

To the Graduate Council:

I am submitting herewith a dissertation written by John Norman Hooker, Jr., entitled "Nonlinear Network Location Models." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Robert S. Garfinkel
Robert S. Garfinkel, Major Professor

We have read this dissertation
and recommend its acceptance:

James R. Vito
Richard E. Rosenthal
Carl G. Wagner
Kenneth C. Gilbert
Bruce A. Ralston

Accepted for the Council:

Lawrence
The Graduate School

NONLINEAR NETWORK LOCATION MODELS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

John Norman Hooker, Jr.

August 1984

Copyright © John Norman Hooker, Jr., 1984

All rights reserved

To my parents

ACKNOWLEDGMENTS

I wish to thank my major professor and colleague, Rob Garfinkel, for his guidance and astute criticism. I similarly benefited from an unusually large and diverse dissertation committee, consisting of Jim Ho, Ken Gilbert, Bruce Ralston, Rick Rosenthal and Carl Wagner. I received valuable suggestions in conversation with several other persons, including Ed Hillsman, Robert Jamison, Tim Lowe, John Mulvey and Doug Shier.

Special credit should go to Catherine Cole of Cole Printing and Word Processing Service for her skillful execution of a difficult typing job.

I also wish to thank my wife and son for their patience as I completed a dissertation in the midst of an already excessive work load. One dissertation would be bad enough, but this one is for a second doctoral degree!

ABSTRACT

Network location models can help identify minimal-cost locations for one or more facilities on a transport or communications network. Cost is a function of distance between facilities and demand points and may be reckoned as dollar cost, average or worst-case travel time, loss in patronage, social inequity or a variety of other measures of disutility. Techniques for finding least-cost solutions have been devised for only linear and a few other cost functions. The main object of this study is to describe a unified approach to solving such problems for a broad class of cost functions, particularly the nonlinear functions characteristic of public-sector location problems. It presents a reasonably efficient, computer-implemented algorithm for solving any single-facility problem in which cost is a convex function of distances, and a second algorithm for solving any small multiple-facility problem in which the cost function is convex and each demand point is served by a closest facility. Both algorithms accommodate a wide variety of side constraints. Numerous applications, especially in connection with social welfare problems, are discussed.

Another object of this study is to explore the metrical structure of networks. Such an exploration is the natural result of the fact that general nonlinear problems, unlike most network location problems hitherto investigated, must be solved "on the network." Networks, conceived as metrical structures rather than the discrete objects of graph theory, are abstracted as "graphic spaces," a type of metric space. A graphic

space is shown to preserve the essential property of networks that permits solution of nonlinear location problems, their decomposability into treelike pieces on which the metric is convex. The fact that a network is a tree if and only if it has a convex metric is generalized to graphic spaces. The algorithm for solving single-facility problems on networks is similarly generalized.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
1.1 The Problem and Its History	6
1.2 General Solution Approach	11
1.3 Convex Functions on Networks	14
2. THE SINGLE-FACILITY PROBLEM	18
2.1 The Problem	19
2.2 Solution of the General Convex Problem	20
2.3 Improvement with an Upper Bounding Technique	25
2.4 Semiseparable Functions	30
2.5 Solution of the Convex, Nondecreasing, Semiseparable Problem	41
2.6 Worst-Case Performance	48
2.7 Computational Experience	54
3. THE MULTIPLE-FACILITY PROBLEM	67
3.1 An Illustration	69
3.2 Solution of the General Convex Problem	76
3.3 Improvement with an Upper Bounding Technique	85
3.4 Solution of the Convex, Nondecreasing, Semiseparable Problem	96
4. A METRICAL THEORY OF NETWORKS	116
4.1 Convexity and Treelike Spaces	118
4.2 Convexity and Graphic Spaces	126
4.3 Finitely Decomposable Graphic Spaces	136
4.4 Models for Graphic Spaces	140

CHAPTER	PAGE
4.5 Convex Optimization on Metric Spaces	145
4.6 Solving Convex Location Problems in Graphic Spaces . .	148
4.7 Solving Convex, Nondecreasing, Semiseparable Location Problems in Graphic Spaces	161
5. APPLICATIONS	172
5.1 Applications Involving Transport Cost or Accessibility	174
5.2 Social Welfare Applications	178
5.3 Applications Involving Social Equality	188
5.4 Obnoxious Facility Location	192
5.5 Applications with Side Constraints	194
6. UNSOLVED PROBLEMS	198
LIST OF REFERENCES	201
APPENDIX	208
VITA	225

LIST OF TABLES

TABLE		PAGE
2-1	Computational Results for Five Random Networks, Showing Percent of Treelike Segments Examined and CPU Time in Seconds for the Center of Gravity and Cent-Dian Objective Functions	56
2-2	Computational Results for Five Random Networks on 40 Nodes, Showing Percent of Treelike Segments Examined and CPU Time in Seconds for the Center of Gravity and Cent-Dian Objective Functions	58
2-3	Computational Results for Four Rail and Water Networks, Showing Percent of Treelike Segments Examined and CPU Time in Seconds for the Center of Gravity and Cent-Dian Objective Functions	59
2-4	Computational Results for Three U.S. Interstate Networks, Showing Percent of Treelike Segments Examined and CPU Time in Seconds for the Center of Gravity and Cent-Dian Objective Functions	60
2-5	Effect on Segment Choice Parameter on Computation Speed, Showing Number of Segments Examined and CPU Time in Seconds for the Center of Gravity Problem on Two Networks .	62
2-6	Effect of Segment Choice Parameter on Computation Speed for Seven Actual Networks	63
2-7	Center of Gravity and Cent-Dians for Three U.S. Interstate Highway Networks	65
3-1	Application of Theorem 3.1 to the Example Described in the Text	94

LIST OF FIGURES

FIGURE	PAGE
2-1 Illustration of Treelike Sets	21
2-2 A Lower Bound for $z(x)$ on an Arc	29
2-3 Contour Plot of a Convex, Nondecreasing Function That Is Not Semiseparable	32
2-4 Counterexample of the Semiseparability of the Weighted Cent-Dian Function	41
2-5 Illustration of Theorem 2.2	43
2-6 Form of a Shortest-Path Tree $T(S)$ on G_w	52
2-7 Centers of Gravity and Cent-dians of Three U.S. Interstate Highway Networks	66
3-1 A Network for Illustration of a Two-Facility Location Problem	70
3-2 Diagram of Partition Zones, Treelike Rectangles, Treelike Regions, and Treelike Subregions for the Example of Fig. 3-1	71
3-3 Illustration of (3.2) in the Case Where $y_{qki} = w_q$	78
3-4 Illustration of Treelike Rectangles, Regions and Subregions	79
3-5 Partitioning of Nodes for Theorem 3.2	93
3-6 Illustration of Theorem 3.1	104
3-7 Representation of the Position of Facilities 1 and 2 in the Cartesian Product of Two Arcs	106
3-8 Extension of Theorem 3.1 to the Case in Which No Facilities Have Fixed Locations	107
4-1 Illustration of a Collection Point	120
4-2 Illustration of Theorem 4.1	121
4-3 A Space in Which Every Line Segment Is Treelike but the Metric Is Not Convex	122

FIGURE		PAGE
4-4	A Nontreelike Space in Which the Metric Is Convex	123
4-5	A Treelike Space in Which Not All Line Segments Are Paths	124
4-6	Illustration of Theorem 4.2	125
4-7	A Nontreelike Space with Convex Metric and Decomposable into Treelike Pieces	127
4-8	Illustration of Lemma 4.1	128
4-9	Illustration of Lemma 4.5	130
4-10	Illustration of Lemma 4.6	131
4-11	Illustration of Theorem 4.3, Case I	134
4-12	Illustration of Theorem 4.3, Case II	135
4-13	A Finitely Decomposable Graphic Space with Infinitely Many Maximal 2-Chains	139
4-14	A Finite Graphic Space That Is Not Modeled by a Subset of Any Network	141
4-15	Part of a Proposed Model for the Graphic Space of Fig. 4-14	142
4-16	A Graphic Space That Is Not Weakly Modeled by a Subset of Any Network	144
4-17	Illustration of Theorem 4.5	147
4-18	A Graphic Space Finitely Decomposable into Treelike Chains on Which the Metric Is Not Convex	149
4-19	A Graphic Space in Which a Line Segment Need Not Contain a Chain of Finitely Many Treelike Subspaces	151
4-20	Illustration of Lemma 4.9	152
4-21	Illustration of Lemma 4.10	153
4-22	Illustration of Case I in Procedure P	156
4-23	Illustration of Case II in Procedure P	157

FIGURE		PAGE
4-24	A Shortest-Path Tree in a Graphic Space	163
4-25	Illustration of Lemma 4.11	165
4-26	Illustration of Lemma 4.12	167
4-27	Illustration of Theorem 4.8	168
5-1	Typical Cost Function for an Obnoxious Facility Location Problem	193
A-1	Calling Relationships among Procedures of the Single-Facility Algorithm	210
A-2	Illustration for Procedure TSHIFT	220

CHAPTER 1

INTRODUCTION

Suppose that one is faced with the task of deciding where to locate one or more facilities on a transport network. One may want to locate a warehouse so as to minimize transport costs, a retail store so as to maximize patronage, several fire or ambulance stations so as to keep response times down, or a medical clinic so as to serve many people conveniently without being too far from anyone. A network location model can help one to make this sort of decision.

In such problems the goal is to find locations that minimize cost in some sense. Cost may be reckoned as dollar cost, number of customers lost, average or worst-case response time, social inequity (somehow quantified), or some other measure of disutility. Demand for access to the facilities is scattered over a network, and cost is a function of the distance or travel time between each demand point and the facility serving it. Normally longer distances mean higher cost, but if one wishes to locate an obnoxious facility such as a garbage dump or power plant, the reverse may be true.

Location problems need not be solved on a network. Some location models allow facilities to be located anywhere on the plane, and others (discrete models) restrict possible locations to a finite set of points. Network models are distinguished by their assumption that demand points and facilities are linked by some kind of transport or communications network. They therefore measure distances along a shortest route

through the network, rather than along a straight line, and they require that demand points and facilities be located along arcs (links) of the network. They usually suppose that all demand points are nodes (end-points of arcs), but this is not very limiting in practice, since one can arbitrarily create a node at any demand point.

Network location models have attracted a good deal of interest because they often strike a workable balance between simplicity and verisimilitude. Planar models tend to have solutions that are simple, intelligible and revealing of the problem structure, but their assumptions are unrealistic. In particular, they are unsuited to the many problems that have a reticular structure, and they can be very hard to solve when facilities are limited to certain regions of the plane. At the opposite pole are discrete models, which are extremely flexible because they permit one to choose any distance measure and any finite set of feasible locations he pleases. But their solution can be a difficult and complex affair that often reveals little about the problem other than an optimal solution (if that). Network models, on the other hand, are easily conceptualized and have a distance measure that suits a large number of applications. Provided the cost function is one for which solution techniques have been devised, their solution is generally easier and more informative than that of discrete models.

An important weakness of network models, however, has been the rather narrow range of cost functions for which solution techniques are available. Algorithms exist for problems in which cost is a weighted sum of distances (p -median problems) or in which cost is the longest

distance from which a demand point is served (p-center problems). We will see that progress toward more comprehensive solution techniques has been limited.

The main purpose of this study is to try to eliminate this weakness. We wish to describe a general approach to solving network location problems that is effective for the great majority of cost functions that are likely to arise. Specifically, we wish to present a single all-purpose algorithm, with computer implementation, that solves all single-facility network location problems (of reasonable size) in which cost is a convex function of distances. These include all problems that have hitherto been solved by a variety of specialized techniques, plus a good many more. We also intend to describe how to solve small multiple-facility problems in which cost is any convex function of distances and each demand point is served by a closest facility. These algorithms will have the added benefit that they permit one to restrict facilities to certain parts of the network and to impose a number of other constraints, a degree of flexibility not enjoyed by most existing algorithms. Since problems with linear cost functions have already been investigated, we may characterize our undertaking as a study of nonlinear network location models.

Solution techniques are the main goal, but we also desire theoretical understanding. We will see that study of nonlinear location problems invites a closer examination of the metrical properties of networks than is customary. This will lead us to characterize networks abstractly as a kind of metric space that preserves the essential properties

that permit solution of network problems. We will prove a number of theorems about such metric spaces and describe a general approach to solving location problems in them. We should emphasize, however, that the more concrete results concerning networks proper can be understood and appropriated in ignorance of these metric space results, which are an extension and generalization of the former.

It may be asked why we restrict our attention to models in which each demand point is served by a closest facility (an issue that obviously does not arise in a single-facility problem). One reason derives from our interest in prescriptive operations research models rather than the descriptive models characteristic of geography or regional science. A geographer generally (though certainly not always) wants to explain or predict the interaction between facilities and demand points, whereas we want to locate facilities and assign them to users in an optimal way. When the aim is to minimize cost, the most natural assignment is that of a demand point to a closest facility. In some optimization problems, as when several stores are located so as to maximize patronage, this may not be a realistic practice. It may also be infeasible, as when facilities have limited capacity. But problems with multiple assignments or constrained capacities require a substantially different approach, and it would be overly ambitious to try to solve them here.

Although this study does not take the descriptive approach often taken by geographers and regional scientists, it was undertaken in an effort to address a problem of central concern to them. This is the problem of how one should deal with public-sector location problems in

which the objectives are many and complex. The location of a fire hall, ambulance station, hospital, food stamp office, recreation center or city dump should be equitable as well as efficient. It often will not do simply to minimize total transport cost or average response time, because the way costs are distributed is also important. Unlike functions that compute total transport cost or average response time, functions that measure social cost in a sufficiently comprehensive sense are almost always nonlinear and therefore require nonlinear location models. Nonlinear cost functions also arise in private-sector applications, particularly in large organizations, where decision criteria are becoming more and more complex. Since no one had devised solution techniques for nonlinear models, aside from the p-center model and a few other special cases, it seemed important to do so.

Public sector location problems tend to be associated, in the regional science literature and in the public mind, with situations in which decisions are handed down from a government bureau. It is presumed that only in an orderly bureaucratic environment can location decisions be taken methodically with such aids as operations research models. Decisions thrashed out in the arena of public debate are thought to be at the mercy of "politics" and inherently irrational. We make no such presumption. Optimization models may serve not only as a guide to government (as well as private) decision makers but as a possible way of injecting rationality into public debate. Perhaps one reason public discourse is often incoherent is that no one has provided the

intellectual tools needed to assimilate the complex data and conflicting objectives that mark social welfare problems. An operations research model, sufficiently well conceived as to be clearly intelligible to the public, could be one of those tools.

In Section 1.1 below we state mathematically the location problem in question and present a brief history of efforts toward its solution. Section 1.2 outlines our own contribution. Section 1.3 contains a few fundamental results concerning convexity that will prove useful.

1.1 The Problem and Its History

Let G be the set of points on a network, and let $V = \{v_1, \dots, v_n\}$ be the set of nodes. Consider the arcs (v_i, v_j) to be rectifiable, and for any $x, y \in G$ let $d(x, y)$ be the distance from x to y along any shortest path from x to y . (Dearing and Francis [1974] provide a more rigorous characterization of a network.) If G^p is the p -fold cartesian product of G with itself, we can regard a vector $x \in G^p$ as a set of possible locations for p facilities. Denote by $d_x(y)$ the distance between $y \in G$ and a closest point in vector x . We wish to solve the network location problem,

$$\min_{x \in DCG^p} z(x) = f(d_x(v_1), \dots, d_x(v_n)) , \quad (1.1)$$

where $z: D \rightarrow R$ is the objective function and $f: R^n \rightarrow R$ is the cost function. Here the objective is to find a minimum-cost set of locations for p facilities, where each node (demand point) is served by a closest

facility. The single-facility problem is a special case and can be written,

$$\min_{x \in \underline{DCG}} z(x) = f(d(x, v_1), \dots, d(x, v_n)) . \quad (1.2)$$

Network location theory is usually said to have begun with Hakimi's papers [1964, 1965] dealing with the p-median and p-center problems. Since then a literature explosion has occurred, as indicated by the figures below, compiled by Tansel, Francis and Lowe [1983a]. These authors [1983a, 1983b] have published a comprehensive review of this literature. The network location literature is also discussed in the excellent survey of locational analysis in general by Francis, McGinnis and White [1983]. There is a text devoted exclusively to location on networks, by Handler and Mirchandani [1979].

<u>Period</u>	<u>Number of Publications</u>
before 1962	1
1962-1967	6
1968-1972	15
1972-1977	27
1978-1982	56

The p-median problem has a linear cost function we may call the median function,

$$f(d) = f(d_1, \dots, d_n) = \sum_{i=1}^n w_i d_i , \quad (1.3)$$

where $w_i \geq 0$, $i = 1, \dots, n$. One of Hakimi's main results is that at least one optimal solution of any p-median problem is a set of nodes.

This illustrates a theme that runs through much subsequent work. It is that network location problems have generally been solved in those cases where candidates for optimal locations can be limited to those in a finite set, such as the set of nodes. In such a problem one need only examine the network long enough to identify the nodes, candidate points and the matrix of distances between them. Thereafter one can generally forget about the network and concentrate on the matrix. Consequently problems of this sort tend to be treated more as integer programming problems than as problems with network structure. Solution techniques in this vein for the p -median problem have included the node partition heuristic of Maranzana [1967]; the greedy heuristics of Kuehn and Hamburger [1963]; the node substitution algorithms of Teitz and Bart [1968], Cornuejols, Fisher and Nemhauser [1977], and Khumawala, Neebe and Dannenbring [1974]; the branch and bound technique of Järvinen, Rajala and Sinervo [1972]; the linear programming approaches of Revelle and Swain [1970], Garfinkel, Neebe and Rao [1974], and Schrage [1975]; and the Lagrangian relaxation methods of Cornuejols et al. [1977], Diehr [1972], Neebe and Rao [1975], and Erlenkotter [1976]. A few techniques do exploit network structure, including Goldman's [1969, 1971] early algorithms for the 1-median problem and the algorithms for tree networks mentioned below.

It is important to note that since most p -median algorithms examine only the matrix of distances between nodes, these algorithms can solve any problem in which the optimal locations are known or constrained to be at nodes, provided the cost function is additively separable—i.e., it has the form,

$$f(d) = \sum_{i=1}^n g_i(d_i) . \quad (1.4)$$

One need only replace the matrix of distances $d(v_i, v_j)$ with a matrix of costs $g_j(d(v_i, v_j))$ (Handler and Mirchandani [1977, p. 68]). For example, Levy [1967] has shown that if each g_i in (1.4) is concave, then at least one optimal location in a single-facility problem is a node; Handler and Mirchandani [1977, p. 27] extend the result to p facilities. It follows that most p -median algorithms solve problems with additively separable concave costs. These same algorithms solve any problem in which the facilities are constrained to be at nodes, provided again the cost function is additively separable.

The p -center problem has a cost function we may call the center function,

$$f(d) = \max_{1 \leq i \leq n} \{w_i, d_i\} , \quad (1.5)$$

where again $w_i \geq 0$, $i = 1, \dots, n$. This problem, too, is solved by isolating a finite set of points (not all nodes) that must contain the solution set. Solution techniques generally rely on a set-covering algorithm and include those of Christofides and Viola [1971], Garfinkel, Neebe and Rao [1977], and Hakimi, Schmeichel and Pierce [1978]. Aside from Goldman's [1972] 1-center algorithm and algorithms for trees (below), there is again little exploitation of network structure once the finite set is identified.

Obnoxious facility location problems are likewise solved by identifying a finite set of candidate locations (that happen to be identical with the treelike segment boundaries defined in Section 1.2), as in Goldman and Dearing [1975], Church and Garfinkel [1978], and Minieka [1983].

One type of network whose structure is exploited is a tree, the setting for a good deal of algorithmic and theoretical work in location theory. On the algorithmic side are the techniques of Dearing and Langford [1975], Picard and Ratliff [1978], and Kariv and Hakimi [1979b] for the p -median problem (the first two with "mutual communication") and those of Francis, Lowe and Ratliff [1978] and Kariv and Hakimi [1979a] for the p -center problem (the former with mutual communication). On the theoretical side, Dearing, Francis and Lowe [1976] have developed a theory of convex optimization on trees parallel to that on R^n . We reproduce in Section 1.3 their definition of a convex function on a network as well as their remarkable result that $d(\cdot, w): G \rightarrow R$ is convex for all $w \in G$ if and only if G is a tree. In many ways this result is a springboard for the present study. They also prove that $d: G \times G \rightarrow R$ is convex on G if and only if G is a tree, plus a number of related results. Lowe [1979] generalizes many of these results to multiple-objective problems on trees. Yet however satisfying may be results proved for trees, they are of limited utility, since most networks that occur in location problems are not trees.

We will see in Section 2.1 that investigation of nonlinear problems (other than the p -center problem) on networks other than trees is quite

limited. For trees, the nonlinear 1-center problem has been solved by Dearing [1977], Francis [1977], and Tansel, Francis, Lowe and Chen [1982]. Work on general networks has consisted of Halpern's [1976, 1978, 1980] investigation of the cent-dian problem, whose cost function is a convex combination of the center and median functions, and Shier and Dearing's [1983] conditions for local minima. No general approach to the solution of nonlinear problems has been suggested.

1.2 General Solution Approach

Our investigation of network location models has four main parts, represented by the ensuing Chapters 2 through 5. Chapters 2 and 3 describe solution techniques for the single- and multiple-facility problems, respectively. Chapter 4 explores some underlying theory and Chapter 5 some applications. We conclude in Chapter 6 with a brief review of some unsolved problems.

In Chapter 2 we propose a general approach to solving single-facility problems that begins with the well-known fact that these problems are relatively easy to solve on trees. This ease of solution traces to the convexity (in a sense to be defined) of the distance function on trees. Fortunately there is a simple way to decompose any network into treelike pieces on which the distance function is likewise convex, and in fact linear. We therefore recommend that network location problems on general networks be solved by solving convex subproblems on treelike components of the network. This approach works whenever the cost function is convex. Since the number of treelike

components grows with the cube of the number of nodes (in the worst case), we show how to avoid solving subproblems on all but a fraction of the components. One way is to use an upper-bounding technique based on the convexity of the cost function. Another is based on the surprising fact that if each subproblem is solved on a shortest-path tree rooted at a component rather than just on the component, it is possible to eliminate many components at once. The latter technique requires that the cost function be nondecreasing and have a property we call semiseparability. We round out the chapter by studying worst-case performance of the algorithms and testing a computer implementation of them.

In Chapter 3 we extend the treelike decomposition techniques to the exact solution of multiple-facility problems. We also show that the techniques for reducing the number of subproblems have analogues, if rather complex ones, in the p -facility case. There is no escaping the fact, however, that the nonlinear p -facility problem is intrinsically very hard and that our algorithms are likely to be practical only for small networks. But if a problem is too large to solve with an exact nonlinear algorithm, one can always approximate the solution by solving the problem restricted to nodes, using one of the p -median algorithms. It may be necessary to increase the density of nodes in areas where facilities are likely to be located. Although one may need a ten- or hundredfold increase in the number of nodes thusly to solve a problem on a network consisting of a few long arcs, a problem of this size can be solved exactly.

Similar reasoning weighs against designing a heuristic rather than an exact algorithm for the p -facility problem. If one is satisfied with an approximate solution, he may as well solve the problem on nodes with one of the existing p -median algorithms.

Both the single- and multiple-facility algorithms require a convex cost function, and admittedly not all cost functions are convex. But arguably the great majority of cost functions likely to arise in practice are either convex or else concave and additively separable. In the former case our nonlinear algorithms apply, and in the latter case most p -median algorithms apply. Also we observe in Chapter 5 that our tree-like decomposition technique can be useful even when the cost function is not convex.

We noted in the previous section that most existing solution techniques work with a matrix of distances between nodes and discrete candidate locations rather than with the network itself. A general nonlinear algorithm, however, must solve a problem "on the network" and so induces one to probe a network's metrical properties more deeply than usual. Traditionally, mathematicians have abstracted networks or graphs as discrete structures—in particular, as sets of ordered pairs of nodes or vertices. Each pair represents an arc or edge and is optionally given a length; Harary [1969] has suggested that the latter distinguish a "network" from a "graph." But our inquiry leads us to abstract networks in a much different way—as a type of metrical structure, in particular a metric space. In Chapter 4 we define a type of metric space that seems to capture a network's essential metrical properties and investigate it.

In so doing we show that two key properties, both important to the solution of location problems, are closely connected: decomposability into treelike pieces, and the fact that a network has a convex metric if and only if it is a tree. We also show how to solve location problems in these networklike metric spaces. In Chapter 6 we suggest still another line of inquiry that can result from conceiving of networks as metrical objects.

Chapter 5 is less a review of concrete situations that call for nonlinear location models than a survey of several nonlinear cost functions that may arise in applications. We indicate the extent to which our algorithms can accommodate these functions as well as several types of side constraints.

1.3 Convex Functions on Networks

In this section we define some notions relating to convex functions on networks and state or establish some of their fundamental properties. We will appeal to these definitions and properties repeatedly in the chapters to follow.

Again let G be the set of points on a network, $V = \{v_1, \dots, v_n\}$ the set of nodes, and $d(x,y)$ the distance between $x, y \in G$ along any shortest path from x to y . Following Dearing, Francis and Lowe [1976], for any $x, y \in G$ and any $\lambda \in [0,1]$, we say that $w \in L_\lambda(x,y)$ when $d(x,w) + d(w,y) = d(x,y)$ and $d(x,w) = \lambda d(x,y)$. Let the line segment $L(x,y)$ between x and y be the union of $L_\lambda(x,y)$ over all $\lambda \in [0,1]$. Let $D \subseteq G$ be a convex set when for all $x, y \in D$, $L(x,y) \subseteq D$. For any convex set define $g: D \rightarrow R$ to be a

convex function on D if for any $x, y \in D$ and any $\lambda \in [0, 1]$, $w \in L_\lambda(x, y)$ implies $g(w) \leq (1 - \lambda)g(x) + \lambda g(y)$.

The first and most important convexity result we will state is the following:

Theorem 1.1 (Dearing, Francis and Lowe [1976]). Given network G with shortest-path metric d , the function $d(\cdot, y): G \rightarrow \mathbb{R}$ is convex for every $y \in G$ if and only if G is a tree network.

The distance measurements of interest in the location models we are considering are those between a node and any other point. Thus a distance function d is a convex metric for our purposes if $d(\cdot, v): G \rightarrow \mathbb{R}$ is convex for all $v \in V$. Theorem 1.1 has a counterpart for convex metrics that can be demonstrated with a slightly altered version of the proof given by Dearing, Francis and Lowe.

Corollary 1.1. A network has a convex metric if and only if it is a tree.

Proof. If the network G is a tree, then the metric d is convex by Theorem 1.1.

Suppose then that d is a convex metric on G , and assume G is not a tree. Let C be a cycle of shortest length ℓ in G . Let v be a node on C , and choose points $x, y, z \in C$ such that $d(v, x) = d(y, z) = \ell/2$ and $d(v, y) = d(x, y) = d(z, v) = d(x, z) = \ell/4$. Then $x \in L_{\frac{1}{2}}(y, z)$. But $d(v, x) = \ell/2 > \frac{1}{2}d(v, y) + \frac{1}{2}d(v, z) = \ell/4$, which contradicts the premise that d is a convex metric. Hence G is a tree.

□

Corollary 1.1 implies that d is a convex metric on network G if and only if $d(\cdot, x)$ is convex for all $x \in G$. But in the graphic spaces of Chapter 4 the former property is weaker than the latter, and it is convenient to distinguish the two properties now.

Since we are considering location models with objective functions of the form (1.1), it is important to establish that certain compositions of cost functions f with distance functions are convex. Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be nondecreasing if $d'_i \geq d_i$, $i = 1, \dots, n$, implies $f(d'_1, \dots, d'_n) \geq f(d_1, \dots, d_n)$.

Lemma 1.1. If (G, d) has nodes $v_1, \dots, v_n$ and convex metric d , and if $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and nondecreasing, then $z: G \rightarrow \mathbb{R}$ given by $z(x) = f(d(x, v_1), \dots, d(x, v_n))$ is convex on G .

Proof. It suffices to show that for any $x, y \in G$, any $\lambda \in [0, 1]$ and any $w \in L_\lambda(x, y)$,

$$z(w) \leq (1 - \lambda)z(x) + \lambda z(y). \quad (1.6)$$

Since d is a convex metric, $d(w, v_i) \leq (1 - \lambda)d(x, v_i) + \lambda d(y, v_i)$ for $i = 1, \dots, n$. Thus since f is nondecreasing and convex, we have $z(w) = f(d(w, v_1), \dots, d(w, v_n)) \leq f((1 - \lambda)d(x, v_1) + \lambda d(y, v_1), \dots, (1 - \lambda)d(x, v_n) + \lambda d(y, v_n)) \leq (1 - \lambda)f(d(x, v_1), \dots, d(x, v_n)) + \lambda f(d(y, v_1), \dots, d(y, v_n)) = (1 - \lambda)z(x) + \lambda z(y)$, and (1.6) follows.

□

Another useful property of convex functions on $\mathbb{R}^n$ is the following.

Lemma 1.2. Let $g: R \rightarrow R^n$ be given by $g(t) = (g_1(t), \dots, g_n(t))$ where each g_i is linear, and let $f: R^n \rightarrow R$ be convex. Then $h: R \rightarrow R$, given by $h(t) = f(g(t))$, is convex.

Proof. Since each g_i is linear and f is convex, we have for any $t_1, t_2 \in R$ that $h((1 - \lambda)t_1 + \lambda t_2) = f(g((1 - \lambda)t_1 + \lambda t_2)) = f((1 - \lambda)g(t_1) + \lambda g(t_2)) \leq (1 - \lambda)f(g(t_1)) + \lambda f(g(t_2)) = (1 - \lambda)h(t_1) + \lambda h(t_2)$. This shows that h is convex.

□

Finally, we say that the problem

$$\min_{x \in D} z(x) \tag{1.7}$$

is a convex problem if D is a convex set and $z: D \rightarrow R$ a convex function. It is well known that if $D \subseteq R^n$, then a locally optimal solution of convex problem (1.7) is globally optimal. If we let G^p be the p -fold cartesian product of network G with itself, then we have,

Lemma 1.3 (Dearing, Francis and Lowe [1976]). If (1.7) is a convex problem and $D \subseteq G^p$, then a locally optimal solution of (1.7) is globally optimal.

CHAPTER 2

THE SINGLE-FACILITY PROBLEM

The object of this chapter is to develop a robust algorithm for solving any single-facility network location problem with a convex cost function. The feasible region may be defined as the union of any finite collection of arcs and closed intervals (of positive length) within arcs of the network.

Section 2.1 states the problem and reviews previous efforts toward its solution. Section 2.2 describes a general approach to solving the problem that decomposes the network into "treelike segments" and solves a convex line search problem on each segment. The remaining sections show how to reduce substantially the number of segments examined when the cost function has certain properties in addition to convexity. Section 2.3 describes an upper bounding technique that can be used when the cost function is differentiable or is the weighted maximum function; the technique can be extended to other nondifferentiable functions on a case-by-case basis. Sections 2.4 and 2.5 develop the notion of a semi-separable function and describe an especially efficient solution technique that applies when the cost function is nondecreasing and semiseparable. Section 2.5 also presents a formal solution algorithm that exploits both techniques. A computer implementation of this algorithm is described in the Appendix. Section 2.6 analyzes the algorithm's worst-case complexity, and the last section reports computational experience.

2.1 The Problem

Consider network G with node set $V = \{v_1, \dots, v_n\}$ and metric d . We wish to solve the network location problem,

$$\min_{x \in DCG} z(x) = f(d(x, v_1), \dots, d(x, v_n)), \quad (2.1)$$

where the cost function $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is convex, and D is the union of finitely many closed intervals on G .

As noted in Section 1.3 (Theorem 1.1 and Corollary 1.1), Dearing, Francis and Lowe [1976] have shown that d is a convex metric if and only if G is a tree. Thus by Lemma 1.1, if f is nondecreasing as well as convex, then z is convex on any tree G . This makes (2.1) relatively easy to solve on a tree (if $D = G$), since by Lemma 1.3 one need only find a local optimum. This can be done by "homing in" on a solution—that is, simply by moving in a direction along which z increases until no such direction can be found, as discussed by Handler and Mirchandani [1979, p. 180], Channing [1978] and Wolf [1977].

The present task, however, is to solve (2.1) on a general network, and aside from work on the 1-median and 1-center problems (Section 1.2), little has been accomplished toward this end. Halpern [1976, 1978, 1980] has studied the "cent-dian" (or "medi-center") problem, whose objective function is a convex combination of the 1-median and 1-center objective functions. This problem attempts to model situations in which one wishes to balance equity and efficiency goals. There is no obvious way to extend Halpern's work to a broader class of objective functions.

Shier and Dearing [1983] have provided necessary and sufficient conditions for local optima when the cost function is an L_p norm of distances ($1 \leq p \leq \infty$). Some of their results can be extended to other cost functions, but they describe no method for enumerating local optima so as to find a global optimum.

2.2 Solution of the General Convex Problem

We now present a general approach to the solution of (2.1) when the cost function f is any convex function. The approach is to decompose each arc of G into "treelike segments," and then to solve a convex line search problem on each treelike segment.

We first define the notion of a "treelike set" and later use it to define a treelike segment, which is essentially its closure. We will say that points x, y on an arc "belong to the same treelike set" if the shortest paths to any node from x are the same as those from y , except for any portion of the paths lying between x and y . In Fig. 2-1, for instance, x_1 and x_2 belong to the same treelike set, but x_2 , x_3 and x_4 belong to three distinct treelike sets. It is clear that in a tree, any two points interior to an arc lie in the same treelike set, whence the name "treelike set."

These ideas can be made precise as follows. Given arc $A = (u, w)$ connecting points u and w and points $x, y \in A$, let $A(x, y)$ denote the portion of A strictly between x and y , and let $A[x, y]$ denote the closure of $A(x, y)$. Let $b(x, y)$ be the length of $A[x, y]$, with $b = b(u, w)$. Then $x, y \in A(u, w)$ belong to the same treelike set if for every $v_1 \in V$,

Lemma 2.1. Let the nodes in V be numbered so that $s_1, \dots, s_n$ is a nondecreasing sequence. Then given arc A from u to w , points $x, y \in A(u, w)$ belong to the same treelike set if and only if

$$s_j = b(u, x) = b(u, y) \text{ for some } j \in \{1, \dots, n\}, \quad (2.4)$$

(in which case $x = y$), or

$$s_j < b(u, x) < s_{j+1} \text{ and } s_j < b(u, y) < s_{j+1},$$

$$\text{for some } j \in \{1, \dots, n-1\}. \quad (2.5)$$

Proof. Suppose first that (2.4) or (2.5) holds. Since (2.4) implies $x = y$, the lemma is trivially true. Thus it remains to show that (2.5) implies (2.2) for all $v_j \in V$. If $i \leq j$, we have from (2.5) and the above remarks that all shortest paths from both x and y contain w , so that (2.2) follows. A similar argument applies for $i > j$.

We now suppose that (2.2) holds for all i and wish to show that either (2.4) or (2.5) holds. Since $0 < b(u, x) < b$ and s_1 ranges from zero (e.g., when $v_1 = w$) to b (when $v_1 = u$), the following cases are exhaustive.

Case I. $b(u, x) = s_i$ for some $i \in \{1, \dots, n\}$. Here $u, w \in L(x, v_i)$, so that by (2.2) $u, w \in L(y, v_i)$ as well. Thus $b(u, y) = s_i$ and (2.4) follows.

Case II. $s_i < b(u, x) < s_{i+1}$ for some $i \in \{1, \dots, n-1\}$. Here all shortest paths from x to v_i contain w , and all from y to v_{i+1} contain u . The former and (2.2) imply that all shortest paths from y to v_i contain w , so that $s_i < b(u, y)$. By similar reasoning the latter implies $b(u, y) < s_{i+1}$, and (2.5) follows.

□

Lemma 2.1 has a number of useful corollaries. First we note that the relation of belonging to the same treelike set is obviously reflexive and symmetric; the lemma makes clear its transitivity. Thus we have,

Corollary 2.1. The relation of belonging to the same treelike set is an equivalence relation.

Thus the relation partitions the interior of any arc into treelike sets. Inspection of Lemma 2.1 also yields,

Corollary 2.2. Let $s'_0, \dots, s'_{m+1}$ be the distinct s_i 's in Lemma 2.1, in nondecreasing order, where $0 \leq m \leq n-2$. Let $z_0, \dots, z_{m+1}$ be the points of $A(u,w)$ such that $b(u, z_i) = s'_i$, $i = 0, \dots, m+1$. Then the treelike sets in $A(u,w)$ are the singletons $\{z_i\}$, $i = 1, \dots, m$ and the open intervals $A(z_i, z_{i+1})$, $i = 0, \dots, m$.

We now define a treelike segment to be the closure of any open treelike set. In Fig. 2-1 for instance, there are six treelike segments, bounded by the nodes and the three heavy tick marks. In general we have from Corollary 2.2,

Corollary 2.3. The treelike segments on an arc $A = A[u,w]$ consist of at least one and at most $n - 1$ adjacent closed intervals jointly covering A , with boundaries at $z_0, \dots, z_{m+1}$ (as defined in Corollary 2.2).

We can now solve (2.1) simply by minimizing $z(x)$ on each treelike segment of G and picking the smallest minimum. If for arc $A = A[u,w]$ we define $h: [0,b] \rightarrow R$ by

$$h(d(u,y)) = f(d(y,v_1), \dots, d(y,v_n)) \quad (2.6)$$

for any $y \in A$, then minimizing $z(x)$ on a treelike segment $A[x_1, x_2] \subseteq A$ is clearly equivalent to solving the line search problem,

$$\begin{aligned} \min \{h(t)\} \quad . \\ b(u, x_1) \leq t \leq b(u, x_2) \quad . \end{aligned} \quad (2.7)$$

We can see that (2.7) is a convex problem as follows. Suppose again that the nodes are ordered so that $s_1, \dots, s_n$ is a nondecreasing sequence. Then by Lemma 2.1 and (2.3) we have that for some $j \in \{1, \dots, n\}$,

$$\begin{aligned} f(d(y, v_1), \dots, d(y, v_n)) &= f(d(u, v_1) + t, \dots, d(u, v_j) \\ &\quad + t, d(w, v_{j+1}) + b - t, \dots, d(w, v_n) + b - t) \end{aligned} \quad (2.8)$$

provided $y \in A[x_1, x_2]$. Thus the arguments of f in (2.6) are linear functions of t . From this and the convexity of f we infer by Lemma 1.2 that h is convex on $[b(u, x), b(u, y)]$.

We now present an algorithm for solving (2.1). It incorporates a simple technique that permits one to exclude the facility from certain parts of the network. One can exclude the facility from any arc by considering that arc "infeasible" and passing over it in Step 1 of the algorithm. Also one can exclude the facility from an open interval within an arc by inserting a node at either end of the interval and regarding the new arc between the nodes as infeasible. Thus the

feasible region D may be defined as the union of any collection of arcs and closed intervals (of positive length) within arcs.

Step 0. Set the treelike segment counter k to zero.

Step 1. Choose any unexamined feasible arc $A = A[u, w]$ of G ; if none remain, go to Step 3. Compute $d(u, v_i)$ and $d(w, v_i)$ for $i = 1, \dots, n$, and use the results to compute $s_1, \dots, s_n$. Sort these in nondecreasing order and determine $s'_0, \dots, s'_{m+1}$ so as to locate the treelike segments on $A[u, w]$ as described in Corollary 2.3.

Step 2. For each treelike segment $A[x_1, x_2]$ on A , let $k = k+1$ and find a solution t^* of the convex line search problem (2.7); denote the corresponding point on A by x_k^* , and let $z_k^* = h(t^*)$. When all the segments of A are enumerated, go to Step 1.

Step 3. Any x_k^* such that $z_k^* = \min_j \{z_j^*\}$ solves (2.1).

2.3 Improvement with an Upper Bounding Technique

It is generally possible to improve substantially the solution technique of the previous section by using an upper bounding technique. The idea is simply to eliminate any arc on which we know that the objective function $z(x)$ cannot achieve a value lower than a previously established upper bound on its optimal value. The advantage here is that none of the treelike segments on an eliminated arc need be examined.

We show in this section how to proceed when f is differentiable or is the weighted maximum function. The method must be adapted to other nondifferentiable functions on a case-by-case basis.

When a treelike segment is examined, we obtain a locally optimal point x_k^* , with $z_k^* = z(x_k^*)$, as in the previous section. Let $z_{\min}$ be the minimum of all the z_k^* 's that have been found so far in the examination of treelike segments. Thus $z_{\min}$ is an upper bound on the global minimum of $z(x)$. When a new arc $A = A[u, w]$ is considered, we may be able to determine that $z(x)$ cannot fall below $z_{\min}$ on A , so that A may be eliminated (except nodes u or w , if either is already an x_k^* for some k). To see how this is done, let $d_i = d(u, v_i)$ and $d_i' = d(w, v_i)$ for $i = 1, \dots, n$, and let $f_i(d_1, \dots, d_n)$ be $\partial f / \partial d_i$ evaluated at $(d_1, \dots, d_n)$. When f is differentiable we have the following result.

Lemma 2.2. Let f in (2.1) be convex and differentiable, and let x lie on arc $A = A[u, w]$, with $t = b(u, x)$. If $v_1, \dots, v_m$ are the nodes v_i such that $s_i < b = b(u, w)$ on arc A , then

$$z(x) \geq z(u) - ta, \quad (2.9)$$

where

$$a = \sum_{i=1}^m |f_i(d_1, \dots, d_n)| - \sum_{i=m+1}^n f_i(d_1, \dots, d_n). \quad (2.10)$$

Proof. Note that for any $x \in A[u, w]$, there is a shortest path from v_i , $i = m+1, \dots, n$, to x that passes through u . Thus $d(x, v_i) = d_i + t$ for $i = m+1, \dots, n$. Also $d(x, v_i) = d_i - t_i$ for some t_i , $i = 1, \dots, m$, where $-t \leq t_i \leq t$. Thus

$$z(x) = f(d(x, v_1), \dots, d(x, v_n)) = f(d_1 - t_1, \dots, d_m - t_m, d_{m+1} + t, \dots, d_n + t) \quad (2.11)$$

But since f is convex and differentiable, we have for any vector d , $f(d + \Delta d) \geq f(d) + \nabla f(d) \cdot \Delta d$. Thus

$$\begin{aligned} f(d_1 - t_1, \dots, d_m - t_m, d_{m+1} + t, \dots, d_n + t) &\geq f(d_1, \dots, d_n) \\ &\quad - \sum_{i=1}^m t_i f_i(d_1, \dots, d_n) + t \sum_{i=m+1}^n f_i(d_1, \dots, d_n) \\ &\geq f(d_1, \dots, d_n) - t \sum_{i=1}^m |f_i(d_1, \dots, d_n)| \\ &\quad + t \sum_{i=m+1}^n f_i(d_1, \dots, d_n), \end{aligned}$$

where the last inequality follows from the fact that $-t \leq t_i \leq t$. This and (2.11) imply (2.9). □

We can obtain a similar result when f is the weighted maximum function, $f(d_1, \dots, d_n) = \max_i \{w_i d_i\}$, $w_i \geq 0$.

Lemma 2.3. Let f in (2.1) be the weighted maximum function, and let x , A , t and $v_1, \dots, v_m$ be as in Lemma 2.2. Then (2.9) holds when

$$a = \begin{cases} w_j & \text{if } 1 \leq j \leq m, \\ -w_j & \text{if } m+1 \leq j \leq n, \end{cases} \quad (2.12)$$

where $w_j d_j = \max_{1 \leq i \leq n} \{w_i d_i\}$.

Proof. It is clear that

$$\max \left\{ \max_{1 \leq i \leq m} [w_i(d_i - t)], \max_{m+1 \leq i \leq n} [w_i(d_i + t)] \right\} \leq \begin{cases} w_j d_j - t w_j & \text{if } 1 \leq j \leq m, \\ w_j d_j + t w_j & \text{if } m+1 \leq j \leq n. \end{cases} \quad (2.13)$$

Also

$$z(x) = \max_{1 \leq i \leq m} \{w_i d(x, v_i)\} \leq \max \left\{ \max_{1 \leq i \leq m} [w_i(d_i - t)], \max_{m+1 \leq i \leq n} [w_i(d_i + t)] \right\}.$$

This and (2.13) imply (2.9). □

Now we can state,

Theorem 2.2. Let f in (2.1) be convex, and let f either be differentiable or be the weighted maximum function. Let x lie on arc $A = A[u, w]$, with $t = b(u, x)$, and let $v_1, \dots, v_m$ be the nodes v_i such that $s_i < b_i$ on arc A . Let a be given by (2.10) or (2.12), and let a' be given by the same formula with d'_i replacing d_i , $i = 1, \dots, n$. Then

$$z(x) \geq z_0 = \begin{cases} z(n) - [z(u) - z(w) + a'b] \frac{a}{a + a'} & \text{if } a, a' > 0, \\ \min \{z(u), z(w)\} & \text{otherwise.} \end{cases} \quad (2.14) \quad (2.15)$$

Proof. By Lemmas 2.2 and 2.3 we have that $z(x) \geq z(u) - ta$ and $z(x) \geq z(w) - (b - t)a'$. These lower bounds describe two lines above which $z(x)$ must lie, as in Fig. 2-2, where $h(t)$ is as defined by (2.6). If $a, a' > 0$, then $z(x)$ is everywhere bounded below by z_0 if the lines

intersect at (t_0, z_0) . Setting $z(u) - t_0 a = z(w) - (b - t_0) a'$, we get $t_0 = [z(u) - z(w) + b a'] / (a + a')$.

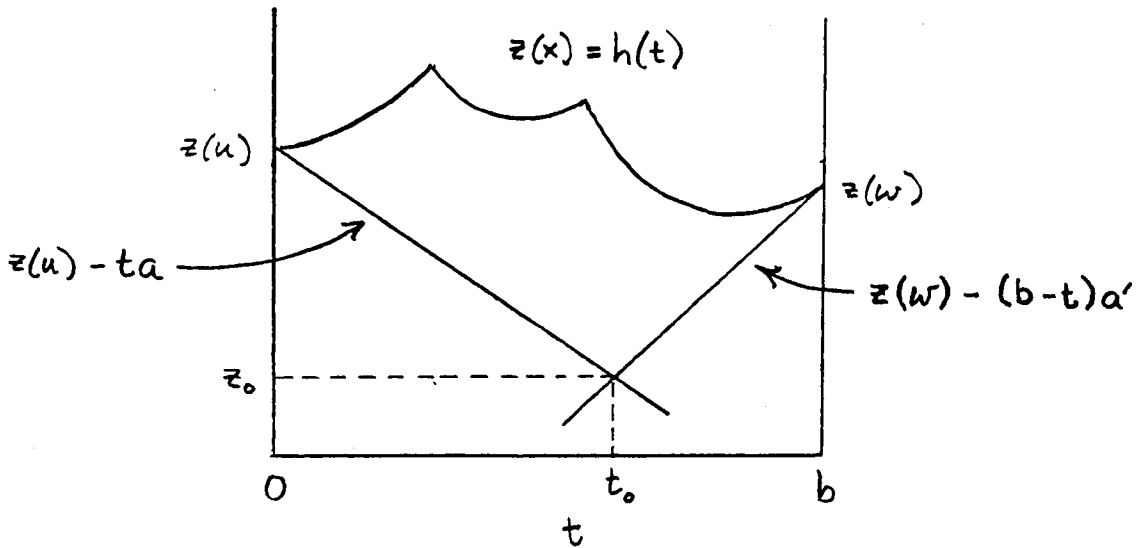


Fig. 2-2. A lower bound for $z(x)$ on an arc.

Since $z_0 = z(n) - t_0 a$, (2.14) immediately follows. If $a < 0$ or $a' < 0$, then obviously (2.15) holds.

□

The solution algorithm with upper bounding is as follows.

Step 0. Set the treelike segment counter k to zero, and set $z_{\min} = \infty$.

Step 1. Choose any unexamined feasible arc $A = A[u, v]$ of G ; if none remain, go to Step 5. Compute $d(u, v_i)$ and $d(w, v_i)$ for $i = 1, \dots, n$, and use the results to compute $s_1, \dots, s_n$.

Step 2. Compute z_0 for A as described in Theorem 2.1. If $z_0 \geq z_{\min}$, go to Step 1.

Step 3. Sort $s_1, \dots, s_n$ in nondecreasing order and determine $s'_0, \dots, s'_{m+1}$ so as to locate the treelike segments on A.

Step 4. For each treelike segment $A[x_1, x_2]$ of A, do the following. Find the solution t^* of the convex line search problem (2.7) on $A[x_1, x_2]$. Set $k = k + 1$ and let x_k^* denote the point on A corresponding to t^* , with $z_k^* = z(d(x_k^*, v_1), \dots, d(x_k^*, v_n))$. Set $z_{\min} = \min \{z_{\min}, z_k^*\}$. When all the segments of A are enumerated, go to Step 1.

Step 5. Any x_k^* such that $z_k^* = \min_j \{z_j^*\}$ solves (2.1).

2.4 Semiseparable Functions

We will show in the next section that if the cost function f is nondecreasing and "semiseparable" as well as convex, then one can eliminate treelike segments in a way that can be more effective than the upper-bounding technique of the previous section. In this section we define semiseparability and establish through a number of lemmas that a large class of cost functions appropriate to network location models are nondecreasing and semiseparable.

We say that a function $f: R^n \rightarrow R$ is nondecreasing if $x'_i \geq x_i$, $i = 1, \dots, n$, implies $f(x'_1, \dots, x'_n) \geq f(x_1, \dots, x_n)$. We define semiseparability as follows. For any permutation π of n elements, let $f_\pi(x_1, \dots, x_n) = f(x_{\pi(1)}, \dots, x_{\pi(n)})$. Let 1_n and 0_n be vectors of n ones and n zeros, respectively. Then define $f: DCR^n \rightarrow R$ to be semiseparable if for all permutations π of n elements,

$$f_\pi(x + \Delta x, y + \alpha 1_t, z) \geq f_\pi(x + \alpha 1_s + \Delta s, y, z) \quad (2.16)$$

implies

$$f_\pi(x, y + \alpha 1_t, z) \geq f_\pi(x + \alpha 1_s, y, z),$$

for all $x, \Delta x \in R^s$, all $y \in R^t$, all $z \in R^u$, and all integers s, t, u with $s, t, u \geq 0$ and $s + t + u = n$, where $\alpha \geq 0, \Delta x \geq 0$, and all the arguments of f_π remain in D .

We first note that although semiseparability appears related to convexity, neither implies the other. Let R^{n+} denote the set of vectors of R^n with all nonnegative components, and consider the nonconvex function $f: R^{2+} \rightarrow R$ given by

$$f(x, y) = \begin{cases} x + y, & \text{if } x + y \leq 1, \\ 1, & \text{otherwise.} \end{cases}$$

Since $f(x, y + \alpha) = f(x + \alpha, y)$ for all $x, y, \alpha \in R^+$, the consequent of (2.16) is true over all of R^{2+} . Thus (2.16) holds, and f is semiseparable even though it is not convex.

Conversely, consider the convex and indeed nondecreasing function $f: R^{2+} \rightarrow R$ given by

$$f(x, y) = \begin{cases} 2x + y, & \text{if } x/3 + y/2 \leq 1, \\ 4(x + y) - 6, & \text{otherwise.} \end{cases}$$

A contour plot of f appears in Fig. 2-3. Note that $f(x + \Delta x, y + \alpha) = f(0 + 1, 0 + 2) = 6 = f(0 + 1 + 2, 0) = f(x + \Delta x + \alpha, y)$, but $f(x, y + \alpha) = f(0, 0 + 2) = 2 < 4 = f(0 + 2, 0) = f(x + \alpha, y)$, which is a violation of (2.16). Thus a convex function need not be semiseparable, even if the function is nondecreasing as well.

It is useful to define a function $f: R^{n+} \rightarrow R$ to be strongly semiseparable if for all permutations π ,

$$f_\pi(x + \beta \Delta x, y + \alpha 1_t, z) = f_\pi(x + \alpha 1_s + \beta \Delta x, y, z) \quad (2.17)$$

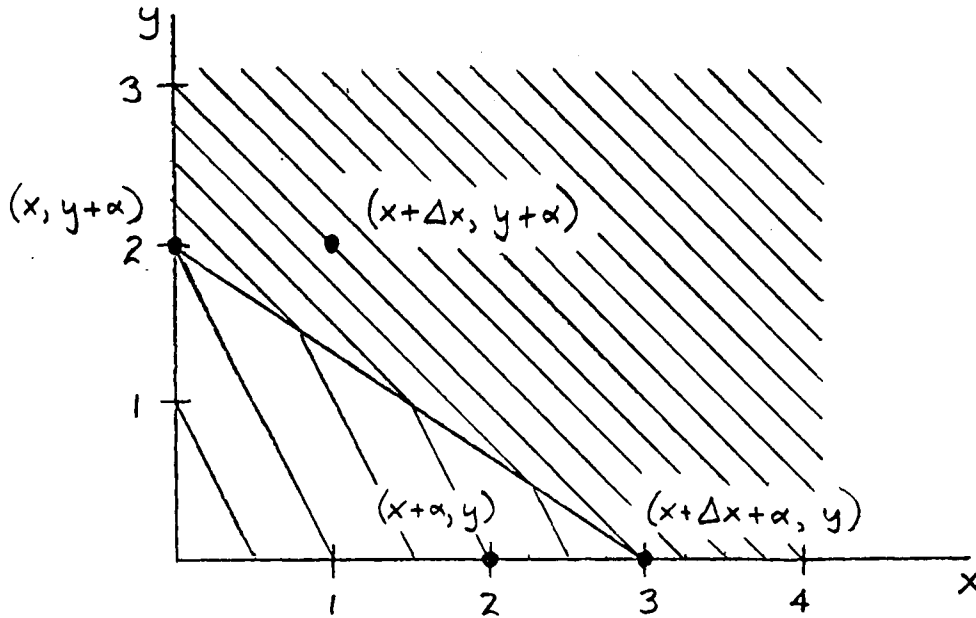


Fig. 2-3. Contour plot of a convex, nondecreasing function that is not semiseparable.

is nonincreasing with respect to $\beta \geq 0$, for all $x, \Delta x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$, and all integers $s, t, u \geq 0$ with $s + t + u = n$, where $\alpha \geq 0$. Clearly, a strongly semiseparable function is semiseparable. Strongly semiseparable functions are useful because:

Lemma 2.4. Any convex combination of strongly semiseparable functions is strongly semiseparable.

Proof. Follows immediately from the definition of strong semiseparability. □

Strongly semiseparable functions can be more easily characterized if we define the generalized directional derivative of $f: D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ as

$$f^0(z; v) = \limsup_{h \rightarrow 0} \sup_{\delta > 0} [f(z + h - \delta v) - f(z + h)] / \delta, \quad (2.18)$$

where $z, v, h \in \mathbb{R}^n$, following Clarke [1975]. It is assumed that f is locally Lipschitz continuous. That is, for every bounded subset $B \subset D$ there is a constant K such that

$$||f(z_1) - f(z_2)|| \leq K ||z_1 - z_2|| ,$$

for all $z_1, z_2 \in B$, where $||\cdot||$ denotes the Euclidean norm. Such a function is almost everywhere differentiable. Given this definition we can state,

Lemma 2.5. If $f: \mathbb{R}^{n+} \rightarrow \mathbb{R}$ is locally Lipschitz continuous such that for any permutation π the generalized directional derivative

$$f_{\pi}^0(x, y, z; l_s, -l_t, 0_u) \quad (2.19)$$

is nondecreasing with respect to each x_j , $j = 1, \dots, s$, for all $x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$, and all integers $s, t, u \geq 0$ with $s + t + u = n$, then f is strongly semiseparable.

Proof. Pick any $s, t, u \geq 0$, $s + t + u = n$, any $\alpha \geq 0$, any $x, \Delta x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$ and any π . We wish to show that (2.17) is nonincreasing with respect to $\beta \geq 0$. This follows if

$$f_{\pi}(x, y + \alpha l_t, z) - f_{\pi}(x + \alpha l_s, y, z) \quad (2.20)$$

is nonincreasing with respect to each x_j , $j = 1, \dots, s$, for all $x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$. Since f is continuous it is integrable on any closed subset of $\mathbb{R}^{n+}$. Then to show (2.20) is nonincreasing it is enough to show that

$$\int_0^\alpha f_\pi^0(x + \alpha 1_s - \gamma 1_s, y + \gamma 1_t, z; 1_s, -1_t, 0_u) d\gamma$$

is nondecreasing with respect to each x_j , $j = 1, \dots, s$. But this follows since (2.19) is nondecreasing with respect to each x_j , by hypothesis. This completes the proof. □

The following corollary provides the motivation for the term "semiseparable." Recall that a matrix (a_{ij}) is diagonally dominant if for each j , $|a_{jj}| \geq \sum_{i \neq j} |a_{ij}|$.

Corollary 2.4. If a twice-differentiable function $f: \mathbb{R}^{n+} \rightarrow \mathbb{R}$ is convex and has a diagonally dominant Hessian at every point in $\mathbb{R}^{n+}$, then f is strongly semiseparable.

Proof. By Lemma 2.5, it is enough to show that (2.19) is nondecreasing with respect to each x_j , $j = 1, \dots, s$, for any $x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$, any integers $s, t, u \geq 0$ with $s + t + u = n$, and any π . But since f is twice differentiable, to say (2.19) is nondecreasing with respect to each x_j is to say

$$\frac{\partial}{\partial x_j} \nabla f_\pi(x, y, z) \cdot (1_s, -1_t, 0_u) \geq 0, \quad j = 1, \dots, s.$$

This follows if

$$\frac{\partial}{\partial x_j} \left[\sum_{i=1}^s \frac{\partial}{\partial x_i} f_\pi(x, y, z) - \sum_{i=1}^t \frac{\partial}{\partial y_i} f_\pi(x, y, z) \right] \geq 0, \quad j = 1, \dots, s.$$

(2.21)

Let $t = (t_1, \dots, t_n) = (x_1, \dots, x_s, y_1, \dots, y_t, z_1, \dots, z_n)$. Then (2.21) follows from

$$\frac{\partial^2}{\partial t_j^2} f(t) - \sum_{\substack{i=1 \\ i \neq j}}^n \left| \frac{\partial^2}{\partial t_i \partial t_j} f(t) \right| \geq 0, \quad j = 1, \dots, s. \quad (2.22)$$

The first term is nonnegative because f is convex. Thus (2.22) holds, since the Hessian of f is diagonally dominant at all points in R^{n+} . This completes the proof.

□

We can now establish some common cost functions to be strongly semiseparable and nondecreasing.

Lemma 2.7. A linear function $f(x) = ax$, with $a, x \in R^{n+}$, is strongly semiseparable and nondecreasing.

Proof. Follows from Corollary 2.4, since the Hessian of f contains all zeros.

□

Lemma 2.8. Any function of the form $f(x) = \sum_{i=1}^n g_i(x_i)$, where g_i is convex, twice differentiable, and nondecreasing, is strongly semiseparable and nondecreasing.

Proof. Clearly the Hessian of f is a diagonal matrix, so that Corollary 2.4 applies.

□

Lemma 2.9. The maximum function $f(t) = \max_i \{t_i\}$ is strongly separable and nondecreasing.

Proof. Obviously f is convex and nondecreasing. By Lemma 2.3 it suffices to prove

$$f^0(x, y, z; 1_s, -1_t, 0_u) \quad (2.23)$$

is nondecreasing with respect to each x_j , $j = 1, \dots, s$, for all $x \in \mathbb{R}^{s+}$, $y \in \mathbb{R}^{t+}$, $z \in \mathbb{R}^{u+}$, and all integers $s, t, u \geq 0$ with $s + t + u = n$. We note that

$$f^0(x, y, z; 1_s, -1_t, 0_u) = \begin{cases} 1 & \text{if } f(x, y, z) = x_i \text{ for some } i, \\ & i = 1, \dots, s; \\ 0 & \text{if } f(x, y, z) > x_j \text{ for } j = 1, \dots, s \\ & \text{and } f(x, y, z) = z_i \text{ for some } i, \\ & i = 1, \dots, u; \\ -1 & \text{otherwise.} \end{cases} \quad (2.24)$$

For, by (2.18) we have

$$\begin{aligned} f^0(x, y, z; 1_s, -1_t, 0_u) = \\ \lim_{h, h', h'' \rightarrow 0; \delta \downarrow 0} \sup \left(\max_{i, j, k} \{x_i + h_i + \delta, y_j + h'_j - \delta, z_k + h''_k\} \right. \\ \left. - \max_{i, j, k} \{x_i + h_i, y_j + h'_j, z_k + h''_k\} \right) / \delta \end{aligned} \quad (2.25)$$

where $h = (h_1, \dots, h_s)$, $h' = (h'_1, \dots, h'_t)$, $h'' = (h''_1, \dots, h''_u)$. If $f(x, y, z) = x_q$ for some q , then for sufficiently small h, h', h'' (2.25) implies

$$f^0(x, y, z; l_s, -l_t, 0_u) = \lim_{h_q \rightarrow 0; \delta \downarrow 0} \sup [(x_q + h_q + \delta) - (x_q + h_q)] / \delta = 1.$$

If $f(x, y, z) > x_i$ for $i = 1, \dots, s$ and $f(x, y, z) = z_q$ for some q , (2.25) implies

$$f^0(x, y, z; l_s, -l_t, 0_u) = \lim_{h_q \rightarrow 0; \delta \downarrow 0} \sup [(z_q + h_q'') - (z_q + h_q'')] / \delta = 0.$$

Finally, if $f(x, y, z) > x_i$ for $i = 1, \dots, s$ and $f(x, y, z) > z_k$ for $k = 1, \dots, u$, then $f(x, y, z) = y_q$ for some q , and

$$f^0(x, y, z; l_s, -l_t, 0_u) = \lim_{h_q \rightarrow 0; \delta \downarrow 0} \sup [(y_q + h_q' - \delta) - (y_q + h_q')] / \delta = -1.$$

This establishes (2.24).

Now to show (2.23) is nondecreasing with respect to any given x_q , we consider three cases.

$$\text{Case I.} \quad \max_{\substack{i=1, \dots, n \\ i \neq q}} \{t_i\} = \max_{\substack{i=1, \dots, s \\ i \neq q}} \{x_i\},$$

where $(t_1, \dots, t_n) = (x_1, \dots, x_s, y_1, \dots, y_t, z_1, \dots, z_u)$. In this case (2.24) implies $f^0(x + \Delta x_q e^q, y, z; l_s, -l_t, 0_u) = 1$ for all $\Delta x_q \geq 0$, where e^q is the q th unit vector. Thus (2.23) is constant and therefore nondecreasing with respect to x_q .

$$\text{Case II.} \quad \max_{\substack{i=1, \dots, n \\ i \neq q}} \{t_i\} > \max_{\substack{i=1, \dots, s \\ i \neq q}} \{x_i\},$$

and

$$\max_{\substack{i=1,\dots,n \\ i \neq q}} \{t_i\} = \max_{k=1,\dots,u} \{z_k\} .$$

By (2.24), $f^0(x + \Delta x_q e^q, y, z; 1_s, -1_t, 0_u)$ is equal to 0 when

$$x_q + \Delta x_q < \max_{k=1,\dots,u} \{z_k\}$$

and equal to 1 otherwise. Thus (2.3) is nondecreasing with respect to x_q .

$$\text{Case III. } \max_{\substack{i=1,\dots,n \\ i \neq q}} \{t_i\} > \max_{\substack{i=1,\dots,s \\ i \neq q}} \{y_i\} ,$$

and

$$\max_{\substack{i=1,\dots,n \\ i \neq q}} \{t_i\} = \max_{j=1,\dots,t} \{y_j\} .$$

By (2.24), $f^0(x + \Delta x_q e^q, y, z; 1_s, -1_t, 0_u)$ is equal to -1 when

$$x_q + \Delta x_q < \max_{j=1,\dots,t} \{y_j\}$$

and equal to 1 otherwise. Again (2.3) is nondecreasing with respect to x_q .

□

The weighted maximum function requires special treatment.

Lemma 2.10. The weighted maximum function $f(t) = \max_i \{w_i t_i\}$, $w_i \geq 0$ for $i = 1, \dots, n$, is convex, semiseparable and nondecreasing.

Proof. Clearly f is convex and nondecreasing. To show f is semi-separable, we pick and $x, \Delta x \in \mathbb{R}^{s+}, y \in \mathbb{R}^{t+}, z \in \mathbb{R}^{u+}$, $s, t, u \geq 0$ with $s + t + u = n$, and assume

$$f(x + \Delta x, y + \alpha 1_t, z) \geq f(x + \alpha 1_s + \Delta x, y, z), \text{ or } y_1 \geq y_2. \quad (2.26)$$

We wish to show,

$$f(x, y + \alpha 1_t, z) \geq f(x + \alpha 1_s, y, z), \text{ or } y_3 \geq y_4. \quad (2.27)$$

Note that if $\alpha = 0$, (2.27) holds trivially. Therefore we assume $\alpha > 0$. This and (2.26) imply $f_1 = f(0_s, y + \alpha 1_t, z) \geq f(0_s, y, z) = f_2$, as well as

$$f(x + \alpha 1_s + \Delta x, 0_t, 0_u) \leq f(0_s, y + \alpha 1_t, z). \quad (2.28)$$

The following cases are exhaustive but not mutually exclusive.

$$\text{Case I. } f(0_s, y + \alpha 1_t, z) = f(0_s, y + \alpha 1_t, 0_u).$$

From (2.28) we have $f(x + \alpha 1_s, 0_t, 0_u) \leq f(0_s, y + \alpha 1_t, 0_u)$, and (2.27) follows.

$$\text{Case II. } f(0_s, y + \alpha 1_t, z) = f(0_s, 0_t, z).$$

From (2.28) we have $f(x + \alpha 1_s, 0_t, 0_u) \leq f(0_s, 0_t, z)$, and (2.27) follows.

□

Unfortunately the weighted maximum function is not strongly semi-separable. Suppose, for instance, that $s = 2$, $t = 1$, $x = (2, 1)$, $y = 0$, $\alpha = 1$, and let the weights be $(w_1, w_2, w_3) = (1, 2, 1)$. Then (2.17) is

$$\max \{2, 2 \cdot 1, 0\} - \max \{3, 2 \cdot 2, 0\} = -2$$

when $\beta \Delta x = (0, 0)$, and (2.10) is

$$\max \{3, 2 \cdot 1, 0\} - \max \{4, 2 \cdot 2, 0\} = -1$$

when $\beta \Delta x = (1, 0)$, so that (2.17) fails to be nonincreasing with respect to x_1 . This is not a counterexample of the semiseparability of the weighted maximum function, however, since $\max \{2, 2 \cdot 1, 0\} < \max \{3, 2 \cdot 2, 0\}$.

Since the weighted maximum function is not strongly semiseparable, we cannot conclude on the basis of Lemma 2.4 that the cent-dian function,

$$f(x) = (1 - \gamma) \sum_i w_i x_i + \gamma \max_i \{u_i x_i\},$$

is semiseparable. Indeed it is not. We present a counterexample suggested by Fig. 2-4, since it will be useful later. Let $\alpha = 1$ in (2.16), and let $(x, y + \alpha 1) = (d(y_1, v_1), d(y_1, v_2), d(y_1, v_3), d(y_1, v_4)) = (0, 2, 1, 1)$, where $s = 3$, $t = 1$. Let the parenthesized numbers in Fig. 2-4 be (a_j, w_j) for each v_j . Then if $\Delta x = (0, 1, 0)$ and $\gamma = \frac{1}{2}$, we have $f(x + \Delta x, y + \alpha 1) = f[(0, 2, 1) + (0, 1, 0), 0 + 1] = 5 \geq 5 = f[(0, 2, 1) + (1, 1, 1) + (0, 1, 0), 0] = f(x + \alpha 1 + \Delta x, y)$, so that the antecedent of (2.16) holds. But $f(x, y + \alpha 1) = f[(0, 2, 1), 0 + 1] = 4 < 4\frac{1}{2} = f[(0, 2, 1) + (1, 1, 1), 0] = f(x + \alpha 1, y)$, so that the consequent of (2.16) and therefore (2.16) are false.

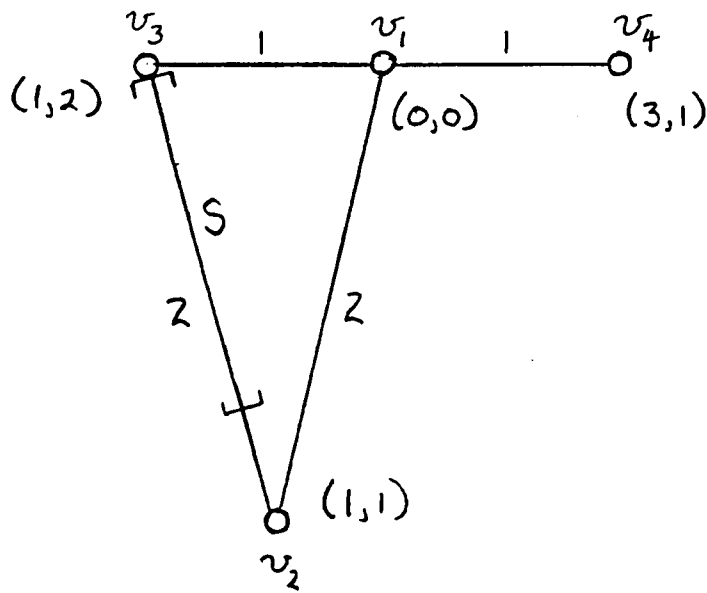


Fig. 2-4. Counterexample of the semiseparability of the weighted cent-dian function.

2.5 Solution of the Convex, Nondecreasing, Semiseparable Problem

We now describe a substantial improvement of the solution technique of Section 2.2 that applies whenever the cost function f is nondecreasing and semiseparable as well as convex. Like the upper-bounding technique of Section 2.3, it eliminates a large number of treelike segments without having to solve convex line search problems on them. Though restricted to nondecreasing and semiseparable cost functions, this technique is yet of considerable practical value, since a large class of useful cost functions are of this sort (Section 2.4 and Chapter 5).

The elimination of segments described in this section can be combined with the elimination of arcs by the upper-bounding algorithm of Section 2.3. It is in fact quite fruitful to do so, since we will see

in Section 2.8 that the two methods work well in tandem. The algorithm presented at the end of this section makes provision for both elimination techniques.

First, some definitions are required. If S is a treelike segment on network G , let $T(S)$ be any shortest path tree rooted at S . That is, $T(S)$ is a spanning tree of G such that for any $x \in T(S)$, the path in $T(S)$ from x to any $v_i \in V$ is a shortest path in G from x to v_i . Given $T = T(S)$, let d^T be the distance function on $T(S)$. We say that a location solves (2.1) on tree $T(S)$ if it solves the problem that results from substitution $T(S)$ for G and d^T for d in (2.1).

Our strategy is to solve a convex location problem on the tree $T(S)$ rather than on just the segment S , since this may allow us to eliminate some treelike segments. The key to the elimination lies in the following theorem.

Theorem 2.2. Let z be defined as in (2.1), where f is convex, nondecreasing and semiseparable. Let S be a treelike segment of network G , and let x solve (2.1) on $T(S)$. Then for any point y lying on the path of $T(S)$ from x to and including S , $z(x) \leq z(y)$.

Proof. It suffices to show that z is nondecreasing on every treelike segment on the path of $T(S)$ from x to and including S , as one moves away from x . Let $A[y_1, y_2]$ be any interval contained in any such segment, where y_1 is closer to S in $T(S)$ and y_2 closer to x . It suffices to show that $z(y_1) \geq z(y_2)$. (Fig. 2-5.)

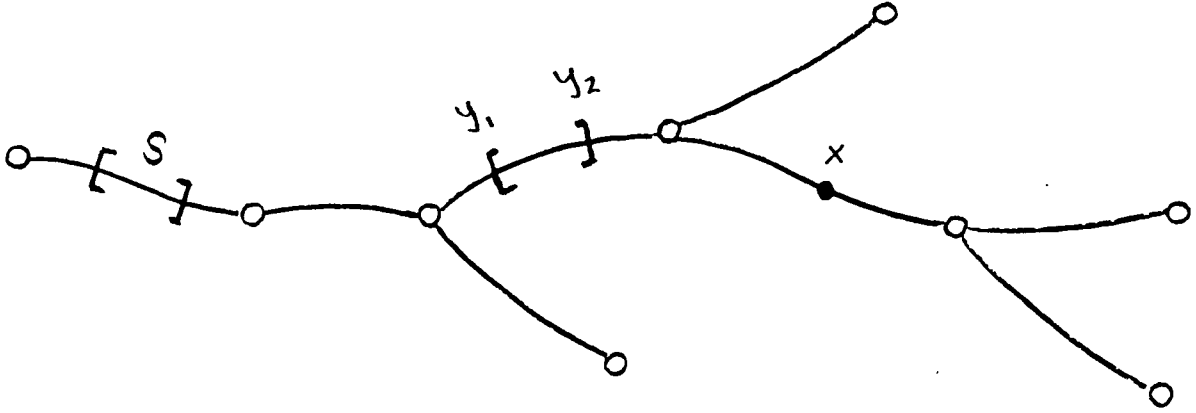


Fig. 2-5. Illustration of Theorem 2.2.

We first define,

$$z^T(y_2) = f(d^T(y_2, v_1), \dots, d^T(y_2, v_n)) = f(d_1^T, \dots, d_n^T), \quad (2.29)$$

setting $d_i^T = d^T(y_2, v_i)$ for brevity. Cut $T(S)$ at y_2 and let $v_1, \dots, v_k$ be the nodes in the part of $T(S)$ containing S (excluding y_2 if y_2 is a node). Then we have,

$$z^T(y_1) = f(d_1^T - \Delta, \dots, d_k^T - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta), \quad (2.30)$$

where Δ is the length of $A[y_1, y_2]$. Note that $d^T(y_1, v_i) = d(y_1, v_i)$ and $d^T(y_2, v_i) = d(y_2, v_i)$ for $i = k + 1, \dots, n$. Thus we can write,

$$z(y_2) = f(d(y_2, v_1), \dots, d(y_2, v_n)) = f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T), \quad (2.31)$$

where $d_i = d(y_2, v_i)$.

Now, since f is nondecreasing, z is convex on $T(S)$. Thus since x minimizes z^T on $T(S)$, we have

$$z^T(y_1) \geq z^T(y_2) . \quad (2.32)$$

This, (2.29) and (2.30) imply,

$$\begin{aligned} f(d_1^T - \Delta, \dots, d_k^T - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \\ \geq f(d_1^T, \dots, d_n^T) . \end{aligned} \quad (2.33)$$

But since $d_i \leq d_i^T$ for all i , and f is semiseparable, (2.33) implies

$$\begin{aligned} f(d_1 - \Delta, \dots, d_k - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \\ \geq f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T) . \end{aligned} \quad (2.34)$$

Since y_1 and y_2 lie in a single treelike segment, it is clear that for $i = 1, \dots, k$, $d(y_1, v_i) = d(y_2, v_i) \pm \Delta$, so that $d(y_1, v_i) = d_i - \Delta_i$, where $\Delta_i \leq \Delta$, and therefore

$$z(y_1) = f(d_1 - \Delta_1, \dots, d_k - \Delta_k, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) . \quad (2.35)$$

But since f is nondecreasing, we have from (2.34) that

$$\begin{aligned} f(d_1 - \Delta_1, \dots, d_k - \Delta_k, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \\ \geq f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T) . \end{aligned}$$

This, (2.31) and (2.35) imply that $z(y_1) \geq z(y_2)$, as desired.

□

We showed in the previous section that the weighted cent-dian function, though convex and nondecreasing, is not semiseparable. We therefore have no guarantee the above theorem holds for it, and we show now that it does not. Consider the network of Fig. 2-3 (p. 32), and define $x, y, \Delta x$ and α as at the end of the previous section. Let S be the treelike segment indicated in the figure, and let z and z^T be the cent-dian objective functions on the graph and on $T(S)$, as in the above proof. Note that $z^T(v_4) = f(x + \alpha 1 + \Delta x, y) = 5$ maximizes z^T on $T(S)$, so that we can say $x = v_4$. But $z(v_1) = f(x, y + \alpha 1) = 4 < 4\frac{1}{2} = f(x + \alpha 1, y) = z(v_4)$, so that Theorem 2.2 fails to hold.

The utility of Theorem 2.2 is that when one locates the solution x of the location problem on $T(S)$, he can immediately eliminate all tree-like segments that lie wholly on the path in $T(S)$ from x to and including S , for no point in any of these segments can be superior to x . Furthermore, finding x is a relatively easy problem because, as remarked in Section 2.1, the single-facility problem is easily solved on a tree when the cost function is convex and nondecreasing, as it is here. Thus one can solve (2.1) by picking uneliminated treelike segments S and solving the single-facility problem on $T(S)$; if the solution x is in S , it is recorded as the k th candidate x_k^* for a globally optimal location on G , and S is eliminated; if x is elsewhere, S and all the segments between S and x are eliminated. The process is continued until all segments are eliminated. Any location x_k^* such that $z(x_k^*) = \min_j \{z(x_j^*)\}$ is a global optimum.

The algorithm below incorporates all three methods hitherto described (Sections 2.2, 2.3, 2.5), using each where applicable. For convenience we adopt the following terminology. Technique 1 is the upper bounding technique of Section 2.3. It can be used whenever the cost function f is convex, provided that either f is differentiable, f is the weighted maximum function, or an analogue of Theorem 2.1 has been developed for a nondifferentiable f other than the weighted maximum function. Technique 2 is the elimination technique described in this section. It can be used whenever f is convex, nondecreasing and semiseparable. The following algorithm solves (2.1) for any convex f , but it is more efficient when Technique 1 and/or Technique 2 is used.

As in Sections 2.2 and 2.3, the algorithm also permits one to exclude the facility from certain parts of the network. One can exclude the facility from any arc by considering that arc "infeasible" and passing over it in Step 1 of the algorithm. Also one can exclude the facility from an open interval within an arc by inserting a node at either end of the interval and regarding the new arc between the nodes as infeasible.

Step 0. Set the relative minimum counter k to zero, and let $z_{\min} = \infty$.

Step 1. Choose any uneliminated feasible arc $A_j = A[u, w]$ of G ; if none remain, go to Step 11. Compute $d_i = d(u, v_i)$ and $d'_i = d(w, v_i)$ for $i = 1, \dots, n$, and record the location in G of the corresponding shortest-path trees, which we may call $T(u)$ and $T(w)$, respectively. If a list L_j of the treelike segments of A_j has already been made, go to Step 4.

Otherwise compute $s_i = \frac{1}{2}[d'_i - d_i + b(u,w)]$ for $i = 1, \dots, n$. If Technique 1 is not used, go to Step 3.

Step 2. Compute z_0 for A_j as described in Theorem 2.1. If $z_0 \geq z_{\min}$, eliminate A_j and go to Step 1.

Step 3. Sort $s_1, \dots, s_n$ in nondecreasing order so as to determine the boundaries $s'_0, \dots, s'_{m+1}$ of A_j 's treelike segments (Corollary 2.2). Let L_j be a list of the segments.

Step 4. Pick a treelike segment S from L_j ; if none remain, eliminate A_j and go to Step 1. Use $T(u)$ and $T(w)$ and d_i, d'_i for $i = 1, \dots, n$ to locate a shortest-path tree $T(S)$ in G rooted at S .

Step 5. If not using Technique 2, determine whether the point x that solves (2.1) lies on A_j ; if so, let $\ell = j$ and go to Step 8; if not, remove S from L_j and go to Step 4. If using Technique 2, find the arc $A_\ell = A[\hat{u}, \hat{w}]$ on which x lies. Do this by following a path along which $z^T(y)$ decreases, starting with A_j , until x is bracketed on an arc A_ℓ . Eliminate all arcs (if any) in $T(S)$ between A_j and A_ℓ , excluding A_j and A_ℓ . If $j = \ell$ go to Step 8.

Step 6. Remove from L_j the segments of A_j that lie between S and A_ℓ on $T(S)$, including S .

Step 7. Compute $\hat{d}_i = d(\hat{u}, v_i)$ and $\hat{d}'_i = d(\hat{w}, v_i)$ for $i = 1, \dots, n$, and compute $\hat{s}_i = \frac{1}{2}[\hat{d}'_i - \hat{d}_i + b(\hat{u}, \hat{w})]$. Perform Step 2 with A_ℓ replacing A_j , and branch to Step 1 if $z_0 \geq z_{\min}$. If $z_0 < z_{\min}$, perform Step 3 with A_ℓ and $\hat{s}_i$ replacing A_j and s_i , resulting in List L_ℓ .

Step 8. Find a point x on A_ℓ that solves (2.1) on $T(S)$. The method of doing this depends on the cost function f . If $\ell \neq j$, go to Step 10.

Step 9. If $x \in S$, set $k = k+1$, $x_k^* = x$, $z_k^* = z(x_k^*)$, and $z_{\min} = \min\{z_{\min}, z_k^*\}$, and remove S from L_j and go to Step 4. Otherwise do the following. If not using Technique 2, remove S from L_j and go to Step 4. If using Technique 2, remove from L_j the segments of A_j that lie between x and S , including S but excluding the segment containing x . Then go to Step 4.

Step 10. Remove from L_ℓ the segments of A_ℓ (if any) that lie between x and S in $T(S)$, excluding the segment containing x . Go to Step 4.

Step 11. Any x_k^* for which $z_k^* = \min_j \{z_j^*\}$ solves (2.1).

2.6 Worst-Case Performance

In this section we determine the worst-case complexity of the algorithm of Section 2.5. We first establish an upper bound on the order of its complexity, and then we show that the upper bound can be achieved in at least one instance.

Suppose we are to solve (2.1) on a network of n nodes and m arcs. In the worst case we try to eliminate segments using Techniques 1 and 2 (as defined in Section 2.5) and succeed in eliminating none. Thus we solve a convex location problem on a tree $T(S)$ rooted at every treelike segment S of the network. Since by Corollary 2.3 an arc contains at most $n - 1$ segments, there are at most $\mathcal{O}(mn)$ segments in the network.

The complexity of finding an optimal location x on $T(S)$ can be analyzed as follows. Let d^T be the distance measure and z^T the median function on $T(S)$. We first observe that since no segments are

eliminated, x must lie on either the arc A containing x or one of the adjacent arcs. (The latter is possible if the path from x to S does not contain an entire segment of A or the arc on which x lies.) Since z^T is convex on $T(S)$, to determine on which arc x lies requires at most that z^T be evaluated twice (to determine whether z^T is increasing or decreasing) at either end of the $n - 1$ or fewer arcs incident to one of the nodes of A . This is a total of $\mathcal{O}(n)$ function evaluations. Each evaluation of $z^T(y)$ requires that TSHIFT (see Appendix) compute distances $d_1^T, \dots, d_n^T$ on $T(S)$ from y to nodes $v_1, \dots, v_n$, a task of complexity $\mathcal{O}(n)$, and that $f(d_1^T, \dots, d_n^T)$ be evaluated. Thus if the evaluation of $f(d_1^T, \dots, d_n^T)$ is of complexity $\mathcal{O}(g(n))$, the evaluation of z^T at one point is $\mathcal{O}(n + g(n))$. Pinpointing x on the arc containing it can be done at worst by bisection search. This requires that z^T be evaluated a number of times that is a logarithmic function of the ratio of the arc length to the desired precision and is therefore independent of n and m . Thus the complexity of finding x on $T(S)$ is $\mathcal{O}(n(n + g(n)))$.

If Technique 2 is not used, one solves a convex problem on S rather than on $T(S)$. Since no arcs other than A are involved, the complexity is $\mathcal{O}(n + g(n))$.

Finally, shortest-path trees must be computed at either end of each of the m arcs, and each tree computation has complexity of $\mathcal{O}(n^2)$ using, say, the Dijkstra algorithm [1959]. This means that the total complexity of computing shortest-path trees is $\mathcal{O}(mn^2)$. (If all shortest paths are computed in advance, the complexity is of lower order, but more storage is required for the $n \times n$ array of distances.) The complexity

of performing the upper bound test for Technique 1 is dominated by that of computing shortest-path trees.

Since there are at most $\mathcal{O}(mn)$ segments on which to solve a convex problem of complexity of at most $\mathcal{O}(n(n + g(n)))$ or $\mathcal{O}(n + g(n))$, we have proved,

Lemma 2.11. If the complexity of evaluating f at one point is $g(n)$, the complexity of the algorithm of Section 2.6 is no more than $\mathcal{O}(mn^2 + mn^2(n + g(n))) = \mathcal{O}(mn^2(n + g(n)))$ if Technique 2 is used and no more than $\mathcal{O}(mn^2 + mn(n + g(n))) = \mathcal{O}(mn(n + g(n)))$ otherwise.

In the common case of $g(n) = \mathcal{O}(n)$, the complexity bounds are simply $\mathcal{O}(mn^3)$ with Technique 2 and $\mathcal{O}(mn^2)$ without.

We now exhibit a problem, which we may call Problem W, in which the bounds of Lemma 2.11 are achieved. Let Problem W be set on a complete network G_w of n nodes and $m = \frac{1}{2}n(n - 1)$ arcs. List the arcs $e_1, \dots, e_m$ in any order and let the length b_t of arc e_t be $2^M + 2^t$, where M is large. Let the objective function z be the median function, with each node assigned a weight of one.

Lemma 2.12. If $M \geq m - 1$, network G_w has $m(n - 1)$ treelike segments.

Proof. Let $b_{ij} = b(v_i, v_j)$ be the length of arc (v_i, v_j) , and let $d_{ij} = d(v_i, v_j)$. Recall from Corollary 2.3 that node v_k generates a segment boundary on the interior of (v_i, v_j) if $0 < s_k < b_{ij}$, where $s_k = \frac{1}{2}[d_{jk} - d_{ik} + b_{ij}]$, or equivalently,

$$-b_{ij} < d_{jk} - d_{ik} < b_{ij} . \quad (2.36)$$

Note that $b_{ij} \leq b_{ik} + b_{kj}$ for all i, j, k , so that $d_{ij} = b_{ij}$. Thus since $-b_{ij} < b_{jk} - b_{ik} < b_{ij}$ for all distinct i, j, k , (2.36) holds for all distinct i, j, k . Thus the interior of every arc (v_i, v_j) contains a segment boundary for each $v_k \neq v_i, v_j$. Since $b_{jk} - b_{ik}$ is different for every $k \neq i, j$, it follows that $d_{jk} - d_{ik}$ is different for every k and that all $n - 2$ segment boundaries on (v_i, v_j) are distinct. Thus G_w has $m(n - 1)$ treelike segments.

□

Lemma 2.13. For sufficiently large M , Technique 1 eliminates no arcs in problem W .

Proof. Note first that the shortest and longest arcs of G_w have lengths $b = 2^M + 1$ and $2^M + 2^{m-1} = (1 + 2^{m-1-M})2^M < \alpha b$, respectively, where $\alpha = 1 + 2^{m-1-M}$. Thus $b \leq b_{ij} \leq \alpha b$ for all i, j . Since $b_{ij} = d_{ij}$ for all i, j , we have from (2.10) that $(n - 2)b \leq a \leq \alpha(n - 2)b$, and similarly for a' as defined in Theorem 2.2. For any arc (u, w) we note that $(n - 1)b \leq z(u) \leq \alpha(n - 1)b$ and similarly for $z(w)$. Hence (2.14) yields that $z_0 \leq \alpha(n - 1)b - [(n - 1)b - \alpha(n - 1)b + (n - 2)b] (n - 2)b / 2\alpha(n - 2)b = (\alpha - 1/2\alpha - 1/2)(n - 1)b - (1/2\alpha)(n - 2)b$. Note that for sufficiently large M , $z_0 < (n - 1)b$. But no relative minimum so far identified can be less than $(n - 1)b$, so that necessarily $z_0 < z_{\min}$, and no arc is eliminated.

□

Lemma 2.14. For sufficiently large M , the segments of G_w can be enumerated in such a way that Technique 2 eliminates none of them in problem W .

Proof. Pick any arc (u,w) of G_w and suppose the nodes of G_w are numbered so that $s_1, \dots, s_n$ (as defined in Section 2.1) is a nondecreasing sequence. Consider a segment $S_k = [s_k, s_{k+1}]$. We show that the optimal point x on $T(S)$ —i.e., the median of $T(S)$ —lies at u or w . First note that since $b_{ij} = d_{ij}$ for all i,j , $T(S)$ has the form depicted in Fig. 2-6.

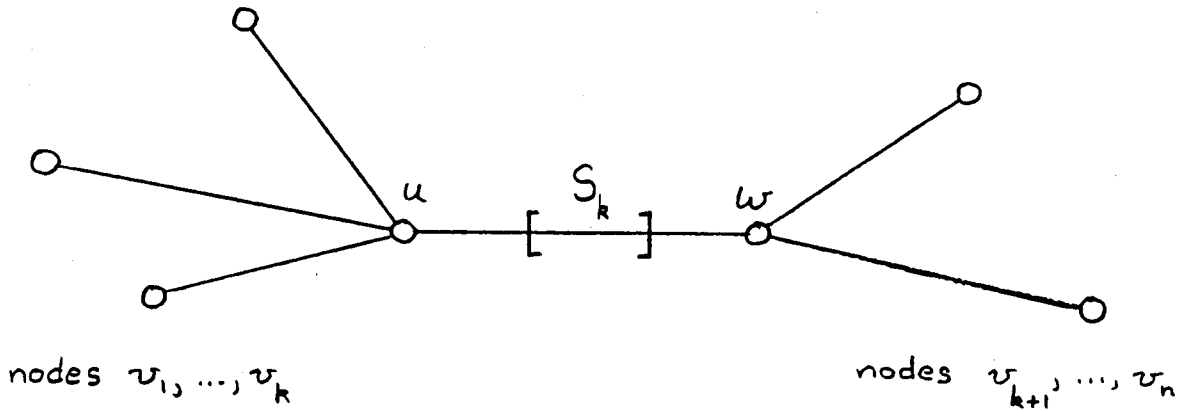


Fig. 2-6. Form of a shortest-path tree $T(S)$ on G_w .

If b and α are as defined for Lemma 2.13, we have

$$\begin{aligned} z^T(u) &= \sum_{1 \leq i \leq k} d^T(u, v_i) + \sum_{k+1 \leq i \leq n} [b(u, w) + d^T(w, v_i)] \\ &\leq [\alpha(k-1)b] + [\alpha b + 2\alpha(n-k-1)b] = z_1^T. \end{aligned}$$

For any node $v_j \neq u$, $j = 1, \dots, k$, we have

$$\begin{aligned} z^T(v_j) &= \sum_{1 \leq i \leq k} [d(v_j, u) + d(u, v_i)] + \sum_{k \leq i \leq n} [d(v_j, u) + b(u, w) + d(w, v_i)] \\ &\geq [b + 2(k-1)b] + [2b + 3(n-k-1)b] = z_2^T. \end{aligned}$$

It is clear that for sufficiently large M , $z_1^T \leq z_2^T$, so that $z^T(u) \leq z^T(v_i)$ for $i = 1, \dots, k$. By similar argument $z^T(w) \leq z^T(v_i)$ for $i = k+1, \dots, n$. It follows that u or w is a median on $T(S)$.

It is easy to see that there is a k , $1 \leq k \leq n$, such that u is a median on $T(S_j)$ for $j = 1, \dots, k$ and w is a median on $T(S_j)$ for $j = k+1, \dots, n-1$. Thus if the order of enumeration is segments $S_1, \dots, S_k$, followed by $S_{n-1}, S_{n-2}, \dots, S_{k+1}$, then no segments are eliminated.

□

From Lemma 2.12 we know that for sufficiently large M , G_w has $m(n-1)$ treelike segments, and from Lemmas 2.13 and 2.14 we know that all the segments S of G_w must be enumerated. Since the median on each $T(S)$ lies at one of the nodes incident to the arc A containing S , it is necessary to evaluate z^T on each of the $n-1$ arcs incident to this node, so as to establish that $z^T(y)$ is in fact nondecreasing as y moves away from the node in every direction. Again, finding a median on any $T(S)$ has complexity $\mathcal{O}(n(n+g(n)))$. If Technique 2 is not used, solving the convex problem on S again has complexity $\mathcal{O}(n+g(n))$. We have proved,

Theorem 2.3. If the complexity of evaluating f at one point is $g(n)$, the complexity of the algorithm of Section 2.6 is, in the worst

case, $\theta(mn^2(n + g(n)))$ if Technique 2 is used and $\theta(mn(n + g(n)))$ otherwise.

Although the use of Technique 2 increases worst-case complexity, we will see in the next section that its use in fact reduces computation time on the several random and real-world networks on which it has been tested.

2.7 Computational Experience

The computer implementation described in the Appendix was tested for efficiency both on random networks and on actual highway, rail and waterway networks. This section reports the results.

All testing was carried out on a DECsystem-10 computer running under the TOPS-10 monitor. The Fortran code was compiled by Version 6 of the Fortran-10 compiler. It should be borne in mind that the code is a research code written for flexibility rather than efficiency. A thoroughgoing streamlining of the code should improve CPU times substantially. Also the DEC-10 computer is slower than many "mainframe" computers now in use.

The computer program was first tested on random networks, generated as follows. The size of the network was set at n nodes and m arcs, and then m numbers uniformly distributed between zero and 10 were randomly assigned positions in the upper triangle of an $n \times n$ distance matrix (diagonal excluded). These represented the arcs and their lengths. If the arcs covered fewer than n nodes, or if the network was disconnected, it was discarded and another generated. Finally, weights uniformly distributed between zero and one were assigned to the nodes.

Table 2-1 displays the performance of the computer code on five random networks, each having twice as many arcs as nodes. Recall that Elimination Technique 1 is the upper bounding technique (Sect. 2.3), and Technique 2 involves solving convex problems on trees rooted at treelike segments (Sect. 2.5). Note in Table 2-1 that the location problem is solved using both the center of gravity and the cent-dian functions.

The cent-dian function requires special explanation. It can be written $z(x) = (1 - \alpha)z_m(x) + \alpha z_c(x)$, where z_m and z_c are the median and center functions, respectively. Since $z_m(x)$ tends to be much larger than $z_c(x)$ and to show greater absolute variations as x moves over the network, the weight α does not accurately reflect the relative importance of z_m and z_c . In an attempt to remedy this we will use an adjusted weight β defined so that when $\beta = 0.5$, z_m and z_c make equal contributions at optimality. That is, $\beta = 0.5$ should correspond to a value of α such that $(1 - \alpha)z_m^* = \alpha z_c^*$, where z_m^* and z_c^* are the optimal values of z_m and z_c on the network in question. To calculate α we set $\beta(1 - \alpha)z_m^* = (1 - \beta)\alpha z_c^*$, which implies

$$\alpha = \frac{\beta z_m^*}{\beta z_m^* + (1 - \beta)z_c^*}, \quad \text{or} \quad \beta = \frac{\alpha z_c^*}{\alpha z_c^* + (1 - \alpha)z_m^*}. \quad (2.38)$$

Thus β should be a fair indication of the relative importance of z_m and z_c . Unless otherwise indicated, $\beta = 0.5$ in the computations described in this section.

It is evident from Table 2-1 that Technique 2 is generally more effective at eliminating treelike segments in random networks than Technique 1, but that either technique is of great advantage both by

Table 2-1. Computational results for five random networks,* showing percent of tree-like segments examined and CPU time** in seconds for the center of gravity and cent-dian objective functions.

Number of Nodes Arcs		z_m^*	z_c^*	Objective Function	No Elimination:		Percent of Segments Examined, CPU Time Using Elimination Technique					
					Segments	CPU	No. 1		No. 2		Nos. 1, 2	
							%	CPU	%	CPU	%	CPU
1.	20	40	39.3	10.86	ctr. of grav. cent-dian	270 270	80 76	3.1 3.0	62 69	3.0 3.3	55 51	2.7 2.8
2.	40	80	101.0	15.20	ctr. of grav. cent-dian	846 846	77 59	16.9 13.2	59 60	16.1 15.6	44 34	12.1 10.0
3.	60	120	267.2	18.26	ctr. of grav. cent-dian	1845 1845	71 32	53.8 27.7	46 45	43.1 43.1	30 14	28.6 16.6
4.	80	160	320.3	20.46	ctr. of grav. cent-dian	3153 3153	70 51	127.7 118.1	46 42	99.6 98.1	31 17	67.6 47.9
5.	100	200	403.7	21.07	ctr. of grav. cent-dian	3972 3972	48 54	139.9 157.7	45 45	166.7 164.1	25 21	104.9 93.3

*Arc lengths are uniformly distributed between 0 and 10; node weights are uniformly distributed between 0 and 1. Arcs are generated as described in the text.

**On a DEC-10 computer.

itself and in combination with the other. Both techniques are more effective in larger networks. Table 2-2 suggests, however, that upper bounding is less effective in denser networks. Technique 2 is slightly more effective in denser networks, so that when both techniques are used these trends tend to cancel.

In both tables Techniques 1 and 2 bring smaller reductions in CPU time than in the number of segments examined. This is due to three factors. One is that the examination of a segment is a more complex affair when Technique 2 is used than otherwise, since Technique 2 solves for each examined segment a convex problem on an entire tree rather than just on the segment. A second factor is that Technique 1 cannot eliminate an arc without first constructing shortest-path trees at either end of it, and this is already a sizeable portion of the work that would be required to process the segments on the arc. Finally, the Fortran code implementing Techniques 1 and 2 is not efficient enough to take full advantage of the reduction in segments examined.

Although the elimination techniques work reasonably well on random networks, they do not effect the order-of-magnitude improvement one would like to see. Yet there is reason to believe that upper bounding, at least, would be more effective on actual networks. An arc near the periphery of a network is generally more likely to be eliminated by an upper bound than one nearer the middle of the network. But random networks, unlike most actual networks, have no clear periphery.

The expectation that upper bounding can excel on actual networks is confirmed by Tables 2-3 and 2-4.

Table 2-2. Computational results for five random networks* on 40 nodes, showing percent of tree-like segments examined and CPU time** in seconds for the center of gravity and cent-dian objective functions.

Number of Nodes		z* m	z* c	Objective Function	No Elimination:		Percent of Segments Examined, CPU Time Using Elimination Technique					
							No. 1		No. 2		Nos. 1, 2	
					Segments	CPU	%	CPU	%	CPU	%	CPU
1.	40	60	204.2	24.67	ctr. of grav. cent-dian	384 384	10.5 11.3	63 47	7.4 6.8	53 51	9.1 10.0	37 21
2.	40	80	101.0	15.20	ctr. of grav. cent-dian	846 846	20.9 20.3	77 59	16.9 13.2	59 60	16.1 15.6	44 34
3.	40	120	51.8	8.19	ctr. of grav. cent-dian	1276 1276	32.7 35.9	85 74	28.7 27.5	44 50	23.2 26.0	40 41
4.	40	160	45.94	6.95	ctr. of grav. cent-dian	2521 2521	61.1 66.4	87 83	54.4 57.4	50 52	38.0 42.7	44 44
5.	40	200	41.17	4.93	ctr. of grav. cent-dian	3366 3366	143.8 88.8	92 81	77.4 68.7	49 53	49.4 53.7	44 47

*Networks were generated in the same way as those for Table 2-1.

**On a DEC-10 computer.

Table 2-3. Computational results for four rail and water networks,* showing percent of tree-like segments examined and CPU time** in seconds for the center of gravity and cent-dian objective functions.

	Number of Nodes	Arcs	z_m^* (10 ³)	z_c^*	Objective Function	No Elimination:		Percent of Segments Examined, CPU Time Using Elimination Technique					
						Segments	CPU	No. 1		No. 2		Nos. 1, 2	
								%	CPU	%	CPU	%	CPU
1.	116	122	32.05	652.3	ctr. of grav. cent-dian	196 196	29.4 29.9	89 22	29.1 21.1	21 46	16.5 22.3	21 17	16.9 21.4
2.	66	72	3.18	156.2	ctr. of grav. cent-dian	128 128	8.7 8.6	66 32	7.1 5.7	44 52	6.3 6.7	33 17	5.9 4.8
3.	130	162	233.49	3211.8	ctr. of grav. cent-dian	447 447	69.6 63.3	11 18	33.3 33.9	85 86	65.1 59.9	7.6 17	32.3 33.9
4.	55	69	10.70	336.6	ctr. of grav. cent-dian	155 155	7.4 7.3	76 77	6.5 6.4	43 40	6.3 6.3	39 33	6.3 5.8

*The networks are taken from an Oak Ridge National Laboratory data base (Peterson [1984]). Specifically, they are:

1. Denver and Rio Grande Western Railroad.
2. Duluth, Missabe and Iron Range Railway.
3. U.S. coastal merchant marine shipping lanes.
4. Great Lakes shipping lanes.

Rail arc lengths are impedances based on the length and class of the link. Waterway arc lengths are travel distances in miles.

**On a DEC-10 computer.

Table 2-4. Computational results for three U.S. interstate networks,* showing percent of tree-like segments examined and CPU time** in seconds for the center of gravity and cent-dian objective functions.

Number of Nodes	Arcs	z_m^* (10^5)	z_c^* (miles)	Objective Function	No Elimination:		Percent of Segments Examined, CPU Time Using Elimination Technique					
					Segments	CPU	No. 1		No. 2		Nos. 1, 2	
							%	CPU	%	CPU	%	CPU
1.	109	152	201.18	1443.5	ctr. of grav. cent-dian	878 878	21 19	98.8 95.6	48.9 43.9	65 68	84.0 79.0	13 12
2.	175	280	155.23	1078.5	ctr. of grav. cent-dian	2196 2196	17 15	349.7 329.1	142.1 127.1	50 53	265.3 249.9	8 9
3.	276	432	506.21	1892.5	ctr. of grav. cent-dian	4264 4264	*** ***	*** ***	*** ***	*** ***	*** ***	6 7

*The networks are depicted by Rand McNally [1984, pp. 2-3]. Specifically, they are:

1. The U.S. west of the Mississippi (except Alaska and Hawaii), including cities along the Mississippi (except New Orleans).
2. The U.S. east of the Mississippi, including cities along the Mississippi.
3. The continental 48 states.

Arc lengths are travel distances in miles. Node weights are populations of cities (if any) at the nodes.

**On a DEC-10 computer.

***Computation omitted because of excessive CPU time.

The two rail networks of Table 2-3 were chosen from a data base covering the track systems of all U.S. railroads (Peterson [1984]) because these two are the densest rail networks of a reasonable size; railroad networks tend to be near-trees. The two waterway networks are of comparable density. The interstate highway networks of Table 2-4 are taken from a Rand-McNally road atlas [1984]. Note that upper bounding performs substantially better than on random networks. The occasional exception can be attributed to the fact that the effectiveness of upper bounding depends largely on whether one is lucky enough to find a low relative minimum early in the solution process. Although Technique 2 performs no better than before, it provides the valuable service of eliminating about half the segments that are not already ruled out by the upper bound. The result is the desired order-of-magnitude improvement on the highway networks and improvement by a factor of five or so on the other networks.

The segment choice parameter is defined in the documentation of procedure SPICK in the Appendix. All the foregoing results were gotten with the parameter set to zero. Table 2-5 suggests that Technique 2 may perform somewhat better if the parameter is set near 0.3 on random networks, but that Techniques 1 and 2 in combination work nearly as well with the parameter at zero as at some other setting. Yet the same table suggests that both techniques may perform better on actual networks when the parameter is near one. This hypothesis is tested in Table 2-6, which shows that the algorithm performs better on some but not all actual networks when the choice parameter is set to one.

Table 2-5. Effect on segment choice parameter on computation speed, showing number of segments examined and CPU time in seconds for the center of gravity problem on two networks. Network A is the random network on 40 nodes and 80 arcs that appears in Tables 2-1 and 2-2 (pp. 56, 58); network B is the U.S. interstate highway network west of the Mississippi (Table 2-4).

Parameter	Network A, Using Elimination Technique				Network B, Using Elimination Technique			
	No. 1		Nos. 1, 2		No. 1		Nos. 1, 2	
	Segments	CPU	Segments	CPU	Segments	CPU	Segments	CPU
0	502	16.1	370	12.1	571	84.0	118	45.0
0.1	507	16.2	346	11.6	541	83.2	150	49.2
0.2	475	15.4	346	11.6	530	81.7	148	49.2
0.3	483	15.5	372	12.3	507	80.8	145	48.7
0.4	490	15.8	366	12.2	493	69.7	136	47.6
0.5	494	15.9	389	13.0	488	78.9	141	48.6
0.6	505	16.4	390	13.1	483	79.4	136	48.3
0.7	518	17.0	390	13.3	481	77.1	131	47.5
0.8	520	17.4	399	13.6	473	78.9	108	45.7
0.9	536	18.3	416	14.4	467	78.2	70	40.7
1.0	539	19.0	410	14.7	449	66.3	70	40.5

Table 2-6. Effect of segment choice parameter on computation speed for seven actual networks. The networks are identical to those of like size appearing in previous tables. The objective function is the center of gravity function.

Network	Nodes	Arcs	Segment Choice Parameter			
			Zero		One	
			Percent Segments Examined	CPU Time	Percent Segments Examined	CPU Time
Railroad	166	122	21	16.9	21	18.0
Railroad	66	72	34	5.9	35	6.3
Coastal shipping	130	162	8	32.3	6	31.8
Great Lakes	55	69	39	6.3	43	6.5
Interstate Highways:						
Western U.S.	109	152	13	45.0	8	40.5
Eastern U.S.	175	280	8	123.0	9	140.0
Lower 48 states	276	432	6.4	414.2	5.7	411.5

Since it may be a matter of interest, the center of gravity and cent-dians of the interstate highway networks are displayed in Table 2-7 and Fig. 2-7. The cent-dians warrant two observations. Note first that the cent-dian for the network west of the Mississippi lies near El Paso, Texas, for $0 \leq \beta \leq 0.25$ and skips to the Denver area for higher values of β . This behavior is not unusual and dictates that if one wishes to compromise the equity and efficiency objectives, he should not have them "meet halfway" by locating between El Paso and Denver. The intermediate locations are worse than both the extremes, however one weights the objectives. Rather, one should locate near El Paso if he values efficiency considerably more than equity ($\beta \leq 0.25$) and near Denver otherwise. (An examination of the computer printout in such a case may reveal that the intermediate locations, though worse, are only slightly worse than the extremes.)

A second observation is that the cent-dian for the eastern U.S. follows an odd J-shaped path as β increases. It hops west from Harrisburg to a point near Pittsburgh; further west to Cambridge, Ohio; down to a point 28 miles south of Cambridge; and back southeast to a point 187 miles south of Pittsburgh. The cent-dian for the 48 states follows a more predictable path from the median at Indianapolis to the center near Salina, Kansas. There is no reason in general, of course, that the cent-dians of a network should follow any kind of recognizable path at all.

Table 2-7. Center of gravity and cent-dians for three U.S. interstate highway networks.*

Network	Center of Gravity	Cent-dians
1. U.S. west of the Mississippi	Las Cruces, NM (intersection of I-10 and I-25, 12 miles north of El Paso, TX, on I-25)	$0 \leq b \leq .25$ (median): Los Cruces, NM $.25 \leq b \leq .33$: Denver, CO $.33 \leq b \leq 1$ (center): 12.5 miles east of Denver on I-70
2. U.S. east of the Mississippi	Cambridge, OH (intersection of I-70 and I-77, 79 miles east of Columbus on I-70)	$0 \leq \beta \leq .08$ (median): Harrisburg, PA $.08 \leq \beta \leq .14$: Washington, PA (25 miles south of Pittsburgh) $.14 \leq \beta \leq .64$: Cambridge, OH $.64 \leq \beta \leq .94$: 28.5 miles south of Cambridge, OH, toward Charleston, WV, on I-77 $.94 \leq \beta \leq 1$ (center): 48.5 miles north of Charleston, WV, toward Pittsburgh on I-79
3. The continental 48 states	St. Louis, MO	$0 \leq \beta \leq .12$ (median): Indianapolis, IN $.12 \leq \beta \leq .20$: intersection of I-57 and I-70, 80 miles SE of Springfield, IL, 138 miles west of Indianapolis toward St. Louis on I-70 $.20 \leq \beta \leq .39$: St. Louis, MO $.39 \leq \beta \leq .46$: Topeka, KS $.46 \leq \beta < 1$ (center): 38½ miles east of Salina, KS, on I-70

*The networks are depicted by Rand McNally [1984, pp. 2-3]. Specifically, they are:

1. The U.S. west of the Mississippi (except Alaska and Hawaii), including cities along the Mississippi (except New Orleans).
2. The U.S. east of the Mississippi, including cities along the Mississippi.
3. The continental 48 states.

Arc lengths are travel distances in miles. Node weights are populations of cities (if any) at the nodes.

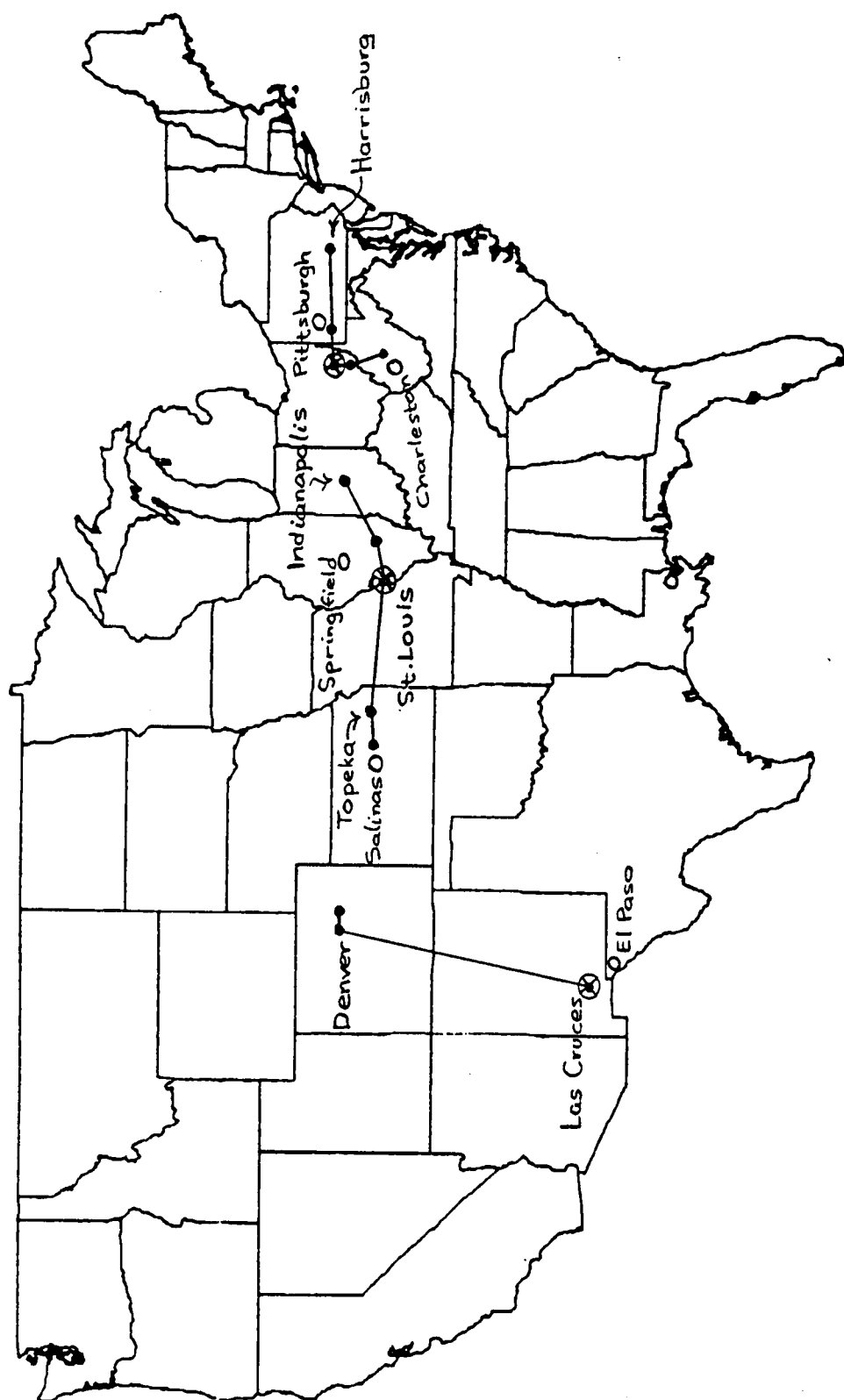


Fig. 2-7. Centers of gravity and cent-dians of three U.S. interstate highway networks. Centers of gravity for the U.S. west of the Mississippi (except Alaska and Hawaii), the entire lower 48, and the U.S. east of the Mississippi are respectively marked by encircled X's, moving left to right. Dots connected by line segments depict cent-dians for the three networks. Table 2-7 indicates their precise locations and the values of β for which they apply.

CHAPTER 3

THE MULTIPLE-FACILITY PROBLEM

We now take up the task of solving multiple-facility network location problems in which each node of the network is served by a closest facility. The object is to find locations for p facilities that minimize some convex function of the distances between the nodes and the facilities serving them. We are therefore dealing with a generalization of the p -median problem, in the sense that the linear cost function of the p -median problem is generalized to any convex function.

No attempt is made to solve location-allocation problems in which an optimal assignment of facilities to nodes is sought, nor to solve problems in which new facilities interact with existing facilities regardless of their locations. Nonetheless, problems in which new facilities are located among existing facilities (and nodes are served by a closest facility) form a subclass of the problems considered here. We also accommodate the possibility that facilities may be excluded from certain arcs or from certain open intervals within arcs.

Again the general approach is to decompose the network into treelike components and to solve convex problems on the components, but with two complications. It is now necessary to enumerate p -tuples of components, and the assignment of facilities to nodes varies in complex ways as the facilities move over the network. For these reasons, especially the latter, the p -facility problem has a much stronger combinatorial flavor than the single-facility problem and is correspondingly harder to solve.

It must be emphasized that the algorithms to follow are intended for small networks, perhaps 20 or 30 nodes, since computation time may otherwise be excessive. The algorithms are designed not so much to be efficient as simply to obtain solutions, albeit for small problems, where none were obtainable before. We argue in Section 1.2, however, that it is probably only on small networks that one is justified in using an exact algorithm to solve a nonlinear p -facility problem. We also argue that there is little sense in designing a heuristic rather than an exact algorithm, since if one desires only an approximate solution he may just as well solve his problem on nodes with one of the p -median algorithms.

It is instructive to begin in Section 3.1 with a simple two-facility example, to illustrate the inherent complexity of the p -facility problem and how one may obtain an exact solution. In Section 3.2 we develop a formal algorithm for solving the general convex problem by exhaustive enumeration of "treelike subregions" of the p -fold cartesian product of the network with itself. In Section 3.3 we generalize the upper-bounding technique of Section 2.3.

Although efficiency is not our primary goal, we investigate in Section 3.4 the extent to which the segment-elimination technique of Section 2.5 can be applied to p -facility problems with convex, non-decreasing and semiseparable cost functions. We find that the main single-facility result, Theorem 2.2, has a weaker and more complicated counterpart in the multiple-facility setting. We also present two formal algorithms that exploit this result in different ways, one for

locating one facility among several existing facilities, and one for locating several facilities, possibly among other existing ones. Still, the primary aim of this last section is to answer the natural question as to whether the single-facility results can be extended to multiple facilities, rather than to invent algorithmic improvements of demonstrable value.

3.1 An Illustration

As usual we let $v_1, \dots, v_n$ be the nodes of network G and let G^p be the p -fold cartesian product of G with itself. For $x \in G^p$ we let

$$d_x(v_i) = \min_{q=1, \dots, p} \{d(x_q, v_i)\}.$$

We wish to solve the p -facility problem,

$$\min_{x \in DCG^p} z(x) = f(d_x(v_1), \dots, d_x(v_n)), \quad (3.1)$$

where the cost function f is convex. The feasible region D is the result of excluding facilities from finitely many open intervals on G .

It is illuminating to study a simple two-facility problem and to use it as a base for generalization. Consider the network G of Fig. 3-1. Suppose we want to solve (3.1) on G , where the cost function is the weighted center-of-gravity function

$$f(d_1, \dots, d_4) = d_1^2 + d_2^2 + d_3^2 + 25d_4^2.$$

To each vector $x = (x_1, x_2)$ of two facility locations there corresponds a partition of the nodes into the set $P_1(x)$ of nodes that are closer to x_1

than x_2 and the set $P_2(x)$ of nodes that are closer to x_2 . (When x_q and x_r are equidistant from v_1 , we can adopt the convention that $v_1 \in P_q(x)$ if $q < r$.)

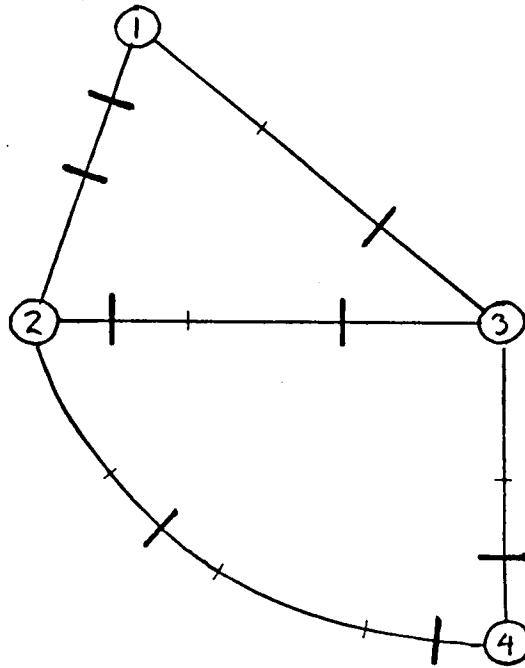


Fig. 3-1. A network for illustration of a two-facility location problem. The light tick marks indicate unit intervals on the arcs, and the heavy marks treelike segment boundaries.

Figure 3-2(a) depicts graphically all possible locations of two facilities on the network of Fig. 3-1. The horizontal axis indicates the position of facility 1, and the vertical axis that of facility 2. The diagram is divided into four large rectangles, each representing the cartesian product of two paths. The upper left rectangle, for instance, represents the product of the cycle 1-2-3-1 with the path 2-4-3.

(a)

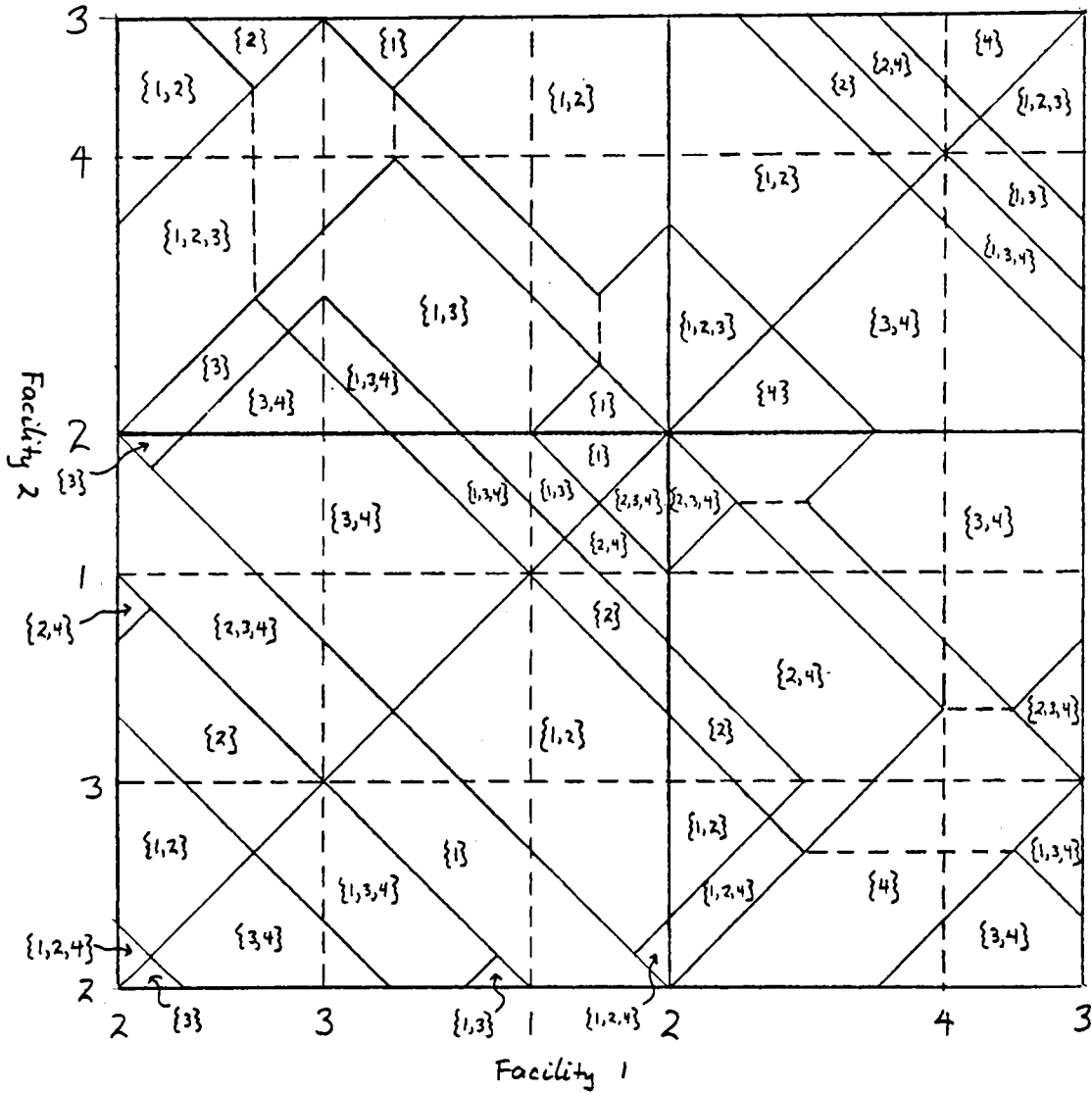


Fig. 3-2. Diagram of partition zones, treelike rectangles, treelike regions, and treelike subregions for the example of Fig. 3-1. Part (a) shows partition zones and treelike regions, while part (b) shows partition zones, treelike rectangles and treelike subregions.

(b)

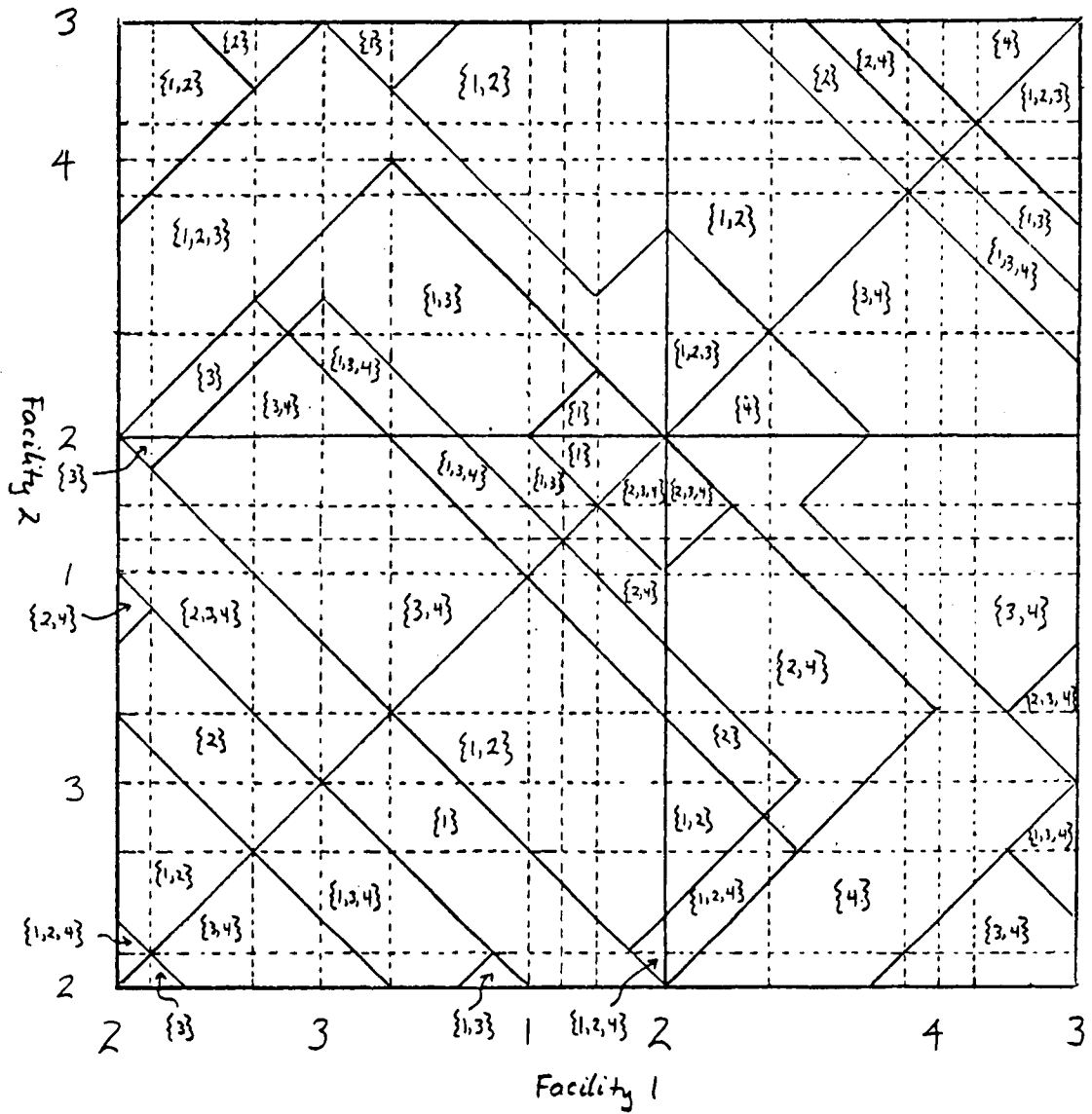


Fig. 3-2 (continued)

Thus every point $x \in G^2$ is represented by a point on the diagram. The smaller regions bounded by solid lines correspond to node partitions. The kite-shaped area in the upper left corner, for instance, represents the set of points x for which $P_1(x) = \{1,2\}$. We will refer to these regions as partition zones and to their boundaries as partition boundaries. Figure 3-2(b) depicts both partition boundaries and a rectangular grid of dotted lines, each line corresponding to a treelike segment boundary on one of the axes. Each small rectangle bounded by dotted (or solid) lines is therefore a cartesian product of two treelike segments. We call these rectangles treelike rectangles. Thus the kite-shaped partition zone just mentioned contains parts of seven treelike rectangles. Portions of the dotted lines of Fig. 3-2(b) appear in Fig. 3-2(a) as dashed lines. We will explain their meaning shortly.

Within any given partition zone the assignment of nodes to facilities is fixed. In such a zone the problem of finding an optimal location is no longer combinatorial in nature. This suggests that whereas a single-facility problem can be solved by decomposing G into treelike segments, a two-facility problem can be solved by decomposing G^2 into regions marked out by partition boundaries as well as treelike segment boundaries.

Let us first consider the cartesian product of the two arcs $(1,2)$ and $(3,4)$, represented by a square (call it square B) in the top center of Fig. 3-2(a). The entire square lies inside the partition zone labeled $\{1,2\}$. Although B contains six treelike rectangles, as depicted in Fig. 3-2(b), $d(x_1, v_1)$ and $d(x_2, v_1)$ are convex (in fact linear) for all points $x_1, x_2 \in G$ represented by points in B and all nodes v_1 . This is

because the treelike segment boundaries on (1,2) do not correspond to any node assigned facility 1 in B, and the segment boundaries on (3,4) do not correspond to any node assigned facility 2. Thus we can find an optimal location on (1,2) x (3,4) by solving the following easy convex problem with linear constraints:

$$\begin{aligned} \min_{(x_1, x_2) \in (1,2) \times (3,4)} z(x_1, x_2) = \min_{t_1, t_2} & t_1^2 + (2 - t_1)^2 + (2 - t_2)^2 + 25t_2^2 \\ \text{s.t.} & 0 \leq t_1 \leq 2 \\ & 0 \leq t_2 \leq 2, \end{aligned}$$

where t_1 is the distance between v_1 and x_1 along (1,2), and t_2 the distance between v_4 and x_2 along (3,4).

Next consider the rectangle B corresponding to (2,3) x (3,4), which contains portions of three partition zones and thus poses at least three location problems. The problem for the portion of B in zone {1,2} is the convex problem below. The first two constraints define the partition boundaries and the last two the boundaries of B.

$$\begin{aligned} \min_{t_1, t_2} & (t_1 + 2)^2 + t_1^2 + (2 - t_2)^2 + 25t_2^2 \\ \text{s.t.} & t_1 + t_2 \leq 3 \\ & t_1 - t_2 \leq 1 \\ & t_1 \geq 0 \\ & 0 \leq t_2 \leq 2. \end{aligned}$$

Here t_1 is the distance between v_2 and x_1 along (2,3), and t_2 is as before. The problem for the portion in zone {2} is similar. The problem for the portion in zone {1,2,3} is a bit different because on arc (2,3) node v_1 both is assigned to facility 1 and generates a treelike segment boundary. We indicate this pictorially with a vertical dashed line in zone {1,2,3} in Fig. 3-2(a). Thus to obtain convex location

problems we must divide the portion of partition zone $\{1,2,3\}$ lying in B into two treelike regions, lying on either side of the dashed line. The convex problems corresponding to these two treelike regions are

$$\begin{aligned} & \min_{t_1, t_2} (t_1 + 2)^2 + t_1^2 + (2 - t_2)^2 + 25t_2^2 \\ & \text{s.t. } t_1 - t_2 \leq 1 \\ & \quad 0 \leq t_1 \leq 2, \quad t_2 \geq 0, \\ \text{and} \\ & \min_{t_1, t_2} (6 - t_1)^2 + t_1^2 + (2 - t_2)^2 + 25t_2^2 \\ & \text{s.t. } t_1 - t_2 \geq 1 \\ & \quad 2 \leq t_1 \leq 3, \quad t_2 \geq 0, \end{aligned}$$

where t_1 and t_2 are as before.

It is evident that a small two-facility problem can be solved by drafting a diagram such as Fig. 3-2(a) and solving convex problems on treelike regions. Since the diagram is symmetric about the diagonal (corresponding to a permutation of facilities), one need enumerate only the treelike regions above the diagonal. In Fig. 3-2(a) there are 59 such regions.

We will see in the next section that when the solution procedure is cast as a formal algorithm suitable for computer implementation, it is convenient to enumerate treelike rectangles rather than partition zones. A treelike rectangle may contain portions of several partition zones, and we call these portions treelike subregions. In Fig. 3-2(b) they are the smallest regions bounded by dotted or solid lines. There are 158 of them above the diagonal. Figure 3-2(a), then, is appropriate to manual solution and Fig. 3-2(b) to computer solution.

3.2 Solution of the General Convex Problem

We wish now to develop an exact algorithm for solving (3.1) when f is any convex function. The general strategy is to enumerate treelike subregions and to solve a convex problem on each subregion: the best subregional solution solves the original problem. Specifically, we enumerate cartesian products of p treelike segments (treelike rectangles), and then for each rectangle we enumerate certain facility-to-node assignments corresponding to partition zones that may overlap the rectangle; the result, in effect, is an enumeration of the treelike subregions within the rectangles.

We first describe how to enumerate subregions and then how to solve a location problem on a subregion. The section closes with a formal statement of the algorithm.

Suppose we examine a p -tuple of arcs $((u_1, w_1), \dots, (u_p, w_p))$ with intent to identify treelike subregions. Let $b_q = b(u_q, w_q)$ be the length of the q th arc in the p -tuple. With each arc (u_q, w_q) we associate an interval $[0, b_q]$, so that every point in the cartesian product of these p arcs corresponds to a point in $B = [0, b_1] \times \dots \times [0, b_p]$. It will be convenient to denote the point on (u_q, w_q) corresponding to $t_q \in [0, b_q]$ by $x_q(t_q)$; the point in the cartesian product of arcs corresponding to $t = (t_1, \dots, t_p) \in B$ is $x(t) = (x_1(t_1), \dots, x_p(t_p))$.

We first locate the treelike rectangles of B . To do this, compute for each (u_q, w_q) , $q = 1, \dots, p$, the numbers $s_{qi} = \frac{1}{2}[d(w_q, v_i) - d(u_q, v_i) + b_q]$ for $i = 1, \dots, n$, as discussed in Section 2.2. For each q let $s'(q, 0), \dots, s'(q, m_q)$ be the distinct s_{qi} 's in nondecreasing order. By

Corollary 2.3 the intervals on (u_q, w_q) corresponding to $[s'(q, k-1), s'(q, k)]$ for $k = 1, \dots, m_q$ are the treelike segments of (u_q, w_q) . Thus given any node v_i , distances from v_i to all points y in the treelike segment corresponding to $[s'(q, k-1), s'(q, k)]$ are measured through the same endpoint of the segment, whether it be $x_q(s'(q, k-1))$ or $x_q(s'(q, k))$. Thus we can say (Fig. 3-3) that if $t \in [s'(q, k-1), s'(q, k)]$, then for $i = 1, \dots, n$,

$$d(x_q(t), v_i) = d(y_{qki}, v_i) + h_{qki}(t), \quad (3.2)$$

where

$$y_{qki} = \begin{cases} u_q & \text{if } d(v_i, x_q(s'(q, k))) \geq d(v_i, x_q(s'(q, k-1))) , \\ w_q & \text{otherwise} , \end{cases} \quad (3.3)$$

and

$$h_{qki}(t) = \begin{cases} t & \text{if } y_{qki} = u_q , \\ b_q - t & \text{otherwise} . \end{cases} \quad (3.4)$$

For future reference we define for $i = 1, \dots, n$ (as illustrated in Fig. 3-3):

$$s'_{qki} = \begin{cases} s'(q, k-1) & \text{if } y_{qki} = u_q , \\ b_q - s'(q, k) & \text{if } y_{qki} = w_q , \end{cases} \quad (3.5)$$

and

$$c_{qk} = s'(q, k) - s'(q, k-1) . \quad (3.6)$$

For each one (u_q, w_q) let us choose a treelike segment corresponding to the interval $[s'(q, k_q - 1), s'(q, k_q)]$. (For typographical reasons we will sometimes drop the subscript q from k_q .) We refer to the cartesian product R of these p intervals as a treelike rectangle within B . Figure 3-4 shows the six treelike rectangles in the cartesian product B of the intervals $[0, 3]$ and $[0, 2]$ corresponding to arcs (2, 3) and (3, 4) from the

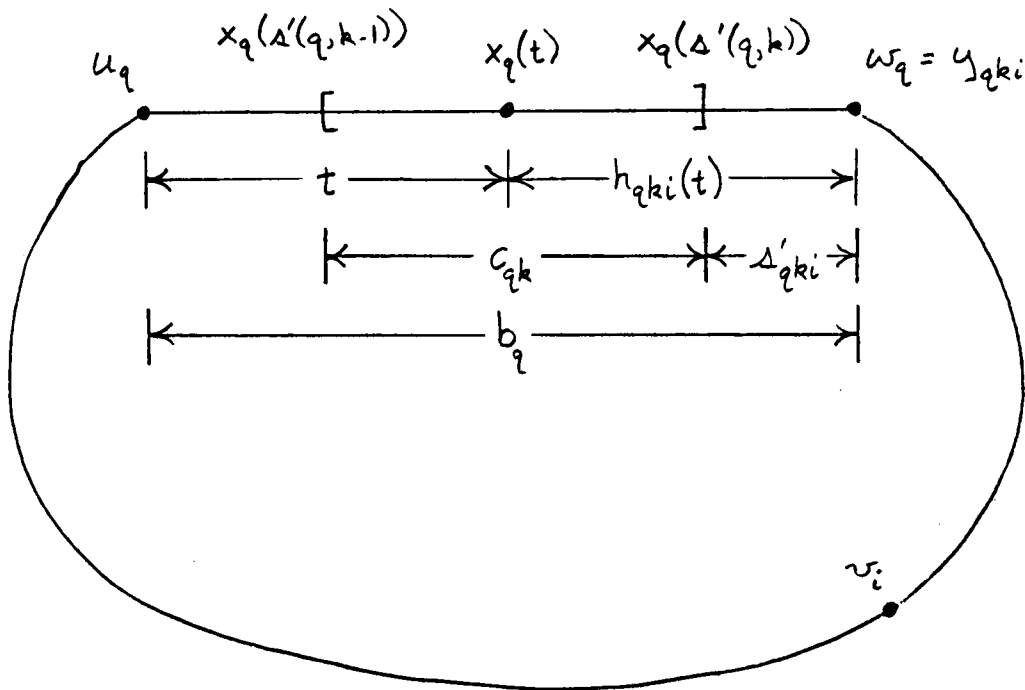


Fig. 3-3. Illustration of (3.2) in the case where $y_{qki} = w_q$.

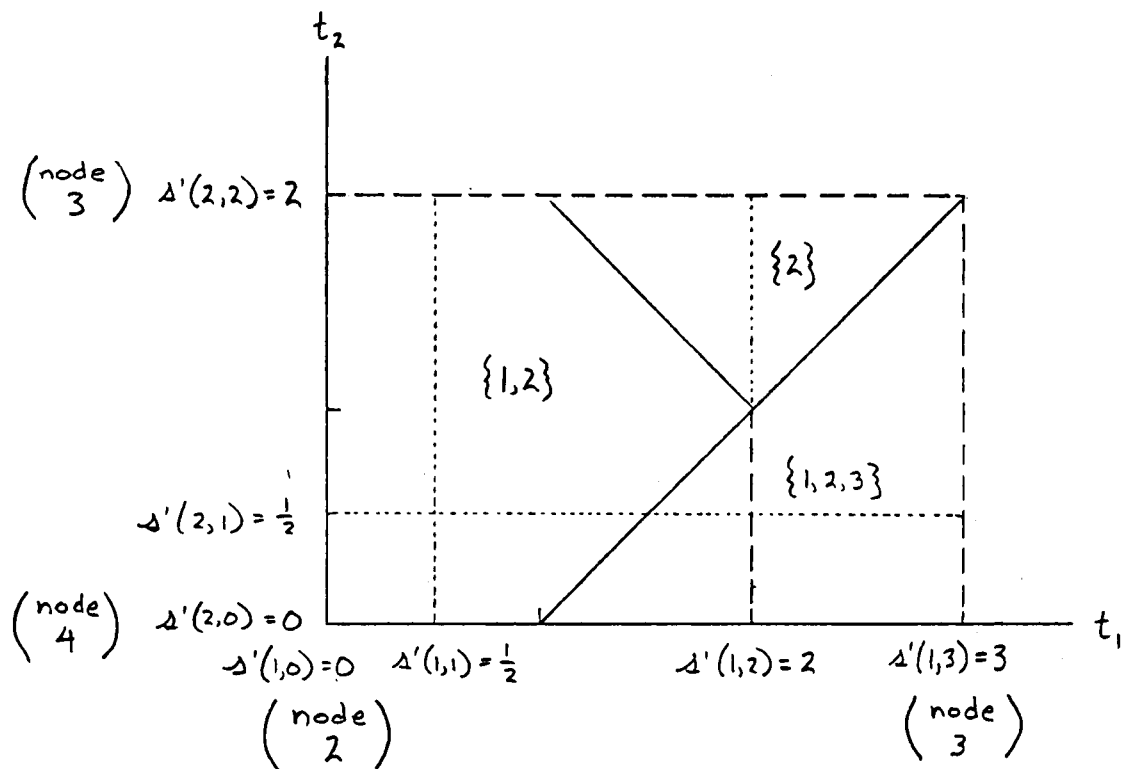


Fig. 3-4. Illustration of treelike rectangles, regions and subregions.

example of the previous section. The horizontal axis corresponds to (v_2, v_3) and the vertical axis to (v_3, v_4) . The treelike rectangles are separated by dashed and dotted lines. The significance of distinguishing dashed and dotted lines is explained in the previous section.

Now that we have identified the treelike rectangles of B , we can find B 's treelike subregions simply by locating the partition boundaries within each treelike rectangle. Consider the rectangle R defined earlier. We first identify the nodes that are assigned to the same facility at all points in R . We note that facility q is assigned to node v_i throughout R , or $v_i \in P_q(x(t))$ for all $t \in R$, if the location $x_q(t)$ of facility q is always closer to v_i than the locations of the other facilities. Therefore, it is not hard to see that $v_i \in P_q(x(t))$ for all $t \in R$ if

$$d(y_{qki}, v_i) + s'_{qki} + c_{qk} < \min_{\substack{r=1, \dots, p \\ r \neq q}} \{d(y_{rki}, v_i) + s'_{rki}\}. \quad (3.7)$$

In the example of Fig. 3-4, we see that node v_2 is assigned facility 1 throughout $[\frac{1}{2}, 2] \times [\frac{1}{2}, 2]$ because $d(y_{132}, v_2) + s'_{132} + c_{13} = d(v_2, v_2) + s'(1, 3) + c_{13} = 0 + 1/2 + 3/2 < 3 + 0 = d(v_3, v_2) + b_2 - s'(2, 2) = d(y_{222}, v_2) + s'_{222}$. By similar reasoning, node v_i is assigned facility q nowhere in R , i.e., $v_i \notin P_q(x(t))$ for all $t \in R$, if

$$d(y_{qki}, v_i) + s'_{qki} > \max_{\substack{r=1, \dots, p \\ r \neq q}} \{d(y_{rki}, v_i) + s'_{rki} + c_{rk}\}. \quad (3.8)$$

In the example, node 4 is assigned facility 1 nowhere in $[\frac{1}{2}, 2] \times [\frac{1}{2}, 2]$ because $d(y_{134}, v_4) + s'_{1,3,4} = d(v_3, v_4) + b_1 - s'(1,3) = 2 + 3 - 1 > 0 + 1/2 + 3/2 = d(v_4, v_4) + s'(2,2) + c_{22} = d(y_{224}, v_4) + s'_{224} + c_{22}$.

Let an assignment of facilities to nodes be a vector $a = (a(1), \dots, a(n))$, where $a(i)$ is the facility assigned to node v_i . An assignment is feasible in R if it obtains at some point in R. We observe that the treelike subregions of R correspond to feasible assignments in R; each subregion is the set of points where the corresponding assignment obtains. We also note that (3.7) and (3.8) impose necessary conditions for the feasibility of an assignment a:

$$\text{if } q \text{ satisfies (3.7) for } v_i, \text{ then } a(i) = q ; \quad (3.9)$$

$$\text{if } q \text{ satisfies (3.8) for } v_i, \text{ then } a(i) \neq q . \quad (3.10)$$

Each assignment satisfying (3.9) and (3.10) corresponds to a possible treelike subregion of R. We say "possible" because (3.9) and (3.10) are not sufficient for feasibility. We also let V^* be the set of nodes whose assignment of facilities is not determined on R by (3.9) and (3.10). In the example, assignments $a = (1,1,1,2)$, $(1,1,2,2)$, $(2,1,1,2)$ and $(2,1,2,2)$ satisfy (3.9) and (3.10) for the treelike rectangle $[\frac{1}{2}, 2] \times [\frac{1}{2}, 2]$. They correspond to regions labeled $\{1,2,3\}$, $\{1,2\}$, $\{2,3\}$, and $\{2\}$. Assignment $(2,1,1,2)$ is strictly speaking feasible although no subregion is labeled $\{2,3\}$, since the single point at which the other subregions meet is its corresponding subregion. Finally, in this instance $V^* = \{1,3\}$.

Having located all the treelike subregions of R (plus some possible subregions that do not appear), we proceed to find an optimal location on each subregion. For each feasible assignment $a = (a(1), \dots, a(n))$ we have the convex problem,

$$\min_{t_1, \dots, t_p} f(d(v_1, x_{a(1)}(t_{a(1)})), \dots, d(v_n, x_n(t_{a(n)}))) , \quad (3.11)$$

$$\text{s.t.} \quad s'(q, k_q - 1) \leq t_q \leq s'(q, k_q), \quad q = 1, \dots, p , \quad (3.12)$$

$$\begin{aligned} d(v_i, x_{a(i)}(t_{a(i)})) &\leq d(v_i, x_q(t_q)) , \\ q &= 1, \dots, p, \quad \text{all } v_i \in V^* . \end{aligned} \quad (3.13)$$

Constraints (3.12) restrict the search to rectangle R , and constraints (3.13) make sure no facility is closer to a node v_i than the facility $a(i)$ that is assigned to v_i . Since (3.9)-(3.10) ensure the latter for nodes outside V^* , it is enough that (3.13) apply to nodes in V^* . If (3.11)-(3.13) has no feasible solution, then the assignment a is infeasible.

Problem (3.11)-(3.13) can be written

$$\begin{aligned} \min_{t_1, \dots, t_p} & f(d(v_1, y_{a(1)k_1}) + h_{a(1)k_1}(t_{a(1)}), \dots, d(v_n, y_{a(n)k_n}) \\ & + h_{a(n)k_n}(t_{a(n)})) \end{aligned} \quad (3.14)$$

$$\text{s.t.} \quad s'(q, k_q - 1) \leq t_q \leq s'(q, k_q) , \quad q = 1, \dots, p ,$$

$$\begin{aligned} d(v_i, y_{a(i)k_i}) + h_{a(i)k_i}(t_{a(i)}) &\leq d(v_i, y_{qk_i}) + h_{qk_i}(t_q) , \\ q &= 1, \dots, \text{all } i \in V^* . \end{aligned}$$

Since the arguments of the objective function of (3.14) are linear functions of t_q , $q = 1, \dots, p$, f is by Lemma 1.2 a convex function of $(t_1, \dots, t_p)$. Also the constraints of (3.14) are linear in t_q , $q = 1, \dots, p$. Returning to our example, we indicate below the problem (3.14) that corresponds to the treelike subregion of $[\frac{1}{2}, 2] \times [\frac{1}{2}, 2]$ (Fig. 2-5) that lies in partition zone $\{1, 2\}$:

$$\begin{aligned} \min_{t_1, t_2} \quad & f(d(v_1, v_2) + t_1, d(v_2, v_2) + t_1, d(v_3, v_3) + b_2 - t_2, d(v_4, v_4) + t_2), \\ \text{s.t.} \quad & s'(1, 1) \leq t_1 \leq s'(1, 2) \\ & s'(2, 1) \leq t_2 \leq s'(2, 2) \\ & d(v_1, v_2) + t_1 \leq d(v_1, v_3) + b_2 - t_2 \\ & d(v_3, v_3) + b_2 - t_2 \leq d(v_3, v_3) + b_1 - t_1. \end{aligned}$$

This reduces to

$$\begin{aligned} \min_{t_1, t_2} \quad & (2 - t_1)^2 + t_1^2 + (2 - t_2)^2 + 25t_2^2 \\ \text{s.t.} \quad & \frac{1}{2} \leq t_1 \leq 2 \\ & \frac{1}{2} \leq t_2 \leq 2 \\ & t_1 + t_2 \leq 3 \\ & t_1 - t_2 \leq 1. \end{aligned}$$

Optimal locations for two facilities on the network of Fig. 3-1 (p. 70) can be found by solving 158 minimization problems of the form (3.14).

We now present an algorithm for solving the convex p -facility problem. As in the single-facility case, we can exclude facilities from certain arcs by marking them as "infeasible" and passing over them in

Step 1. We can also exclude facilities from open intervals within arcs by inserting a node at either end of each interval and regarding the new arc in between as infeasible. Finally, we provide for the possibility that some facilities are fixed at certain locations in advance.

Step 0. Let $A(1), \dots, A(m_0)$ be the feasible arcs, and let facilities $\hat{p} + 1, \dots, p$ be fixed at locations $x_{\hat{p}+1}^*, \dots, x_p^*$ in advance. Set the relative minimum counter K to zero, and set $z_{\min} = \infty$. Calculate $d(x_q^*, v_i)$ for $i = 1, \dots, n$ and $q = \hat{p} + 1, \dots, p$.

Step 1. Choose a $\hat{p}$ -tuple of arcs $(A(i_1), \dots, A(i_{\hat{p}})) = ((u_1, w_1), \dots, (u_{\hat{p}}, w_{\hat{p}}))$ not already chosen, where $1 \leq i_q \leq m_0$ and $i_q \leq i_{q-1}$ for $q = 1, \dots, \hat{p}$. If none remain, go to Step 7. Let $b_q = b(u_q, w_q)$ for $q = 1, \dots, \hat{p}$.

Step 2. Calculate $d(u_q, v_i)$, $d(w_q, v_i)$ and $s_{qi} = \frac{1}{2}[d(w_q, v_i) - d(u_q, v_i) + b_q]$ for $i = 1, \dots, n$, $q = 1, \dots, \hat{p}$. For each $q \in \{1, \dots, p_0\}$ arrange the s_{qi} 's in ascending order, and let $s'(q, 0), \dots, s'(q, m_q)$ be the distinct members of the ordered list.

Step 3. Choose a treelike rectangle by choosing a p_0 -tuple of integers $(k_1, \dots, k_{\hat{p}})$ that has not already been chosen, with $0 \leq k_q \leq m_q$ for $q = 1, \dots, \hat{p}$. If none remain go to Step 1. (From here out, if k appears without a subscript, k_q is understood.) For $i = 1, \dots, n$ and $q = 1, \dots, \hat{p}$ calculate $d(v_i, x_q(s'(q, k_q))) = \min \{d(v_i, u_q) + s'(q, k_q), d(v_i, w_q) - b_q - s'(q, k_q)\}$. Define y_{qki} , $h_{qki}(t)$ and s'_{qki} as in (3.2)-(3.5), $i = 1, \dots, n$, $q = 1, \dots, \hat{p}$, and define c_{qk} as in (3.6), $q = 1, \dots, \hat{p}$. For $q = \hat{p} + 1, \dots, p$ define $c_{qk} = 0$ and $y_{qki} = x_q^*$, $s'_{qki} = 0$, and $h_{qki}(t) = 0$, $i = 1, \dots, n$.

Step 4. Let V^* be the set of nodes that satisfy (3.7) for no facility q and that fail to satisfy (3.8) for at least two facilities q , $q \in \{1, \dots, p\}$.

Step 5. Choose a (possible) treelike subregion by making an assignment $a(1), \dots, a(n)$ of facilities to nodes that is consistent with (4.11) and (4.12) and that has not already been made. If none remain, go to Step 3.

Step 6. Solve problem (3.14), replacing p with $\hat{p}$ in the objective function and first set of constraints. Set $K = K + 1$ and let x_K^* be the solution of (3.14), with $z_K^* = z(x_K^*)$. Set $z_{\min} = \min \{z_{\min}, z_K^*\}$ and go to Step 5.

Step 7. Any solution x_K^* such that $z_K^* = z_{\min}$ solves (3.1).

3.3 Improvement with an Upper Bounding Technique

It is possible to eliminate p -tuples of arcs with an upper-bounding technique that generalizes the technique of Section 2.3. Again we cover the cases in which f is differentiable or is the weighted maximum function. The method must be adapted to other nondifferentiable functions on a case-by-case basis.

Let $z_{\min}$ be the smallest local minimum for $z(x_1, \dots, x_p)$ discovered on treelike subregions hitherto examined. Thus $z_{\min}$ is an upper bound on the global minimum of $z(x_1, \dots, x_p)$. When a new p -tuple of arcs $((u_1, w_1), \dots, (u_p, w_p))$ is considered, we may be able to determine

that $z(x_1, \dots, x_p)$ cannot fall below $z_{\min}$ when $(x_1, \dots, x_p)$ lies in the cartesian product of these arcs, so that the p -tuple can be eliminated. The situation is complicated by the fact that the assignment of facilities to nodes may differ from one place to another in the cartesian product.

We must first develop some notation. We again associate interval $[0, b_q]$ with (u_q, w_q) , where $b_q = b(u_q, w_q)$ is the length of (u_q, w_q) , $q = 1, \dots, p$. We set $s_{qi} = \frac{1}{2}[d(w_q, v_i) - d(u_q, v_i) + b_q]$ for $i = 1, \dots, n$, $q = 1, \dots, p$. For $t = (t_1, \dots, t_p) \in B = [0, b_1] \times \dots \times [0, b_p]$ we let $a(i; t)$ be the facility assigned to node v_i when the facilities lie at $(x_1(t_1), \dots, x_p(t_p))$. Thus if $q = a(i; t)$,

$$d(x_q(t_q), v_i) \leq d(x_r(t_r), v_i) \quad (3.15)$$

for all $r \neq q$. Finally, let $d_i = d(u_{a(i,0)}, v_i)$, $i = 1, \dots, n$, where 0 is a vector of p zeros.

It will also prove convenient to let B_c be the set of corners of B ; that is, B_c is the set of points $(t_1, \dots, t_p) \in B$ such that each t_q is zero or b_q . For $i = 1, \dots, n$ we set $a(i) = a(i; 0)$ and

$$A(i) = \bigcup_{t \in B_c} a(i; t).$$

Thus $A(i)$ is the set of facilities assigned to v_i at some corner of B . We can show now that $A(i)$ is likewise the set of facilities assigned to v_i at some point of B (corner or otherwise). That is,

$$A(i) = \bigcup_{t \in B} a(i; t), \quad i = 1, \dots, n. \quad (3.16)$$

For suppose $q = a(i; t)$ for some noncorner $t \in B$; it suffices to show that $q \in A(i)$. But we know (3.15) holds for all $r \neq q$. Since $d(x_r(t_r), v_i)$ attains a minimum on (u_r, w_r) at $t_r = 0$ or $t_r = b_r$, (3.15) holds for all $r \neq q$ when $t_r = 0$ or $t_r = b_r$. Thus $q = a(i; t)$ for some corner t of B , so that $q \in A(i)$.

The upper bounding technique is based on the following generalization of Lemma 2.2. Again let $f_1(d) = \partial f / \partial d_1$ evaluated at $d = (d_1, \dots, d_n)$.

Lemma 3.1. Let f in (3.1) be differentiable and convex, and let $t' = (t'_1, \dots, t'_p) \in B$. Let $v_{m+1}, \dots, v_n$ be the nodes v_i for which $s_{a(i)i} = b_{a(i)}$ and $a(i; t) = a(i)$ for every $t \in B_c$. Then we have the bound,

$$z(x_1(t'_1), \dots, x_p(t'_p)) \geq z(u_{a(1);0}, \dots, u_{a(p);0}) - \sum_{q=1}^p c_q t'_q, \quad (3.17)$$

where

$$c_q = \sum_{\substack{i=1 \\ q \in A(i)}}^m |f_1(d)| - \sum_{\substack{i=m+1 \\ q=a(i)}}^n f_1(d). \quad (3.18)$$

Proof. We first show that

$$d_{x(t')}(v_i) = d_i + t'_{a(i)}, \quad i = m+1, \dots, n. \quad (3.19)$$

Note that since $s_{a(i)i} = b_{a(i)}$ when $i = m+1, \dots, n$, a shortest path from $x_{a(i)}(t'_{a(i)})$ to v_i passes through $u_{a(i)}$. Thus

$$d(x_{a(i)}(t'_{a(i)}), v_i) = d_i + t'_{a(i)}. \quad (3.20)$$

But we are also given that $a(i;t) = a(i)$ for all $t \in B_c$. Thus v_i is closer to facility $a(i)$ than to the others when $t \in B_c$, which is to say in particular that when $t_{a(i)} = b_{a(i)}$,

$$d(x_{a(i)}(t_{a(i)}), v_i) \leq d(x_r(t_r), v_i) \quad (3.21)$$

for all $r \neq a(i)$ and $t_r \in [0, b_r]$. Thus (3.21) holds for $t_{a(i)} = b_{a(i)}$ and all $t_r \in [0, b_r]$, $r \neq a(i)$. Since by (3.20) the left side of (3.21) increases with $t_{a(i)}$, (3.21) holds for all $t_{a(i)} \in [0, b_{a(i)}]$ as well as all $t_r \in [0, b_r]$, $r \neq a(i)$. Thus facility $a(i)$ is assigned to v_i at all points in B . This and (3.20) imply (3.19), as desired.

Next we show that

$$d_{x(t^*)}(v_i) = d_i - t_i^*, \quad i = 1, \dots, m, \quad (3.22)$$

where

$$-\max_{q \in A(i)} \{t_q\} \leq t_i^* \leq \max_{q \in A(i)} \{t_q\}. \quad (3.23)$$

Clearly $d_{x(t^*)}(v_i)$ is the distance from v_i to the closest facility in $A(i)$, since by (3.16) $A(i)$ contains all the facilities that are closest to v_i at some point in B . Thus,

$$d_{x(t^*)}(v_i) = \min_{q \in A(i)} \{d(x_q(t_q^*), v_i)\}. \quad (3.24)$$

But obviously for all $q \in A(i)$,

$$d(x_q(t_q^*), v_i) = d(u_q, v_i) - t_{qi}^*, \quad (3.25)$$

where $-t_{qi}^* \leq t_{qi}^* \leq t_{qi}^*$. (3.24) and (3.25) allow us to write

$$\begin{aligned} d_{x(t')}(v_i) &= \min_{q \in A(i)} \{d(u_q, v_i) - t'_{qi}\} = \min_{q \in A(i)} \{d(u_q, v_i)\} - t_i^* \\ &= d_i - t_i^*, \end{aligned}$$

where t_i^* satisfies (3.23). Thus we have shown (3.22).

Finally, since f is convex we know that for vectors d and Δd , $f(d + \Delta d) \geq f(d) + \nabla f(d) \cdot \Delta d$. Thus

$$\begin{aligned} f(d_1 - t_1^*, \dots, d_m - t_m^*, d_{m+1} + t'_{a(m+1)}, \dots, d_n + t'_{a(n)}) \\ \geq f(d) - \sum_{i=1}^m t_i^* f_i(d) + \sum_{i=m+1}^n t'_{a(i)} f_i(d). \end{aligned} \quad (3.26)$$

But by (3.23) we have

$$\sum_{i=1}^m t_i^* f_i(d) \leq \sum_{i=1}^m (\max_{q \in A(i)} \{t_q\}) |f_i(d)| \leq \sum_{i=1}^m \left(\sum_{q \in A(i)} t_q \right) |f_i(d)|.$$

This, (3.19), (3.22) and (3.26) imply that

$$\begin{aligned} z(x(t')) &= f(d_{x(t')}(v_1), \dots, d_{x(t')}(v_n)) \geq f(d) \\ &\quad - \sum_{i=1}^m \left(\sum_{q \in A(i)} t_q \right) |f_i(d)| + \sum_{i=m+1}^n t'_{a(i)} f_i(d), \end{aligned}$$

and (3.17) follows. □

We prove a similar result for the weighted maximum function, $f(x) = \max_i \{w_i x_i\}$.

Lemma 4.2. Let f in (3.1) be the weighted maximum function, and let t' and $v_{m+1}, \dots, v_n$ be as in Lemma 4.1. Then (3.17) holds when

$$c_q = \begin{cases} w_j & \text{if } 1 \leq j \leq m \quad \text{and } q \in A(j), \\ -w_j & \text{if } m+1 \leq j \leq n \quad \text{and } q = a(j), \\ 0 & \text{otherwise,} \end{cases} \quad (3.27)$$

where $w_j d_j = \max_{1 \leq i \leq n} \{w_i d_i\}$.

Proof. From (3.19) and (3.22) we have that

$$\begin{aligned} x(x(t')) &= \max_{1 \leq i \leq n} \{w_i d_{x(t')}(v_i)\} \geq \\ &\max \{ \max_{1 \leq i \leq m} [w_i (d_i - t_i^*)], \max_{m+1 \leq i \leq n} [w_i (d_i - t'_{a(i)})] \} \end{aligned} \quad (3.28)$$

where t_i^* satisfies (3.23). But clearly

$$\begin{aligned} &\max \{ \max_{1 \leq i \leq m} [w_i (d_i - t_i^*)], \max_{m+1 \leq i \leq n} [w_i (d_i - t'_{a(i)})] \} \\ &\geq \begin{cases} w_j d_j - w_j t_j^* & \text{if } 1 \leq j \leq m, \\ w_j d_j + w_j t'_{a(j)} & \text{if } m+1 \leq j \leq n. \end{cases} \end{aligned} \quad (3.29)$$

Since by (3.23)

$$w_j d_j - w_j t_j^* \geq w_j d_j - w_j \sum_{q \in A(j)} t'_q,$$

(3.28) and (3.29) imply (3.17).

□

The lower bound (3.17) on the value of $z(x(t))$ for $t \in B$ is based on the value of z at the corner $x(0) = (u_1, \dots, u_p)$ of B . Similar lower bounds can be got for each corner e of B . Suppose then we define $u(e) = (u_1(e), \dots, u_p(e))$ by

$$u_q(e) = \begin{cases} u_q & \text{if } e_q = 0, \\ w_q & \text{if } e_q = b_q, \end{cases} \quad (3.30)$$

and $t(e) = (t_1(e), \dots, t_p(e))$ by

$$t_q(e) = \begin{cases} t_q & \text{if } e_q = 0, \\ b_q - t_q & \text{if } e_q = b_q, \end{cases} \quad (3.31)$$

$q = 1, \dots, p$. Then we have,

Theorem 3.1. Let f in (3.1) be convex, and let f either be differentiable or be the weighted maximum function. Let t' and $v_{m+1}, \dots, v_n$ be as in Lemma 3.1. For $q = 1, \dots, p$ and $e \in B_c$ let $c_q(e)$ be c_q as given by (3.18) or (3.27), replacing (in Lemma 3.1 or 3.2) each u_q by $u_q(e)$ and each w_q by the node of (u_q, w_q) opposite $u_q(e)$. Then $z(x(t')) \geq z_0$ if z_0 is the minimum value of z in the linear program,

$$\begin{aligned} & \min_{t_1, \dots, t_p} z \\ \text{s.t.} \quad & 0 \leq t_q \leq b_q, \quad q = 1, \dots, p, \end{aligned} \quad (3.32)$$

$$z \geq z(u(e)) - \sum_{q=1}^p c_q(e) t_q(e), \quad \text{all } e \in B_c.$$

Proof. It suffices to observe that by (3.17),

$$z(x(t')) \geq z(u(e)) - \sum_{q=1}^p c_q(e) t'_q(e), \text{ for all } e \in B_c.$$

□

By the way of example consider the two-facility problem of the previous section. We will find z_0 for the pair of arcs (v_2, v_3) , (v_4, v_3) , for which B is illustrated in Fig. 3-5. Here $A(1) = \{1, 2\}$, $A(2) = \{1\}$, $A(3) = \{1, 2\}$, and $A(4) = \{2\}$, as is evident from the figure. For each corner $e = (e_1, e_2)$ Table 3-1 displays d_i , $f_i = f_i(d_1, d_2, d_3, d_4)$, and $a(i)$ for $i = 1, \dots, 4$, as well as c_1 and c_2 . The original node numbers are retained throughout. If the nodes were renumbered as specified in Lemma 3.1, then $v_{m+1}, \dots, v_n$ would be the nodes listed in the column headed "+". The linear program (3.30) that results is

$$\begin{aligned} & \min \quad z \\ & \quad t_1, t_2 \\ \text{s.t.} \quad & 0 \leq t_1 \leq 3 \\ & 0 \leq t_2 \leq 2 \\ & z \geq 8 - 8t_1 - 8t_2 \\ & z \geq 18 - 12t_1 - 6t_2 \\ & z \geq 104 - 8t_1 - 104t_2 \\ & z \geq 118 - 12t_1 - 106t_2. \end{aligned}$$

The optimal solution occurs at $(t_1, t_2) = (1, 1)$ with $z_0 = 0$, so that $((v_2, v_3), (v_4, v_3))$ certainly cannot be eliminated. The true minimum of $z(x(t))$ for $t \in B$ is $z(x(0, 0)) = 8$.

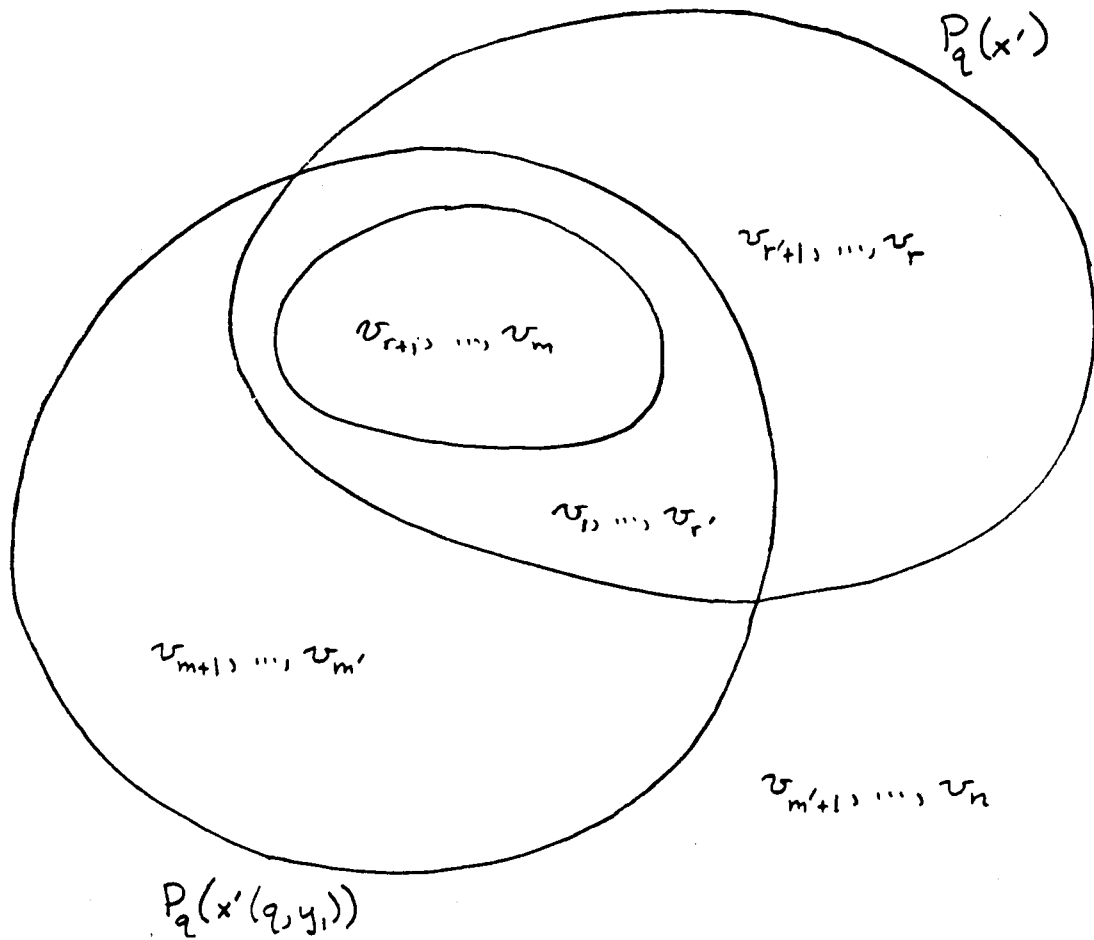


Fig. 3-5. Partitioning of nodes for Theorem 3.2.

Table 3-1. Application of Theorem 3.1 to the example described in the text.

(e_1, e_2)	(d_1, d_2, d_3, d_4)	$(a(1), a(2), a(3), a(4))$	(f_1, f_2, f_3, f_4)	+	c_1	c_2
(0,0)	(2,0,2,0)	(1,1,2,2)	(4,0,4,0)	2,4	$f_1 + f_3 - f_2 = 8$	$f_1 + f_3 - f_4 = 8$
(3,0)	(3,3,0,0)	(1,1,1,2)	(6,6,0,0)	4	$f_1 + f_2 + f_3 = 12$	$f_1 + f_3 - f_4 = 6$
(0,2)	(2,0,0,2)	(1,1,2,2)	(4,0,0,100)	2	$f_1 + f_3 - f_2 = 8$	$f_1 + f_3 + f_4 = 104$
(3,2)	(3,3,0,2)	(1,1,1,2)	(6,6,0,100)	none	$f_1 + f_2 + f_3 = 12$	$f_1 + f_3 + f_4 = 106$

The upper bounding technique can be made part of the algorithm of the previous section by replacing Step 2 with the following.

Step 2. Calculate distances $d(u_q, v_i)$ and $d(w_q, v_i)$, $i = 1, \dots, n$, for each $q \in \{1, \dots, \hat{p}\}$. If not using upper bounding, go to Step 3. Let B_c be the set of $\hat{p}$ -tuples $e = (e_1, \dots, e_{\hat{p}})$ such that $e_q = 0$ or b_q for $q = 1, \dots, \hat{p}$. Set $i = 0$.

Step 2a. Set $i = i + 1$. If $i > n$ go to Step 2e. Set $A(i) = \phi$.

Step 2b. For each corner $e \in B_c$ do the following. For $q = 1, \dots, \hat{p}$ let $u_q(e)$ and $t_q(e)$ be defined by (3.30) and (3.31). For $q = \hat{p} + 1, \dots, p$ let $u_q(e) = x'_q$ and $t_q(e) = 0$. Find $a(e; i) = \min \{q \mid 1 \leq q \leq p \text{ and } d(u_q(e), v_i) = \min_{1 \leq r \leq p} [d(u_r(e), v_i)]\}$, and set $A(i) = A(i) \cup \{a(e; i)\}$ and $d_i = d(u_{a(e; i)}(e), v_i)$, $i = 1, \dots, n$.

Step 2c. For each $e \in B_c$ do the following. Set $M(e; i) = \text{true}$; $M(e; i)$ will remain true if v_i would be one of the nodes $v_{m+1}, \dots, v_n$ if they were numbered as in Lemma 3.1. If $e_{a(e; i)} = 0$ and $s_{a(e; i)} i < b_{a(e; i)}$, or $e_{a(e; i)} = b_{a(e; i)}$ and $s_{a(e; i)} i > 0$, then $M(e; i) = \text{false}$. If $a(i; i) \neq a(e'; i)$ for some $e' \in B_c$ with $e'_{a(e; i)} = e_{a(e; i)}$, then $M(e; i) = \text{false}$.

Step 2d. Go to Step 2a.

Step 2e. For each q , $q = 1, \dots, p$, and each $e \in B_c$, calculate

$$c_q(e) = \sum_{\substack{i \\ M(e; i) = \text{false} \\ q \in A(i)}} f_i(d_1, \dots, d_n) - \sum_{\substack{i \\ M(e; i) = \text{true} \\ q = a(e; i)}} f_i(d_1, \dots, d_n)$$

where $d_i = d(u_{a(i;e)}, v_i)$, $i = 1, \dots, n$.

Step 2f. Solve the linear program (3.32), with $\hat{p}$ replacing p in the objective function and the first set of constraints. Let z_0 be the minimum value of z . If $z \geq z_{\min}$ go to Step 1.

3.4 Solution of the Convex, Nondecreasing, Semiseparable Problem

In our discussion of the semiseparable single-facility problem (Sect. 2.5), we showed the following. If x is an optimal location on a shortest-path tree $T(S)$ rooted at a treelike segment S , then no point on the path in $T(S)$ from S to x can be a better location on the original network than x . This allowed us to eliminate segments between S and x , including S .

Two similar, if weaker and more complicated results can be obtained in the p -facility case. If the cost function is again convex, nondecreasing and semiseparable, then one can eliminate segments somewhat as in the single-facility case if all but one of the facilities have locations fixed in advance. This is an important special case of the general problem, since very often one wants to add a new facility to existing ones, reassigning nodes that are closer to the new facility than to any of the old ones. The second result is that if the cost function is also additively separable, this elimination procedure can be generalized so as to be useful in the general case in which few or none of the facility locations are fixed in advance.

Suppose p facilities are located at $x' = (x'_n)$ and we wish to eliminate a portion of the network as a possible optimal location of facility q , given that the other facilities are fixed at locations $x'_1, \dots, x'_{q-1}, x'_{q+1}, \dots, x'_p$. Let $T_q(x')$ be a tree of shortest paths from $x'_q = x_q(t'_q)$ to all nodes in $P_q(x')$ (i.e., to all nodes assigned facility q when the facilities lie at $x'_1, \dots, x'_p$). Let $L_q^T(y; x')$ be the line segment (i.e., path) of $T_q(x')$ connecting x'_q with $y \in T_q(x')$. For $y_1, y_2 \in T_q(x')$ let $d_q^T(y_1, y_2; x')$ be the distance from y_1 to y_2 as measured on $T_q(x')$. It is convenient to write $x'(q, y)$ for $(x'_1, \dots, x'_{q-1}, y, x'_{q+1}, \dots, x'_p)$. Finally, let x_q^* be an optimal location on $T_q(x')$, in the sense that x_q^* solves the convex problem,

$$\min_{y \in T_q(x')} z_q^T(x'(y)) = f(d_{qx'(q,y)}^T(v_1), \dots, d_{qx'(q,y)}^T(v_n)) \quad (3.33)$$

where

$$d_{qx'(q,y)}^T(v_1) = \begin{cases} d_q^T(y, v_1; x') & \text{if } v_1 \in T_q(x') , \\ d_{x'}(v_1) & \text{otherwise .} \end{cases}$$

A result analogous to the single-facility result would be that $z(x'(q, y))$ is nonincreasing as y moves along the path $P = L_q^T(x_q^*; x')$ from x'_q to x_q^* . But as y moves along this path, there may be a change in nodes assigned facility q , and this complicates the picture. Nonetheless we will show that $z(x'(q, y))$ is nonincreasing so long as no new nodes are assigned facility q , even if facility q loses some nodes to other facilities. The result is in fact a bit stronger: if one looks only at the intervals on P of points y for which no nodes are assigned

facility q that were not assigned q from the start (at $y = x_q'$), then $z(x'(q,y))$ is nonincreasing as y moves all the way from x_q' to x_q^* .

If f is additively separable, we can get a closely related result. We say that $f: D \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is additively separable if there are functions $g_i: \mathbb{R} \rightarrow \mathbb{R}$, $i = 1, \dots, n$, such that $f(d_1, \dots, d_n) = g_1(d_1) + \dots + g_n(d_n)$. Roughly, the result is that as y moves from x_q' toward x_q^* , $z(x'(q,y))$ is nonincreasing so long as $d(y, v_i)$ is nonincreasing for all nodes v_i assigned to q as y moves along P . Additive separability of f is important because it guarantees that $f(d_1, \dots, d_n) \geq f(d_1, \dots, d_m, e_{m+1}, \dots, e_n)$ implies $f(d_1', \dots, d_m', d_{m+1}, \dots, d_n) \geq f(d_1', \dots, d_m', e_{m+1}, \dots, e_n)$ for all arguments d_1, d_1', e_1 .

These results say in effect that the solution improves as one moves any given facility q toward the corresponding tree solution x_q^* , subject to certain conditions. They do not say that the solution improves if one simultaneously moves several facilities toward the corresponding tree solutions, and indeed it does not in general. Thus these results are useful when one is locating facility q among $p - 1$ other fixed facilities. We will show later, however, that if f is additively separable, the solution improves even when several facilities are moved simultaneously toward the corresponding tree solutions, provided the facilities other than facility q do not move so far as to cross a partition boundary.

Formally, the results are as follows.

Theorem 4.2. Let z be defined as in (3.1), where f is convex, nondecreasing and semiseparable. Pick any location $x' \in G^P$ of facilities

and any particular facility q , and let x_q^* solve (3.33). Let E be the set of points y in $L_q^T(x_q^*; x')$ such that $P_q(x'(q, y)) \subseteq P_q(x')$, and denote by y' the point of E closest to x_q^* . If f is additively separable, let y'' be the point in $L_q^T(x_q^*; x')$ closest to x_q^* such that for each node v_i , $d(\cdot, v_i)$ is decreasing (as one moves away from x_q^*) at all points $y \in L_q^T(x_q^*; x')$ for which $v_i \in P_q(x'(q, y)) - P_q(x')$; otherwise let y'' be the point in $L_q^T(x_q^*; x')$ closest to x_q^* for which $L_q^T(y''; x') \subseteq E$. Then for any $y \in E$, $z(x'(q, y)) \geq z(x'(q, y'))$, and for any $y \in L_q^T(y''; x')$, $z(x'(q, y)) \geq z(x'(q, y''))$.

Proof. Define

$$\tilde{z}(x'(q, y)) = f(\tilde{d}_{x'(q, y)}(v_1), \dots, \tilde{d}_{x'(q, y)}(v_n)) , \quad (3.34)$$

where

$$\tilde{d}_{x'(q, y)}(v_i) = \begin{cases} d_{x'(q, y)}(v_i) & \text{if } v_i \in P_q(x') , \\ d_{x'}(v_i) & \text{otherwise .} \end{cases}$$

Thus $\tilde{z}(x'(q, y))$ and $z(x'(q, y))$ are equal when $y \in E$ but may differ otherwise. Note that to prove the theorem it suffices to show that $\tilde{z}(x'(q, y))$ is nonincreasing along $L_q^T(y'; x')$ as y moves away from x_q^* , and that $z(x'(q, y))$ is nonincreasing along $L_q^T(y''; x')$.

Let $w_0, \dots, w_k$ be the points along $L_q^T(y'; x')$ that are treelike segment boundaries, partition boundaries, or identical with y' or y'' . Clearly $L_q^T(y''; x') \subseteq L_q^T(y'; x')$, so that $y'' = w_{K'}$ for some K' . Since z and $\tilde{z}$ are continuous functions of y it is enough to show that $\tilde{z}$ is nonincreasing on each interval $A[w_{k-1}, w_k]$ for $k = 1, \dots, K$, and that z

is nonincreasing on each $A[w_{k-1}, w_k]$ for $k = 1, \dots, K'$. To establish this we show

$$\bar{z}(x'(q, y_1)) \geq \bar{z}(x'(q, y_2)) \quad (3.35)$$

for any interval $A[y_1, y_2] \subseteq A[w_{k-1}, w_k]$, $k = 1, \dots, K$, and

$$z(x'(q, y_1)) \geq z(x'(q, y_2)) \quad (3.36)$$

for any such interval, $k = 1, \dots, K'$.

Cut $T_q(x')$ at y_2 and let $v_1, \dots, v_r$ be the nodes in the part of $T_q(x')$ containing x'_q (excluding y_2 , if y_2 is a node), and let $v_{r+1}, \dots, v_m$ be the rest of the nodes in $T_q(x')$. We note first that $v_{r+1}, \dots, v_m$ must all be in $P_q(x'(q, y_1))$. For, each of these nodes v_i is in $P_q(x')$ and can fail to be in $P_q(x'(q, y_1))$ only if x'_q is closer to v_i than y_1 . But clearly $d_q^T(y_1, v_i; x') = d(y_1, v_i)$, since the path of $T_q(x')$ connecting y_1 and v_i is a shortest path. Thus x'_q is at least as far from v_i as y_1 .

Having shown that $v_{r+1}, \dots, v_m \in P_q(x'(q, y_1))$, we can partition the nodes as shown in Fig. 3-5 (p. 93). Let $d_i^T = d_q^T(y_2, v_i; x')$ for $i = 1, \dots, m$, and let $d_i = d(y_2, v_i)$ for $i = 1, \dots, n$. We then have

$$z_q^T(x'(q, y_2)) = f(d_1^T, \dots, d_m^T, d_{m+1}, \dots, d_n). \quad (3.37)$$

For brevity let $\tilde{d}_i = \tilde{d}_{x'(q, y_2)}(v_i)$ for $i = r' + 1, \dots, m'$. Then we have

$$\begin{aligned} \bar{z}(x'(q, y_2)) = f(d_1, \dots, d_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_{r'}, d_{r+1}, \dots, d_m, \\ \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n). \end{aligned} \quad (3.38)$$

Since $d_i^T = d_i$ for $i = r + 1, \dots, m$, this becomes

$$\begin{aligned} \bar{z}(x'(q, y_2)) = f(d_1, \dots, d_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, d_{r+1}^T, \dots, d_m^T, \\ \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n) . \end{aligned} \quad (3.39)$$

Meanwhile we can write,

$$\begin{aligned} z_q^T(x'(q, y_1)) = f(d_1^T - \Delta, \dots, d_r^T - \Delta, d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, \\ d_{m+1}, \dots, d_n) , \end{aligned} \quad (3.40)$$

where Δ is the length of $A[y_1, y_2]$. Also since f is nondecreasing, $z_q^T(x'(q, y))$ is convex for $y \in T_q(x')$. Thus since x_q^* solves (3.33), we have

$$z_q^T(x'(q, y_1)) \geq z_q^T(x'(q, y_2)) . \quad (3.41)$$

This, (3.37) and (3.40) imply

$$\begin{aligned} f(d_1^T - \Delta, \dots, d_r^T - \Delta, d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, d_{m+1}, \dots, d_n) \\ \geq f(d_1^T, \dots, d_m^T, d_{m+1}, \dots, d_n) . \end{aligned} \quad (3.42)$$

It is clear that for any $y \in T_q(x')$, and any i , $i = 1, \dots, m$,

$$d(y, v_i) \leq d_q^T(y, v_i; x') . \quad (3.43)$$

We also observe that $\tilde{d}_i \leq d_i^T$ for $i = r' + 1, \dots, r$. For, v_i is in $P_q(x')$ but not in $P_q(x'(q, y_1))$, which means that as y moves along $L_q^T(y_1, x')$ toward y_2 , there is a point $y = y^*$ at which v_i (last) leaves $P_q(x'(q, y))$. Thus in a neighborhood of y^* , $d(y, v_i)$ is increasing as y

approaches y^* from x'_q . This means $d_q^T(y, v_i; x')$ is increasing as y approaches y^* , which implies that v_i is on the same side of y^* in $T_q(x')$ as x'_q is. Thus $d_q^T(y, v_i; x')$ continues to increase as y approaches y_2 , so that

$$d_q^T(y^*, v_i; x') \leq d_q^T(y_2, v_i; x') = d_i^T. \quad (3.44)$$

But since v_i leaves $P_q(x'(q, y))$ exactly at $y = y^*$, we must have $d(y^*, v_i) = \tilde{d}_{x'(q, y_2)}(v_i) = \tilde{d}_i$. Thus, (3.43) and (3.44) imply that $\tilde{d}_i \leq d_i^T$, $i = r' + 1, \dots, r$.

Now since by (3.43) $d_i \leq d_i^T$ for $i = 1, \dots, r'$, and since $\tilde{d}_i \leq d_i^T$ for $i = r' + 1, \dots, r$, the semiseparability of f and (4.38) allow us to infer,

$$\begin{aligned} & f(d_1 - \Delta, \dots, d_{r'} - \Delta, \tilde{d}_{r'+1} - \Delta, \dots, \tilde{d}_r - \Delta, \\ & \quad d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, \\ & \quad d_{m'+1}, \dots, d_n) \\ & \geq f(d_1, \dots, d_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, d_{r+1}^T, \dots, d_m^T, \\ & \quad \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n). \end{aligned} \quad (3.45)$$

Since $A[y_1, y_2]$ lies in a single treelike segment, we know that $d(y_1, v_i) = d(y_2, v_i) \pm \Delta$ for $i = 1, \dots, r'$. Thus we can write $d(y_1, v_i) = d_i - \Delta_i$, where $\Delta_i \leq \Delta$. This permits us to say,

$$\begin{aligned} \tilde{z}(x'(q, y_1)) &= f(d_1 - \Delta_1, \dots, d_{r'} - \Delta_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, \\ & \quad d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n). \end{aligned} \quad (3.46)$$

But since $\Delta_i \leq \Delta$ and f is nondecreasing, (3.45) implies,

$$\begin{aligned} & f(d_1 - \Delta_1, \dots, d_{r'} - \Delta_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, \\ & \quad \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n) \\ & \geq f(d_1, \dots, d_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, d_{r+1}^T, \dots, d_m^T, \\ & \quad \tilde{d}_{m+1}, \dots, \tilde{d}_{m'}, d_{m'+1}, \dots, d_n) . \end{aligned} \quad (3.47)$$

Now, (3.39), (3.41) and (3.47) imply (3.35), as desired.

It remains to show (3.36). We distinguish two cases.

Case I: f is not additively separable. In this case $L_q^T(y''; x') \subset E$, so that $P_q(x'(q, y_1)) \subset P_q(x')$. Thus $z(x'(q, y)) = \tilde{z}(x'(q, y))$ for all $y \in L_q^T(y''; x')$, and (3.36) follows from (3.35).

Case II: f is additively separable. Here we observe that since $v_i \in P_q(x'(q, y_2))$ for $i = m+1, \dots, m'$,

$$\begin{aligned} z(x'(q, y_2)) &= f(d_1, \dots, d_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, \\ & \quad d_{r+1}^T, \dots, d_m^T, d_{m+1}, \dots, d_{m'}, d_{m'+1}, \dots, d_n) . \end{aligned} \quad (3.48)$$

Also, by hypothesis $d(y, v_i)$ is increasing on $A[y_1, y_2]$ (as y moves from y_1 to y_2) for $i = m+1, \dots, m'$. Thus since $A[y_1, y_2]$ lies on a single treelike segment,

$$\begin{aligned} z(x'(q, y_1)) &= f(d_1 - \Delta_1, \dots, d_{r'} - \Delta_{r'}, \tilde{d}_{r'+1}, \dots, \tilde{d}_r, \\ & \quad d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, d_{m+1} + \Delta, \dots, d_{m'} + \Delta, \\ & \quad d_{m'+1}, \dots, d_n) . \end{aligned} \quad (3.49)$$

Since f is nondecreasing, (3.47) implies

$$\begin{aligned}
 & f(d_1 - \Delta_1, \dots, d_r - \Delta_r, \tilde{d}_{r+1}, \dots, \tilde{d}_r, d_{r+1}^T + \Delta, \dots, d_m^T + \Delta, \\
 & \quad \tilde{d}_{m+1} + \Delta, \dots, \tilde{d}_m + \Delta, d_{m+1}, \dots, d_n) \\
 & \geq f(d_1, \dots, d_r, \tilde{d}_{r+1}, \dots, \tilde{d}_r, d_{r+1}^T, \dots, d_m^T, \\
 & \quad \tilde{d}_{m+1}, \dots, \tilde{d}_m, d_{m+1}, \dots, d_n) . \tag{3.50}
 \end{aligned}$$

Because f is additively separable, (3.48), (3.49) and (3.50) imply (3.36). This completes the proof. $\square$

It is helpful to illustrate Theorem 3.1 with the example of the foregoing sections. Suppose facilities 1 and 2 are at locations $x' = (x'_1, x'_2)$ shown in Fig. 3-6, where the location of facility 2 is

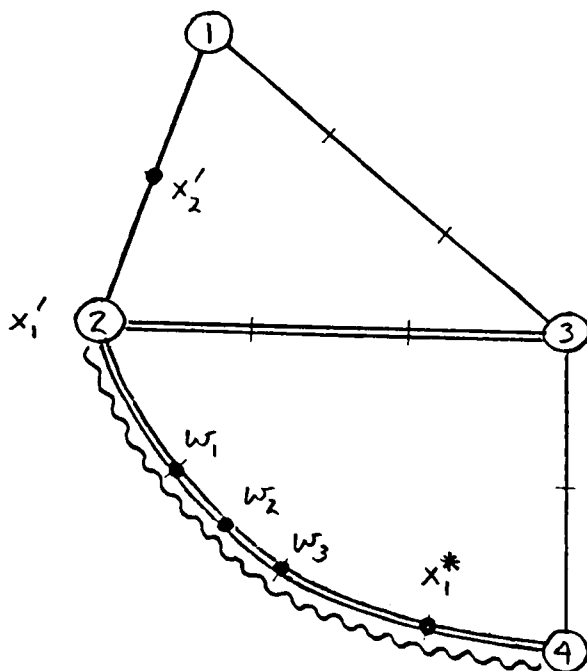


Fig. 3-6. Illustration of Theorem 3.1.

considered to be fixed in advance. Here, $q = 1$, $P_1(x') = \{2,3,4\}$, $P_2(x') = \{1\}$, and $T_1(x')$ is shown with double lines in the figure. The solution x_1^* of (3.33) is indicated and the wavy line is $L_q^T(x_1^*; x')$. Treelike section and partition boundaries along $L_q^T(x_1^*; x')$ are x_1' , w_1 , w_2 and w_3 . Figure 3-7 represents the cartesian product of arcs (2,4) and (1,2) with $B = [0,4] \times [0,2]$. Points representing x' , (x_1^*, x_2') , etc., are marked. Note that as one moves from x' to (x_1^*, x_2') , no new nodes are assigned to facility 1, although facility 1 loses nodes 2 and 3 at least temporarily. Thus $y' = y'' = x_1^*$ and $E = L_1^T(x_1^*; x')$. By Theorem 3.1 $z(x'(1,y)) = z((y,x_2'))$ should improve as y moves from x_1' to x_1^* , and indeed it improves from $z(x_1', x_2') = 110$ to $z(x_1^*, x_2') = 36$. Thus we can eliminate segments $A[x_1', w_1]$, $A[w_1, w_2]$ and $A[w_2, w_3]$ on $L_1^T(x_1^*; x')$ from consideration.

The extension of Theorem 3.1 to the general problem in which no facilities have fixed locations is suggested by this same example. Suppose facility 1 is located at $\tilde{x}_2$ rather than x_2' (Fig. 3-8), and let $y \in L_q^T(x_1^*; x')$. Since $\tilde{x}_2$ is in the same partition zone as x_2' , $T_q((x_1', \tilde{x}_2)) = T_q(x')$. Also since f is additively separable, it is not hard to see that x_1^* solves (3.33) when facility 2 is $\tilde{x}_2$ as well as when it is at x_2' . Thus since no new nodes are assigned to facility 1 as one moves from $(x_1', \tilde{x}_2)$ to $x_1^*, \tilde{x}_2$, we know from Theorem 3.2 that $z((y, \tilde{x}_2))$ is non-increasing as y moves from x_1' to x_1^* . Thus we can eliminate segments $A[x_1', \tilde{w}_1]$, $A[\tilde{w}_1, \tilde{w}_2]$ and $A[\tilde{w}_2, \tilde{w}_3]$ of $L_1^T(x_1^*; \tilde{x})$ (Fig. 3-8). Since the same reasoning applies to any $\tilde{x}_2$ in the same partition zone as x_2' , we can eliminate all treelike subregions wholly inside the double lines of Fig. 3-8 in which facility 1 is assigned no nodes other than those in $P_1(x')$.

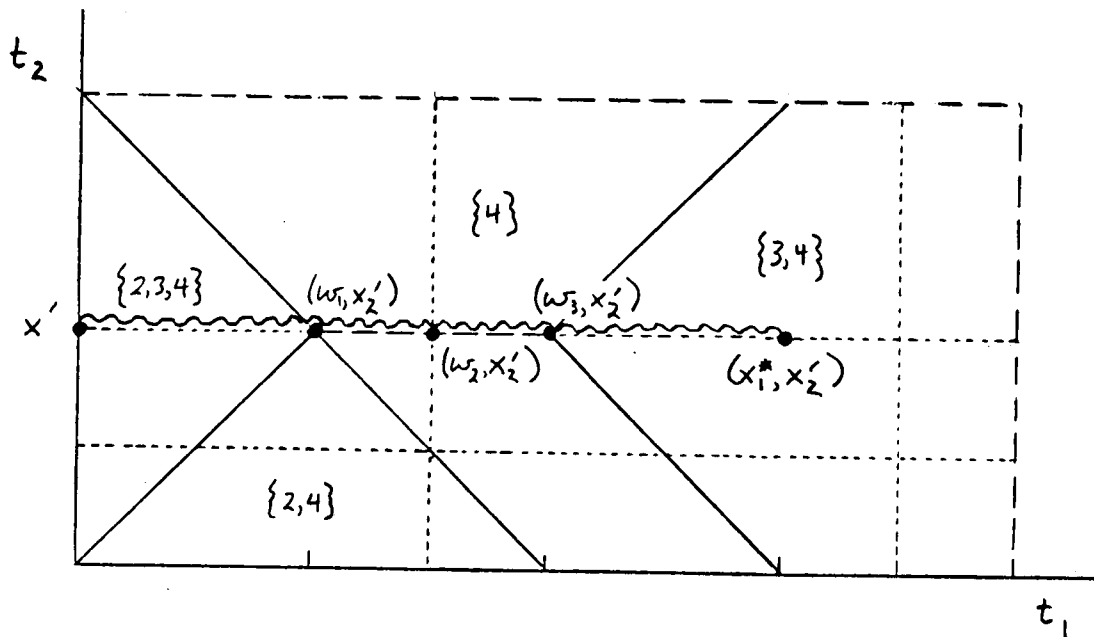


Fig. 3-7. Representation of the position of facilities 1 and 2 in the cartesian product of two arcs. Arc (2,4) corresponds to the horizontal axis, arc (1,2) to the vertical. Point labels are the names of points on the network represented by the points labeled.

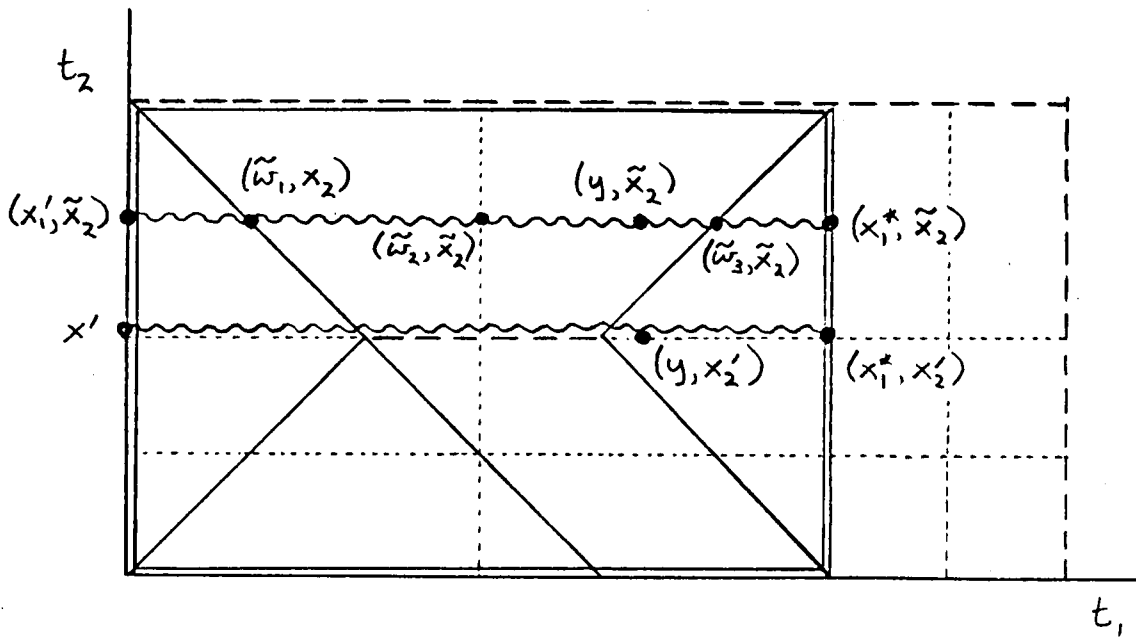


Fig. 3-8. Extension of Theorem 3.1 to the case in which no facilities have fixed locations.

Since the nine treelike subregions wholly inside the double lines satisfy this property, we can eliminate them all.

We make these ideas precise in Corollary 3.1 and the algorithms that follow.

Corollary 3.1. Let f in (3.1) be convex, nondecreasing and additively separable, and let x' and x_q^* be as in Theorem 3.2. Choose any $\tilde{x} \in G^P$ for which $P_q(\tilde{x}(q, x_q')) = P_q(x')$, and let $\tilde{E}$ be the set of points y in $L_q^T(x_q^*; x')$ such that $P_q(\tilde{x}(q, y)) \subseteq P_q(x')$. Denote by $\tilde{y}'$ the point of $\tilde{E}$ closest to x_q^* . Let $\tilde{y}''$ be the point in $L_q^T(x_q^*; x')$ closest to x_q^* such that for each node v_i , $d(\cdot, v_i)$ is decreasing (as one moves away from x_q') at all points $y \in L_q^T(x_q^*(x_q^*; x'))$ for which $v_i \in P_q(\tilde{x}(q, y)) - P_q(x')$. Then for any $y \in \tilde{E}$,

$$z(\tilde{x}(q, y)) \geq z(\tilde{x}(q, \tilde{y}')) , \quad (3.51)$$

and for any $y \in L_q^T(\tilde{y}''; x')$,

$$z(\tilde{x}(q, y)) \geq z(\tilde{x}(q, \tilde{y}'')) . \quad (3.52)$$

Proof. To apply Theorem 3.2, we must show that x_q^* solves (3.33) when x' is replaced by $\tilde{x}(q, x_q')$. But since x_q^* solves (3.33) as stated, we have that for any $y \in T_q(x')$, $z_q^T(x'(q, y)) \geq z_q^T(x'(q, x_q^*))$. This implies by (3.33),

$$\begin{aligned} & f(d_{qx'}^T(q, y)(v_1), \dots, d_{qx'}^T(q, y)(v_n)) \\ & \geq f(d_{qx'}^T(q, x_q^*)(v_1), \dots, d_{qx'}^T(q, x_q^*)(v_n)) , \end{aligned} \quad (3.53)$$

where by definition of $d_{qx'}^T(q, y)(v_i)$,

$$d_{qx'}^T(q, y)(v_i) = d_{qx'}^T(q, x_q^*)(v_i) = d_{x'}(v_i) \quad (3.54)$$

when $v_i \notin P_q(x')$. But we are given that $P_q(x') = P_q(\tilde{x}(q, x_q'))$, so that (3.54) holds when $v_i \notin P_q(\tilde{x}(q, x_q'))$. Also $T_q(x') = T_q(\tilde{x}(q, x_q'))$, so that when $v_i \in P_q(\tilde{x}(q, x_q'))$ we have that

$$\begin{aligned} d_{qx'}^T(q, y)(v_i) &= d_q^T(y, v_i; x') = d_q^T(y, v_i; \tilde{x}(q, x_q')) \\ &= d_{q\tilde{x}}^T(q, y)(v_i), \end{aligned} \quad (3.55)$$

and similarly

$$d_{qx'}^T(q, x_q^*)(v_i) = d_{q\tilde{x}}^T(q, x_q^*)(v_i). \quad (3.56)$$

Thus since f is additively separable, (3.54)-(3.56) allow us to infer from (3.53) that

$$\begin{aligned} &f(d_{q\tilde{x}}^T(q, y)(v_1), \dots, d_{q\tilde{x}}^T(q, y)(v_n)) \\ &\geq f(d_{q\tilde{x}}^T(q, x_q^*)(v_1), \dots, d_{q\tilde{x}}^T(q, x_q^*)(v_n)), \end{aligned}$$

for all $y \in T_q(\tilde{x}(q, v_q')) = T_q(x')$. This implies that $z_q^T(\tilde{x}(q, y)) \geq z_q^T(\tilde{x}(q, x_q^*))$ for all such y . In other words, x_q^* solves (3.33) when x' is replaced by $\tilde{x}(q, x_q')$. Given this, (3.51) and (3.52) follow from a direct application of Theorem 4.2.

□

We first present an algorithm that optimally locates one facility among $p - 1$ other fixed facilities. It is essentially an adaptation of the algorithm of Section 2.5 that exploits Theorem 3.2. As in the single-facility problem, one can exclude the facility from any arc by

considering the arc "infeasible" and passing over it in Step 1 of the algorithm. Also one can exclude the facility from an interval within an arc by inserting a node at either end of the interval and regarding the new arc between the nodes as infeasible.

Step 0. Let facilities 2, ..., p be fixed at locations $x'_2, \dots, x'_p$. Set the relative minimum counter K to zero, and set $z_{\min} = \infty$.

Step 1. Choose any uneliminated feasible arc $A_j = (u, w)$; if none remain, go to Step 11. For $i = 1, \dots, n$ compute $d(u, v_i)$, $d(w, v_i)$, $d(x'_q, v_i)$, $q = 2, \dots, p$. If a list L_j of the treelike subregions of A_j has already been made, go to Step 4. Otherwise compute $s_i = \frac{1}{2}[d(w, v_i) - d(u, v_i) + b(u, w)]$ for $i = 1, \dots, n$. If Technique 1 is not used, go to Step 3.

Step 2. Set $i = 0$, $\hat{p} = 1$ and try to eliminate arc (u, w) using Steps 2a-2f of Section 3.3. If (u, w) is eliminated, go to Step 1.

Step 3. Locate treelike subregions on A_j as follows. For $i = 1, \dots, n$ let

$$d_i = \min_{q=2, \dots, p} \{d(x'_q, v_i)\},$$

and

$$h_i = \begin{cases} d_i - d(u, v_i) & \text{if } d(u, v_i) \leq d(w, v_i) , \\ b(u, w) - d_i + d(w, v_i) & \text{otherwise .} \end{cases}$$

Set $s_1, \dots, s_n, h_1, \dots, h_n$ in nondecreasing order and let $h'_1, \dots, h'_r$ be the distinct members of the list. Let L_j be a list of the treelike subregions $[h_{\ell-1}, h_\ell]$, $\ell = 1, \dots, r$, on arc A_j .

Step 4. Pick a treelike subregion S in L_j ; if none remain, go to Step 1. Let x'_1 be any point in S and solve (3.33) in the following way. First determine whether the solution y^* of (3.33) lies on A_j . If not, go to Step 6.

Step 5. Set $\text{soln} = \text{true}$ and $y^0 = y^*$. If $y^* \in S$ or Technique 2 is not used, set $K = K + 1$, $x_K^* = y^*$, $z_K^* = z_1^T(x'(1, y^*))$, $z_{\min} = \min\{z_{\min}, z_K^*\}$, remove S from L_j and go to Step 4. Let $\hat{u} = x'_1$, and let $\hat{w}$ be the node of (u, w) that is closer to y^* . Let $j = j$, $\hat{y} = \hat{u}$ and go to Step 7.

Step 6. Set $\text{soln} = \text{false}$ and let y^0 be the node of A_j that minimizes $z_1^T(x'(1, y))$. If not using Technique 2, remove S from L_j and go to Step 4. Let $j = j$, $\hat{u} = x'_1$, $\hat{w} = y^0$, and $\hat{y} = \hat{u}$.

Step 7. If f is not additively separable or $\hat{y} \neq \hat{u}$, go to Step 8. As y moves from $\hat{u}$ to $\hat{w}$, find the point $\hat{y}$ at which $d(y, v_1)$ begins increasing for some $v_1 \in P_1(x'(1, y)) - P_1(x')$; let $\hat{y} = \hat{w}$ if there is no such point. Remove from L_j all treelike subregions between $\hat{u}$ and y^0 or $\hat{y}$, whichever is closer to $\hat{u}$.

Step 8. Remove from L_j any treelike subregions S between $\hat{u}$ and y^0 such that $P_1(x'(1, y)) \leq P_1(x')$ for all $y \in S$. If $\text{soln} = \text{true}$ go to Step 4.

Step 9. Let y enter the one arc $A_{j'} = (\hat{w}, w')$ for which $z_1^T(x'(1, y))$ becomes smaller than $z_1^T(x'(1, \hat{w}))$. Let $j = j'$, $(\hat{u}, \hat{w}) = (\hat{w}, w')$. Compute $\hat{s}_1, \dots, \hat{s}_n$ for arc A_j as they were computed for A_j in Step 1. If using Technique 1 perform Step 2 with $(u, w) = (\hat{u}, \hat{w})$, and branch to Step 1 if $(\hat{u}, \hat{w})$ is eliminated. Compute $\hat{h}_1, \dots, \hat{h}_p$ for arc A_j as in Step 3, and create a list L_j of treelike subregions of A_j .

Step 10. Determine whether the solution of (3.33) lies on A_j . If so, set $\text{soln} = \text{true}$ and $y^0 = y^*$. If not, set $\text{soln} = \text{false}$ and $y^0 = \hat{w}$. Go to Step 7.

Step 11. Any point x_K^* such that $z_K^* = z_{\min}$ solves (3.1).

□

We now present a second algorithm that solves the p -facility problem with the help of Corollary 3.1. It enumerates treelike rectangles, and for each rectangle enumerates possible assignments of facilities to nodes, thereby covering all treelike subregions. Although the path from x_q' to the optimal point x_q^* on the associated tree $T_q(x')$ may extend beyond the arc on which x_q' lies, the algorithm below follows the path only to the end of the arc. It also exploits (3.51) of Corollary 3.1 but not (3.52). These simplifications make the algorithm much more straightforward, probably without a large reduction in efficiency on the small networks for which it is intended.

Step 0. Let $A(1), \dots, A(m_0)$ be the feasible arcs, and let facilities $\hat{p} + 1, \dots, p$ be fixed at locations $x_{\hat{p}+1}', \dots, x_p'$ in advance. Set the relative minimum counter K to zero, and set $z_{\min} = \infty$. Calculate $d(x_q', v_i)$ for $i = 1, \dots, n$ and $q = \hat{p} + 1, \dots, p$.

Step 1. Choose a $\hat{p}$ -tuple of arcs $(A(i_1), \dots, A(i_{\hat{p}})) = ((u_1, w_1), \dots, (u_{\hat{p}}, w_{\hat{p}}))$ not already chosen, where $1 \leq i_q \leq m_0$ and $i_q \leq i_{q-1}$ for $q = 1, \dots, \hat{p}$. If none remain, go to Step 12. Let $b_q = b(u_q, w_q)$ for $q = 1, \dots, \hat{p}$.

Step 2. Calculate $d(u_q, v_i)$, $d(w_q, v_i)$ and $s_{qi} = \frac{1}{2}[d(w_q, v_i) - d(u_q, v_i) + b_q]$ for $i = 1, \dots, n$, $q = 1, \dots, \hat{p}$. If Technique 1 is not used, go to Step 4.

Step 3. Set $i = 0$ and try to eliminate p -tuple $((u_1, w_1), \dots, (u_p, w_p))$ using Steps 2a-2f of Section 3.3. If the p -tuple is eliminated, go to Step 1.

Step 4. For each $q \in \{1, \dots, \hat{p}\}$ arrange the s_{qi} 's in ascending order, and let $s'(q, 0), \dots, s'(q, m_q)$ be the distinct members of the ordered list. Let L be a list of the treelike rectangles of the form $[s'(1, k_1 - 1), s'(1, k_1)] \times \dots \times [s'(p, k_{\hat{p}} - 1), s'(p, k_{\hat{p}})]$ for all $k_q, 1 \leq k_q \leq m_q, q = 1, \dots, \hat{p}$.

Step 5. Choose a treelike rectangle $R = [s'(1, k_1 - 1), s'(1, k_1)] \times \dots \times [s'(p, k_{\hat{p}} - 1), s'(p, k_{\hat{p}})]$ from L ; if none remain, go to Step 1. For $i = 1, \dots, n$ and $q = 1, \dots, \hat{p}$, calculate $d(v_i, x_q(s'(q, k_q))) = \min \{d(v_i, u_q) + s'(q, k_q), d(v_i, w_q) + b_q - s'(q, k_q)\}$. For $q = 1, \dots, \hat{p}$ define $y_{qki}, h_{qki}(t), s'_{qki}, i = 1, \dots, n$, and c_{qk} as in (3.3)-(3.6). For $q = \hat{p} + 1, \dots, p$ define $y_{qki} = x'_q, h_{qki}(t) = 0, s'_{qki} = 0, i = 1, \dots, n$, and $c_{qk} = 0$.

Step 6. Let V^* be the set of nodes that satisfy (3.7) for no facility q and that fail to satisfy (3.8) for at least two facilities $q, q \in \{1, \dots, p\}$. Let $C(a) = \emptyset$ for all possible assignments a of facilities to nodes. $C(a)$ will contain constraints that delimit regions of R that, for assignment a , have been ruled out by virtue of Corollary 4.1 as possibly containing an optimal location.

Step 7. Choose a (possible) treelike subregion by making an assignment $a = (a(1), \dots, a(n))$ of facilities to nodes that is consistent with (3.9) and (3.10) and has not already been made. If none remain, remove R from L and go to Step 5.

Step 8. See whether the chosen treelike subregion is infeasible or has been ruled out by virtue of Corollary 3.1 by solving the linear program:

$$\begin{aligned}
 & \min_{t_1, \dots, t_{\hat{p}}} \quad 1 & (3.57) \\
 \text{s.t.} \quad & s'(q, k_q - 1) \leq t_q \leq s'(q, k_q), \quad q = 1, \dots, \hat{p}, \\
 & d(v_i, y_{a(i)ki}) + h_{a(i)ki}(t_{a(i)}) \leq d(v_i, y_{qki}) \\
 & \quad + h_{qki}(t_q), \quad q \in 1, \dots, p, \text{ all } i \in V^*, \\
 & \text{plus all constraints in } C(a).
 \end{aligned}$$

If (3.57) has no feasible solution, assignment a can be ruled out; go to Step 7. Otherwise, let $t' = (t'_1, \dots, t'_{\hat{p}})$ be a feasible solution of (3.57).

Step 9. Choose any $q \in \{1, \dots, \hat{p}\}$ and solve (3.33) on the current p -tuple of arcs. Do this by solving the convex problem

$$\begin{aligned}
 & \min_{t_q} f(d^T(t_q, v_1), \dots, d^T(t_q, v_n)) & (3.58) \\
 \text{s.t.} \quad & 0 \leq t_q \leq b_q,
 \end{aligned}$$

where

$$d^T(t_q, v_i) = \begin{cases} d(v_i, y_{qki}) + h_{qki}(t_q) & \text{if } q = a(i), \\ d(v_i, y_{a(i)ki}) + h_{a(i)ki}(t'_{a(i)}) & \text{otherwise.} \end{cases}$$

Let t_q^* be a solution.

Step 10. For each assignment $(a'(1), \dots, a'(n))$ of facilities to nodes that has the property that $a'(i) = q$ only if $a(i) = q$, $i = 1, \dots, n$, add the constraints

$$t_q \leq t_q^* \text{ and } t_q \geq t_q', \text{ if } t_q^* \leq t_q',$$

or

$$t_q \geq t_q^* \text{ and } t_q \leq t_q', \text{ if } t_q^* > t_q',$$

to $C(a')$.

Step 11. If t_q^* is in the current treelike segment of arc (u_q, w_q) , rectangle R cannot be eliminated, and a location problem must be solved on R . Thus if $t_q^* \in [s'(q, k_q - 1), s'(q, k_q)]$, solve the convex problem

$$\begin{aligned} \min_{t_1, \dots, t_{\hat{p}}} \quad & f(d(v_1, y_{a(1)k_1}) + h_{a(1)k_1}(t_{a(1)}), \dots, \\ & d(v_n, y_{a(n)k_n}) + h_{a(n)k_n}(t_{a(n)})), \end{aligned} \quad (3.59)$$

$$\text{s.t.} \quad s'(q, k_q - 1) \leq t_q \leq s'(a, k_q), \quad q = 1, \dots, \hat{p},$$

$$\begin{aligned} d(v_i, y_{a(i)k_i}) + h_{a(i)k_i}(t_{q(i)}) &\leq d(v_i, y_{qk_i}) + h_{qk_i}(t_q), \\ q &= 1, \dots, p, \text{ all } i \in V^*, \end{aligned}$$

plus all constraints in $C(a)$.

(The constraints in $C(a)$ can be omitted if their presence makes (3.59) more difficult to solve.) If (3.59) has a feasible solution, let $t^* = (t_1^*, \dots, t_p^*)$ be an optimal solution, set $K = K + 1$, $x_K^* = x(t^*)$, $z_K^* = z(x_K^*)$, and $z_{\min} = \min \{z_{\min}, z_K^*\}$. Go to Step 7.

Step 12. Any x_K^* such that $z_K^* = z_{\min}$ solves (3.1).

CHAPTER 4

A METRICAL THEORY OF NETWORKS

We have remarked the interesting fact that a network has a convex metric if and only if it is a tree (Corollary 1.1). One is inclined to ask, what is the essential characteristic of networks that is responsible for this fact?

We have also observed that any network is composed of pieces, "treelike segments," on which the metric is convex (Section 2.2). This permits us to solve a location problem on a network by solving convex location problems on some or all of the treelike components of the network, provided the cost function is convex. One wonders whether there are more general structures than networks that decompose into "treelike" pieces on which the metric is convex and that allow a similar solution of location problems on them.

We show in this chapter that both questions can be answered at once. We first prove that any metric space having a rather general property we call "treelikeness" necessarily has a convex metric, suitably defined for the more general setting. We then define a "graphic space" to be a metric space that is in a certain sense decomposable into treelike pieces. We prove that a graphic space is treelike if and only if it has a convex metric, whereas a nongraphic space with a convex metric need not be treelike. In other words, decomposability into treelike pieces appears to be just what is needed to ensure the equivalence of treelikeness and convexity of metric. We go on to show

that a graphic space having certain finiteness properties is made up of treelike components on which the metric is convex, and that location problems can indeed be solved on such graphic spaces in a way analogous to their solution on networks.

Our main conclusion, then, is that two important metrical properties of networks—decomposability into treelike pieces and the equivalence of treelikeness and convexity of metric—are intimately related. Insofar as these properties are definitive of networks, conceived as metrical structures, the results to follow can be seen as comprising a metrical theory of networks.

We begin in Section 4.1 by defining convexity and treelikeness in an abstract setting and showing that all treelike spaces have a convex metric. We also show that a line segment connecting two points of a treelike space need not be pathlike unless the two points are nodes. In Section 4.2 we define a graphic space and prove through a series of lemmas that a graphic space is treelike if and only if it has a convex metric. Section 4.3 sets out sufficient conditions for the decomposability of a graphic space into finitely many treelike pieces, and Section 4.4 investigates the extent to which graphic spaces are modeled by networks or subsets of networks. The last three sections deal with the solution of location problems on graphic spaces. Section 4.5 explains when, in general, a local optimum on a metric space is a global optimum. Section 4.6 proves that the same conditions that ensure finite decomposability in Section 4.3 ensure decomposability into finitely many pieces on which the metric is convex. Thus one can solve location problems on

a broad class of graphic spaces by solving finitely many convex subproblems. This and the equivalence theorem of Section 4.2 are the main results of the chapter. The final section proves that the segment elimination theorem of Chapter 2, Theorem 2.2, holds for graphic spaces in general.

4.1 Convexity and Treelike Spaces

Recall that if $d: S \times S \rightarrow \mathbb{R}$, (S, d) is a metric space if for all $x, y, z \in S$, $d(x, y) = d(y, x)$, $d(x, x) = 0$, $d(x, y) > 0$ if $x \neq y$, and $d(x, y) \leq d(x, z) + d(z, y)$. A natural definition of betweenness, introduced by Pasch [1882], is that $z \neq x, y$ is between x and y if $d(x, y) = d(x, z) + d(z, y)$. As in the case of networks, we say $z \in L_\lambda(x, y)$ if $d(x, z) = \lambda d(x, y)$, $\lambda \in (0, 1)$, and z is between x and y . We also set $L_0(x, y) = \{x\}$, $L_1(x, y) = \{y\}$. Again the line segment $L(x, y)$ connecting x and y is the set containing x, y and the points between them, so that $L(x, y)$ is the union of $L_\lambda(x, y)$ over all $\lambda \in [0, 1]$.

If $L(z_0, \dots, z_k)$ denotes $L(z_0, z_1) \cup \dots \cup L(z_{k-1}, z_k)$, we say that $L(z_0, \dots, z_k)$ is a chain (in particular, a k-chain) connecting z_0 and z_k if $d(z_0, z_k) = d(z_0, z_1) + \dots + d(z_{k-1}, z_k)$. It will be convenient to adopt the convention that $C(y_0, \dots, y_m)$ denotes any chain of the form $L(z_0, \dots, z_k)$, where $y_0 = z_0, y_m = z_k$, and $\{y_1, \dots, y_{m-1}\} \subseteq \{z_1, \dots, z_{k-1}\}$.

As on networks, let $D \subseteq S$ be a convex set if $L(x, y) \subseteq D$ for all $x, y \in D$. Then function $f: D \rightarrow \mathbb{R}$ is convex on a convex set D if for all $\lambda \in [0, 1]$ and all $z \in L_\lambda(x, y)$, $f(z) \leq (1 - \lambda)f(x) + \lambda f(y)$.

We should note that other definitions of a convex set in an abstract setting have been advanced. Menger [1928] requires that a convex set in a metric space contain a point between every pair of its points. This makes all convex sets dense and enables Menger to prove that any two distinct points of a convex set are joined by a line segment that is homeomorphic to the unit interval. Such a definition is too strict for our purposes, since we wish to deal with spaces that may contain finitely many points. Jamison [1979, 1981, 1982] has more generally characterized convex sets as members of an "alignment," of which the convex subsets of a metric space, as we define them, form a special case. Another interesting special case is that in which a subgraph containing all "chordless paths" connecting any pair of its vertices is considered a convex subset of the containing graph. Farber and Jamison [1983] and Shier [1984] have investigated the consequences of testing a function for convexity along chordless paths rather than along line segments (as above), with interesting results. But since these results seem to apply only to functions defined exclusively on vertices, we feel justified in using our more restrictive type of convexity, at least for the time being.

If (S,d) is a metric space and $V \subseteq S$ is a set V of points arbitrarily distinguished as nodes, we refer to (S,d,V) as a metric space with nodes. The major role of nodes is to serve as the points with respect to which treelikeness is defined. We first define a point u_v in a chain $C(x,y) \subseteq S$ to be a collection point of $C(x,y)$ with respect to $v \in C(x,y)$

if any $z \in C(x,y)$ satisfies $d(z,v) = d(z,u_v) + d(u_v,v)$. Then $C(x,y)$ is a treelike chain if it contains a collection point u_v with respect to each $v \in V$ (Fig. 4-1). Finally, (S,d,V) is a treelike space if every 2-chain in S is a treelike chain, a definition we will justify shortly.

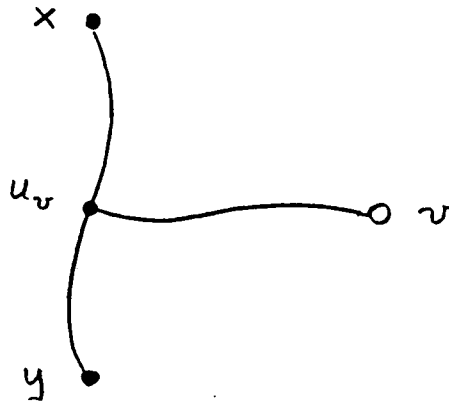


Fig. 4-1. Illustration of a collection point. Here and elsewhere, a black dot represents any point of a metric space with nodes, whereas an open dot represents a node. An arc connecting two points x,y generally represents the points (if any) lying between x and y . Naturally these points need not form an arc in the ordinary sense.

As in Section 1.3, we say that d is a convex metric if $d(\cdot, x)$ is convex on S for every $x \in V$ (not necessarily every $x \in S$). We do this partly because when solving location problems, we measure distances only from facilities to nodes. Also we note below that $d(\cdot, x)$ need not be convex in a treelike space for all $x \in S$. But we can prove the following.

Theorem 4.1. If (S,d,V) is treelike then d is a convex metric.

Proof. Pick any $x,y \in S$, $v \in V$. We wish to show that for any $\lambda \in [0,1]$ and any $z \in L_\lambda(x,y)$,

$$d(z,v) \leq (1 - \lambda)d(x,v) + \lambda d(y,v) . \quad (4.1)$$

Since (S,d,V) is treelike there is a $u_v \in L(x,z,y)$ such that

$$d(z,v) = d(z,u_v) + d(u_v,v) . \quad (4.2)$$

In particular (Fig. 4-2),

$$d(x,v) = d(x,u_v) + d(u_v,v) , \quad (4.3)$$

$$d(y,v) = d(y,u_v) + d(u_v,v) . \quad (4.4)$$

From (4.2)-(4.4), it suffices to show,

$$d(z,u_v) \leq (1 - \lambda)d(x,u_v) + \lambda d(y,u_v) . \quad (4.5)$$

Since $u_v \in L(x,z,y)$, we have $u_v \in L(z,y)$ or $u_v \in L(x,z)$. Therefore,

$$d(x,u_v) = d(x,z) + d(z,u_v) \quad (4.6)$$

$$\text{or } d(y,u_v) = d(y,z) + d(z,u_v) . \quad (4.7)$$

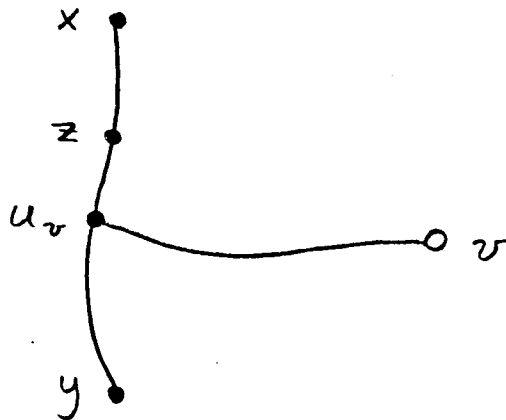


Fig. 4-2. Illustration of Theorem 4.1.

First suppose (4.6) holds. Substitution into (4.5) yields

$$\lambda[d(z, u_v) - d(y, u_v)] \leq (1 - \lambda)d(x, z) . \quad (4.8)$$

But since $d(z, u_v) - d(y, u_v) \leq d(z, u_v) + d(y, u_v) = d(z, y)$, (8) is implied by $\lambda d(z, y) = (1 - \lambda)d(x, z)$, which holds because $d(z, y) = (1 - \lambda)d(x, y)$ and $d(x, z) = \lambda d(x, y)$. Similarly, (4.7) implies (4.5). $\square$

We should note that it is not enough to ensure convexity of metric that every line segment be a treelike chain. The space of Fig. 4-3, for instance, has $V = S = \{x, y, w, z\}$, and every line segment is treelike. (For $L(x, z)$, $u_x = u_w = x$, $u_z = u_y = y$, and similarly for $L(x, w)$, $L(z, y)$ and $L(w, y)$. Also $L(x, y) = L(z, w) = S$, so that $L(x, y)$ and $L(z, w)$ are trivially treelike.) But $d(\cdot, w)$ is not convex on $L(x, y)$. We can observe, however, that neither of the 2-chains $L(x, z, y)$ and $L(x, w, y)$ is a treelike chain, as required for a treelike space.

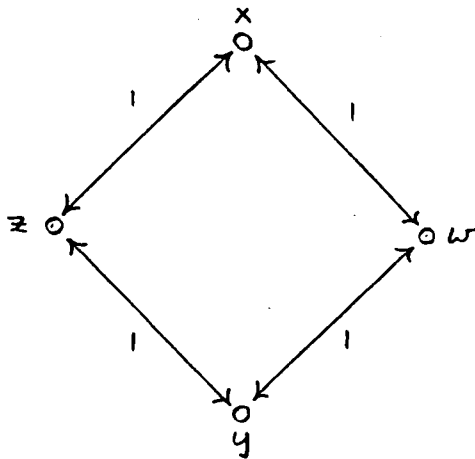


Fig. 4-3. A space in which every line segment is treelike but the metric is not convex. The arrows do not represent points of the space but only mark distances between points.

We should also note that $d(\cdot, x)$ need not be convex for all points x in a treelike space. If we let $V = \phi$ in the example of Fig. 4-3, the space is trivially treelike, but $d(\cdot, w)$ is again not convex. We therefore require of a convex metric d only that $d(\cdot, v)$ be convex for all $v \in V$.

Although treelikeness is sufficient for convexity of the metric, the converse is not true. Consider the space of Fig. 4-4, for which $S = V = \{x, y, z, w\}$. We see that $d(\cdot, v)$ is convex for any node v , but neither $L(x, z, y)$ nor $L(x, w, y)$ is a treelike chain. We can observe here that neither of these chains is a chain of treelike pieces, since neither $L(x, z)$ nor $L(z, y)$, for instance, is treelike. This suggests the definition of a graphic space in the next section.

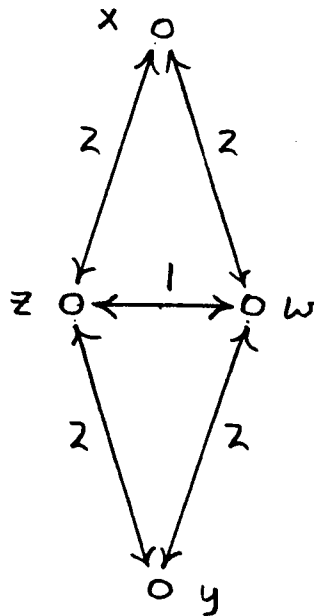


Fig. 4-4. A nontreelike space in which the metric is convex.

We conclude this section with a discussion of paths. One would expect that a treelike space would contain only one "path," in some sense, between any 2 points. Let us say that a line segment $L(x,y)$ is a path if for any $z_0, z_1, \dots, z_k \in L(x,y)$, with $z_0 = x$, $z_k = y$, and $d(z_0, z_i) < d(z_0, z_{i+1})$ for $i = 0, \dots, k-1$, the set $L(z_0, \dots, z_k)$ is a chain. We can observe that not all line segments in a treelike space need be paths. The space of Fig. 4-5 has $S = \{x, y, z_1, z_2\}$ and $V = \{x\}$. It is clearly treelike, but $L(x, z_1, z_2, y)$ is not a chain. We can, however, show the following.

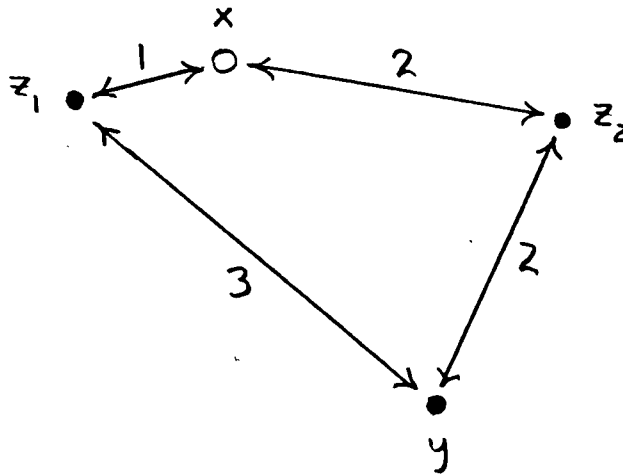


Fig. 4-5. A treelike space in which not all line segments are paths.

Theorem 4.2. Every line segment connecting 2 nodes in a treelike space is a path.

Proof. Suppose to the contrary that some line segment $L(x,y)$ is not a path. Then for some $z_1, \dots, z_{k-1} \in L(x,y)$, with $d(x, z_i) < d(x, z_{i+1})$ for $i = 1, \dots, k-1$ and $z_k = y$, $L(x, z_1, \dots, z_{k-1}, y)$ is not a chain. Thus $d(x, y) < d(x, z_1) + \dots + d(z_{k-1}, y)$. Let j be the smallest integer such that $d(x, z_j) < d(x, z_1) + \dots + d(z_{j-1}, z_j)$. Clearly, j exists and $j > 1$. Note that $L(x, z_{j-1}, z_j, y)$ is not a chain, since $d(x, y) \leq d(x, z_j) + d(z_j, y) < d(x, z_1) + \dots + d(z_{j-1}, z_j) + d(z_j, y) = d(x, z_{j-1}) + d(z_{j-1}, z_j) + d(z_j, y)$. Since the space is treelike, $L(z_{j-1}, z_j)$ contains collection points u_x and u_y with respect to x and y (Fig. 4-6). We consider three cases.

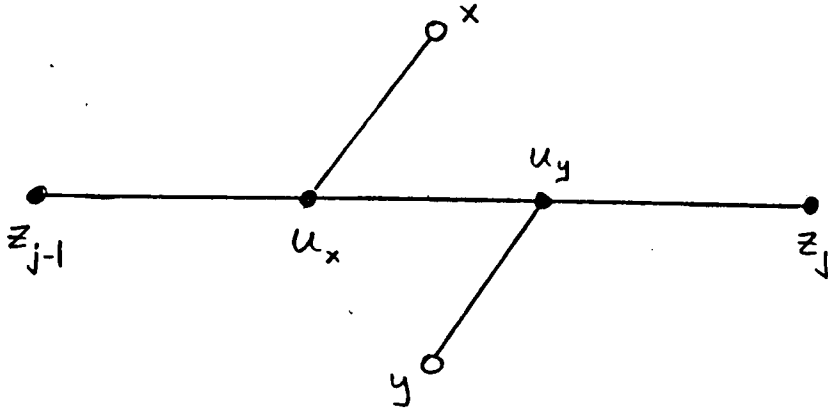


Fig. 4-6. Illustration of Theorem 4.2.

Case I. $u_x = z_{j-1}$ and $u_y = z_j$. This implies $d(x, z_j) = d(x, z_{j-1}) + d(z_{j-1}, z_j)$ and, since $z_j \in L(x, y)$, $d(x, y) = d(x, z_j) + d(z_j, y)$. Thus $d(x, y) = d(x, z_{j-1}) + d(z_{j-1}, z_j) + d(z_j, y)$, so that $L(x, z_{j-1}, z_j, y)$ is a chain, a contradiction.

Case II. $u_x = z_j$ and $u_y = z_{j-1}$. Since $d(x, z_{j-1}) < d(x, z_j)$, this implies $d(x, z_j) + d(z_j, z_{j-1}) < d(x, z_j)$, which is impossible.

Case III. At least one of u_x and u_y is interior to $L(z_{j-1}, z_j)$. Take the former case, so that $u_x \neq z_{j-1}, z_j$. Since $d(x, z_{j-1}) < d(x, z_j)$, we observe that $u_y \neq z_{j-1}$. Then we have $d(x, z_{j-1}) + d(z_{j-1}, y) = d(x, u_x) + d(u_x, z_{j-1}) + d(u_y, z_{j-1}) + d(u_y, y) > d(x, u_x) + d(u_x, u_y) + d(u_y, y) \geq d(x, y)$, which violates the assumption that $z_{j-1} \in L(x, y)$. The argument is similar if $u_y \neq z_{j-1}, z_j$.

Since all three cases lead to a contradiction, the line segment $L(x, y)$ must be a path.

□

4.2 Convexity and Graphic Spaces

In this section we define a graphic space and show that a graphic space is treelike if and only if its metric is convex.

We noted in the previous section that convexity of metric may imply treelikeness when any 2-chain is (or contains) a chain of treelike pieces. But another condition is needed. Consider the space of Fig. 4-7, for which $V = \{v\}$. The metric is clearly convex, and every 2-chain consists of one or two treelike pieces. But the space is not treelike because the 2-chain $L(x, x, y) = L(x, y)$ is not treelike.

The missing condition is analogous to the requirement that intersections of three or more arcs in a network be regarded as nodes. We first say that $z \in C(x, y)$ is an exit point of chain $C(x, y)$ if z is a

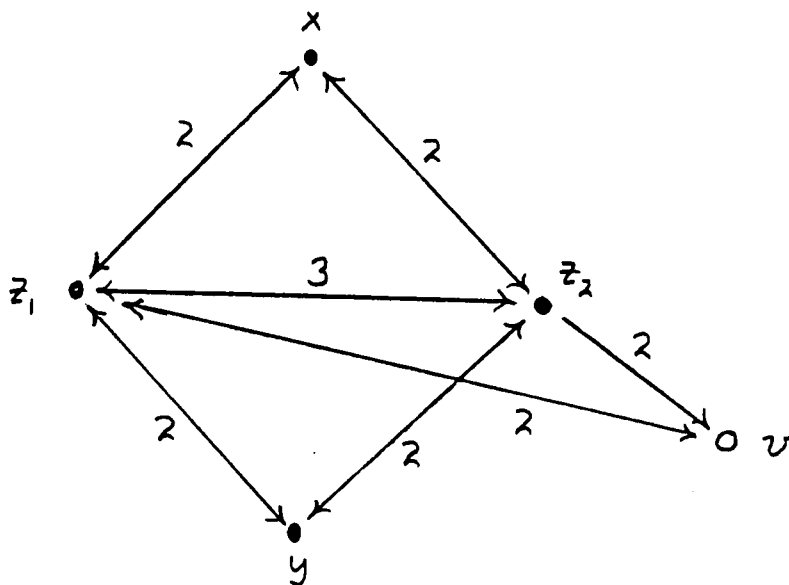


Fig. 4-7. A nontreelike space with convex metric and decomposable into treelike pieces.

collection point u_v of $C(x,y)$ with respect to some $v \notin C(x,y)$. Also $z \in C(x,y)$ is interior to $C(x,y)$ if $z \neq x,y$. If (S,d,V) is a metric space with nodes, $z \in S$ is an interior exit point of (S,d,V) if it is an interior exit point of some chain in S . Finally, a metric space with nodes (S,d,V) is a graphic space if it satisfies two conditions: (a) every 2-chain $L(x,z,y)$ in S contains a chain $C(x,y)$ of finitely many treelike chains, and (b) V contains all the interior exit points of (S,d,V) . We note that the space of Fig. 4-7 is not a graphic space because interior exit points z_1 and z_2 are not nodes. If they were, the metric would no longer be convex.

To prove the equivalence of treelikeness and convexity of metric in graphic spaces, we must first establish some properties of treelike chains. The following is basic.

Lemma 4.1. If line segment $L(x,y)$ contains a treelike chain $C(x,y)$, then $C(x,y)$ contains all the nodes of $L(x,y)$.

Proof. Let z be any node in $L(x,y)$. Since $C(x,y)$ is treelike there is a collection point $u_z \in C(x,y)$ with respect to z (Fig. 4-8). Thus,

$$\begin{aligned} d(x,z) &= d(x,u_z) + d(u_z,z) \\ d(y,z) &= d(y,u_z) + d(u_z,z) \end{aligned} \quad (4.9)$$

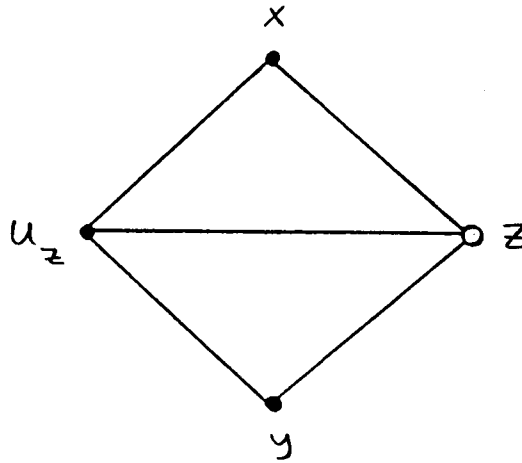


Fig. 4-8. Illustration of Lemma 4.1.

But since $u_z, z \in L(x,y)$, we have

$$d(x,u_z) + d(u_z,y) = d(x,z) + d(z,y) . \quad (4.10)$$

Substituting (4.9) into (4.10), we get $d(u_z,z) = 0$, or $u_z = z$. Thus $z \in C(x,y)$.

□

It will be useful to show, in the following two lemmas, that any chain of the form $C(x,z,y)$ in a graphic space contains a chain of finitely many treelike segments containing z .

Lemma 4.2. If $C(x,z) \equiv L(x,v_1, \dots, v_k, z)$ and $C(z,y) \equiv L(z,w_1, \dots, w_j, y)$ are chains and $z \in L(x,y)$, then $C(x,z,y) \equiv C(x,z)UC(z,y)$ is a chain.

Proof. We first show

$$L(x,z) \subseteq L(x,y) \quad \text{and} \quad L(z,y) \subseteq L(x,y). \quad (4.11)$$

Let $w \in L(x,z)$. Then $d(x,z) = d(x,w) + d(w,z)$, which implies $d(x,y) = d(x,w) + d(w,z) + d(z,y) \geq d(x,w) + d(w,y) \geq d(x,y)$. This implies $d(x,w) + d(w,y) = d(x,y)$, or $w \in L(x,y)$ and $L(x,z) \subseteq L(x,y)$. Similarly, $L(z,y) \subseteq L(x,y)$. From (4.11) we have $C(x,z,y) \subseteq L(x,y)$. Also since $z \in L(x,y)$, we have $d(x,v_1) + \dots + d(v_k, z) + d(z, w_1) + \dots + d(w_j, y) = d(x,z) + d(z,y) = d(x,y)$, so that $C(x,z,y)$ is a chain.

□

Lemma 4.3. Every 2-chain $L(x,z,y)$ in a graphic space contains a chain $C(x,z,y)$ of finitely many treelike chains.

Proof. Consider any 2-chain $L(x,z,y)$. Since the space is graphic, $L(x,z)$ contains a chain $C(x,z)$ and $L(z,y)$ a chain $C(z,y)$ of finitely many treelike chains. Since $L(x,z,y)$ is a chain, $z \in L(x,y)$, and by Lemma 4.2 $C(x,z,y) \equiv C(x,z)UC(z,y)$ is a chain. Obviously $C(x,z,y) \subseteq L(x,z,y)$ and $z \in C(x,z,y)$, as desired.

□

Two more simple lemmas about chains are needed.

Lemma 4.4. If $L(z_0, \dots, z_k)$ is a chain, then $L(z_0, \dots, z_j)$ is a chain for $j = 0, \dots, k$.

Proof. By the definition of a chain and the triangle inequality,

$$\begin{aligned} d(z_0, z_k) &= d(z_0, z_1) + \dots + d(z_{k-1}, z_k) \\ &\geq d(z_0, z_j) + d(z_j, z_{j+1}) + \dots + d(z_{k-1}, z_k) \geq d(z_0, z_k) \end{aligned}$$

which implies $d(z_0, z_j) = d(z_0, z_1) + \dots + d(z_{j-1}, z_j)$. Thus $z_1, \dots, z_{j-1} \in L(z_0, z_j)$ and $L(z_0, \dots, z_j)$ is a chain.

□

Lemma 4.5. If $L(x, z, y)$ is a chain and $u \in L(x, z)$, $v \in L(z, y)$, then $z \in L(u, v)$ (Fig. 4-9).

Proof. By definition of a chain and the triangle inequality,
 $d(x, y) = d(x, u) + d(u, z) + d(z, v) + d(v, y) \geq d(x, u) + d(u, v) + d(v, y) \geq d(x, y)$, which implies $d(u, z) + d(z, v) = d(u, v)$, or $z \in L(u, v)$

□

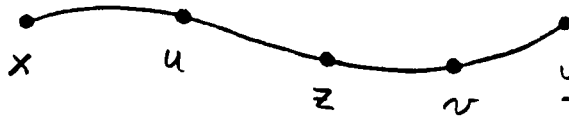


Fig. 4-9. Illustration of Lemma 4.5.

It is convenient to break the equivalence proof of convexity and treelikeness down into lemmas.

Lemma 4.6. If (S,d,V) has a convex metric, then any chain $C(x,w,y)$ of 2 treelike chains $C(x,w)$ and $C(w,y)$ is itself a treelike chain.

Proof. Suppose to the contrary that $C(x,w,y)$ is not a treelike chain. Let $v \in V$ be a node with respect to which $C(x,w,y)$ has no collection point. Let u_1, u_2 be collection points of treelike chains $C(x,w)$, $C(w,y)$ with respect to v (Fig. 4-10). Note first that we must have $u_1 \neq w$. For if $u_1 = w$, then for any $z \in C(x,w)$, $d(z,v) = d(z,w) + d(w,v) = d(z,w) + d(w,u_2) + d(u_2,v) = d(z,u_2) + d(u_2,v)$. Thus u_2 can serve as a collection point of $C(x,w)$ and hence $C(x,w,y)$ with respect to v , which contradicts the hypothesis. Similarly, $u_2 \neq w$.

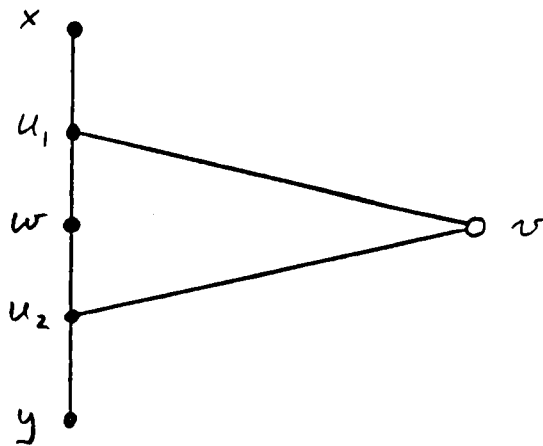


Fig. 4-10. Illustration of Lemma 4.6.

We have shown $u_1, u_2 \neq w$. By Lemma 4.5, $w \in L(u_1, u_2)$. Since $u_1 \neq u_2$, $w \in L_\lambda(u_1, u_2)$ where $\lambda \in (0, 1)$. To show d is not convex it suffices to show

$$d(w, v) > (1 - \lambda)d(u_1, v) + \lambda d(u_2, v) . \quad (4.12)$$

But we have

$$d(w, v) = d(w, u_1) + d(u_1, v) \quad (4.13)$$

and

$$d(w, u_1) + d(u_1, v) = d(w, u_2) + d(u_2, v) . \quad (4.14)$$

From (4.14)

$$d(u_2, v) = d(u_1, v) + (2\lambda - 1)d(u_1, u_2) . \quad (4.15)$$

Substitution of (4.13) and (4.15) into (4.12) yields

$$\lambda d(u_1, u_2) > \lambda(2\lambda - 1)d(u_1, u_2),$$

which holds because $\lambda \in (0, 1)$ and $d(u_1, u_2) > 0$. Thus we have (4.12), as desired.

□

Lemma 4.7. If all chains of 2 treelike chains in (S, d, V) are treelike, then any chain $C(x, y)$ of finitely many treelike chains is treelike.

Proof. Let $C(x, y) = C(z_0, z_1)U \dots UC(z_{k-1}, z_k)$, where $z_0 = x$, $z_k = y$, and each $C(z_i, z_{i+1})$ is treelike, $i = 1, \dots, k - 1$. We prove by induction that $C(z_0, z_j) = C(z_0, z_1)U \dots UC(z_{j-1}, z_j)$ is a treelike chain

for all j , $j = 1, \dots, k$. By hypothesis $C(z_0, z_1)$ is a treelike chain. Suppose, then, that $C(z_0, z_j)$ is a treelike chain. By Lemma 4.4, $C(z_0, z_{j+1}) = C(z_0, z_j)UC(z_j, z_{j+1})$ is a chain. But it is also treelike, since $C(z_0, z_j)$ and $C(z_j, z_{j+1})$ are treelike. This completes the proof. $\square$

Theorem 4.3. If (S, d, V) is graphic, then it is treelike if and only if d is a convex metric.

Proof. If (S, d, V) is treelike, then by Theorem 4.1 d is a convex metric. Suppose, then, that d is convex, and show that any 2-chain $L(x, z, y)$ is treelike. Pick any $v \in V$, and we will show that $L(x, z, y)$ has a collection point with respect to v . Since (S, d, V) is graphic, Lemma 4.3 implies that $L(x, z, y)$ contains a chain $C(x, z, y)$ of finitely many treelike chains. By Lemmas 4.6 and 4.7, $C(x, z, y)$ is itself treelike and therefore contains a collection point u_v with respect to v . We will show that u_v serves as a collection point for $L(x, z, y)$, so that $L(x, z, y)$ is treelike. There are 2 cases.

Case I. $u_v = x$ or $u_v = y$. Suppose without loss of generality that $u_v = x$ (Fig. 4-11). We will show that for any $t \in L(x, z, y)$, $d(t, v) = d(t, x) + d(x, v)$, so that x is a collection point for all of $L(x, z, y)$. Since $u_v = x$, we have $d(y, v) = d(y, x) + d(x, v) = d(y, t) + d(t, x) + d(x, v) \geq d(y, t) + d(t, v) \geq d(y, v)$, using the triangle inequality. Thus $d(t, v) = d(t, x) + d(x, v)$, as desired.

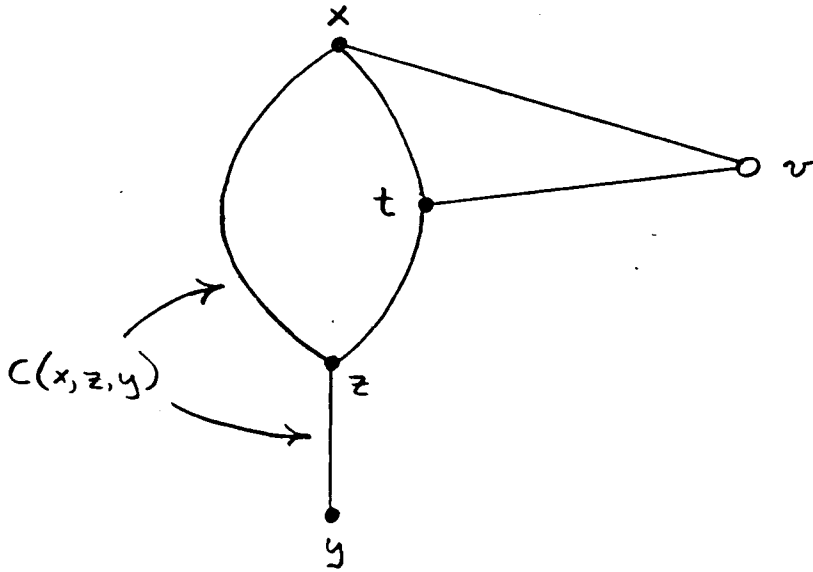


Fig. 4-11. Illustration of Theorem 4.3, Case I.

Case II. $u_v \neq x, y$ (Fig. 4-12). In this case u_v is interior to $C(x, z, y)$. If $v \in C(x, z, y)$ then $u_v (=v)$ is obviously a collection point for $L(x, z, y)$. Thus we suppose $v \notin C(x, z, y)$, so that u_v is an interior exit point and therefore a node. Again we show that for any $t \in L(x, z, y)$, $d(t, v) = d(t, u_v) + d(u_v, v)$. If $t \in L(x, z)$, we know t lies in a chain $C(x, t, z)$ of finitely many treelike chains, and that $L(z, y)$ contains a chain $C(z, y)$ of finitely many treelike chains; in this case let $C(x, t, y) = C(x, t, z) \cup C(z, y)$. If $t \in L(z, y)$, similarly let $C(x, t, y) = C(x, z) \cup C(z, t, y)$. Now from Lemmas 4.6 and 4.7, $C(x, t, y)$ is itself treelike. Thus since v is a node, $C(x, t, y)$ contains a collection point u'_v with respect to v . Also by Lemma 4.1 $u_v \in C(x, t, y)$, so that

$$d(u_v, v) = d(u_v, u'_v) + d(u'_v, v) . \quad (4.16)$$

Now if $u'_v = x$ or y , then by the argument of Case I u'_v is a collection point for all of $L(x, z, y)$, and the theorem is proved. Thus we may assume u'_v is interior to $C(x, t, y)$. If $v \in C(x, t, y)$, obviously $u'_v (=v)$ is a collection point for all of $L(x, z, y)$, so that we may assume $v \notin C(x, t, y)$. So, u'_v is an interior exit point and therefore a node. By Lemma 4.1 $u'_v \in C(x, z, y)$, so that

$$d(u'_v, v) = d(u'_v, u_v) + d(u_v, v) . \quad (4.17)$$

(4.16) and (4.17) imply that $d(u_v, u'_v) = 0$ or $u_v = u'_v$. Thus u_v is a collection point for $C(x, t, y)$, and $d(t, v) = d(t, u_v) + d(u_v, v)$, as desired.

□

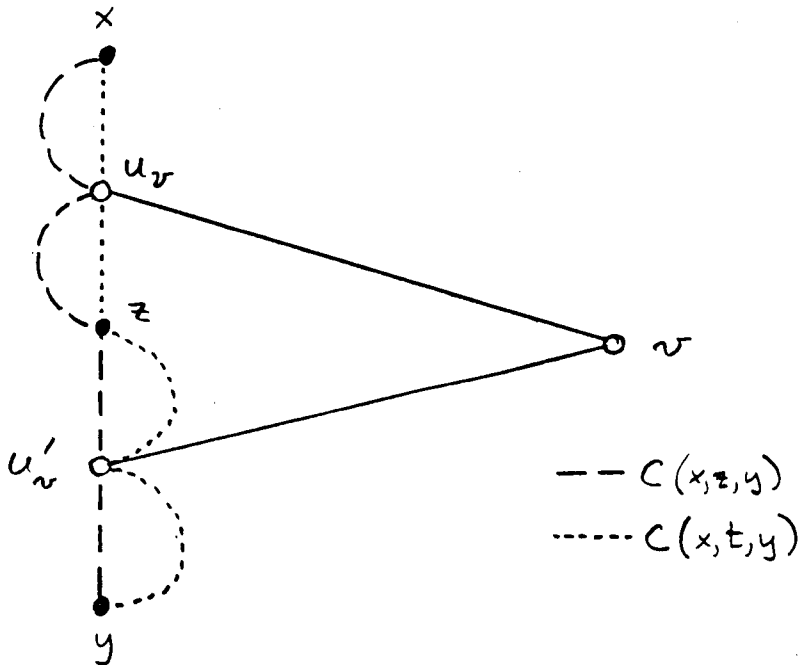


Fig. 4-12. Illustration of Theorem 4.3, Case II.

4.3 Finitely Decomposable Graphic Spaces

If we are to solve location problems on graphic spaces by enumerating treelike components, we obviously prefer that the components be finite in number, as they are for ordinary networks. However, this need not be the case. Let us say that a graphic space (S,d,V) is finitely decomposable if S is the union of finitely many treelike chains. In this section we describe conditions that ensure finite decomposition.

It is trivial to observe that the set S underlying a graphic space (S,d,V) is a union of treelike chains, since every point of S is itself a treelike chain. The difficulty is that the union need not be finite. It is true that every 2-chain $L(x,z,y)$ of S contains a chain $C(x,y)$ of finitely many treelike chains, but this does not ensure finite decomposition, if only because S may be the union of no finite collection of 2-chains.

Consider for example a graphic space (S,d,V) that contains arbitrarily large distances. Let S be the real line, d the Euclidean metric, and $U = \{0\}$. Every 2-chain in S is itself a treelike chain, so that (S,d,V) is indeed graphic. But S is decomposable neither into finitely many 2-chains nor into finitely many treelike chains. Or consider a space (S,d,V) in which S fails to contain all its limit points. Let S be the half-open interval $[0,1)$, and d and V as before. Again every 2-chain is a treelike chain, but no finite union of treelike chains in S exhausts S . Finally, consider the space (S,d,V) for which S is the union over all $\theta \in [0,2\pi)$ of line segments in R^2 extending from the origin

to $(\cos\theta, \sin\theta)$, where $V = \{(0,0)\}$. Given $r_1, r_2 \in [0,1]$ the distance from $(r_1 \sin\theta_1, r_1 \cos\theta_1)$ to $(r_2 \sin\theta_2, r_2 \cos\theta_2)$ is $|r_2 - r_1|$ if $\theta_1 = \theta_2$ and $r_1 + r_2$ otherwise. Here again, each 2-chain is itself a treelike chain; but no finite collection of treelike chains covers S .

A natural way to ensure finite decomposition is to rule out these three types of cases. We easily rule out the first by requiring that the underlying metric space (S,d) be bounded. That is, there is a finite number D such that $d(x,y) \leq D$ for all $x,y \in S$. To require that (S,d) contain its limit points, we must define the idea of a convergent sequence of points in a metric space. Following Blumenthal [1953], let us say that an infinite sequence of points $x_1, x_2, \dots$ of S (not necessarily distinct) is convergent if $\lim_{n \rightarrow \infty} d(x_n, x_{n+1}) = 0$. Also $x \in S$ is a limit of $x_1, x_2, \dots$ if $\lim_{n \rightarrow \infty} d(x_n, x) = 0$. The metric space (S,d) is complete if S contains a limit of any convergent sequence of points in S . We require, then, that (S,d) be complete.

To rule out the third kind of case, let us say that a 2-chain $L(x,z,y)$ is maximal in S if it is properly contained in no 4-chain of S having the form $L(x',x,z,y,y')$. Then we require that S have finitely many maximal 2-chains.

Lemma 4.8. If the metric space (S,d) underlying graphic space (S,d,V) is bounded and complete, then every 2-chain of S is contained in some maximal 2-chain of S .

Proof. Pick any 2-chain $L(x_0, z, y_0)$ of S . Construct a sequence of 2-chains $L(x_0, z, y_0), L(x_1, z, y_1), \dots$, each containing the previous one,

as follows. For any $n \geq 1$, if $L(x_{n-1}, z, y_{n-1})$ is maximal, let $x_n = x_{n-1}$ and $y_n = y_{n-1}$. Otherwise let $L(x_n, x_{n-1}, z, y_{n-1}, y_n)$ be a 4-chain properly containing $L(x_{n-1}, z, y_{n-1})$, so that $L(x_n, z, y_n)$ is the next 2-chain in the sequence. Note that for all $n \geq 1$,

$$d(x_{n-1}, y_{n-1}) = d(x_{n-1}, z) + d(z, y_{n-1}) , \quad (4.18)$$

$$\left. \begin{aligned} d(x_n, z) &= d(x_n, x_{n-1}) + d(x_{n-1}, z) , \\ d(z, y_n) &= d(z, y_{n-1}) + d(y_{n-1}, y_n) . \end{aligned} \right\} \quad (4.19)$$

Since (S, d) is bounded, the sequences $\{d(x_n, z)\}$ and $\{d(z, y_n)\}$, $n = 0, 1, \dots$, are nondecreasing and bounded above. This implies by (4.19) that the sequences $\{d(x_n, x_{n-1})\}$ and $\{d(y_{n-1}, y_n)\}$, $n = 1, 2, \dots$, are convergent, so that the sequences of points $\{x_n\}$ and $\{y_n\}$ are likewise convergent. Since (S, d) is complete, S contains limits x^* and y^* of these two point sequences, respectively.

We can now show that $L(x^*, z, y^*)$ is a maximal 2-chain containing $L(x_0, z, y_0)$. To see that $L(x^*, z, y^*)$ is a 2-chain, note that for any $n \geq 1$, $d(x^*, y^*) \geq d(x_n, y_n) - d(x_n, x^*) - d(y_n, y^*) = d(x_n, z) + d(z, y_n) - d(x_n, x^*) - d(y_n, y^*)$, the last equality by (4.18). Taking limits as $n \rightarrow \infty$, we get $d(x^*, y^*) = d(x^*, z) + d(z, y^*)$, so that $L(x^*, z, y^*)$ is a 2-chain. Finally, $L(x^*, z, y^*)$ contains $L(x_0, z, y_0)$ and is maximal by construction.

□

Theorem 4.4. If the metric space (S, d) underlying graphic space (S, d, V) is bounded and complete, and S contains finitely many maximal 2-chains, then (S, d, V) is finitely decomposable.

Proof. Since (S,d,V) is graphic, we know by Lemma 4.3 that any 2-chain $L(x,z,y)$ contains a chain $C(x,z,y)$ of finitely many treelike chains, with $z \in C(x,z,y)$; let us collect them into a set $T[L(x,z,y)]$. Then the union U of the sets $T[L(x,z,y)]$ over all maximal 2-chains $L(x,z,y)$ is a finite collection of treelike chains. Finally, any $z = L(z,z,z)$ in S is by Lemma 4.8 contained in a maximal 2-chain $L(x,z,y)$ and therefore in at least one of the sets in U . This proves that S is the union of finitely many treelike chains. □

It is interesting to note that even when (S,d) is bounded and complete, a finitely decomposable graphic space (S,d,V) may have infinitely many maximal 2-chains. Consider a space (S,d,V) with $S = \{x,y,z_0,z_1, \dots\}$ and $V = \emptyset$ (Fig. 4-13). Let $d(x,y) = d(x,z_1) = 1$,

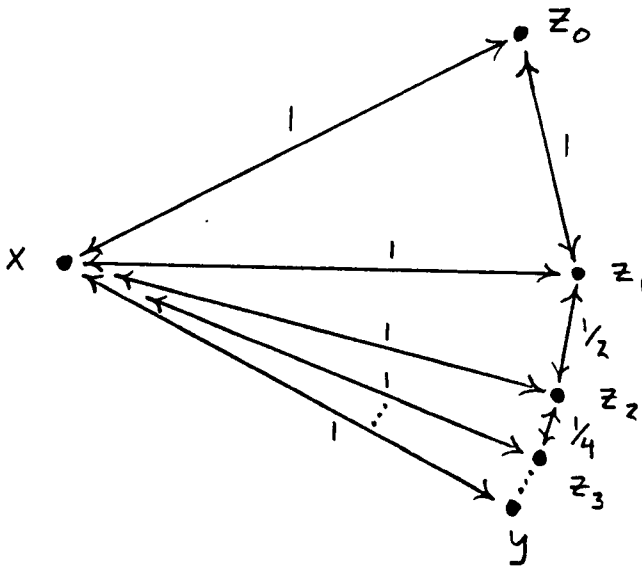


Fig. 4-13. A finitely decomposable graphic space with infinitely many maximal 2-chains.

$d(z_i, y) = 2^{1-i}$, $d(z_i, z_j) = 2^{1-i} - 2^{1-j}$ for $i = 0, 1, \dots, j = i, i+1, \dots$. The space is not only graphic but treelike, since there are no nodes. It is finitely decomposable since $S = L(z_0, y)$, a treelike space and therefore a treelike subspace. But each 2-chain $L(x, x, z_i)$, $i = 0, 1, \dots$, is maximal.

4.4 Models for Graphic Spaces

Since graphic spaces are designed with networks in mind, the question arises as to whether there are any graphic spaces that are not "modeled" by networks. If there are such exotic spaces, then the extension of a network location algorithm to graphic spaces enables one to solve location problems on a broader class of structures. We show in this section that many graphic spaces fail to be modeled by networks or even subsets of networks.

Let $(G, \tilde{d}, N)$ be a network with point set G , metric $\tilde{d}$ and node set N . Let $\tilde{S} \subseteq G$ and $\tilde{V} = \tilde{S} \cap N$. Then $(\tilde{S}, \tilde{d}, \tilde{V})$ models graphic space (S, d, V) if there is a one-to-one correspondence (bijection) $\phi: S \rightarrow \tilde{S}$ such that $\phi(V) = \tilde{V}$ and $d(x, y) = \tilde{d}(\phi(x), \phi(y))$ for all $x, y \in S$.

We have already exhibited several graphic spaces that contain finitely many points and are therefore not modeled by networks. But a finite graphic space may be modeled by a subset of a network. The space of Fig. 4-3 (p.122), for instance, is modeled by the nodes of the cyclic network on four nodes in which each arc has unit length. In fact, it is not hard to see that any subset $(\tilde{S}, \tilde{d}, \tilde{V})$ of network $(G, \tilde{d}, N)$ models graphic space $(S, d, V) = (\tilde{S}, \tilde{d}|_{\tilde{S}}, \tilde{V})$, provided $\tilde{S}$ contains all the treelike

segment boundaries in G . Thus one can solve a location problem on $\tilde{S}$ by solving it on the graphic space $\tilde{S}$ models. This is potentially useful because it permits one to solve a location problem on any network subset containing all the network's treelike segment boundaries.

There are graphic spaces, however, even finite ones, that are not modeled by a subset of any network. We offer the following counter-example. Let (S,d,V) be the graphic space of 5 points shown in Fig. 4-14, where $S = \{x,y,z_1,z_2,z_3\}$ and $V = \{x,y\}$. Distances between adjacent points are shown. First, we note that the space has no

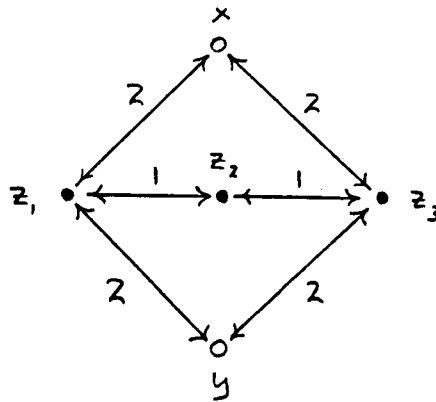


Fig. 4-14. A finite graphic space that is not modeled by a subset of any network.

interior exit points. Also $L(x,y)$ is itself a treelike chain, and $L(z_1,z_3)$ contains the chain of 2 treelike chains, $L(z_1,z_2)$ and $L(z_2,z_3)$. Thus (S,d,V) is graphic. Now suppose, contrary to the claim, that subset $\tilde{S}$ of some network G with nodes $\tilde{V}$ models (S,d,V) , using correspondence ϕ (Fig. 4-15). Consider a shortest path L_1 in G from $\phi(x)$ to

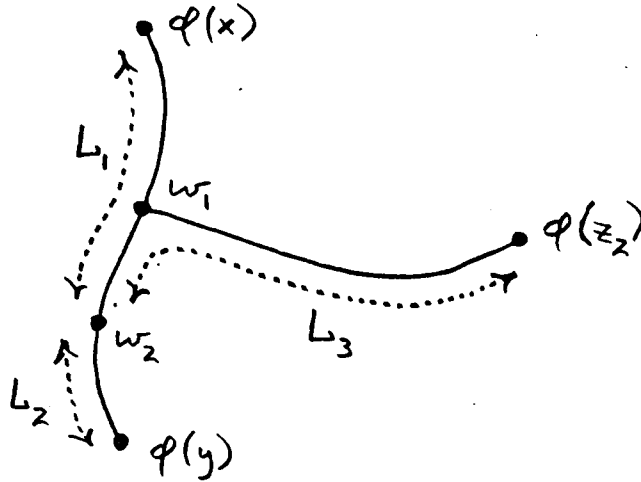


Fig. 4-15. Part of a proposed model for the graphic space of Fig. 4-14.

$\phi(z_1)$, a shortest path L_2 from $\phi(y)$ to $\phi(z_1)$, and a shortest path L_3 from $\phi(z_2)$ to $\phi(z_1)$. L_1 and L_3 must meet at some point w_1 , and L_2 and L_3 at some point w_2 . If $w_1 = w_2 = \phi(z_1)$, $\phi(z_1)$ is a node in G , even though z_1 is not a node in S . This violates the requirement that $\phi(V) = \tilde{V}$. Thus we may suppose without loss of generality that $w_1 \neq \phi(z_1)$. Then if $\tilde{d}$ is the metric on G , $d(x, z_2) = \tilde{d}(\phi(x), \phi(z_2)) \leq \tilde{d}(\phi(x), w_1) + \tilde{d}(w_1, \phi(z_2)) < \tilde{d}(\phi(x), \phi(z_1)) + \tilde{d}(\phi(z_1), \phi(z_2)) = d(x, z_1) + d(z_1, z_2) = d(x, z_2)$, a contradiction.

The above example suggests that if ϕ is permitted to map nonnodes of a graphic space to nodes in the network model, then the graphic space of Fig. 4-14 and many other spaces are modeled by subsets of networks.

Let us relax the condition $\phi(V) = \tilde{V}$ to $\phi(V) \subset \tilde{V}$ and say that any $(\tilde{S}, \tilde{d}, \tilde{V})$ that meets the relaxed condition weakly models (S, d, V) . In this case one can solve a location problem on (S, d, V) by solving it on $(\tilde{S}, \tilde{d}, \tilde{V})$ with the stipulation that $v \in \tilde{V}$ receives positive weight only if $\phi^{-1}(v) \in V$.

It is clear that many graphic spaces fail even to be weakly modeled by a subset of a network. A space without nodes, such as $(\mathbb{R}^n, d, \phi)$ with Euclidean metric d , is trivially graphic but may obviously be embeddable in no network. The same is true of many spaces with nodes, such as the space (S, d, V) of Fig. 4-16. Here $S = \{v\} \cup \{x_{ij} \mid j = 1, 2, \dots, 2^i, i = 0, 1, \dots\}$ and $V = \{v\}$. Also for all $x_{ij}, x_{pq} \in S - \{v\}$ ($i \leq p$), $d(x_{ij}, v) = 2^{-i}$ and

$$d(x_{ij}, x_{pq}) = \begin{cases} 2^{-i} - 2^{-p} & \text{if } j2^{p-i-1} < q \leq j2^{p-i}, \\ 2^{-i} + 2^{-p} & \text{otherwise.} \end{cases}$$

(S, d, V) has no interior exit points and can be shown to be graphic without much difficulty. But it clearly can be embedded in no network.

One is inclined to say that the space of Fig. 4-16 cannot be weakly modeled because it is not finitely decomposable. This suggests,

Conjecture 4.1. Any finitely decomposable graphic space is weakly modeled by a subset of a network.

We can show rather easily that any finite graphic space (S, d, V) can be weakly modeled. Let us say that $x, y \in S$ are adjacent if $L(x, y) = \{x, y\}$. Then construct graph $(G, \tilde{d}, N)$ by letting $N = S$ and connecting every pair x, y of adjacent points in S with an arc (x, y) of length $d(x, y)$.

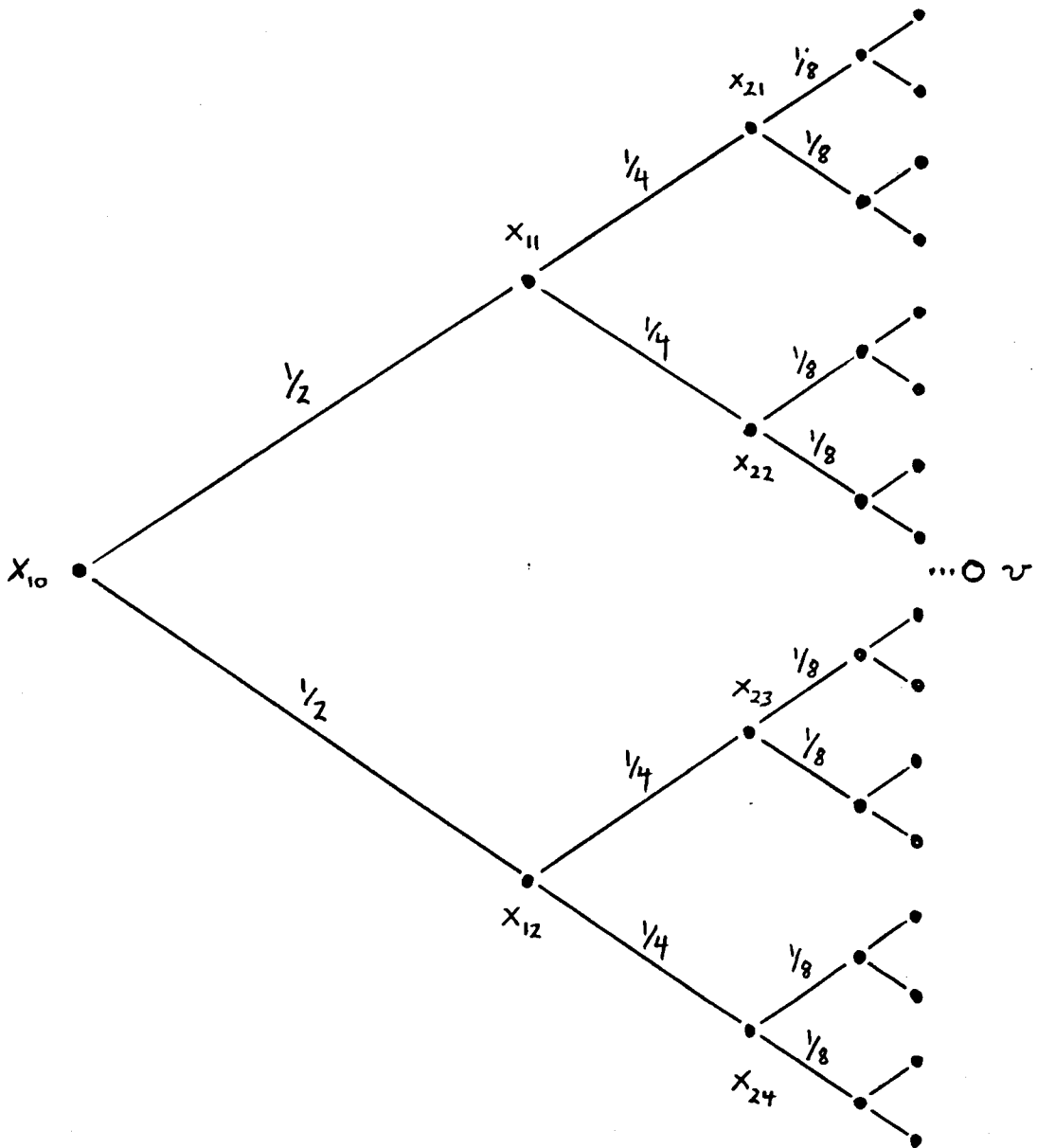


Fig. 4-16. A graphic space that is not weakly modeled by a subset of any network.

Obviously, $(N, \tilde{d}, N)$ weakly models (S, d, V) . But it does so only in a trivial sense, since all the points of S correspond to nodes of G , and the arcs of G play no role in reflecting the structure of (S, d, V) aside from marking the distance between points. We leave further consideration of Conjecture 4.1 to another occasion.

4.5 Convex Optimization on Metric Spaces

Generally the advantage of minimizing a convex function on a convex set is that any local minimum is a global minimum. But this need not be the case in a metric space. We have defined convex subsets of a metric space in such a way (Sect. 4.1) that any entire metric space is trivially convex. But a local minimum on a metric space scarcely need be a global minimum. It is conventional to say that a function f has a local minimum at x if x minimizes f in some ball of positive radius centered at x . By this convention, any f has a local minimum at every point x of a finite metric space, since a sufficiently small ball about x excludes all other points. Obviously f need not have a global minimum at every point, even if f is convex.

In this section we redefine the notion of a local minimum in a way more suited to our definition of a convex set. We also state a condition on metric spaces, "dense connectedness," that is sufficient to ensure that every local minimum of a convex function is a global minimum.

It is easier to identify a local minimum than a global minimum because one can verify that f has a local minimum at x by examining only

a small neighborhood of x . In classical analysis it suffices that x minimize f in an arbitrarily small neighborhood of x . But it may also be easy to verify that x minimizes f in a small neighborhood that is not arbitrarily small. In a finite metric space, for instance, a small neighborhood about x may contain only a handful of points at which f need be evaluated. Let us say, then, that a δ -neighborhood of x is any ball centered at x of radius greater than δ , $\delta \geq 0$. If δ is fixed, we say that f has a local minimum at x if x minimizes f in some δ -neighborhood of x . When $\delta = 0$, we have the conventional sort of local minimum.

Even for positive δ a local minimum of a convex f need not be a global minimum. A metric space in which all distances between distinct points are greater than δ , for instance, would be one in which f has a local minimum at every point but a global minimum at possibly only one point. Let us say that a metric space (S, d) is densely connected for a fixed δ if for every $x, y \in S$ and every δ -neighborhood N of x , $[L(x, y) \cap N] - \{x\}$ is nonempty. Then we can prove,

Theorem 4.5. If metric space (S, d) is densely connected and f is convex on a convex subset D of S , then any local minimum of f on D is a global minimum on D .

Proof. Let f have a local minimum at $z^0 \in D$ (Fig. 4.17). Suppose contrary to the claim that $f(z_0) > f(z^*)$ for some $z^* \in D$. Since z^0 is a local minimum, there is a δ -neighborhood N of z^0 in which $f(z^0)$ is a minimum. Since (S, d) is densely connected, N contains some $x \in L(z^0, z^*)$,

where $x \neq z^0$; $x \in D$ because D is convex. Thus we have $f(z^0) \leq f(x) > f(z^*)$, which violates the convexity of f on D .

□

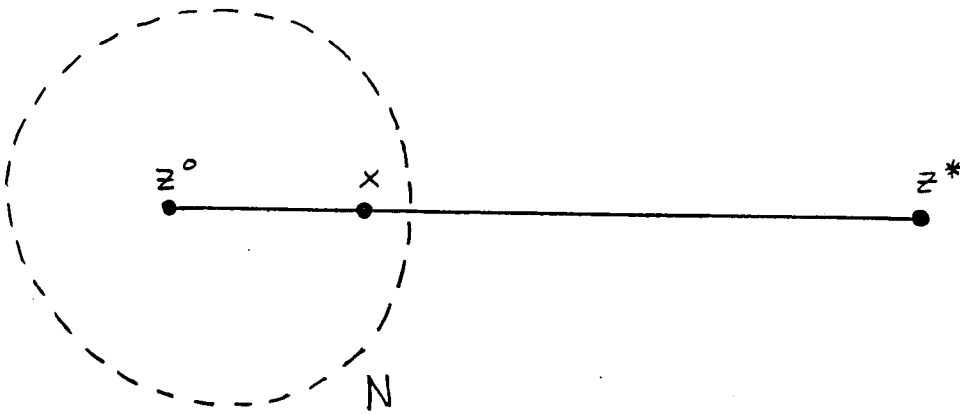


Fig. 4-17. Illustration of Theorem 4.5.

In the remaining sections of this chapter it is presumed that the metric space underlying every graphic space is densely connected for a choice of δ that makes it relatively easy to identify local minima. Otherwise it is of little use to decompose a graphic space into treelike pieces on which convex location problems may be solved.

4.6 Solving Convex Location Problems in Graphic Spaces

Given a finitely decomposable graphic space (S, d, V) , where $V = \{v_1, \dots, v_n\}$, we wish to solve the location problem.

$$\min_{x \in \underline{DCS}} z(x) = f(d(x, v_1), \dots, d(x, v_n)) , \quad (4.20)$$

where the cost function f is convex. When S is a network we can solve (4.20), as in Section 2.2, by solving the convex problem

$$\min_{x \in C} z(x) = f(d(x, v_1), \dots, d(x, v_n)) \quad (4.21)$$

on every treelike segment C of the network. Our aim in this section is to extend this technique to a broad class of finitely decomposable graphic spaces: those with finitely many maximal 2-chains and with a bounded and complete underlying metric space. We will also require that the feasible set D be a union of finitely many convex subsets of S .

We begin by observing that if (S, d, V) is decomposed into a finite number of treelike chains C , (4.21) need not be a convex problem. The space (S, d, V) in Fig. 4-18, for instance, is graphic and finitely decomposable into the single treelike chain $L(x, y)$. But the metric d is not convex on $L(x, y)$. Suppose, however, that we decompose S into the four treelike chains $L(x, w)$, $L(x, z)$, $L(z, y)$, $L(w, y)$. Clearly d is convex on these chains, as desired. We can observe that this is so because each of the four chains, considered in isolation, is a treelike space.

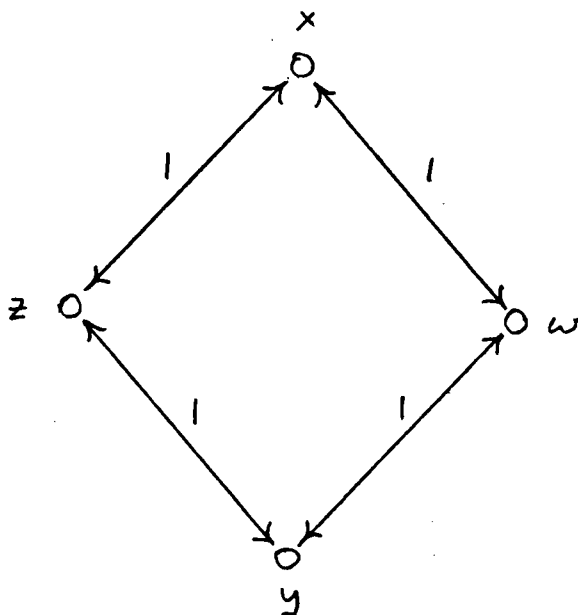


Fig. 4-18. A graphic space finitely decomposable into treelike chains on which the metric is not convex. Here $S = V = \{x, y, z, w\}$.

Let us say that a convex set $T \subset S$ is a treelike subspace of (S, d, V) if every 2-chain in T is a treelike chain of S and $(T, d|_{T \times T}, V \cap T)$ is a treelike space, where $d|_{T \times T}$ denotes the restriction of d to domain $T \times T$. We say that a chain of treelike subspaces is a chain of treelike chains that are also treelike subspaces. Thus $L(x, y)$ in the above example contains a chain $L(x, z, y)$ of two treelike subspaces, and a similar chain $L(x, w, y)$, so that S is decomposable into four treelike subspaces. As expected, we have

Theorem 4.6. The metric d is always convex on a treelike subspace T of graphic space (S, d, V) , in the sense that $d(\cdot, v)$ is convex on T for all $v \in V$.

Proof. It suffices to show that $d(\cdot, v)$ is convex on any line segment $L(x, y)$ in T for any $v \in V$. But since $L(x, y)$ is a treelike chain of S , $L(x, y)$ contains a collection point u_v with respect to v , so that

$$d(w, v) = d(w, u_v) + d(u_v, v) \quad (4.22)$$

for any $w \in L(x, y)$. We first show that $d(\cdot, u_v)$ is convex on $L(x, y)$. This is obviously the case if $u_v = x$ or $u_v = y$, so that we may assume that u_v is interior to $L(x, y)$. If $v \in L(x, y)$, we can set $u_v = v$; if $v \notin L(x, y)$, u_v is an interior exit point. In either case u_v is a node. Since $u_v \in T$ and $(T, d|_{T \times T}, V \cap T)$ is a treelike space, Theorem 4.1 implies that $d(\cdot, u_v)$ is convex on T and therefore on $L(x, y)$. Since $d(u_v, v)$ is constant, it follows from (4.22) that $d(\cdot, v)$ is convex on $L(x, y)$.

□

To reduce (4.20) to a collection of convex subproblems (4.21), then, it suffices to decompose (S, d, V) into a finite collection of treelike subspaces. But it need not even be the case that a line segment $L(x, y)$ of S contain a chain of finitely many treelike subspaces, even though it must contain a chain of finitely many treelike chains. Consider the illustrative space (S, d, V) of Fig. 4-19. Here $S = \{v, x_0, y_0, z_0, w_0, x_1, y_1, w_1, \dots\}$ and $V = \{v\}$. We define $d(z_i, v) = 2^{2-i}$, $d(x_i, v) = d(y_i, v) = d(w_i, v) = 2^{1-i}$, $i = 0, 1, \dots$. We define distances between points in $S - \{v\}$ by regarding any subset $S_n = \{x_0, y_0, z_0, w_0, \dots, x_n, y_n, z_n, w_n, w_{n+1}\}$ as containing the nodes of a network N_n having arcs (x_i, z_i) , (x_i, w_i) , (x_i, z_{i+1}) , (y_i, z_i) , (y_i, w_i) , (y_i, z_{i+1}) , (z_i, w_i) , (w_i, z_{i+1}) , $i = 0, 1, \dots, n$, each having length 2^{-i} . Then the distance

between two points in S_n is the corresponding distance in N_n . Note that there are no interior exit points and that every line segment is itself a treelike chain, so that (S, d, V) is a graphic space. But since the 2-chain $L(x_i, w_i, y_i)$ has no collection point with respect to v for $i = 0, 1, \dots$, no line segment of the form $L(x_i, v)$, $L(y_i, v)$, $L(z_i, v)$ or $L(w_i, v)$ is a treelike space. Since the last member of any chain $C(z_0, v)$ of finitely many treelike chains must have one of these forms, $L(z_0, v)$ contains no chain $C(z_0, v)$ of finitely many treelike subspaces.

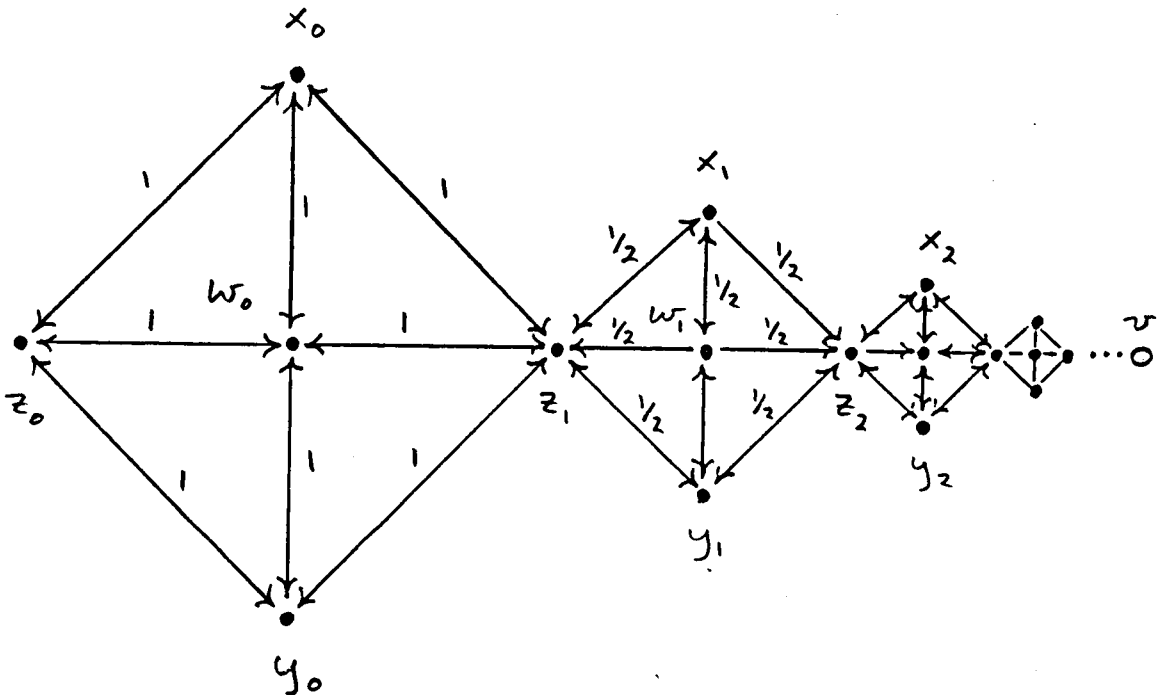


Fig. 4-19. A graphic space in which a line segment need not contain a chain of finitely many treelike subspaces.

We can show, however, that the same conditions that guarantee finite decomposability in Theorem 4.4 also guarantee a decomposition into finitely many treelike subspaces. The crucial result is that if (S, d, V) has finitely many maximal 2-chains, then every 2-chain $L(x, z, y)$ in S contains a chain $C(x, y)$ of finitely many treelike subspaces. To prove this we need two lemmas.

Lemma 4.9. If $L(x, z, y)$ is a 2-chain and $w \in L(x, z)$, then for any $x' \in L(x, z)$, $w \in L(x, x', y)$ implies $w \in L(x, x', z)$ (Fig. 4-20).

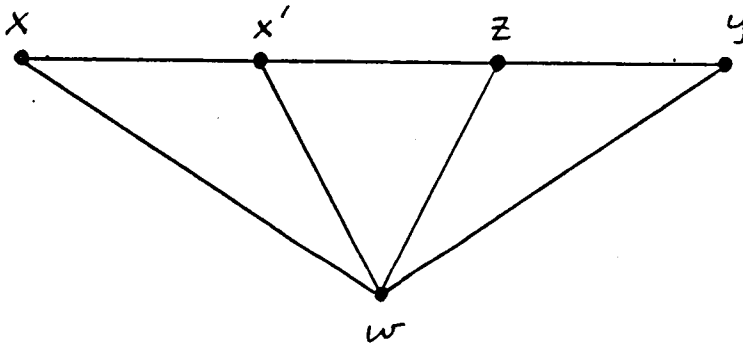


Fig. 4-20. Illustration of Lemma 4.9.

Proof. Let $w \in L(x, x', y)$, so that $w \in L(x, x')$ or $w \in L(x', y)$. In the former case, trivially $w \in L(x, x', z)$, so that we may assume $w \in L(x', y)$. By Lemma 4.5, $z \in L(x', y)$, and we have $d(x', w) + d(w, y) = d(x', z) + d(z, y)$ as well as $d(x, w) + d(w, z) = d(x, x') + d(x', z)$. Adding these equations and observing that $d(x, x') + d(x', z) + d(z, y) = d(x, w) + d(w, y)$, we have $d(x', w) + d(w, z) = d(x', z)$. Thus $w \in L(x', z)$, which implies $w \in L(x, x', z)$, as desired.

□

Lemma 4.10. Let line segment $L(x, y)$ contain a 2-chain $L(x', z', y')$, and let 2-chain $L(x^*, z', y^*)$ contain the 2-chain $L(x, z', y)$ (Fig. 4-21). If $x', y' \in L(x^*, z', y^*)$, then either $x' \in L(x, z')$ and $y' \in L(z', y)$, or $x' \in L(z', y)$ and $y' \in L(x, z')$.

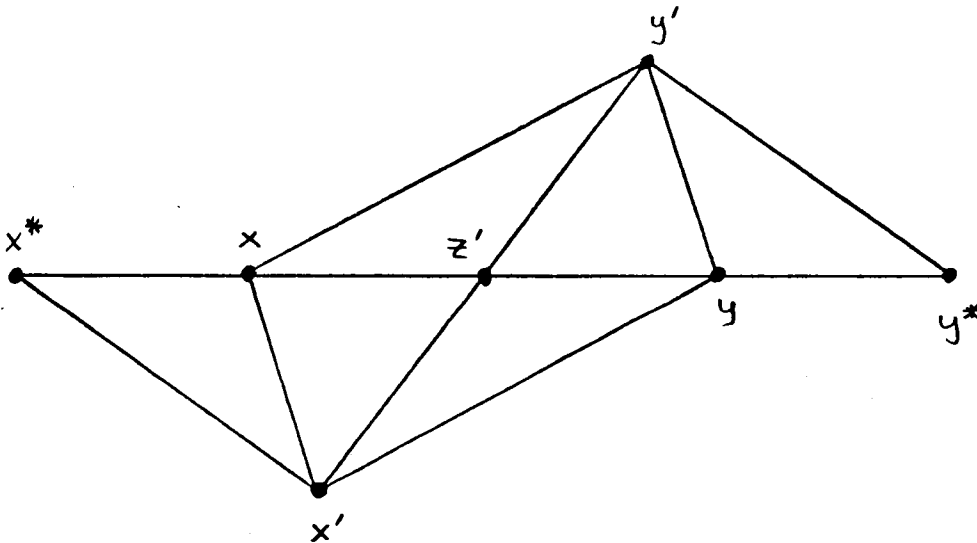


Fig. 4-21. Illustration of Lemma 4.10.

Proof. Since $L(x, z', y) \subset L(x^*, z', y)$, either $x \in L(x^*, z')$ and $y \in L(z', y^*)$, or $x \in L(z', y^*)$ and $y \in L(x^*, z')$. We begin by assuming the former. Lemma 4.5 and $x \in L(x^*, z')$ imply that $x \in L(x^*, y)$. Thus $d(x^*, y) = d(x^*, x) + d(x, y) = d(x^*, x) + d(x, z') + d(z', y)$. From this and the triangle inequality we have that $d(x^*, x') + d(x', y) \geq d(x^*, x) + d(x, z') + d(z', y)$, or

$$d(x', y) \geq d(x^*, x) + d(x, z') + d(z', y) - d(x^*, x'). \quad (4.23)$$

But since $x', z' \in L(x, y)$, we have

$$d(x, x') + d(x', y) = d(x, z') + d(z', y). \quad (4.24)$$

Substituting (4.23) into (4.24) and simplifying, we get

$$d(x, x') \leq d(x^*, x') - d(x^*, x). \quad (4.25)$$

Since $x \in L(x^*, z')$, we have that $d(x^*, x') + d(x', z') = d(x^*, x) + d(x, z')$, or

$$d(x', z') = d(x^*, x) + d(x, z') - d(x^*, x'). \quad (4.26)$$

Adding (4.25) and (4.26), we get that $d(x, x') + d(x', z') \leq d(x, z')$, which implies that $x' \in L(x, z')$. By symmetry $y' \in L(z', y)$. If we begin by assuming $x \in L(z', y^*)$ and $y \in L(x^*, z')$, we can show in similar fashion that $x' \in L(z', y)$ and $y' \in L(x', z')$.

□

Theorem 4.7. If a graphic space (S, d, V) has finitely many maximal 2-chains, then every 2-chain $L(x, z, y)$ in S contains a chain $C(x, y)$ of finitely many treelike subspaces.

Proof. Take any $L(x,y)$ in S . We will construct a chain $C(x,y)$ of finitely many treelike subspaces with a recursive procedure P . Basically, the operation of P is as follows. We begin with a chain $C(x,y) = C(x_0,y_0)$ of finitely many treelike chains, which we know to exist because (S,d,V) is graphic. If all the treelike chains are treelike spaces, we are done. Otherwise, for each treelike chain $C(x_1,y_1)$ of $C(x_0,y_0)$ that is not a treelike space, we call P to replace $C(x_1,y_1)$ with a chain $C'(x_1,y_1)$ of treelike chains. If for each call of P the resulting chain $C'(x_1,y_1)$ is a chain of treelike subspaces, we are done. Otherwise we call P again for each treelike chain $C(x_2,y_2)$ that is not a treelike subspace, moving us one level deeper into an enumeration tree. We will show that the procedure must terminate, leaving a chain $C(x,y)$ of finitely many treelike spaces.

Four arguments are passed to P on each call: n , $C(x_n,y_n)$, L^* and F^* . n is the depth in the enumeration tree. $C(x_n,y_n)$ is a treelike chain in $C(x,y)$ that fails to be a treelike space and that P will replace with a chain of treelike chains. The function of the remaining arguments becomes clear in the algorithm. Briefly, L^* is a list of $n - 1$ nested 2-chains $L(x_k,z_k,y_k)$, $k = 1, \dots, n - 1$. F^* is a list of $n - 1$ objects, some of them nodes and others maximal 2-chains, that is useful in proving the finiteness of the algorithm.

We begin by calling $P(1, C(x_1,y_1), \phi, \phi)$ for each treelike chain $C(x_1,y_1)$ of $C(x_0,y_0)$ that fails to be a treelike subspace. The procedure P follows.

Procedure $P(n, C(x_n, y_n), L^*, F^*)$. Since $C(x_n, y_n)$ is not a treelike subspace, there is a node $v_n \in C(x_n, y_n)$ with respect to which some 2-chain $L(x'_n, z'_n, y'_n)$ in $C(x_n, y_n)$ has no collection point. We consider two cases.

Case I. $v_n \notin L(x_n, w, y_n)$ for some $w \in L(x'_n, z'_n, y'_n)$ (Fig. 4-22). Choose some such w and set $z_n = w$. Since (S, d, V) is graphic, $L(x_n, w, y_n)$ contains a chain $C'(x_n, y_n)$ of finitely many treelike chains. Replace $C(x_n, y_n)$ in $C(x, y)$ with $C'(x_n, y_n)$. For each treelike chain $C(x_{n+1}, y_{n+1})$ of $C'(x_n, y_n)$ that is not a treelike subspace, call $P(n+1, C(x_{n+1}, y_{n+1}), L^* \cup \{L(x_n, w, y_n)\}, F^* \cup \{v_n\})$. If there are no such chains, stop.

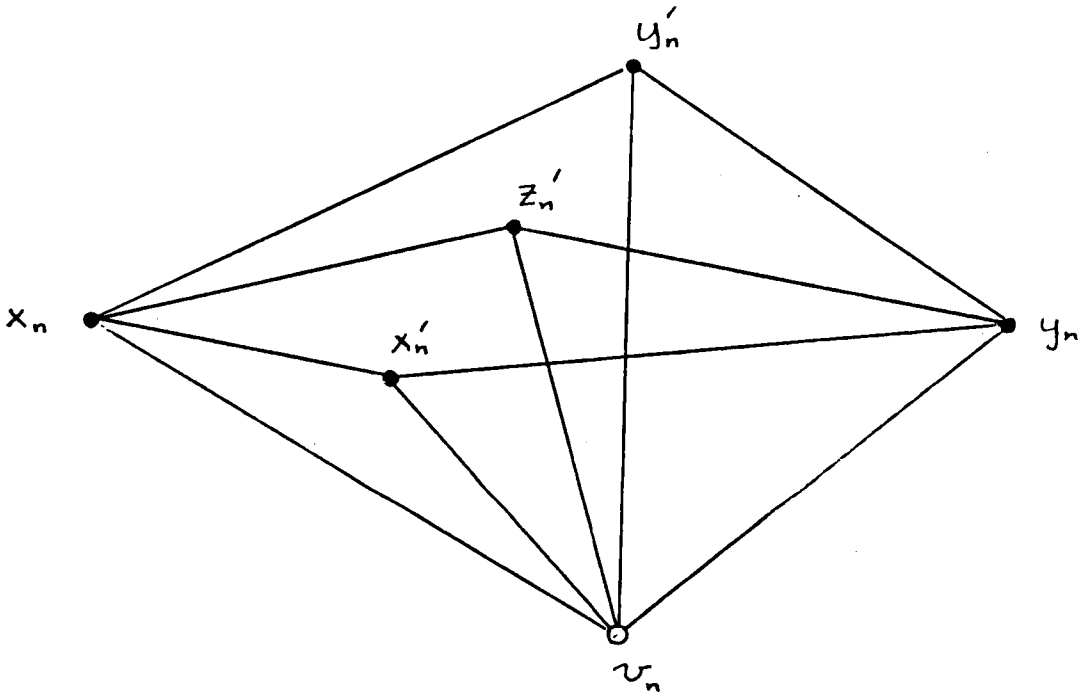


Fig. 4-22. Illustration of Case I in Procedure P.

Case II. $v_n \in L(x_n, w, y_n)$ for all $w \in L(x'_n, z'_n, y'_n)$ (Fig. 4-23). By Lemma 4.8, $L(x_n, z'_n, y_n)$ lies in some maximal 2-chain $L(x_n^*, z'_n, y_n^*)$. Since (S, d, V) is graphic, $L(x_n, z'_n, y_n)$ contains a chain $C'(x_n, y_n)$ of finitely many treelike chains. Let $C'(x_n, y_n)$ replace $C(x_n, y_n)$ in $C(x, y)$. For each treelike chain $C(x_{n+1}, y_{n+1})$ of $C'(x_n, y_n)$ that is not a treelike subspace, call $P(n + 1, C(x_{n+1}, y_{n+1}), L * U\{L(x_n, z'_n, y_n)\}, F * U\{L(x_n^*, z'_n, y_n^*)\})$. If there are no such chains, stop. This completes Procedure P.

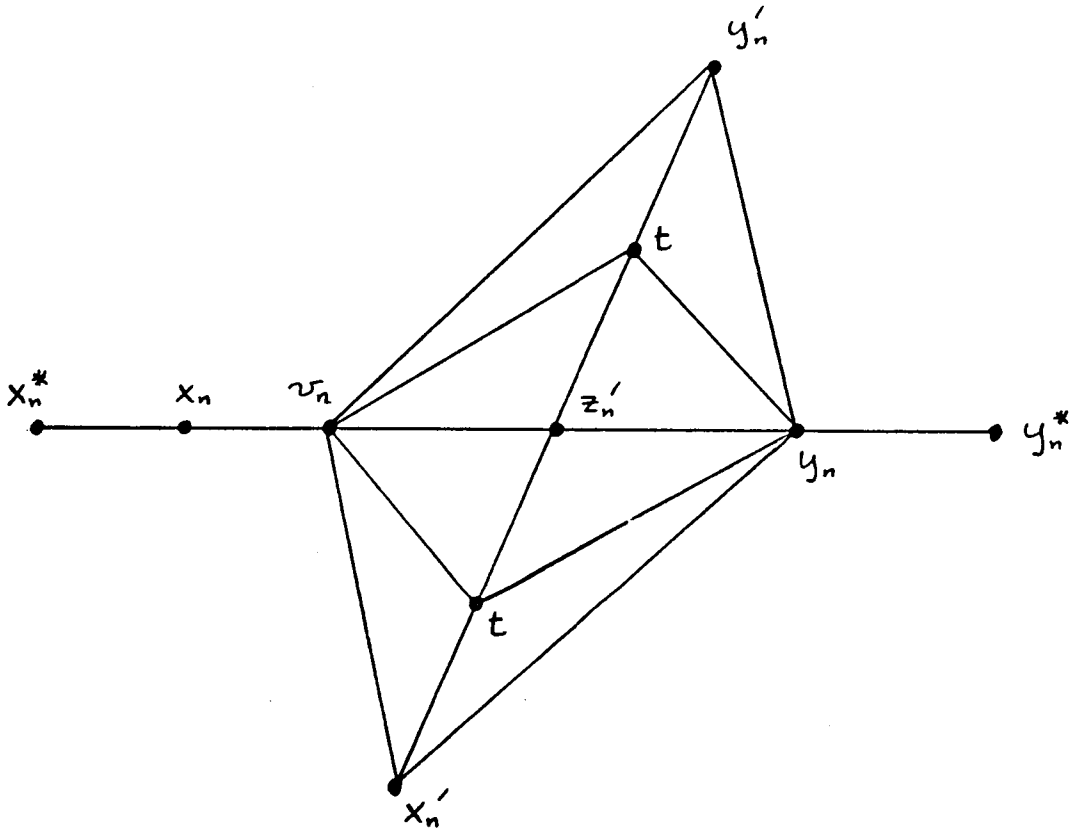


Fig. 4-23. Illustration of Case II in Procedure P.

We first show that at any call of P , all the nodes in F^* are distinct. Note that it suffices to show that

$$v_k \notin L(x_k, z_k, y_k) , \quad (4.27)$$

$$v_k \in L(x_j, z_j, y_j) , \quad (4.28)$$

$$L(x_{n+1}, z_{n+1}, y_{n+1}) \subseteq L(x_n, z_n, y_n) , \quad (4.29)$$

for all $n \geq 0$ and all levels j at which a node v_j is added to F^* in Case I, where k is the level at which a node v_k is next added to F^* . Clearly (4.27) holds by the Case I hypothesis, since $z_k = w$. Also (4.28) holds because $v_k \in C(x_k, y_k) \subseteq C'(x_j, y_j) \subseteq L(x_j, z_j, y_j)$. Since $L(x_n, z_n, y_n)$ contains the chain $C'(x_n, y_n)$ of treelike chains including $C(x_{n+1}, y_{n+1})$, $z_{n+1} \in L(x_{n+1}, y_{n+1})$ implies (4.29).

We next show that at any call of P , all the maximal 2-chains in F^* are distinct. Note that it suffices to show that

$$x'_k, y'_k \in L(x_j^*, z_j^*, y_j^*) , \quad (4.30)$$

$$x'_k \notin L(x_k^*, z_k^*, y_k^*) \text{ or } y'_k \notin L(x_k^*, z_k^*, y_k^*) , \quad (4.31)$$

for all levels j at which a maximal 2-chain $L(x_j^*, z_j^*, y_j^*)$ is added to F^* in Case II, where k is the level at which a maximal 2-chain is next added to F^* . Obviously (4.30) holds, since $x'_k, y'_k \in L(x_j, z_j, y_j) \subseteq L(x_j^*, z_j^*, y_j^*)$.

To show that (4.31) holds, we suppose to the contrary that $x'_k, y'_k \in L(x_k^*, z_k^*, y_k^*)$ and derive that $L(x'_k, z'_k, y'_k)$ has a collection point

with respect to v_k . By Lemma 4.10, $x'_k \in L(x_k, z'_k)$ and $y'_k \in L(z'_k, y_k)$, or $x'_k \in L(z'_k, y_k)$ and $y'_k \in L(x_k, z'_k)$. Also by the Case II description $v_k \in L(x_k, z'_k, y_k)$. Due to symmetry, we can assume without loss of generality that $x'_k \in L(x_k, z'_k)$, $y'_k \in L(z'_k, y_k)$ and $v_k \in L(x_k, z'_k)$. Again by the case description $v_k \in L(x_k, x'_k, y_k)$. Thus by Lemma 4.9, $v_k \in L(x_k, x'_k, z'_k)$, so that either $v_k \in L(x_k, x'_k)$ or $v_k \in L(x'_k, z'_k)$. If $v_k \in L(x'_k, z'_k)$, v_k is a collection point of $L(x'_k, z'_k, y'_k)$ with respect to itself, so that we may assume $v_k \in L(x_k, x'_k)$. In this case we can show that x'_k is a collection point of $L(x'_k, z'_k, y'_k)$ by showing that for any $t \in L(x'_k, z'_k, y'_k)$,

$$d(v_k, t) = d(v_k, x'_k) + d(x'_k, t) . \quad (4.32)$$

Since $x'_k \in L(x_k, z'_k)$ and $y'_k \in L(z'_k, y_k)$, $L(v_k, x'_k, z'_k, y'_k, y_k)$ is a chain. Thus by Lemma 4.4 $x'_k \in L(x_k, y'_k)$. Given this, we infer from $v_k \in L(x_k, x'_k)$, $t \in L(x'_k, y'_k)$ and Lemma 4.5 that $x'_k \in L(v_k, t)$. Now (4.32) follows immediately, so that x'_k is a collection point of $L(x'_k, z'_k, y'_k)$. This violates the hypothesis that $L(x'_k, z'_k, y'_k)$ has no such collection point, so that (4.31) must be true.

Having demonstrated (4.30) and (4.31), we conclude that all the maximal 2-chains in F^* , as well as all the nodes in F^* , are distinct. Thus if the above algorithm did not terminate, the cardinality of F^* would grow without bound. This is impossible, however, since (S, d, V) has finitely many nodes and finitely many maximal 2-chains. Thus the algorithm terminates, producing a chain $C(x, y)$ of finitely many treelike subspaces.

□

The proof of the following corollary is parallel to that of Lemma 4.3.

Corollary 4.1. Every 2-chain $L(x,z,y)$ in a graphic space with finitely many maximal 2-chains contains a chain $C(x,z,y)$ of finitely many treelike subspaces.

We can now derive the finite decomposability result we wanted.

Corollary 4.2. If the metric space (S,d) that underlies graphic space (S,d,V) is bounded and complete, and S contains finitely many maximal 2-chains, then (S,d,V) has a finite decomposition into treelike subspaces.

Proof. Since (S,d,V) is graphic and has finitely many maximal 2-chains, we know by Corollary 4.1 that any 2-chain $L(x,z,y)$ contains a chain $C(x,z,y)$ of finitely many treelike spaces, with $z \in C(x,z,y)$. The rest of the proof is like that of Theorem 4.6.

□

Before stating an algorithm that solves location problem (4.1), we should provide for the possibility that the feasible set D is a proper subset of S . This is straightforward if D is a union of finitely many convex subsets of S . For we have,

Lemma 4.11. Let T be a treelike subspace of graphic space (S,d,V) and D a convex subset of S . Then $T \cap D$ is a treelike subspace of (S,d,V) .

Proof. $T \cap D$ is convex because T and D are convex. Also since T is a treelike subspace, any 2-chain in $T \cap D$ contains a collection point u_v with respect to any $v \in V \cap T \cap D$. Thus $(T \cap D, d|_{T \cap D}, V \cap T \cap D)$ is a tree-like space. This shows that $T \cap D$ is a treelike subspace. □

Now we can state a simple algorithm that solves (4.20).

Step 0. Find a finite decomposition of (S, d, V) into treelike subspaces $T_1, \dots, T_n$. Let the feasible region $D = D_1 \cap \dots \cap D_m$. Let $C_1, \dots, C_p$ be the distinct and nonempty sets in the set $\{T_i \cap D_j \mid i = 1, \dots, n, j = 1, \dots, m\}$. Set $k = 0$ and $z_{\min} = \infty$.

Step 1. Let $k = k + 1$. If $k > p$, go to Step 3. Otherwise solve the convex problem (4.21) with $C = C_k$. Let x_k^* be a solution of (4.1), with $z_k^* = z(x_k^*)$.

Step 2. Set $z_{\min} = \min \{z_{\min}, z_k^*\}$. Go to Step 1.

Step 3. Any x_k^* such that $z_k^* = z_{\min}$ solves (4.1).

4.7 Solving Convex, Nondecreasing, Semiseparable Location Problems in Graphic Spaces

Let the cost function f in (2.1) and (4.20) be convex, nondecreasing and semiseparable. We proved in Chapter 2 (Theorem 2.2) that, given such an f , no point between a treelike segment S of a network G and an optimal location x on the shortest-path tree $T(S)$ rooted at S can be a better location on G than x . This fact permits us to pass over all treelike segments between S and x in a search for a treelike segment containing a globally optimal location, thereby speeding up the search.

In this section we demonstrate an analogous result in graphic spaces whose underlying metric space is complete. We first develop the notion of a shortest-path tree in a graphic space and show that such a tree has convexity properties similar to those of a network tree. We then prove an analogue of Theorem 2.2 for graphic spaces. We conclude with a generalized algorithm for solving convex, nondecreasing, semi-separable single-facility location problems in finitely decomposable graphic spaces whose underlying metric space is complete.

Let (S, d, V) be a graphic space whose underlying metric space (S, d) is complete. We can construct a shortest-path tree $T(r)$ rooted at any $r \in S$ much as one would do so for a network. For any $v \in V$, let $L_r(v)$ be the set containing r and all $v' \in V$ such that $v' \in L(r, v)$. If $L_r(v)$ contains any points besides r and v , arbitrarily choose one as the immediate predecessor $p(v)$ of v ; otherwise let $p(v) = r$. Do this for all $v \in V$, and let $T(r)$ be the union of $L(v, p(v))$ over all $v \in V$. We may refer to each $L(v, p(v))$ as an arc of $T(r)$ and to $L(v, p(v)) - \{v, p(v)\}$ as the interior of the arc. For instance, in the graphic space of Fig. 4-24 we can define the arcs of $T(r)$ to be $L(r, v_1)$, $L(r, v_2)$, $L(r, v_3)$, and $L(v_3, v_4)$, so that $T(r)$ contains the points within the dashed line.

It is convenient to let $P(v)$ be the set of predecessors of $v \in V$, which are the points $y \in S$ that bear the ancestral of the immediate predecessor relation to v ; that is, there is a series of nodes $w_1, \dots, w_k$ ($k \geq 0$) such that $y = p(w_1)$, $w_1 = p(w_2)$, $\dots$, $w_k = p(v)$. Then the join $j(v, v')$ of $v, v' \in V$ is the last common predecessor of v and v' ; that is, $j(v, v') \in P(v) \cap P(v')$, but $j(v, v') = p(v')$ implies $v' \notin P(v) \cap P(v')$.

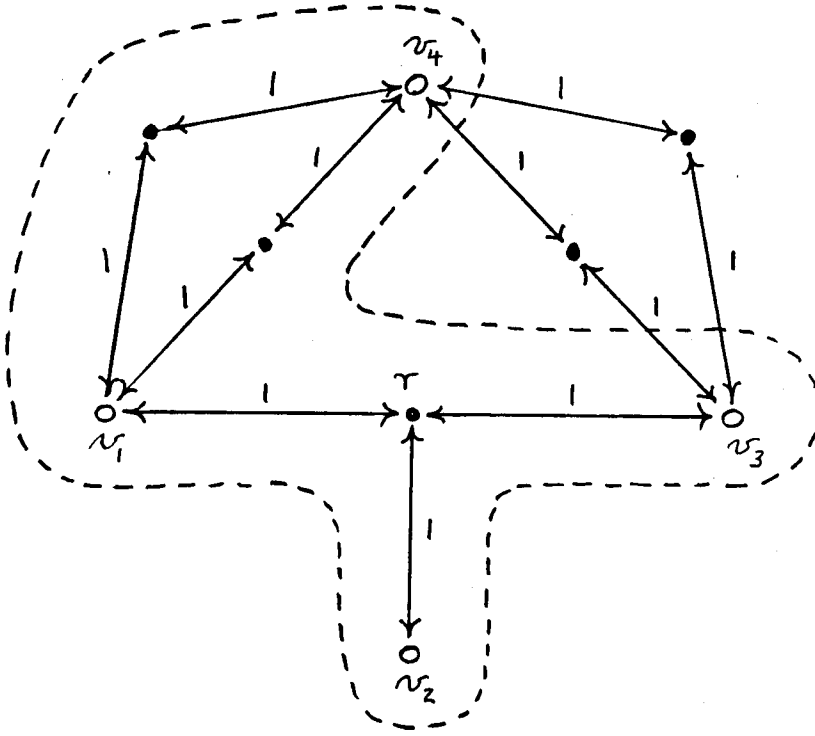


Fig. 4-24. A shortest-path tree in a graphic space.

We define distances $d^T(\cdot, \cdot)$ and line segments $L^T(\cdot, \cdot)$ on $T(r)$ in the obvious way. For $w \in P(v)$ and $v \in V$, define $d^T(w, v) = d(w, v)$ and $L^T(w, v) = L(w, w_1, \dots, w_k, v)$, where $w = p(w_1)$, $w_1 = p(w_2)$, $\dots$, $w_k = p(v)$. For any $v, v' \in V$, define $d^T(v, v') = d(v, j(v, v')) + d(v', j(v, v'))$ and $L^T(v, v') = L^T(v, j(v, v')) \cap L^T(v', j(v, v'))$. If $y \in L(w_1, w_2)$ and $y' \in L(w'_1, w'_2)$, where $w_1 = p(w_2)$ and $w'_1 = p(w'_2)$, define

$$d^T(y, y') = \begin{cases} d(y', w'_2) + d^T(w'_2, w_1) + d(w_1, y) & \text{if } w'_2 \in P(w_1), \\ d(y, w_2) + d^T(w_2, w'_1) + d(w'_1, y') & \text{if } w_2 \in P(w'_1), \\ d(y, w_1) + d^T(w_1, w') + d(w'_1, y') & \text{otherwise,} \end{cases}$$

and

$$L^T(y, y') = \begin{cases} L(y', w_2') + L^T(w_2', w_1) + L(w_1, y) & \text{if } w_2' \in P(w_1) , \\ L(y, w_2) + L^T(w_2, w_1') + L(w_1', y') & \text{if } w_2 \in P(w_1') , \\ L(y, w_1) + L^T(w_1, w') + L(w', y') & \text{otherwise .} \end{cases}$$

Also

$$u \in L_\lambda^T(y, y') \quad \text{if } n \in L^T(y, y') \quad \text{and} \quad d(y, u) = \lambda d(y, y') .$$

To show that $d^T(y, y')$ is well defined, we must show that one can uniquely identify arcs $L(w_1, w_2)$ and $L(w_1', w_2')$ in which y and y' respectively lie. If y is a node it may lie in several arcs, but we can arbitrarily let $L(w_1, w_2) = L(y, p(y))$, and similarly for y' . But if y is interior to an arc, we need the following result.

Lemma 4.11. Let the metric space underlying graphic space (S, d, V) be complete, and let $r \in S$. If $y \in T(r)$ lies in the interior of an arc of $T(r)$, then it lies in no other arc of $T(r)$.

Proof. Suppose to the contrary that y lies in the interior of arc $L(w, v)$ and lies in some distinct arc $L(w', v')$, where $w = p(v)$ and $w' = p(v')$. Then $v \neq v'$, but possibly $w = w'$. Also $v \neq w'$, and $I = L(w, v) \cap L(w', v') - \{w, w'\}$ is nonempty (Fig. 4-25). Since (S, d) is complete and I is nonempty, there is a point $z \in I$ such that $d(z, w') = \sup_{y \in I} \{d(y, w')\}$. Clearly $z \neq v$, since otherwise $v \neq w'$ implies $v \in L(w', v')$ and $w' \neq p(v')$, contrary to hypothesis. Since (S, d, V) is graphic, $L(w, z, v)$ contains a chain $C(w, z, v)$ of treelike chains. Let C_z be a treelike chain of

$C(w, z, v)$ containing z . We first show that the collection point $u_{v'}$ of C_z with respect to v' is z . We know that

$$d(z, v') = d(z, u_{v'}) + d(u_{v'}, v'). \quad (4.33)$$

This means $u_{v'} \in L(z, v')$. But since $u_{v'} \in L(z, v)$, we have $u_{v'} \in I$, so that by the definition of I , $d(u_{v'}, v') \geq d(z, v')$. This and (4.33) imply that $d(z, u_{v'}) = 0$, or $z = u_{v'}$.

Now we show that z is an interior collection point of some chain C'_z in $C(w, z, v)$. We know $z \in C_z$. If z is interior to C_z , just let $C'_z = C_z$. Otherwise z is an endpoint of some treelike chain $C(z, z') \neq C_z$ in $C(w, z, v)$. If $u_{v'}$ is a collection point of $C(z, z')$ with respect to v' ,

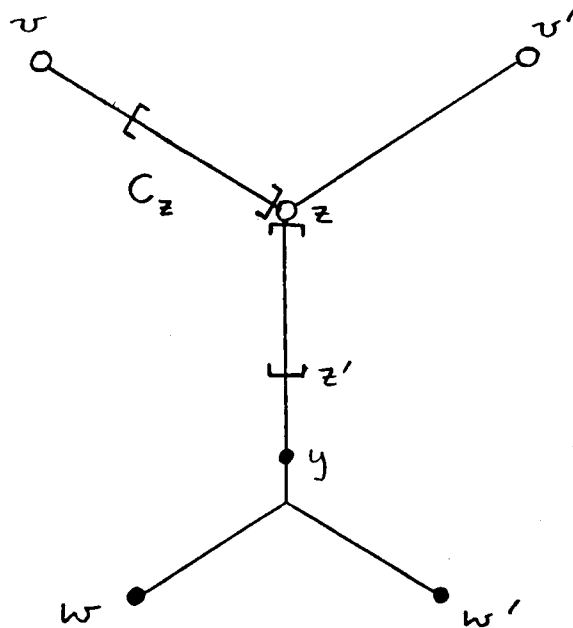


Fig. 4-25. Illustration of Lemma 4.11.

we have that $d(z, v') = d(z, n'_{v'}) + d(n'_{v'}, v')$. But by definition of z , $d(z, v') \leq d(u_{v'}, v')$, so that $d(z, u'_{v'}) = 0$ and $z = u'_{v'}$. Thus z can serve as a collection point of $C'_z = C_z \cup C(z, z')$ with respect to v' , and z is obviously interior to C'_z .

We can now note that z is a node. For either $v' \in C_z$ or $v' \notin C_z$. In the former case it is obvious that $z = v'$, and v' is a node. In the latter case z is an interior exit point of C'_z , so that z is again a node. Since $v \neq v'$ and $z \in L(w, v)$, we see that $w \neq p(v)$, contrary to hypothesis. We conclude that I must be empty, so that $y \notin L(w', v')$. □

Given the above definition of distance d^T on a shortest-path tree $T(r)$, it is easy to show that $T(r)$ has convexity properties similar to those of a network tree. We say that $f: T(r) \rightarrow R$ is convex on $T(r)$ if for any $x, y \in T(r)$, any $\lambda \in [0, 1]$, and any $u \in L_\lambda^T(x, y)$, $f(u) \leq (1 - \lambda)f(x) + \lambda f(y)$. The following lemma is parallel to Lemma 1 of Dearing, Francis and Lowe [1976, p. 633].

Lemma 4.12. If $u \in L^T(x, y)$ and $v \in V$, then $u \in L^T(x, v)$ or $u \in L^T(y, v)$.

Proof. We may assume without loss of generality that $u \in L^T(r, x)$. Clearly $j(v, x) \in L^T(r, x)$, so that we have two cases (Fig. 4-26). Case I: $j(v, x) \in L^T(u, x)$. In this case, clearly $u \in L^T(y, v)$. Case II: $j(x, v) \in L^T(r, u)$. Here $u \in L^T(x, v)$. □

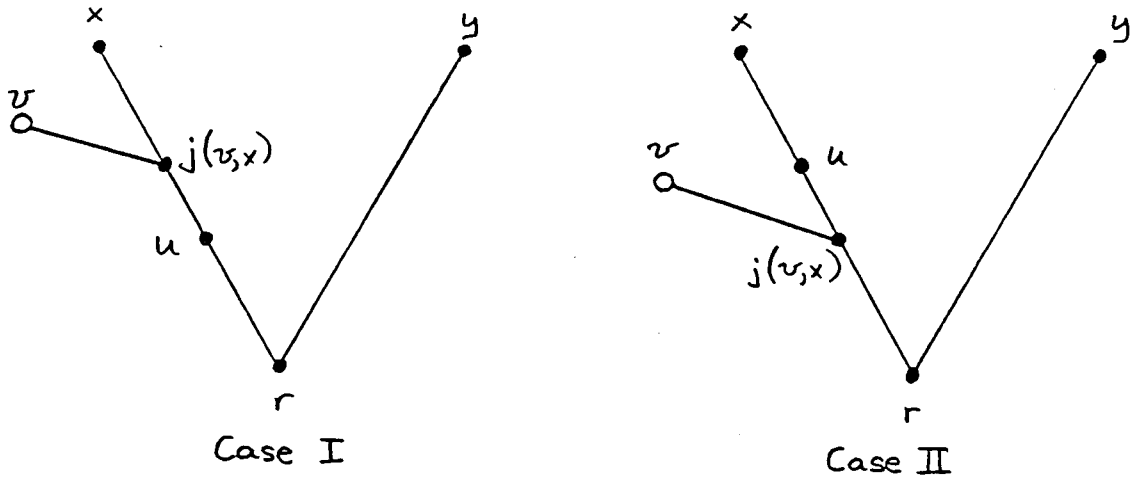


Fig. 4-26. Illustration of Lemma 4.12.

Given Lemma 4.12, the proof of the convexity of $d^T(\cdot, v)$ is identical to the proof of Lemma 2 in Dearing, Francis and Lowe [1976].

Lemma 4.13. Given shortest-path tree $T(r)$ in a graphic space (S, d, V) , where (S, d) is complete, $d^T(\cdot, v)$ is convex on $T(r)$ for all $v \in V$.

An analogue of Lemma 1.1 follows, and its proof is the same.

Lemma 4.14. Given the conditions of Lemma 4.13, with $V = \{v_1, \dots, v_n\}$, if $f: R^n \rightarrow R$ is convex and nondecreasing, then $z: T(r) \rightarrow R$ given by $z(x) = f(d^T(x, v_1), \dots, d^T(x, v_n))$ is convex on $T(r)$.

We now prove a generalization of Theorem 2.2.

Theorem 4.8. Let z be defined as in (4.20), where f is convex, nondecreasing and semiseparable, and let the metric space (S,d) underlying graphic space (S,d,V) be complete. For any $r \in S$, let x solve

$$\min_{y \in T(r)} z(y) = f(d^T(y, v_1), \dots, d^T(y, v_n)) . \quad (4.34)$$

Then any $y \in L^T(r, x)$ satisfies $z(y) \geq z(x)$.

Proof. Let $L^T(r, x) = L(r, w_1, \dots, w_m, x)$. It suffices to show that $z(y) \geq z(w_j)$ for all $y \in L(w_{j-1}, w_j)$ and all $j \in \{1, \dots, m+1\}$, where $w_0 = r$ and $w_{m+1} = x$ (Fig. 4-27). For convenience let $w = w_j$.

We first define,

$$z^T(w) = f(d^T(w, v_1), \dots, d^T(w, v_n)) = f(d_1^T, \dots, d_n^T) , \quad (4.35)$$

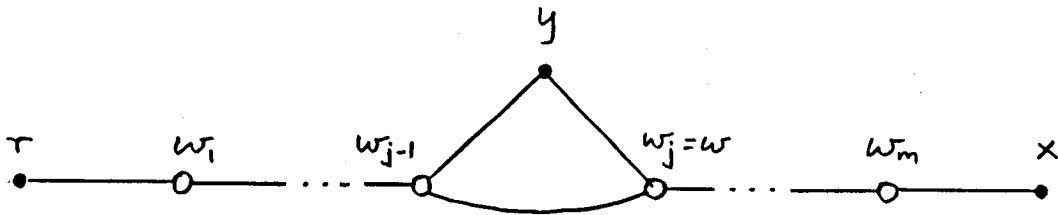


Fig. 4-27. Illustration of Theorem 4.8.

setting $d_i^T = d^T(w, v_i)$ for brevity. Cut $T(r)$ at w and let $v_1, \dots, v_k$ be the nodes in the part of $T(r)$ containing r , excluding w . Then by definition of d^T ,

$$z^T(y) = f(d_1^T - \Delta, \dots, d_k^T - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta), \quad (4.36)$$

where $\Delta = d(y, w)$. Note that $d^T(y, v_i) = d(y, v_i)$ and $d^T(w, v_i) = d(w, v_i)$ for $i = k + 1, \dots, n$. Thus from (4.35) we can write,

$$z(w) = f(d(w, v_1), \dots, d(w, v_n)) = f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T), \quad (4.37)$$

setting $d_i = d(w, v_i)$.

Since f is convex and nondecreasing, we have by Lemma 4.14 that z^T is convex on $T(r)$. Since x solves (4.34), $z^T(y) \geq z^T(x)$. Thus since $w \in L^T(y, x)$,

$$z^T(y) \geq z^T(w). \quad (4.38)$$

This, (4.35) and (4.36) imply,

$$f(d_1^T - \Delta, \dots, d_k^T - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \geq f(d_1^T, \dots, d_n^T). \quad (4.39)$$

But by the triangle inequality $d_i \leq d_i^T$ for all i . Therefore, since f is semiseparable, (4.39) implies

$$\begin{aligned} f(d_1 - \Delta, \dots, d_k - \Delta, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \\ \geq f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T). \end{aligned} \quad (4.40)$$

Also for any $i \in \{1, \dots, k\}$ the triangle inequality yields $d(w, v_i) \leq d(w, y) + d(y, v_i) = \Delta + d(y, v_i)$. Hence $d(y, v_i) = d_i - \Delta_i$, where $\Delta_i \leq \Delta$ and $i \in \{1, \dots, k\}$, so that

$$z(y) = f(d_1 - \Delta_1, \dots, d_k - \Delta_k, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta). \quad (4.41)$$

But since f is nondecreasing, we have from (4.40) that

$$\begin{aligned} f(d_1 - \Delta_1, \dots, d_k - \Delta_k, d_{k+1}^T + \Delta, \dots, d_n^T + \Delta) \\ \geq f(d_1, \dots, d_k, d_{k+1}^T, \dots, d_n^T). \end{aligned}$$

This, (4.37) and (4.41) imply that $z(y) \geq z(w)$, as desired.

□

The following algorithm for solving (4.20) exploits Theorem 4.8. We again suppose that the feasible set D is a union of finitely many convex subsets of S , and that (S, d, V) is decomposable into finitely many treelike subspaces. We require in addition that (S, d) be complete and that f be convex, nondecreasing and semiseparable.

Step 0. Find a finite decomposition of (S, d, V) into treelike subspaces $T_1, \dots, T_p$. Let the feasible set $D = D_1 \cup \dots \cup D_m$. Let $C_1, \dots, C_q$ be the distinct nonempty sets in the set $\{T_i \cap D_j \mid i = 1, \dots, p, j = 1, \dots, m\}$. Set $k = 0$ and $z_{\min} = \infty$.

Step 1. Choose any uneliminated set C_j ; if none remain, go to Step 5.

Step 2. Pick any $r \in C_j$ and construct a shortest-path tree $T(r)$. Solve the convex problem (4.34) for the optimal location x on $T(r)$. Eliminate all sets C_i contained in $L^T(r, x)$. If C_j is eliminated, go to Step 3. Otherwise go to Step 4.

Step 3. If $x \in C_j$, set $k = k + 1$, $x_k^* = x$, $z_k^* = z(x_k^*)$, and $z_{\min} = \min \{z_k^*, z_{\min}\}$. Go to Step 1.

Step 4. Let $k = k + 1$, solve (4.21) on C_j , and let x_k^* be a solution. Set $z_k^* = z(x_k^*)$ and $z_{\min} = \min \{z_k^*, z_{\min}\}$. Eliminate C_j and go to Step 1.

Step 5. Any x_k^* such that $z_k^* = z_{\min}$ solves (4.20).

CHAPTER 5

APPLICATIONS

A wide variety of location problems, especially public-sector problems, require or admit nonlinear network location models. In this chapter we describe a number of such problems and indicate how they may be solved by the algorithms of Chapters 2 and 3.

Location models requiring special nonlinear algorithms for solution seem to arise most often in public-sector contexts where the concern is to maximize social welfare. The private sector is most often concerned to minimize transport costs or maximize the accessibility (minimize the inaccessibility) of, say, a retail outlet. Both transport cost and inaccessibility are usually concave functions of distance. When they are, no special nonlinear algorithm is necessary because there is always an all-node solution, so that the problem is soluble by one of the p -median algorithms.

Distance-related social costs, however, are not generally concave functions of distance. Although concave transport cost functions play a role in the reckoning of social cost, the inequity or unfairness of a facility's location is at least as important. We will see that cost functions that attempt to measure social disutility tend to be convex and not concave.

Although location problems requiring nonlinear algorithms most often occur in the public sector, this is not to say they are rare in the private sector. Transport cost and other matters of concern to a

business sometimes give rise to nonconcave cost functions. A firm may also want to consider public welfare in its location decisions, perhaps because it wants to promote public relations, is subject to regulation, or simply is concerned about its contribution to society.

One goal of this chapter is to illustrate the wide variety of nonlinear location problems that can be solved by the algorithms of Chapters 2 and 3. These algorithms accommodate any network location model in which the number of facilities is fixed, the demand points are located at nodes, cost is a convex function of distances, and each node is served by a nearest facility. Furthermore, a slight adaptation of the algorithms permits solution of problems with side constraints (Section 5.5). We will also see (Section 5.1) that problems with nonconvex cost functions need not be insoluble. Decomposition into treelike segments, on which our algorithms are based, can be helpful even when the cost function is not convex. We have already noted that if the function is concave, the problem can be solved with one of the p -median algorithms.

It is convenient to classify applications under five overlapping categories. We begin in Section 5.1 with applications that often, due to a concave cost function, do not require a nonlinear algorithm: those in which one wishes to minimize transport cost or maximize accessibility. We also discuss what to do when the cost function is neither concave nor convex. Section 5.2 discusses some of the objective functions that have been proposed for maximizing social welfare.

Section 5.3 extends the discussion by focusing on measures of equality that may be relevant to a social welfare problem. Section 5.4 describes the large variety of obnoxious facility location problems that we are equipped to solve. The last section shows how to solve problems with various kinds of side constraints; these turn up in both private and public sector contexts.

It is worthwhile to review the algorithms we have developed and their abilities. The general convex algorithm of Section 2.2 solves any single-facility problem for which the cost function is convex. When f is differentiable or is one of certain nondifferentiable functions (to be addressed on a case-by-case basis), the upper bounding technique of Section 2.3 can speed the solution. If f is nondecreasing and semiseparable as well as convex, the efficient single-facility algorithm of Section 2.5 applies. Small multiple-facility problems with convex cost functions can be solved by the algorithm of Section 3.2. Again differentiable and certain nondifferentiable cost functions permit an easier solution (Section 3.3), as do functions that are nondecreasing and semiseparable (Section 3.4). The multiple-facility algorithms are untested at this writing, however.

5.1 Applications Involving Transport Cost or Accessibility

A network location model most obviously suits a problem in which one wishes to minimize transport costs or maximize accessibility. Here "cost" may be interpreted as dollar cost, time, or some other measure of

disutility. "Accessibility" of a facility to a node is usually measured as the degree of "spatial interaction" between them, which in practical terms is the volume of traffic between them. So that we consistently speak of minimization, it is convenient to minimize the negative of accessibility, or inaccessibility, rather than to maximize accessibility. Then using the notation of Section 1.1, the problem of minimizing transport cost or inaccessibility can be stated,

$$\min z(x) = f(d_x(v_1), \dots, d_x(v_n)) = \sum_{i=1}^n g_i(d_x(v_i)) , \quad (5.1)$$

where each g_i is possibly nonlinear. Obviously, if each g_i is convex, then so is f . We also showed in Section 2.4 that if each g_i is convex and nondecreasing, then f is convex, nondecreasing and strongly semi-separable. This permits solution by the fast algorithms of Section 2.5 in the case of single-facility problems and Section 3.4 in the case of multiple-facility problems.

We have already remarked that transport cost is generally a concave function of distance, since unit transport costs tend to decrease with distance. Such problems have an all-node solution, and there is no need for an algorithm designed for nonlinear models. But nonconcave transport cost functions do exist, as when all provisions for the trip must be carried along, resulting in a heavier load for longer trips. These problems can be solved with the algorithms of Chapters 2 and 3. The same algorithms can solve the large class of problems for which the cost

function is both concave and convex (i.e., linear), albeit it is doubtless more efficient to find an all-node solution simply by examining the nodes.

Inaccessibility, likewise, is generally taken to be a concave function of distance. The most popular measure of inaccessibility is that provided by the gravity model (Olsson [1965], Taylor [1975]). In general form the gravity model puts

$$g_i(d) = -Kw_i h_i(d) , \quad i = 1, \dots, n , \quad (5.2)$$

where w_i represents the "mass" of node i and K a constant analogous to the universal gravitational constant of Newtonian physics. In the early social physics models, introduced in the last century and more recently revived by Stewart [1941], $h_i(d) = d^{-2}$. In this case (5.2) resembles the inverse-square law for gravitational attraction, and w_i represents the population at location i and $g_i(d)$ the flow of people between location i and the facility serving it.

It is clear that f is concave whenever each h_i in (5.2) is convex, and in most versions of the gravity model, each h_i is convex. Obviously the Newtonian function $h(d) = d^{-2}$ is convex. The "principle of least effort" proposed by Zipf [1946] puts $kh(d) = 1/d$, another convex function. It is common to set $h(d) = d^b$ and estimate b statistically from empirical travel data. When $d \geq 1$ or $d \leq 0$, as it normally is, the resulting function is convex. Exponential models of the form $h(d) = k \exp[-b x(d)]$ are also popular. The geneticist Goux [1962] has classified these models as exponential (simpliciter), $s(d) = d$; normal, $s(d) =$

d^2 ; square-root exponential, $s(d) = d^{\frac{1}{2}}$; pareto, $s(d) = \log d$; and log-normal, $s(d) = (\log d)^2$. In each case except the normal and log normal, $h(d)$ is convex if $k \geq 0$ and $b \geq 0$, resulting in a concave cost function f . The normal case is convex when $d \geq 1/\sqrt{2b}$, and the log normal when $d \geq e$.

Gravity models in which the cost function is a type of exponential function have been shown to ensue from a maximum-entropy distribution of traffic flows (e.g., Wilson [1980]). Such models have the advantage of a rigorous theoretical underpinning rather than dubious analogies with physics. But it appears to make sense to maximize entropy only when the total traffic flow is fixed, whereas the point of an optimization model is to maximize the total flow.

Since f in (5.2) is normally concave, it is usually possible to maximize accessibility on a network by examining nodes only. The nonlinear algorithms of Chapters 2 and 3 are useful only in the exceptional case in which the inaccessibility functions g_i are convex and not concave.

We conclude by observing that a problem for which the cost function f is neither convex nor concave need not be insoluble. Our method of decomposing the network into treelike segments can yet be helpful. Finding an optimal location on a treelike segment is no longer a convex problem, but the arguments of the cost function f are still linear functions of a scalar t , as in (2.8). Thus it may be reasonably easy to solve the line search problem (2.7) on a treelike segment even when f is not convex.

5.2 Social Welfare Applications

It is in the nature of location problems that competing interests must be resolved. A facility serves many demand points, and if it is closer to one it is more distant from others. The most straightforward compromise is to choose the most efficient location: the one that minimizes total transport cost, maximizes total accessibility, or whatever. But the most efficient location may place an unreasonable hardship on certain groups or be otherwise unjust. This fact has been recognized in the regional science literature, if belatedly (Morrill [1974], McAllister [1976], Morrill and Symons [1977], Bigman and ReVelle [1978, 1979], Domański [1979]).

Consider the following recent example. In 1982 Mayor Randy Tyree of Knoxville proposed that two new fire stations be built with the help of Federal funds and that six stations in older parts of the city be closed, with a net savings in long-run cost (City of Knoxville [1982]). He defended this proposal in part by pointing out that stations in older sections of the city are positioned more closely together than stations elsewhere, since they were built at a time when horses drew fire wagons at a fraction of the speed achieved by modern fire trucks. For this reason six of the old stations could be replaced with two new ones without degrading the level of service. This was taken to mean roughly that the city's worst-case response time would not be increased.

Inner city residents vigorously opposed the Mayor's proposal, partly on the grounds that population is denser and fires more frequent

in the older neighborhoods. The implication was that total fire damage would be less if stations were more densely positioned in the old neighborhoods than in the suburbs, since the city's average response time would be less. At this writing the issue remains unresolved.

It is clear that the Mayor's position, fully articulated, is that equity is the proper objective in fire hall location, in the sense that no section of the city should unnecessarily suffer a longer response time than other sections. Rather, fire halls should be distributed so that the longest response time citywide is kept as short as possible. The position of the inner city groups, fully articulated, is that efficiency is the proper objective and that residents of sparsely populated suburbs should tolerate long response times to achieve a shorter average time citywide. The Mayor implicitly favors something resembling a center function (Section 1.1) as the measure of disutility, since it measures the worst-case response time. The inner city groups implicitly favor the median function, since it measures a weighted average response time. Perhaps a middle-of-the-road group would be happy with the cent-dian function, which is a convex combination of the median and center functions, although it is difficult to see how any given weighting of the two functions could be rationally defended. In any case, this need to reconcile equity and efficiency is typical of location problems in which social welfare is of concern. The fire hall dispute is curiously atypical in one respect, however, since in it one finds the more affluent suburban residents defending an equitable solution and the low-income groups defending an efficient solution, just the reverse of the usual.

If equity is to be an important element of location models, it must somehow be quantified. The received view seems to be that equity and justice, unlike wealth or cost, inherently resist quantification. If so, any discussion of equity-sensitive cost functions is an idle pastime. But on reflection it is far from obvious that wealth is in principle quantifiable in a way that equity is not. How in principle, for instance, can one justify the claim that my house is 1.28 times as valuable as yours? How does one quantify the relative advantage of two stories versus one, or of a wooded lot versus a convenient location? Yet despite the fundamental difficulties involved, we manage to assign precise dollar figures that are generally regarded as having some relation to value. If we can overcome theoretical difficulties in the measurement of wealth, perhaps we can do the same in the measurement of equity.

It may be objected that wealth measurement has the advantage that prices of goods are determined in a market economy, whereas the value of a just distribution of goods is not. But this is only a practical, not a theoretical advantage. Only the most doctrinaire laissez-faire economist would contend that a good's market price is by definition an indication of its value. The fact that the market economy provides measures of wealth does not in itself imply that those measures are theoretically defensible in a way that measures of equity are not. On the contrary, it suggests that we are misled into thinking that wealth, and not equity, is inherently quantifiable by the market's ability to generate prices for goods.

Suppose, then, it is agreed that measures of wealth enjoy no obvious theoretical underpinning lacked by measures of equity. It may still be maintained that the lack of a generally accepted practice for quantifying equity (analogous to the market's practice of quantifying wealth) makes equity measures worthless in public-sector decision making. If a certain policy can be shown to maximize wealth as measured in the conventional way, even if that measurement is problematic, the public is likely to be favorably disposed toward the policy. To defend a policy by showing that it maximizes equity would not be nearly so effective, even if the measurement of equity were no less legitimate in theory than that of wealth.

One can read this objection two ways. If the concern is merely to be able to influence the public, then one should dispense with both wealth and equity measurement and engage the services of a demagogue. But if the concern is to influence the public through rational persuasion, then it is premature to claim that equity measurement cannot appeal to reason where wealth measurement can. Arguably, if the public is rationally persuaded to accept market valuations of wealth, it is because these valuations are partially legitimated in the public mind by the relation (such as it is) between prices and consumer preferences. High-priced goods are generally those that consumers are willing to work longer or sacrifice more pleasures to obtain. This suggests that an equity measure somehow grounded in individual preferences could strike the public as being no less reasonable than market valuations of wealth. Such measures have been proposed. Perhaps the best known is that of

John Rawls [1971], who argues (roughly) that a just distribution of wealth is one of which anyone could approve at the beginning of history, without knowing which station in life he will occupy. Rawls deduces from this conception that the proper way to measure distributive justice is to measure the wealth of the least advantaged class of people. Perhaps neither this nor any other accounting of equity is entirely convincing, but if the public finds it reasonable to accept the market's highly imperfect valuation of wealth, its accounting standards are not high.

Confident that to do so is not an idle pasttime, we devote the rest of this section to a review of one systematic approach to deriving formulas for calculating social welfare. The next section deals with measures of equality, often considered an important element of social welfare. In either case we examine the properties of the resulting cost functions and indicate how location problems in which they occur may be solved.

Beginning with the seminal work of Kenneth Arrow [1950, 1963], several attempts have been made to find mathematical techniques for reconciling competing interests in a rational way. Arrow first addressed the ordinal formulation of the problem: given each individual's ranking of a list of alternative policies, how should they be ranked by the society as a whole? More recently, several investigators (notably Sen [1976, 1977], Strasnick [1976], d'Aspremont and Gevers [1977], Deschamps and Gevers [1978], and Maskin [1978]) have examined the cardinal formulation of the problem. It asks how a social ranking may be

derived from the numerical cost (or benefit) each individual experiences under each alternative policy. This formulation is well suited to location theory, since generally it is natural to quantify the cost to each demand point of a given facility location. The problem typically lies in aggregating the costs in a way that reflects the desired conception of social cost.

The principal result of research in this area is that if one requires his method of aggregation to satisfy a very few simple properties that seem dictated by rationality, he is severely limited in the aggregation formulas he may use. Moreover, the precise formulas that are admissible depend on the extent to which it is meaningful to compare the costs experienced by different individuals.

We follow Roberts' [1980] account of these results, adapted mutatis mutandis to the context of location theory. Let $g_i(d_i)$, $i = 1, \dots, n$, be the cost to individual (demand point) i that results if d_i is the distance to the nearest facility. Let $f_g(d)$ be the social cost of a facility location that results in distance vector $d = (d_1, \dots, d_n)$. For convenience we say that $g(d) = (g_1(d_1), \dots, g_n(d_n))$ and that $g(d) \geq g'(d)$ if $g_i(d) \geq g'_i(d)$ for $i = 1, \dots, n$. We will also say that g is convex, linear, etc., when we mean that $g_1, \dots, g_n$ are convex, linear, etc.

We assume that f has unrestricted domain, which is to say that f_g is defined for all g . Also f must satisfy the pareto condition that for all d , $f_g(d) \geq f_{g'}(d)$ whenever $g(d) \geq g'(d)$. Finally, f must exhibit independence of irrelevant alternatives. This provides that given a set

D of distance vectors, if $g(d) = g'(d)$ for all $d \in D$, then $f_g(d) \leq f_g(d')$ if and only if $f_{g'}(d) \leq f_{g'}(d')$ for all $d, d' \in D$.

We characterize the degree of comparability of individual costs by specifying how the cost figures $g_i(d_i)$ may be transformed without affecting the social ranking. Let us say that $\phi = (\phi_1, \dots, \phi_n)$, where $\phi_i: R \rightarrow R$, is an invariance transformation if $f_g(d) = f_{\phi \circ g}(d)$ for all d , where $\phi \circ g = (\phi_1 \circ g_1, \dots, \phi_n \circ g_n)$ is a composition of functions. For example, it may be legitimate to compare only differences between costs. That is, the costs for individuals i and j may sensibly be said to differ by twice as much as those for individuals i and k , but beyond this the precise cost figures are arbitrary. This is known as unit comparability. It is characterized by saying that any ϕ for which $\phi_i(c_i) = \alpha_i + \beta c_i$, $\beta > 0$, $i = 1, \dots, n$, is an invariance transformation.

It can be shown that given unit comparability, any social cost function f that satisfies the above properties must compute a positive linear combination of individual costs:

$$f_g(d) = \sum_{i=1}^n w_i g_i(d_i), \quad w_i > 0, \quad i = 1, \dots, n. \quad (5.3)$$

In other words, unit comparability implies a utilitarian formula for computing social costs.

Suppose next that individual costs are cardinal but are not comparable at all. That is, any ϕ for which $\phi_i(c_i) = \alpha_i + \beta_i c_i$, $\beta_i \geq 0$, $i = 1, \dots, n$, is an invariance transformation. Then one can prove a result

analogous to Arrow's original impossibility theorem, to the effect that there must be a dictator. That is, some individual j enjoys the property that $f_g(d) \geq f_g(d')$ whenever $g_j(d_j) \geq g_j(d'_j)$. In this case we can write,

$$f_g(d) = \alpha + \beta d_j, \beta > 0. \quad (5.4)$$

Individual costs are said to have full comparability if any ϕ for which $\phi_i(c_i) = \alpha + \beta c_i$, $\beta > 0$, $i = 1, \dots, n$, is an invariance transformation. This permits comparison of cost levels as well as differences. Let

$$\bar{g}(d) = \sum_{i=1}^n g_i(d_i)/n,$$

and let (c) be the n -vector $(c_1, \dots, c_n)$. Then under full comparability f must have the form,

$$f_g(d) = \bar{g}(d) + h(g(d) - \bar{g}(d)), \quad (5.5)$$

where h is homogeneous of degree 1, meaning that $h(\beta c) = \beta h(c)$ for all $\beta \in \mathbb{R}$, $c \in \mathbb{R}^n$. Since $h(0) = 0$, $h(g(d) - \bar{g}(d))$ can be seen as the social cost incurred by inequality. An example of such a function h is the maximum function, for which (5.5) becomes

$$f_g(d) = \bar{g}(d) + \gamma \max_i \{g_i(d_i) - \bar{g}(d)\}. \quad (5.6)$$

When $g_i(d_i) = w_i d_i$, $w_i \geq 0$, $i = 1, \dots, n$, this is a weighted cent-dian function. When $\gamma = 1$ we have the weighted center function, a type of Rawlsian rule for determining social cost. We will review other examples shortly.

Suppose that individual costs may be measured on a ratio scale, so that any ϕ for which $\phi_i(c_i) = \beta c_i$, $\beta > 0$, $i = 1, \dots, n$, is an invariance transformation. Then the social cost function must be nondecreasing and homothetic; i.e., it must be a nondecreasing function of the form,

$$f_g(d) = u(h(g(d))) , \quad (5.7)$$

where $h: \mathbb{R}^n \rightarrow \mathbb{R}$ is homogeneous of degree 1 and u is strictly increasing. If we also require that f_g be additively separable (defined in Section 1.1), it must be of the form

$$f_g(d) = \sum_{i=1}^n w_i [g_i(d_i)]^r / r, \quad w_i \geq 0, \quad i = 1, \dots, n, \quad (5.8)$$

where by convention

$$f_g(d) = \sum_{i=1}^n w_i \log g_i(d_i)$$

when $r = 0$. When $r = 1$ the function is utilitarian. The center of gravity of function results from $r = 2$. As $r \rightarrow \infty$ we obtain the lexicographic maximum function, which has the property that $f_g(d) \geq f_g(d')$ whenever $d_i \geq d'_i$ for $i = 1, \dots, m$ ($0 \leq m \leq n$) and $d_{m+1} > d'_{m+1}$. This is

often described as a generalization of the Rawlsian criterion, but a close reading of Rawls [1971, p. 83] reveals that his criterion is equally general.

We now investigate the convexity properties of social cost functions (5.3)-(5.8). It is clear that (5.3) and (5.4) are not only convex but nondecreasing and semiseparable as well (Lemma 2.7). Thus the efficient algorithm of Sections 2.5 and 3.4 may be used to solve location problems involving these cost functions.

Unfortunately (5.5) is not in general convex, even if g is convex. If the inequality measure h is the minimum function, for example, then h is homogeneous of degree 1 but not convex, so that f_g is not convex. But of all the homogeneous of degree 1 inequality measures we will examine in the next section (maximum, range, mean deviation, standard deviation, unnormalized Gini coefficient), only one (standard deviation) fails to be convex. If in addition g is linear (as it is in many applications), f_g is convex (Lemma 1.3). It is likely, then, that (5.5) will be convex in a given application. In particular the special case (5.6) is convex if g is convex, since the maximum function is convex and nondecreasing. If g is the identity function, (5.6) is nondecreasing and semiseparable as well (Lemmas 2.4, 2.7, 2.9). We noted in Section 2.4 that semiseparability of f_g does not obtain in general if g is linear. For in this case the second term of (5.6) is semiseparable but not strongly semiseparable, so that its sum with the first term need not be semiseparable.

The social cost function (5.7) is likewise not in general convex. But since u is increasing, (5.7) is convex whenever h and u are convex. Also, (5.8) is clearly convex for all $r \geq 1$. In social welfare applications it is likely that $r \geq 1$ since $r > 1$ results in more equally distributed costs, and $r < 1$ in less equally distributed costs, than the utilitarian solution given by $r = 1$.

We conclude, then, that the algorithms of Chapters 2 and 3 will accommodate most social welfare applications of cost functions (5.3)-(5.8). In addition, the efficient algorithms of Sections 2.4 and 3.4 are applicable to cost functions (5.3), (5.4) and sometimes (5.6).

5.3 Applications Involving Social Equality

In this section we continue our survey of social welfare applications by focusing on location models with an egalitarian objective function. We will enumerate a number of inequality measures and determine whether they are convex functions, so as to know whether location problems in which they occur can be solved by the algorithms of Chapters 2 and 3. The functions f_g , g and $\bar{g}$ are defined as in the previous section.

We have discussed the Rawlsian function,

$$f_g(d) = \max_i \{g_i(d_i) - \bar{g}(d)\}, \quad (5.9)$$

which is convex if g is linear, since the maximum function is convex (Lemma 1.2). A function truer to Rawls would measure the cost to the

most disadvantaged class rather than the most disadvantaged individual, but functions that do this are likewise convex when g is linear. One such function would be,

$$f_g(d) = \sum_{i \in H(d)} [g_i(d_i) - g(d)]/k, \quad (5.10)$$

where $H(d)$ is the set of the k individuals experiencing the highest cost $g_i(d_i)$. To show that (5.10) is convex when g is linear, it suffices to show that

$$\sum_{i \in H(d)} g_i(d_i)$$

is convex. This follows because if $\bar{d} = (1 - \gamma)d + \gamma d'$ for $\gamma \in [0, 1]$, we have

$$\begin{aligned} \sum_{i \in H(\bar{d})} g_i(\bar{d}_i) &\leq \sum_{i \in H(\bar{d})} [(1 - \gamma)g_i(d_i) \\ &+ \gamma g_i(d'_i)] \leq (1 - \gamma) \sum_{i \in H(d)} g_i d_i + \gamma \sum_{i \in H(d')} g_i d_i, \end{aligned}$$

where the first inequality is due to the convexity of g and the second to the definition of $H(d)$ and $H(d')$.

Sen [1973] reviews several other measures of inequality. A simple one is the range of costs,

$$f_g(d) = \max_i \{g_i(d_i)\} - \min_i \{g_i(d_i)\}. \quad (5.11)$$

The relative range is got by normalizing (5.11)—that is, by dividing it by $g(\bar{d})$. Since the maximum function is convex and the minimum function

concave, (5.11) is convex if g is linear (Lemma 1.2). The normalization of (5.11) is not in general convex. A variation of (5.11) would be the gap between the upper and lower classes,

$$f_g(d) = \sum_{i \in H(d)} g_i(d_i)/k - \sum_{i \in L(d)} g_i(d_i)/k', \quad (5.12)$$

where $L(d)$ is the set of the k' individuals experiencing the lowest cost. We showed that the first term of (5.12) is convex when g is linear; the second term is similarly shown to be concave. Thus (5.12) is convex when g is linear.

Another simple measure of inequality is the mean deviation,

$$f_g(d) = \sum_{i=1}^n |g_i(d_i) - \bar{g}(d)|/n, \quad (5.13)$$

which is convex when g is linear. Very similar is the variance,

$$f_g(d) = \sum_{i=1}^n [g_i(d_i) - \bar{g}(d)]^2/n, \quad (5.14)$$

which is likewise convex when g is linear. The standard deviation (square root of the variance) is not in general convex. But since the variance is a strictly increasing function of the standard deviation, one can always replace a standard deviation objective function with a variance function without changing the optimal solution.

A particularly rich measure of inequality is the Gini coefficient. Recall that the amount of curvature of a Lorenz curve indicates the

degree of inequality in a distribution of costs (or wealth). If the Lorenz curve is a straight diagonal line, there is perfect equality. The (normalized) Gini coefficient is the ratio of the area between this diagonal line and the actual Lorenz curve to the total area under the diagonal. It is remarkable that the unnormalized Gini coefficient can be calculated,

$$f_g(d) = (1/2n^2) \sum_{i=1}^n \sum_{j=1}^n |g_i(d_i) - g_j(d_j)|. \quad (5.15)$$

The normalized coefficient is again got by dividing by $\bar{g}(d)$. Since the absolute value function is convex, (5.15) is convex if g is linear (Lemma 1.2). Unfortunately the normalization of (5.14) is not in general convex.

Another interesting measure is Theil's [1967] entropy measure of inequality, given by

$$f_g(d) = \sum_{i=1}^n g_i(d_i) \log n g_i(d_i). \quad (5.16)$$

The "entropy" of the cost distribution is got by subtracting (5.16) from $\log n$, so that greater equality corresponds to greater entropy. (5.16) is convex whenever g is convex and nondecreasing. To prove this it is enough to show that $g_i(d_i) \log g_i(d_i)$ has a nondecreasing first derivative and is therefore convex. But the derivative is $g'_i(d_i)[1 + \log g_i(d_i)]$, which is clearly nondecreasing when g_i is convex and nondecreasing.

We conclude with Dalton's [1920] measure of inequality. It was originally expressed as the ratio of the sum of utilities resulting from an equal distribution of existing wealth to the sum resulting from the actual distribution. It may be rephrased in terms of costs as follows,

$$f_g(d) = \frac{\sum_{i=1}^n g_i(d_i)}{\sum_{i=1}^n g_i(\bar{d})}, \quad (5.17)$$

where $\bar{d} = \sum_i d_i / n$. This measure makes sense only if g is convex, so that an unequal distribution of distances in fact results in a higher cost than an equal distribution of the same total distance. To this extent the measure has normative implications. Regrettably, (5.17) is not a convex function of distance.

We conclude that when inequality is measured by Rawlsian criteria, the cost range, the mean deviation or variance of costs, or the unnormalized Gini coefficient, a location model that minimizes inequality is soluble by the algorithms of Chapters 2 and 3, provided the individual costs are linear functions of distance. When inequality is measured by Theil's entropy formula, individual costs need only be convex, nondecreasing functions of distance. The algorithms break down if the normalized Gini coefficient or the Dalton measure is used.

5.4 Obnoxious Facility Location

Obnoxious facility location problems, even those with linear cost functions, have hitherto required special techniques for their solution (Section 1.1). Moreover, different cost functions have called for

different techniques. A strength of the single-facility algorithm of Chapter 2 is that it can solve practically any obnoxious facility problem that is likely to arise, including all that have been solved to date. The multiple-facility algorithm of Chapter 3 can solve practically any small problem involving several facilities.

The key is that the individual cost functions in obnoxious facility problems are likely to be convex. The disutility of being near an obnoxious facility can be expected to drop at a lesser and lesser rate as one moves away from the facility (Fig. 5-1). As examples one may consider the odor and unsightliness of a dump or industrial district, the smoke or radiation risk near a power plant, the noise of an airport, or the congestion near a shopping mall. One generally combines the individual costs by taking a weighted average, a maximum, or perhaps a convex combination of the two. In each case the total cost is a convex function of distance if the individual cost functions are convex, so that the location problem is soluble.

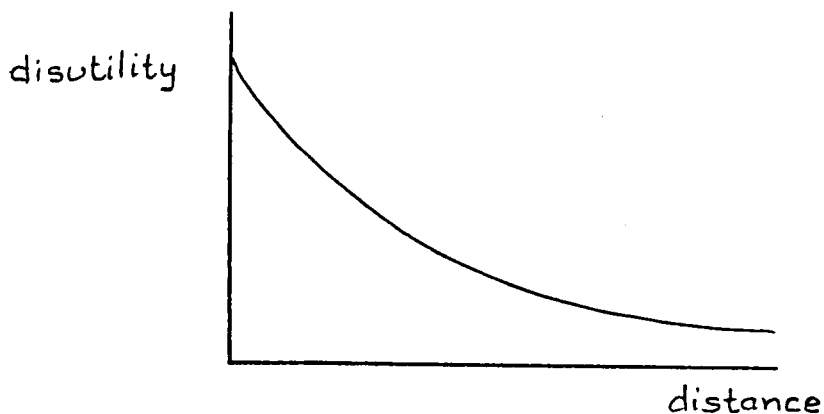


Fig. 5-1. Typical cost function for an obnoxious facility location problem.

In many applications it is both desirable and undesirable to be near a facility, such as a shopping mall or airport. In these cases the cost can be viewed as the sum of the cost of being too near and the cost of being too far. If the cost of being too far is a convex function of distance, the total cost function is likely to be convex, and the location problem is soluble. Even if the cost of being too far is nonconvex, its sum with the cost of being too near may yet be convex, if the positive curvature (second derivative) of the latter is great enough. Therefore the algorithms of Chapters 2 and 3 can solve a large class of obnoxious/beneficial facility location problems.

One type of nonconvex cost function may arise when a firm or government is required to compensate residents in the vicinity of an obnoxious facility, perhaps with a larger payment to residents who live very close to it (Blount [1983]). Here cost is a step function of distance, which is not a convex function. Yet if the function is approximated as a piecewise linear function, the approximation is very likely to be convex, and the problem becomes soluble.

5.5 Applications with Side Constraints

Our solution algorithms, with slight modification, will accommodate a large class of location problems with side constraints. We first show how to solve problems with a few types of network coverage constraints by designing appropriate cost functions. We then present a general method of solving constrained problems that involves a slight alteration of the algorithms.

The simplest sort of coverage constraint requires that every node i lie within a certain distance r_i of a facility. If the original convex cost function is $f(d) = f(d_1, \dots, d_n)$, this constraint can be implemented by using the modified cost function,

$$\tilde{f}(d) = f(d) + M \sum_{i=1}^n \max \{0, d_i - r_i\}, \quad (5.18)$$

where M is very large. Since the second term of (5.18) is a convex function of d , (5.18) is convex. To require in an obnoxious facility problem that every node i lie a distance of at least r_i from all facilities, replace $d_i - r_i$ in (5.17) with $r_i - d_i$.

When $r_i = r$, $i = 1, \dots, n$, (5.17) can be written,

$$\tilde{f}(d) = f(d) + M \max \{0, \max \{d_i\} - r\}. \quad (5.19)$$

If one desires only that the mean of the k greatest distances not exceed r , he can modify (5.19) as follows,

$$\tilde{f}(d) = f(d) + M \max \{0, \sum_{i \in H(d)} d_i / k - r\}, \quad (5.20)$$

where $H(d)$ contains the indices of the k vertices most distant from the nearest facility. Since as we showed in Section 5.3 the summation term of (5.20) is a convex function of d , (5.20) is convex. As with (5.18), (5.19) and (5.20) can be modified so as to keep obnoxious facilities from getting too close to nodes.

Let the ball of radius r about a point x be the set of points on the network that are within a distance r of x . Then we just showed how

to implement inclusion and exclusion constraints with respect to a ball about a node. It is possible to implement constraints with respect to other sets as well. Exclusion constraints are relatively easy to implement. We can exclude facilities from a ball centered at any x by inserting a node at x and using the technique of the previous paragraph. The algorithms of Chapters 2 and 3 already provide for excluding facilities from any arc or interval on an arc. In the latter case, nodes must be inserted at either end of the excluded interval. We can exclude facilities from any union of balls or intervals by excluding them from each ball or interval individually. Probably these devices are adequate for any exclusion constraints that may arise. Inclusion constraints, on the other hand, are more difficult. We know how to include at least one facility in any ball, but it is unclear how we may include at least one facility in a union or even an intersection of balls. However, when inclusion constraints derive from a desire to see that no demand point is too far from a facility, as is the likely situation, inclusion constraints with respect to balls suffice.

We now describe a general way to modify slightly the algorithms of Chapters 2 and 3 so as to accommodate a large class of side constraints. We first take up the single facility case. Suppose that a solution $x \in G$ of (2.1) must satisfy a set of constraints of the form,

$$g_c(d(x, v_1), \dots, d(x, f_n)) < 0, \quad c = 1, \dots, C. \quad (5.21)$$

We solve (2.1) by decomposing the problem into convex problems of the form (2.7). To solve a problem constrained by (5.21), we need only

append these constraints to each convex subproblem (2.7). The appended constraints would be written,

$$\begin{aligned} g_c(d(u, v_1) + t, \dots, d(u, v_j) + t, d(w, v_{j+1}) + b - t, \\ \dots, d(w, v_n) + b - t) < 0, \quad c = 1, \dots, C, \end{aligned} \quad (5.22)$$

where the nodes $v_1, \dots, v_n$ are ordered as for (2.8). The feasible set for (2.7) is $[b(u, x_1), b(u, x_2)]$. If its intersection with the feasible set for constraints (5.22) is convex in every subproblem, then (2.7) remains a convex problem as amended, and the algorithm may proceed.

In the p-facility case we have a set of constraints of the form,

$$g_c(d_x(v_1), \dots, d_x(v_n)) < 0, \quad c = 1, \dots, C, \quad (5.23)$$

where $x \in G^P$. This time we append constraints (5.23) to each convex subproblem (3.14). The appended constraints take the form,

$$\begin{aligned} g_c(d(v_1, y_{a(1)k_1}) + h_{a(1)k_1}(t_{a(1)}), \dots, \\ d(v_n, y_{a(n)k_n} + h_{a(n)k_n}(t_{a(n)})) < 0, \\ c = 1, \dots, C. \end{aligned} \quad (5.24)$$

Here the feasible regions for (3.14) and (5.24) must have a convex intersection in every subproblem.

CHAPTER 6

UNSOLVED PROBLEMS

In the course of the foregoing investigation a number of unsolved problems arose. In this chapter we briefly pose some of the problems pertaining to the theoretical material of Chapters 2, 3 and 4. Although the application of the theory is beset with at least as many unsolved problems, we will not discuss them here.

In Chapter 2 we forewent an average-case performance analysis of the single-facility algorithm because to do so would require solving several nontrivial problems of a more general character. The most obvious one is that of determining the expected number of treelike segments in a network of m arcs on n nodes. Computing expected values of this sort requires some knowledge of the distribution of shortest-path lengths in random networks, a subject in which little or no progress has been made. If it had, it could yet be based on an inappropriate notion of random network. It is difficult to determine even the maximum number of treelike segments in a network of a given size. We showed (Lemma 2.12) that the maximum number of segments for a complete network is $m(n - 1)$, but the maximum for other types of networks is not known in general.

One way to determine the maximum number of treelike segments is to enumerate all relevantly different networks of a given size. This problem of enumerating networks suggests an interesting avenue of

inquiry. We can say that two networks are the "same" in a mathematically interesting sense if they have the same configuration of line segments. More precisely, a network can be identified with a collection of sets, satisfying certain properties, such that each set corresponds to a pair of nodes and contains the nodes in the line segment connecting that pair. Two networks are the "same," then, if they reduce to the same collection of sets. It should not be hard to prove a relationship between the makeup of these sets and the number of treelike segments on each arc. In particular, two networks that are the "same" should have the same number of segments, so that one can determine the maximum number of segments by enumerating all collections of sets representing networks of a given size.

The result of such an inquiry would be a type of network theory that identifies networks, still conceived as metrical objects, with discrete structures rather than the metric spaces of Chapter 4. A fundamental theorem would be that for any given collection of sets satisfying the appropriate conditions, one can exhibit a concrete network that is represented by that collection.

Several other questions suggest themselves. How can the single-facility algorithms be modified to accommodate directed networks? How does the number of treelike segments vary with network size when the network is planar, or has integer arc lengths? Is there a more efficient strategy for enumerating segments in the algorithm of Section 2.5? Can one estimate the probability that a node is an optimal solution?

The major unanswered question of Chapter 3 is the computational one as to how large a problem can be practically solved by an exact multiple-facility algorithm. One way to answer the question would be to write a computer implementation of the algorithms presented and to test it. One could also develop a theoretical understanding of the complexity of the algorithms.

Chapter 4 leaves us with Conjecture 4.1, whose truth is yet to be decided. It states that any finitely decomposable graphic space is weakly modeled by a subset of a network. One may also ask whether there is an efficient procedure for decomposing a finitely decomposable graphic space into a minimal number of treelike subspaces, since solution of location problems is easier when there are fewer treelike subspaces on which to solve subproblems. On the practical side, one should determine to what extent the graphic space location algorithms are useful for solving network problems with complex feasible regions. It is clear that any subset (feasible region) of a network containing all the network's treelike segment boundaries is a finitely decomposable graphic space. The only remaining potential barrier to solution would be a lack of dense connectivity. It may also be useful to solve as graphic space problems those network problems in which only a few nodes are demand points. One would simply designate as nodes of the graphic space only the demand points and the other nodes that turn out to be interior exit points. The result should be a relatively small number of treelike subspaces. The main unresolved issue of Chapter 4, however, is whether it is in the long run fruitful to study location problems in metric spaces.

LIST OF REFERENCES

LIST OF REFERENCES

- Arrow, K. 1950. A Difficulty in the Concept of Social Welfare. Journal of Political Economy 58, 328-346.
- Arrow, K. 1963. Social Choice and Individual Values, Second Edition. Wiley, New York.
- Bigman, D. and C. ReVelle. 1978. The Theory of Welfare Considerations in Public Facility Location Problems. Geographical Analysis 10, 229-240.
- Bigman, D. and C. ReVelle. 1979. An Operational Approach to Welfare Considerations in Applied Public-Facility-Location Models. Environment and Planning A 11, 83-95.
- Blount, S. E. 1983. Facility Siting and Neighbor Compensation. Unpublished Senior Thesis, Princeton University, Princeton, New Jersey.
- Blumenthal, L. M. 1953. Theory and Applications of Distance Geometry. Oxford University Press, London.
- Channing, C. 1978. Facility Location on Deterministic Undirected Trees with Increasing Convex Costs. Working paper, Rensselaer Polytechnic Institute.
- Christofides, N., and P. Viola. 1971. The Optimum Location of Multi-Centers on a Graph. Operational Research Quarterly 22, 145-154.
- Church, R. L., and R. S. Garfinkel. 1978. Locating an Obnoxious Facility on a Network. Transportation Science 12, 107-117.
- City of Knoxville. 1982. Knoxville Fire Station Location Study. Knoxville, Tennessee.
- Clarke, F. H. 1975. Generalized Gradients and Applications. Transactions of the American Mathematical Society 205, 247-262.
- Cornuejols, G., M. L. Fisher and G. L. Nemhauser. 1977. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. Management Science 23, 789-810.
- D'Aspremont, C. and L. Gevers. 1977. Equity and the Informational Basis of Collective Choice. Review of Economic Studies 46, 199-210.
- Dalton, H. 1920. The Measurement of the Inequality of Incomes. Economic Journal 30.
- Dearing, P. M. 1977. Minimax Location Problems with Nonlinear Costs. Journal of Research, National Bureau of Standards 82, 65-72.

- Dearing, P. M. and G. J. Langford. 1975. The Multifacility Total Cost Location Problem on a Tree Network. Technical Report No. 209, Dept. of Mathematical Sciences, Clemson University, Clemson, South Carolina.
- Dearing, P. M. and R. L. Francis. 1974. A Minimax Location Problem on a Network. Transportation Science 8, 333-343.
- Dearing, P. M., R. L. Francis and T. J. Lowe. 1976. Convex Location Problems on Tree Networks. Operations Research 24, 628-642.
- Deschampes, R. and L. Gevers. 1978. Leximin and Utilitarian Rules: A Joint Characterization. Journal of Economic Theory 17, 143-163.
- Diehr, G. 1972. An Algorithm for the p-Median Problem. Working Paper 191, Western Management Science Institute, University of California at Los Angeles.
- Dijkstra, E. W. 1959. A Note on Two Problems in Connection with Graphs. Numerische Mathematik 1, 269-271.
- Domański, R. 1979. Accessibility, Efficiency and Spatial Organization. Environment and Planning A 11, 1189-1206.
- Erlenkotter, D. 1976. A Dual-Based Procedure for Uncapacitated Facility Location. Operations Research 26, 992-1009.
- Farber, M. and R. E. Jamison. 1983. Convexity in Graphs and Hypergraphs. University of Waterloo Faculty of Mathematics Research Report CORR 83-46, Waterloo, Ontario.
- Francis, R. L. 1977. A Note on a Nonlinear Minimax Location Problem on a Tree Network. Journal of Research, National Bureau of Standards 82, 73-80.
- Francis, R. L., L. F. McGinnis and J. A. White. 1983. Locational Analysis. European Journal of Operations Research 12, 220-252.
- Francis, R. L., T. J. Lowe and H. D. Ratliff. 1978. Distance Constraints for Tree Network Multifacility Location Problems. Operations Research 26, 570-596.
- Garfinkel, R. S., A. W. Neebe and M. R. Rao. 1977. The m-Center Problem: Minimax Facility Location. Management Science 23, 1133-1142.
- Garfinkel, R. S., A. W. Neebe and M. R. Rao. 1974. An Algorithm for the m-Median Plant Location Problem. Transportation Science 8, 217-236.
- Goldman, A. J. 1969. Optimum Locations for Centers in Networks. Transportation Science 3, 352-360.

- Goldman, A. J. 1971. Optimal Center Location in Simple Networks. Transportation Science 5, 212-221.
- Goldman, A. J. 1972. Minimax Location of a Facility in a Network. Transportation Science 6, 407-418.
- Goldman, A. J., and P. M. Dearing. 1975. Concepts of Optimal Location for Partially Noxious Facilities. Bulletin of the Operations Research Society of America 23, Supplement 1, B-31.
- Goux, J. M. 1962. Structure de l'espace et migration. In Human Displacements, Entretiens de Monaco en Sciences Humaines: Premier Session, J. Sutter (ed.).
- Hakimi, S. L. 1964. Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph. Operations Research 12, 450-459.
- Hakimi, S. L. 1965. Optimum Distribution of Switching Centers in a Communications Network and Some Related Graph-Theoretic Problems. Operations Research 13, 462-475.
- Hakimi, S. L., E. F. Schmeichel and J. G. Pierce. 1978. On p-Centers in Networks. Transportation Science 12, 1-15.
- Halpern, J. 1976. The Location of a Center-Median Convex Combination on an Undirected Tree. Journal of Regional Science 16, 237-245.
- Halpern, J. 1978. Finding Minimal Center-Median Convex Combinations (Cent-Dian) of a Graph. Management Science 24, 535-544.
- Halpern, J. 1980. Duality in the Cent-Dian of a Graph. Operations Research 28, 722-735.
- Handler, G. Y. and P. B. Mirchandani. 1979. Location on Networks: Theory and Algorithms. The MIT Press, Cambridge, Massachusetts.
- Harary, F. 1969. Graph Theory. Addison-Wesley, Reading, Massachusetts.
- Jamison, R. E. 1979. A Convexity Characterization of Ordered Sets. Proceedings, 11th Southeast Conference for Combinatorics, Graph Theory and Computing, Congressus Numerantium 29, 535-544.
- Jamison, R. E. 1981. Convexity and Block Graphs. Congressus Numerantium 33, 129-142.
- Jamison, R. E. 1982. A Perspective on Abstract Convexity: Classifying Alignments by Varieties. In Convexity and Related Geometry, Proceedings of the Second University of Oklahoma Conference, pp. 113-150, D. C. Kay and M. Breen (eds.). Marcel Dekker, New York.

- Jarvinen, P., I. Rajala and H. Sinervo. 1972. A Branch-and-Bound Algorithm for Seeking the p-Median. Operations Research 20, 173-178.
- Kariv, O. and S. L. Hakimi. 1979a. An Algorithmic Approach to Network Location Problems. I: The p-Centers. SIAM Journal of Applied Mathematics 37, 513-538.
- Kariv, O. and S. L. Hakimi. 1979b. An Algorithmic Approach to Network Location Problems. II: The p-Medians. SIAM Journal of Applied Mathematics 37, 539-560.
- Khumawala, B. M., A. W. Neebe and D. G. Dannenbring. 1974. A Note on El-Shaieb's New Algorithm for Locating Sources among Destinations. Management Science 21, 230-233.
- Kuehn, A. A., and M. J. Hamburger. 1963. A Heuristic Program for Locating Warehouses. Management Science 9, 643-666.
- Levy, J. 1967. An Extended Theorem for Location on a Network. Operational Research Quarterly 18, 433-442.
- Lowe, T. J. 1979. Efficient Solutions in Multiobjective Tree Network Location Problems. Transportation Science 12, 298-316.
- Maranzana, F. E. 1964. On the Location of Supply Points to Minimize Transport Costs. Operational Research Quarterly 15, 261-270.
- Maskin, E. 1978. A Theorem on Utilitarianism. Review of Economic Studies 45, 93-96.
- McAllister, D. M. 1976. Equity and Efficiency in Public Facility Location. Geographical Analysis 8, 47-63.
- Menger, K. 1928. Untersuchungen über allgemeine Metrik. Mathematische Annalen 100, 75-163.
- Minieka, E. 1983. Anticenters and Antimedians of a Network. Networks 13, 359-364.
- Morrill, R. L. 1974. Efficiency and Equity of Optimum Location Models. Antipode 6, 41-46.
- Morrill, R. L. and J. Symons. Efficiency and Equity Aspects of Optimum Location. Geographical Analysis 9, 215-225.
- Neebe, A. W., and M. R. Rao. 1975. A Subgradient Approach to the m-Median Problem. Technical Report No. 75-12, Curriculum in Operations Research and Systems Analysis, University of North Carolina at Chapel Hill.

- Olsson, G. 1965. Distance and Human Interaction: A Review and Bibliography. Regional Science Research Institute, Philadelphia.
- Pasch, M. 1882. Vorlesungen über Neuere Geometrie. B. G. Teubner Verlag.
- Pasch, M. and M. Dehn. 1926. Vorlesungen über neuere Geometrie, zweite Auflage, mit einem Anhang: Die Grundlegung der Geometrie in historischen Entwicklung. Springer Verlag, Berlin.
- Peterson, B. E. 1984. Interline, a Railroad Routing Model: Program Description and User's Manual. Oak Ridge National Laboratory Technical Monograph ORNL/TM-8944, Oak Ridge, Tennessee.
- Picard, J. C. and H. D. Ratliff. 1978. A Cut Approach to the Rectilinear Distance Location Problems. Operations Research 26, 422-434.
- Rand McNally. 1984. Road Atlas. Rand McNally, New York.
- Rawls, J. 1971. A Theory of Justice. Harvard University Press, Cambridge. Massachusetts.
- Revelle, O., and R. W. Swain. 1970. Central Facilities Location. Geographical Analysis 2, 30-42.
- Roberts, K. W. S. 1980. Interpersonal Comparability and Social Choice Theory. Review of Economic Studies 47, 421-440.
- Schrage, L. 1975. Implicit Representation of Variable Upper Bounds in Linear Programming. Mathematical Programming Study 4, 118-132.
- Sen, A. 1973. On Economic Inequality. W. W. Norton, New York.
- Sen, A. 1976. Welfare Inequalities and Rawlsian Axiomatics. Theory and Decision 7, 243-262.
- Sen, A. 1977. On Weights and Measures: Informational Constraints in Social Welfare Analysis. Econometrica 45, 1539-1572.
- Shier, D. R. 1984. Personal communication.
- Shier, D. R. and P. M. Dearing. 1983. Optimal Locations for a Class of Nonlinear, Single-Facility Location Problems on a Network. Operations Research 31, 292-303.
- Stewart, J. Q. 1941. An Inverse Distance Formulation for Certain Social Influences. Science 93, 89-90.
- Strasnick, S. The Problem of Social Choice: Arrow to Rawls. Philosophy and Public Affairs 5, 241-273.

- Tansel, B. C., R. L. Francis and T. J. Lowe. 1983a. Location on Networks: A Survey. Part I: The p-Center and p-Median Problems. Management Science 29, 482-497.
- Tansel, B. C., R. L. Francis and T. J. Lowe. 1983b. Location on Networks: A Survey. Part II: Exploiting Tree Network Structure. Management Science 29, 498-511.
- Tansel, B. C., R. L. Francis, T. J. Lowe and M. L. Chen. 1982. Duality and Distance Constraints for the Nonlinear p-Center Problem and Covering Problem on a Tree Network. Operations Research 30, 725-743.
- Taylor, P. J. 1975. Distance Decay in Spatial Interactions. Geo Abstracts, Ltd., East Anglia, Norwich.
- Teitz, M. B., and P. Bart. 1968. Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. Operations Research 16, 955-961.
- Theil, H. 1967. Economics and Information Theory. North-Holland, Amsterdam.
- Wilson A. G. 1980. Entropy in Urban and Regional Modelling. Pion, Ltd., London.
- Wolf, U. C. 1977. Location Problems on Tree Networks. Unpublished M.S. Thesis, Tel-Aviv University, Israel.
- Zipf, G. K. 1946. The P_1P_2/D Hypothesis: On the Inter-City Movements of Persons. American Sociological Review 11, 677-86.

APPENDIX

COMPUTER IMPLEMENTATION OF THE SINGLE-FACILITY ALGORITHM

In this appendix we describe a Fortran implementation of the algorithm of Section 2.5. We first define the data structures, using the same variable names as appear in the Fortran code. We then present an abbreviated version of the computer program in an informal pseudocode, showing only those operations that are essential to understanding how the algorithm works. A listing of the Fortran code itself is available from the author on request.

We adopt the following notational conventions. Variable names are written with lower-case letters, procedure names with upper case. The parameters passed to a procedure are written in parentheses following the procedure name. Subscripted variables are written in the form "array(i)," where "array(*)" denotes the entire array. Assignment statements such as "array(*) = constant" or "array(*) = array2(*)" assign a constant or array2(i), respectively, to array(i) for each i. The notation "array(n1:n2)" indicates that space is reserved for subscripts n1 through n2. In the pseudocode, material enclosed in slashes (/) is commentary.

The principal data structures are as follows. For each node i and arc j , $w(i)$ is the node weight and $b(j)$ the arc length. The network is read in the form of an arc list, in which each line indicates two node numbers, an arc length, and the weights of the two nodes. For each node i , arcs incident to i are stored as a linked list with header array $nptr(*)$ and pointer array $iptr(*)$. That is, for each i , $j_1 = nptr(i)$ is the number of a first arc incident to i , $j_2 = iptr(j_1)$ a second arc incident to i , $j_3 = iptr(j_2)$ a third arc, and so on. Although in the present implementation the arcs are considered undirected, the origin (listed first on each line of the arc list) is distinguished from the destination node of an arc, as follows. Each arc (i,k) is given two numbers: one number j , which appears in the list for the origin node i and for which $k = \text{node}(j)$, and a second number j' , which appears in the list for node k and for which $k = -\text{node}(j')$. For efficiency, all the arcs of which i is the origin node appear first in the list. If j is the last arc in a list, then $iptr(j) = 0$. For convenience, we will refer to the arc list for node i as AL_i .

The numbers of the treelike segments of each arc j are stored in a linked list we may call SL_j , with header array $nsptr(*)$ and pointer array $isptr(*)$. This permits easy addition and removal of segments. The distance from the origin node to the nearer boundary of segment k is $off(1,k)$, and to the further boundary is $off(2,k)$. When a segment of a node j is first identified, it is assigned a number k and added to list SL_j . If a relative minimum occurs at point x on k , the distance between x and the origin node of j is stored in $xsav(k)$ and the relative minimum in $zsav(k)$, and $ifseg(k)$ is set to true. If x is not on k , segment k is removed from SL_j .

Other important data structures include an array $d(*,*)$ of shortest distances between four nodes and the other nodes, an array $label(*,*)$ showing predecessors in the corresponding shortest path trees, and an array $dr(*)$ for shortest distances from a specified point to all nodes in a fixed shortest-path tree (see procedure TSHIFT).

The procedures are diagrammed in Fig. A-1, in which the higher procedures call the lower ones to which they are connected. The user must provide procedures F to evaluate the cost function and FMIN to solve the convex line search problem (2.7). If the user wishes to employ Technique 1, he must supply procedure DELF to calculate the gradient of the cost function.

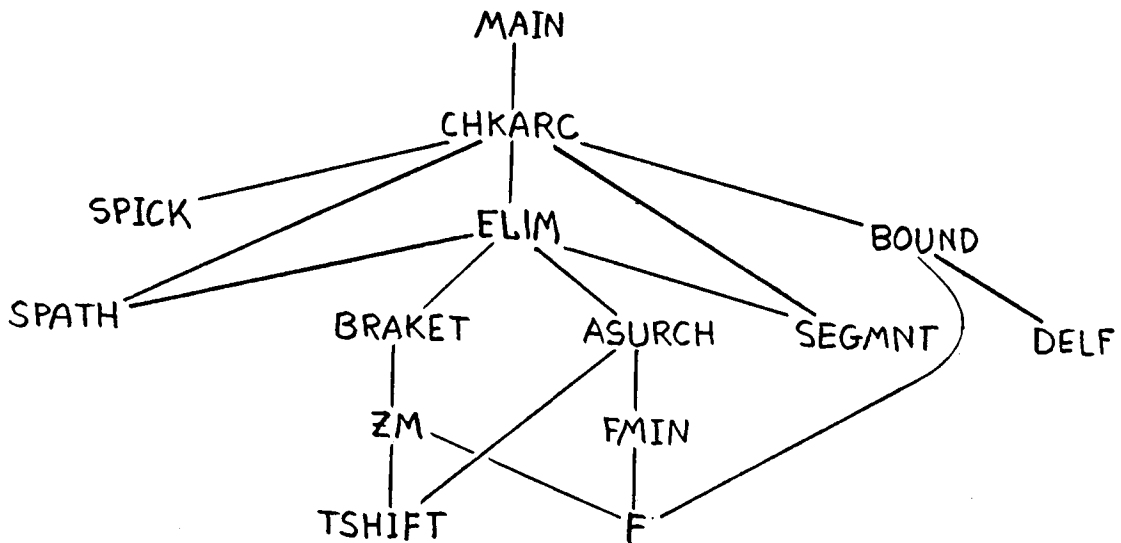


Fig. A-1. Calling relationships among procedures of the single-facility algorithm.

Briefly, the functions of the other procedures are as follows. MAIN picks the next arc A for examination and prints the solution when it is found. CHKARC calls BOUND in an attempt to eliminate A with upper bounding and, failing that, calls SPICK to pick a treelike segment S on A for examination. ELIM then tries to eliminate S. It first calls BRACKET to identify the arc B of the shortest-path tree $T(S)$ rooted at S that contains the optimal location x on $T(S)$. It then calls ASURCH to pinpoint the position of x on B. If x lies in S, ELIM records it as a

relative minimum. Otherwise, segments between x and S may be eliminated; ELIM eliminates segments on B , and BRAKET eliminates segments on A as well as entire arcs between A and B . SPATH constructs shortest-path trees in order n^2 time whenever they are needed, and SEGMNT locates the treelike segments of an arc. ZM calls F to evaluate z . Since ASURCH needs distances between points of $T(S)$ as measured in $T(S)$, it calls TSHIFT to compute these in order n time on the basis of known distances between nodes and the root of $T(S)$.

The pseudocode is as follows.

MAIN PROCEDURE

```

/The important global variables are as follows./
declare global
ntop, /highest node number used/
m/ /number of arcs/
nmax, nmax, lsmax, /maximum number of nodes, arcs, treelike segments/
w(1:nmax), b(1:mmax), /node weights, arc lengths/
nptr(1:nmax), iptr(1:2*mmax), /arc lists/
nsptr(1:mmax), isptr(1:lsmax), /segment lists/
node(1:2*mmax), /list of "other" node for each arc/
off(1:2, 1:lsmax), /locations of segments/
ifarc(1:mmax), ifseg(1:lsmax), /flags arcs, segments that have been
                                examined/
d(1:4, 1:nmax), label(1:4, 1:nmax), /shortest-path tree distances,
                                predecessors/
xsav(1:lsmax), zsav(1:lsmax), /relative minima: points and objective
                                function values/
alpha, /parameter for cost function, set by user/
dscore(1:nmax), /array for node choice heuristic (see below)/
zzmin; /lowest relative minimum obtained so far/

/Initialize variables./

dscore(*) = 0; ifarc(*) = false; ifseg(*) = false;
zsav(*) = infinity; zzmin = infinity;

/Call a routine that reads the arc list and sets up arrays nptr(*),
iptr(*), node(*), w(*), b(*)./
call READ;

/Start choosing arcs for examination. The chosen arc  $j$  is incident to a
node  $i$  for which  $dscore(j)$  is a maximum, where  $dscore(j)$  is generated by
a heuristic that favors nodes that are nearer the "periphery" of the
network. In the current implementation,  $dscore(i)$  is the distance
between  $i$  and a point near the optimal location on the shortest-path
tree rooted at the last treelike segment examined. Nodes  $i$  are picked

```

and $dscore(i)$'s are negated until unexamined segments are found, at which point the $dscore(i)$'s are reset to their positive values and the process repeats./

```
do forever;
  find node i for which dscore(i) is max;
  if dscore(i) < 0 then stop;
  dscore(i) = - dscore(i);
  find an arc j in ALi for which ifarc(j) = false;
  if there is such a j, then do the following;
    n1 = i; n2 = iabs(node(j)); ifpath = true;
    /Each call of CHKARC examines a segment on arc j; the segment
    examined is jseg. If jseg = 0, no segments remain on j./

    do until jseg = 0;
      call CHKARC(n1, n2, j, jseg, ifpath);
      if jseg ≠ 0 then dscore(*) = abs(dscore(*));
    end;
  end;
end;
```

/The relative minima now lie on the remaining treelike segments in the lists SL_i and are indexed by the segment numbers k. The value of the global minimum is stored as $zzmin$. Thus a list of global and local minima can be generated as follows. Each line of output is of the form

$$(i, i') [t_1, t_2] t h(t) \text{ flag ,}$$

where (i, i') is the arc containing a relative minimum x , which lies at a distance t from i and lies in a treelike segment occupying interval $[t_1, t_2]$ on (i, i') , and where $h(t)$ is $z(x)$. Each line is flagged to indicate a global or merely a local minimum./

```
do i = 1 to ntop;
  j = nptr(i);
  do until j = 0 or node(j) < 0;
    k = nsptr(j);
    do until k = 0;
      flag = 'local minimum';
      if zsav(k) ≤ zzmin then flag = 'global minimum';
      print i, node(j), off(1,k), off(2,k), xxav(k), zsav(k),
        flag;
    end;
  end;
end loop;

end MAIN PROCEDURE;
```

```
procedure CHKARC (n1, n2, j, jseg, ifpath);
```

```
  /Examine a segment on arc j = (n1,n2).
```

```
  jseg = 0 is returned to indicate that no segment on j remains to be examined.
```

```
  jseg > 0 is returned to indicate which segment on j was examined.
```

```
  ifpath = true indicates that shortest-path trees rooted at n1 and n2 are to be computed in this procedure call; it is returned false./
```

```
  jseg = 0; if ifarc(j) then return;
```

```
  /Compute shortest-path trees rooted at n1 and n2, and try to eliminate j using the upper bound zzmin./
```

```
  if ifpath then do;
```

```
    call SPATH(1,n1); call SPATH(2,n2);
```

```
    if using Technique 1 then call BOUND(j);
```

```
    if ifarc(j) = true then return;
```

```
  end;
```

```
  /Find treelike segments on arc j if not already found./
```

```
  if ifseg(j) = false then call SEGMNT(n1,n2,j,1,2) and set ifseg(j) = true;
```

```
  /Call a heuristic procedure that picks a segment jseg on arc j for examination./
```

```
  call SPICK(j,jseg); if jseg = 0 then return;
```

```
  /Try to eliminate segment jseg./
```

```
  call ELIM(n1,n2,j,jseg,ifpath);
```

```
  return;
```

```
end CHKARC;
```

```
Procedure ELIM(n1,n2,j,jseg,ifpath);
```

```
  /Try to eliminate segment jseg on arc j = (n1,n2). If jseg contains a relative minimum, its location is stored as xsav(jseg) and its value as zsav(jseg); otherwise, jseg and possibly other segments are eliminated. For convenience let T(jseg) be a shortest-path tree rooted at segment jseg on arc j, and let x be a point that minimizes z on T(jseg). Then arc jj = (m1,m2) is the arc containing x; jjseg is the segment on jj that contains x; and smid is the location on j of the center of jseg./
```

```

smid = (off(1,jseg) + off(2,jseg))/2;

/Find the arc jj = (m1,m2) containing x./

jj = j; call BRAKET(n1,n2,jj,jseg,m1,m2,idir);

/Find where x lies on arc jj./

Call ASURCH(m1,m2,jj,j,smid,xmin,zmin);

/If x lies on jseg, record it as a relative minimum./

if j = jj and off(1,jseg) ≤ xmin ≤ off(2,jseg) then do;
    xsav(jseg) = xmin; zsav(jseg) = zmin; zzmin = min(zzmin,zmin);
end;

/Otherwise eliminate all segments on arc jj between jseg and x. Begin
by setting idir in the case j = jj./

if j = jj then do;
    if xmin < off(1,jseg) then idir = -1; else idir = 1;
end;
if ifarc(jj) = true then return;

/If not using Technique 2, eliminate jseg only./

if not using Technique 2 then do;
    remove jseg from SLj; return;
end;

/Find shortest-path trees at m1 and m2 if segments have not been
generated for jj; trees are stored in d(3,*) and d(4,*) so as not to
overwrite trees rooted at n1 and n2. Try to eliminate jj using upper
bounding. If jj remains, generate its segments./

if ifseg(jj) = false then do;
    call SPATH(3,m1); call SPATH(4,m2); ifpath = true; if using
    Technique 1 then do;
        call BOUND(jj); if ifarc(jj) then return;
    end;
    call SEGMNT(m1,m2,jj,3,4); ifseg(jj) = true;
end;

/Now eliminate segments./

if idir = 1 then do;
    if j ≠ jj then remove from SLjj all segments k for which xmin >
        off(2,k);
    else remove from SLj all segments k for which xmin > off(2,k) and
        off(1,k) ≥ off(1,jseg);

```

```

else do;
  if j  $\neq$  jj then remove from SLjj all segments k for which  $x_{\min} < \text{off}(1,k)$ ;
  else remove from SLj all segments k for which  $x_{\min} < \text{off}(1,k)$  and  $\text{off}(1,k) \leq \text{off}(1,j\text{seg})$ ;
end;

end ELIM:

```

Procedure BRAKET(n1,n2,j,jseg,m1,m2,jj,idir);

/Find the arc jj = (m1,m2) containing the optimal location x on the shortest-path tree T(jseg) rooted at segment jseg on arc j = (n1,n2). Also,

```

idir = 0  if j = jj;
         1  if j lies on the m1 side of jj in T(jseg);
        -1  if j lies on the m2 side of jj in T(jseg);
idir2 = 0  if j = jj;
         -1  if jj lies on the n1 side of j in T(jseg);
         1  if jj lies on the n2 side of j in T(jseg).
eps = small offset used to compute sign of derivative of z./

```

```

smid = (off(1,jseg) + off(2,jseg))/2;

```

/Say that z is increasing on arc (n1,n2) if it increases as one moves from n1 toward n2. Then if z is decreasing at n1 on (n1,n2) and increasing at n2, (n1,n2) contains x. In this case there is nothing to do./

```

idir = 0; idir2 = 0; m1 = n1; m2 = n2; jj = j;

```

/Function subroutine ZM evaluates function z at a given point; slopel and slope2 are proportional to slopes of z at n1 and n2./

```

slopel = ZM(n1,n2,j,eps,j,smid) - ZM(n1,n2,j,0,j,smid);
slope2 = ZM(n1,n2,j,b(j),j,smid) - ZM(n1,n2,j,b(j)-eps,j,smid);
if slopel  $\leq$  0 and slope2  $\geq$  0 then return;

```

/If x cannot be bracketed on (n1,n2) by examining the slopes of z at n1 and n2, start moving along T(jseg) in a direction for which z decreases. Let arc jj = (nn1,nn2) be the current arc, and let arc jjj, denoted (ntemp,nn1) or (nn2,ntemp) depending on the direction of motion, be a candidate for the next arc to be traversed. Let jold be the arc traversed before jj, with jold = 0 if there is no such arc./

```

nn1 = n1; nn2 = n2; jj = j;

```

/The following block of code applies if z decreases in the n1 direction (slopel > 0)./

if slopel > 0 then do;

idir2 = -1; jjj = nptr(nnl); jold = 0; slope = infinity;

/Find an arc (ntemp,nnl) along which z decreases as one moves away from nnl./

do until jjj = 0 or slope > 0;

ntemp = abs(node(jjj));

/Determine whether the shortest path from jseg to ntemp lies in T(n1) (ndx = 1) or in T(n2) (ndx = 2). Then skip (ntemp,nnl) if it is not part of T(jseg)./

ndx = 1;

if d(1,ntemp) + smid > d(2,ntemp) + b(j) - smid then ndx = 2;

if label (ndx,ntemp) = nn1 and ntemp ≠ nn2 then

slope = ZM(ntemp,nnl,jjj,b(jjj),j,smid)

-ZM(ntemp,nnl,jjj,b(jjj)-eps,j,smid);

if slope ≤ 0 then jjj = iptr(jjj);

end;

/If jj = 0 then x is bracketed on arc jj = (nnl,nn2), which is possibly arc j = (n1,n2)./

if jjj = 0 then do;

if node(jj) = nn1 then do;

m1 = nn2; m2 = nn1; idir = 1; end;

else do; m1 = nn1; m2 = nn2; idir = -1; end;

return;

end;

/Otherwise, the search continues. But first eliminate segments that were passed over. If jold = 0, eliminate segments of arc j lying between jseg and node n1; otherwise eliminate the entire arc jold./

if jold = 0 then remove from SL_j all segments k such that $off(1,k) \leq off(1,jseg)$;

else ifarc(jold) = true;

/Move on to another arc (ntemp,nnl)./

jj = smaller arc number for arc jjj;

jold = jj; nn2 = nn1; nn1 = ntemp;

end;

/The following block applies if z decreases in the n2 direction (slope2 < 0)./

```

else do;
  idir2 = -1; jjj = nptr(nnl); jold = 0; slope = infinity;
  do until jjj = 0 or slope < 0;
    ntemp = abs(node(jjj));
    ndx = 1;
    if d(1,ntemp) + smid > d(2,ntemp) + b(j) - smid then ndx = 2;
    if label (ndx,ntemp) = nn2 and ntemp ≠ nn1 then
      slope = ZM(nn2,ntemp,jjj,eps,jj,smid)
            -ZM(nn2,ntemp,jjj,0,jj,smid);
    if slope ≤ 0 then jjj = iptr(jjj);
  end;
  if jjj = 0 then do;
    if node(jj) = nn2 then do;
      m1 = nn1; m2 = nn2; idir = 1; end;
    else do; m1 = nn2; m2 = nn1; idir = -1; end;
    return;
  end;
  if jold = 0 then remove from SLj all segments k such that off(1,k)
    ≥ off(1,jseg);
  else ifarc(jold) = true;
    jj = smaller arc number for arc jjj;
    jold = jj; nn1 = nn2; nn2 = ntemp;
end;
return;

end BRAKET;

```

Procedure ASURCH(n1,n2,jj,j,t,xmin,zmin);

/Minimize z on arc jj = (n1,n2), where distances are measured on a shortest-path tree T(t) rooted at a distance t from the origin node of arc j. The minimizing point lies at a distance xmin from n1 on jj, and the value of z at xmin is zmin. The first task is to call a procedure that efficiently computes the distance dr(i) from n1 to each node i as measured in tree T(t)./

call TSHIFT(n1,n2,jj,0,j,t);

/Now call a user-supplied subroutine that finds the minimizing point./

call FMIN(ntop,nptr,dr,w,b(jj),xmin,zmin,alpha);
return;

end ASURCH;

```
Procedure SPATH(k,nroot);
```

```
/Find a tree of shortest paths rooted at node nroot to all nodes of the
network. The distance from nroot to node i is stored in d(k,i). The
node immediately preceding node i as one moves away from nroot in the
tree is stored in label(k,i). The algorithm used is the classical
Dijkstra algorithm [Dijkstra 1959]./
```

```
end SPATH;
```

```
Procedure SEGMNT(n1,n2,j,k1,k2);
```

```
/Locate treelike segments on arc j = (n1,n2), where distances from n1
and n2 to each node i are stored in d(k1,i) and d(k2,i), respectively.
The segments are listed in  $SL_j$  and their locations stored in off(*,*).
The quantities  $s_1, \dots, s_n$  (defined in Section 2.2) are stored locally
in offset(*). Begin by computing  $s_1, \dots, s_n$ ./
```

```
declare offset(1:lsmax);
```

```
k = 0;
```

```
do i = 1 to ntop;
```

```
    if label(1,i) > 0 then do;
```

```
        s = (d(k2,i) - d(k1,i) + b(j))/2;
```

```
        if 0 < s < b(j) then do;
```

```
            k = k + 1; offset(k) = s; and;
```

```
        end;
```

```
end;
```

```
sort array offset(*) in ascending order;
```

```
/Find distinct elements of offset(*) and record corresponding segments./
```

```
lim = k; let k be number of next segment; off(1,k) = 0;
```

```
do kk = 2 to lim;
```

```
    if offset(kk) > off(1,k) then do;
```

```
        off(2,k) = offset(kk); put k in  $SL_j$ ;
```

```
        let k be number of next segment;
```

```
        off(1,k) = offset(kk);
```

```
    end;
```

```
end;
```

```
off(2,k) = b(j); put k in  $SL_j$ ; return;
```

```
end SEGMNT;
```

Procedure SPICK(j,jseg);

/Pick the next segment jseg on arc j = (n1,n2) to be examined. The following heuristic rule is used. If the solution x on the previous shortest-path tree was on the n1 side of its root (idir2 = -1), pick a segment nearest the spot on (n1,n2) that is parm of the way ($0 \leq \text{parm} \leq 1$) from the middle of the arc to n2. If the previous x was on the n2 side of the root (idir = 1), do the same on the n1 side of the arc. Experience has shown that parm = 1 behaves poorly on most networks, and that parm = 0 tends to be nearly as good as the optimal choice of parm./

jseg = 0;
 choose a segment k in SL_j that minimizes
 $\text{off}(1,k) + \text{off}(2,k)^j - b(j)(1 - \text{idir2} \cdot \text{parm});$
 if SL_j is nonempty, let jseg = k;
 return;

end SPICK;

Procedure TSHIFT(n1,n2,jj,tt,j,t);

/This procedure computes distances from a given "reference point" to all nodes, as measured on a given shortest-path tree. Let the reference point lie on arc jj = (n1,n2) at a distance tt from origin node n1. Let the shortest-path tree T(t) be one rooted at a distance t from the origin node of arc j. Naturally, arc jj should belong to T(t). The distance dr(i) from each node i to the reference point, as measured on T(t), is computed; dr(i) is negated if a shortest path from the root of T(t) to the reference point passes through n2 rather than n1; this is useful in procedure ZM. The distances from the root to each i are already implicit in d(1,*) and d(2,*); the algorithm shifts these distance measurements on T(t) so as to compute dr(*) in order n time./

dr(*) = infinity;

/Let arc j = (nn1,nn2). We first compute the distances dij from the root to the reference point by way of nni and nj, where i and j are 1 or 2 (Fig. A-2). Then we set ndx = 1 if a shortest path P from the root to the reference point runs through nn2, with ndx = 2 otherwise. Using these data we characterize P as running from the reference point a distance tt0 to n0, from n0 a distance d(ndx,n0) to nn1 or nn2, and from here a distance t0 to the root./

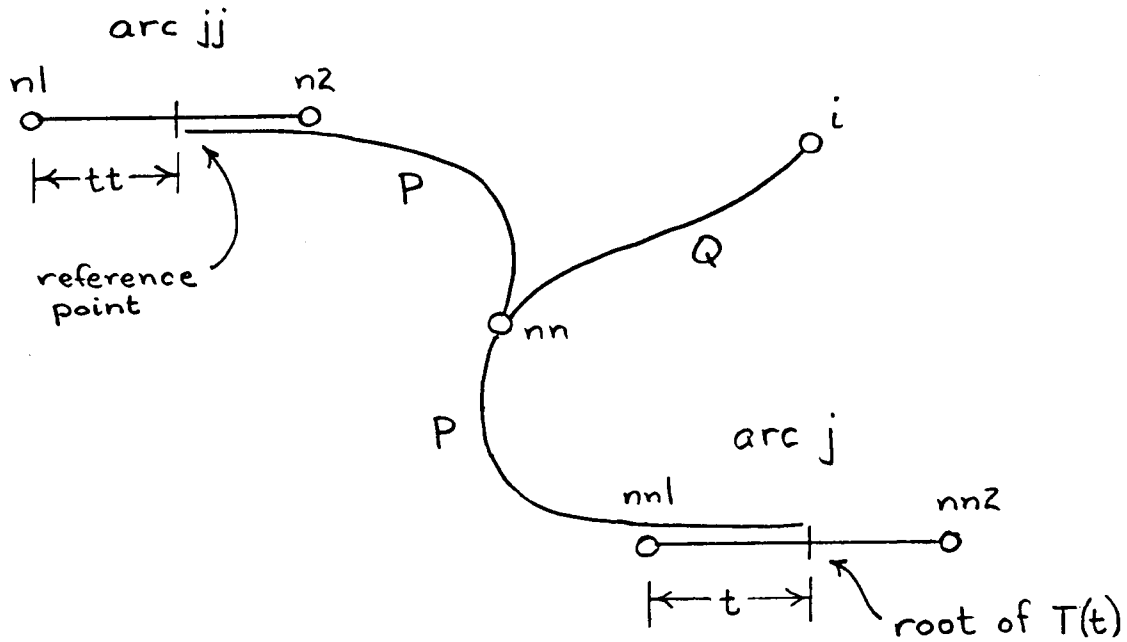


Fig. A-2. Illustration for procedure TSHIFT. Here, $n0 = n1$, $t0 = t$, $nn0 = nn2$, $tt0 = b(jj) - tt$, $ndx = 2$.

```

d11 = d(1,n1) + t + tt; d12 = d(1,n2) + t + b(jj) - tt;
d21 = d(2,n1) + b(j) - t + tt; d22 = d(2,n2) + b(j) - t + b(jj) - tt;
ndx = 1; if min(d11,d12) > min(d21,d22) then ndx = 2;
if j ≠ jj then do;
  if tt ≠ 0 then do;
    if min(d11,d21) ≤ min(d12,d22) then do;
      n0 = n1; nn0 = n2; tt0 = tt; end;
    else do; n0 = n2; nn0 = n1; tt0 = b(jj) - tt; end;
  end;
  else do;
    if d(ndx,n1) ≤ d(ndx,n2) then do;
      n0 = n1; nn0 = n2; tt0 = tt; end;
    else do; n0 = n2; nn0 = n1; tt0 = b(jj) - tt; end;
  end;
end;
else do; ndx = 1; n0 = n1; nn0 = n2; tt0 = tt; end;

```

/Next compute the distance along P from the reference point to every node on P./

```

nn = n0;
do until nn = 0;
    dr(nn) = d(ndx,n0) + tt0 - d(ndx,nn);
    if n0 ≠ n1 then dr(nn) = -dr(nn); nroot = nn; nn = label(ndx,nn);
end;

/Now compute the distance dr(i) for all other nodes i./

do i = 1 to ntop;
    if nptr(i) > 0 then do;

        /First set nndx = 1 if the path in T(t) from i to the root
        passes through nn1, nndx = 2 if it passes through nn2./

        nndx = 1;
        if d(1,i) + t > d(2,i) + b(j) - t then nndx = 2;

        /Follow the path Q from i to the root in T(t) until reaching a
        node nn for which dr(nn) has been calculated (Case A below).
        Then dr(i) is the length of Q plus d(nn). If no such nn is
        reached (Case B), then i is the first node to be processed
        that is on the side of the root opposite the reference point./

        nn = i; nnold = 0
        do until nn = 0 or dr(nn) ≠ infinity;
            nnold = nn; nn = label(nndx,nn);
        end;

        /Case A. We have two subcases: the last arc of Q is j (Case
        A1) or is another one (Case A2)./

        if nn = 0 then do;
            if nnold = nn0 then do; /Case A1./
                dr(i) = d(nndx,i) - d(nndx,nn) - abs(dr(nn));
                if n0 = n1 then dr(i) = -dr(i);
            end;
            else do; /Case A2./
                dr(i) = d(nndx,i) - d(nndx,nn) + abs(dr(nn));
                if n0 ≠ n1 then dr(i) = -dr(i);
                if dr(nn) < 0 then dr(i) = -abs(dr(i));
            end;
        end;

        end;

        /Case B. Here we distinguish cases in which j = jj or j ≠
        jj./

        else do;
            if j = jj then do;
                dr(i) = d(nndx,i) + b(j) - tt0;
                dr(i) = -dr(i)
            end;

```

```

        else do;
            dr(i) = d(nndx,i) + abs(dr(nroot)) + b(j);
            if n0 ≠ n1 then dr(i) = -dr(i)
        end;
    end;
end;
end loop;
return;

end TSHIFT;

```

Procedure BOUND(j);

/Try to eliminate arc j using the current upper bound zzmin. Let $h(t) = z(y)$ where y lies a distance t from the origin node of arc j. Then by convexity we know from Section 2.3 that z cannot achieve on arc j a value lower than

$$fmin = h(0) - [b(j) \cdot aa(2) - h(0) + h(b(j))] \cdot \frac{aa(1)}{aa(1) + aa(2)} ,$$

where $h(t) = z(y)$ when y lies a distance t from the origin node of arc j,

$$aa(1) = \sum_{i=1}^n g_i(0), \quad aa(2) = \sum_{i=1}^n g_i(b(j)) ,$$

and

$$g_i(t) = \begin{cases} |h_i(t)| & \text{if } d(2,i) < d(1,i) + b(j) , \\ -h_i(t) & \text{otherwise} , \end{cases}$$

and $h_i(t) = \partial f / \partial d_i$ evaluated at $d(1,i) + t$.

If $aa(1) \leq 0$ or $aa(2) \leq 0$ we set fmin to the minimum of $h(0)$ and $h(b(j))$./

```

dimension aa(1:2), ff(1:2), k1(1:2), k2(1:2);
k1(*) = 1,2; k2(*) = 2,1;

```

/First compute aa(*)./

```

do k = 1 to 2;
  aa(k) = 0; dr(*) = d(k,*);
  ff(k) = F(ntop,nptr,dr,w,alpha);
  call DELF(ntop,nptr,dr,w,df,alpha);
  do i = 1 to ntop;
    if d(k2(k),i) = d(k1(k),i) + b(j) then
      aa(k) = aa(k) - df(i);
    else aa(k) = aa(k) + abs(df(i));
  end;
end loop;

/Compute fmin, and eliminate arc j if possible./

if aa(1) + aa(2) > 0 then do;
  fmin = ff(1) + (b(j)*aa(2) - ff(1) + ff(2))
    *aa(1)/(aa(1) + aa(2));
  fmin = min(fmin,ff(1)); fmin = min(fmin,ff(2));
end;
else fmin = min(ff(1),ff(2));
if fmin ≥ zzmin then ifarc(j) = true;
return;

end BOUND:

```

Function procedure F(n,nptr,d,w,alpha);

/This is a user-supplied function subroutine that calculates the value of the cost function f./

```

declare
n,      /largest node number/
nptr(1:n),      /nptr(i) = 0 indicates that node i does not appear in the
                network./
d(1:n),      /vector of arguments of f; d(i) is the distance from the point
                at which f is to be evaluated to node i. Take absolute
                values, since some d(i)'s are negative./
w(1:n),      /w(i) is the weight on node i./
alpha;      /parameter that may be used in the cost function, as in the
                cent-dian function./

```

end F;

```
Procedure FMIN(n,nptr,d,w,b,xmin,zmin,alpha);
```

```
/This is a user-supplied procedure that minimizes z(y) on a treelike
segment of some given arc./
```

```
declare
```

```
n, nptr(1:n), w(1:n), alpha, /same as in procedure F./
```

```
d(1:n), /d(i) is the distance from a fixed node on the given arc,
which we may designate at the "first node," to node i. d(i)
is negated if a shortest path from the first node to i passes
through the other node of the given arc./
```

```
b, /length of the given arc/
```

```
xmin, /distance of minimizing point from the first node. If xmin
lies inside the treelike segment in which the distances d(*)
are valid, then it is a relative minimum on the network;
otherwise it need not be./
```

```
zmin; /value of z at xmin/
```

```
/This procedure must solve the line search problem,
```

$$\min_{0 \leq t \leq b} f(t + d(1), \dots, t + d(n)) . /$$

```
end FMIN;
```

```
Procedure DELF(n,nptr,d,w,df,alpha);
```

```
/This routine is supplied by the user if upper bounding is used. It
calculates the gradient of the cost function f at the point (d(1), ...,
d(n))./
```

```
declare
```

```
n,nptr(1:n), d(1:n), w(1:n), alpha, /as in procedure F/
```

```
df(1:n); /gradient vector, returned/
```

```
/If f is the weighted maximum function, return df(i) = w(i) for some i
for which abs(d(i))*w(i) = max(abs(d(j))*w(j)), and return df(j) = 0 for
j ≠ i./
```

```
end DELF;
```

VITA

John Norman Hooker, Jr., was born in Maryville, Tennessee, on May 15, 1949. He attended public schools in East Tennessee and was graduated from Bearden High School, Knoxville, in June 1967. He received an A.B. degree in mathematics from Princeton University in June 1971 and matriculated at Vanderbilt University the following fall to study philosophy. He received a Ph.D. from that institution in 1974 and, in the period 1974-77, taught philosophy at Mary Baldwin College, Vanderbilt University, and Memphis State University. He then entered The University of Tennessee and received an M.S. degree in management science in March 1979. In January of that year he had joined the research staff of Oak Ridge National Laboratory, where his work concerned transportation modeling. His employment there, twice interrupted when he assumed visiting faculty positions in The University of Tennessee Management Science Program, ran through July 1984. Upon receiving a Ph.D. degree in that program the following month, he became Assistant Professor of Operations Research at Carnegie-Mellon University.

The author's papers have appeared in Philosophical Studies, Logique et analyse, Kant-Studien, The Southwestern Journal of Philosophy, Transportation Science, Mathematics and Computers in Simulation and other journals. He is a member of the Operations Research Society of America and the American Philosophical Association.