bar{\lambda}) = \emptyset$ or $Q_i(\bar{\lambda}) \neq \emptyset$

1 If $Q_i(\bar{\lambda}) = \emptyset$

Since $\bar{\lambda} \in F \Rightarrow L_i(\bar{\lambda}) \geq 0 \Rightarrow L_i(\bar{\lambda}) = l_i$ ($\tilde{x}_i = b_i$) and

$$Min_{k \in P} R_i^k(\bar{\lambda}) \geq 0 \Rightarrow R_i^j(\bar{\lambda}) \geq 0$$

$$R_i^j(\bar{\lambda}) = R_i^{jL}(\bar{\lambda}) \geq 0 \quad (6)$$

Then

$$b_{ij} - d(\overline{L_j(\bar{\lambda})}, b_i) \geq 0 \Rightarrow d(\overline{L_j(\bar{\lambda})}, b_i) \leq b_{ij} \quad (7)$$

Therefore

$$d(\tilde{x}_i, \tilde{x}_j) = d(\overline{L_j(\bar{\lambda})}, b_i) \leq b_{ij} \quad (8)$$

2 If $Q_i(\bar{\lambda}) \neq \emptyset$ then

Since $\bar{\lambda} \in F \Rightarrow L_i(\bar{\lambda}) \geq 0 \Rightarrow$

$$L_i(\bar{\lambda}) = Min_{k \in Q_i(\bar{\lambda})} L_i^k(\bar{\lambda}) \quad \geq \quad l_i - Min_{k \in P - Q_i(\bar{\lambda})} R_i^k(\bar{\lambda})$$

$$Min_{k \in P - Q_i(\bar{\lambda})} R_i^k(\bar{\lambda}) \quad \geq \quad l_i - L_i(\bar{\lambda})$$

$l_i - L_i(\bar{\lambda}) = d(\overline{L_i(\bar{\lambda})}, b_i)$ and since $j \in P - Q$

$$R_i^j(\bar{\lambda}) \quad \geq \quad d(\overline{L_i(\bar{\lambda})}, b_i)$$

$$R_i^j(\bar{\lambda}) = R_i^{jL}(\bar{\lambda}) = \quad b_{ij} - d(\overline{L_j(\bar{\lambda})}, b_i) \geq \quad d(\overline{L_i(\bar{\lambda})}, b_i) \tag{9}$$

$$b_{ij} \geq \quad d(\overline{L_j(\bar{\lambda})}, b_i) + d(\overline{L_i(\bar{\lambda})}, b_i) \geq \quad d(\tilde{x}_i, \tilde{x}_j) \tag{10}$$

CASE 2-B $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ and $\tilde{x}_j = \overline{R_j(\bar{\lambda})}$, $j \notin Q_i(\bar{\lambda})$ and $R_i^j(\bar{\lambda}) = R_i^{jR}(\bar{\lambda})$

The reasoning is the same with CASE 2-A. But equations (6), (7), (8), (9) and (10) should be changed with (11), (12), (13), (14) and (15) respectively.

$$R_i^j(\bar{\lambda}) = R_i^{jR}(\bar{\lambda}) \quad \geq \quad 0 \tag{11}$$

$$b_{ij} - d(\overline{R_j(\bar{\lambda})}, b_i) \quad \geq 0 \Rightarrow d(\overline{R_j(\bar{\lambda})}, b_i) \leq \quad b_{ij} \tag{12}$$

$$d(\tilde{x}_i, \tilde{x}_j) = d(\overline{R_j(\bar{\lambda})}, b_i) \leq b_{ij} \tag{13}$$

$$R_i^j(\bar{\lambda}) = R_i^{jR}(\bar{\lambda}) = \quad b_{ij} - d(\overline{R_j(\bar{\lambda})}, b_i) \quad \geq d(\overline{L_i(\bar{\lambda})}, b_i) \tag{14}$$

$$b_{ij} \geq \quad d(\overline{R_j(\bar{\lambda})}, b_i) + d(\overline{R_i(\bar{\lambda})}, b_i) \geq \quad d(\tilde{x}_i, \tilde{x}_j) \tag{15}$$

Consider now $j = r$, $\tilde{x}_r = x$ where ($x \in S_r : d(x, a_r) = \bar{\lambda}$)

For any i such that $(r, i) \in I$ $\tilde{x}_i$ is located at either $\overline{L_i(\bar{\lambda})}$ or $\overline{R_i(\bar{\lambda})}$. Without loss of generality, suppose $\tilde{x}_i = \overline{L_i(\bar{\lambda})}$ (i) $r \in Q_i(\bar{\lambda})$ or (ii) $r \notin Q_i(\bar{\lambda})$.

(i) Since $\bar{\lambda} \in F$ $L_i(\bar{\lambda}) \geq 0 \Rightarrow L_i(\bar{\lambda}) = Min_{k \in Q_i(\bar{\lambda})} L_i^k(\bar{\lambda})$

Since $r \in Q_i(\bar{\lambda})$

$$L_i(\bar{\lambda}) \leq L_i^r(\bar{\lambda}) = b_{ir} - d(x, a_i) \quad (16)$$

$$L_i(\bar{\lambda}) = d(\overline{L_i(\bar{\lambda})}, a_i) \quad (17)$$

From (16) and (17)

$$d(\tilde{x}_r, \tilde{x}_i) \leq d(\overline{L_i(\bar{\lambda})}, a_i) + d(x, a_i) \leq b_{ir}$$

The other case is similar to this. □

Corollary 1 *If P1 has no feasible solution, then the Algorithm terminates infeasible.*

Proof The proof is by contradiction. Let us assume that P1 has no feasible but the algorithm terminates feasible, that is with a nonempty F. Then from theorem 1, we can construct a feasible solution which contradicts with the fact that P1 is infeasible. □

Theorem 2 *If P1 has a feasible solution , then the Algorithm terminates feasible.*

Proof Let $\bar{X} = (\bar{x}_1, \bar{x}_2, ...\bar{x}_m)$ be a feasible solution to P1, that is,

$$d(\bar{x}_j, \bar{x}_k) \leq b_{j,k} \text{ for } (j, k) \in I$$

$$\bar{x}_j \in S_j \text{ for } j = 1, 2..m$$

We need to prove that there exists a $\bar{\lambda}$ such that $0 \leq \bar{\lambda} \leq l_r$ and $L_j(\lambda) \geq 0$ and $R_j(\lambda) \geq 0$ for $j \in J - \{r\}$. Then such a $\bar{\lambda}$ is in F, therefore $F \neq \emptyset$

The proof is by induction on the order of process. But for the sake of simplicity, we rename the nodes according to the order of process, that is if $i = A[j]$ then $i \Leftarrow j$

1 $k = 1$,

From feasibility of $\bar{X}$, we have $\bar{x}_1 \in S_1$ Let $\bar{\lambda} = d(\bar{x}_1, a_1) \Rightarrow 0 \leq \bar{\lambda} \leq l_1$

2. $k = 2$

$$\Gamma^{-1}(2) = \{1\}(\text{ from the orientation phase })$$

$$\mathrm{K}_{\emptyset}(\bar{\lambda}) = \begin{cases} l_2 & \text{if } R_2^1(\bar{\lambda}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$\mathrm{K}_{\{1\}}(\bar{\lambda}) = L_2^1(\bar{\lambda})$

$$L_2(\bar{\lambda}) = Max\{K_{\emptyset}(\bar{\lambda}), K_{\{1\}}(\bar{\lambda})\}$$

From the feasibility of $\bar{X}$ we know that $d(\bar{x}_2, \bar{x}_1) \leq b_{12}$ then from Lemma 1 either a_2 or b_2 is included in the shortest path between $\bar{x}_2$ and $\bar{x}_1$.

If a_2 is in the shortest path then,

$$d(\bar{x}_2, \bar{x}_1) = d(\bar{x}_2, a_2) + d(a_2, \bar{x}_1) \leq b_{12}$$
$$0 \leq d(\bar{x}_2, a_2) \leq b_{12} - d(a_2, \bar{x}_1) = L_2^1(\bar{\lambda})$$

If b_2 is in the shortest path then,

$$d(\bar{x}_2, \bar{x}_1) = d(\bar{x}_2, b_2) + d(b_2, \bar{x}_1) \leq b_{12}$$
$$0 \leq d(\bar{x}_2, b_2) \leq b_{12} - d(b_2, \bar{x}_1) = R_2^1(\bar{\lambda})$$

So either $L_2^1(\bar{\lambda}) \geq 0$ or $R_2^1(\bar{\lambda}) \geq 0$

Without loss of generality, let $L_2^1(\bar{\lambda}) \geq 0$ then $L_2(\bar{\lambda}) \geq K_{\{1\}}(\bar{\lambda}) = L_2^1(\bar{\lambda}) \geq 0$

Assume we have proved that $L_i(\bar{\lambda}) \geq 0$ for $i \leq k-1$. Now, given $L_i(\bar{\lambda}) \geq 0$ for $i \leq k-1$, we will prove that $L_k(\bar{\lambda}) \geq 0$

$$\Gamma^{-1}(k) \subseteq \bigcup_{i=1}^{k-1}\{i\}$$

(This is by construction. All in-neighbors of k are processed before k)

From feasibility of $\bar{X}$ we can write

$$d(\bar{x}_j, \bar{x}_k) \leq b_{jk} \text{ for } j \in \Gamma^{-1}(k)$$

From Lemma 1, the shortest path between $\bar{x}_j$ and $\bar{x}_k$ includes either a_k or b_k. Then

$$d(\bar{x}_j, \bar{x}_k) = d(\bar{x}_j, a_k) + d(a_k, \bar{x}_k) \leq b_{jk} \text{ or}$$
$$d(\bar{x}_j, \bar{x}_k) = d(\bar{x}_j, b_k) + d(b_k, \bar{x}_k) \leq b_{jk} \text{ for } j \in \Gamma^{-1}(k)$$

then for $j \in \Gamma^{-1}(k)$

$$\begin{aligned} 0 \leq d(a_k, \bar{x}_k) &\leq b_{jk} - d(\bar{x}_j, a_k) \leq L_k^j(\bar{\lambda}) \text{ or} \\ 0 \leq d(b_k, \bar{x}_k) &\leq b_{jk} - d(\bar{x}_j, b_k) \leq R_k^j(\bar{\lambda}) \end{aligned}$$

($[a_j, L_j(\bar{\lambda})]$ includes $\bar{x}_j$ but the extreme point might give a shorter distance, we used inequality instead of equality in the second place for both of the lines.)

Then for every in-neighbor j of k $L_k^j(\bar{\lambda}) \geq 0$ or $R_k^j(\bar{\lambda}) \geq 0$ or both. Moreover they are large enough to cover point $\bar{x}_k$. That is for $j \in \Gamma^{-1}(k)$

$$\begin{aligned} L_k^j(\bar{\lambda}) &\geq d(a_k, \bar{x}_k) or \\ R_k^j(\bar{\lambda}) &\geq d(b_k, \bar{x}_k) \end{aligned}$$

Let us define a subset of $\bar{Q}$ of $\Gamma^{-1}(k)$ as follows

$$\bar{Q} = \{j \in \Gamma^{-1}(k) : L_k^j(\bar{\lambda}) \geq d(a_k, \bar{x_k})\} \text{ then}$$

$$K_{\bar{Q}}(\bar{\lambda}) = \begin{cases} Min_{j \in \bar{Q}} L_k^j(\bar{\lambda}) & \text{if } Min_{j \in \bar{Q}} L_k^j(\lambda) \geq l_k - Min_{j \in P - \bar{Q}} R_k^j(\bar{\lambda}) \\ -1 & \text{otherwise} \end{cases}$$

Let us check whether $Min_{j \in \bar{Q}} L_k^j(\lambda) \geq l_k - Min_{j \in P - \bar{Q}} R_k^j(\bar{\lambda})$ or not.

$$Min_{j \in \bar{Q}} L_k^j(\lambda) + Min_{j \in P - \bar{Q}} R_k^j(\bar{\lambda}) \geq d(\bar{a_k}, \bar{x_k}) + d(\bar{b_k}, \bar{x_k}) = l_k$$

Since the condition holds $K_{\bar{Q}}(\bar{\lambda}) = Min_{j \in Q} L_k^j(\bar{\lambda}) \geq 0$

Since $L_k(\bar{\lambda}) = Max_{Q \subseteq P}\{K_Q(\bar{\lambda})\} \Rightarrow L_k(\bar{\lambda}) \geq K_{\bar{Q}}(\bar{\lambda}) \geq 0$

Therefore $\bar{\lambda} \in F \neq \emptyset$. □

Corollary 2 *If Algorithm terminates infeasible, P1 has no feasible solution.*

Proof The proof is by contradiction. Suppose the algorithm terminates infeasible and there exists a location vector which is feasible to P1.From Theorem 2, we know that if there exists a feasible solution to P1, the algorithm terminates feasible, which is a contradiction. □

2.3.4 Shapes of $L_i^j(\lambda)$ and $R_i^j(\lambda)$ Graphs

In this section, we will investigate the shapes of $L_i^j(\lambda)$ and $R_i^j(\lambda)$. They are important because we will use the information about their linearity in the complexity discussion.

$$\mathrm{L}_i^j(\lambda) = \begin{cases} Min\{(b_{ir} - d(x, a_i)), l_i\} & \text{if } j = r \\ Max\{L_i^{jL}(\lambda), L_i^{jR}(\lambda)\} & \text{if } j \neq r \end{cases}$$

For $j = r$, from Lemma 1;

$$L_i^r(\lambda) = Min\{(b_{ir} - Min\{\lambda + d(a_r, a_i), l_r - \lambda + d(b_r, a_i)\}), l_i\}$$

All numbers other than λ can be calculated a priori, so they can be considered as constants.

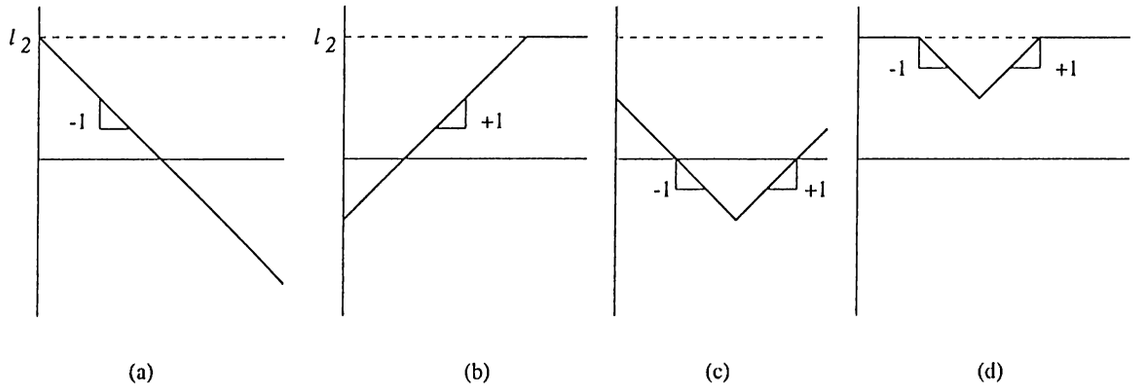


Figure 2.11: Possible Shapes of $L_i^r(\lambda)$

The first part is the maximum of two functions that are linear in λ. So it is a piecewise linear convex function with at most two pieces, then $L_i^r(\lambda)$

is minimum of two convex functions, therefore it can be convex, concave or neither convex nor concave. These possibilities are shown in Figure 2.11.

Other observation on $L_i^r(\lambda)$ concerns its slope. The slope is either +1, -1 or 0.

Consider the second facility in the order $i \Leftarrow A[2]$. Its unique in-neighbor is r. So $L_i(\lambda)$ is determined as follows:

$$\mathrm{K}_{\emptyset}(\lambda) = \begin{cases} l_i & \text{if } R_i^r(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{r\}}(\lambda) = L_i^r(\lambda)$$

$$L_i(\lambda) = Max\{K_{\emptyset}(\lambda), K_{\{r\}}(\lambda)\}$$

$K_{\emptyset}(\lambda)$ is a discontinuous function with at most two jump points. So $L_i(\lambda)$ is a discontinuous function with slopes +1,-1 and 0.

For $j \neq r$

$L_i^{jL}(\lambda) = Min\{(b_{ij} - Min\{L_j(\lambda) + d(a_i, a_j), l_j - L_j(\lambda) + d(b_j, a_i)\}), l_i\}$ and

$L_i^{jR}(\lambda) = Min\{(b_{ij} - Min\{R_j(\lambda) + d(a_i, b_j), l_j - R_j(\lambda) + d(a_j, a_i)\}), l_i\}$

So $L_i^{jL}(\lambda)$ ($L_i^{jR}(\lambda)$) is the minimum of $C_1 - L_j(\lambda)$ and $C_2 + L_j(\lambda)$($C_1 - R_j(\lambda)$ and $C_2 + R_j(\lambda)$) where C_1 and C_2 are constants. So $L_i^{jL}(\lambda)$ ($L_i^{jR}(\lambda)$) is linear, discontinuous function with slopes +1,-1 or 0 and so is $R_i(\lambda)$. It can be shown by induction that each $L_i(\lambda)$ and $R_i(\lambda)$ is linear, discontinuous functions with slopes +1,-1 and 0.

2.3.5 Complexity

Let us investigate the complexity of each phase of the algorithm.

1 **Orientation Phase** In this phase we just calculate the degrees of the nodes in LN and make comparisons. The order of this step is $O(m)$.

2 Distance Calculation Phase

1 We used Dijkstra for at most n times. Therefore, complexity of this step is $O(n^3)$.

2 For each distance of interest we compare 4 functions. Then 16 comparisons in total are made for pair $(j,k) \in I$. We know that comparison of two linear function takes constant time, Dyer [4]. Because of our Broken Wheel assumption, there can be at most $2m-3$ pairs. Therefore, the order of this step is $O(m)$.

3 Reduction Phase

For each new facility

1 In-degree of any node in DLN is restricted to 2, due to Broken Wheel assumption. For each in-neighbor, 3 comparisons are made for each of $L_i^{jL}(\lambda)$, $L_i^{jR}(\lambda)$, $R_i^{jL}(\lambda)$ and $R_i^{jL}(\lambda)$ summing up to 12 comparisons. These functions are linear discontinuous functions but number of jumps are restricted with a fixed number. So comparing these functions still takes constant time. Then one comparison of two functions is made to determine each of $L_i^j(\lambda)$, $R_i^j(\lambda)$.

2 $L_i(\lambda)$ and $R_i(\lambda)$ is calculated by comparing 2^k $K_Q(\lambda)$ or $M_Q(\lambda)$ values. These values can be calculated in constant time, since each is calculated via comparing two linear functions. k with our assumption can be at most 2. And choosing the maximum of 4 linear functions is also performed in constant time.

Therefore, the order of the reduction phase is $O(m)$.

4 Construction Phase

In this step, for each new facility, we find the extreme point of $S_j(\bar{\lambda})$ that reaches located x_i where i is the unique out-neighbor of j. This is done by comparing two numbers ($L_i^{jL}(\bar{\lambda})$ and $L_i^{jR}(\bar{\lambda})$). The numbers were calculated in previous step. So the order of this phase is $O(m)$.

2.3.6 Example

Here we will give a small example on which we show the application of the algorithm.

Data for the Example

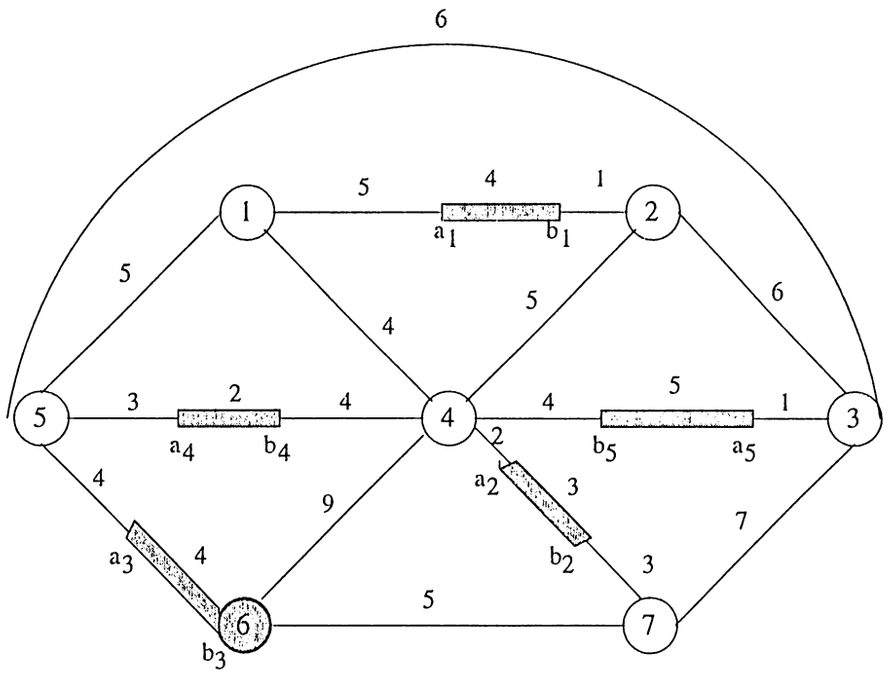


Figure 2.12: Transport Network

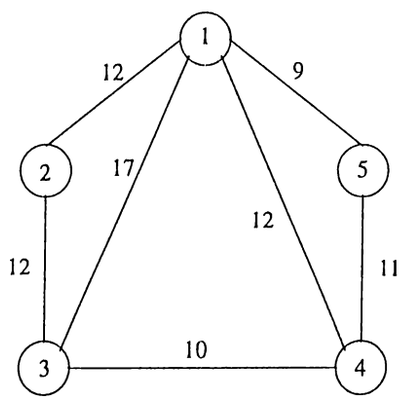


Figure 2.13: b_{jk} information

1. ORIENTATION

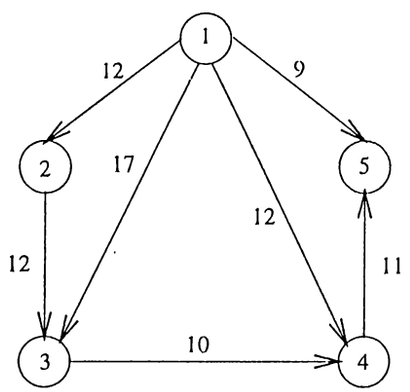


Figure 2.14: *DLN*

2. DISTANCE CALCULATION

$$D = \begin{array}{c} \\ a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \\ b_4 \\ a_5 \\ b_5 \end{array} \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & a_3 & b_3 & a_4 & b_4 & a_5 & b_5 \\ 0 & 4 & 11 & 14 & 14 & 18 & 13 & 13 & 12 & 13 \\ & 0 & 8 & 11 & 17 & 15 & 12 & 10 & 8 & 10 \\ & & 0 & 3 & 15 & 11 & 8 & 6 & 12 & 7 \\ & & & 0 & 12 & 8 & 11 & 9 & 11 & 10 \\ & & & & 0 & 4 & 7 & 9 & 11 & 16 \\ & & & & & 0 & 11 & 13 & 13 & 13 \\ & & & & & & 0 & 2 & 10 & 10 \\ & & & & & & & 0 & 12 & 8 \\ & & & & & & & & 0 & 5 \\ & & & & & & & & & 0 \end{bmatrix}$$

3 REDUCTION

0. $F = [0, 4], k = 2$

1. $i \Leftarrow A[2] = 2$

$$\Gamma^{-1}(2) = \{1\}$$

$$L_2^1(\lambda) = Min\{12 - Min\{\lambda + 11, 12 - \lambda\}, 3\}$$
$$R_2^1(\lambda) = Min\{12 - Min\{\lambda + 14, 14 - \lambda\}, 3\}$$

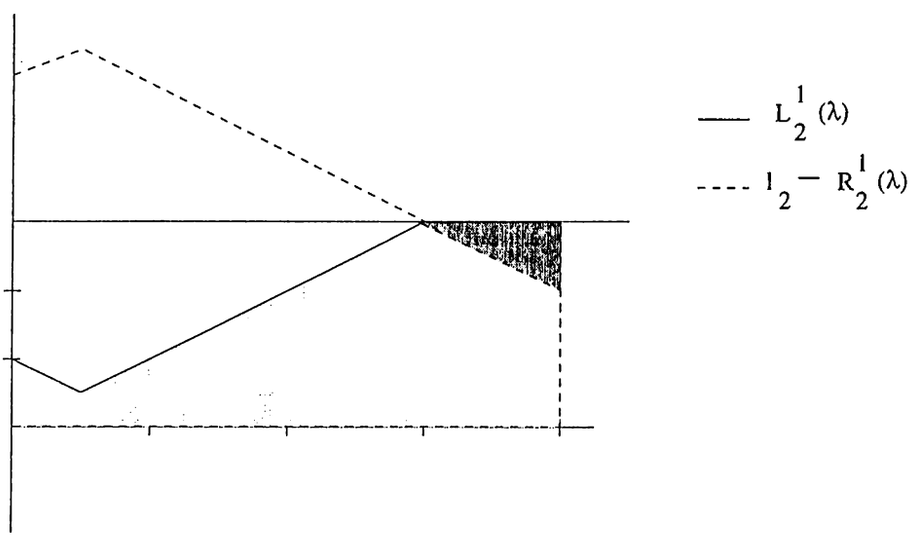


Figure 2.15: $S_2(\lambda)$

2.

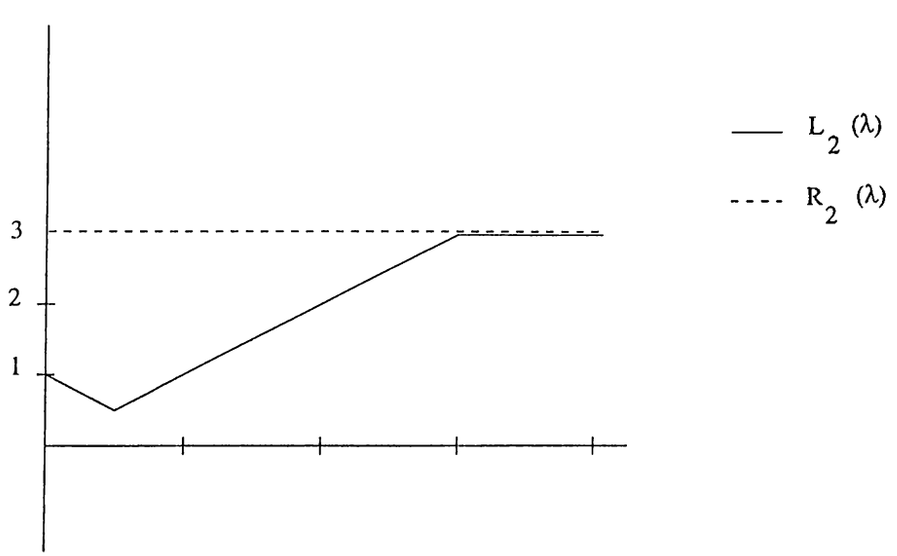


Figure 2.16: $L_2(\lambda)$ and $R_2(\lambda)$

3. $\Lambda = \emptyset$

4. $F = [0, 4]$

5. (c) $F \neq \emptyset$, $k \neq 5$ $k \Leftarrow k + 1 = 3$

1. $i \Leftarrow A[3] = 3$

$$\Gamma^{-1}(3) = \{1, 2\}$$

$$L_3^{2L}(\lambda) = Min\{12 - Min\{15 + L_2(\lambda), 15 - L_2(\lambda)\}, 4\}$$
$$L_3^{2R}(\lambda) = Min\{12 - Min\{12 + R_2(\lambda), 18 - R_2(\lambda)\}, 4\}$$
$$R_3^{2L}(\lambda) = Min\{12 - Min\{11 + L_2(\lambda), 11 - L_2(\lambda)\}, 4\}$$
$$R_3^{2R}(\lambda) = Min\{12 - Min\{8 + R_2(\lambda), 14 - R_2(\lambda)\}, 4\}$$

$$L_3^2(\lambda) = Max\{L_3^{2L}(\lambda), L_3^{2R}(\lambda)\}$$
$$R_3^2(\lambda) = Max\{R_3^{2L}(\lambda), R_3^{2R}(\lambda)\}$$

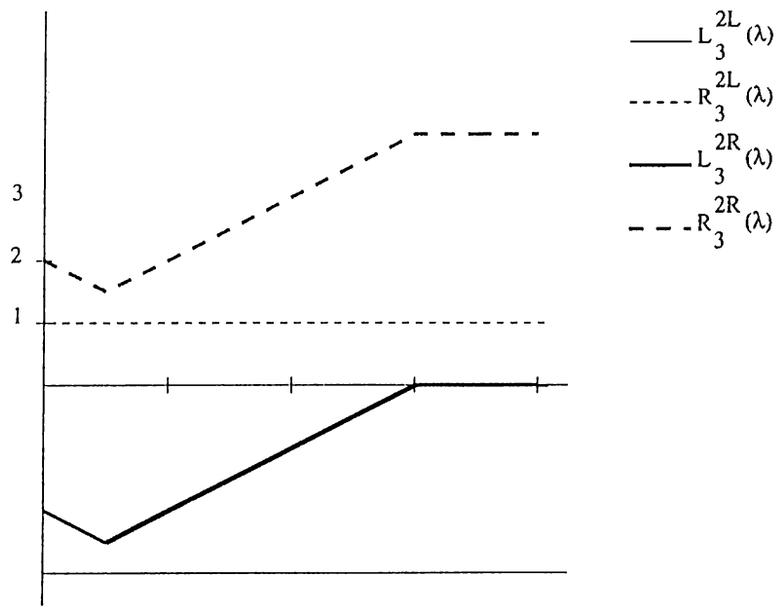


Figure 2.17: $L_3^2(\lambda)$ and $R_3^2(\lambda)$ determination

$$L_3^1(\lambda) = Min\{17 - Min\{\lambda + 14, 21 - \lambda\}, 3\}$$
$$R_3^1(\lambda) = Min\{12 - Min\{\lambda + 18, 19 - \lambda\}, 3\}$$

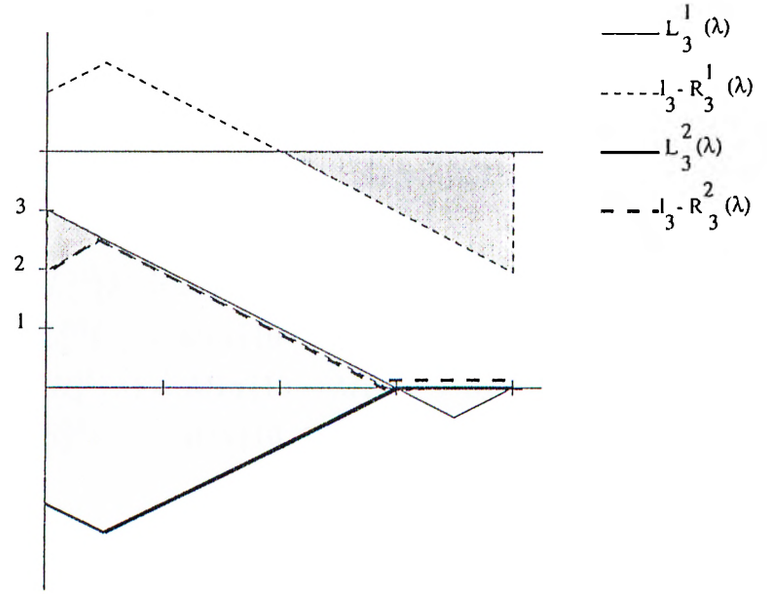


Figure 2.18: $S_3(\lambda)$

2.

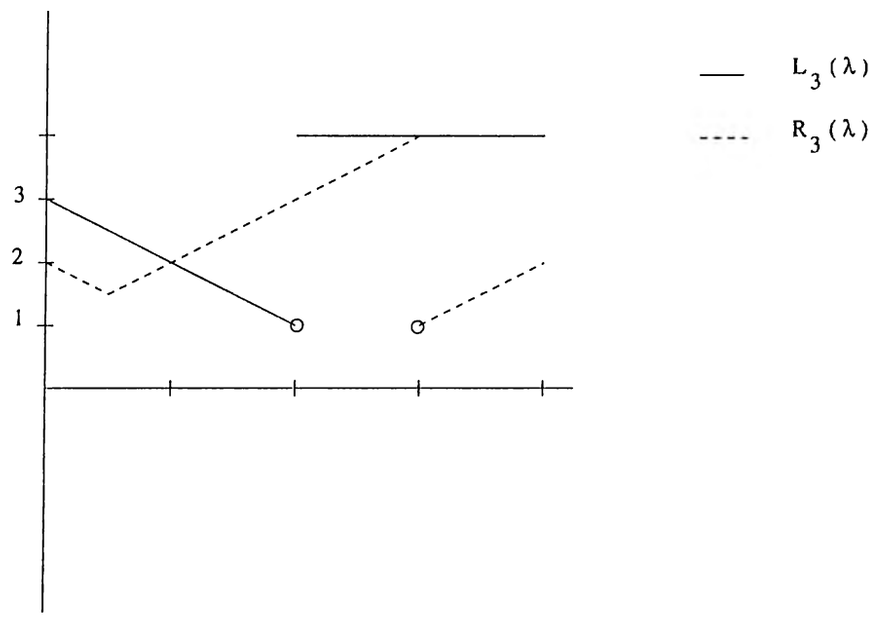


Figure 2.19: $L_3(\lambda)$ and $R_3(\lambda)$

3. $\Lambda = \emptyset$

4. $F = [0, 4]$

5. (c) $F \neq \emptyset$, $k \neq 5$ $k \Leftarrow k + 1 = 4$

1. $i \Leftarrow \Lambda[4] = 4$

$$\Gamma^{-1}(4) = \{1, 3\}$$

$$L_4^{3L}(\lambda) = Min\{10 - Min\{7 + L_3(\lambda), 15 - L_3(\lambda)\}, 2\}$$
$$L_4^{3R}(\lambda) = Min\{10 - Min\{11 + R_3(\lambda), 11 - R_3(\lambda)\}, 2\}$$
$$R_4^{3L}(\lambda) = Min\{10 - Min\{9 + L_3(\lambda), 17 - L_3(\lambda)\}, 2\}$$
$$R_3^{2R}(\lambda) = Min\{10 - Min\{13 + R_3(\lambda), 13 - R_3(\lambda)\}, 2\}$$

$$L_4^3(\lambda) = Max\{L_4^{3L}(\lambda), L_4^{3R}(\lambda)\}$$
$$R_4^3(\lambda) = Max\{R_4^{3L}(\lambda), R_4^{3R}(\lambda)\}$$

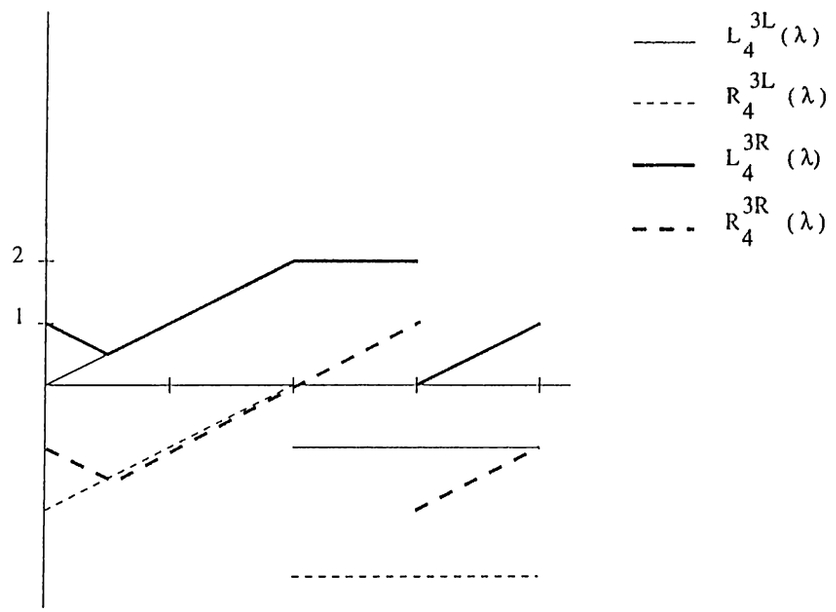


Figure 2.20: $L_4^3(\lambda)$ and $R_4^3(\lambda)$ determination

$$L_4^1(\lambda) = Min\{12 - Min\{\lambda + 13, 16 - \lambda\}, 2\}$$
$$R_4^1(\lambda) = Min\{12 - Min\{\lambda + 13, 14 - \lambda\}, 2\}$$

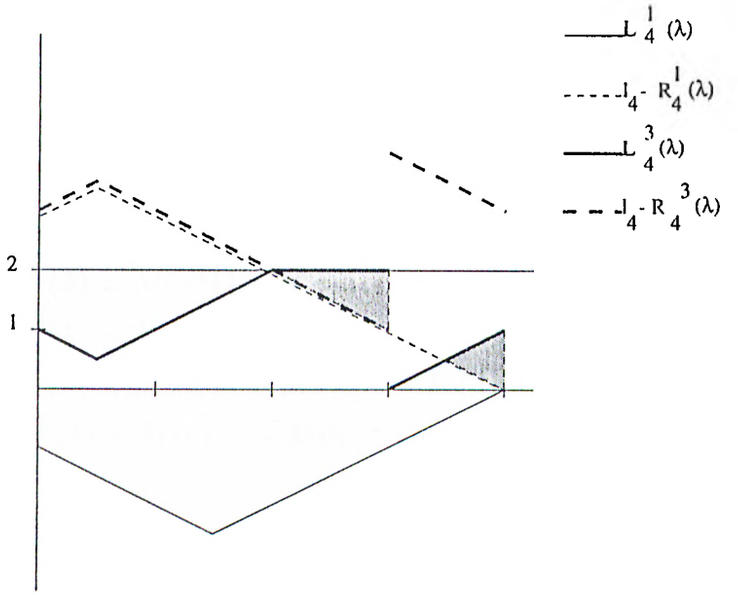


Figure 2.21: $S_4(\lambda)$

2.

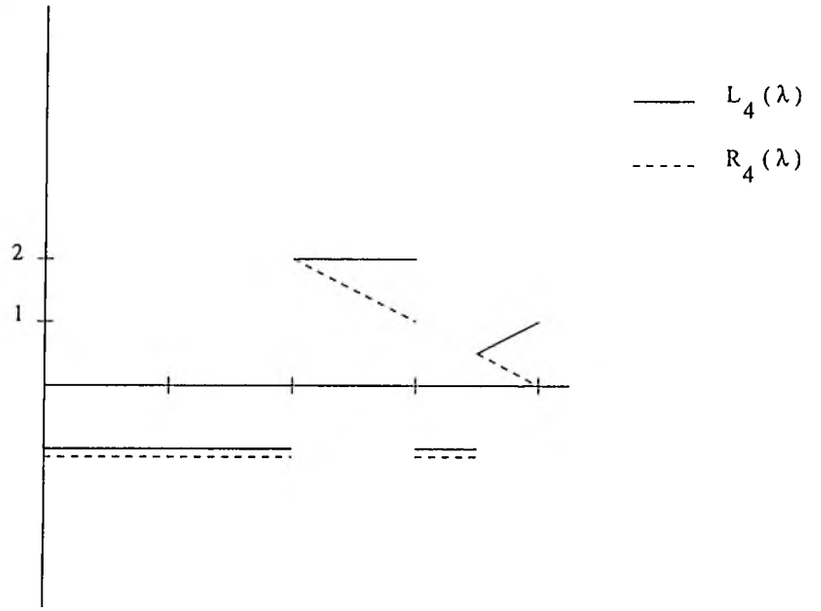


Figure 2.22: $L_4(\lambda)$ and $R_4(\lambda)$

3. $\Lambda = [0,2) \cup (3,3.5)$

4. $F = [2,3] \cup [3.5,4]$

5. (c) $F \neq \emptyset$, $k \neq 5$ $k \Leftarrow k+1 = 5$

1. $i \Leftarrow A[5] = 5$

$$\Gamma^{-1}(5) = \{1,4\}$$

$$L_5^{4L}(\lambda) = Min\{11 - Min\{10 + L_4(\lambda), 14 - L_4(\lambda)\}, 5\}$$
$$L_5^{4R}(\lambda) = Min\{11 - Min\{12 + R_4(\lambda), 12 - R_4(\lambda)\}, 5\}$$
$$R_5^{4L}(\lambda) = Min\{11 - Min\{10 + L_4(\lambda), 10 - L_4(\lambda)\}, 5\}$$
$$R_5^{4R}(\lambda) = Min\{11 - Min\{8 + R_4(\lambda), 12 - R_4(\lambda)\}, 5\}$$

$$L_5^4(\lambda) = Max\{L_5^{4L}(\lambda), L_5^{4R}(\lambda)\}$$
$$R_5^4(\lambda) = Max\{R_5^{4L}(\lambda), R_5^{4R}(\lambda)\}$$

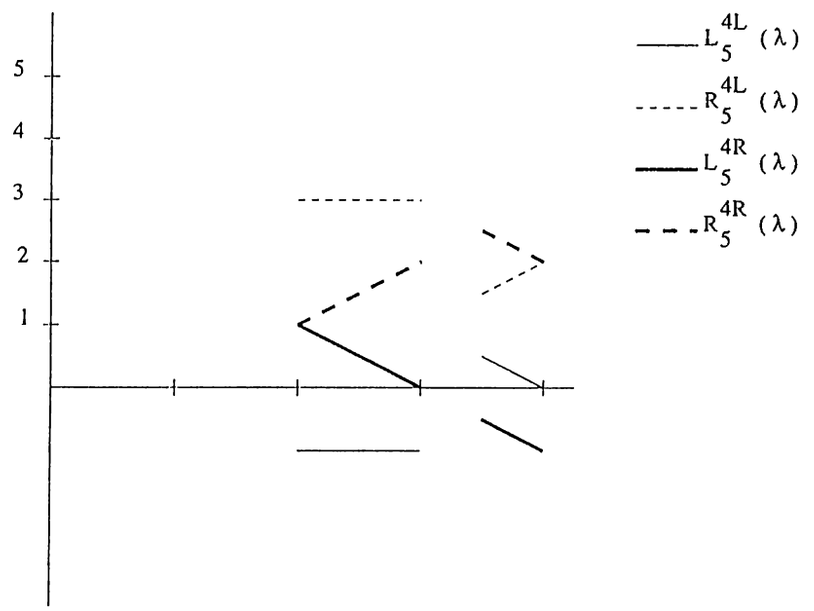


Figure 2.23: $L_5^4(\lambda)$ and $R_5^4(\lambda)$ determination

$$L_5^1(\lambda) = Min\{9 - Min\{\lambda + 12, 12 - \lambda\}, 5\}$$
$$R_5^1(\lambda) = Min\{9 - Min\{\lambda + 13, 14 - \lambda\}, 5\}$$

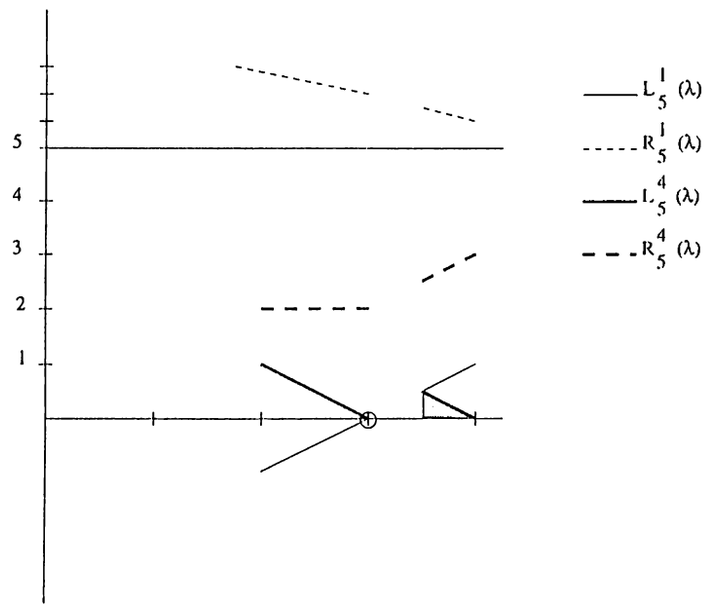


Figure 2.24: $S_5(\lambda)$

2.

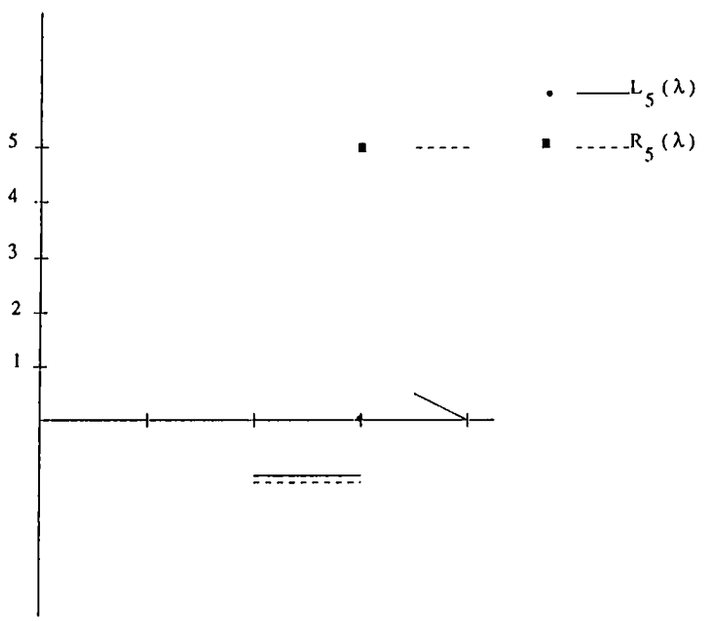


Figure 2.25: $L_5(\lambda)$ and $R_5(\lambda)$

3. $\Lambda = [2, 3)$

4. $F = \{3\} \cup [3.5, 4]$

5. (c) $F \neq \emptyset$, $k = 5$ STOP FEASIBLE.

4.CONSTRUCTION

0 $Located = P = \emptyset$, $S = \{5\}$

Choose $3 \in F$

Locate $x_1 = x$, $x \in S_1$ and $d(x, a_1) = 3$

$Located = \{1\}$

1 Choose 5 from S, $P = \{5\}$

Locate $x_5 = \overline{L_5(3)}$, $x_5 \in S_5$ and $d(x_5, a_5) = L_5(3) = 0$

$Located = \{1, 5\}$

1.1 $P \neq \emptyset$ Choose $5 \in P$

$W_5 = \Gamma^{-1}(5) - Located$

$W_5 = \{4\}$

Determine $K_{Q_5(3)}(3)$ that determine $L_5(3)$

$K_{Q_5(3)}(3) = Min\{L_5^1(3), L_5^4(3)\}$

$Q_5(3) = \{1, 4\}$ then $4 \in Q_5(3)$

1.1.1 Choose $4 \in W_5$

1.1.1.1 $4 \in Q_5(3)$ and $L_5^4(3) = L_5^{4R}(3)$ then

Locate $x_4 = \overline{R_4(3)}$,$x_4 \in S_4$ and $d(x_4, b_4) = R_4(3) = 1$

$Located = \{1, 4, 5\}$

$W_5 \Leftarrow \emptyset$

$P \Leftarrow \{4\}$

1.1 $P \neq \emptyset$ Choose $4 \in P$

$W_4 = \Gamma^{-1}(4) - Located$

$W_4 = \{3\}$

Determine $M_{Q_4(3)}(3)$ that determine $R_4(3)$

$Q_4(3) = \{3\}$

1.1.1 Choose $3 \in W_4$

1.1.1.1 $3 \in Q_4(3)$ and $L_4^3(3) = L_4^{3R}(3)$ then

Locate $x_3 = \overline{R_3(3)}$, $x_3 \in S_3$ and $d(x_3, b_3) = R_3(3) = 4$

$Located = \{1, 3, 4, 5\}$

$W_4 \Leftarrow \emptyset$

$P \Leftarrow \{3\}$

1.1 $P \neq \emptyset$ Choose $3 \in P$

$W_3 = \Gamma^{-1}(3) - Located$

$W_3 = \{2\}$

Determine $M_{Q_3(3)}(3)$ that determine $R_3(3)$

$Q_3(3) = \{2\}$

1.1.1 Choose $2 \in W_3$

1.1.1.1 $2 \in Q_3(3)$ and $L_3^3(3) = L_3^{2L}(3)$ then

Locate $x_2 = \overline{L_2(3)}$, $x_2 \in S_2$ and $d(x_2, a_2) = L_2(3) = 3$

$Located = \{1, 2, 3, 4, 5\}$

$W_3 \Leftarrow \emptyset$

$P \Leftarrow \{2\}$

1.1 $P \neq \emptyset$ Choose $2 \in P$

$W_2 = \Gamma^{-1}(2) - Located$

$W_2 = \emptyset$

$P \Leftarrow \emptyset$

1.2 $P = \emptyset$

$S \Leftarrow S - \{5\} = \emptyset$

2. $S = \emptyset$ STOP. All facilities are located.

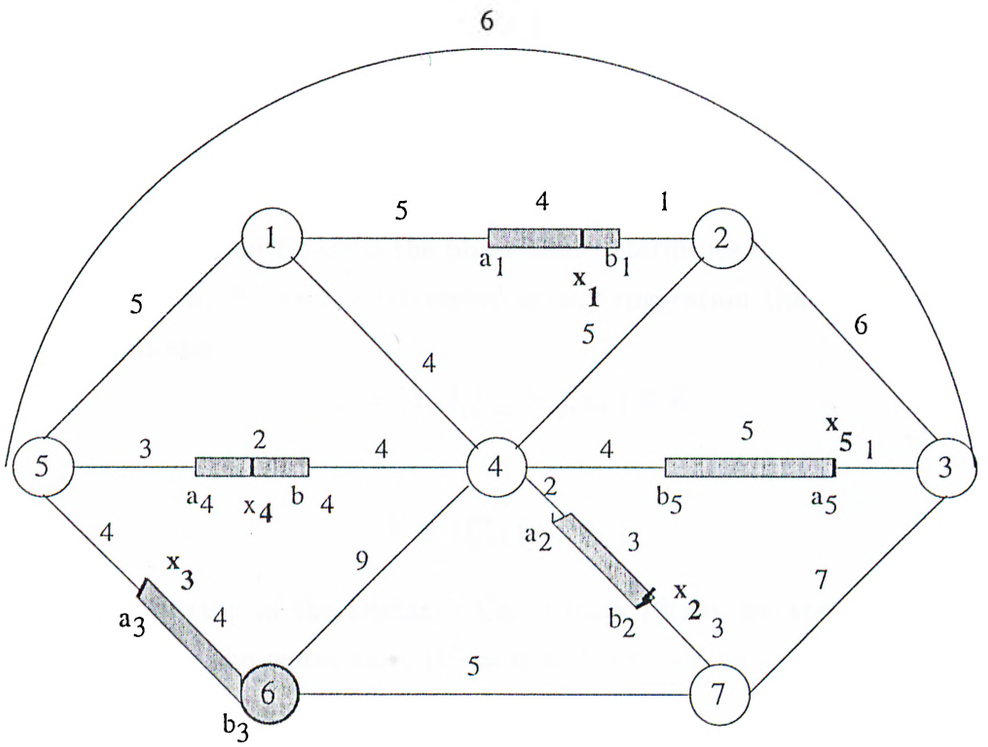


Figure 2.26: Solution

$$d(x_1, x_2) = 12 \leq 12$$
$$d(x_1, x_3) = 17 \leq 17$$
$$d(x_1, x_4) = 12 \leq 12$$
$$d(x_1, x_5) = 9 \leq 9$$
$$d(x_2, x_3) = 12 \leq 12$$
$$d(x_3, x_4) = 8 \leq 10$$
$$d(x_4, x_5) = 11 \leq 11$$

The solution found is feasible.

2.4 IMPROVING EFFICIENCY

2.4.1 Economy in Distance Calculation Phase

Distance Calculation Phase is the phase that determines the complexity of the whole algorithm. So we are interested in any suggestion that will reduce the order of that step.

$$S_j = [a_j, b_j] \subseteq [v_{p_j}, v_{q_j}] \in E$$

and

$$\overline{V} = \bigcup_{j=1}^{m} \{v_{p_j}, v_{q_j}\}$$

In the first step of the Distance Calculation Phase, we apply Dijkstra for every $v \in \overline{V}$. In the worst case, $|\overline{V}| = n$ and we need to apply Dijkstra for n times.

Suppose after applying Dijkstra for v_1, we obtain the following shortest path tree rooted at v_1.

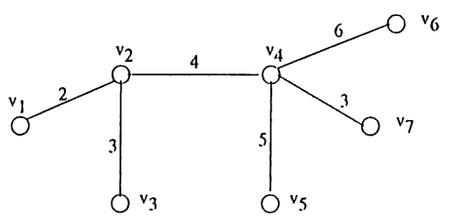


Figure 2.27: Shortest path tree rooted at v_1

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	2	5	6	11	12	9
v_2		0	3	4	9	7	10
v_3			0				
v_4				0	5	6	3
v_5					0		
v_6						0	
v_7							0

The first row of the matrix is what we expect to gain from applying Dijkstra's Algorithm for v_1, but if the shortest path from v_1 to v_i includes v_j then the shortest path between v_i and v_j is also determined. See the matrix above. The underlined entries are the by product of first application of Dijkstra. So we may not need to repeat Dijkstra for k times.

Applying Dijkstra for $v \in \overline{V}$ and keeping only shortest path distances between v and $v_j \in \overline{V}$ does not use all the information that is produced. That's why some of the distances need to be calculated more than once. Of course, keeping the nodes that appear in the shortest paths does not improve the worst case performance (in the worst case the graph is a complete graph and satisfies triangular inequality, then no shortest path include a node other than the ends), but it improves the average performance.

2.4.2 Preprocessing for b_{jk}s

After Distance Calculation Phase, we can identify some of the infeasibility of the problem or redundancy of some of the constraints before going any further by using calculated distances between extreme points of S_j and S_k and b_{jk}.

1. If $b_{jk} = 0$ then

   (a) If $S_j \cap S_k = \emptyset$ then the problem is infeasible.

   (b) If $S_j \cap S_k \neq \emptyset$ then the size of the problem can be reduced by one.

   Let $S_p = S_j \cap S_k$ and $b_{ip} = Min\{b_{ij}, b_{ik}\}$ (when $(i,j) \notin I$ then $b_{ij}\infty$) and $I_p = \{i : b_{ip} < \infty\}$.

   We can delete nodes j and k and all the edges attached to these nodes. Add node p instead and add edges $\{[i,p] : i \in I_p\}$ with lengths b_{ip}. The problem data can be preprocessed in this way to eliminate all b_{jk}'s that are zero. The preprocessing either concludes the problem is infeasible or reduces the data to positive bounds between new facilities.

2. If $b_{jk} \geq maxd_{jk}$ where $maxd_{jk}$ is defined as follows:

   (a) If $S_j, S_k \subseteq [v_p, v_q] \in E$, $S_j \cap S_k \neq \emptyset$ and length of $[v_p, v_q] = L$

   $$maxd_{jk} = [Max\{\omega_{pq}(b_j), \omega_{pq}(b_k)\} - Min\{\omega_{pq}(a_j), \omega_{pq}(a_k)\}].L$$

   (b) If $S_j \cap S_k \neq \emptyset$

   $$maxd_{jk} = l_j + l_k + Min\{d(a_j, a_k), d(a_j, b_k), d(b_j, a_k), d(b_j, b_k)\}$$

   Then we can delete edge $[j, k]$ from LN. Since for any choice of x_j and x_k:

   $$d(x_j, x_k) \leq maxd_{jk} \leq b_{jk}$$

   Because this constraint is redundant, $x_j \in S_j$ and $x_k \in S_k$ forces $d(x_j, x_k) \leq b_{jk}$ to be satisfied automatically.

3. For (j, k) pairs, such that $S_j \cap S_k \neq \emptyset$, If

   $$b_{jk} < Min\{d(a_j, a_k), d(a_j, b_k), d(b_j, a_k), d(b_j, b_k)\}$$

   then the problem is infeasible. Since, there does not exist a pair, x_j and x_k, that satisfies $d(x_j, x_k) \leq b_{jk}$.

2.4.3 Efficiency in Calculation of $L_i(\lambda)$ and $R_i(\lambda)$

In order to calculate $L_i(\lambda)$ and $R_i(\lambda)$ we need to compare 2^k $K_Q(\lambda)$ and $M_Q(\lambda)$ values. There might be some economy when these comparisons are performed in a special order.

Observation 5 *When $K_\emptyset(\lambda) \geq 0 (\equiv (K_\emptyset(\lambda) = l_i)$) then we do not need to calculate other $K_Q(\lambda)$ values, since*

$$L_i(\lambda) = K_\emptyset(\lambda)$$

Proof $K_Q(\lambda) \leq L_i^j(\lambda)$ for some $j \in P$. By definition $L_i^j(\lambda) \leq l_i \ \forall j \in P$. Therefore

$$K_\emptyset(\lambda) = l_i \geq L_i^j(\lambda) \geq K_Q(\lambda)$$

So $L_i(\lambda) = K_\emptyset(\lambda)$ □

Observation 6 *Suppose for a specific λ, say $\bar{\lambda}$, $L_i^k(\bar{\lambda}) < 0$ for some $k \in P$ Then*

$$K_Q(\bar{\lambda}) < 0 \text{ for any } Q \text{ such that } k \in Q$$

Proof

$$K_Q(\bar{\lambda}) = \begin{cases} Min_{j\in Q} L_i^j(\bar{\lambda}) & \text{if } Min_{j\in Q} L_i^j(\bar{\lambda}) \geq l_i - Min_{j\in P-Q} R_i^j(\bar{\lambda}) \\ -1 & \text{otherwise} \end{cases}$$

If $K_Q(\bar{\lambda}) = -1$ we are done. if $K_Q(\bar{\lambda}) = Min_{j\in Q} L_i^j(\bar{\lambda})$ then since $k \in Q$ $Min_{j\in Q} L_i^j(\bar{\lambda}) \leq L_i^k(\bar{\lambda}) < 0$ So $K_Q(\bar{\lambda}) < 0$ □

Observation 7 *Suppose for a specific λ, say $\bar{\lambda}$ $R_i^k(\bar{\lambda}) < 0$ for some $k \in P$ Then*

$$K_Q(\bar{\lambda}) < 0 \text{ for any } Q \text{ such that } k \in P - Q$$

Proof

$$K_Q(\bar{\lambda}) = \begin{cases} Min_{j\in Q} L_i^j(\bar{\lambda}) & \text{if } Min_{j\in Q} L_i^j(\bar{\lambda}) \geq l_i - Min_{j\in P-Q} R_i^j(\bar{\lambda}) \\ -1 & \text{otherwise} \end{cases}$$

$K_Q(\bar{\lambda}) = Min_{j\in Q} L_i^j(\bar{\lambda})$ if $Min_{j\in Q} L_i^j(\bar{\lambda}) \geq l_i - Min_{j\in P-Q} R_i^j(\bar{\lambda})$

$k \in P - Q$ and $R_i^k(\bar{\lambda}) < 0$ then

$$Min_{j\in P-Q} R_i^j(\bar{\lambda}) \leq R_i^k(\bar{\lambda}) < 0$$

Therefore, $Min_{j\in Q} L_i^j(\bar{\lambda}) \geq l_i - Min_{j\in P-Q} R_i^j(\bar{\lambda}) > l_i$ which is impossible. So $K_Q(\bar{\lambda}) = -1$ □

These observations can be used systematically to decrease the number of comparisons for some interval of λ. Because of the symmetry these kinds of reductions are also applicable to the calculation of $R_i^j(\lambda)$ Here is an example:

Let us define

$$I_{L_k} = \{\lambda : L_i^k(\lambda) < 0\} \text{ and } I_{R_k} = \{\lambda : R_i^k(\lambda) < 0\}$$

It is obvious that if $\lambda \in I_{L_k} \cap I_{R_k}$ then the problem is not feasible for this value of λ

Consider the following determination of $L_i(\lambda)$ and $R_i(\lambda)$ where $\Gamma^{-1}(i) = \{k_1, k_2, k_3\}$ with I_{L_k} and I_{R_k} information.

$$I_{L_{k_1}} = [0,4] \cup [5,6] \quad I_{L_{k_2}} = [0,2] \cup [4,5.5] \quad I_{L_{k_3}} = \emptyset$$
$$I_{R_{k_1}} = [0,1] \quad I_{R_{k_2}} = [1,3] \cup [5,6] \quad I_{R_{k_3}} = [4,6]$$

Delete $[0,1]$ (Since $L_i^{k_1}(\lambda) < 0$ and $R_i^{k_1}(\lambda) < 0$) and delete $[1,2] \cup [5,5.5]$ (Since $L_i^{k_2}(\lambda) < 0$ and $R_i^{k_2}(\lambda) < 0$)

For $\lambda \in [2,3]$

$L_i^{k_2}(\lambda) < 0$ and $R_i^{k_1}(\lambda) < 0$. Then k_1 must appear in Q whereas k_2 must be in $P - Q$. Then possible Q's are $Q_1 = \{1,3\}$ and $Q_2 = \{1\}$

For $\lambda \in [5.5,6]$

$L_i^{k_2}(\lambda) < 0$ and $R_i^{k_1}(\lambda) < 0$ and $R_i^{k_1}(\bar{\lambda})$. Then k_1 and k_3 must appear in Q whereas k_2 must be in $P - Q$. Then possible Q is: $Q_1 = \{1,3\}$

Chapter 3

EXTENSIONS

It is possible to modify the algorithm presented in Chapter 2 to solve a larger set of problem instances. The algorithm finds a feasible solution to P1,

(P1)
$$d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I$$
$$x_j \in S_j \text{ for } j \in J$$

where

R.1 $S_j \bigcap S_k = \emptyset$ for $(j,k) \in I$

R.2 I is chosen so that LN is isomorphic to a subgraph of BW_m.

R.3 Each S_j is a subedge $[a_j, b_j]$ in edge $[v_{p_j}, v_{q_j}]$

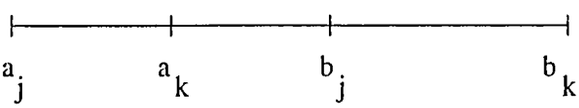
In this chapter, we manage to relax the first restriction totally while remaining polynomial and provide some extensions to the other relaxations again without increasing the complexity too much. We will first explain the difficulties that result from the relaxations then present our suggestion for modification of the algorithm and give the complexity of the work that should be done additionally.

3.1 RELAXATION OF R1

3.1.1 Difficulties

First of all, let us mention the difficulties that arises from the relaxation of this restriction. In the algorithm, we frequently refer to Lemma 1, which states that given $S_j \cap S_k = \emptyset$, $x_j \in S_j$ and $x_k \in S_k$, any shortest path between x_j and x_k includes either a_k or b_k. Using this lemma, we can state that, regardless of the number of pieces $S_j(\lambda)$ has, if we have an expression for the extreme points of $S_j(\lambda)$, we can form $S_k^j(\lambda)$. Moreover, $S_k^j(\lambda)$ has at most two pieces (each of the piece should contain either a_k or b_k).

But when $S_j \cap S_k \neq \emptyset$, shortest path between x_j and x_k that does not need to include a_k or b_k. For example;

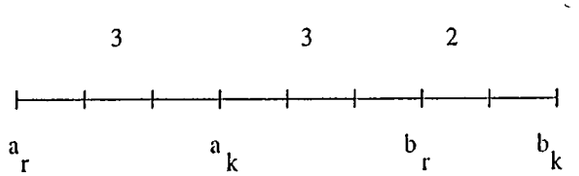


Let $x_j \in [a_k, b_j]$ then for any $x_k \in S_k$ there might be a shortest path between x_j and x_k does not include a_k or b_k.

For $x_j \in [a_j, a_k]$ the previous ideas are still valid. So special treatment for the points in the intersection is necessary.

3.1.2 Suggested Extension

Suppose $S_j \cap S_k \neq \emptyset$ and (j, k) is an arc in DLN. From the reduction phase we know that it is possible to construct $S_j(\lambda)$ without interference of its relation with S_k. Considering the following case will provide an insight.



Suppose $(r,k) \in DLN$, $S_r \cap S_k \neq \emptyset$ and $b_{rk} = 1$.

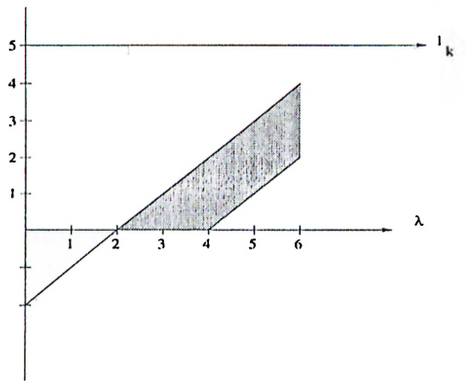


Figure 3.1: Feasible region determination graph of S_k

The shaded region identifies the pairs (x, x_k) that satisfy $d(x, x_k) \leq 1$. As can easily be seen, it is not possible to express the feasible region by means of the region between a unique linear function and the axis or $y = l_k$ line since a point is reachable from S_j does not guarantee that either a_j or b_j is also reachable. We need two linear functions to express the boundaries of the feasible region.

At any λ, $S_j(\lambda)$ may consist of many pieces (if in-degree of j is K then, at any λ, there can be at most $K+1$ pieces.) One differentiation about these pieces is that their Q sets are different. That is, the subset of in-neighbors of j, that cover points in that piece via its left pieces is common to all points in a piece.

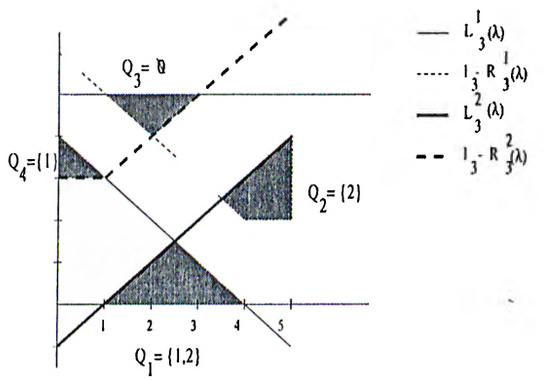


Figure 3.2: Pieces and Q's for $S_3(\lambda)$

Let us denote each piece of S_j by $P_Q(\lambda)$. Then for a specific $\bar{\lambda}$, consider a $P_Q(\bar{\lambda})$ and call its extreme points $y_1(\bar{\lambda})$ and $y_2(\bar{\lambda})$.

Let us define

$$S_k^{jQ}(\bar{\lambda}) = \{z \in S_k : d(x,y) \leq b_{jk} \text{ for some } y \in P_Q(\bar{\lambda})\}$$

Then $S_k^{jQ}(\bar{\lambda})$, the feasible region that this piece forms in S_k, is determined as follows:

1. If $P_Q(\bar{\lambda}) \cap S_k = \emptyset$ then

$$S_k^{jQ}(\bar{\lambda}) = N(y_1(\bar{\lambda}), b_{jk}) \cup N(y_2(\bar{\lambda}), b_{jk})$$

2. If $P_Q(\bar{\lambda}) \cap S_k \neq \emptyset$ then

$$S_k^{jQ}(\bar{\lambda}) = N(y_1(\bar{\lambda}), b_{jk}) \cup N(y_2(\bar{\lambda}), b_{jk}) \cup P_Q(\bar{\lambda}) \cap S_k$$

So a special treatment is required for the pieces that appear in the intersection. Previously we did not keep $K_Q(\lambda)$ and $M_Q(\lambda)$ if they do not determine $L_j(\lambda)$ and $R_j(\lambda)$. But for the pairs that $S_j \cap S_k \neq \emptyset$, which can be detected at the distance calculation step we should keep all $K_Q(\lambda)$ and $M_Q(\lambda)$.

Let us define

$$LL_k^{jQ}(\lambda) = Max_{x \in S_k^{jQ}(\bar{\lambda})} d(x, b_k)$$

$$UL_k^{jQ}(\lambda) = Max_{x \in S_k^{jQ}(\bar{\lambda})} d(x, a_k)$$

In the rest of this section, we will first explain the situation for a fixed λ, give the definitions and then give a parametric approach. We need to investigate all possible ordering of a_j, a_k, b_j, b_k. For details refer to Figure 2.10.

Suppose for $\bar{\lambda} \in F$, $S_j(\bar{\lambda})$ consists of three pieces. $P_{Q_1}(\bar{\lambda})$, $P_{Q_2}(\bar{\lambda})$ and $P_{Q_3}(\bar{\lambda})$

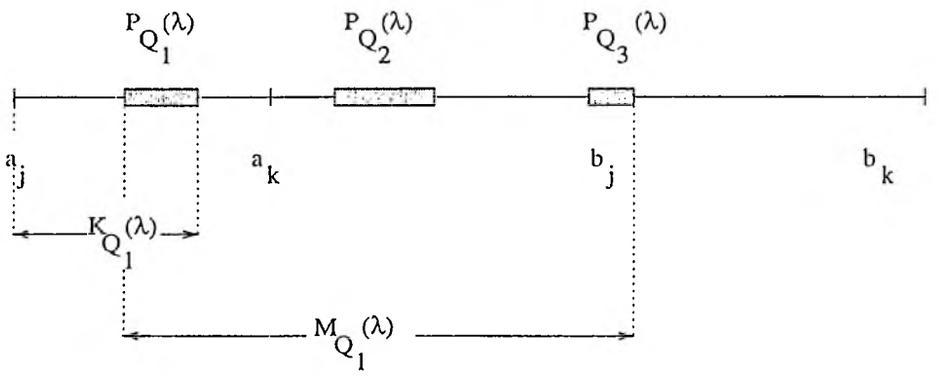


Figure 3.3: $S_j(\bar{\lambda})$

$K_Q(\lambda)$ is the maximum distance from a_j to a point in P_Q, if it is positive and if it is negative then this indicates that for that value of λ there is no piece in S_j that is formed by the intersection of the left pieces of Q's and the right pieces of $P \backslash Q$'s on S_j. $M_Q(\lambda)$ is calculated similarly with respect to b_j.

Case1 $Q \in Q^+(\bar{\lambda})$ and $K_Q(\bar{\lambda}) \leq d(a_j, a_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$\mathrm{LL}_k^{jQ}(\lambda) = \begin{cases} l_k & \text{if } UL_k^{jQ}(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

In our example

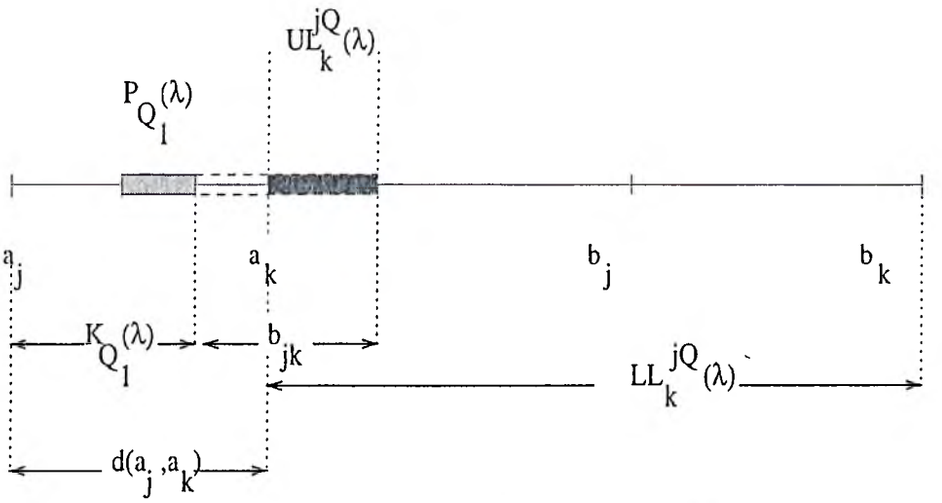


Figure 3.4: $UL_k^{jQ_1}(\bar{\lambda})$ and $LL_k^{jQ_1}(\bar{\lambda})$

Case2 $Q \in Q^+(\bar{\lambda})$ and $K_Q(\bar{\lambda}) > d(a_j, a_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

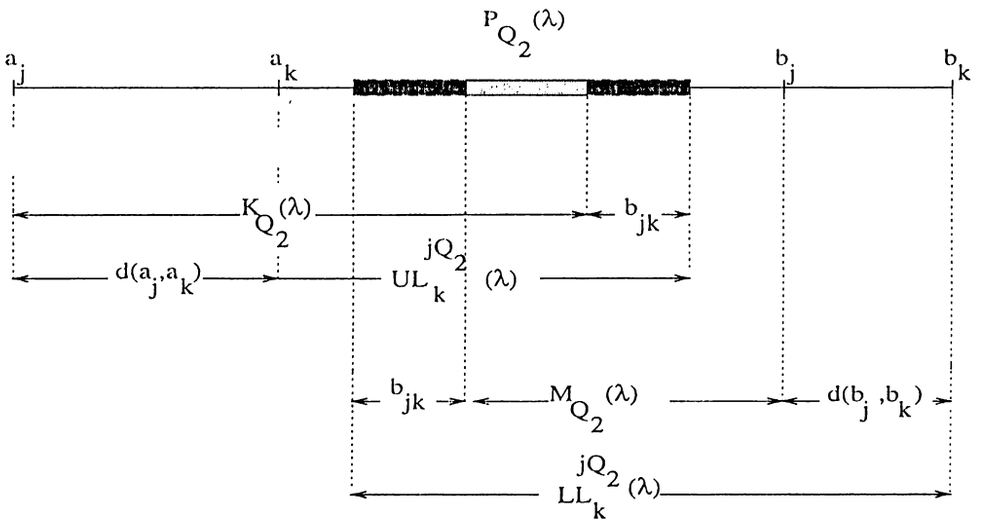


Figure 3.5: $UL_k^{jQ_2}(\bar{\lambda})$ and $LL_k^{jQ_2}(\bar{\lambda})$

We can summarize our findings for all possible realizations of a_j, a_k, b_j, b_k as follows:

If $S_j, S_k \subseteq [v_p, v_q] \in E$, (j,k) is an arc in DLN, $S_j \cap S_k \neq \emptyset$, then $S_k^j(\lambda)$ is found as follows

Determine $S_j(\lambda)$ and all $M_Q(\lambda)$ and $K_Q(\lambda)$'s.

1. If $\omega_{pq}(a_j) \leq \omega_{pq}(a_k)$ and $\omega_{pq}(b_j) \leq \omega_{pq}(b_k)$

   For all $Q \in P$

   (a) $K_Q(\bar{\lambda}) \leq d(a_j, a_k)$

   $$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

   $$\mathrm{LL}_k^{jQ}(\lambda) = \begin{cases} l_k & \text{if } UL_k^{jQ}(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

(b) $K_Q(\bar{\lambda}) > d(a_j, a_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

2. If $\omega_{pq}(a_j) \leq \omega_{pq}(a_k)$ and $\omega_{pq}(b_j) > \omega_{pq}(b_k)$

For all $Q \in P$

(a) If $K_Q(\bar{\lambda}) \leq d(a_j, a_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$\mathrm{LL}_k^{jQ}(\lambda) = \begin{cases} l_k & \text{if } UL_k^{jQ}(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

(b) If$d(a_j, a_k) < K_Q(\bar{\lambda}) \leq d(a_j, b_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

(c) If $K_Q(\bar{\lambda}) > d(a_j, b_k)$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

$$\mathrm{UL}_k^{jQ}(\lambda) = \begin{cases} l_k & \text{if } LL_k^{jQ}(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

3. If $\omega_{pq}(a_j) > \omega_{pq}(a_k)$ and $\omega_{pq}(b_j) \leq \omega_{pq}(b_k)$

For all $Q \in P$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} + d(b_j, b_k), l_k\}$$

4. If $\omega_{pq}(a_j) > \omega_{pq}(a_k)$ and $\omega_{pq}(b_j) > \omega_{pq}(b_k)$

For all $Q \in P$

(a) $K_Q(\bar{\lambda}) \leq d(a_j, b_k)$

$$UL_k^{jQ}(\lambda) = Min\{K_Q(\bar{\lambda}) + b_{jk} - d(a_j, a_k), l_k\}$$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

(b) $K_Q(\bar{\lambda}) > d(a_j, b_k)$

$$LL_k^{jQ}(\lambda) = Min\{M_Q(\bar{\lambda}) + b_{jk} - d(b_j, b_k), l_k\}$$

$$\mathrm{UL}_k^{jQ}(\lambda) = \begin{cases} l_k & \text{if } LL_k^{jQ}(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

After calculating all $UL_k^{jQ}(\lambda)$ and $LL_k^{jQ}(\lambda)$, we intersect them with $S_k^i(\lambda)$ where i's are other in-neighbors of k.

Consider the following feasible region determination graph of $S_j(\lambda)$ and use of it in the calculation of $S_k^j(\lambda)$. In Figure 3.6, we expand S_j by b_{jk} and then in Figure 3.7, we intersect what we found with S_k.

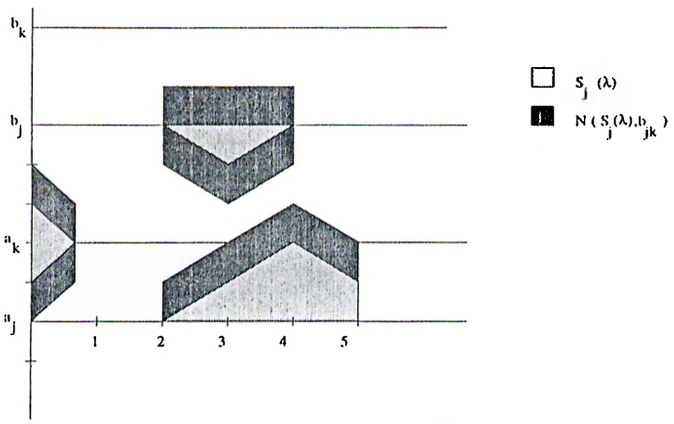


Figure 3.6: Expansion of $S_j(\lambda)$ by b_{jk}

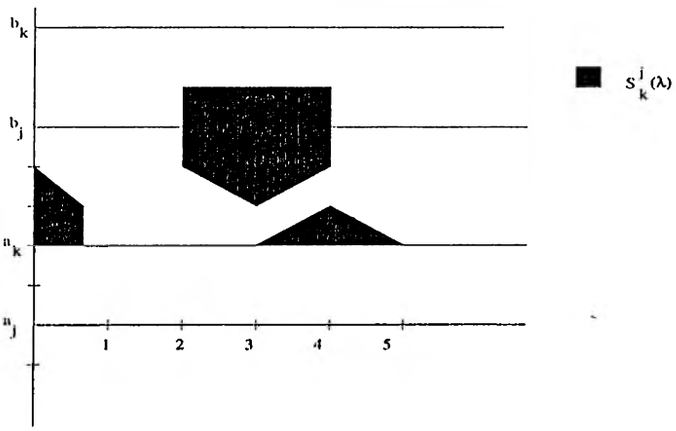


Figure 3.7: Intersection of $N(S_j, b_{jk})$ with S_k

3.2 RELAXATION OF R2

Restriction 2 guarantees that after the orientation step the nodes of DLN (except r) has at most one out-neighbor. In Chapter 2, we mentioned the difficulties related with the nodes whose out-degrees are more than one. Briefly, in those cases, there is no finite dominating set of points at which the facility can be located.

One immediate reaction to such a restriction is to decompose the constraint set into subsets so that each subset satisfies restriction 2, find *all* feasible solutions to each subset and then find the intersection of these feasible solution sets. Obviously, if the algorithm terminates infeasible for any of subsets, then the whole problem is infeasible.

Our algorithm does not find all feasible solutions but finds *a* feasible solution, if there is any. Here we will add a phase to the algorithm so that at the end we obtain the composite region of each facility. We will give definition of composite region later. The definition is taken from Tansel and Yesilkokcen [13].

3.2.1 Second Reduction

Suppose that the facilities are renamed according to the order specified in the orientation phase. Now let us apply the reduction phase of the algorithm to this DLN. As is clear from the definition of $S_i(\lambda)$, for any $x_i \in S_i(\lambda)$ we can find points $x_j \in S_j$ for $j < i$ that satisfy the relevant distance constraints,i.e. the constraints that include x_k where $k \leq i$. That is why $S_i(\lambda)$ is a composite region for x_i for the partial problem which consists of $d(x_j, x_k) \leq b_{jk}$ for $1 \leq j < k \leq i$ and $x_j \in S_j$ for $j \leq i$. But for the same x_i, we cannot guarantee that there exists x_j where $j > i$ that satisfy whole constraint set. Then $S_i(\lambda)$s , except $i = m$, are not composite regions for the whole problem.

Now we will add a second reduction phase which moves in the reverse

3.2 RELAXATION OF R2

Restriction 2 guarantees that after the orientation step the nodes of DLN (except r) has at most one out-neighbor. In Chapter 2, we mentioned the difficulties related with the nodes whose out-degrees are more than one. Briefly, in those cases, there is no finite dominating set of points at which the facility can be located.

One immediate reaction to such a restriction is to decompose the constraint set into subsets so that each subset satisfies restriction 2, find *all* feasible solutions to each subset and then find the intersection of these feasible solution sets. Obviously, if the algorithm terminates infeasible for any of subsets, then the whole problem is infeasible.

Our algorithm does not find all feasible solutions but finds *a* feasible solution, if there is any. Here we will add a phase to the algorithm so that at the end we obtain the composite region of each facility. We will give definition of composite region later. The definition is taken from Tansel and Yesilkokcen [13].

3.2.1 Second Reduction

Suppose that the facilities are renamed according to the order specified in the orientation phase. Now let us apply the reduction phase of the algorithm to this DLN. As is clear from the definition of $S_i(\lambda)$, for any $x_i \in S_i(\lambda)$ we can find points $x_j \in S_j$ for $j < i$ that satisfy the relevant distance constraints,i.e. the constraints that include x_k where $k \leq i$. That is why $S_i(\lambda)$ is a composite region for x_i for the partial problem which consists of $d(x_j, x_k) \leqq b_{jk}$ for $1 \leq j < k \leq i$ and $x_j \in S_j$ for $j \leq i$. But for the same x_i, we cannot guarantee that there exists x_j where $j > i$ that satisfy whole constraint set. Then $S_i(\lambda)$s , except $i = m$, are not composite regions for the whole problem.

Now we will add a second reduction phase which moves in the reverse

direction of the first reduction, from m to 1, and recursively reduces $S_j(\lambda)$ to $F_j(\lambda)$.

If $|\Gamma(j)| = 1$ and $\Gamma(j) = \{k\}$;

$$F_j(\lambda) = \{x_j \in S_j(\lambda) : d(x_j, x_k) \leq b_{jk} \text{ for some } x_k \in F_k(\lambda)\}$$

If $|\Gamma(j)| = 0$

$$F_j(\lambda) = S_j(\lambda)$$

Second Reduction

Step 0. *Initialization*

For node 1

$$\mathrm{F}_1(\lambda) = \begin{cases} \lambda & \text{if } \lambda \in F \\ \emptyset & \text{otherwise} \end{cases}$$

For node $k \in S$

$$F_k(\lambda) = S_k(\lambda)$$

Step 1. *Recursive Step*

For node i whose out-neighbor j is reduced;

$$\text{Calculate } L_i^{jL}(\lambda), L_i^{jR}(\lambda) R_i^{jL}(\lambda) R_i^{jR}(\lambda) \text{ for } \lambda \in F.$$

Then

$$L_i^j(\lambda) = Max\{L_i^{jL}(\lambda) L_i^{jR}(\lambda)\}$$
$$R_i^j(\lambda) = Max\{R_i^{jL}(\lambda) R_i^{jR}(\lambda)\}$$

Let

$$T_i^j(\lambda) = \begin{cases} [\mathrm{a}_i, \overline{L_i^j(\lambda)}] & \text{if } L_i^j(\lambda) \geq 0, R_i^j(\lambda) < 0 \\ [\overline{R_i^j(\lambda)}, b_i] & \text{if } L_i^j(\lambda) < 0, R_i^j(\lambda) \geq 0 \\ [\mathrm{a}_i, \overline{L_i^j(\lambda)}] \cup [\overline{R_i^j(\lambda)}, b_i] & \text{if } L_i^j(\lambda) \geq 0, R_i^j(\lambda) \geq 0 \end{cases}$$

$$F_i(\lambda) = S_i(\lambda) \cap T_i^j(\lambda)$$

As you might have noticed, for $\bar{\lambda} \in F$, $L_i^j(\bar{\lambda}) \geq 0$ or $R_i^j(\bar{\lambda}) \geq 0$. That is if $\lambda \in F$ then all $F_j(\bar{\lambda}) \neq \emptyset$. This follows from the following observation. $F_j(\bar{\lambda})$ includes at least the point that will be found in construction step if we choose $\lambda = \bar{\lambda} \in F$. Consequently, no one λ is eliminated in this step, but we reduce $S_j(\lambda)$ to a subset of it , $F_j(\lambda)$ for $\lambda \in F$.

Then for any $\lambda \in F$,

$$(\lambda, F_2(\lambda), F_3(\lambda), ...F_m(\lambda))$$

is a vector of set of points such that if we choose a point $\bar{x}_j$ in $F_j(\lambda)$ then we can construct a feasible location vector $X = (x_1, x_2, ...x_m)$ where $x_j = \bar{x}_j$. For $i < j$, x_is are located according to the application of Construction Step for the functions in the First Reduction and for $i > j$, x_is are located according to the application of the Construction Step to the Second Reduction.

In fact, the composite region vector which is independent of λ is:

$$(F, \cup_{\lambda \in F} F_2(\lambda), \cup_{\lambda \in F} F_3(\lambda), ... \cup_{\lambda \in F} F_m(\lambda))$$

F_i is the projection of $F_i(\lambda)$ onto the y axis.

The complexity of this step is the same with the complexity of the First Reduction. Because the calculation of each of $L_i^{jL}(\lambda), L_i^{jR}(\lambda), R_i^{jL}(\lambda)$ and $R_i^{jR}(\lambda)$ takes constant time (comparison of three linear functions for each) and then intersecting $T_i^j(\lambda)$ with $S_i(\lambda)$ can be done in constant time. Since we need to compare $L_i^j(\lambda)$ s with $K_Q(\lambda)$s (number of $K_Q(\lambda)$ function is limited by 4 by the assumption that LN is isomorphic to BW_m).

Then, since the operations for a facility takes constant time, the time bound of second reduction is also $O(m)$.

Here is the application of the Second Reduction to the example that was given in Chapter 2.

Step 0. $F_1(\lambda) = \lambda$ for $\lambda \in \{3\} \cup [3.5, 4]$

$S = \{5\}$ $F_5(\lambda) = S_5(\lambda)$ for $\lambda \in \{3\} \cup [3.5, 4]$

Step 1. Determination of $F_4(\lambda)$

$$L_4^{5L}(\lambda) = Min\{11 - Min\{10 + L_5(\lambda), 15 - L_5(\lambda)\}, 2\}$$
$$L_4^{5R}(\lambda) = Min\{11 - Min\{10 + R_5(\lambda), 15 - R_5(\lambda)\}, 2\}$$
$$R_4^{5L}(\lambda) = Min\{11 - Min\{12 + L_5(\lambda), 13 - L_5(\lambda)\}, 2\}$$
$$R_4^{5R}(\lambda) = Min\{11 - Min\{8 + R_5(\lambda), 17 - R_5(\lambda)\}, 2\}$$

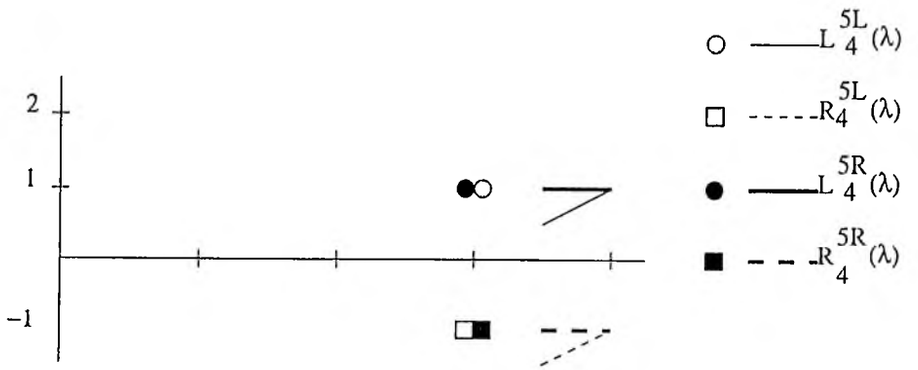


Figure 3.8: $L_4^5(\lambda)$ and $R_4^5(\lambda)$ determination

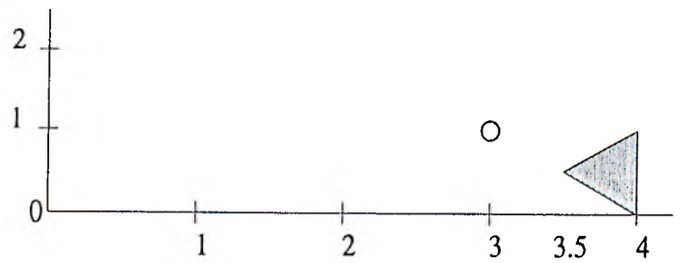


Figure 3.9: $F_4(\lambda)$

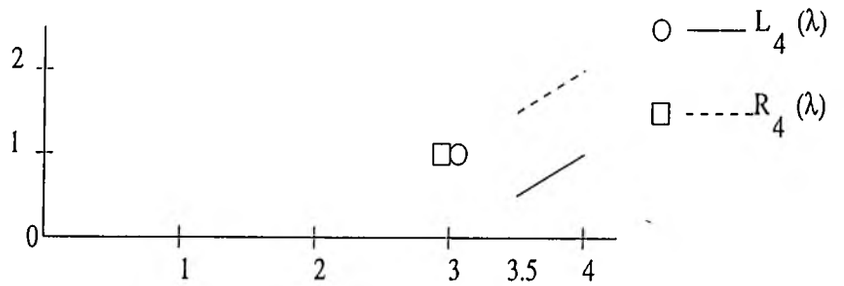


Figure 3.10: $L_4(\lambda)$ and $R_4(\lambda)$

Step 1. Determination of $F_3(\lambda)$

$$L_3^{4L}(\lambda) = Min\{10 - Min\{7 + L_4(\lambda), 11 - L_4(\lambda)\}, 4\}$$
$$L_3^{4R}(\lambda) = Min\{10 - Min\{9 + R_4(\lambda), 9 - R_4(\lambda)\}, 4\}$$
$$R_3^{4L}(\lambda) = Min\{10 - Min\{11 + L_4(\lambda), 15 - L_4(\lambda)\}, 4\}$$
$$R_3^{4R}(\lambda) = Min\{10 - Min\{13 + R_4(\lambda), 13 - R_4(\lambda)\}, 4\}$$

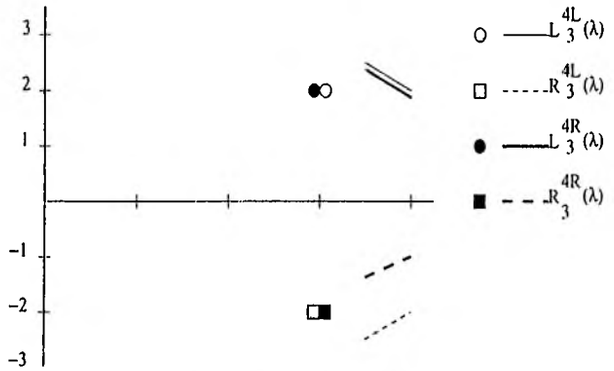


Figure 3.11: $L_3^4(\lambda)$ and $R_3^4(\lambda)$ determination

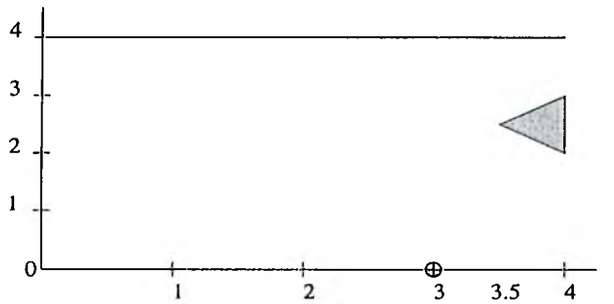


Figure 3.12: $F_3(\lambda)$

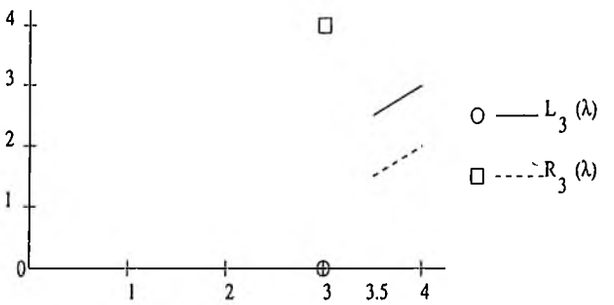


Figure 3.13: $L_3(\lambda)$ and $R_3(\lambda)$

Step 1. Determination of $F_2(\lambda)$

$$L_2^{3L}(\lambda) = Min\{12 - Min\{15 + L_3(\lambda), 15 - L_3(\lambda)\}, 3\}$$
$$L_2^{3R}(\lambda) = Min\{12 - Min\{11 + R_3(\lambda), 19 - R_3(\lambda)\}, 3\}$$
$$R_2^{3L}(\lambda) = Min\{12 - Min\{12 + L_3(\lambda), 12 - L_3(\lambda)\}, 3\}$$
$$R_2^{3R}(\lambda) = Min\{12 - Min\{8 + R_3(\lambda), 16 - R_3(\lambda)\}, 3\}$$

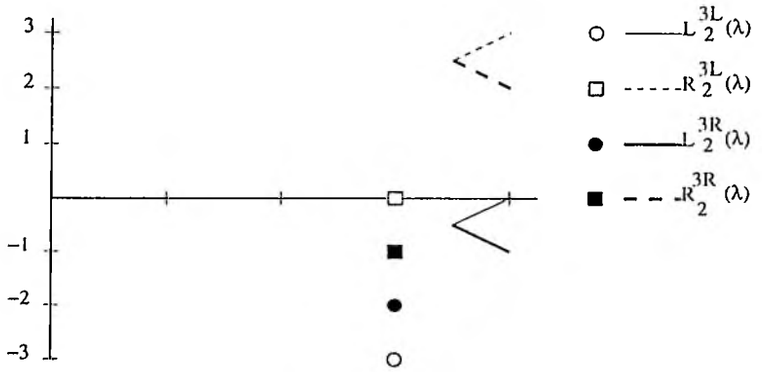


Figure 3.14: $L_2^3(\lambda)$ and $R_2^3(\lambda)$ determination

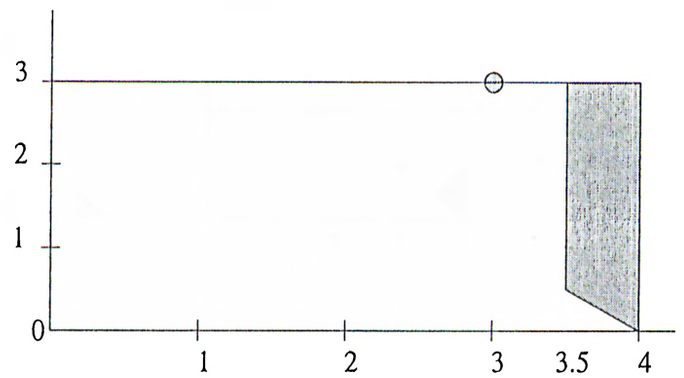


Figure 3.15: $F_2(\lambda)$

In the following page, we give $S_j(\lambda)$ and $F_j(\lambda)$ sets together.

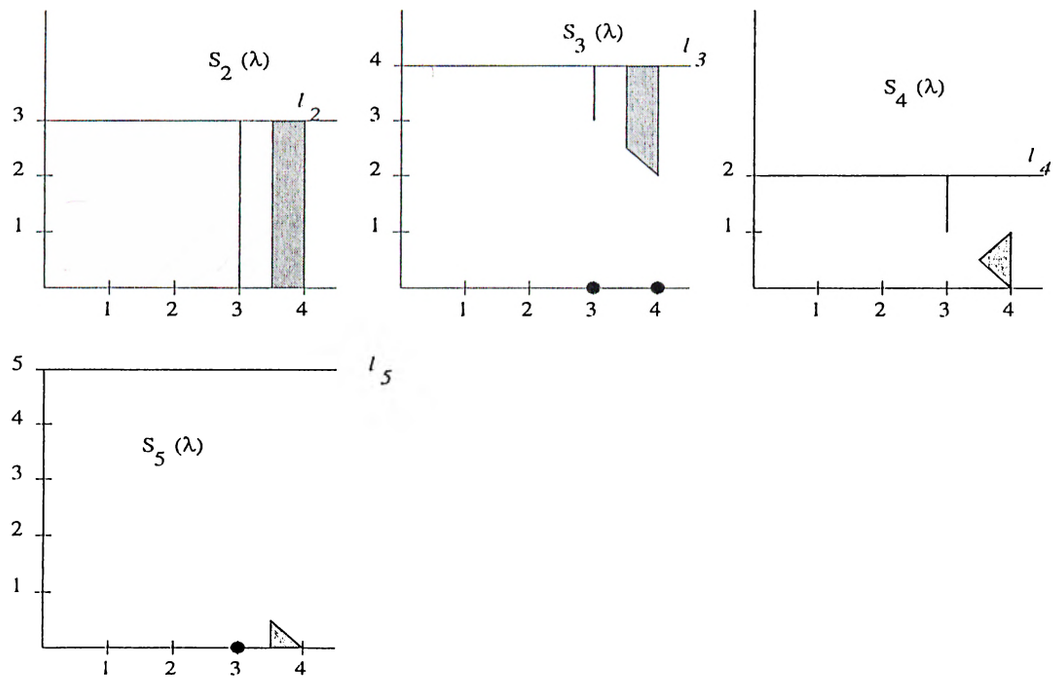


Figure 3.16: After First Reduction

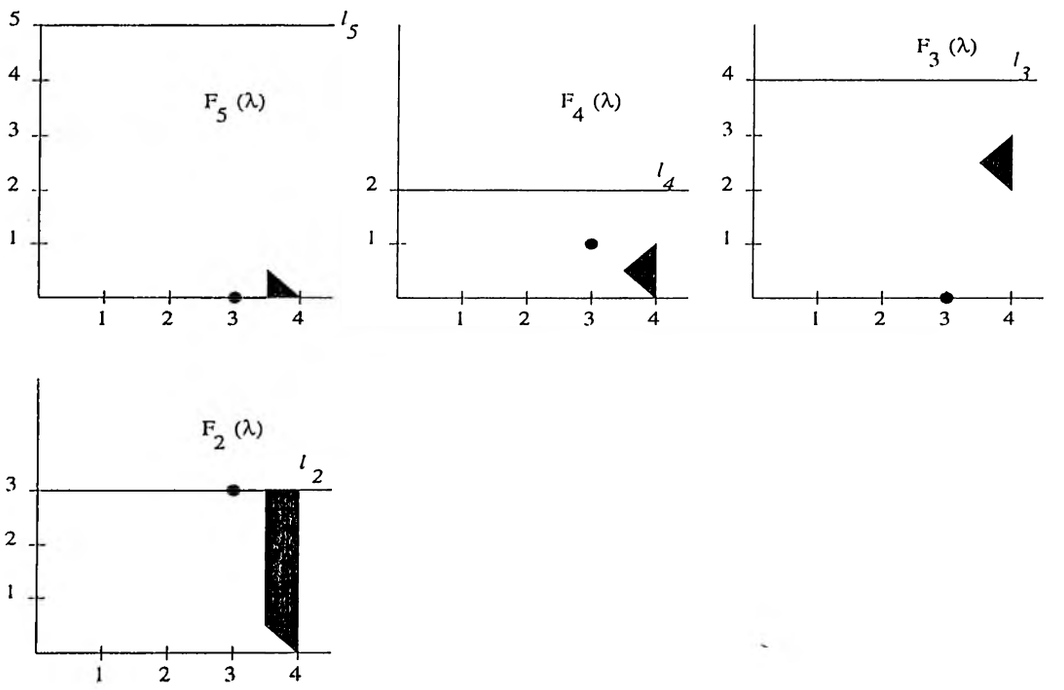


Figure 3.17: After Second Reduction

For $\lambda = 3.5$

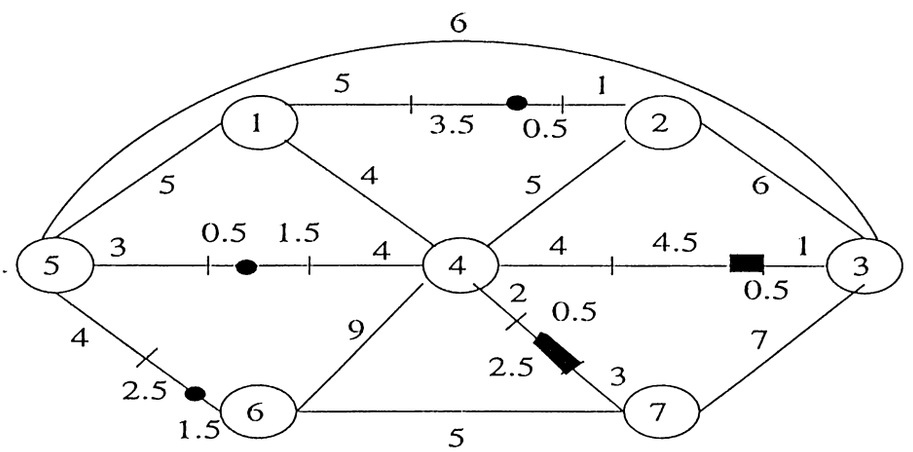
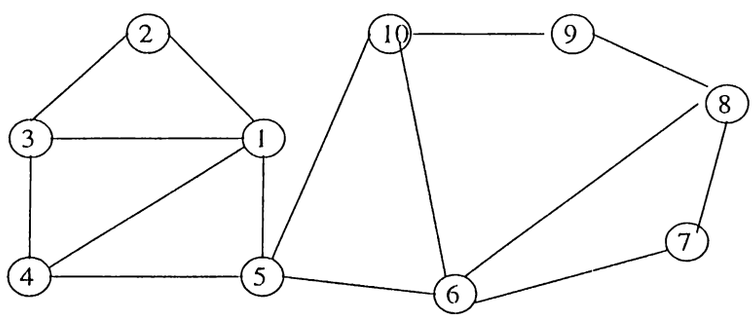


Figure 3.18: Composite Regions for $\lambda = 3.5$

As it can be seen for any $\bar{x}_j$ chosen in the shaded region of S_j, we can find other x_is that satisfy all the constraints.

3.2.2 Decomposition

Suppose we have the following Linkage Network, $LN = (M, I)$



It can be seen from the figure that there is no node whose removal leaves a collection of subtrees (that is no node is common to all cycles). The question is, can we find a solution to that problem by using the algorithm we have?

Suppose we partition the arc set into two subsets :

$Arc_1 = \{(1,2),(1,3),(1,4),(1,5),(2,3),(3,4),(4,5)\}$

$$Arc_2 = \{(6,5),(6,7),(6,8),(6,9),(6,10),(7,8),(8,9),(9,10),(10,5)\}$$

Then the problem:

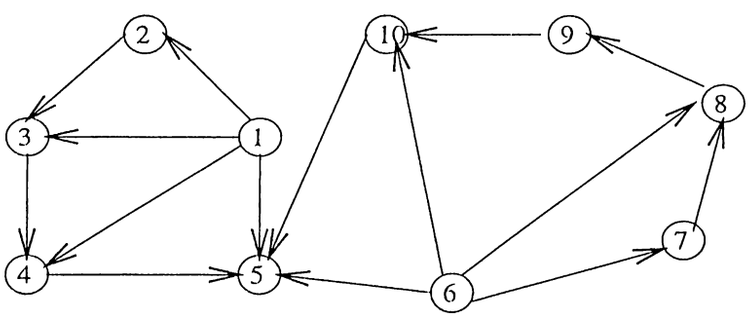
$$(P1) \qquad d(x_j, x_k) \leq b_{jk} \quad \text{for } (j,k) \in I \text{ and } 1 \leq j < k \leq 5$$
$$x_j \in S_j \quad \text{for } 1 \leq j \leq 5$$

$$(P2) \qquad d(x_j, x_k) \leq b_{jk} \quad \text{for } (j,k) \in I \text{ and } 5 \leq j < k \leq 10$$
$$x_j \in S_j \quad \text{for } 5 \leq j \leq 10$$

can be solved via the algorithm that was introduced in Chapter 2. Moreover, we can find the composite regions of the facilities with respect to $P1$ and $P2$.

If there were no common node then the Cartesian product of the composite region vectors will be our solution. But now we have a common node ,5.

We can assign directions to LN_1 and LN_2 (linkage networks of the respective problems) such that 5 is the last element of the order arrays A_1, A_2.



Let $\lambda_1 \in [0, l_1]$ and $\lambda_2 \in [0, l_6]$

We can determine independently $S_5(\lambda_1)$ and $S_5(\lambda_2)$. These are composite regions of S_5 for the problems $P1$ and $P2$. Let us denote F of problem $P1$ and $P2$ as F_1 and F_2, respectively. Then,

If $[\bigcup_{\lambda_1 \in F_1} S_5(\lambda_1)] \bigcap [\bigcup_{\lambda_2 \in F_2} S_5(\lambda_2)]$ is nonempty, then we can construct a feasible solution to the original problem. Since

$$\text{if } \bar{x} \in [\bigcup_{\lambda_1 \in F_1} S_5(\lambda_1)] \bigcap [\bigcup_{\lambda_2 \in F_2} S_5(\lambda_2)], \text{ then}$$

$$\bar{x} \in \bigcup_{\lambda_1 \in F_1} S_5(\lambda_1) \text{ and}$$

$$\bar{x} \in S_5(\bar{\lambda}_1) \text{ for some } \bar{\lambda}_1 \in F_1.$$

Then we can locate x_4, x_3, x_2 by using the construction phase of the algorithm and given x_5 is located at $\bar{x}$ (location of x_1 is already decided by the choice of $\bar{\lambda_1}$.

Similarly,

$$\bar{x} \in (\bigcup_{\lambda_2 \in F_2} S_5(\lambda_2)) \text{ and}$$

$$\bar{x} \in S_5(\bar{\lambda}_2)) \text{ for some } \bar{\lambda}_2 \in F_2$$

Then we can locate x_{10}, x_9, x_8, x_7 by using the construction phase of the algorithm and given x_5 is located at $\bar{x}$ (location of x_6 is already decided by the choice of $\bar{\lambda_2}$).

Then we obtain a feasible location vector $X = (x_1, x_2, ...\bar{x}, x_6, ...x_{10})$.

Findings in this part can be generalized as follows:

Suppose for cut vertex $j \in M$, the blocks can be characterized as subgraphs of LN such that there exists a node whose deletion breaks all cycles. Suppose (V_i, E_i) for $i = 1, 2, ..k$ are blocks of j, and they are isomorphic to a subgraph of $BW_{|V_i|}$

Apply the algorithm for each (V_i, E_i) independent of the other subsets. Then $F_j^i = \bigcup_{\lambda_i \in F_i} F_j(\lambda_i)$ is the composite region for facility j with respect to the constraint set that are specified by the arcs in block i. If j is the root of block i then $F_j^i = F_i$

If $\bigcap_{i=1}^k F_j^i = \emptyset$ then there is no feasible solution to the whole problem.

If $x \in \bigcap_{i=1}^k$ then locate j at x and find λ_i for which $x \in F_j(\lambda_i)$ for $i = 1, 2..k$. Then find the exact locations of the other facilities by applying construction phase to each subset independently with x and λ_i information.

Here is the test that we will apply in order to decide whether the problem can be solvable via the algorithm we proposed or not:

For each node i of LN:

Remove node i from LN and obtain k many components (where $1 \leq k \leq m-1$). Then for each component j, add node i and arcs between i and nodes of component j and apply the orientation phase of the algorithm to the resulting graph. If all components are feasible then we can solve that problem by our algorithm.

Now, let us consider the complexity of solving a problem via decomposition. Since distance calculation phase will be applied only once, the order of the algorithm is maximum of n and the size of the largest component which is surely less than m. Therefore, the order of the algorithm is $O(n^3)$.

3.3 RELAXATION OF R3

First of all, let us mention the difficulties that result from relaxation of this restriction.

In the algorithm at any iteration of reduction phase, we first reduce S_j to $S_j(\lambda)$ according to the distance constraints between j and its in-neighbors.

The reduction is based on intersection of left and right pieces that the in-neighbors form on S_j. Then we determine the extreme points of $S_j(\lambda)$ in order to use them in determination of $S_k(\lambda)$, where k is the unique out-neighbor of j.

We proved in observation 3 that the expansion of the extreme points of $S_j(\lambda)$ is the same as the expansion of the whole $S_j(\lambda)$. This observation can be stated only if S_j is restricted to an edge and $S_j \cap S_k = \emptyset$. Since if one of the interior point of S_j is a node then it is possible that this node reaches to S_k while a_j and b_j cannot.

Moreover, our observation ' $S_j^i(\lambda)$ has at most two pieces, $[a_j, \overline{L_j^i(\lambda)}]$ and $[\overline{R_j^i(\lambda)}, b_j]$ ' is valid only if S_j is restricted to an edge, $S_j \cap S_k = \emptyset$ and S_j is a

convex subedge (that is, for any x,$y \in S_j$, the shortest path between x and y is included in S_j.)

Since most of the ideas are based on restricting S_j to an edge, it does not seem easy to relax this assumption.

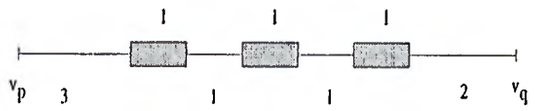
But we can relax restriction 3 in the following way:

Suppose the transport network satisfies the triangular inequality constraints. That is, length of each edge is equal to the shortest path distance between its ends. Then we can replace R3 with:

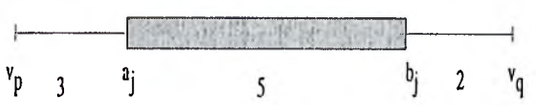
- Each S_j is restricted to an edge of the transport network

Now, we allow S_j to have disjoint segments but all of them should be on the same edge.

Let us demonstrate our suggestion on an example:



Let us take S_j as the minimal subedge that contains S_j and assign the end of that minimal subedge as a_j and b_j.



Determine $S_j(\lambda)$ as before. Suppose 1 and 2 are in-neighbors of j and the feasible region determination graph is as follows:

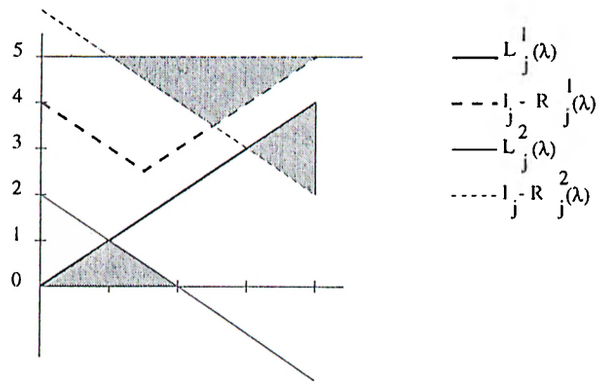
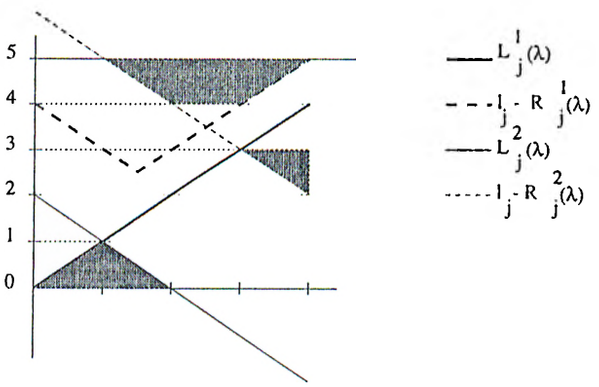


Figure 3.19: $S_j(\lambda)$ before consideration of feasibility

Now we will remove the points that does not appear in real S_j. That is, we will erase the shaded regions where $y \in (1,2) \cup (3,4)$.



. Figure 3.20: $S_j(\lambda)$ after consideration of feasibility

If erasing causes $S_j(\lambda) = \emptyset$ for some λ, then we will remove that λ from F. We need to determine $L_j(\lambda)$ and $R_j(\lambda)$ by considering the graph in Figure 3.20.

Pictorially, all we need to do is to find the upper and lower envelope of the shaded region in Figure 3.20. Here is the modification that we propose for the algorithm.

We can express S_j in terms of intervals:

$$S_j = [0,1] \cup [2,3] \cup [4,5]$$

Then (1,2) and (3,4) intervals do not belong to S_j.

$$\mathrm{K}_{\emptyset}(\lambda) = \begin{cases} l_j & \text{if } R_j^1(\lambda) \geq 0 \text{ and } R_j^2(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{1\}}(\lambda) = \begin{cases} L_j^1(\lambda) & \text{if } L_j^1(\lambda) \geq l_j - R_j^2(\lambda) \text{ and } L_j^1(\lambda) \in S_j \\ 1 & \text{if } L_j^1(\lambda) \geq l_j - R_j^2(\lambda) \text{ and } L_j^1(\lambda) \in (1,2) \\ 3 & \text{if } L_j^1(\lambda) \geq l_j - R_j^2(\lambda) \text{ and } L_j^1(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{2\}}(\lambda) = \begin{cases} L_j^2(\lambda) & \text{if } L_j^2(\lambda) \geq l_j - R_j^1(\lambda) \text{ and } L_j^2(\lambda) \in S_j \\ 1 & \text{if } L_j^2(\lambda) \geq l_j - R_j^1(\lambda) \text{ and } L_j^1(\lambda) \in (1,2) \\ 3 & \text{if } L_j^2(\lambda) \geq l_j - R_j^1(\lambda) \text{ and } L_j^2(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{K}_{\{1,2\}}(\lambda) = \begin{cases} A(\lambda) = min\{L_j^1(\lambda), L_j^2(\lambda)\} & \text{if } A(\lambda) \in S_j \\ 1 & \text{if } A(\lambda) \in (1,2) \\ 3 & \text{if } A(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$L_j(\lambda) \;=\; Max\{K_{\emptyset}, K_{\{1\}}(\lambda), K_{\{2\}}(\lambda), K_{\{1,2\}}(\lambda)\}$$

For $R_j(\lambda)$,

$$\mathrm{M}_{\emptyset}(\lambda) = \begin{cases} A(\lambda) = min\{R_j^1(\lambda), R_j^2(\lambda)\} & \text{if } A(\lambda) \in S_j \\ 1 & \text{if } A(\lambda) \in (1,2) \\ 3 & \text{if } A(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_{\{1\}}(\lambda) = \begin{cases} R_j^2(\lambda) & \text{if} L_j^1(\lambda) \geq l_j R_j^2(\lambda) \text{ and } R_j^2(\lambda) \in S_j \\ 1 & \text{if} L_j^1(\lambda) \geq l_j R_j^2(\lambda) \text{ and } R_j^2(\lambda) \in (1,2) \\ 3 & \text{if} L_j^1(\lambda) \geq l_j R_j^2(\lambda) \text{ and } R_j^2(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_{\{2\}}(\lambda) = \begin{cases} R_j^1(\lambda) & \text{if} L_j^2(\lambda) \geq l_j R_j^1(\lambda) \text{ and } R_j^1(\lambda) \in S_j \\ 1 & \text{if} L_j^2(\lambda) \geq l_j R_j^1(\lambda) \text{ and } R_j^1(\lambda) \in (1,2) \\ 3 & \text{if} L_j^2(\lambda) \geq l_j R_j^1(\lambda) \text{ and } R_j^1(\lambda) \in (3,4) \\ -1 & \text{otherwise} \end{cases}$$

$$\mathrm{M}_{\{1,2\}}(\lambda) = \begin{cases} l_j & \text{if } L_j^1(\lambda) \geq 0 \text{ and } L_j^2(\lambda) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

$$R_j(\lambda) = Max\{M_\emptyset, M_{\{1\}}(\lambda), M_{\{2\}}(\lambda), M_{\{1,2\}}(\lambda)\}$$

The number of comparisons increase by the number of disjoint segments of S_j. Tansel and Yesilkokcen [13] proved that there can be at most $n+1$ disjoint segments of S_j. Therefore, we remain polynomial.

Suppose now the LN is decomposable,i.e, has a cut vertex whose blocks are characterized by 'There exists a node whose deletion breaks all cycles', and G satisfies triangular inequalities. Under these circumstances, let us solve DC.

It was shown how to handle DC_1 constraints and obtain S_j which may consists of up to $|E|(n+1)$ disjoint segments. Now solving DC calls for two decision:

1. Decide on the edge that S_j is on for $j = 1, 2..m$

2. Choose the exact location x_j given that each S_j is restricted on the edge specified in the first decision step.

Decision 2 is exactly the algorithm we proposed with its all possible extensions. Decision 1 requires an enumerative based algorithm. Let E_j be the set of edges, e such that $S_j^e = e \cap S_j \neq \emptyset$. Then choosing $S_j^e \in E_j$ requires $O(|E|^m)$ effort in the worst case. But the techniques that we described in Section 2.5 can be used to improve the average efficiency. For example suppose we have three new facilities and each S_j appears on three edges. Let us denote the possible edges choices for $j = 1, 2, 3$ by S_j^1, S_j^2 and S_j^3. Then our search tree will be

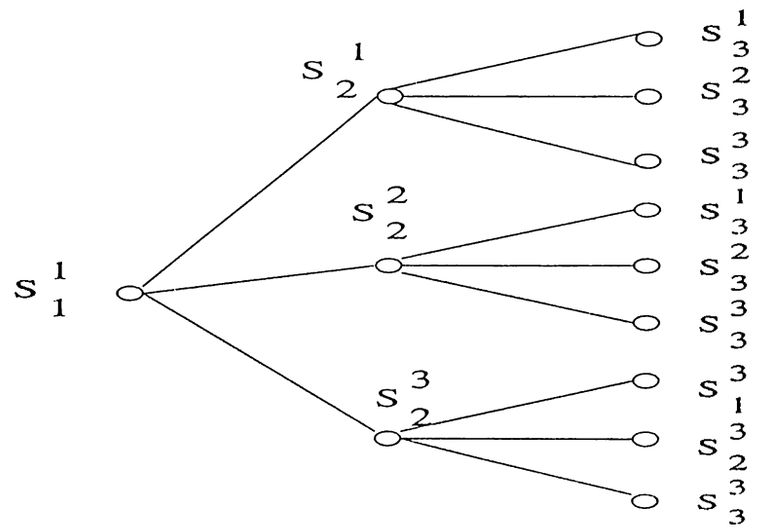


Figure 3.21: Search Tree rooted at S_1^1 before preprocessing

But if the minimum distance between S_1^1 and S_2^3 is larger than b_{12} and if S_2^1 and S_3^3 are apart from each other more than b_{23} then we can reduce the search tree as follows:

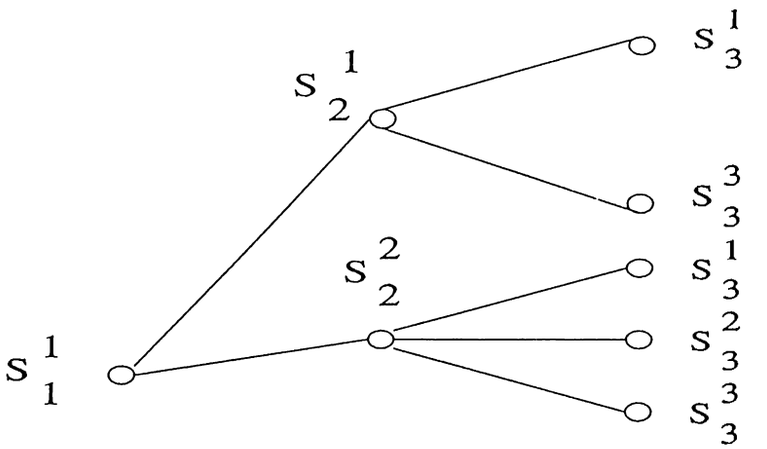


Figure 3.22: Search Tree rooted at S_1^1 after preprocessing

Chapter 4

CONCLUSION

In this thesis we studied *Distance Constraints*. The problem is NP-Complete for cyclic networks and polynomially solvable for tree networks. The only known polynomially solvable case for cyclic networks is the case when the linkage network is a tree. So up to now, there does not exist any polynomially solvable case which has no *tree* assumption on any part of the data.

We do not require a tree structure either for the transport network or the linkage network, but we make assumptions on some other parts of the data. We assume that each new facility is restricted to an a priori specified feasible region which is confined to an edge. Then, with this assumption we can solve DC where the linkage network has a cut vertex whose blocks are characterized by a cyclic structure with the restriction that there exists a node whose deletion breaks all cycles.

In Chapter 2 we provide an algorithm for

$$\text{(P1)} \qquad d(x_j, x_k) \leq b_{jk} \text{ for } (j,k) \in I,$$
$$x_j \in S_j \text{ for } j \in J$$

where

R.1 $S_j \bigcap S_k = \emptyset$ for $(j,k) \in I$

R.2 I is chosen so that LN is isomorphic to a subgraph of BW_m.

R.3 Each S_j is a subedge $[a_j, b_j]$ in edge $[v_{p_j}, v_{q_j}]$

With these restrictions, our method is to reduce the feasible region for facility j, which is initially S_j to a subset of it which is conditional to the point that the facility chosen as a root is located at, such that every point in the reduced feasible region of new facility satisfies the partial distance constraints containing only the processed ones.

In Chapter 3, we provide some extensions to the class that can be solved via the algorithm proposed.

First, we relax $S_j \bigcap S_k = \emptyset$ for $(j,k) \in I$ assumption. In the original case, we only expand only the end points of the subedge that covers $S_j(\lambda)$ but now we need to expand the extreme point of all pieces that $S_j(\lambda)$ has. We need to expand the linear expressions that are candidates for extreme points of the pieces, $K_Q(\lambda)$ and $M_Q(\lambda)$ functions. Due to our Broken wheel assumption, it is possible to assign directions so that each node has at most two in-neighbors and therefore there are four $K_Q(\lambda)$ and $M_Q(\lambda)$ functions.

Secondly, we modified the algorithm so that it gives the composite regions of feasibility for new facilities. The method is to apply the reduction rules in the reverse order. Then we relax restriction $R2$ and if there is a cut vertex in the linkage network whose blocks fulfill the Broken Wheel assumption then we provide a test to identify such cases together with a method to solve such cases which uses the algorithm in Chapter 2 as a subroutine.

Lastly, we relax the assumption that S_j is a convex set restricted on an edge of transport network assumption. We still require each S_j to be restricted on an edge but now we allow S_j to have disjoint segments.

Polynomially solvable cases that we have identified are these cases, but we can propose a solution to the case where the only restriction is the linkage

network is decomposable. This solution is not polynomial but the average performance can be improved by some other preprocessing techniques.

Future Research Directions:

The first decision should be made more efficiently. If an enumeration based scheme will be used then more preprocessing techniques should be found.

Decomposition idea can be enlarged. We have tried the case where the deletion of an edge leaves a feasible linkage network, but some self-referencing problems occur. This case should be investigated more deeply and the reasons of difficulty can be analyzed.

Appendix A

PROOFS

Observation 1

(a) Assume $L_2^1(\lambda) \geq 0$ *and let* $\overline{L_2^1(\lambda)}$ *be the unique point in* S_2 *which is* $L_2^1(\lambda)$ *units away from* a_2. *That is,*

$$\overline{L_2^1(\lambda)} \in S_2 \text{ and } d(\overline{L_2^1(\lambda)}, a_2) = L_2(\lambda).$$

Then any point $y \in [a_2, \overline{L_2^1(\lambda)}]$ *satisfies* $d(x, y) \leq b_{12}$

(b) Assume $R_2^1(\lambda) \geq 0$ *and let* $\overline{R_2^1(\lambda)}$ *be the unique point in* S_2 *which is* $R_2^1(\lambda)$ *units away from* b_2. *That is,*

$$\overline{R_2^1(\lambda)} \in S_2 \text{ and } d(\overline{R_2^1(\lambda)}, b_2) = R_2(\lambda).$$

Then any point $y \in [\overline{R_2^1(\lambda)}, b_2]$ *satisfies* $d(x, y) \leq b_{12}$

Proof (a) By extending the idea in Lemma 1,

$$\begin{aligned} d(x,y) &= Min\{(d(x,a_2) + d(y,a_2), (d(x,b_2) + d(y,b_2)\} \\ d(x,y) &\leq d(x,a_2) + d(a_2,y) \end{aligned}$$

Since

$$L_2^1(\lambda) = Min\{(b_{12} - d(a_2,x)), l_2\}$$

$$L_2^1(\lambda) \leq b_{12} - d(a_2, x)$$
$$d(a_2, x) \leq b_{12} - L_2^1(\lambda)$$

Since $y \in [a_2, \overline{L_2^1(\lambda)}] \Rightarrow d(y, a_2) \leq L_2^1(\lambda)$

Therefore,

$$d(x, y) \leq d(y, a_2) + d(x, a_2) \leq b_{12} - L_2^1(\lambda) + L_2^1(\lambda)$$

And $d(x, y) \leq b_{12}$.

(b) can be proven similarly. □

Remark 1 *If* $L_2^1(\lambda) = l_2$ *then* $R_2^1(\lambda) \geq 0$

Proof $$L_2^1(\lambda) = Min\{(b_{12} - d(a_2, x)), l_2\}$$

If $L_2^1(\lambda) = l_2$ then

$$\begin{aligned} b_{12} - d(a_2, x) &\geq l_2 \\ b_{12} &\geq d(a_2, x) + l_2 \\ b_{12} &\geq d(a_2, b_2) + d(a_2, x) \geq d(b_2, x) \\ b_{12} - d(b_2, x) &\geq 0 \end{aligned}$$

Since $$R_2^1(\lambda) = min\{(b_{12} - d(b_2, x)), l_2\}$$

$R_2^1(\lambda)$ is minimum of two nonnegative numbers. Therefore, $R_2^1(\lambda) \geq 0$ □

Consequences of Remark :

$$\begin{aligned} L_2^1(\lambda) = l_2 &\Rightarrow R_2^1(\lambda) \geq 0 \\ R_2^1(\lambda) < 0 &\Rightarrow L_2^1(\lambda) = b_{12} - d(a_2, x) < l_2 \\ R_2^1(\lambda) = l_2 &\Rightarrow L_2^1(\lambda) \geq 0 \\ L_2^1(\lambda) < 0 &\Rightarrow R_2^1(\lambda) = b_{12} - d(b_2, x) < l_2 \end{aligned}$$

We will use these results in the proof of the following observation.

Observation 2

$$S_2^1(\lambda) = \begin{cases} \emptyset & \text{if } L_2^1(\lambda) < 0, R_2^1(\lambda) < 0 \\ [a_2, \overline{L_2^1(\lambda)}] & \text{if } L_2^1(\lambda) \geq 0, R_2^1(\lambda) < 0 \\ [\overline{R_2^1(\lambda)}, b_2] & \text{if } L_2^1(\lambda) < 0, R_2^1(\lambda) \geq 0 \\ [a_2, \overline{L_2^1(\lambda)}] \cup [\overline{R_2^1(\lambda)}, b_2] & \text{if } L_2^1(\lambda) \geq 0, R_2^1(\lambda) \geq 0 \end{cases}$$

Proof $T_2^1(\lambda) \subseteq S_2^1(\lambda)$ from Observation 1 (a) and (b). Let us now prove that $S_2^1(\lambda) \subseteq T_2^1(\lambda)$.

The proof is by contradiction.We need to investigate all cases.

Let $y \in S_2^1(\lambda)$ but $y \notin T_2^1(\lambda)$

1. $L_2^1(\lambda) < 0, R_2^1(\lambda) < 0$ then $T_2^1(\lambda) = \emptyset$.

   Let $y \in S_2^1(\lambda)$

   $$d(x,y) = Min\{d(y,a_2) + d(a_2,x), d(y,b_2) + d(b_2,x)\}$$

   Without loss of generality, suppose

   $$d(x,y) = d(y,a_2) + d(a_2,x)$$

   Since $L_2^1(\lambda) < 0$, $L_2^1(\lambda) = b_{12} - d(a_2,x)$

   $$d(x,y) = d(y,a_2) + b_{12} - L_2^1(\lambda)$$

   Since $d(y,a_2) \geq 0$ and $L_2^1(\lambda) < 0$

   $d(x,y) > b_{12}$. It is a contradiction $y \notin S_2^1(\lambda)$.

2. $L_2^1(\lambda) \geq 0, R_2^1(\lambda) < 0$ then $T_2^1(\lambda) = [a_2, \overline{L_2^1(\lambda)}]$

   Let $y \in S_2^1(\lambda)$ but $y \notin T_2^1(\lambda)$ then $d(y,a_2) > L_2^1(\lambda)$

   Since $R_2^1(\lambda) < 0, L_2^1(\lambda) = b_{12} - d(a_2,x)$

   $$d(x,y) = Min\{d(y,a_2) + d(a_2,x), d(y,b_2) + d(b_2,x)\}$$

   - If $d(x,y) = d(y,a_2) + d(a_2,x)$

     $$d(x,y) = d(y,a_2) + b_{12} - L_2^1(\lambda)$$

     Since $d(y,a_2) > L_2^1(\lambda)$

     $d(x,y) > b_{12}$ It is a contradiction $y \notin S_2^1(\lambda)$

- If $d(x,y) = d(y,b_2) + d(b_2,x)$

  $$d(x,y) = d(y,b_2) + b_{12} - R_2(\lambda)$$

  Since $d(y,b_2) \geq 0$ and $R_2(\lambda) < 0$

  $d(x,y) > b_{12}$ It is a contradiction $y \notin S_2(\lambda)$

3. $L_2^1(\lambda) < 0, R_2^1(\lambda) \geq 0$ Symmetric to Case 2.

4. $L_2^1(\lambda) \geq 0, R_2^1(\lambda) \geq 0$ then, $T_2^1(\lambda) = [a_2, \overline{L_2^1(\lambda)}] \cup [\overline{R_2^1(\lambda)}, b_2]$

   Let $y \in S_2^1(\lambda)$ but $y \notin T_2^1(\lambda)$

   $$d(x,y) = Min\{d(y,a_2) + d(a_2,x), d(y,b_2) + d(b_2,x)\}$$

   - If $L_2^1(\lambda) = l_2$ or $R_2^1(\lambda) = l_2$ then $T_2^1(\lambda) = S_2$ So, $y \in S_2^1(\lambda) \subseteq S_2 = T_2^1(\lambda)$ and $y \notin T_2^1(\lambda)$ is a contradiction.

   - If $L_2^1(\lambda) < l_2$ and $R_2^1(\lambda) < l_2$ then

     Since $y \notin T_2^1(\lambda)$

     $d(y,a_2) > L_2^1(\lambda)$ and $d(y,b_2) > R_2^1(\lambda)$

     $$d(x,y) \quad = Min \quad \{d(y,a_2) + d(a_2,x), d(y,b_2) + d(b_2,x)\}$$

     Without loss of generality

     $$\begin{aligned} d(x,y) &= d(y,a_2) + d(a_2,x) \\ d(x,y) &= d(y,a_2) + b_{12} - L_2^1(\lambda) \end{aligned}$$

     Since $d(y,a_2) > L_2^1(\lambda) \Rightarrow d(x,y) > b_{12}$

So, $S_2^1(\lambda) \supseteq T_2^1(\lambda)$ and $S_2^1(\lambda) \subseteq T_2^1(\lambda)$.

Therefore, $S_2^1(\lambda) = T_2^1(\lambda)$ □

Observation 4 *Given* $d(y_1(\bar{\lambda}), a_p) > b_{ip}$ *and* $d(y_1(\bar{\lambda}), b_p) > b_{ip}$ *and* $A_p = Max\{b_{ip} - d(y_2(\bar{\lambda}), a_p), b_{ip} - d(y_2(\bar{\lambda}), a_p)\} \geq 0$

Then, point $z_2(\bar{\lambda})$ *is well defined and only points in* $[y_2(\bar{\lambda}), z_2(\bar{\lambda})]$ *provides nonempty* $S_p^i(\bar{\lambda})$ *where*

$$z_2(\bar{\lambda}) \in S_i \text{ and } d(y_2(\bar{\lambda}), z_2(\bar{\lambda})) = A_p$$

Proof First let us prove that point $z_2(\bar{\lambda})$ is well defined.

$$z_2(\bar{\lambda}) \in S_i \text{ and } d(y_2(\bar{\lambda}), z_2(\bar{\lambda})) = A_p$$

So if A_p is guaranteed to be less than l_i $z_2(\bar{\lambda})$ is well defined.

Suppose $A_p \geq l_i$ then

$$Max\{b_{ip} - d(y_2(\bar{\lambda}), a_p), b_{ip} - d(y_2(\bar{\lambda}), a_p)\} \geq l_i$$

Without loss of generality

$$\begin{aligned} b_{ip} - d(y_2(\bar{\lambda}), a_p) &\geq l_i \\ b_{ip} &\geq l_i + d(y_2(\bar{\lambda}), a_p) \end{aligned}$$

Since $d(y_1(\bar{\lambda}), y_2(\bar{\lambda})) \leq l_i$ for any $\bar{\lambda}$

$$b_{ip} \geq d(y_1(\bar{\lambda}), y_2(\bar{\lambda})) + d(y_2(\bar{\lambda}), a_p) \geq d(y_1(\bar{\lambda}), a_p)$$

Then $b_{ip} - d(y_1(\bar{\lambda}), a_p) \geq 0$ which is a contradiction. So we have proved that $z_2(\bar{\lambda})$ is a well defined point.

Now let us prove that only $[y_2(\bar{\lambda}), z_2(\bar{\lambda})]$ provides nonempty $S_p^i(\bar{\lambda})$

Let $x \in [y_2(\bar{\lambda}), z_2(\bar{\lambda})]$

$$A_p = Max\{b_{ip} - d(y_2(\bar{\lambda}), a_p), b_{ip} - d(y_2(\bar{\lambda}), a_p)\}$$

Without loss of generality suppose

$$A_p = b_{ip} - d(y_2(\bar{\lambda}), a_p) \tag{1}$$

Since $x \in [y_2(\bar{\lambda}), z_2(\bar{\lambda})]$

$$d(x, y_2(\bar{\lambda})) \leq A_p \tag{2}$$

$$d(x, a_p) \leq d(x, y_2(\bar{\lambda})) + d(y_2(\bar{\lambda}), a_p)) \tag{3}$$

From (1),(2) and (3)

$$d(x, a_p) \leq b_{ip}$$

So $a_p \in N(x, b_{ip}) \cap S_p \neq \emptyset$ □

Observations About Algorithm

Orientation Phase

Observation 8 *If $R \neq V'$, $Q \neq \emptyset$.*

Proof This statement appear in step 1, so $R \neq \emptyset$. The proof is by contradiction:

Suppose $Q = \emptyset$ for some R then

$A = R$ and $B = V' \backslash R$ then

We asume that LN is connected (if it were not, we can decompose the problem into parts and solve each one indpendent of the others)

Then for any $i \in A$ and $j \in B$ there should be a path connecting them. Let this path $n_1, n_2, ...n_p$ where $p < m - 1$ and $n_1 = i$ and $n_p = j$.

$n_i \in A$ and $n_{i+1} \in B$ is true at least for one i. Then $\Gamma(n_i) \supseteq n_{i+1}$. So, $\Gamma(A) \supseteq n_{i+1}$ and $n_{i+1} \in \Gamma(A) \cap B$. therefore, $n_{i+1} \in (V' \backslash R) \cap \Gamma(R)$. □

2. Construction Phase

Observation 9 *The node set in DLN can be partition into subsets such that each subset includes only one node with zero out-degree, say node k and all nodes thatappear exactly one path from root to k.*

Proof Let p appears in both of two subset that are determined with the procedure explained in the observation. From r to p there might be different paths but from p there is only one place to go $\Gamma(p)$, if it exist s, which is unique and from there, there is one place to go $\Gamma(\Gamma(p))$ until we reach one of the node with zeroout-degree. So, there cannot be a node that appears in two subsets. □

Bibliography

[1] D. Chhajed and T.J. Lowe. m-median and m-center problems with mutual communication: Solvable special cases. *Operations Research*, 40:S56–S66, 1992.

[2] D. Chhajed and T.J. Lowe. Solving structured multifacility location problems efficiently. *SIAM Journal of Applied Mathematics*, 28:104–115, 1994.

[3] P.M. Dearing, R.L. Francis, and T.J.Lowe. Convex location problems on tree networks. *Operations Research*, 24:628–642, 1976.

[4] M.E. Dyer. Linear time algorithms for two and three variable linear programs. *SIAM Journal of Computing*, 13(1):31–45, 1984.

[5] E. Erkut, R.L. Francis, T.J. Lowe, and A. Tamir. Equivalent mathematical programming formulations of monotonic tree network location problems. *Operations Research*, 37:447–461, 1989.

[6] E. Erkut, R.L. Francis, and A. Tamir. Distance constrained multifacility minimax location problems on tree networks. *Networks*, 22:37–54, 1992.

[7] A.J. Kolen. *Tree Network and Planar Location Theory.* Center for Mathematics and Computer Science, P.O. Box 4079, 10009 AB Amsterdam, the Netherlands, 1986.

[8] N. Megiddo. Combinatorial optimization with rational objective functions. *SIAM Journal of Applied Mathematics*, 4:414–424, 1979.

[9] R.L.Francis, T.J.Lowe, and H.D.Ratliff. Distance constraints for tree network multifacility location problems. *Operations Research*, 26:570–590, 1978.

[10] B.C. Tansel, R.L. Francis, and T.J. Lowe. Binding inequalities for tree network location problems with distance constraints. *Transportation Science*, 14:107–124, 1980.

[11] B.C. Tansel, R.L. Francis, and T.J. Lowe. A biobjective multifacility minimax location problem on a tree network. *Transportation Science*, 16:407–429, 1982.

[12] B.C. Tansel and N.G. Yesilkokcen. Polynomially solvable cases of multifacility distance constraints on cyclic networks. Technical Report IEOR-9311, Department of Industrial Engineering, Bilkent University, Ankara, Turkey, 1993.

[13] B.C. Tansel and N.G. Yesilkokcen. Composite regions of feasibility for certain classes of distance constrained network location problems. *Transportation Science*, 30(2), 1995.

VITA

Hülya Emir was born in Elazığ, Turkey, in 1973. She attended to the Department of Industrial Engineering, Bilkent University, in 1990 and graduated in July 1995. In September 1995, she joined to the Department of Industrial Engineering at Bilkent University as a research assistant. From that time to the present, she worked with Dr. Barbaros Tansel for her graduate study at the same department.