

POLYNOMIALLY SOLVABLE CASES OF
MULTIFACILITY DISTANCE CONSTRAINTS ON
CYCLIC NETWORKS

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

Halil Oğuzhan Yedigöller

July, 1993

T
57.85
-447
1993

POLYNOMIALLY SOLVABLE CASES OF
MULTIFACILITY DISTANCE CONSTRAINTS ON
CYCLIC NETWORKS

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

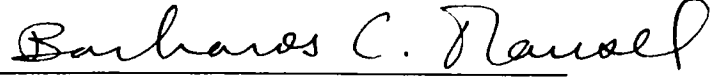
By
Naile Gülcan Yeşilkökçen
July, 1993

Naile Gülcan YEŞİLKÖKÇEN
Tarih: 1993.07.01

57.85
.747
1993

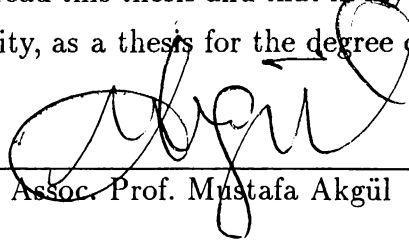
B014211

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 Ç. 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.



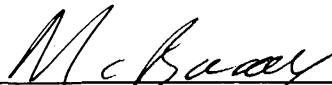
Assoc. Prof. Mustafa Akgül

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. M. Cemal Dinçer

Approved for the Institute of Engineering and Sciences:



Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

POLYNOMIALLY SOLVABLE CASES OF MULTIFACILITY DISTANCE CONSTRAINTS ON CYCLIC NETWORKS

Naile Gülcan Yeşilkökçen
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Barbaros Ç. Tansel
July, 1993

Distance Constraints Problem is to locate one or more 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 \mathcal{NP} -Complete on cyclic networks and polynomially solvable on trees. Although theory for tree networks is well-developed, there is virtually no theory for cyclic networks. In this thesis, we identify a special class of instances for which we develop theory and algorithms that are applicable to any metric space defining the location space. We require that the interaction between new facilities has a tree structure. The method is based on successive applications of EXPANSION and INTERSECTION operations defined on subsets of the location space. Application of this method to general networks yields strongly polynomial algorithms. Finally, we give an algorithm that constructs an ϵ -optimal solution to a related minimax problem.

Key words: Distance Constraints, Network Location, Minimax Problem with Mutual Communication.

ÖZET

GENEL SERİMLERDE ÇOKTESİSLİ UZAKLIK KISITLARI PROBLEMİNİN POLİNOM ZAMANDA ÇÖZÜLEBİLİR DURUMLARI

Naile Gülcan Yeşilkökçen

Endüstri Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Doç. Dr. Barbaros Ç. Tansel

Temmuz, 1993

Uzaklık Kısıtları Problemi, bir serim üzerinde bir yada daha fazla yeni tesisi, yeni tesislerle varolan tesisler arasındaki ve yeni tesis çiftleri arasındaki uzaklıklar belli üst değerleri geçmeyecek biçimde yerleştirme problemidir. Problemin genel serimlerde \mathcal{NP} -Zorluğu ağaç serimlerde ise polinom zamanda çözülebilirliği bilinmektedir. Ağaç serimler için geliştirilmiş temel kuramlar olmasına karşın, genel serimlerde geliştirilmiş hiçbir kuram ve algoritma bulunmamaktadır. Bu tez çalışmasında, biz uzaklık kısıtlarının yeni tesisler arasındaki ilişkilerin ağaç serimi biçiminde olduğu özel bir sınıfını çözen ve yerleşim uzayı olarak alınan herhangi bir metrik uzaya uygulanabilen bir yöntem sunuyoruz. Yöntem, yerleşim uzayının alt kümelerinde tanımlanmış GENİŞLETME ve KESİŞTİRME işlemlerini temel almaktadır. Bu yöntemin genel serimlerde uygulaması polinom zamanlı algoritmalar verir. Son olarak, ilgili bir enküçük-enbüyük problemine ϵ -eniye çözüm üreten bir algoritma veriyoruz.

Anahtar Kelimeler: Uzaklık Kısıtları, Serim Yerleşimi, Enküçük-enbüyük Problemi.

To my father and mother

ACKNOWLEDGEMENT

I am indebted to Associate Professor Barbaros Tansel for his invaluable guidance, encouragement and above all, for the enthusiasm which he inspired on me during this study.

I am grateful to every individual of my family whose love and support have always been with me in spite of the distances that kept us apart.

I would like to extend my thanks to Ogan Ocalı for his love, patience and encouragement especially at times of despair and hardship. My special thanks are due to my former and present officemates Ash Sencer Erdem, Pınar Keskinocak, Mehmet Özkan and Sibel Salman for their interest and encouragement in every step of this study.

Contents

1	INTRODUCTION	1
1.1	Introduction and Overview	1
1.2	Networks, Embedding, and Distance	4
1.3	Distance Constraints and m -Center Problem with Mutual Communication	6
2	EXISTING THEORY FOR TREE NETWORKS	9
2.1	Convexity on Tree Networks	9
2.1.1	Convexity of Distance	11
2.1.2	Implications of Convexity of Distance	13
2.2	Distance Constraints on Tree Networks	14
2.2.1	Single Facility Case	14
2.2.2	Multifacility Case	16
2.3	Minimax Problem with Mutual Communication	21
2.3.1	Distance Constrained $PMMC$ on Tree Networks	22

2.4	Biobjective Multifacility Minimax and Vector Minimization . . .	24
2.5	Other Approaches	26
2.6	DC is \mathcal{NP} -Complete on General Networks	27
3	DISTANCE CONSTRAINTS AND EXPAND/INTERSECT METHOD IN GENERAL METRIC SPACES	30
3.1	EXPAND/INTERSECT Method for General Metrics	32
3.2	Analysis of $SEIP$	37
3.3	Regions of Feasibility	39
3.4	Other Set Constraints and Distance Constraints	44
3.5	Cyclic LN_B	45
4	COMPUTATIONAL METHODS FOR NETWORKS	47
4.1	Single Facility Case	48
4.1.1	Construction of the Feasible set for $DC(1)$	50
4.2	Multifacility Case	55
4.2.1	Example	56
4.2.2	EXPAND/INTERSECT Operation on Networks . . .	60
5	A MINIMAX EXTENSION	69
5.1	ϵ -Optimal Solution to $PMMC$	71
5.2	Solving $PMMC$ on Networks	73

CONTENTS ix

6 SUMMARY AND CONCLUSION 76

List of Figures

1.1	Illustration of the 1-1 Mapping from $[0, 1]$ onto an Edge	5
1.2	Possible Shapes of $d(x, y)$	6
2.1	Example of a Linkage Network (LN)	17
2.2	Failure of Sufficiency of SC on Cyclic Networks	18
2.3	Failure of Sufficiency of SC for Node Restricted DC	19
2.4	Example of the Construction of G' from G	29
3.1	Example of LN and Related LN_B	31
3.2	Expansion of S by r (in the Plane with Euclidean Distance) . . .	32
3.3	Example to Illustrate $SEIP$	36
3.4	New Root in the Second Application.	43
3.5	Example of Failure for Cyclic LN_B	46
4.1	Illustration of $N_e^i \equiv N(v_i, c_i) \cap e$ for Various Possibilities	50
4.2	Example to Illustrate Segments on a Given Edge	53
4.3	(a) Example for Multifacility Case	57

LIST OF FIGURES

xi

4.3 (b) Example Continued	58
4.3 (c) Example Continued	59
4.4 Illustration of Expansion Steps for S_e	63
4.5 Illustration of ERI	65

Chapter 1

INTRODUCTION

1.1 Introduction and Overview

A location problem, in its most general form is to determine locations of several new facilities in a given location space so as to provide goods and services to existing facilities and/or to each other under one or multiple criteria, and possibly subject to a set of constraints. The typical measure for the quality of service is some function of distances between facilities.

Depending on the location space, distance function, objective criteria, number of new facilities to be located, type of interaction between facilities, presence of uncertainty in several parameters of the problem, and possibly some other factors, different types of location problems can be characterized. In a recent survey of Brandeau and Chiu [1], over 50 representative problems on location theory are provided with their formulations and relations to each other. There is also an extensive bibliography of location literature by Domschke and Drexl [5].

Network location problems occur when the location space is a network. A road network, a river network, an air transport network or a network of shipping lanes may constitute the network of interest. In most of the network

location problems, the new facilities are idealized as points and may be located anywhere on the network. Constraints may be imposed on the problem so that the new facilities are within specified distances of existing facilities and other new facilities as well. A comprehensive survey by Tansel, Francis, and Lowe [17, 18] provides a review of network location problems with specific emphasis on work which deals directly with the network of interest and exploits the structure of the network. In part I of the survey, the p -center and p -median problems are considered and in part II, several network location problems including multifacility minimax and minisum problems with mutual communication, distance constraints, multifacility, and path location problems on tree networks are reviewed. A rather recent survey by Labbé, Peeters, and Thisse [11] discusses median and center problems, economic models of location as well as the discrete location models of network location problems.

In this thesis, we work on *Distance Constraints* which requires locating several new facilities on a network so that they are within specified distances of existing facilities and within specified distances of one another. Although the theory for tree networks is well developed and efficient algorithms exist, there is virtually no theory and algorithms on general networks regarding the distance constraints. Here, we go one step forward and provide algorithms of polynomial complexity which solves distance constraints problem on general networks for a special structure of the interaction between new facilities. We also consider the *Multifacility Minimax Problem with Mutual Communication* in the context of distance constraints. We now provide an overview of the thesis.

In the remainder of Chapter 1, we first define our terminology and give formal definitions of *embedded networks* and *distances* with their properties. Next we define *Distance Constraints Problem* on networks. We also introduce *Multifacility Minimax Problem with Mutual Communication* which is also called *m-Center Problem with Mutual Communication* in the literature. We give an equivalent formulation of the problem in terms of distance constraints.

In Chapter 2, we provide a survey of literature on Distance Constraints

and related minimax problem. Since almost all of the related literature we are aware of deals with these problems on tree networks, we provide an overview of existing theory and algorithms on tree networks. We also cite some of the characteristics of those problems on trees which may provide a partial answer to the question of why tree networks are more tractable than general (cyclic) networks.

Our main results are provided in Chapter 3. Here, we deviate from the network context and approach the problem from a different perspective. We provide a new method to solve distance constraints defined in any metric space. The basic idea is to apply *EXPANSION* and *INTERSECTION* operations defined on subsets of the location space. For a special structure of interaction between new facilities, this approach leads to exact algorithms for determining the consistency or inconsistency of distance constraints and for finding a feasible solution if it exists. We also consider a broader problem of characterizing the set of all feasible solutions for all new facilities, by defining a *region of feasibility* for each new facility. This problem has not been addressed in the literature except for the single facility case. The algorithms we provide here are applicable to any metric space with differences arising only in algorithmic details.

In Chapter 4, we apply our approach to network spaces and construct strongly polynomial algorithms to determine the consistency or inconsistency of distance constraints and to find a feasible location vector if it exists. We also construct the regions of feasibility.

In Chapter 5, we provide an equivalent formulation of Multifacility Minimax Problem with Mutual Communication in terms of distance constraints with parametric distance bounds. We give a polynomial algorithm to construct an ϵ – *optimal* solution for fixed ϵ that relies on a bisection search on the optimal objective value. Each trial in the search requires solving a set of distance constraints with fixed distance bounds.

In Chapter 6, we give a summary of the report and we pose unresolved questions on the problems discussed. We also give directions for future research.

1.2 Networks, Embedding, and Distance

Before defining the problems in consideration we develop our terminology. Throughout the thesis we may also give other definitions as necessary. Note that, we use graph, node and arc interchangeably with network, vertex and edge respectively. Basic definitions of networks, embedding and distance provided here are due to Dearing, Francis and Lowe [4].

Given a set of n distinct vertices, $\mathcal{V} = \{\nu_1, \dots, \nu_n\}$ which is a subset of a given infinite set \mathcal{F} , and a set of undirected edges consisting of unordered pairs (ν_i, ν_j) of distinct vertices, an *undirected network* or *graph* \mathcal{G} is a pair $(\mathcal{V}, \mathcal{E})$ with a length function that assigns a positive real number l_{ij} to each edge (ν_i, ν_j) . An *embedding* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in some space \mathcal{S} is a set G in \mathcal{S} such that $G = \cup\{[v_i, v_j] : (\nu_i, \nu_j) \in \mathcal{E}\}$ where $[v_i, v_j]$ is the image of the unit interval under a one-to-one mapping $T_{ij} : [0, 1] \rightarrow \mathcal{S}$, with $T_{ij}(0) = v_i$ and $T_{ij}(1) = v_j$ with v_1, \dots, v_n being distinct points in \mathcal{S} representing ν_1, \dots, ν_n , respectively. Define $V = \{v_1, \dots, v_n\}$ and $E = \{[v_i, v_j] : (\nu_i, \nu_j) \in \mathcal{E}\}$. V is the vertex set of G and E is the edge set of G .

For a point x in $[v_i, v_j]$ there exists a unique λ in $[0, 1]$ such that $T_{ij}(\lambda) = x$. If we denote the inverse of T_{ij} by Λ_{ij} then $\Lambda_{ij}(x) = \lambda$ if and only if $T_{ij}(\lambda) = x$. A point x in $[v_i, v_j]$ defines two subarcs $[v_i, x] = \{y : T_{ij}(\alpha) = y \text{ for some } \alpha \in [0, \Lambda_{ij}(x)]\}$ and $[x, v_j] = \{y : T_{ij}(\alpha) = y \text{ for some } \alpha \in [\Lambda_{ij}(x), 1]\}$. If $x = T_{ij}(\lambda)$, then the lengths of $[v_i, x]$ and $[x, v_j]$ are defined as λl_{ij} and $(1 - \lambda)l_{ij}$ respectively. Figure 1.1 illustrates the idea.

A subset Q of G is *connected* if there do not exist two disjoint subsets A and B of G such that A intersects Q , B intersects Q , and $Q \subseteq A \cup B$. A *path* joining $x_1 \in G$ and $x_2 \in G$ is a minimal connected subset of G containing x_1 and x_2 . The *length of a path* is equal to the sum of the lengths of all edges or subedges on this path. Network G is *connected if and only if* every pair of vertices is joined by a path. A network whose any two distinct points are joined by a single path is called a *tree*, denoted T .

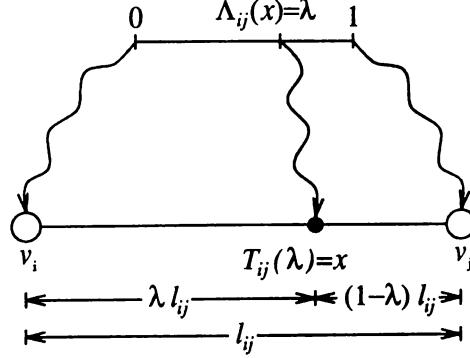


Figure 1.1: Illustration of the 1-1 Mapping from $[0, 1]$ onto an Edge

For any two points $x, y \in G$, define $d(x, y)$ to be the length of a shortest path, denoted $P(x, y)$, connecting x and y . $d(x, y)$ is appropriately called the distance between x and y since the function $d(\cdot, \cdot)$ has the well-known properties of nonnegativity, symmetry and triangle inequality of a metric. That is, $\forall x, y \in G$:

- (1) $d(x, y) \geq 0, d(x, x) = 0$ (Nonnegativity)
- (2) $d(x, y) = d(y, x)$ (Symmetry)
- (3) $d(x, y) \leq d(x, z) + d(z, y) \quad \forall z \in G$ (Triangle Inequality)

Hence, G together with distance d constitutes a well-defined metric space. We should also note that from the compactness of an embedded edge, G is compact and $d(x, y)$ as a function of x and for fixed y is continuous.

The distance between a given point of G and a variable point along an edge of G , has the following property :

Theorem 1.1 *Let y be a fixed point on G . Let $\lambda \in [0, 1]$ with $T_{pq}(\lambda) = x \in [v_p, v_q] \in E$. Then $d(T_{pq}(\lambda), y)$ is a function of λ which is continuous, piecewise linear concave with at most two pieces.*

Proof : From the definition of distance we have $d(x, y) = d(T_{pq}(\lambda), y) = \min\{\lambda l_{pq} + d(v_p, y), (1 - \lambda)l_{pq} + d(v_q, y)\}$. Let $\alpha = \lambda l_{pq}$ and define $g_p(\alpha) =$

$\alpha + d(v_p, y)$, $g_q(\alpha) = (l_{pq} - \alpha) + d(v_q, y)$, and $g(\alpha) = d(x, y) = \min\{g_p(\alpha), g_q(\alpha)\}$. $g_p(\alpha)$ is a linear function with slope +1 and $g_q(\alpha)$ is a linear function with slope -1. This implies that $g(\alpha) = d(x, y)$ is a piecewise linear concave function with at most two pieces. \square

Figure 1.2 illustrates the possible shapes of $g(\alpha) = d(x, y)$ with x being a variable point on $[v_p, v_q] \in E$ and y being a fixed point on G .

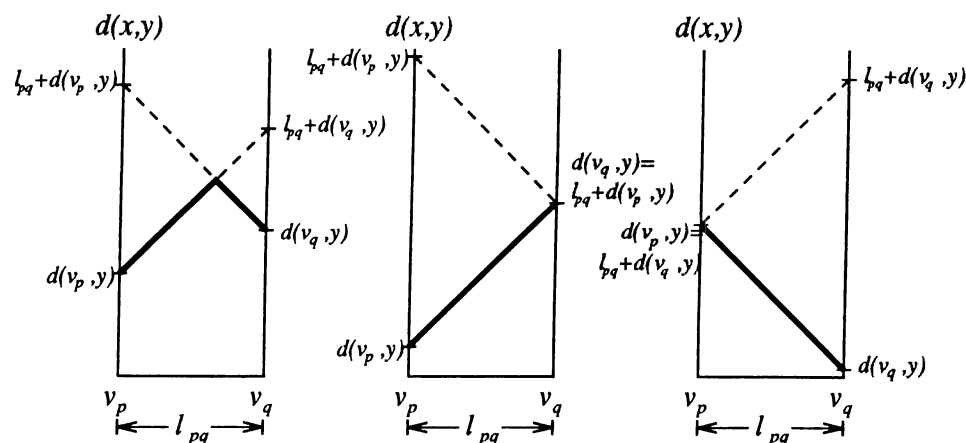


Figure 1.2: Possible Shapes of $d(x, y)$

1.3 Distance Constraints and m -Center Problem with Mutual Communication

Distance constraints arise when new facilities should not be located too far from existing facilities as well as from other new facilities. For example, consider emergency service units such as fire stations, ambulances, and police patrol vehicles. In order to avoid fatalities, damage to human life and excessive property losses, it may be appropriate to require service units to be within a specified driving time of any point in the region they serve. Distance constraints ensure the service units reach the emergencies within specified critical response times. In military scenarios, response units may be required not to be too far from

their supply bases and from each other in order to reinforce other units if a need arises.

Another important factor that justifies the consideration of distance constraints is that the analysis of these constraints facilitates the analysis of minimax type location problems since some of these problems can be reformulated in the form of distance constraints with parametric bounds.

Let G be the network of interest. Existing facilities are defined by the nodes of the network and new facilities are to be located anywhere on the network satisfying the given distance constraints. We denote by G^m the m -fold Cartesian product of G by itself and denote any location vector in G^m by $X = (x_1, \dots, x_m)$.

The *Single Facility Distance Constraints Problem*, abbreviated as $DC(1)$, is to find a point x in G , if it exists, such that

$$d(x, v_i) \leq c_i, \quad i = 1, \dots, n$$

where c_i are known nonnegative constants.

The *Multifacility Distance Constraints*, $DC(m)$, is to find a location vector $X = (x_1, \dots, x_m) \in G^m$, if it exists, such that

$$\begin{aligned} d(x_j, v_i) &\leq c_{ji}, \quad 1 \leq j \leq m, \quad 1 \leq i \leq n \\ d(x_j, x_k) &\leq b_{jk}, \quad 1 \leq j < k \leq m \end{aligned}$$

where c_{ji} , b_{jk} are known nonnegative constants, some of which may be infinity. Clearly, if any one of the bounds c_{ji} , b_{jk} is ∞ then that particular constraint has no effect on the solution of distance constraints. Hence we may omit such constraints from the formulation. Let $I_C = \{(j, i) : c_{ji} < \infty\}$ and $I_B = \{(j, k) : b_{jk} < \infty\}$. Note that I_C is a subset of $\{1, \dots, m\} \times \{1, \dots, n\}$ and I_B is a subset of $\{(j, k) : 1 \leq j < k \leq m\}$. Then we rewrite the distance constraints (DC) as follows:

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

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

This time c_{ji} , b_{jk} are known nonnegative finite constants. Note that $DC(m)$ includes $DC(1)$ as a special case.

From now on we will not distinguish between single and multifacility distance constraints, we will only use DC to indicate the distance constraints in general. DC is said to be *consistent* if there exists a solution vector $X = (x_1, \dots, x_m) \in G^m$ such that (DC.1) and (DC.2) are satisfied. Otherwise DC is *inconsistent*. Determining the consistency or inconsistency of DC is known to be \mathcal{NP} -Complete for arbitrary networks [12].

A problem related to DC is the Multifacility Minimax Problem with Mutual Communication. The original form of the unconstrained problem is as follows:

Given nonnegative weights w_{ji} , v_{jk} for all index pairs

$$\begin{aligned} \text{Min } & f(X) \\ \text{where } f(X) &= \max\{f_1(X), f_2(X)\} \\ \text{with } f_1(X) &= \max_{1 \leq j \leq m, 1 \leq i \leq n} w_{ji}d(x_j, v_i) \\ f_2(X) &= \max_{1 \leq j < k \leq m} v_{jk}d(x_j, x_k) \\ x_1, \dots, x_m &\in G \end{aligned}$$

This problem can be reformulated in terms of distance constraints as follows:

$$\begin{aligned} \text{Min } & z \\ \text{subject to } & \\ & d(x_j, v_i) \leq z/w_{ji} \quad , \quad (j, i) \in I_C \\ & d(x_j, x_k) \leq z/v_{jk} \quad , \quad (j, k) \in I_B \\ & x_1, \dots, x_m \in G \end{aligned}$$

where w_{ji} , v_{jk} are the positive constants for the given index pairs. The problem is to find points $x_1, \dots, x_m \in G$ for which maximum distance between specified pairs of facilities is as small as possible.

Chapter 2

EXISTING THEORY FOR TREE NETWORKS

2.1 Convexity on Tree Networks

Most of the network location problems are well-solved when the network is a tree whereas a substantial computational effort is required when the network is cyclic. Part of the answer to the question why tree networks are more tractable than cyclic ones comes from the convexity results on trees. Yet convexity does not provide a full answer because some of the well-known nonconvex functions like the *p-center* and *p-median* problems are efficiently solved on tree networks whereas no polynomial order algorithms are known to exist to solve them on arbitrary networks.

The convexity discussion presented here is due to Dearing, Francis and Lowe [4]. The convexity results given in the paper have important applications for the four problems we describe next. First let us specify the total cost function g and minimax function h defined on G^m . For $X = (x_1, \dots, x_m) \in G^m$

$$\begin{aligned}g(X) &= \sum_{j=1}^m \sum_{i=1}^n w_{ji} d(x_j, v_i) + \sum_{1 \leq j < k \leq m} v_{jk} d(x_j, x_k) \\h(X) &= \max \{ \max_{i \leq j \leq m, 1 \leq i \leq n} w_{ji} d(x_j, v_i), \max_{1 \leq j < k \leq m} v_{jk} d(x_j, x_k) \}\end{aligned}$$

where w_{ji} and v_{jk} are known nonnegative constants.

Upper bounds on the distances between new and existing facilities as well as between pairs of new facilities can be imposed. Those bounds are named as distance constraints (DC) and are introduced in Chapter 1. Let the feasible set of solutions to DC be

$$S = \{X \in G^m : X \text{ satisfies } DC.1 \text{ and } DC.2\}.$$

Clearly if $S \neq \emptyset$ then DC is consistent, otherwise DC is inconsistent. We have two unconstrained multifacility problems:

$$\min_{X \in G^m} g(X) \quad (P1)$$

$$\min_{X \in G^m} h(X) \quad (P2)$$

($P1$) is called the Minisum (or m -Median) Problem with Mutual Communication and ($P2$) is called the Minimax (or m -Center) Problem with Mutual Communication. The constrained versions of these problems are as follows:

$$\min_{X \in S} g(X) \quad (C - P1)$$

$$\min_{X \in S} h(X) \quad (C - P2)$$

where S is the feasible set of DC defined above.

Given nonnegative weights w_{ji} , v_{jk} , the functions g and h are convex, also the feasible set S of solutions to DC is a convex subset of G^m in a well-defined sense (explained in the next subsection) *if and only if* the underlying network is a tree. Hence the problems $P1$, $P2$, $C - P1$, $C - P2$ are convex optimization problems on convex feasible regions for all data choices *if and only if* the network is a tree. This implies the following important result for tree networks.

Result 2.1 *A local minimum to $P1$, $P2$, $C - P1$, $C - P2$ is a global minimum. \square*

2.1.1 Convexity of Distance

In order to address the question of convexity of functions g and h , the following distance functions are considered first.

$$\begin{aligned} f_1(x : a) &= d(x, a) \quad , \quad x \in G, a \text{ is a fixed point in } G \\ f_2(x_1, x_2) &= d(x_1, x_2) \quad , \quad (x_1, x_2) \in G^2 \end{aligned}$$

Let us first define the basic notation and give the definitions of convex sets and convex functions on a network. For details refer to [4]. For every $Y = (y_1, \dots, y_m)$, $Z = (z_1, \dots, z_m) \in G^m$ and for all $0 \leq \lambda \leq 1$ we define $L_\lambda(X) = \{X \in G^m : d(y_i, x_i) + d(x_i, z_i) = d(y_i, z_i), d(x_i, z_i) = \lambda d(y_i, z_i), 1 \leq i \leq m\}$. Then the line segment in G^m is defined as $L(Y, Z) = \bigcup_{0 \leq \lambda \leq 1} L_\lambda(Y, Z)$. For $m = 1$, $L(y, z)$ is simply the union of all shortest paths connecting y and z in G . Any subset S of G^m is said to be *convex* if for every $Y, Z \in S$ $L(Y, Z) \subseteq S$. If S is a convex subset of G^m and f is a real valued function with domain G^m and range R_+ , then f is said to be *convex on S* if given any $Y, Z \in S$, $f(X) \leq \lambda f(Y) + (1 - \lambda)f(Z)$ for every $X \in L_\lambda(Y, Z)$ and every $0 \leq \lambda \leq 1$.

In the proof of convexity of the functions f_1 and f_2 the following lemma, which we provide without proof, is used. The reader is referred to [4] for the proof.

Lemma 2.1 *Let w, y, z be any three points of a tree. Then, for all points x in $L(y, z)$, $x \in L(w, y) \cup L(w, z)$. \square*

Now, we have the following lemma which shows that f_1 is a convex function on trees.

Lemma 2.2 *Let a be any fixed point in G . Then the function $f_1(x : a)$ is convex on G if and only if G is a tree. \square*

Proof : Assume $f_1(x : a)$ is convex. Suppose G is not a tree. Then there exists a cycle C of shortest length in G . Let l be the length of the cycle.

We can choose points x, y, z and a in C such that $d(a, x) = d(y, z) = \frac{1}{2}$, $d(a, y) = d(x, y) = d(z, a) = d(x, z) = \frac{1}{4}$. Then $f_1(x : a) = \frac{1}{2} > \frac{1}{2}f_1(y : a) + \frac{1}{2}f_1(z : a) = \frac{1}{4}$ which provides a contradiction to the assumption that $f_1(x : a)$ is convex. Hence G is a tree.

Suppose the network is a tree. Then we have to show that $\forall x \in L_\lambda(y, z)$ and $0 \leq \lambda \leq 1$

$$f_1(x : a) \leq \lambda f_1(y : a) + (1 - \lambda)f_1(z : a) \quad (2.1)$$

or equivalently,

$$d(x, a) \leq \lambda d(y, a) + (1 - \lambda)d(z, a). \quad (2.2)$$

For $\lambda = 0$ and $\lambda = 1$, $x = z$ and $x = y$ so the inequality holds. Assume now $0 < \lambda < 1$. From the definition of $L_\lambda(y, z)$ we have

$$d(y, z) = d(y, x) + d(x, z) \quad (2.3)$$

and

$$\lambda = \frac{d(x, z)}{d(y, z)} \quad (1 - \lambda) = \frac{d(x, y)}{d(y, z)}. \quad (2.4)$$

So we can rewrite the inequality (2.2) as follows:

$$d(x, a)d(y, z) \leq d(x, z)d(y, a) + d(x, y)d(z, a). \quad (2.5)$$

$x \in L(y, z)$ and Lemma 2.1 implies $x \in L(y, a) \cup L(z, a)$. Without loss of generality we can assume $x \in L(z, a)$. Then

$$d(a, x) + d(x, z) = d(a, z). \quad (2.6)$$

Then we have

$$\begin{aligned} d(x, a)d(y, z) &= d(x, a)[d(y, x) + d(x, z)] \\ &= d(x, a)d(y, x) + d(x, a)d(x, z) \\ &\leq d(x, a)d(y, x) + [d(x, a) + d(y, a)]d(x, z) \\ &= [d(a, z) - d(x, z)]d(y, x) + [d(x, y) + d(y, a)]d(x, z) \\ &= d(a, z)d(y, x) + d(y, a)d(x, z). \end{aligned}$$

The first equality comes from (2.3), the inequality is a consequence of the triangle inequality and the next equality is a consequence of (2.6). Hence we have proven (2.5) which is equivalent to (2.2), so $f_1(x : a)$ is convex. \square

We now give the convexity result for f_2 .

Lemma 2.3 f_2 is convex on G^2 if and only if G is a tree. □

The necessity simply follows from the analysis, if $f_2(x_1, x_2)$ is convex on G^2 then for any fixed x_2 , $f_2(x_1 : x_2)$ is convex and hence $f_1(x : a)$ is convex which implies G is a tree by Lemma 2.2. The sufficiency part is proved considering the two cases and uses Lemma 2.1 and some properties of convex sets. The proof is technical but not too difficult, so we omit it and refer the reader to [4] for details.

2.1.2 Implications of Convexity of Distance

Having proved the convexity of distance on tree networks, it is easy to show that the functions h and g are convex. g is a function of sum of distances multiplied by positive constants. Multiplication of a convex function with a positive constant is also convex. So g is convex on trees. Similarly h is the maximum of a set of convex functions. It is also well known that max of a set of convex functions is also convex. Hence, h is convex on trees. This analysis imply that $(P1)$ and $(P2)$ are convex optimization problems on trees. Hence a local minimum to $(P1)$ and $(P2)$ is a global minimum.

With convexity of distance functions from Lemmas 2.2 and 2.3, and with the property that intersection of convex sets is convex and level sets of convex functions are also convex, we can conclude that the feasible set S of DC is also convex. Convexity of the set S and convexity of functions g and h imply that the constrained problems $(C - P1)$ and $(C - P2)$ are convex optimization problems on convex sets when the network is a tree. Hence a local optimum to these problems is a global optimum.

2.2 Distance Constraints on Tree Networks

Distance constraints are first defined by Dearing, Francis, and Lowe [4] in their work which is discussed in the previous section. They establish that the feasible set of distance constraints is convex *if and only if* the underlying network is a tree. Francis, Lowe, and Ratliff [8] developed the theory for distance constraints on tree networks. They give necessary and sufficient conditions for DC to be consistent, and also they provide algorithms that find a feasible solution whenever one exists.

2.2.1 Single Facility Case

Recall that the distance constraints for a single new facility is as follows: Given $c_i \geq 0 \forall i$, find x in T such that

$$d(x, v_i) \leq c_i, \quad i \in I \quad (DC(1))$$

Francis, Lowe, and Ratliff [8] show that there exists a feasible solution $x \in T$ to distance constraints *if and only if* the conditions $d(v_i, v_j) \leq c_i + c_j$ are satisfied $1 \leq i < j \leq n$. If we define $N(a, r) = \{x \in T : d(x, a) \leq r\}$ to be the *neighborhood* around a *center* a with *radius* r , then we can have an equivalent formulation of distance constraints in terms of neighborhoods $N(v_i, c_i)$ centered at v_i with radius c_i . To satisfy the i -th constraints of $DC(1)$, a feasible solution $x \in T$ must be in the neighborhood defined by that constraint, that is $x \in N(v_i, c_i)$. So x must be in every neighborhood defined by all of the constraints. This implies that x must be in the intersection of all neighborhoods. Therefore the intersection of all neighborhoods define the set of all feasible solutions. Let $S = \bigcap_{i=1}^n N(v_i, c_i)$. If $S = \emptyset$ then DC is inconsistent else it is consistent. We should note here that the above results also hold for arbitrary networks and in general for arbitrary metric spaces. However, other results obtained for trees are not valid for arbitrary networks.

A neighborhood defined on a tree network is nothing but a subtree. A result

due to Horn [10] states that intersection of subtrees of a tree is nonempty *if and only if* every pairwise intersection is nonempty. From this result it follows that the intersection of two neighborhoods $N(v_i, c_i)$ and $N(v_j, c_j)$ defined on a tree is nonempty *if and only if* the distance between two vertices (centers) of these neighborhoods is less than the sum of their radii, that is *if and only if* the condition $d(v_i, v_j) \leq c_i + c_j$ is satisfied. Based on these properties, necessary and sufficient conditions follow:

Property 2.1 *DC(1) is consistent if and only if $d(v_i, v_j) \leq c_i + c_j$, $1 \leq i < j \leq n$.* \square

This property provides a basis for the *Sequential Intersection Procedure (SIP)* developed in [8] to construct the feasible set of solutions to *DC*, by intersecting neighborhoods pair at a time in an arbitrary order.

The intersection of two neighborhoods $N(a, r)$ and $N(a', r')$, whenever nonempty, is a neighborhood $N(\bar{a}, \bar{r})$ with unique center \bar{a} and radius \bar{r} . $N(\bar{a}, \bar{r})$ is termed the *Composite Neighborhood Representation* of $N(a, r)$ and $N(a', r')$. Three cases may occur regarding the intersection :

- Case 1 : $d(a, a') + r \leq r'$ (i.e. $N(a, r) \subseteq N(a', r')$)
Case 2 : $d(a, a') + r' \leq r$ (i.e. $N(a', r') \subseteq N(a, r)$)
Case 3 : otherwise

Depending on the cases described above, the computation of \bar{a} and \bar{r} is as follows:

$$\bar{a} = \begin{cases} a & \text{if Case 1 occurs} \\ a' & \text{if Case 2 occurs} \\ \text{the point in } L(a, a') \text{ for which} & \text{if Case 3 occurs} \\ d(a, \bar{a}) = \frac{1}{2}[d(a, a') + r - r'] & \end{cases}$$

$$\bar{r} = \begin{cases} r & \text{if Case 1 occurs} \\ r' & \text{if Case 2 occurs} \\ \frac{1}{2}[r + r' - d(a, a')] & \text{if Case 3 occurs} \end{cases}$$

Then we have the following property.

Property 2.2 *Given \bar{a} , \bar{r} defined above, $N(a, r) \cap N(a', r') = N(\bar{a}, \bar{r})$ whenever $\bar{r} \geq 0$.* \square

If $\bar{r} < 0$ we define $N(\bar{a}, \bar{r}) = \emptyset$. Starting with arbitrary two neighborhoods, in each step *SIP* produces a composite neighborhood representation of two neighborhoods, discards the two old ones, inserts the new representation, and repeats the procedure with the new list of neighborhoods. The list size is decreased by one at each step. Hence, at the final step *SIP* produces a composite neighborhood representation of all neighborhoods $N(v_i, c_i)$, $i = 1, \dots, n$ in $n - 1$ steps. Therefore the computational complexity of *SIP* is $O(n)$. From the analysis of *SIP*, we have the following property.

Property 2.3 *If $\bigcap_{i=1}^n N(v_i, c_i) \neq \emptyset$ then there exists a neighborhood $N(a, r)$ with center a and radius r such that $N(a, r) = \bigcap_{i=1}^n N(v_i, c_i)$.* \square

Analysis of the computation of \bar{a} and \bar{r} for composite neighborhood representation leads to the observation that \bar{a} does not change by reducing each neighborhood by ϵ , and \bar{r} is reduced by ϵ . From this observation next property follows:

Property 2.4 *If $N(a, r) = \bigcap_{i=1}^n N(v_i, c_i)$ then $N(a, r - \epsilon) = \bigcap_{i=1}^n N(v_i, c_i - \epsilon)$ for $\epsilon \leq r$.*

Property 2.4 enables a sensitivity analysis on the bounds of distances.

2.2.2 Multifacility Case

Let us first restate the *DC* for multiple new facilities:

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

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

The necessary and sufficient conditions for consistency of DC are provided in [8] in terms of $n(n-1)/2$ inequalities named *Separation Conditions (SC)*. In order to define these conditions, an auxiliary network (which we call *Linkage Network (LN)* throughout the thesis) is constructed by using the sets I_C and I_B . For each existing facility i , there exists a node E_i , $1 \leq i \leq n$, associated with this existing facility, and for each new facility j there exists a node N_j , $1 \leq j \leq m$, associated with this new facility. Two nodes N_j and E_i are joined by an undirected arc of length c_{ji} for each $(j, i) \in I_C$, and two nodes N_j, N_k are joined by an undirected arc of length b_{jk} for each $(j, k) \in I_B$. The node set of LN is $\{N_1, \dots, N_m\} \cup \{E_1, \dots, E_n\}$ and arc set of LN is $\{(N_j, E_i) : (j, i) \in I_C\} \cup \{(N_j, N_k) : (j, k) \in I_B\}$. Figure 2.1 below provides an example for the network LN .

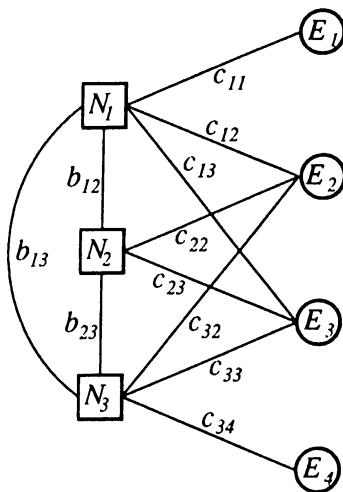


Figure 2.1: Example of a Linkage Network (LN)

It is reasonable to assume that LN is connected, otherwise each component of LN can be handled independent of other components. That is, DC decomposes into as many independent subproblems as there are components in LN .

Let $\mathcal{L}(E_p, E_q)$ be the length of a shortest path between E_p and E_q on LN . Then the following theorem of [8] is the most fundamental result regarding the consistency of DC .

Theorem 2.1 (*Separation Conditions (SC)*) DC is consistent if and only if $d(v_i, v_j) \leq \mathcal{L}(E_i, E_j)$ for $1 \leq i < j \leq n$. \square

We note here that for single new facility SC are just those conditions $d(v_i, v_j) \leq c_i + c_j$, $1 \leq i < j \leq n$ which we have stated in the context of neighborhood intersections. The necessity part of the theorem is proved by a repeated application of the triangle inequality. Hence it is applicable to general networks. The sufficiency of SC is proved by means of an algorithm called *Sequential Location Procedure (SLP)* which constructs a feasible solution if one exists or concludes that DC is inconsistent. This implication is not extendible to general networks. Figure 2.2 below provides an example to failure of sufficiency of SC even on a simple cycle of 3 nodes. Although the separation conditions are satisfied DC is inconsistent.

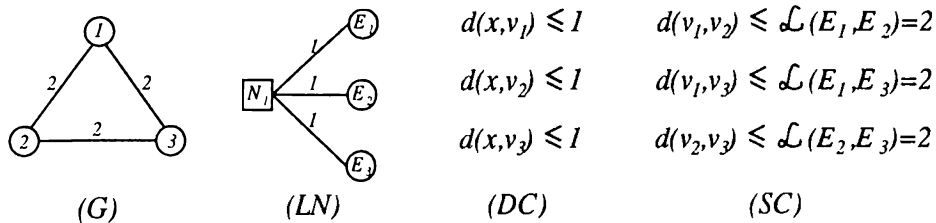


Figure 2.2: Failure of Sufficiency of SC on Cyclic Networks

Sufficiency of SC also fails when new facilities are restricted to nodes of the network. A simple example is provided in Figure 2.3 below.

The following string model represents how SLP works. Imagine that the tree is represented by appropriately inscribing straight line segments on a board such that each line segment represents an arc. At each vertex v_i strings of length c_{ji} are fastened for each new facility such that $(j, i) \in I_C$. At each step a tip vertex is chosen and all the strings fastened at that vertex are pulled tight on

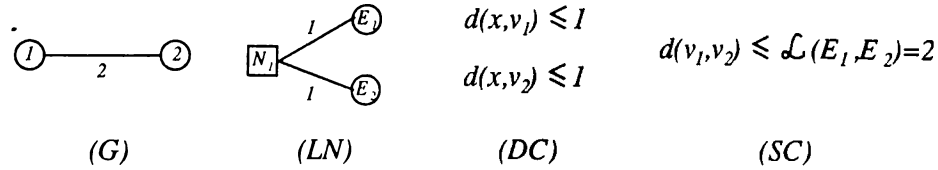


Figure 2.3: Failure of Sufficiency of SC for Node Restricted DC

the edge towards the adjacent vertex. If all strings reach that vertex then they are engaged there and the tip vertex, just processed, together with its incident edge is removed from the board. The procedure is repeated with the resulting tree. If at some step, some of the strings do not reach the adjacent vertex, the shortest one is selected. The point on the edge where the endpoint of the shortest string reaches is the location of the new facility it is associated with. Those strings that are pulled from the chosen tip are engaged at that point. Then all the strings left at the nonprocessed nodes that are related to the newly located new facility are pulled towards the point where the new facility is just located. This provides a feasibility check for the present location of that new facility. If all the strings reach then the location is feasible and all the strings related to this new facility are removed from the board. However, if at least one of the strings do not reach then the algorithm terminates *infeasible*. New strings are engaged to the point of location of this new facility for those unplaced new facilities that are related to the current new facility. The algorithm continues in this manner treating each placed new facility like an existing facility, until either all new facilities are placed or the the current tree reduces to a point. In the latter case all unplaced new facilities are located at that point.

If the separation conditions hold SLP described above always finds a feasible location vector in $O(m(m+n))$ time and it is also proven in [8] that SLP is a best algorithm for determining consistency of DC .

Tansel, Francis and Lowe [15] provide extensions of the results obtained in [8]. They characterize uniquely located new facilities by means of binding separation conditions.

A separation condition holding at equality is called *binding*. Let $P(E_i, E_j)$ be a path in LN . $P(E_i, E_j)$ is said to be a *tight path* if $\mathcal{L}(E_i, E_j) = d(v_i, v_j)$. We say new facility k is *uniquely* located if its location point is the same in all feasible solutions to DC . The following property relates uniquely located new facilities and tight paths to each other.

Property 2.5 *New facility k is uniquely located if and only if N_k is on at least one tight path in LN .* \square

An immediate corollary to this property suggests that DC has a unique feasible solution *if and only if* every new facility node lies on at least one tight path.

Another result, enabling to locate new facilities on tight paths without using SLP is provided with the next property.

Property 2.6 *The nodes representing new and existing facilities in a tight path $P(E_i, E_j)$, occur with the same ordering and spacing in the path as do the corresponding locations in $P(v_i, v_j)$ on the tree. Also all facilities on the tight path are uniquely located.* \square

A sketch of the proof of this property can be given by a string analogue of the problem. Assume buttons representing new and existing facilities on the tight path $P(E_i, E_j)$ are placed on the corresponding nodes of the tree and the string path of length $\mathcal{L}(E_i, E_j)$ is pulled tight from E_i to E_j on the tree, the length of the string path exactly matches $d(v_i, v_j)$ in T , since the corresponding separation condition is binding.

2.3 Minimax Problem with Mutual Communication

The theory of separation conditions developed in Section 2.2 can be applied to Multifacility Minimax Problem with Mutual Communication defined below:

$$\begin{aligned} \text{Min}_{X \in G^m} \quad & f(X) \\ \text{where } f(X) \quad &= \max \{f_1(X), f_2(X)\} \\ \text{with } f_1(X) \quad &= \max \{w_{ji}d(x_j, v_i) : (j, i) \in I_C\} \\ f_2(X) \quad &= \max \{v_{jk}d(x_j, x_k) : (j, k) \in I_B\} \end{aligned}$$

All weights w_{ji} , v_{jk} are positive real numbers.

The problem is first defined in [4] in the presence of distance constraints and it is also established that the function f is convex on tree networks. The problem is shown to be \mathcal{NP} -Hard by Kolen [12] on general networks. The equivalent version of the problem in terms of distance constraints is solved in [8]. An equivalent formulation of the problem which we denote by $PMMC$ is as follows:

$$\begin{aligned} (PMMC) \quad & \text{Min} \quad z \\ & \text{subject to} \\ & d(x_j, v_i) \leq z/w_{ji} \quad , \quad (j, i) \in I_C \\ & d(x_j, x_k) \leq z/v_{jk} \quad , \quad (j, k) \in I_B \\ & X \in G^m \end{aligned}$$

Let z^* be the minimum z such that constraint set of $PMMC$ has a feasible solution. Existence of z^* is guaranteed due to continuity and compactness considerations and due to consistency of DC for sufficiently large z .

Given z , we can construct an auxiliary network which is the linkage network $LN(z)$ as dependent on z in a similar manner as described in Section 2.2. The edge lengths on $LN(z)$ are the corresponding reciprocal weights $1/w_{ji}$ or $1/v_{jk}$ multiplied by z . Let $\mathcal{L}_z(E_i, E_j)$ be the length of a shortest path in $LN(z)$. Then $\mathcal{L}_z(E_i, E_j) = z\mathcal{L}_1(E_i, E_j)$ where $\mathcal{L}_1(E_i, E_j)$ is the length of a shortest

path in $LN(1)$ with z taken to be unity. If we define $n_{ij} = \mathcal{L}_1(E_i, E_j)$ then we have $\mathcal{L}_z(E_i, E_j) = zn_{ij}$. Note that n_{ij} is just the sum of the reciprocal weights on the path $P(E_i, E_j)$ of $LN(1)$. Hence the separation conditions

$$d(v_i, v_j) \leq \mathcal{L}_z(E_i, E_j) \quad , \quad 1 \leq i < j \leq n \quad (2.7)$$

are equivalent to

$$d(v_i, v_j) \leq zn_{ij} \quad , \quad 1 \leq i < j \leq n \quad (2.8)$$

Then from (2.8) we have $z \geq d(v_i, v_j)/n_{ij}$, $1 \leq i < j \leq n$. So minimum feasible z is found by taking $z^* = \max_{1 \leq i < j \leq n} \{d(v_i, v_j)/n_{ij}\}$. Once z^* is computed, a feasible location vector can be constructed by applying *SLP* to constraints

$$\begin{aligned} d(x_j, v_i) &\leq z^*/w_{ji} \quad , \quad 1 \leq i < j \leq n \\ d(x_j, x_k) &\leq z^*/v_{jk} \quad , \quad 1 \leq j < k \leq m \end{aligned}$$

Complexity of solving *PMMC* is determined by the complexity of finding z^* which requires finding all shortest path distances in $LN(1)$. Using Dijkstra's shortest path algorithm once for each E -node, z^* can be calculated in $O(n(m+n)^2)$ time which dominates complexity of *SLP*.

An implication of tight paths discussed in Section 2.2 for *PMMC* is that new facilities on the tight paths are located uniquely without using *SLP* and the rest of the new facilities can be located by *SLP* operating on a smaller set of distance constraints.

2.3.1 Distance Constrained *PMMC* on Tree Networks

Erkut, Francis and Tamir [7] considers *PMMC* on tree networks in the presence of distance constraints. From the analysis in the previous sections it is easy to find the equivalent formulation of the problem as:

$$\begin{array}{ll}
(C - PMMC) & \text{Min } z \\
& \text{subject to} \\
& d(x_j, v_i) \leq z/w_{ji} \quad , \quad (j, i) \in I_C \\
& d(x_j, x_k) \leq z/v_{jk} \quad , \quad (j, k) \in I_B \\
& d(x_j, v_i) \leq c_{ji} \quad , \quad (j, i) \in I'_C \\
& d(x_j, x_k) \leq b_{jk} \quad , \quad (j, k) \in I'_B \\
& X \in G^m
\end{array}$$

where I'_C and I'_B are defined to indicate that they are not necessarily equivalent to I_C and I_B although defined similarly.

Analysis in [7] relies basically on the results obtained in [8] for the distance constraints. Authors propose two algorithms to solve $C - PMMC$. The first one is a polynomial algorithm which performs a binary search over the optimum objective value z and requires data to be rational numbers. The second algorithm is a strongly polynomial one which employs the general parametric approach suggested by Megiddo [14].

Binary search algorithm is based on two stages. In the first stage an interval of prespecified length that contains the optimum objective function value is found by performing a bisection search over a finite set of z values and in the second stage the exact optimal value of z is found. SLP is used to check for the feasibility of distance constraints at each iteration. Final step is to apply a shortest path algorithm to find the exact value of z .

Parametric approach depends on the results of Megiddo [14] for converting a (strongly) polynomial algorithm for a given combinatorial problem into a strongly polynomial algorithm to solve the parametric version of this combinatorial problem.

Let $F(z) = \min\{L(E_h, E_i) - d(v_h, v_i) : 1 \leq h < i \leq n\}$ where $L(E_h, E_i)$ is the shortest *direct* path in $LN(z)$ and a direct path is defined to be a path whose intermediate nodes are all N -nodes and terminal nodes are two distinct E -nodes. It was shown by Tansel et al. [15] that an equivalent version of separation conditions can be stated in terms of shortest direct paths. For some

value of z , $F(z)$ is attained for some direct path on $LN(z)$, so with each value of z an optimizing direct path can be associated. The real line can be decomposed into finite set of intervals on which all z have the same optimizing direct path in each interval, i.e. $F(z)$ is linear in each interval. At least one and at most two of these intervals contain z^* .

For a specified value of z an algorithm which uses a shortest path algorithm as a subroutine, is used to compute the value of $F(z)$. This algorithm is the primary algorithm for solving the parametric version of the problem with z being the parameter.

The algorithm is initiated by some *uncertainty interval* I_0 which contains z^* . The shortest path algorithm is applied parametrically with arc lengths that are some linear functions of z . Since this algorithm applies additions, subtractions, and comparisons only, it generates linear functions. For the comparison of two linear functions, the intersection point z' , called the *critical value*, is found. The sign of $F(z')$ is computed so that whether $z' \leq z^*$ or not is determined. In either case the interval of uncertainty can be reduced. So the algorithm can proceed to next step, without a need to specify a value for z in the current interval. This step is called *comparison resolution step*. This step is continued until termination where an interval containing z^* is left with the following properties:

There is the same optimizing pair for all values of z in the interval, i.e. $F(z)$ is linear over this interval, so z^* can easily be computed. The complexity of the parametric approach is $O(mn^2(m+n)^2)$.

2.4 Biobjective Multifacility Minimax and Vector Minimization

Tansel, Francis and Lowe [16] consider biobjective multifacility minimax problem on tree networks, which involves as objectives the maximum of the weighted

distances between specified pairs of new and existing facilities and maximum of the weighted distances between specified pairs of new facilities. The *Biobjective Multifacility Minimax Problem BMM* can be stated as follows:

$$\begin{aligned} \text{vector minimize} \quad & \{f(X) : x \in T^m\} \\ \text{where } f(X) = & (f_1(X), f_2(X)) \\ f_1(X) = & \max \{w_{ji}d(x_j, v_i) : (j, i) \in I_C\} \\ f_2(X) = & \max \{v_{jk}d(x_j, x_k) : (j, k) \in I_B\} \end{aligned}$$

[16] characterizes efficient points of *BMM* by making use of separation conditions and provides an $O(m^2(m+n)^2)$ procedure to construct the efficient frontier which is the set of two tuples (z_1, z_2) constituting the objective values of efficient points.

First the problem is transformed into an equivalent form in terms of distance constraints, i.e. $Z = (z_1, z_2)$ is to be vector minimized subject to

$$\begin{aligned} d(x_j, v_i) & \leq z_1/w_{ji} \quad , \quad (j, i) \in I_C \\ d(x_j, x_k) & \leq z_2/v_{jk} \quad , \quad (j, k) \in I_B \end{aligned}$$

The graph $LN(Z)$ is constructed associated with the above *DC*. Then it is shown that the following are equivalent.

- (a) A location of vector X is efficient.
- (b) At least one of the arcs (N_j, N_k) , $(j, k) \in I_B$ is in a tight path in $LN(Z)$ where $Z = f(X)$.

Further general results are provided which unify the necessary and sufficient conditions for efficiency of a broad class of location problems which involve t ($t \geq 1$) minimax type objectives.

The *t-objective Multifacility Minimax Problem* can be defined as follows:

$$\begin{aligned}
& \text{vector minimize} && \{f(X) : x \in T^m\} \\
& \text{where } f(X) &= & (f_1(X), \dots, f_t(X)) \\
& f_r(X) &= & \max [\max \{w_{ji}^r d(x_j, v_i) : (j, i) \in I^r \cap I_C\}, \\
& & & \max \{v_{jk}^r d(x_j, x_k) : (j, k) \in I^r \cap I_B\}], \\
& & & 0 \leq r \leq t
\end{aligned}$$

I^r is assumed to be a nonempty subset of $I_C \cup I_B$, and weights w_{ji}^r and v_{jk}^r are positive for each $1 \leq r \leq t$. $f_r(X)$ is defined as the r -th objective.

For the t -objective multifacility minimax problem a linkage network LN_{BC} is constructed. The node set of LN_{BC} is $\{N_1, \dots, N_m\} \cup \{E_1, \dots, E_n\}$ and the arc set is as follows: For each r , $1 \leq r \leq t$ if $(j, i) \in I^r \cap I_C$ then there is an arc $a_r(N_j, E_i)$ of length z_r/w_{ji}^r and if $(j, k) \in I^r \cap I_B$ then there is an arc $a_r(N_j, N_k)$ of length z_r/v_{jk}^r . Thus there exist several arcs between a pair of nodes of LN_{BC} . It is proven in [16] that X is efficient *if and only if* for every $r \in \{1, \dots, t\}$ with $z_r > 0$, at least one arc associated with the r -th objective is in a tight path in LN_{BC} , where $Z = f(X)$.

2.5 Other Approaches

Erkut, Francis, Lowe, and Tamir [6] consider the multifacility location problem on tree networks subject to distance constraints. All constraints and the objective function 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 [6] that such problems are equivalent to mathematical programming problems which, when each function is expressed using only maximization and summation operations on nonnegatively weighted arguments, are linear programming problems of polynomial dimensions. This result may constitute another partial answer to the question why tree networks are more tractable than cyclic network problems since they have equivalent mathematical programming formulations, while cyclic network versions of the same problems do not.

We also consider two papers by Chhajed and Lowe [2, 3] which are the only papers related to more general graphs than trees. The main concern in [2] is the minisum problem with mutual communication, but their approach is applicable to node restricted version of the minimax problem with mutual communication. The minisum problem is reformulated in the form of a graph theoretic *Node Selection Problem* defined on a special graph which is solved in polynomial time when the graph that denotes the interaction between pairs of new facilities has the special structure of *series-parallel* graphs.

In [3], authors consider a generic multifacility location problem, which subsumes as special cases several \mathcal{NP} -Hard location problems. The generic problem is solved in polynomial time for the *k-tree* structured interaction between new facilities. This paper identifies the broadest efficiently solvable classes of several difficult problems which includes the node restricted version of the minimax problem with mutual communication.

The problem we consider in this thesis is a feasibility problem whereas works in [2, 3] consider the related optimization form. The basic deviation of our approach from that of [2] and [3] is twofold: First, we consider the continuous version of the problem while [2, 3] restrict the locations of new facilities to finitely many candidate points. Second, we require that the interaction between new facilities has a tree structure whereas in [2] and [3] the problem is solved for series-parallel and *k-tree* structured interactions. Hence our approach is a generalization of [2] and [3] in one respect while it is a restriction in the other.

2.6 *DC* is \mathcal{NP} -Complete on General Networks

Up to now, we have provided the basic theory for tree networks, and cited only two works on networks more general than trees that covered the node restricted version of the related minimax problem as subcases. In this section we provide the basic result by Kolen [12] regarding the distance constraints on general networks.

The problem of finding a clique of size c of a given graph is known to be \mathcal{NP} -Complete (see [9]). The problem *CLIQUE* which we state next, is reduced to the distance constraints problem of deciding whether there exists a feasible solution to the given set of distance constraints. So the distance constraints problem is also \mathcal{NP} -Complete.

CLIQUE

Instance: A graph $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and an integer c .

Question: Does there exist a subset $C \subseteq V$ of cardinality c such that $[v_j, v_k] \in E$ if $\{v_j, v_k\} \subseteq C$?

Let $DC - d$ denote the problem of deciding whether there exists a feasible solution to the given set of distance constraints. $DC - d$ is stated next.

DC-d

Instance: A graph $G' = (V', E')$ with $V' = \{v'_1, \dots, v'_m\}$ $m \in \mathcal{N}$, lengths $\ell(e) \in \mathcal{Z}_+$, $e \in E'$, $c_{ji} \in \mathcal{Q}_+$, $1 \leq j \leq m$ and $1 \leq i \leq n$, $b_{jk} \in \mathcal{Q}_+$, $1 \leq j < k \leq m$.

Question: Is there a set $X = \{x_1, \dots, x_m\}$ on G' such that

$$\begin{aligned} d(x_j, v_i) &\leq c_{ji} \quad , \quad 1 \leq j \leq m \quad , \quad 1 \leq i \leq n, \\ d(x_j, x_k) &\leq b_{jk} \quad , \quad 1 \leq j < k \leq m? \end{aligned}$$

Now we state the basic result regarding the distance constraints on general networks.

Theorem 2.2 $DC - d$ is \mathcal{NP} -Complete.

Proof: $DC - d$ belongs to the class \mathcal{NP} since by using standard shortest path techniques we can test whether a given solution satisfies the bounds in polynomial time.

We shall reduce the problem *CLIQUE* to $DC - d$. Let an instance of *CLIQUE* be given by $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and an integer c . The

corresponding instance of $DC - d$ is given by $V' = V \cup W \cup U \cup \{p\} \cup \{q\} \cup \{r\}$, where $W = \{w_1, \dots, w_n\}$ and $U = \{u_1, \dots, u_n\}$, $E' = \{[p, v_i], [q, v_i], [r, v_i], [v_i, w_i], [w_i, u_i] : i = 1, \dots, n\} \cup \{[v_i, u_j] : [v_i, v_j] \in E, i \leq j\}$, $m = 3c$, $\ell(e) = 1$ for all $e \in E$, $c_{ji} = 1$ for $1 \leq j \leq m$, $1 \leq i \leq n$, $b_{jk} = 1$ for $1 \leq j < k \leq n$. (See example in Figure 2.4 with $c = 3$.)

We claim that $G = (V, E)$ contains a clique of size c if and only if there exists points x_1, \dots, x_m on $G' = (V', E')$ such that the weighted distances are less than or equal to the given upper bounds, i.e.

$$d(x_j, p) \leq 1, \quad d(x_{j+c}, q) \leq 1, \quad d(x_{j+2c}, r) \leq 1, \quad j = 1, \dots, c \quad (2.9)$$

$$d(x_j, x_{j+c}) \leq 1, \quad d(x_{j+c}, x_{j+2c}) \leq 1, \quad j = 1, \dots, c, \quad (2.10)$$

$$d(x_i, x_{j+2c}) \leq 1, \quad 1 \leq i < j \leq c. \quad (2.11)$$

Let x_1, \dots, x_{3c} be the points on $G' = (V', E')$ satisfying (2.9), (2.10) and (2.11). Since $d(p, q) = d(q, r) = 1$, from (2.9) and (2.10) we have $x_j \in V$, say $x_j \in v_j$, $x_{j+c} = w_j$ and $x_{j+2c} = u_j$, $j = 1, \dots, c$. It follows from (2.11) that $[v_i, u_j] \in E'$, i.e., $[v_i, v_j] \in E$, $i < j$. Hence $\{v_j : j = 1, \dots, c\}$ is a clique in the graph $G = (V, E)$.

Let $\{v_j : j = 1, \dots, c\}$ be a clique of size c in G . Define $x_j = v_j$, $x_{j+c} = w_j$, $x_{j+2c} = u_j$, $j = 1, \dots, c$. Then x_1, \dots, x_{3c} satisfy (2.9), (2.10) and (2.11). \square

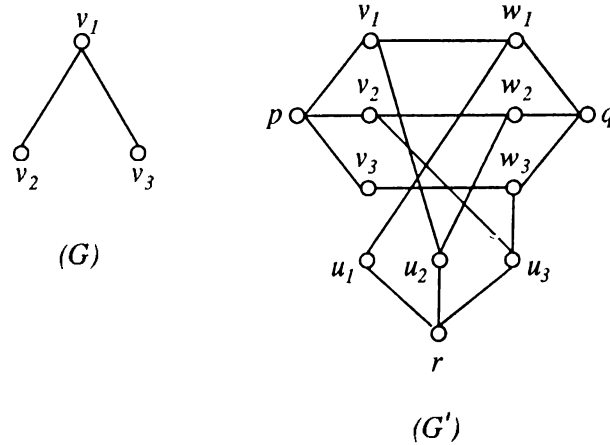


Figure 2.4: Example of the Construction of G' from G .

Chapter 3

DISTANCE CONSTRAINTS AND EXPAND/INTERSECT METHOD IN GENERAL METRIC SPACES

In this chapter we deviate from the network context and introduce a new approach to solve distance constraints in general metric spaces. To pose the problem in its general form, let L be a metric space with distance d that satisfies the usual axioms of nonnegativity, symmetry and triangle inequality of a metric. Some well known instances of (L,d) are obtained by taking L to be the k -dimensional Euclidean space and d to be ℓ_p -norm. Our particular interest in the next chapter will be in the case where L is an embedded network and $d(x,y)$ is the length of a shortest path between x and y in L . The main results we give in this chapter are true for arbitrary metric spaces.

We have previously defined distance constraints on networks. The definition of DC on general metrics is similar. We take v_1, \dots, v_n be n fixed points in L , rather than vertices in G , with v_i being the location of existing facility i . New facilities are to be placed at points x_1, \dots, x_m any where in the location space to

satisfy distance bounds on specified pairs of facilities. So we have:

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

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

$$x_1, \dots, x_m \in L$$

Again c_{ji} , b_{jk} are nonnegative finite constants and I_C and I_B specify the pairs of facilities for which there is a bound on their separation.

In Chapter 2, we have defined an auxiliary network called the *Linkage Network LN* which conveniently represented the data of the problem. Since, construction of LN does not rely on the specific location space, an equivalent auxiliary network LN is also valid for DC defined in general metric spaces. Now, we denote the portion of LN spanning only N_j 's and the arcs between them by LN_B . Figure 3.1 shows an example of a linkage network and related LN_B .

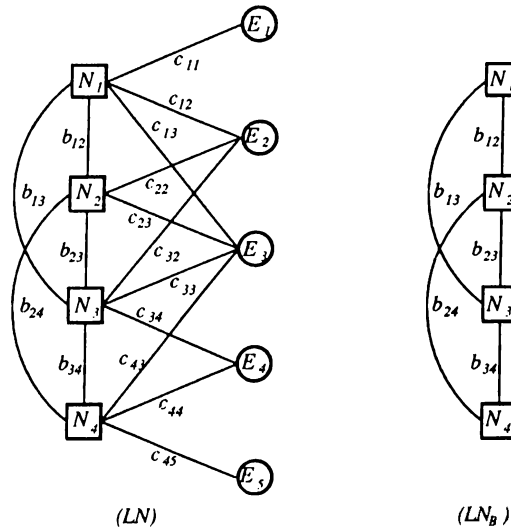


Figure 3.1: Example of LN and Related LN_B

In the next section we give a new approach to solve DC in general metric spaces for the case when LN_B has a tree structure. The basic idea of the method depends on successively performing two set operations *EXPANSION* and *INTERSECTION* defined on subsets of the location space.

3.1 EXPAND/INTERSECT Method for General Metrics

Given a nonempty subset S of L and a nonnegative constant r , the *expansion of S by r* is the set of all points in L for which there exists y in S with $d(x, y) \leq r$. We denote the expansion of S by r by $N(S, r)$. $N(S, r)$ includes all points of L that are reachable from at least one point in S within r distance units. For example, if L is the plane, d is the Euclidean distance, and S is the ball centered at \bar{x} with radius α , then $N(S, r)$ is the ball centered at \bar{x} with radius $\alpha + r$. (Figure 3.2)

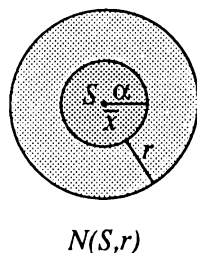


Figure 3.2: Expansion of S by r (in the Plane with Euclidean Distance)

The definition of expansion implies $x \in N(S, r)$ if and only if $x \in N(y, r)$ for some point $y \in S$ if and only if $x \in \bigcup_{y \in S} N(y, r)$. Hence, we have $N(S, r) = \bigcup_{y \in S} N(y, r)$. With this observation we may think of $N(S, r)$ as the neighborhood of S within r distance units (this explains the choice of our notation $N(\cdot, r)$).

Note that the definition of expansion does not require any assumptions on S other than it being nonempty. S may be finite, infinite, connected, disconnected, open or closed. If S is compact, then any expansion of S is also compact provided that r is finite and L is closed.

First, let us consider $DC(1)$. Since we are looking for a point x in L such that $d(x, v_i) \leq c_i$, $i \in I = \{1, \dots, n\}$, the feasible set for $DC(1)$ is $S = \bigcap_{i \in I} N(v_i, c_i)$. Each neighborhood $N(v_i, c_i)$ is nonempty since we assume c_i

are finite nonnegative constants. The intersection may or may not be empty. How one computes the neighborhoods and their intersection is a computational question that requires a specific knowledge on the type of location space under consideration. We defer the computational side to Chapter 4 for the case of networks.

Consider now DC with $m > 1$. For $j = 1, \dots, m$, define I_j to be the subset of I consisting of indices I for which $(j, i) \in I_C$. I_j is the set of indices of the E -nodes in the linkage network that are connected to N_j with an arc. Let $J = \{1, \dots, m\}$.

We may rewrite (DC.1) in the form

$$d(x_j, v_i) \leq c_{ji} \quad , \quad i \in I_j \quad , \quad j \in J.$$

Feasibility requires each x_j be in the set $S_j = \bigcap_{i \in I_j} N(v_i, c_{ji})$ for $j \in J$. If $I_j = \emptyset$, we take $S_j = L$. Now, an equivalent statement of DC is as follows:

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

$$x_j \in S_j \quad , \quad j \in J. \quad (DC.1')$$

It is evident from (DC.2) and (DC.1') that if LN_B is disconnected, then DC decomposes into independent subproblems each corresponding to a component of LN_B . So we may justifiably assume LN_B is connected.

We now give a method to construct a feasible location vector X , whenever it exists, under the assumption LN_B is a tree. To motivate the method, let (j, k) be a fixed pair in I_B . If we expand S_j by b_{jk} units and intersect this expansion with S_k , then every location x_k in this intersection (when nonempty) allows the choice of some x_j in S_j such that $d(x_j, x_k) \leq b_{jk}$. This follows from the fact that $x_k \in S_k \cap N(S_j, b_{jk})$ implies $x_k \in N(S_j, b_{jk}) = \bigcup_{y \in S_j} N(y, b_{jk})$ so that there exists a point y in S_j such that $x_k \in N(y, b_{jk})$. Clearly then we may take $x_j = y$ to satisfy the constraint $d(x_j, x_k) \leq b_{jk}$.

We call the operation $S_k \leftarrow S_k \cap N(S_j, b_{jk})$ the *EXPAND/INTERSECT Operation from S_j into S_k* . The algorithm performs this operation sequentially

and is termed the *Sequential Expand/Intersect Procedure (SEIP)*.

Before we introduce *SEIP* we give an indexing convention for LN_B . (Note that LN_B is assumed to be a tree.) We root the tree at a new facility and relabel it, if necessary as m (i.e. as N_m) and relabel the nodes so that the children of node j have smaller indices than j . If j_1, \dots, j_k are children of j , we say j is the parent of j_i , $i = 1, \dots, k$. The numbering implies $j > j_i$ for each child j_i of j . Let J_k denote the set of indices of the children of a given node N_k .

SEIP consists of two phases. In the first phase, we process nodes beginning with leaves of LN_B and moving towards 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 N_k in some iteration, we aim to find the intersection of S_k with expansions of all of its children. This is done sequentially in an arbitrary order of child indices. The intermediate intersections are denoted as F'_k . Initially, $F'_k = S_k$. Then F'_k is replaced by $F'_k \cap N(F_j, b_{jk})$ for some child N_j of N_k . This operation is repeated for each remaining child of N_k . Each intersection causes F'_k to either shrink or remain the same. When the intersection is found with all children of N_k , the final set F'_k is defined to be F_k .

During the procedure, if F_k becomes empty at any stage, this indicates inconsistency of *DC*. We terminate infeasible when this happens.

Construction of the sets F_k , $k = 1, \dots, m$ constitutes the first phase of *SEIP*. If all F_k are nonempty, the second phase is initiated. In the second phase, actual locations of new facilities are found in reverse order, moving from the root towards the leaves. A node is eligible for processing in some iteration of *Phase 2* only if its unique parent is already processed (that is, the location of the parent facility is already selected). The constructed solution x_1, \dots, x_m is feasible (to be proven).

SEIP (Sequential Expand/Intersect Procedure)(Constructs a Feasible Solution to DC when LN_B is a tree)**Phase 1.** (*Input* : $S_1, \dots, S_m, \{b_{jk} : (j, k) \in LN_B\}$, metric $d(\cdot)$ and L , root index)**Initial** : $F_j = S_j$ for every childless j .**Main Step** : Choose any N_k for which F_j is available for every child N_j of N_k .Set $F'_k \leftarrow S_k$.For each child N_j of N_k apply *EXPAND/INTERSECT Operation*:Construct the expansion $N(F_j, b_{jk})$ and compute $F'_k \leftarrow F'_k \cap N(F_j, b_{jk})$.After all children of N_k are processed set $F_k \leftarrow F'_k$.If $F_k = \emptyset$ then terminate *infeasible*.If F_m is computed (i.e. the root node is processed) and $F_m \neq \emptyset$, then go to *Phase 2*.

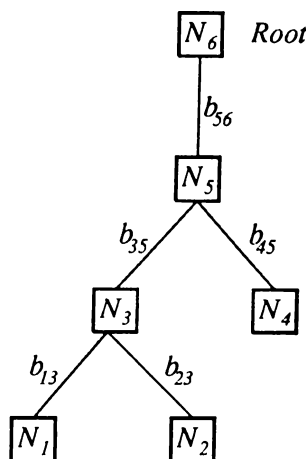
Otherwise repeat the main step.

Phase 2 . (*Input* : F_1, \dots, F_m all nonempty, $\{b_{jk} : (j, k) \in LN_B\}$)**Initial** : Choose an arbitrary point y in F_m and set $x_m = y$.**Main Step** : Choose any N_k whose unique parent has already been processed.That is, choose any N_k for which $x_{p(k)}$ is already selected where $p(k)$ is the index referring to the unique parent $N_{p(k)}$ of N_k .Construct $F_k(x_{p(k)}) = N(x_{p(k)}, b_{k,p(k)}) \cap F_k$.Choose any y in $F_k(x_{p(k)})$ and put $x_k = y$ (the proof $F_k(x_{p(k)}) \neq \emptyset$ will be given).If x_1, \dots, x_m are all chosen, then *stop*.

Otherwise repeat the main step.

To illustrate *SEIP*, suppose LN_B is as shown in Figure 3.3.

Assuming N_6 is selected as the root node, we initialize *Phase 1* with $F_1 = S_1$, $F_2 = S_2$, $F_4 = S_4$. Then we choose N_3 and construct $F_3 = N(F_1, b_{13}) \cap N(F_2, b_{23}) \cap S_3$. The next node to choose is N_5 yielding $F_5 = N(F_3, b_{35}) \cap N(F_4, b_{45}) \cap S_5$. Finally, N_6 is chosen to construct $F_6 = N(F_5, b_{56}) \cap S_6$.

Figure 3.3: Example to Illustrate *SEIP*

Phase 2 selects nodes in reverse order from the root towards the leaves. A legal sequence of selection is $N_6, N_5, N_4, N_3, N_1, N_2$ which begins with x_6 arbitrarily chosen in F_6 , followed by $x_5 \in F_5 \cap N(x_6, b_{56})$, $x_4 \in F_4 \cap N(x_5, b_{45})$, $x_3 \in F_3 \cap N(x_5, b_{35})$, $x_1 \in F_1 \cap N(x_3, b_{13})$, $x_2 \in F_2 \cap N(x_3, b_{23})$. The selection of each x_k is arbitrary in its sets $F_k \cap N(F_{p(k)}, b_{k,p(k)})$. The constructed vector (x_1, \dots, x_6) satisfies $d(x_1, x_3) \leq b_{13}$, $d(x_2, x_3) \leq b_{23}$, $d(x_3, x_5) \leq b_{35}$, $d(x_4, x_5) \leq b_{45}$, $d(x_5, x_6) \leq b_{56}$, $x_j \in S_j$, $j = 1, \dots, 6$ and is feasible.

It is evident that *Phase 1* of *SEIP* terminates in at most $m - 1$ repetitions of the main step each corresponding to an expand/intersect operation defined by a distinct arc of LN_B . Likewise, *Phase 2* terminates in m steps with each step corresponding to a selection of x_k .

In the next section we will prove the correctness of the algorithm which will in turn lead to the proof of the necessary and sufficient conditions for consistency of DC which we state next.

Theorem 3.1 *DC is consistent if and only if Phase 1 SEIP terminates feasible (i.e. $F_j \neq \emptyset$ for all $j = 1, \dots, m$).*

Observe that F_1, \dots, F_m can always be constructed for any metric space as long as LN_B is a tree. The time bound of the construction is $O(mg(L, d))$ where

$g(L, d)$ is the time bound for *one* expand/intersect operation as dependent on the metric space (L, d) . When L is a network, $g(L, d)$ is polynomial in the number of new and existing facilities and the number of edges of the network (will be proven in Chapter 4). Hence, consistency check is done in polynomial time for networks.

3.2 Analysis of *SEIP*

This section proves the correctness of *SEIP* and the necessary and sufficient conditions for consistency of *DC* given in the previous section.

The following lemma justifies the correctness of the second phase of *SEIP*.

Lemma 3.1 *In Phase 2 of SEIP, $F_k(x_{p(k)}) \neq \emptyset$, $k = 1, \dots, m - 1$.*

Proof : Let $k \in \{1, \dots, m - 1\}$. The algorithm chooses k only if $x_{p(k)}$ is already located. Clearly there is at least one such k at the beginning since $x_m \in F_m \neq \emptyset$ is already picked so that every children k of m can be processed. When the children of m are picked, then their children can be processed and so on. So, there is no difficulty as far as choosing an index k whose parent $p(k)$ has already been processed. Now we must show $F_k(x_{p(k)}) \neq \emptyset$. Let $q = p(k)$. Then $F_k(x_{p(k)}) = F_k(x_q) = N(x_q, b_{kq}) \cap F_k$ where the last equality follows from the definition of $F_k(x_{p(k)})$. If $q = m$ then $x_m \in F_m$. If $q < m$, then from the inductive structure of the procedure we know that $x_q \in F_q(x_{p(q)}) \subseteq F_q$ which implies that $x_q \in F_q$. This then implies, using the definition of F_q , $x_q \in S_q \cap C_q$ where $C_q = \bigcap_{j \in J_q} N(F_j, b_{jq})$ which is a subset of $N(F_k, b_{kq})$ since k is one of the children of q . Hence, $x_q \in N(F_k, b_{kq})$ so that there exists a point y in F_k such that $d(x_q, y) \leq b_{kq}$. Then $y \in N(x_q, b_{kq})$ and $y \in F_k$ so that $y \in N(x_q, b_{kq}) \cap F_k = F_k(x_q)$. Hence, $F_k(x_q) = F_k(x_{p(k)}) \neq \emptyset$. \square

The next theorem gives the sufficient conditions for consistency of *DC*. In *SEIP* we enter into *Phase 2* only if all F_1, \dots, F_m are nonempty, that is when

Phase 1 terminates feasible, and the theorem proves that if *Phase 1* of *SEIP* terminates feasible we can construct a location vector which is feasible to *DC*. Hence *DC* is consistent.

Theorem 3.2 x_1, \dots, x_m constructed by *Phase 2* of *SEIP* is feasible.

Proof : Clearly, $x_k \in F_k(x_{p(k)})$ for all k . Since $F_k(x_{p(k)}) = N(x_{p(k)}, b_{k,p(k)}) \cap F_k$, for all $k \neq m$ $d(x_k, x_{p(k)}) \leq b_{k,p(k)}$. Further, $x_k \in F_k$ implies $x_k \in S_k$ (since $F_k \subseteq S_k$) so that $d(x_k, v_r) \leq c_{kr} \forall r \in I_k$ which completes the proof. \square

The next lemma is significant because it proves that the sets constructed by *Phase 1* of *SEIP* covers all the feasible solutions. That is each F_j contains the j -th components of all the feasible location vectors. Note that the converse is not true, that is not all points in a given F_j need to be feasible (except for the root node which will be proven in the next section).

Lemma 3.2 Let $X = (x_1, \dots, x_m)$ be a feasible location vector to *DC* then $x_j \in F_j$, $j = 1, \dots, m$.

Proof : (by induction) Let τ be the set of indices of all leaf vertices. From the initialization step of *Phase 1*, $F_k = S_k \forall k \in \tau$. Since $(DC.1')$ imply $x_j \in S_j \forall j$, the statement is true for all childless vertices, i.e. $x_k \in F_k \forall k \in \tau$.

Now let $q \in \{1, \dots, m\} - \tau$ and assume $x_i \in F_i$ for all descendants of q (inductive hypothesis). Feasibility implies from $(DC.1')$ that $x_q \in S_q$ and from $(DC.2)$ that $d(x_j, x_q) \leq b_{jq}$ for all $j \in J_q$. Then from the definition of neighborhood we have $x_q \in N(x_j, b_{jq}) \forall j \in J_q$. Hence, $x_q \in \bigcap_{j \in J_q} N(x_j, b_{jq}) \cap S_q$. From the inductive hypothesis we have $x_j \in F_j \forall j \in J_q$. Hence $N(x_j, b_{jq}) \subseteq N(F_j, b_{jq})$ which implies $x_q \in \bigcap_{j \in J_q} N(F_j, b_{jq}) \cap S_q$. Observe that the right hand side is simply F_q by the construction scheme in *Phase 1*. Hence we have $x_q \in F_q$ which completes the proof. \square

An immediate corollary to the above lemma proves the necessity part for the consistency of *DC*.

Corollary 3.1 *If DC is consistent then $F_j \neq \emptyset$ for all $j = 1, \dots, m$ (i.e. Phase 1 of SEIP terminates feasible).*

Proof : Consistency of DC imply that there exists at least one feasible location vector $X = (x_1, \dots, x_m)$. In Lemma 3.2 it is proven that for any feasible location vector X , $x_j \in F_j$ $j = 1, \dots, m$. Hence $F_j \neq \emptyset$ for all $j = 1, \dots, m$. That is, Phase 1 of SEIP terminates feasible.

3.3 Regions of Feasibility

In this section we are interested in a broader problem of characterizing the set of all feasible solutions by defining a *region of feasibility* for each x_j . This problem has not been addressed in the literature except for $m = 1$. To make the notion precise, define for $j = 1, \dots, m$, the set

$$L_j = \{x \in L : \text{there exists a feasible location vector } X \text{ such that } x_j = x\}.$$

L_j is the collection of the j -th component of all feasible location vectors. That is, L_j is the *projection* of the set of all feasible solutions onto the location space L in the j -th coordinate.

Lemma 3.2 implies that the j -th components of all feasible location vectors are included in the corresponding sets F_j constructed in Phase 1 of SEIP. The sets L_j , by definition, consists of all such points. Hence we have $L_j \subseteq F_j$, $j = 1, \dots, m$.

These sets are significant because they readily allow to construct a feasible solution. For example, assume say L_1 is (somehow) computed, place new facility 1 at a point in L_1 and solve DC for the remaining $m - 1$ new facilities by changing the status of new facility 1 to an existing facility. This immediately suggests a recursive procedure that eliminates new facilities one at a time from DC and changing their status to existing facilities in subsequent steps. This way we can construct as many feasible location vectors as desired.

We call L_j the *region of feasibility for new facility j* . Any location in L_j is a feasible choice for x_j in the sense that it allows to locate all other new facilities feasibly. Furthermore, each L_j is a maximal set having this property.

Of course, in order these definitions to be operational we have to find a way to construct these sets efficiently. The following theorem readily solves this question.

Let F_m be the set constructed by *Phase 1* of *SEIP* for the root node N_m of LN_B .

Theorem 3.3 $F_m = L_m$.

Proof : As we have indicated earlier Lemma 3.2 implies $L_j \subseteq F_j$ for all $j = 1, \dots, m$. Hence we have $L_m \subseteq F_m$. From the construction scheme in *Phase 2* of *SEIP* we know that for all $x \in F_m$ we can construct a location vector $X = (x_1, \dots, x_m)$ with $x_m = x$ which is feasible (from Theorem 3.2). By definition L_m contains all such points which implies $F_m \subseteq L_m$. Hence the proof is complete. \square

The above theorem immediately suggests a way to construct all L_j , $j = 1, \dots, m$. Simply root the tree LN_B at each node and apply *SEIP*. However, we have a much more efficient way to construct all L_j , $j = 1, \dots, m$. We first give the following lemma which provides a stronger characteristic of expand/intersect operation.

Lemma 3.3 Let S_1 and S_2 be arbitrary two sets in a metric space. Then

$$N(S_1, r) \cap S_2 = N(F_1, r) \cap S_2$$

where $F_1 = N(S_2, r) \cap S_1$ and r is a nonnegative real number.

Proof : Since $F_1 \subseteq S_1$, we can write $S_1 = (S_1 - F_1) \cup F_1$. Then $N(S_1, r) \cap S_2 = N[(S_1 - F_1) \cup F_1, r] \cap S_2 = [N(S_1 - F_1, r) \cup N(F_1, r)] \cap S_2 = [N(S_1 -$

$F_1, r) \cap S_2] \cup [N(F_1, r) \cap S_2]$. So, if we can show that $N(S_1 - F_1, r) \cap S_2 = \emptyset$ then the proof will be complete.

Let $y \in (S_1 - F_1)$. Since $y \in S_1$ and $y \notin F_1$, we have $y \notin N(S_2, r)$. It follows then $\forall y \in (S_1 - F_1)$ we have $y \notin N(S_2, r)$. Let $u \in N(S_1 - F_1, r)$. Then for some $y \in (S_1 - F_1)$ $d(y, u) \leq r$. If $u \in S_2$, then $d(y, u) \leq r$ implies $y \in N(S_2, r)$ which is impossible. Hence $u \notin S_2$. We have shown that $u \in N(S_1 - F_1, r)$ implies $u \notin S_2$ then we have $N(S_1 - F_1, r) \cap S_2 = \emptyset$. \square

The lemma states that a recursive application of expand/intersect operation on two distinct sets is actually the same as applying this operation on these sets independently. More than that the intersection sets contain the actual sets of points which supply the pairs (x, y) that satisfy the constraints $d(x, y) \leq r$.

The next theorem suggests a simple algorithmic way to construct all L_j 's for $j = 1, \dots, m$.

Theorem 3.4 *Let an altered version Phase \mathcal{Q} of Phase 2 of SEIP be given as follows:*

Phase 2'. (Input : F_1, \dots, F_m all nonempty, $\{b_{jk} : (j, k) \in LN_B\}$)

Initial : Set $F_m^* \leftarrow F_m$.

Main Step : Select any k whose parent $p(k)$ has already been processed (i.e. $F_{p(k)}^*$ is already constructed).

EXPAND/INTERSECT Operation : Construct the expansion $N(F_{p(k)}^*, b_{k,p(k)})$ and compute $F_k^* = N(F_{p(k)}^*, b_{k,p(k)}) \cap F_k$.

If all F_1^*, \dots, F_m^* are constructed then STOP. Otherwise, repeat the main step.

Then $F_j^* = L_j$ for all $j = 1, \dots, m$.

Proof : From the initialization step of Phase \mathcal{Q} we have $F_m^* = F_m$ and by Theorem 3.3 we have $F_m^* = F_m = L_m$. Since SEIP enters into Phase \mathcal{Q} only

if all F_1, \dots, F_m are nonempty we have $F_m^* \neq \emptyset$. Now, if we expand F_m^* by b_{km} units and intersect with F_k where k is a child of m then we have

$$\begin{aligned} F_k^* &= N(F_m^*, b_{km}) \cap F_k \\ &= N(F_m, b_{km}) \cap F_k \\ &= N\left[\left(\bigcap_{j \in J_m} N(F_j, b_{jm} \cap S_m)\right), b_{km}\right] \cap F_k \end{aligned}$$

where the third equality follows from the construction of F_m in *Phase 1*. Note that $N(F_m^*, b_{km}) \cap F_k \neq \emptyset$ because from the construction of F_m ($= F_m^* = L_m$) for all $x \in F_m$ we have at least one point y in F_k such that $d(x, y) \leq b_{km}$ and also y in $N(F_m^*, b_{km})$. Now, if we rewrite the same thing with child k separated we have

$$F_k^* = N\left[\left(N(F_k, b_{km}) \cap \left(\bigcap_{j \in J_m - \{k\}} N(F_j, b_{jm}) \cap S_m\right)\right), b_{km}\right] \cap F_k.$$

Let $K_k = \bigcap_{j \in J_m - \{k\}} N(F_j, b_{jm}) \cap S_m$. Then Lemma 3.3 states that

$$N\left[\left(N(F_k, b_{km}) \cap K_k\right), b_{km}\right] \cap F_k = N(K_k, b_{km}) \cap F_k.$$

Hence we have

$$F_k^* = N\left[\bigcap_{j \in J_m - \{k\}} N(F_j, b_{jm}) \cap S_m, b_{km}\right] \cap F_k.$$

At this stage, observe that we can view the last RHS as if we rooted LN_B at node N_k and applied *Phase 1* of *SEIP*. Call this the second application of Phase 1. In the second application, node N_k becomes the parent of node N_m and the children of node N_m are now nodes identified by the indices in $J_m - \{k\}$ and while the children of node N_k are the nodes identified by the indices in $J_k \cup \{m\}$ (See Figure 3.4).

From the above analysis, it is direct to observe that the sets F_j , $j \in J - \{k, m\}$ of the first application of *Phase 1* remain the same in the second application, only the sets F_k and F_m are altered. Let F_k^2 and F_m^2 denote the sets altered in the second application, so that $F_m^2 = \bigcap_{j \in J_m - \{k\}} N(F_j, b_{jm}) \cap S_m$

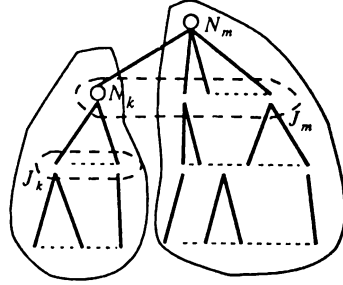


Figure 3.4: New Root in the Second Application.

and $F_k^2 = N(F_m^2, b_{km}) \cap F_k$. From the construction of F_k in the first application we rewrite F_k^2 as follows:

$$F_k^2 = N(F_m^2, b_{km}) \cap \left(\bigcap_{i \in J_k} N(F_i, b_{ik}) \cap S_k \right).$$

This is simply the final construction step in the second application of *Phase 1*. Hence by Theorem 3.3 we have $F_k^2 = L_k$. Since F_k^2 is also equivalent to F_k^* we have $F_k^* = L_k$. This result suggests that once we have applied *Phase 1* of *SEIP* and obtained $F_m = L_m$, we can construct L_j for each child j of m by one backward application of expand/intersect operation. Then we can apply the same reasoning to the children of all nodes N_j , $j \in J_m$ (i.e. to grand children of node N_m) This recursive application of expand/intersect operation backward from root to leaves of LN_B gives *Phase 2*. Hence, at any iteration a region of feasibility is constructed for a child new facility whose unique parent has already been processed. \square

An immediate result of Theorem 3.4 is that the construction of regions of feasibility for all new facilities can be done in the same order of the original *SEIP* as dependant on the complexity of one expand/intersect operation on the given location space.

3.4 Other Set Constraints and Distance Constraints

Consider now a more general version of the problem where, in addition to distance constraints, there are other constraints that restrict each x_j to a subset of the location space. The additional constraints may be appropriate, for example, to restrict new facilities to finitely many candidate points or to disallow locating them in certain forbidden regions. Let Ω_j be a subset of L which restricts location of x_j to points in Ω_j . Consider the problem of finding a location vector (x_1, \dots, x_m) in L_m such that

$$d(x_j, x_k) \leq b_{jk} \quad , \quad (j, k) \in I_B \quad (DC.1)$$

$$d(x_j, v_i) \leq c_{ji} \quad , \quad (j, i) \in I_C \quad (DC.2)$$

$$x_j \in \Omega_j \quad , \quad j = 1, \dots, m \quad (SC)$$

Here, (SC) refers to arbitrary set constraints. We assume Ω_j is nonempty for each j otherwise there is no solution. Other than this we make no assumptions on Ω_j 's. For example, the vertex restricted version of distance constraints on a network may be posed by taking $\Omega_j = V \equiv \{v_1, \dots, v_n\}$ for all j . If different new facilities are restricted to different vertex subsets, we may take $\Omega_j = V_j$ where each V_j is a subset of V . We may also assume $\Omega_j = L$ for certain j while Ω_j is a proper subset of L for other j . If L^* is a forbidden region in L where none of the new facilities can be placed, we may model this situation by taking $\Omega_j = L - L^* \forall j$. Of course, it is also possible to specify a different forbidden region L_j^* for each j in which case we take $\Omega_j = L - L_j^*$.

Let us call the collection of constraints $(DC.1)$, $(DC.2)$, and (SC) , the *General Distance Constraints (GDC)*. Assuming that LN_B defined by I_B is a tree network, we can solve (GDC) using our existing methods as follows: First, construct $\hat{S}_j = \bigcap_{i \in I_j} N(v_i, c_{ji})$ for each j where I_j is the set of existing facility indices i for which $(j, i) \in I_C$. Then construct $S_j = \hat{S}_j \cap \Omega_j$ for $j = 1, \dots, m$. If S_j is empty for any j then GDC is inconsistent. Otherwise, use $SEIP$ with input (S_1, \dots, S_m) , $B = \{b_{jk}\}$, (L, d) and the root index m . $SEIP$ terminates infeasible if and only if GDC is inconsistent. Otherwise, *Phase 2* constructs a

feasible solution to GDC .

The correctness of this approach is well justified by observing that none of the proofs in Section 3 depend on the particular structure of the sets S_j .

We conclude that $SEIP$ handles all versions of distance constraints where new facilities may be restricted to arbitrary finite or infinite subsets of L provided that LN_B is a tree (or forest).

3.5 Cyclic LN_B

An immediate idea to extend the method we propose to cyclic LN_B is to apply expand/intersect operation recursively until final sets reach their smallest sizes. For this we may devise two different approaches as follows:

(1) Replace each arc (N_j, N_k) of LN_B with a pair of directed arcs (N_j, N_k) and (N_k, N_j) with lengths $b_{jk} = b_{kj}$ respectively. At each iteration, starting from a given 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.

(2) Apply $SEIP$ on different spanning trees of LN_B . At each iteration take the previously constructed sets as input to the new spanning tree processing. Continue this process until no set is changed from one iteration to other (or for all spanning trees of LN_B).

If DC is consistent, at some step all the distance constraints will be satisfied for all sets and no set will be trimmed further. So these processes should stop with nonempty sets after some finite number of steps.

The basic difficulty of the cyclic case comes from the fact that even if the final sets have been computed somehow, their being nonempty does not imply consistency of DC . Let F_j^* be the final sets constructed. Consider the following simple example (Figure 3.5) with 3 new facilities whose location space

is a simple cycle. 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 $F_3^* = \{x_3, x_6\}$. where the distance between two consecutive points is 1. Since the example is very simple, by some trial and error one can easily observe that no matter how and in which order he/she applies expand/intersect operation the sets F_1^* , F_2^* , F_3^* cannot be decreased further. So F_1^* , F_2^* , F_3^* are all nonempty. However, DC is inconsistent.

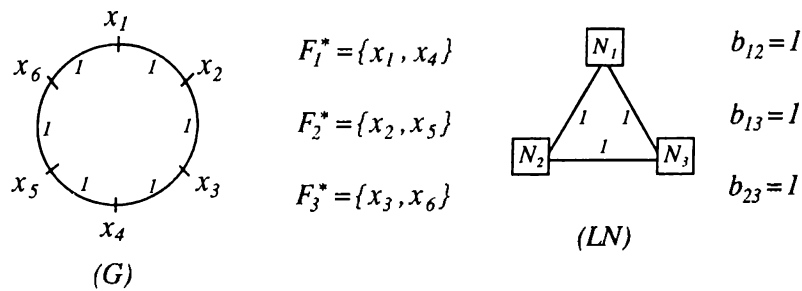


Figure 3.5: Example of Failure for Cyclic LN_B

In the example, even though the distance constraints are satisfied individually, there does not exist any feasible location vector which satisfy all three constraints simultaneously. This suggests that the basic drawback of expand/intersect operation is that it considers the locations of new facilities pair at a time while it tries to consider others implicitly. Because of the special structure of trees this approach proves to be valid for the case when the interaction between new facilities has a tree structure, but for cyclic LN_B the problem stands as an open question.

Chapter 4

COMPUTATIONAL METHODS FOR NETWORKS

So far we have given algorithms for general metric spaces. We now take L to be a network and specialize these algorithms to obtain strongly polynomial methods. Let G be an embedded network with vertex set $V = \{v_1, \dots, v_n\}$ and edge set E . For $x, y \in G$, $d(x, y)$ is the length of a shortest path connecting x and y . $d(\cdot, \cdot)$ defined in this manner is a distance function that satisfies the axioms stated earlier.

We need some additional notation. Given a point x in some edge $e = [v_s, v_t]$, the partial edges defined by x are denoted as $[v_s, x]$ and $[x, v_t]$. The length of edge $e = [v_s, v_t]$ is denoted as l_e or l_{st} . For $x \in [v_s, v_t]$, we define $\theta_e(x)$ to be the length of the partial edge $[v_s, x]$ and define $\theta'_e(x)$ to be the length of $[x, v_t]$. Note that $\theta_e(x) + \theta'_e(x) = l_e$ and that $0 \leq \theta_e(x), \theta'_e(x) \leq l_{st}$. For any two points x, y in $e = [v_s, v_t]$, $s < t$ with $\theta_e(x) \leq \theta_e(y)$, the interval $[\theta_e(x), \theta_e(y)]$ defines a *segment* $[x, y] = [v_s, y] \cap [x, v_t]$.

The next section focuses on the single facility case. The results for $DC(1)$ are needed later in the multifacility case.

4.1 Single Facility Case

When $m = 1$, distance constraints are simply $d(x, v_i) \leq c_i$, $i \in I = \{1, \dots, n\}$ and the feasible set is $S = \bigcap_{i \in I} N(v_i, c_i)$. $DC(1)$ is consistent if and only if $S \neq \emptyset$.

We construct S one edge at a time. That is, for each $e \in E$, we construct $S \cap e$, then find S by taking the union of $S \cap e$ over all $e \in E$. Since $S \cap e = [\bigcap_{i \in I} N(v_i, c_i)] \cap e = \bigcap_{i \in I} [N(v_i, c_i) \cap e]$, we first construct $N(v_i, c_i) \cap e \forall i \in I$, then find their intersection.

For notational simplification, let $N_e^i \equiv N(v_i, c_i) \cap e$ and let $d_{ij} = d(v_i, v_j) \forall i, j$. To compute N_e^i on a given edge $e = [v_s, v_t]$ of length l_{st} , we define the following parameters:

$$\alpha = \min \{d_{is}, d_{it}\} \quad (1a)$$

$$\beta = \max \{d_{is}, d_{it}\} \quad (1b)$$

$$\gamma = \min \{d_{is} + l_{st}, d_{it} + l_{st}, \frac{1}{2}(d_{is} + d_{it} + l_{st})\} \quad (1c)$$

where dependence of α, β, γ on indices i, s, t implicitly understood. Observe that $\alpha \leq \beta \leq \gamma$ where the first inequality is obvious. To justify $\beta \leq \gamma$, observe that $d_{is} \leq d_{is} + l_{st}$ (because $l_{st} \geq 0$); $d_{is} \leq d_{it} + l_{st}$ because d_{is} is the length of a shortest path between v_i and v_s while $d_{it} + l_{st}$ is the length of a path from v_i to v_s that visits v_t . To show $d_{is} \leq \frac{1}{2}(d_{is} + d_{it} + l_{st})$ we observe $d_{is} = \frac{1}{2}(d_{is} + d_{is}) \leq \frac{1}{2}(d_{is} + d_{it} + l_{st})$ where the last inequality is a consequence of $d_{is} \leq d_{it} + l_{st}$. Hence $d_{is} \leq \gamma$. A similar argument shows $d_{it} \leq \gamma$ proving that $\beta \leq \gamma$.

It now follows from $\alpha \leq \beta \leq \gamma$ that the distance bound c_i associated with v_i satisfies exactly one of the following inequalities :

- (i) $c_i < \alpha$
- (ii) $c_i \geq \gamma$
- (iii a) $\alpha \leq c_i < \gamma$ with $\alpha \leq c_i < \beta < \gamma$
- (iii b) $\alpha \leq c_i < \gamma$ with $\alpha \leq \beta \leq c_i < \gamma$

In case (i), neither v_s nor v_t is reachable from v_i within c_i distance units implying $N_e^i = \emptyset$.

In case (ii), we have $N_e^i = e$. To justify this, if $c_i \geq d_{is} + l_{st} = \gamma$, for any point x in e , we have $c_i \geq d_{is} + l_{st} \geq d_{is} + \theta_e(x) \geq d(v_i, x)$. Thus, $N_e^i = e$ under this assumption. If $c_i \geq d_{it} + l_{st} = \gamma$, then for any x in e , we have $c_i \geq d_{it} + l_{st} \geq d_{it} + \theta'(x) \geq d(v_i, x)$, implying again $x \in N_e^i \quad \forall x \in e$. If $c_i \geq \frac{1}{2}(d_{is} + d_{it} + l_{st}) = \gamma$, then for any $x \in e$, we have

$$\begin{aligned} 2d(x, v_i) &= 2 \min\{d_{is} + \theta(x), d_{it} + l_{st} - \theta(x)\} \\ &\leq [d_{is} + \theta_e(x)] + [d_{it} + l_{st} - \theta_e(x)] \\ &= d_{is} + d_{it} + l_{st} \leq 2c_i \end{aligned}$$

implying that $x \in N_e^i \quad \forall x \in e$.

In case (iiia), $\alpha \leq c_i < \beta$ implies one end point of e is reachable from v_i while the other is not. If v_s is reachable v_t is not, then $\alpha = d_{is} \leq c_i < d_{it} = \beta$ so that there is a unique point x in e for which $d_{is} + \theta_e(x) = c_i$. Let this point be called p_i^s . Clearly $p_i^s \neq v_t$ otherwise v_t is also reachable from v_i (via v_s). In the other case (with $\alpha = d_{it} \leq c_i < d_{is} = \beta$), define p_i^t to be the unique point x on e for which $d_{it} + \theta'_e(x) = c_i$. It is direct to conclude that $N_e^i = [v_s, p_i^s]$ if $d_{is} \leq c_i < d_{it}$ and $N_e^i = [p_i^t, v_t]$ if $d_{it} \leq c_i < d_{is}$.

In case (iiib), $\beta \leq c_i$ implies both endpoints of e are reachable from v_i so that p_i^s and p_i^t are both well defined. The fact that $c_i < \gamma$ implies the union of $[v_i, p_i^s]$ and $[p_i^t, v_t]$ is a proper subset of e , otherwise

$$\begin{aligned} d_{is} + d_{it} + l_{st} &\leq d_{is} + \theta_e(p_i^s) + d_{it} + \theta_e(p_i^t) \\ &= d_{is} + (c_i - d_{is}) + d_{it} + c_i - d_{it} = 2c_i \end{aligned}$$

which contradicts $c_i < \gamma$.

Hence we conclude $N_e^i = [v_i, p_i^s] \cup [p_i^t, v_t]$ where the two subarcs in the union are disjoint.

We summarize these in the next lemma.

Lemma 4.1 Given v_i and $e = [v_s, v_t]$ with α, β, γ defined in (1), we have:

- (i) $c_i < \alpha \Rightarrow N_e^i = \emptyset$
 - (ii) $c_i \geq \gamma \Rightarrow N_e^i = e$
 - (iiia) $\alpha \leq c_i < \beta < \gamma \Rightarrow N_e^i = [v_i, p_i^s] \quad \text{if } d_{is} = \alpha$
 and $N_e^i = [p_i^t, v_t] \quad \text{if } d_{it} = \alpha$
 - (iiib) $\alpha \leq \beta \leq c_i < \gamma \Rightarrow N_e^i = [v_i, p_i^s] \cup [p_i^t, v_t]$
- where $[v_i, p_i^s]$ and $[p_i^t, v_t]$ are disjoint. \square

We illustrate these results in Figure 4.1 with a simple example of 3 nodes.

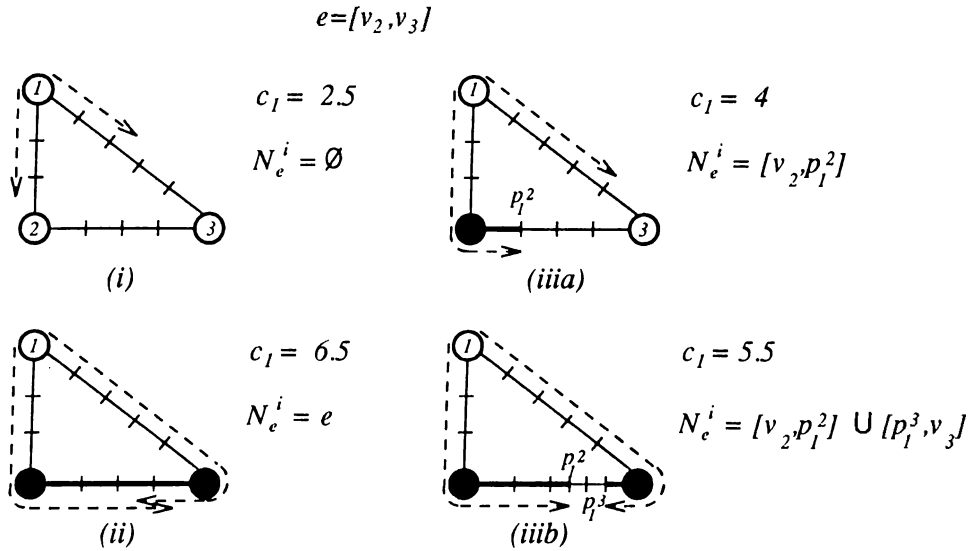


Figure 4.1: Illustration of $N_e^i \equiv N(v_i, c_i) \cap e$ for Various Possibilities

4.1.1 Construction of the Feasible set for $DC(1)$

We now return to the question of computing $S_e = \bigcap_{i \in I} N(v_i, c_i)$. Let $S_e = S \cap e$. Since $S = \bigcup_{e \in E} S_e$, it suffices to provide an algorithm for constructing S_e . We call this procedure the *Edge Restricted Sequential Intersection Procedure (ERSIP)*.

ERSIP (Constructs the part of the feasible set on the given edge)

(Input : $c_1, \dots, c_n, d_{ij} \forall i, j \in I, e = [v_s, v_t]$)

Initial. $i = 1, S_e = e.$

Main Step. Compute N_e^i using Lemma 4.1.

Assign $S_e \leftarrow S_e \cap N_e^i.$

If $S_e = \emptyset$ or $i = n$, go to *termination*.

Otherwise, assign $i \leftarrow i + 1$ and repeat the main step.

Termination Output $S_e.$

The updating procedure for S_e in the main step requires a record of segments of S_e . This is accomplished by keeping an ordered list of numbers $A_e = \langle a_1, \dots, a_k \rangle$ where initially $k = 2, a_1 = 0, a_2 = l_{st}$. The closed interval $[a_j, a_{j+1}]$ defined by two consecutive numbers a_j, a_{j+1} in the list, with j being *odd*, identifies a subedge $[x_j, x_{j+1}]$ which is a segment of S_e where x_j is the point on e for which $\theta_e(x_j) = a_j$ ($j = 1, \dots, k$). Note that, for some j which is odd, it is possible that $a_j = a_{j+1}$ in which case $[a_j, a_{j+1}]$ identifies a degenerate segment of S_e consisting of a single point $x_j = x_{j+1}$. Similarly, the open interval (a_l, a_{l+1}) with l being *even*, identifies a subedge (x_l, x_{l+1}) which lies outside of S_e .

When N_e^i is computed we have the following possibilities (from Lemma 4.1):

- (i) If $N_e^i = \emptyset$, then $S_e = \emptyset$ and we terminate.
- (ii) If $N_e^i = e$, then S_e remains unchanged.
- (iii) If $N_e^i = [v_s, p_i^s]$, we assign $a \leftarrow \theta_e(p_i^s)$. Two cases may occur; either $a \in [a_j, a_{j+1}]$ for some odd index j , in which case the intersection is identified by the numbers a_1, \dots, a_j of the list A and a , or $a \in (a_j, a_{j+1})$ for some even index j , in which case the intersection is identified by the numbers a_1, \dots, a_j of the list. So in either case we delete the elements a_{j+1}, \dots, a_k from A_e . If $a \in [a_j, a_{j+1}]$ and j is odd we insert a at the end

of the list A_e , and reassign $k \leftarrow j + 1$. If $a \in (a_j, a_{j+1})$ and j is even we leave the list as it is (i.e. $A_e = \langle a_1, \dots, a_j \rangle$ and reassign $k \leftarrow j$).

If $N_e^i = [p_i^t, v_i]$ a similar update procedure is applied that truncates A_e from the left (beginning). Specifically, we assign $a \leftarrow \theta_e(p_i^t)$ and find the odd (even) index j for which $a \in [a_j, a_{j+1}]$ ($a \in (a_j, a_{j+1})$). If $a \in [a_j, a_{j+1}]$ and j is odd then the intersection is identified by the numbers a, a_{j+1}, \dots, a_k , so we delete a_1, \dots, a_j from A_e , insert a at the beginning of A_e and reassign $k \leftarrow k - j + 1$. If $a \in (a_j, a_{j+1})$ and j is even then the intersection is identified by the numbers a_{j+1}, \dots, a_k , so we delete a_1, \dots, a_j from the list and reassign $k \leftarrow k - j$.

(iiib) If $N_e^i = [v_s, p_i^s] \cup [p_i^t, v_t]$ with $N_e^i = [v_s, p_i^s] \cap [p_i^t, v_t] = \emptyset$, we assign $a \leftarrow \theta_e(p_i^s)$, $b \leftarrow \theta_e(p_i^t)$ and identify the odd (or even) indices j and l for which $a \in [a_j, a_{j+1}]$ (or $a \in (a_j, a_{j+1})$) and $b \in [a_l, a_{l+1}]$ (or $b \in (a_l, a_{l+1})$). Note that $j \leq l$. Four cases may occur:

- (1) $a \in [a_j, a_{j+1}]$, $b \in [a_l, a_{l+1}]$ (both j and l are odd). Then the intersection is identified by the numbers a_1, \dots, a_j, a and b, a_{l+1}, \dots, a_k . So we delete a_{j+1}, \dots, a_l from A_e unless $j = l$ in which case no deletion occurs, insert a and b into the list in the order $a_j \leq a < b \leq a_{l+1}$ and reassign $k \leftarrow k - l + j + 2$.
- (2) $a \in [a_j, a_{j+1}]$, $b \in (a_l, a_{l+1})$ (j is odd, l is even). Then the intersection is identified by the numbers a_1, \dots, a_j, a and a_{l+1}, \dots, a_k . So we delete a_{j+1}, \dots, a_l from A_e , insert a between the numbers a_j and a_{l+1} in the list, and reassign $k \leftarrow k - l + j + 1$.
- (3) $a \in (a_j, a_{j+1})$, $b \in [a_l, a_{l+1}]$ (j is even, l is odd). Then the intersection is identified by the numbers a_1, \dots, a_j and b, a_{l+1}, \dots, a_k . So we delete a_{j+1}, \dots, a_l from A_e , insert b between the numbers a_j and a_{l+1} in the list, and reassign $k \leftarrow k - l + j + 1$.
- (4) $a \in (a_j, a_{j+1})$, $b \in (a_l, a_{l+1})$ (both j and l are even). Then the intersection is identified by the numbers a_1, \dots, a_j and a_{l+1}, \dots, a_k . So we delete a_{j+1}, \dots, a_l from A_e and reassign $k \leftarrow k - l + j$.

With this updating mechanism the current set S_e is uniquely represented by the intervals $[a_j, a_{j+1}]$, $j \in K \equiv \{j : 1 \leq j \leq k, j \text{ is odd}\}$ so that $S_e = \bigcup_{j \in K} [x_j, x_{j+1}]$ where x_i is the point on e for which $\theta_e(x_i) = a_i$, $i = 1, \dots, k$.

It is evident from the updating procedure that the list A_e can grow in size in a given iteration only if case (iiib-1) occurs with $j = l$. In that case A grows by two elements which implies the number of intervals grows by one. Hence the number of segments of S_e grows by at most one in a given iteration. Since we start with $S_e = e$, the number of segments in the terminating S_e is at most $n + 1$. This bound is attainable on edges that are redundant (an edge whose length is greater than the distance between the endvertices of the edge). In the absence of redundant edges, the final number of segments can at most be $n - 1$ because case (iiia) is the only case that can occur when $i = s, t$. The remaining $n - 2$ vertices may increase the number of segments one at a time which gives a maximum of $n - 1$ segments in the final set. Figure 4.2(a) illustrates a redundant edge of length 8 with $n + 1 = 6$ segments in the final set and (b) illustrates a nonredundant edge of length 4 with $n - 1 = 4$ segments.

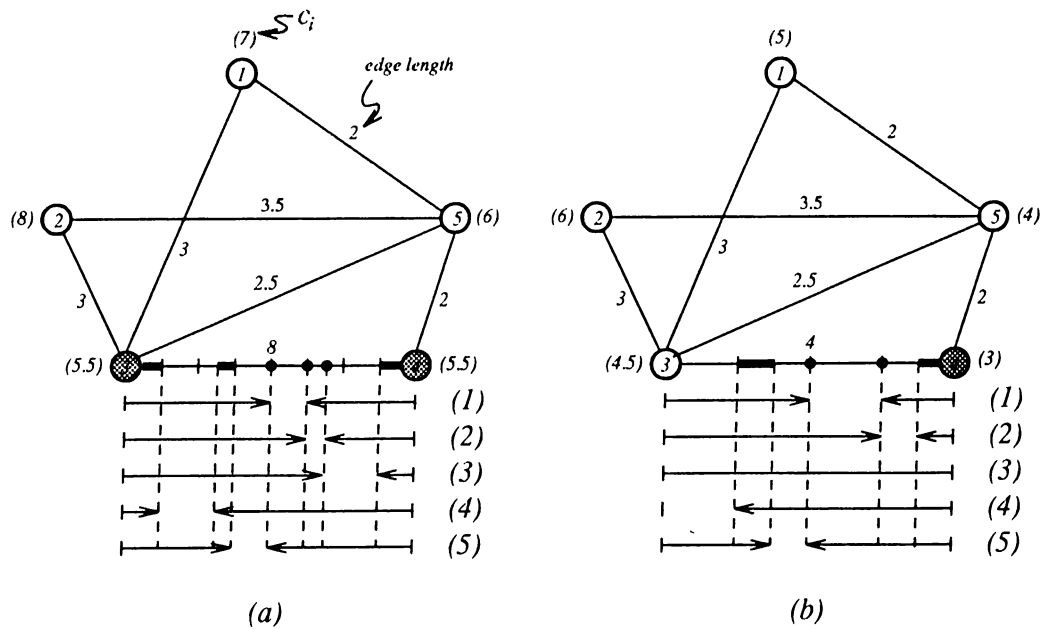


Figure 4.2: Example to Illustrate Segments on a Given Edge

Now, let $\#(S)$ represent the number of disjoint parts of a subset S of G . That is, if S is the union of k disjoint sets each of which is a maximal connected set, then $\#(S) = k$. Based on the foregoing analysis we have.

Theorem 4.1 *Let S be the feasible set to $DC(1)$. For all data choices, $\#(S)$ is at most $|E|(n + 1)$.*

Proof : Since $\#(S_e) \leq n + 1 \forall e \in E$, $S = \bigcup_{e \in E} S_e$ is such that $\#(S) \leq |E|(n + 1)$. \square

This theorem is significant because it identifies a deviation from the tree network case. It was proven in [8] that when the underlying network is a tree the intersection of all neighborhoods is a connected region (subtree). In the cyclic case the previous analysis together with the theorem shows that the intersection of all neighborhoods is in general a split region which may consist of up to $O(|E|n)$ number of disjoint parts.

The next theorem provides that we have a strongly polynomial algorithm to construct S .

Theorem 4.2 *Construction of S is $O(|E|n^2)$*

Proof : First we compute the time bound for constructing S_e for a given $e \in E$. Computation of N_e^i via Lemma 4.1 is done in constant time. Finding the appropriate indices j or l requires a comparison of a (or b) with the elements of A_e . This can be done in $O(\log k)$ time via binary search on A_e where k is the current size of the list A_e at each iteration. Insertion and deletion operations per item can be done in constant time. At a given iteration we insert at most two items into the list, but it can be the case that all the elements of the list have to be deleted so that a given iteration requires $O(k)$ operations. Since we have at most n iterations and k can increase by at most two at each iteration up to $2(n + 1)$, S_e is computed in $O(n^2)$. Repeating the procedure for all edges we get the stated result.

4.2 Multifacility Case

We now focus on the case $m > 1$. The particular form of the constraints defined by (DC.2) and (DC.1') provides considerable insight on some of the computational difficulties inherent in the problem. In the tree network case, each S_j is a subtree due to the convexity of neighborhoods $N(v_i, c_i)$, $i \in I_j$. When G is cyclic, there is a major problem: in general, each S_j 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 (Theorem 4.1). If we let S_j^k be the k -th disjoint subset in S_j (where $k = 1, \dots, \#(S_j)$), finding a feasible solution to DC calls for two decisions:

- (1) decide which S_j^k each x_j will be in,
- (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 from among $\prod_{j=1}^m \#(S_j)$ 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 $S_j^{k_j}$ for a unique index $k_j \in \{1, \dots, \#(S_j)\}$. Let $\bar{S}_j = S_j^{k_j}$ for the j -th new facility. We now have the restricted problem

$$\begin{aligned} d(x_j, x_k) &\leq b_{jk}, & (j, k) \in I_B, & & (DC.2) \\ x_j &\in \bar{S}_j, & j = 1, \dots, m & & (\overline{DC.1}) \end{aligned}$$

which we call \overline{DC} . \overline{DC} is more closely related to the tree network problem since each \bar{S}_j is now a connected set. Despite this resemblance, the restricted problem on G is nontrivial while the problem on a tree is efficiently solvable. As a matter of fact, we do not know of any method from the existing literature that attempts to solve \overline{DC} on general networks.

Our initial attempts to solve it on cyclic networks led us to believe that there are major computational difficulties caused by the presence of cycles which is why we found it appropriate to follow a different line of attack based on expand/intersect operations. The strength of the *SEIP* approach is its ability to circumvent the first decision on how to select S_j^{kj} for each j . Instead of doing that, the procedure goes directly into the construction of a feasible solution where each x_j is restricted to S_j rather than a particular disjoint set S_j^{kj} in S_j . While *SEIP* provides significant computational advantages in this regard, its inability to handle problems with cyclic LN_B is a computational barrier that must be addressed in future research.

We now focus on the specialization of *SEIP* to networks. To use *SEIP*, we must first compute S_1, \dots, S_m . Since S_j is the solution set of a single facility problem defined by the set of existing facilities i in I_j , the method in Section 6.1 constructs each S_j in $O(|E||I_j|^2)$ time. With $|I_j| \leq n \forall j$, the worst case time bound for constructing S_1, \dots, S_m is $O(|E|mn^2)$.

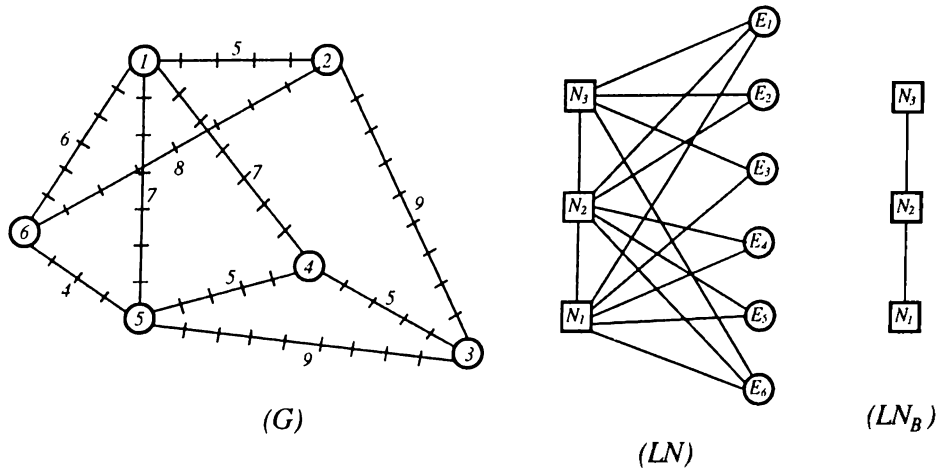
Before we go in to the computational details of performing an expand/intersect operation on a network, we provide an example to motivate the main ideas.

4.2.1 Example

Consider the example network G shown in Figure 4.3(a). The numbers next to edges are the edge lengths and the distance matrix is given. The distance bounds c_{ji} and b_{jk} are given in the matrices C and B in the same figure. This data defines the linkage network LN and its subgraph LN_B . The sets S_1, S_2, S_3 for this example are computed as described in Section 4.1 and they are shown in Figure 4.3(b) by means of lists A_{st}^j ($j = 1, 2, 3$) for each edge $[v_s, v_t]$ of G . *SEIP* processes nodes of LN_B in the order 1, 2, 3 (node 3 is the root). For example, in computing $N(S_1, b_{12})$, with $b_{12} = 5$, each list A_{st}^1 that defines $S_1 \cap [v_s, v_t]$ is updated appropriately to obtain a new list A_{st}^1 which now defines the edge restricted expansion $N(S_1, b_{12}) \cap [v_s, v_t]$. The specifics of how the updating is done will be explained in the next subsection, but the reader can verify the

new lists by moving from each endpoint of each segment of S_1 by $b_{12}(= 5)$ units in all possible directions in the network. Next, the intersection F_2 of $N(S_1, b_{12})$ with S_2 is computed on an edge by edge basis by updating the lists A_{st}^2 . This gives an edge by edge description of F_2 . Finally, $F_3 = N(F_2, b_{23}) \cap S_3$ is constructed by computing the expansion $N(F_2, b_{23}) \cap [v_s, v_t]$ for each edge, then constructing the intersection for each edge.

Once, F_1, F_2, F_3 are available, Phase 2 begins by selecting x_3 in F_3 (Figure 4.3(c), bottom). The selected x_3 is the single point of F_3 on edge $[v_1, v_5]$. Next, x_2 is selected as the nearest point in F_2 to x_3 . The selection of a nearest point is not a general rule, but it always works since $F_2(x_3) = N(x_3, b_{23}) \cap F_2$ is guaranteed to be nonempty and a nearest point in F_2 to x_3 is definitely in this set. The final solution (x_1, x_2, x_3) is shown in the figure.



	1	2	3	4	5	6
1	0	5	12	7	7	6
2	5	0	9	12	12	8
3	12	9	0	5	9	13
4	7	12	5	0	7	9
5	7	12	9	7	0	4
6	6	8	13	9	4	0

	N_1	N_2	N_3
E_1	7	13	7
E_2	-	7	9
E_3	11	-	12
E_4	7	13	-
E_5	5	10	-
E_6	9	8	10

	N_1	N_2	N_3
N_1	0	5	-
N_2	5	0	4
N_3	-	4	0

Figure 4.3: (a) Example for Multifacility Case

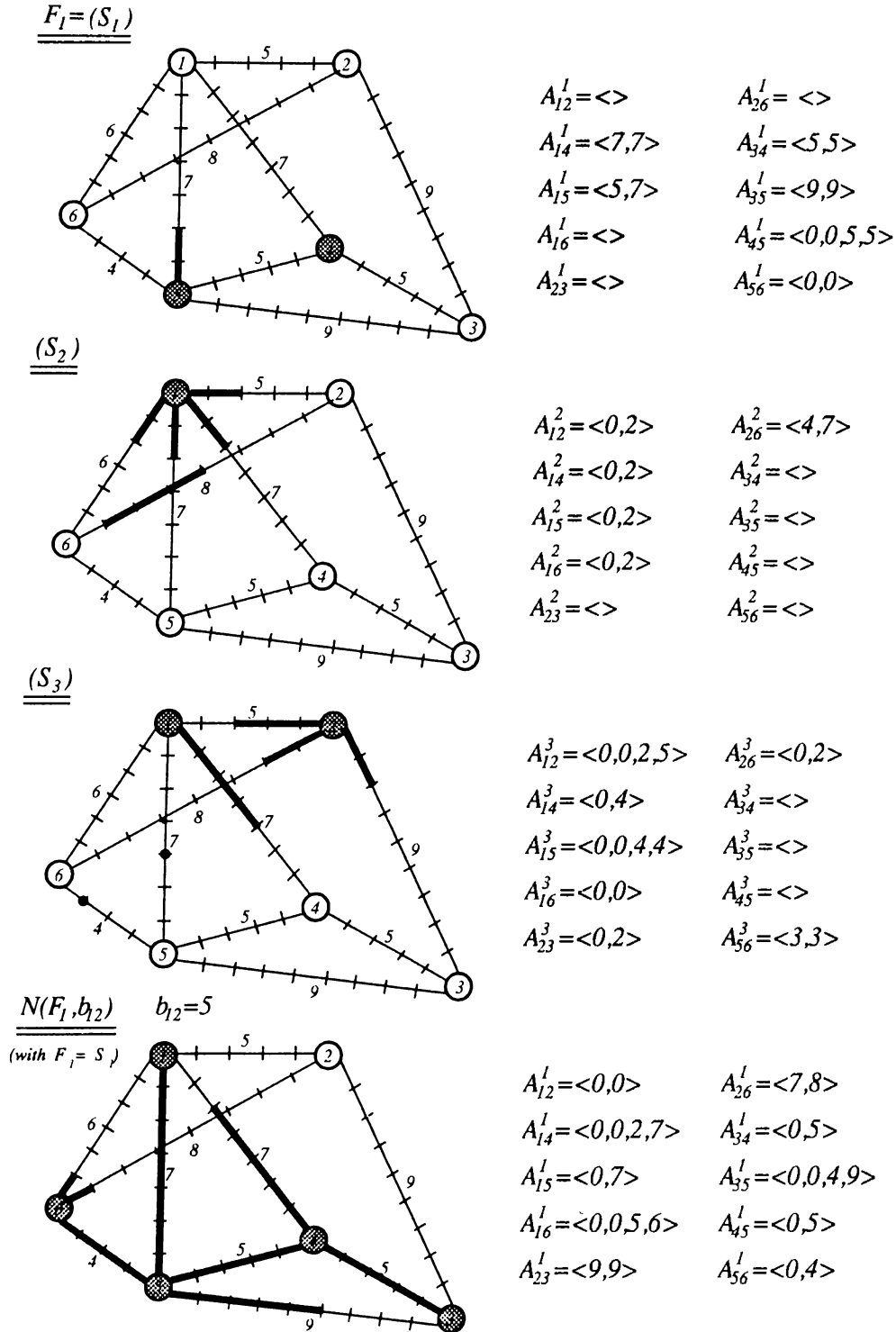


Figure 4.3: (b) Example Continued

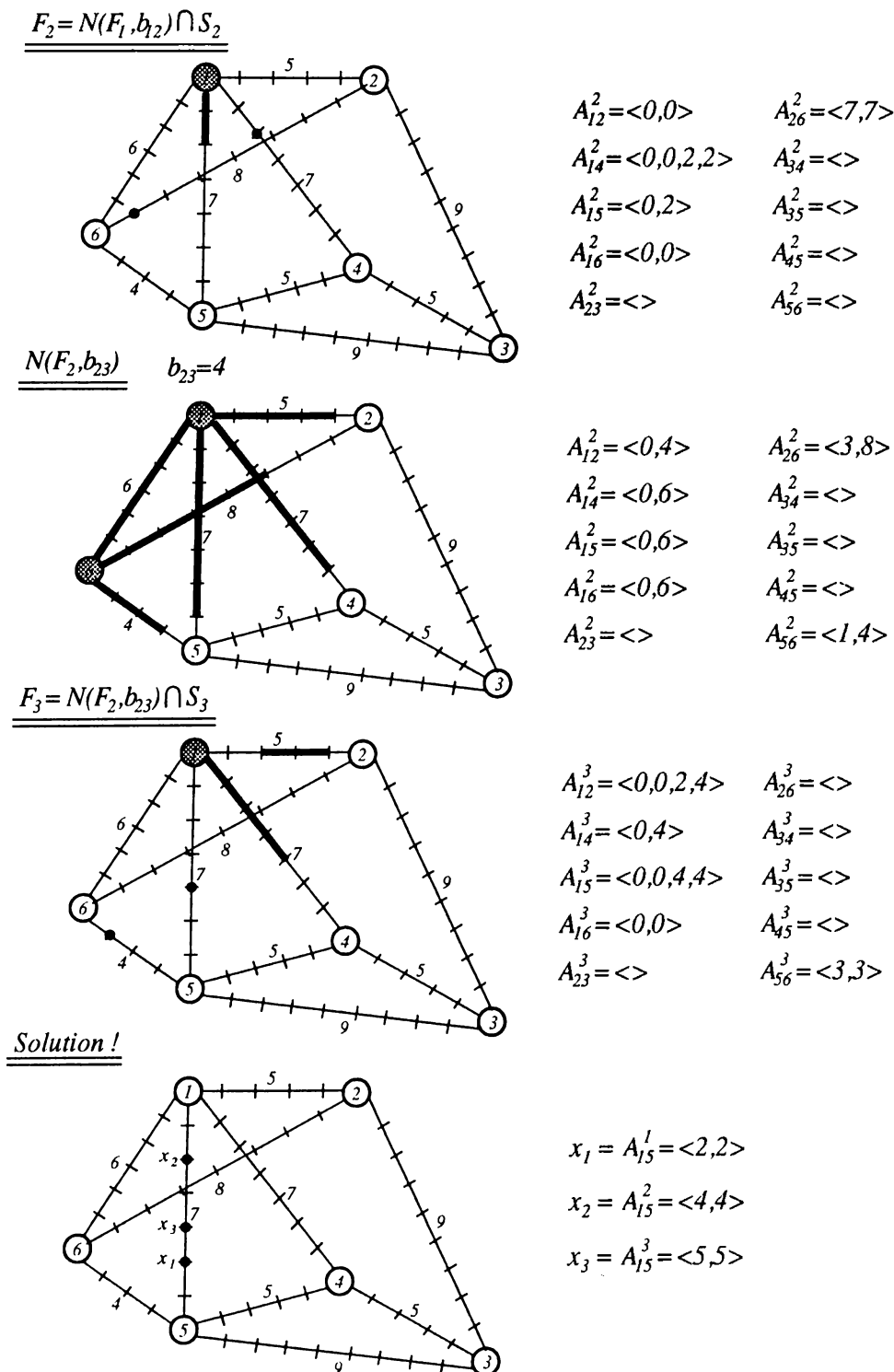


Figure 4.3: (c) Example Continued

4.2.2 EXPAND/INTERSECT Operation on Networks

SEIP uses the expand/intersect operation once for each arc of LN_B . To explain the mechanics of this operation, let S and S' be two subsets of G for which we want to compute $F' = N(S, b) \cap S'$ where S, S', b are given, F' is to be constructed.

The description of S is given by an ordered list $A_e = \langle a_1, \dots, a_{k(e)} \rangle$ for each edge e of E so that $S \cap e$ is the union of subedges $[x_j, x_{j+1}]$ where j is any odd index in $\{1, \dots, k(e)\}$ and x_j denotes the point on $e = [v_s, v_t]$ for which the subedge $[v_s, x_j]$ has length a_j . Note that $a_j \leq a_{j+1}$, $j = 1, \dots, k(e) - 1$ and $k(e)$ is an even integer. Similarly, the description of S' is given by ordered lists $A'_e = \langle a'_1, \dots, a'_{k'(e)} \rangle$, $e \in E$.

To construct F' , we go on an edge by edge basis. That is, we construct $F' \cap e$ for each $e \in E$, then find F' by taking the union of $F' \cap e$ over all $e \in E$. Since $F' \cap e = (N(S, b) \cap S') \cap e = (N(S, b) \cap e) \cap (S' \cap e)$, it suffices to describe the mechanics of computing the expansion of S by b restricted to edge e , then of computing the intersection with $S' \cap e$. For notational simplicity, let $F'_e = F' \cap e$, $N_e(S, b) = N(S, b) \cap e$, $S_e = S \cap e$, and $S'_e = S' \cap e$.

Preprocessing

Before we construct $N_e(S, b)$, we have a preprocessing step: For each vertex v_i we calculate a parameter δ_i which is defined as

$$\delta_i = \min_{x \in S} d(x, v_i) = \min_{e \in E} \delta_i(e)$$

where, for $e = [v_s, v_t]$,

$$\delta_i(e) = \min\{a_1 + d_{si}, l_{st} - a_{k(e)} + d_{ti}\}$$

with a_1 and $a_{k(e)}$ being the first and last elements of A_e (if the list is null, take $\delta_i(e)$ to be ∞). δ_i simply identifies the distance of a nearest point in S to a given vertex v_i while $\delta_i(e)$ is the distance of a nearest point in S_e to v_i .

To compute δ_i , we consider at most two numbers per edge which results in $2|E|$ operations for fixed i . All δ_i are computed in $2|E|n$ operations. Let $b_i = b - \delta_i$, $i = 1, \dots, n$. We will use b_i in the computation of $N(S, b)$.

Expansion

To compute $N_e(S, b)$ on edge $e = [v_s, v_t]$, we first focus on $N_e(S_e, b)$. Given the list $A_e = \langle a_1, \dots, a_{k(e)} \rangle$ that describe S_e , if A_e is null then $N_e(S_e, b) = \emptyset$. Otherwise, to compute $N_e(S_e, b)$ we update the nonempty list A_e as follows:

If A_e contains exactly two elements, go to (4), otherwise choose any two adjacent indices $l, l+1$ in $\{1, \dots, k(e)\}$ where l is even. Hence, a_l specifies the rightmost point of a segment of S_e and a_{l+1} specifies the leftmost point of another one with the open interval defining a subedge that lies outside of S_e .

- (1) If $a_{l+1} - a_l > 2b$, then set $a_l \leftarrow a_l + b$, $a_{l+1} \leftarrow a_{l+1} - b$.
- (2) If $a_{l+1} - a_l \leq 2b$, then delete a_l and a_{l+1} from A_e and assign $k(e) \leftarrow k(e) - 2$.
- (3) Repeat (1) and (2) for every even index l not considered so far then continue to (4).
- (4) If $b < a_1$, then assign $a_1 \leftarrow a_1 - b$; if $b \geq a_1$, then assign $a_1 \leftarrow 0$.
Similarly, if $b < l_{st} - a_{k(e)}$, then assign $a_{k(e)} \leftarrow a_{k(e)} + b$; if $b \geq l_{st} - a_{k(e)}$, then assign $a_{k(e)} \leftarrow l_{st}$.

The complexity of the above procedure is $O(k(e))$ where $k(e) = 2\#(S_e)$. Hence we construct $N_e(S_e, b)$ in $O(\#(S_e))$.

Remark 4.1 *At the end of the above procedure, if $b < l_{st}$, then*

$$a_i + b \leq a_{i+1} \quad i \in \{1, \dots, k(e)\}, \quad i \text{ odd,}$$

and if $b \geq l_{st}$, then $A_e = \langle 0, l_{st} \rangle$ (assuming the initial list is nonempty).

It is direct to verify that $N_e(S, b) = N_e(v_s, b_s) \cup N_e(v_t, b_t) \cup N_e(S_e, b)$ (in the event b_s or b_t is negative, the corresponding neighborhood is taken to be null).

Let A_e be the final list that represents $N_e(S_e, b)$. To obtain the final construction $N_e(S, b) = N_e(v_s, b_s) \cup N_e(v_t, b_t) \cup N_e(S_e, b)$, given the values b_s, b_t , we construct $N_e(S_e, b) \cup N_e(v_s, b_s)$ first and then construct $N_e(S_e, b) \cup N_e(v_s, b_s) \cup N_e(v_t, b_t)$. For $N_e(S_e, b) \cup N_e(v_s, b_s)$ we update the list A_e as follows :

- (1) If $b_s < 0$, then $N_e(v_s, b_s) = \emptyset$, so A_e remains the same.
- (2) If initially $A_e = \langle \rangle$ (i.e. $N_e(S_e, b) = \emptyset$), then insert $a_1 = 0$ and $a_2 = \min\{b_s, l_{st}\}$ into the list and assign $k(e) = 2$.

Otherwise, if $b_s \geq 0$ then two cases may occur; either $b_s < a_1$ in which case the union is identified by the numbers $0, b_s, a_1, \dots, a_{k(e)}$, or $b_s \geq a_1$ in which case the union is identified by the numbers $0, a_2, \dots, a_{k(e)}$ (due to Remark 4.1). So, in the first case, we insert 0 and b_s at the beginning of the list in the order $0 \leq b_s < a_1$ and reassign $k(e) \leftarrow k(e) + 2$. In the second case, we only reassign $a_1 \leftarrow 0$.

Next, to compute $N_e(S_e, b) \cup N_e(v_s, b_s) \cup N_e(v_t, b_t)$ we update the list A_e (currently representing $N_e(S_e, b) \cup N_e(v_s, b_s)$) as follows:

- (1') If $b_t < 0$, then $N_e(v_t, b_t) = \emptyset$, so A_e remains the same.
- (2') If initially $A_e = \langle \rangle$ (i.e. $N_e(S_e, b) \cap N_e(v_s, b_s) = \emptyset$), then insert $a_1 = \max\{0, l_{st} - b_t\}$ and $a_2 = l_{st}$ into the list and assign $k(e) = 2$.

Otherwise, if $b_t \geq 0$ then two cases may occur; either $b_t < l_{st} - a_{k(e)}$ in which case the union is identified by the numbers $a_1, \dots, a_{k(e)}, l_{st} - b_t, l_{st}$, or $b_t \geq l_{st} - a_{k(e)}$ in which case the union is identified by the numbers $a_1, \dots, a_{k(e)-1}, l_{st}$ (due to Remark 4.1). So, in the first case, we insert $l_{st} - b_t$ and l_{st} at the end of the list in the order $a_{k(e)} < l_{st} - b_t \leq l_{st}$ and reassign $k(e) \leftarrow k(e) + 2$. In the second case, we only reassign $a_{k(e)} \leftarrow l_{st}$.

Figure 4.4 gives an illustrative example for the expansion operation where the top figure gives the initial set S_e , the middle figure gives the expansion

$N_e(S_e, b)$, and the bottom figure gives the final construction $N_e(S, b)$.

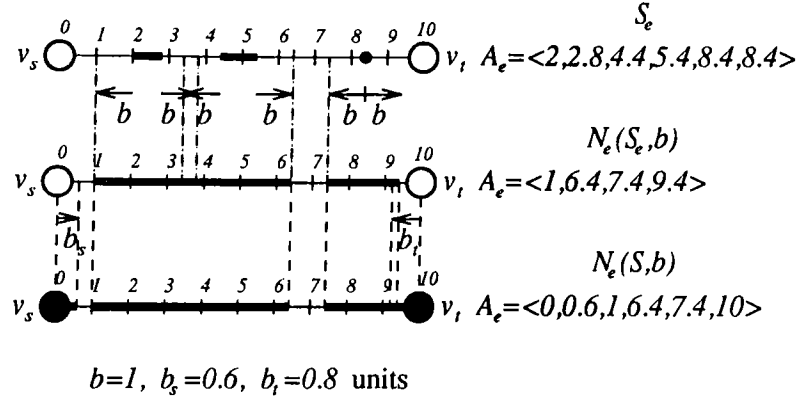


Figure 4.4: Illustration of Expansion Steps for S_e .

Updating in (1)-(2) and in (1')-(2') is done in constant time. So the complexity of constructing $N_e(S_e, b)$ (which is $O(\#(S_e))$) dominates. Hence we construct the expansion $N_e(S, b)$ in $O(\#(S_e))$ time.

The final list A_e updated by (1), (2), (1') and (2') represents the expansion $N_e(S, b)$. It is direct to observe that the number of elements in the final A_e is increased by at most four implying that the number of segments is increased by at most two. Also, observe that an increase is possible only if either $0 \leq b_s < a_1$ or $0 \leq b_t < l_{st} - a_{k(e)}$ or both. We summarize these in the next observation.

Observation 4.1 $\#(N_e(S, b)) \leq \#(S_e) + 2$

Intersection

With $N_e(S, b)$ constructed as described so far, now we must construct $N_e(S, b) \cap S'_e$ to obtain F'_e . This is accomplished by combining the two lists $A_e = \langle a_1, \dots, a_{k(e)} \rangle$ and $A'_e = \langle a'_1, \dots, a'_{k'(e)} \rangle$ representing the sets $N_e(S, b)$ and S'_e , respectively. To do so, we perform the intersection from the nearest point of both sets to a given endvertex, say v_s , of the edge $e = [v_s, v_t]$. The first two elements in the current list defines the first segment that belongs to the set

defined by that list. For example, initially, $[a_1, a_2]$ is the first interval defining the first segment of $N_e(S, b)$. Likewise, $[a'_1, a'_2]$ is the first interval that defines the first segment of S' on e . To see if the two segments intersect, we compute $b_F = \max\{a_1, a'_1\}$ and $b_L = \min\{a_2, a'_2\}$, then form the partial ordered list $B_e = \langle b_F, b_L \rangle$ which captures the first segment of the intersection unless $b_L < b_F$ in which case B_e is null (yet) because $[a_1, a_2]$ and $[a'_1, a'_2]$ do not intersect. We then delete all elements in both lists that are less than or equal to the maximum of b_F or b_L ; that is, we delete the parts of $N_e(S, b)$ and S'_e that are considered so far. If the deletion causes a list to begin with an initially even indexed element, we insert the maximum of b_F and b_L at the first position of that list and continue to grow B_e in this way by comparing the first two members of both lists to see if they have an intersection.

The following procedure accomplishes the construction of F'_e as described above. We call this procedure *Edge Restricted Intersection (ERI)*.

ERI (Performs the intersection operation on a given edge)
 (*Input* : ordered lists A_e and A'_e representing $N_e(S, b)$ and S'_e respectively.)

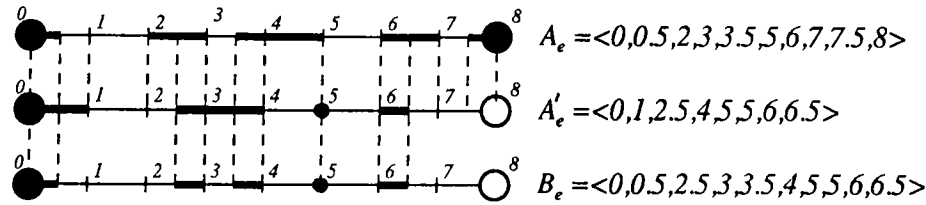
Initial. $B_e = \langle \rangle$.

Main Step. Assign $b_F \leftarrow \max\{a_1, a'_1\}$, $b_L \leftarrow \min\{a_2, a'_2\}$ and $b^* \leftarrow \max\{b_F, b_L\}$.
 If $b_F \leq b_L$, insert b_F, b_L at the end of the current list B_e , in the order $b_F \leq b_L$.
 Delete all elements of A_e and A'_e that are less than or equal to b^* and assign $k(e) \leftarrow k(e) - (\text{number of elements deleted from } A_e)$, $k'(e) \leftarrow k'(e) - (\text{number of elements deleted from } A'_e)$.
 If either list become empty, then go to *termination*.
 If $k(e)$ is odd, then insert b^* at the beginning position of A_e and assign $k(e) \leftarrow k(e) + 1$.
 Similarly, if $k'(e)$ is odd, then insert b^* at the beginning position of A'_e and assign $k'(e) \leftarrow k'(e) + 1$.
 Repeat the main step.

Termination. Output B_e .

We illustrate the above procedure in Figure 4.5.

Observation 4.2 *The maximum number of iterations of ERI is $\frac{k(e)+k'(e)}{2} - 1$.*



iter'n	b_F	b_L	b^*	B_e	A_e	$k(e)$	A'_e	$k'(e)$
1	0	0.5	0.5	<0,0.5>	<2,3,3.5,5,6,7,7.5,8>	8	<1,2.5,4,5,5,6,6.5>	7
							<0.5,1,2.5,4,5,5,6,6.5>	8
2	2	1	2		<3,3.5,5,6,7,7.5,8>	7	<2.5,4,5,5,6,6.5>	6
					<2,3,3.5,5,6,7,7.5,8>	8		
3	2.5	3	3	<0,0.5,2.5,3>	<3.5,5,6,7,7.5,8>	6	<4,5,5,6,6.5>	5
							<3,4,5,5,6,6.5>	6
4	3.5	4	4	<0,0.5,2.5,3,3.5,4>	<5,6,7,7.5,8>	5	<5,5,6,6.5>	4
					<4,5,6,7,7.5,8>	6		
5	5	5	5	<0,0.5,2.5,3,3.5,4,5,5>	<6,7,7.5,8>	4	<6,6.5>	2
6	6	6.5	6.5	<0,0.5,2.5,3,3.5,4,5,5,6,6.5>	<7,7.5,8>		<> STOP!	

$$B_e = \langle 0, 0.5, 2.5, 3, 3.5, 4, 5, 5, 6, 6.5 \rangle$$

Figure 4.5: Illustration of ERI.

It is evident that the elements of B_e is computed by scanning the elements of the union of A_e and A'_e from the beginning to the end (i.e. left to right). Because of the way b^* is calculated, at least three elements will be deleted from the union of elements in A_e and A'_e in a given iteration (a_1 and a'_1 will definitely be deleted and at least one of a_2 or a'_2 will also be deleted). However, we also insert b^* into the list A_e (A'_e) when $k(e)$ ($k'(e)$) is odd. Observe that if exactly three elements are deleted, two of them are deleted from one of the lists and the remaining one from the other. Hence, $k(e)$ and $k'(e)$ cannot both be odd at the same time in this case. So the number of insertions is at most one for the case of three deletions. If more than three deletions are made, the number of insertions is at most two. Thus, the number of deletions exceeds

the number of insertions by at least two in each iteration which implies that the total number of elements of the union of A_e and A'_e decreases by at least two in every iteration. Finally, observe that in the last iteration when one of the lists become empty we do not enter into the insertion (of b^*) step which implies that at least three elements are deleted from the union of the lists in the last iteration. Since at least one element remains in one of the lists that will not be processed, the maximum number of iterations of ERI is as stated in Observation 4.2.

Before proceeding further, observe that starting with B_e empty, at each iteration of ERI we insert either two (b_F and b_L) or no elements into the intersection list B_e . Since the maximum number of iterations of ERI is $(\frac{k(e)+k'(e)}{2} - 1)$, B_e can consist of at most $k(e) + k'(e) - 2$ elements (equivalently, $((k(e) + k'(e))/2) - 1$ segments) which implies that the number of segments in the intersection is bounded above by the total number of segments in the input sets minus one. That is,

Property 4.1 $\#(F'_e) = \#(N_e(S, b) \cap S'_e) \leq \#(N_e(S, b)) + \#(S'_e) - 1. \quad \square$

Complexity

Property 4.1 and the fact that expansion operation can increase the number of segments of the initial set S_e by at most two (Observation 4.1) implies the following important result:

Number of disjoint parts of F'_e constructed by an expand/intersect operation from S into S' on a given edge e can consist of at most the total number of segments of the initial sets S and S' plus one, i.e.

$$\#(F'_e) \leq \#(S_e) + \#(S'_e) + 1. \quad (4.11)$$

Observe that the upper bound on $\#(F'_e)$ is also an upper bound on the complexity of an expand/intersect operation on a given edge. So, the bound on

the number of segments of the sets constructed by expand/intersect operations in *SEIP* facilitates the analysis of complexity of the procedure.

We now give the complexity analysis of *SEIP* on the entire network.

Let $S_j(e)$ be the part of the feasible region of new facility j with respect to existing facilities that lies on a given edge, i.e. $S_j(e) = S_j \cap e$. From Section 6 we know that $\#(S_j(e)) \leq n + 1$. In each iteration of *SEIP* we apply an expand/intersect operation on two subsets of G related to two different new facilities. To construct the set F_k of a parent node k at a given iteration we apply the expand/intersect operation $|D_k|$ times (where D_k is the set of indices of the descendents of k), once with each descendent j of k . Then, it follows from (4.11) that

$$\#(F_k(e)) \leq \#(S_k(e)) + \sum_{j \in D_k} (\#(S_j) + 1). \quad (4.12)$$

Since initially $\#(S_j) \leq n + 1$ for all $j = 1, \dots, m$, we get

$$\#(F_k(e)) \leq (n + 1) + |D_k|(n + 2) = (|D_k| + 1)(n + 1) + |D_k|. \quad (4.13)$$

So, one application of expand/intersect operation on a given edge in some iteration of *SEIP* requires $O(|D_k|n)$ effort. Observe that $|D_k| \leq m - 1$ where equality is achieved for the root node only. So the complexity of one expand/intersect operation is bounded by $O(mn)$ for a given edge and bounded by $O(|E|mn)$ (this also dominates the complexity of preprocessing) for the entire network. Since we apply expand/intersect operation $m - 1$ times it follows that, given the initial feasible regions S_j , $j = 1, \dots, m$, the complexity of *Phase 1* of *SEIP* on general networks is $O(|E|m^2n)$. It is direct to observe that the complexity of *Phase 2* is also bounded by the same order.

Recall that the sets S_1, \dots, S_m are constructed with a time bound of $O(|E|mn^2)$. This result together with the foregoing analysis gives the next theorem.

Theorem 4.3 *If LN_B is a tree, the complexity of determining consistency or inconsistency of DC on arbitrary networks is $O(|E|mn(m + n))$.*

Since $L_j = F_j^*$ for all $j = 1, \dots, m$ in *Phase 2'* the number of disjoint parts of L_j on a given edge can at most be $O(mn)$ (by the fact that L_j can be constructed via *SEIP* by rooting the tree LN_B at node N_j) Hence construction of all regions of feasibility can be done within the same order of complexity.

Chapter 5

A MINIMAX EXTENSION

In this chapter we give an extension of the results provided in Chapters 3 and 4 to an optimization problem. The problem we deal with is the Multifacility Minimax Problem with Mutual Communication which have been introduced in Chapter 1.

Recall that the original formulation of the problem is as follows:

$$\begin{aligned} \text{Min} \quad & f(X) \\ \text{where } f(X) &= \max\{f_1(X), f_2(X)\} \\ \text{with } f_1(X) &= \max\{w_{ji}d(x_j, v_i) : (j, i) \in I_C\} \\ f_2(X) &= \max\{v_{jk}d(x_j, x_k) : (j, k) \in I_B\} \\ X = (x_1, \dots, x_m) &\in L^m \end{aligned}$$

where w_{ji} , v_{jk} are positive real numbers, and I_C and I_B specify pairs of indices for which the distances are of interest. The problem is to locate new facilities with respect to existing facilities and other new facilities so as to minimize the maximum of the weighted distances between specified pairs.

An equivalent formulation of the problem, which we denote by *PMMC*, in terms of distance constraints with parametric bounds is given as follows:

$$\begin{aligned}
& \text{Min } z \\
& \text{subject to} \\
& d(x_j, v_i) \leq z/w_{ji} \quad , \quad (j, i) \in I_C \qquad (PDC.1) \\
& d(x_j, x_k) \leq z/v_{jk} \quad , \quad (j, k) \in I_B \qquad (PDC.2) \\
& x_1, \dots, x_m \in L
\end{aligned}$$

Observe that if we fix the value of z then the problem simply results in the distance constraints (PDC.1) and (PDC.2). We denote by $PDC(z)$ the distance constraints as dependent on z .

Considering the parametric constraints (PDC.1) and (PDC.2), for a given value of z we can construct an auxiliary network $LN(z)$ as dependent on z , similar to the way we described earlier in Chapters 2 and 3. That is, $LN(z)$ consists of N -nodes representing new facilities and E -nodes representing existing facilities and arcs between them representing the pairs of facilities for which the distances are of interest. More formally we construct $LN(z)$ as follows:

Node set of LN is $\{N_1, \dots, N_m\} \cup \{E_1, \dots, E_n\}$ and there is an arc (N_j, E_i) of a new and existing facility pair for each $(j, i) \in I_C$ and there is an arc (N_j, N_k) of a pair of new facilities for each $(j, k) \in I_B$. The length of the arc (N_j, E_i) is z/w_{ji} and the length of the arc (N_j, N_k) is z/v_{jk} . Note that for a fixed value of z , $LN(z)$ is essentially the same as LN which have been described for the distance constraints with fixed bounds. Each arc length in $LN(z)$ is the product of a reciprocal weight and z . We denote by $LN_B(z)$ the subgraph of $LN(z)$ that spans only the new facility nodes and the arcs between them.

In the next section we provide an algorithm to solve $PMMC$ within an ϵ -interval of the optimal objective value. We again require that the interaction between new facilities has a tree structure, i.e. $LN_B(z)$ is a tree.

5.1 ϵ -Optimal Solution to *PMMC*

Our main approach here is to construct an ϵ -optimal solution to the optimization problem *PMMC* by applying a bisection search on the objective function value z .

Observe that for a fixed value of z the problem is to determine the consistency or inconsistency of $PDC(z)$. If $PDC(z)$ turns out to be inconsistent then this means that the current value of z is not large enough to admit a feasible solution so we have to increase the value of z . If $PDC(z)$ turns out to be feasible with alternate locations for all new facilities then we may conclude that the current value of z can be decreased further without violating the consistency of the distance constraints. Continuing in this manner at a specific value z^* of z , $PDC(z^*)$ will be feasible but for any $\epsilon \geq 0$ $PDC(z^* - \epsilon)$ will be infeasible. Clearly z^* is the optimum objective function value of *PMMC* which is also optimal to the original formulation of the Multifacility Minimax Problem with Mutual Communication.

Direct computation of z^* is not available in the literature except for the case when either the location space is a tree network or there is only a single new facility to locate. We could not find a way to compute the value of z^* directly as well.

Here we apply the idea of sequential decrease and/or increase of the value of z given in the previous paragraph with the hope of approaching sufficiently close to the optimal objective function value z^* . We set a prespecified value ϵ indicating how close we want to approach to optimum. We then take a sufficiently large value of z as upper bound \bar{z} and take lower bound \underline{z} to be "0", as zero is a natural lower bound on the objective value of the problem. We apply a bisection search on the objective value until we reach an ϵ -neighborhood of z^* . Each iteration in the search requires to solve a set of distance constraints with fixed bounds. For this we apply *SEIP* introduced in the previous chapters and that is why we require $LN_B(z)$ to be a tree. Note that depending on the location space better bounds can be obtained initially.

Following the same line of reasoning as in Chapter 3, for a fixed value of z we can get the following equivalent reformulation of $PDC(z)$:

$$d(x_j, x_k) \leq z/v_{jk} \quad , \quad (j, k) \in I_B \quad (PDC.2)$$

$$x_j \in S_j(z) \quad , \quad j = 1, \dots, m \quad (PDC.1')$$

where $S_j(z)$ is defined to be the region of feasibility for new facility j with respect to existing facilities only.

This reformulation enables to use the procedures introduced in Chapter 3. For a given value of z we can solve solve the given set of distance constraints $PDC(z)$ by first constructing the regions of feasibility with respect to existing facilities only, and then applying *SEIP* to incorporate the effect of the interaction between new facility pairs also.

Now, we state the algorithm which we call ϵ - *MINIMAX* to find the ϵ -optimum solution to *PMMC*.

ϵ -**MINIMAX** (Finds an ϵ -optimal solution to *PMMC*)

(Input : $\epsilon, \underline{z}, \bar{z}, \{w_{ji} : (j, i) \in I_C\}, \{v_{jk} : (j, k) \in I_B\}$)

Initial. $z^* \leftarrow (\underline{z} + \bar{z})/2$.

Main Step. If $(\bar{z} - \underline{z}) \leq \epsilon$ then \bar{z} is an ϵ optimal solution, go to *termination* with $z^* = \bar{z}$.

Otherwise, assign $z^* \leftarrow (\underline{z} + \bar{z})/2$, solve $PDC(z^*)$.

If $PDC(z^*)$ is inconsistent then assign $\underline{z} \leftarrow z^*$.

If $PDC(z^*)$ is consistent then assign $\bar{z} \leftarrow z^*$.

Repeat the main step.

Termination. Construct a feasible location vector to $PDC(z^*)$.

Observe that at any iteration the upper bound \bar{z} identifies a value of z which results in a consistent set of $PDC(z)$ and \underline{z} identifies a value of z which results in an inconsistent set of $PDC(z)$. At every iteration we either decrease the value of \bar{z} or increase the value of \underline{z} so that the interval $[\underline{z}, \bar{z}]$ containing optimal

objective value z^* is decreased at each step. When the length of the interval is less than or equal to ϵ , we conclude that the optimal value z^* is captured within an ϵ -neighborhood. Since at each iteration \bar{z} supplies a feasible solution while \underline{z} does not we set $z^* = \bar{z}$ and find a feasible location vector that satisfies $PDC(z^*)$.

At each iteration of $\epsilon - MINIMAX$ we decrease the current interval $[\underline{z}, \bar{z}]$ by half until the length of the interval is less than or equal to ϵ . So we apply k iterations where k is determined by the inequality:

$$2^k \geq \frac{(\bar{z} - \underline{z})}{\epsilon}$$

From the above inequality we conclude that the number of iterations k of ϵ -MINIMAX procedure is $O(\log((\bar{z} - \underline{z})/\epsilon))$. Let $h(L, d)$ be the time bound for solving a given set of distance constraints as dependant on the location space L with distance function d . Then finding an ϵ -optimal solution to $PMMC$ is $O(\log(\frac{\bar{z} - \underline{z}}{\epsilon})h(L, d))$.

In the next section we specialize to network spaces and provide exact computational bounds. We also provide better initial bounds to start the $\epsilon - MINIMAX$ Procedure.

5.2 Solving $PMMC$ on Networks

Let G be an embedded network with node set V and edge set E . Now, we consider the problem $PMMC$ defined on networks. As we have indicated earlier, rather than selecting a large value of z as the upper bound and taking 0 as the lower bound, depending on the location space we can find better bounds. So, here we provide such initial bounds for the case of general networks.

Observation 5.1 *Let $z_j = \min\{\max\{w_{ji}d(x_j, v_i) : (j, i) \in I_C\}\}$. Then $\underline{z} = \max\{z_j : 1 \leq j \leq m\}$ is a lower bound to $PMMC$ defined on a network G .*

Proof : It is direct to observe that \underline{z} is the solution to the problem *PMMC* for which the constraints (*PDC.2*) are relaxed. Hence any solution to this relaxation has a lower objective function value than the solution to the original *PMMC*. \square

The above observation is true for all metric spaces.

Observation 5.2 *Let T be any spanning tree of G and let $PMMC - T$ be the problem *PMMC* defined on T . Then the solution of $PMMC - T$ provides an upper bound on *PMMC* defined on G .*

Proof : Clearly $PMMC - T$ is a restricted version of the original *PMMC* defined on G where the location space is restricted to a spanning tree of G . Since *PMMC* is a minimization problem, the solution to the restricted version always has an optimum value which is at least as large as the optimum value of the original *PMMC* defined on G . \square

Of course, in order these bounds to be operational, we have to find a way to compute them efficiently.

For the case of the lower bound, each z_j is a solution to a *weighted 1-center* problem for new facility j . Kariv and Hakimi [13] provide an $O(|E|n \log n)$ algorithm to solve weighted one center of a network. We have to solve m different weighted one center problems corresponding to all new facilities. So computation of the lower bound for *PMMC* can be done in $O(|E|mn \log n)$ effort.

In the paper by Francis, Lowe, and Ratliff [8] the distance constraints are analyzed in detail on tree networks and the extension to the problem $PMMC - T$ is given. It is proven that the optimal objective value of $PMMC - T$ can be calculated directly as follows:

$$\bar{z} = \max\{d_T(v_s, v_t)/n_{st} : 1 \leq s < t \leq n\}$$

where $d_T(v_s, v_t)$ is defined to be the length of a shortest path between nodes v_s and v_t on the spanning tree T and n_{st} is defined to be the length of a shortest

path between two existing facility nodes E_s and E_t in the linkage network $LN(1)$ which is constructed as defined in the beginning of this chapter by taking the value of z as unity. Hence n_{st} is simply the sum of the reciprocal weights of the arcs of a shortest path $P(E_s, E_t)$ on $LN(1)$. It is well known that calculation of distance between all pairs of nodes on a tree is $O(n^2)$. The distances n_{st} between all existing facility node pairs in the linkage network can be calculated in $O(n(m+n)^2)$ time by applying Dijkstra's shortest path algorithm once for each E -node.

Both of these computational bounds are dominated by the construction of a feasible solution or determining the consistency or inconsistency of DC on networks which is proven to be $O(|E|mn(m+n))$. So the proposed ways to obtain good initial upper and lower bounds are justified computationally in the network case.

Finally from the analysis in this chapter and the results of Chapter 4, we conclude that construction of an ϵ -optimal solution to the Multifacility Minimax Problem with Mutual Communication defined on arbitrary networks is $O(\log \frac{(\bar{z}-z)}{\epsilon})|E|mn(m+n)$.

Chapter 6

SUMMARY AND CONCLUSION

In this thesis, we focused on the distance constraints problem. Consideration of distance constraints is well-justified in two respects: first, it is possible that in real life problems, scenarios that require bounds on the distances between facilities may arise; second, it facilitates the analysis of some optimization problems which have equivalent formulations in terms of distance constraints with parametric bounds.

Our particular interest is in the problem where the location space is a general network. With this motivation we introduced the distance constraints and a related optimization problem defined on networks.

We then provided a detailed survey of the related literature for the problem defined on networks. This survey showed that almost all the work done on networks restricted the location space to a tree network. There are very good theoretical results and algorithms developed for the tree network case, but the theory for general networks is virtually nonexistent. In this thesis, we have gone one step forward and solved a new class of instances of distance constraints on general networks. However, we are restricted to trees in another part of the problem data, because we require the structure of the interaction

between new facilities to be a tree. Those works that go beyond the tree case solve the related optimization problems and restrict the location of new facilities to finitely many points [2, 3]. However, they consider a broader class of instances in another sense, that is, they assume that the interaction between new facilities has a series-parallel [3] or a k -tree [2] structure. The basic result for general networks is Kolen's [12] proof stating that determining consistency or inconsistency of distance constraints is \mathcal{NP} -Complete.

In this study, we followed a different line of attack to the distance constraints problem. This was an entirely new approach and resulted in exact algorithms for determining consistency or inconsistency of the distance constraints and constructing a feasible solution if it exists, in any metric space for a class of instances where the interaction between new facilities has a tree structure with all other interactions being arbitrary. We determine the consistency or inconsistency of DC by an algorithm termed $SEIP$ which constructs a feasible location vector whenever it exists or concludes that DC is inconsistent. The basic idea is to apply expansion and intersection operations defined on the subsets of the location space, sequentially. We also address the problem of constructing the set of all feasible locations for each new facility, which has not been addressed in the literature except for the single facility case. A slightly altered version of $SEIP$ solves this problem with the same time bound.

The new approach is then applied to the case where the location space is a general network. Application of the new method to the problem defined on networks resulted in strongly polynomial algorithms. It has been proven that determining the consistency or inconsistency of DC and construction of a feasible solution if it exists can be done in $O(|E|mn(m+n))$ time on general networks. Also the sets of all feasible locations for all new facilities can be constructed within the same time bound.

Finally, we dealt with the related optimization problem, called the Minimax Problem with Mutual Communication. A bisection search has been proposed on the optimal objective value which, at each iteration, required solving a set of distance constraints with fixed bounds. For the network case, this approach

provided a polynomial algorithm for finding an ϵ -optimal solution.

There are basically three directions of further research:

The case of cyclic interaction structure between new facilities is still an open question and requires further research.

Since our approach is applicable to any metric space, application of this approach to other location spaces constitutes a second direction for future research.

The third direction for research is related to the regions of feasibility for all new facilities defined in Chapter 3. Since we can construct regions of feasibility efficiently it may be good to apply a sensitivity analysis on the distance bounds to see the behavior of the solution for different values. For the related optimization problem, sensitivity analysis may help to characterize some critical pairs that determine the optimal objective value z^* .

Bibliography

- [1] BRANDEAU, M. L., CHIU, S. S., "An Overview of Representative Problems in Location Research," *Managt. Sci.*, Vol. 35 (1989), pp. 645-674
- [2] CHHAJED, D. AND LOWE, T. J., "Solving Structured Multifacility Location Problems Efficiently," Faculty Working Paper 91-0170, Dept. of Buss. Adm., 350 Commerce West, Univ. of Illinois at Urbana Champaign, Champaign, IL61820.
- [3] ——— AND ———, " m -Median and m -Center Problems with Mutual Communication: Solvable Special Cases," *Oper. Res.*, Vol 40 (1992), pp. S56-S66.
- [4] DEARING, P. M., FRANCIS, R. L. AND LOWE, T. J., "Convex Location Problems on Tree Networks," *Oper. Res.*, Vol 24 (1976), pp. 628-642.
- [5] DOMSCHKE, W. AND DREXL, A., *Location and Layout Planning*, Lecture Notes in Economics and Mathematical Systems 238, Springer-Verlag, Berlin and New York, 1985.
- [6] ERKUT, E., FRANCIS, R. L., LOWE, T. J. AND TAMIR, A., "Equivalent Mathematical Programming Formulations of Monotonic Tree Network Location Problems," *Oper. Res.*, Vol. 37 (1989), pp.447-461.
- [7] ———, ——— AND TAMIR, A., "Distance Constrained Multifacility Minimax Location Problems on Tree Networks," *Networks*, Vol. 22 (1992), pp. 37-54.

- [8] FRANCIS, R. L., LOWE, T. J. AND RATLIFF, H. D., "Distance Constraints for Tree Network Multifacility Location Problems," *Oper. Res.*, Vol. 26(1978), pp. 570-590.
- [9] GAREY, M. R. AND JOHNSON, D. S., *Computers and Interactibility : A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.
- [10] HORN, W. A., "Three Results for Trees, Using Mathematical Induction," *J. Res. Nat. Bur. Stnds.*, Vol 76B (1972), pp. 39-43
- [11] LABBE, M., PEETERS, D. AND THISSE, J. F., "Location on Networks," Working Paper 92-13, European Institute for Advanced Studies in Management, Rue d'Egmont 13.B 1050 Brussel, Begium.
- [12] KOLEN, A. J. W., *Tree Network and Planar Location Theory*, Center for Mathematics and Computer Science, P.O. Box 4079, 10009 AB Amsterdam, the Netherlands, 1986.
- [13] KARIV, O. AND HAKIMI, S. L., "An Algorithmic Approach to Network Location Problems. Part I : The p -Centers," *SIAM J. Appl. Math.*, Vol. 37 (1979), pp. 513-538.
- [14] MEGIDDO, N., "Combinatorial Optimization with Rational Objective Functions," *Math. Oper. Res.*, Vol. 4 (1979), pp. 414-424.
- [15] TANSEL, B. Ç., FRANCIS, R. L. AND LOWE, T. J., "Binding Inequalities for Tree Network Location Problems with Distance Constraints," *Trans. Sci.*, Vol. 14 (1980), pp. 107-124.
- [16] ———, ——— AND ———, " A Biobjective Multifacility Minimax Location Problem on a Tree Network," *Trans. Sci.*, Vol. 16 (1982), pp. 407-429.
- [17] ———, ——— AND ———, " Location on Networks: A Survey. Part I : The p -Center and p -Median Problems," *Mangmt. Sci.*, Vol 29 (1983), pp. 482-497.

- [18] ———, ——— AND ———, " Location on Networks: A Survey. Part II : Exploiting Tree Network Structure," *Mangmt. Sci.*, Vol 29 (1983), pp. 498-511.