University of London

Imperial College of Science and Technology

Department of Management Science

THE OPTIMAL LOCATION OF FACILITIES ON A NETWORK

by

ROBERTO DIÉGUEZ GALVÃO, B.Sc., M.Sc.

A thesis submitted for the
Degree of Doctor of Philosophy and
the Diploma of Imperial College

May 1977

Abstract

Problems of optimally locating facilities on networks fall within two main categories, namely minisum and minimax location problems. This thesis studies the p-median problem, an uncapacitated minisum location problem that consists of locating a given number of facilities (say p) on a network, so that the sum of shortest distances from each of the nodes of the network to its nearest facility is minimized.

Two formulations of a linear programming (LP) relaxation of the problem are examined. A general formulation produces very large linear programmes, and is therefore unsuitable for use in large-scale networks. A decomposition formulation produces smaller LP's but often does not converge. The importance of this LP relaxation lies in the fact that it often produces integer solutions that are optimal solutions to the p-median problem itself.

Two lower bounds are then developed: a graph-theoretical bound, based on shortest spanning trees and arborescences, and a dual bound, based on the dual of the LP relaxation of the problem. The latter proved to be a very good bound, and is used in the branch-and-bound algorithm developed in Chapter 5.

The algorithm of Chapter 5 is a direct tree search algorithm. It cascades through two lower bounds in a way designed to save computing time, and uses an upper bound to further reduce the size of the search. The computational results obtained through this algorithm represent a substantial advancement over existing exact solution procedures for the problem. It produces optimal solutions for networks of up to 30 vertices in less than 2 minutes in a CDC 7600 computer, for every possible value of p.

Finally, heuristic methods are investigated and tested in a number of problems. Heuristics based on λ-optimal substitution methods are described, and computational results are given for the particular cases of $\lambda = 1$ and $\lambda = 2$.

Acknowledgements

I would like to thank my supervisor, Dr. Nicos Christofides, of the Department of Management Science, Imperial College, for his advice and encouragement during the course of my work. I would also like to thank Professor S. Eilon, Head of the Department, and Dr. R.B. Flavell, lecturer in the same department, for their help in some phases of the research.

I am very grateful to Professor E.M.L. Beale, Department of Mathematics, Imperial College, for his advice on linear programming decomposition, and to Dr. A.W. Neebe, University of North Carolina, U.S.A., for having forwarded to me a computer code developed by him.

The financial support of the Brazilian Ministry of Education and Culture, for a period of four years, is gratefully acknowledged. Finally, I wish to thank Ms. Anne Usher, Anne de Sayrah and Susan Hayden for their excellent work in typing this thesis.

To my parents, without whose help and encouragement this thesis would not have been possible.

To Stuart, who was never able to complete his degree in Economics.

CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER ONE

INTRODUCTION

1.1 Network Location Problems

The problem of optimally locating facilities on a network falls within two main categories, namely minisum and minimax location problems. In minisum location problems the objective is to determine the location of a given number of facilities (say p), so that the sum of shortest distances from each of the network demand centres* to its nearest facility is minimized. The objective in minimax problems is to locate the p facilities so that the largest travel distance (or time) from any network demand centre to its nearest facility is minimized.

A related problem can be defined in the minimax category. It consists of finding the minimum number p of facilities (and their location), so that all demand centres in the network are within a critical distance δ from at least one of the facilities. Minimax problems appear in practice in the location of emergency facilities such as hospitals and fire stations.

This thesis studies a particular case of the uncapacitated minisum network location problem, often referred to as the p-median problem. The p-median problem consists of locating p facilities on a network, so that the sum of shortest distances from each of the nodes of the network to its nearest facility is minimized. There are no restrictions on the capacities of the facilities, and fixed costs are assumed not to vary with the location of the facility, thus not appearing in the problem's objective function.

Two theorems by Hakimi [48, 49] restrict the search of the optimal

* A network demand centre is defined here as being a site located either on the arcs or nodes of the network, from which demand for goods or services is generated.

p-median to the nodes of the network, i.e. it can be shown that the search for the optimal p points need not consider points on links other than the two ends. If n is the number of nodes in the network, the p-median problem has a total of $\binom{n}{p}$ feasible solutions, and solution by complete enumeration is not feasible even for problems of moderate size. The problem of finding the optimal p-median of a network can be made slightly more general by associating with each node x_i a weight v_i, in which case the objective function to be minimized becomes the sum of weighted distances.

The p-median problem appears in practice in a variety of forms: the location of switching centres in telephone networks, substations in electric power networks, supply depots in a road network, schools in a rural area. Assume, for example, that the population distribution of a given rural area is known. It is required that p primary schools be built in the area, so as to minimize the total distance travelled by the school children.

The school location problem can be represented by a network of n nodes, each node corresponding to one region in the area. Node weights can be used to represent the relative sizes of the school-age population of each of the regions. Existing roads between regions should link the corresponding nodes of the network. Given the length of each of the connecting roads, the problem is actually a p-median problem.

1.2 The p-median problem as a special case of facility location problems

The facility location problem consists of determining the site of one or more facilities (supply depots, schools, hospitals, etc.) to serve customers in a given geographical area. The selection of sites should be made in such a way that a well defined objective function is optimized, subject to constraints relevant to the problem.

Facility location problems have been classified in different ways by different authors. In an excellent review paper, Revelle et al. [89] attach great importance to the ownership of the facilities (private or public) and suggest different objective functions for the two cases. Eilon, Watson-Gandy and Christofides [27] have chosen to tie their classification to the approach used to solve the problem (Infinite Set Approach or Feasible Set Approach). They also enumerate some advantages and disadvantages of each of the two approaches.

In a comment on the paper by Revelle et al., Robers [91] proposes the following three-way classification:

A. Location in a Plane with Infinite Solution Space,

B. Location in a Plane with Finite Solution Space,

C. Location on a network.

Problems in category A are characterized by (1) an infinite solution space (facilities may be located anywhere in the plane), and (2) distance measurement according to a particular metric. The second type of problem is characterized by restricting the location of the facilities to a number of predetermined sites. Finally, location on a network is characterized by (1) a solution space consisting of points on the network, and (2) distance measurement along the network.

The p-median problem is now shown to be a special case of the uncapacitated facility location problem. The latter is a B problem in the classification given above. There are no restrictions on the permissible capacities of the facilities, and the objective function includes both fixed and variable costs.

The uncapacitated facility location problem can be formulated as a mixed-integer programming problem:

$$\text{Minimize} \qquad Z = \sum_{i \in I} F_i \, y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} \, \xi_{ij} \qquad (1.1)$$

Subject to

$$\sum_{i\in I} \xi_{ij} = 1, \; j\in J \tag{1.2}$$

$$1 \leqq \sum_{i\in I} y_i \leqq p \tag{1.3}$$

$$\xi_{ij} \leqq y_i, \; i\in I, \; j\in J \tag{1.4}$$

$$y_i \in \{0,1\}, \; i\in I \tag{1.5}$$

$$\xi_{ij} \geqq 0, \; i\in I, \; j\in J, \tag{1.6}$$

where:

I = (1,2,...,n) - set of feasible location sites for the facilities;

J = (1,2,...,m) - set of user (customer) locations;

F_i - Fixed cost associated with opening a facility at location $\underline{i}$;

c_{ij} - Cost of supplying customer $\underline{j}$ from facility $\underline{i}$;

ξ_{ij} - Fraction of the demand of customer $\underline{j}$ supplied from facility $\underline{i}$;

p - Maximum number of facilities that can be built;

$$y_i = \begin{cases} 1 \text{ if a facility is located at site } \underline{i} \\ 0 \text{ otherwise} \end{cases}$$

If (i) $I \equiv J$ = (1,2,...,n) coincide with the nodes of a network, (ii) $d_{ij} \equiv c_{ij}$ correspond to distances measured along the arcs of the network, (iii) it is decided that exactly $\underline{p}$ facilities must be built, and (iv) all fixed costs F_i are equal, Equations (1.1) to (1.6) become:

Minimize
$$Z = \sum_{i\in I} \sum_{j\in J} d_{ij} \xi_{ij} \tag{1.7}$$

Subject to

$$\sum_{i\in I} \xi_{ij} = 1, \; j\in J \tag{1.8}$$

$$\sum_{i\in I} \xi_{ii} = p \tag{1.9}$$

$$\xi_{ij} \leqq \xi_{ii}, \quad i \in I,\ j \in J\ ,\ i \neq j \tag{1.10}$$

$$\xi_{ij} = \begin{cases} 1 \text{ if customer } \underline{j} \text{ is allocated} \\ \quad \text{to facility } \underline{i} \\ 0 \text{ otherwise} \end{cases} \tag{1.11}$$

Equations (1.7) to (1.11) correspond to the integer programming formulation of the p-median problem (see chapter 3). In this formulation $\xi_{ii} = 1$ implies that a facility is located at site $\underline{i}$ and $\xi_{ii} = 0$ otherwise. Since it is not possible to have a fractional facility located at a site, $\xi_{ii} \in \{0,1\}$, $i \in I$. Furthermore, since there are no capacity restrictions and no economies of scale, no one destination will be supplied by more than one facility in the optimal solution. Hence, $\xi_{ij} \in \{0,1\}$, $i \in I$, $j \in J$.

The p-median problem is therefore a special case of the uncapacitated facility location problem. Note that since the number of facilities has been fixed a priori at exactly $\underline{p}$, and all fixed costs F_i are equal, the F_i need not be included in the formulation of the p-median problem.

1.3 Basic graph theory definitions

The basic graph theory concepts defined in the present section are the ones used throughout this thesis. The definitions given generally correspond to those of [12].

A graph (or network) G is a collection of vertices or nodes $x_1, x_2, \ldots, x_n$ (denoted by the set X), and a collection of lines $a_1, a_2, \ldots, a_m$ (denoted by the set A) joining all or some of the vertices. The graph G is then fully described and denoted by the doublet (X,A).

If the lines in A have a direction they are called arcs and the resulting graph is called a directed graph. If the lines have no orientation they are called links and the graph is nondirected. In cases where $G = (X,A)$ is a directed graph but it is desirable that the direction of the arcs in A be disregarded, the nondirected counterpart of G will be written as $\bar{G} = (X,\bar{A})$.

A path in a directed graph is any sequence of arcs where the final vertex of one arc is the initial vertex of the next arc. A simple path is a path which does not use the same arc more than once. An elementary path is a path which does not use the same vertex more than once. An elementary path is also simple, but the reverse is not necessarily true.

A graph is said to be arc-weighted if a cost (length,weight) c_{ij} is associated with every arc (x_i,x_j) of the graph. If a weight v_i is associated with every vertex x_i of a graph the resulting graph is then called vertex-weighted. A nonweighted graph is defined in this thesis as an arc-weighted graph whose vertices have all unit-weights, i.e. an arc-weighted graph for which $v_i = 1 \ \forall \ i$.

The number of arcs which have a vertex x_i as their initial vertex is called the outdegree of vertex x_i (call this outdegree $d_o(x_i)$). Similarly, the number of arcs which have x_i as their final vertex is called the indegree of vertex x_i $(d_t(x_i))$. For a nondirected graph the degree of a vertex x_i is equal to the number of links connected to x_i. When no confusion can arise it will be denoted simply by d_i.

A graph $G = (X,A)$ is said to be complete if, for every pair of vertices x_i and x_j in X, there exists a link (x_i,x_j) in $\bar{G} = (X,\bar{A})$, i.e. there must be at least one arc joining every pair of vertices. The complete nondirected graph of n vertices is denoted by K_n.

A graph $G = (X,A)$ is said to be symmetrical if, whenever an arc (x_i,x_j) is one of the arcs in the set A, the opposite arc (x_j,x_i) is also in A.

Matrices of a Graph

A convenient way of representing a graph $G = (X,A)$ algebraically is through its adjacency matrix. The adjacency matrix of G is denoted by $A = [a_{ij}]$ and is given by

$$a_{ij} = 1 \text{ if arc } (x_i,x_j) \text{ exists in } G$$
$$a_{ij} = 0 \text{ if arc } (x_i,x_j) \text{ does not exist in } G.$$

If a cost c_{ij} is associated with every arc (x_i,x_j) of the graph, it is possible to calculate the shortest path between all pairs of vertices of the graph [34, 83]. A matrix can then be formed with the corresponding shortest distances $d(x_i,x_j)$. The matrix $D(G) = [d(x_i,x_j)]$ is called the distance matrix of the graph.

When a weight v_i is associated with every vertex x_i of a graph, this graph must be transformed into a complete graph before a corresponding p-median problem can be solved. Any graph can be transformed into a complete graph through the computation of its distance matrix. In the case of vertex-weighted graphs, the computation of the distance matrix must be followed by the multiplication of each element of every row or column by the appropriate vertex weight*. The resulting weighted matrix can be then represented by a complete symmetrical graph. The arcs of this graph represent the weighted lengths of the corresponding shortest paths.

1.4 Outline of the thesis

This thesis is concerned with the p-median problem. The emphasis is on exact solution methods for the problem, although some heuristic

* In a network location problem for which the flow is directed into the facilities - as, for example, when the facilities are schools to which children must travel - the rows of the distance matrix must be weighted. If the reverse is true and the flow is from the facilities, the columns of the distance matrix must be weighted.

procedures are also investigated.

Chapter 2 is a literature survey. The survey covers the broader field of facility location problems, but its main part is dedicated to the p-median problem and related minisum and minimax network location problems.

In Chapter 3 two different formulations of the LP relaxation of the p-median problem are investigated. The general LP formulation produces very large linear programmes. This disadvantage is overcome by a recent LP decomposition formulation. The very degenerate nature of the decomposition formulation and the ensuing convergence problems are analysed and tested.

Chapter 4 is dedicated to lower bounds. Two new lower bounds are developed for the problem, namely the "graph-theoretical bound" and the "dual bound". Unlike other existing bounds, the graph-theoretical bound makes use of the graph-theoretical properties of the problem. The dual bound is based on the dual of the LP relaxation of the problem. The latter is a very good bound, a fact of decisive importance in branch-and-bound algorithms.

A direct tree search algorithm is the object of Chapter 5. The principles on which this algorithm is based are discussed, and the embedding of the bounds of Chapter 4 into the search is explained. Computational results for networks ranging from 10 to 30 vertices, and for a wide range of values of $\underline{p}$ are then given.

Heuristics are investigated in Chapter 6. The existing vertex substitution method of Teitz and Bart [98] is extended into a family of heuristics, the λ-optimal substitution heuristic methods. The particular cases of $\lambda = 1$ and $\lambda = 2$ are studied in detail. A simple vertex addition heuristic is introduced, and its use as a 'pre-processor' for the λ-optimal substitution methods is described. Computational results are given for the resulting heuristics.

The main contributions of this thesis to the field of optimal location of facilities on a network are:

(i) The development of new and "tight" lower bounds for the p-median problem;

(ii) their use in a direct tree search algorithm that represents a substantial advancement in the area of exact solution methods for the problem; and

(iii) the detailed investigation of the LP decomposition formulation of Garfinkel et al. [41], and in particular the problems arising from the large-scale degeneracy of this formulation.

The branch-and-bound algorithm produces optimal solutions for 30-vertex networks in less than 2 minutes in a CDC 7600 computer, for every possible value of p. It is both faster (in terms of time) and much more efficient (in terms of number of nodes) than other branch-and-bound algorithms available in the literature [30, 55]. While other exact solution methods [41, 78] may on occasion solve the problem for n = 30, these other methods cannot guarantee an optimal solution for every possible value of p, and may in fact fail on much smaller networks.

As for the LP decomposition of Garfinkel et al., the extensive testing of their algorithm carried out in this thesis has uncovered serious convergence problems, and shown that this lack of convergence is due to the very degenerate nature of the LP decomposition master problem. The hope that the embedding of this formulation into the branch-and-bound algorithm of Chapter 5 would overcome the convergence problems did not materialize, in spite of the large perturbations caused by the branching.

CHAPTER TWO

LITERATURE SURVEY

2.1 Introduction

Historically, contemporary location analysis started with Alfred Weber [101], who examined the location of a plant with the objective of minimizing transportation costs in relation to three points (two sources of raw materials and a single market). In one form or another, this is a very old problem in pure mathematics. It was considered as early as 1647 by Cavalieri. Fagnano, Tedenat, Heinen and Steiner made important contributions to its solution from the middle of the 18th to the middle of the 19th century [16].

It is not the objective of this survey to make a very detailed review of the literature on facility location. Detailed surveys are available elsewhere, such as those by Eilon et al. [27, Chapter 2], Revelle et al. [89] and Domschke [23]. The vastness of published work on location analysis is atested by the 226 papers listed by Francis and Goldstein [35] in their selective bibliography. Elshafei [28] gives a total of 82 references in a recent survey of facility location studies.

After a brief review of the general area of location analysis, the present survey concentrates on the p-median and related minisum network location problems.

2.2 Location with infinite solution space

The facility location problem with a minisum objective and infinite solution space is examined in depth in chapters 3 to 6 of [27]. Numeric-analytic heuristic methods can handle non-linear cost functions, provided that the cost functions are monotonic and continuous. In a more recent paper, Watson-Gandy and Eilon [100] investigate discontinuous delivery costs.

Multifacility location problems with infinite solution space are also called multisource Weber problems. They are divided into Euclidean and rectilinear distance problems, depending on the metric according to which distances are measured. In Euclidean distance problems distances are measured according to

$$d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 , \qquad (2.1)$$

where

d_{ij} - distance between points $\underline{i}$ and $\underline{j}$,

and (x_i, y_i) - coordinates of the i^{th} point in a rectangular system.

Rectilinear distance problems have their distances measured by

$$d_{ij} = |x_i - x_j| + |y_i - y_j| . \qquad (2.2)$$

The multisource Weber problem has been investigated by Cooper [16, 17, 18] , Kuenne and Soland [66, 67] and Morris [82] , among others. A related problem, that of locating new facilities in relation to existing ones has been the object of several papers, by Cabot, Francis and Stary [11] , Rao [88] and Juel and Love [56] .

In its simplest form, the Weber problem involves $\underline{m}$ customers with known location on a plane, the location of customer $\underline{j}$ being determined by a pair (x_j, y_j) of cartesian coordinates. The problem is to determine the coordinates (x_i, y_i) of each of a given number $\underline{p}$ of facilities to serve the $\underline{m}$ customers, so as to minimize the following cost function:

$$C = \sum_{i=1}^{p} \sum_{j=1}^{m} \xi_{ij} v_j d_{ij} , \qquad (2.3)$$

where

v_j - weighting factor related to customer j.

d_{ij} - distance between facility i and customer j, given by:

Equation (2.1) [Euclidean metric] , or

Equation (2.2) [Rectilinear metric] .

$$\xi_{ij} = \begin{cases} 1 \text{ if customer } j \text{ is served from facility } i \\ 0 \text{ otherwise} \end{cases}$$

If there are no capacity or other constraints, the solution to the above problem can be found through partial differentials with respect to x_i and y_i:

$$\frac{\partial C}{\partial x_i} = \sum_{j=1}^{m} [\xi_{ij} v_j \ (x_i - x_j)/d_{ij}] = 0, \quad i = 1,\ldots,p \qquad (2.4)$$

$$\frac{\partial C}{\partial y_i} = \sum_{j=1}^{m} [\xi_{ij} v_j \ (y_i - y_j)/d_{ij}] = 0, \quad i = 1,\ldots,p \qquad (2.5)$$

If equations (2.4) and (2.5) are solved for x_i and y_i it follows that

$$x_i = \sum_{j=1}^{m} (\xi_{ij} v_j x_j/d_{ij}) \ / \ \sum_{j=1}^{m} (\xi_{ij} v_j/d_{ij}), \ i = 1,\ldots,p \qquad (2.6)$$

$$y_i = \sum_{j=1}^{m} (\xi_{ij} v_j y_j/d_{ij}) \ / \ \sum_{j=1}^{m} (\xi_{ij} v_j/d_{ij}), \ i = 1,\ldots,p \qquad (2.7)$$

These equations can be solved iteratively, as shown by Eilon et al. [27] and Cooper [16] . Let the superscript k indicate the iteration parameter. The iteration equations for x_i and y_i are simply [16] :

$$x_i^{k+1} = \sum_{j=1}^{m} (\xi_{ij} v_j x_j/d_{ij}^k) \ / \ \sum_{j=1}^{m} (\xi_{ij} v_j/d_{ij}^k), \quad i = 1,\ldots,p \qquad (2.8)$$

$$y_i^{k+1} = \sum_{j=1}^{m} (\xi_{ij} v_j y_j/d_{ij}^k) \ / \ \sum_{j=1}^{m} (\xi_{ij} v_j/d_{ij}^k), \quad i = 1,\ldots,p \qquad (2.9)$$

After each iteration the customers are reallocated to the relocated facilities, and the ξ_{ij}'s are modified prior to the next iteration.

It has been shown by Palermo [85] , Kuhn and Kuenne [70] and Haley [51] , that Equation (2.3) is convergent in the case of a single facility (p=1): the cost function being convex, it has a single unique optimal solution. In the general case ($p > 1$) Equation (2.3) has multiple local minima and the iterative scheme of Equations (2.8) and (2.9) only converges to a local minimum.

2.3 Location with finite solution space

A simplified definition of the facility location problem with finite solution space is as follows. Given a number of demand points for a certain product, each with a demand D_j, and a number of alternative sites where facilities may be built to satisfy these demands, determine where the facilities should be placed, and which demand points are to be served by each of the facilities [87, 89] . There may or may not be restrictions on the size (capacity) of the facilities. The objective is to minimize the sum of the fixed costs of the facilities plus the variable transportation costs.

When there are restrictions on the size of the facilities the problem is usually called the capacitated facilities location problem. If these restrictions do not exist, the problem is known as the uncapacitated (or the simple) facility location problem [61, 97] . A general formulation for the uncapacitated facility location problem was given in Section 1.2. The general case in which there are restrictions on the size of the facilities can be formulated as [62] :

Minimize

$$Z = \sum_{i \in I} F_i \, y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} \, X_{ij} \qquad (2.10)$$

Subject to

$$\sum_{i \in I} X_{ij} \geqq D_j, \quad j \in J \tag{2.11}$$

$$\sum_{j \in J} X_{ij} \leqq S_i y_i, \quad i \in I \tag{2.12}$$

$$y_i \in \{0,1\}, \quad i \in I \tag{2.13}$$

$$X_{ij} \geqq 0, \quad i \in I,\ j \in J \tag{2.14}$$

where

X_{ij} - Amount supplied to customer _j_ from facility i;

D_j - Demand at area _j_;

S_i - Capacity of facility _i_.

The cost functions included in the objective function can be made more general than the one shown in Equation (2.10). Instead of the fixed costs associated with opening and operating a facility, and linear transportation costs, it may be necessary, in the case where the facility is a warehouse, to consider variable warehousing and delivery costs which are nonlinear [4, 5, 25, 31, 33, 65].

Perhaps the first algorithm to guarantee an optimal solution for the uncapacitated case was the one by Efroymson and Ray [25]. They assume that the fixed cost F_i is a single fixed charge. Their method can be also extended to include the case in which F_i is concave and consists of several linear segments.

Efroymson and Ray utilize a tree search algorithm.* They use a linear programming formulation that can be solved by inspection to resolve the subproblems at the nodes of the tree.

* Tree search or branch-and-bound algorithms are examined in more detail in Chapter Five. A good survey on branch-and-bound methods is provided by Lawler and Wood [71].

Spielberg [94, 95] has considered essentially the same problem, but his algorithms contain added features that speed up computation and can accommodate some side conditions. Khumawala [60] reports good computational results for this class of problem.

Algorithms for the capacitated case have been proposed by Davis and Ray [20], Gray [46], Marks [77], Sá [92] and Akinc and Khumawala [1]. Marks' model is more general in that he considers the facilities to be intermediate points between sources of product and the customers to whom these products are to be sent. In all algorithms of the above mentioned references for the capacitated case, transportation costs are assumed to be linear, and a fixed cost F_i is associated with facility $\underline{i}$. More general cost functions are considered by Soland [93] in a recent paper. All exact procedures mentioned above are branch-and-bound methods.

2.4 Location on networks

It was mentioned in Section 1.2 that location on networks is a special case of location in a plane with finite solution space. In network location problems the solution space is restricted to the arcs and vertices of the network, and distances must be measured along the arcs of the network.

Network location problems are characterized by the nature of the objective function to be optimized. In problems involving the location of emergency facilities, such as hospitals and fire stations, the objective is to locate a given number of facilities so that the largest travel distance (or time) from any network demand centre to its nearest facility is minimized. These are the minimax network location problems. In other cases, such as in the location of depots in a distribution network, a more appropriate objective is to minimize the total distance travelled. The latter are the minisum location problems, of which the p-median problem is a special case.

Minimax network location problems are briefly reviewed in Section 2.4.1. This is followed by a much more detailed review of minisum problems, with special emphasis given to the p-median problem.

2.4.1 Minimax location on networks

There are a large variety of minimax network location problems. In a thorough and comprehensive study of the subject, Handler [52] identified ten different models for such problems. In order to facilitate the identification of the several models, Handler proposed the following notation:

Facility Location Set		Demand Location Set		No. of Centres/ Max. Distance		Network Type	
$\left\{{N \atop P}\right\}$	/	$\left\{{N \atop P}\right\}$	/	$\left\{{p \atop \delta^{-1}}\right\}$	/	$\left\{{T \atop G}\right\}$	,

where N,P denote the node and point sets,*. T, G denote tree and cyclic graphs, and p,δ refer to the number of facilities and to the critical distance respectively. The critical distance is the maximal allowed distance between a demand centre and its nearest facility. The symbol "δ^{-1}" is used for inverse problems. In inverse problems what is sought is the determiniation of the minimal number of facilities (and their location), so that all demand centres are within a critical distance δ from at least one of the facilities (see Section 1.1).

The best known minimax network location problem is P/N/p/G. Demand centres are restricted to the vertices of the network, but facilities may be located either on vertices or on arcs connecting the vertices. This problem is known as the multi-centre problem or the

* The node set includes all vertices of the network. The point set comprises all points of the network, either vertices or points on arcs connecting the vertices.

absolute p-centre problem. In its formulation the number p of centres is fixed, and what is required is their location so that the maximal distance between any vertex of the network and its nearest facility is minimized.

The inverse of the absolute p-centre problem, denoted by $P/N/\delta^{-1}/G$, can be stated as follows: For a given critical distance δ, find the smallest number (and location) of facilities, so that all vertices of the network lie within this critical distance from at least one of the facilities. This problem is closely related to the absolute p-centre problem, and usually the same method can be used to solve both problems.

The State of the Art

The vertex centre (N/N/1/G) and the absolute centre (P/N/1/G) problems were introduced and solved by Hakimi [48]. Goldman [44] also presented an algorithm for P/N/1/G, but the algorithm does not guarantee an optimal solution. As a special case of his algorithm, Goldman derived an efficient algorithm for P/N/1/T.

The absolute p-centre problem (P/N/p/G) was also proposed by Hakimi [49]. Subsequently, solution algorithms for this problem were produced by Minieka [80] and Christofides and Viola [15]. An algorithm for N/N/p/G was given by Toregas, Swain, Revelle and Bergman [99]. All algorithms mentioned above involve repeated solutions of generalized set covering problems.

The work of Handler [52] represents a substantial advancement in the field of minimax network location problems. He developed better algorithms and studied problems that had previously received very little attention, such as P/P/p/G and N/P/p/G.

Minimax problems can be also defined for the more general case of location in a plane, with distances measured according to either the Euclidean or the rectilinear metric systems. These formulations have the characteristics of the models discussed in 2.2 and 2.3. They were the object of papers by Dearing and Francis [21], Elzinga, Hearn and Randolph [32] and Wesolowsky [103], among others.

2.4.2 p-Medians and absolute p-medians

Minisum network location problems can take several forms, depending on the costs included in the objective function and the form of the constraints. In these problems the optimal locations of the facilities are called the medians of the network. The difficulty in solving such problems is not due to variations in the objective function or to additional constraints, but is inherent in the pure median problem itself. This thesis is concerned with the pure p-median problem. The generalized p-median problem, however, is briefly discussed at the end of this section.

The Median and Multiple Medians

For a given network N = (X,A) it is possible to define two transmission numbers for every vertex $x_i \in X$. Let

$$\sigma_0(x_i) = \sum_{x_j \in X} v_j \, d(x_i,x_j) \ , \tag{2.15}$$

and

$$\sigma_t(x_i) = \sum_{x_j \in X} v_j \, d(x_j,x_i) \ , \tag{2.16}$$

where

$d(x_i,x_j)$ - shortest distance from vertex x_i to vertex x_j;

v_j - weight of vertex x_j.

The numbers $\sigma_0(x_i)$ and $\sigma_t(x_i)$ are called respectively the outtransmission and the intransmission of vertex x_i. The number $\sigma_0(x_i)$ is the sum of the entries of row x_i of a matrix obtained by multiplying every column j of the distance matrix D(N) = $[d\ (x_i,x_j)]$ by v_j; $\sigma_t(x_i)$ is the sum of the entries of column x_i of a matrix obtained by multiplying every row j of the distance matrix D(N) by v_j.

A vertex $\bar{x}_0$ for which

$$\sigma_0(\bar{x}_0) = \min_{x_i \in X} [\sigma_0(x_i)] \tag{2.17}$$

is called the outmedian of the network N, and a vertex $\bar{x}_t$ for which

$$\sigma_t(\bar{x}_t) = \min_{x_i \in X} [\sigma_t(x_i)] \tag{2.18}$$

is called the inmedian of N.

The single median can be readily generalized to the p-median. Let X_p be a subset of the set X of the vertices of the network N(X, A), and let X_p contain p vertices. Define:

$$d(X_p, x_j) = \min_{x_i \in X_p} [d(x_i, x_j)] , \tag{2.19}$$

and

$$d(x_j, X_p) = \min_{x_i \in X_p} [d(x_j, x_i)] . \tag{2.20}$$

If x_i' is the vertex of X_p which produces the minimum in equations (2.19) or (2.20), it can be said that vertex x_j is allocated to x_i'. The transmission members for the set X_p of vertices are then defined in ways analogous to those for a single vertex, i.e.

$$\sigma_0(X_p) = \sum_{x_j \in X} v_j \, d(X_p, x_j) , \tag{2.21}$$

and

$$\sigma_t(X_p) = \sum_{x_j \in X} v_j \, d(x_j, X_p) , \tag{2.22}$$

where $\sigma_0(X_p)$ and $\sigma_t(X_p)$ are the outtransmission and the intransmission of the set X_p of vertices.

A set $\bar{X}_{p0}$ for which

$$\sigma_0(\bar{X}_{p0}) = \min_{X_p \subseteq X} [\sigma_0(X_p)] \tag{2.23}$$

is called the p-outmedian of the network N, and similarly for the *p-inmedian*.

It is not computationally practical to use Equations (2.19) through (2.23) directly to find p-medians of networks of even moderate size. Hence the need to develop more practical methods for the computation of p-medians.

Absolute p-Medians

In order to simplify the discussion consider a nondirected network N, drop the suffices *O* and *t* and take the case of the 1-median first. The question arises as to whether there exists a point *y* on some link (not necessarily a vertex) of N so that the transmission

$$\sigma(y) = \sum_{x_j \in X} v_j \, d(y, x_j) \qquad (2.24)$$

is less than that of the median of N. The point $\bar{y}$ with the minimum $\sigma(y)$ would then be called the *absolute median* of N.

Goldstein [45] proved that an absolute median of a tree is always at a vertex of the tree. Hakimi [48] generalized Goldstein's result and proved that there is no point $\bar{y}$ with $\sigma(\bar{y}) < \sigma(\bar{x})$, i.e.

Theorem 2.1 - There exists at least one vertex x of N = (X, A) for which $\sigma(x) \leqq \sigma(y)$ for any arbitrary point *y* on N.

In a later paper, Hakimi [49] generalized Theorem 2.1 to the case of absolute p-medians:

Theorem 2.2 - There exists at least one subset $X_p \subset X$ containing *p* vertices, such that $\sigma(X_p) \leqq \sigma(Y_p)$ for any arbitrary set Y_p of *p* points on the links or vertices of the network N = (X, A).

The proofs of Theorems 2.1 and 2.2 are given in [48] and [49] respectively.

In view of Theorems 2.1 and 2.2 the search for optimal solutions to the p-median problem can be limited to the vertices of the network. As a consequence, in the p-median problem the location of both demand centres and facilities is restricted to the vertices of the network.

The Generalized p-Median

In the pure p-median problem the only costs to be minimized are variable costs associated with distances between vertices. The p-median problem can be made more general if fixed costs F_i are associated with the vertices of the network, in the same way fixed costs are associated with potential facility location sites in the models of Section 2.3. The generalized p-median problem can be then defined as follows [12].

Given a network $N = (X, A)$, with distance matrix $D(N) = [d(x_i, x_j)]$, vertex weights v_i and vertex fixed costs F_i, the problem is to find a subset $\overline{X}_p$ containing p vertices so that

$$Z = \sum_{x_i \in X_p} F_i + \sigma(X_p) \qquad (2.25)$$

is minimized.

Thus the objective is to minimize not just the transmission $\sigma(X_p)$ of X_p but the total function Z which includes a fixed cost F_i for every vertex x_i in X_p. The p-median problem then corresponds to the case in which all F_i are equal (say F) so that the first term of Equation (2.25) becomes a constant (equal to pF), and does not influence the search for the optimal set $\overline{X}_p$.

A version of the p-median problem that is often encountered in practice is one in which $\overline{X}_p$ is not required to contain exactly p vertices, but any number less than or equal to p. The problem becomes then to minimize Equation (2.25) subject to $|X_p| \leqq p$.

Finally, the capacitated p-median problem can be also defined. A restriction on the maximum value that the number

$$\sum_{x_j \text{ allocated to } x_i} v_j \qquad (2.26)$$

can take for $\forall\ x_i \in \overline{X}_p$, can be added to the formulation of the p-median problem. Equation (2.26) is a measure of the throughput transmitted from x_i, and is therefore also a measure of the physical size of a facility located at vertex x_i.

As already pointed out, the main difficulty in solving minisum network location problems rests with the pure p-median problem. Until this problem is satisfactorily resolved, there is little point in attempting to solve generalized p-median problems.

2.4.3 Generalization of Hakimi's fundamental theorems

Since Hakimi first proved Theorems 2.1 and 2.2 his results have been generalized by several authors.

Transmission functions $\sigma(X_p)$ defined as the sum of arbitrary concave functions of weighted distances are studied by Levy [73] , Goldman [42] and Hakimi and Maheshwary [50] . Levy [73] proves that Theorems 2.1 and 2.2 are valid for transmissions that are concave with respect to distance. In a later paper, Hakimi and Maheshwari [50] show that, under fairly general assumptions, one could, without loss of optimality, restrict the location of facilities to the vertices of the network in a wide range of minisum network location problems. Conclusions drawn by Hakimi and Maheshwari are:

1. Theorems 2.1 and 2.2 hold when capacity constraints are placed on the arcs of the networks;
2. The theorems will generally not hold for the capacitated case, unless the location of more than one facility at a single vertex is allowed.

Wendell and Hurter [102] establish some necessary and sufficient conditions for optimal solutions to minisum network location problems to occur at the vertices of the network. They show that for problems in which:

1. There exist constraints on arc capacities, and
2. Transportation costs are nondecreasing concave (this has been generalized to include cases in which these costs differ from arc to arc),

Theorems 2.1 and 2.2 remain valid. These sufficiency conditions are very similar to some of the results obtained by Hakimi and Maheshwari [50].

Whereas some theoretical advances have been made in minisum network location problems, computational difficulties abound even for the pure p-median problem. A survey of solution methods available in the literature for the p-median problem is provided in the next section.

2.4.4 Methods for the p-median problem

Several algorithms, both exact and heuristic solution methods, have been proposed for the solution of the p-median problem. The exact solution methods are:

1. Branch-and-bound algorithms [30, 55];
2. Two different formulations of the linear programming (LP) relaxation of the integer programming (IP) formulation of the problem [41, 90];
3. An alternative approach via linear programming [78], based on Lagrange multipliers and parametric linear programming.

Heuristic methods are reviewed in greater detail in Chapter Six. The more fundamental heuristics proposed for the problem, however, are briefly described in this section.

Branch-and-bound Methods

Järvinen, Rajala and Sinervo [55] appear to have been the first to solve the p-median problem through branch-and-bound. Their algorithm starts with all facilities "open".* A lower bound defined by the authors is then used to successively "close" facilities until exactly p facilities are left "open". The iterative process continues until all feasible solutions have been implicitly evaluated.

A different branch-and-bound procedure was developed by El-Shaieb [30]. In his algorithm the tree branches represent assignments of sources (facilities) and destinations. Locations are added one at a time to either the source or the destination set to form the next branches. From each node of the tree there are two branches. One of the branches corresponds to adding a location to the source set, while the other branch corresponds to adding the same location to the destination set. At the end of each branch there is a node that contains the corresponding source and destination sets.

Two lower bounds were developed by El-Shaieb for his algorithm. One of the bounds is reported to be efficient for small values of p, whereas the other is shown to perform better for the larger values of p. The algorithm was tested for the 10, 20 and 30 major metropolitan centres in the United States, with p=2, 4 and 6.

It is very difficult to compare the efficiency of El-Shaieb's algorithm to that of Järvinen et al. Not only the test problems of the two papers are different, but also the computers and even the level of the programming languages used by the respective authors differ substantially.

Khumawala, Neebe and Dannenbring [63] attempt to compare El-Shaieb's algorithm with other exact and heuristic procedures for the p-median problem. In this attempt El-Shaieb's results are tabulated alongside results obtained through the following methods:

* An "open" facility is defined here as a vertex of the network temporarily assigned to be one of the medians. A "closed" facility is a vertex of the network temporarily assigned to the nonmedian set.

1. The Teitz and Bart heuristic method [98];
2. An algorithm originally designed for minimax network location problems;
3. The Linear Programming/Group Theoretic algorithm of Garfinkel, Neebe and Rao [41]. This algorithm is reviewed below in some detail.

Khumawala et al. conclude that a comparative evaluation is very difficult, and content themselves with making a few comments on each of the methods considered by them.

A Linear Programming Relaxation of the Integer Programming Formulation of the Problem

The integer programming formulation of the p-median problem has already been given in Section 1.2 [Equations (1.7) to (1.11)]. For the sake of convenience this formulation is repeated below:

Minimize

$$Z = \sum_{i \in I} \sum_{j \in J} d_{ij} \, \xi_{ij} \tag{2.27}$$

Subject to

$$\sum_{i \in I} \xi_{ij} = 1, \quad j \in J \tag{2.28}$$

$$\sum_{i \in I} \xi_{ii} = p \tag{2.29}$$

$$\xi_{ij} \leqq \xi_{ii}, \quad i \in I, \ j \in J \ , \ i \neq j \tag{2.30}$$

$$\xi_{ij} = \begin{cases} 1 & \text{if customer } \underline{j} \text{ is allocated to facility } \underline{i} \\ 0 & \text{otherwise} \end{cases} \tag{2.31}$$

If the {0,1} constraints represented by Equation (2.31) are relaxed to

$$\xi_{ij} \geqq 0, \quad i \in I, \quad j \in J \ , \tag{2.32}$$

the resulting problem is a linear programming problem. Note that in the LP relaxation an upper bound of value 1 on ξ_{ij} is not necessary, since $\xi_{ij} \leqq 1$ is implied by Equation (2.28).

Revelle and Swain [90] used the IBM Mathematical Programming System (MPS) package to solve the general LP formulation given by Equations (2.27) to (2.30) and (2.32). They report that a 30-vertex, 6-median problem required 173 MPS iterations and 1.51 minutes of computer time to converge to an optimal integer solution on an IBM 360/65.

The solution to the LP is not necessarily all-integer and fractional values of ξ_{ij} can and do occur. Revelle and Swain report, however, that fractional values of ξ_{ij} occur rarely. In the unlikely event of a non-integer solution, they recommend a branch-and-bound scheme to resolve the problem with integers. Unfortunately, very little computational experience is reported with respect to the branch-and-bound scheme.

The main problem with the general LP formulation above is that it produces very large linear programmes. For a network of $\underline{n}$ vertices, the number of variables $\underline{n^2}$ and the number of constraints $\underline{n^2 + 1}$. Revelle and Swain suggest cutting down the number of constraints by adding the assignment constraints given by Equation (2.30) only as needed. In a generalization of the LP relaxation to a class of location-allocation problems, Morris [81] experimented with this technique. He concludes that even when this procedure is used, the use of LP for large scale problems is precluded.

Garfinkel, Neebe and Rao [41] solve the IP relaxation by decomposition, thus considerably reducing the size of the problem. In their decomposition formulation the LP basis of the master problem contains only $\underline{n + 2}$ rows, and each of the $\underline{n}$ subproblems can be solved by inspection. Due to the very degenerate nature of the LP basis of the master problem, however, in many cases the algorithm fails to converge. This lack of convergence is a very serious problem, and prevents the decomposition formulation from effectively solving the problem.

Difficulties with convergence are practically not mentioned by the authors of [41]. An extensive study of this phenomenon is made in Chapter Three of this thesis. The general LP formulation of Revelle and Swain is also discussed in the same chapter.

In the Garfinkel et al. paper, the LP decomposition formulation represents only part of the work. In cases of non-integer termination of the LP, the integer formulation of the problem is attacked through group theoretic techniques and a dynamic programming recursion. Garfinkel et al. report some computational experience with their proposed procedures.

Finally, an alternative approach via linear programming is given by Marsten [78]. He shows that the solution corresponding to the optimal p-median of a network [as described in Equations (2.27) to (2.31)], is an extreme point of a certain polyhedron H, and that all other p-medians for $1 \leqq p \leqq n$ are also extreme points of H. Using Lagrange multipliers and parametric linear programming, Marsten gives a method of traversing a path among a few of the extreme points of H. This path successively generates the p-medians of the network N in descending order of $\underline{p}$, although for some values of $\underline{p}$ the solution may be missed and never generated, or, conversely, extreme points of H may be generated which do not correspond to p-medians of N, i.e. contain fractional values of ξ_{ij}.

Thus, although Marsten's method is both theoretically and computationally attractive, it may fail to produce the p-median of a network for the specific value of $\underline{p}$ that may be required. In [78] Marsten reports the case of a complete 33-vertex network, all of whose optimal p-medians were successfully generated for p = 33,32,...,10, but whose optimal 9-median and 8-median could not be obtained by his method.

Heuristic Methods

Heuristic methods for the p-median problem first appeared in papers by Maranzana [76] and Teitz and Bart [98]. The method put forward by Maranzana parallels in several respects one of the heuristics devised by Cooper in [17] for the continuous case. This method is referred to as the partition method, and in essence is approaches the p-median by finding successive single vertex medians of p subsets of destination vertices, each associated with one source, and then adjusting the subsets before repeating the process. A similar approach was later studied by Surkis [96].

Teitz and Bart [98] describe a heuristic method based on vertex substitution. The method proceeds by choosing any p vertices at random to form an initial set S, which is assumed to be an approximation to the optimal p-median set $\overline{X}_p$. The method then tests if any vertex $x_j \in (X-S)$ can replace a vertex $x_i \in S$ and so produce a new set $S' = S \cup \{x_j\} - \{x_i\}$ yielding a better solution to the problem than the solution implied by the set S. If so, vertex x_i is replaced by vertex x_j and a new set S' is obtained which is a better approximation to $\overline{X}_p$. The same tests are now performed on the new set S', and the procedure is repeated until a set $\overline{S}$ is finally obtained for which no replacement of any vertex in $\overline{S}$ by a vertex in $(X - \overline{S})$ produces a set whose implied solution is better than the solution produced by $\overline{S}$. This final set $\overline{S}$ is then taken to be the required approximation to $\overline{X}_p$.

Contrary to what was initially conjectured by Revelle et al. [89], the vertex substitution method does not produce an optimal solution in all cases. Counter examples to this conjecture can be found in [12] and [55].

Due to the importance of the methods of Maranzana [76] and Teitz and Bart [98], they will be described in greater detail in Chapter Six.

2.5 Conclusions

The p-median and related network location problems have been surveyed in the present chapter. In addition, network location models have been related to more general models in location analysis, of which they are a special case. The survey was not only concerned with models and methods of solution, but also with definitions, theorems and cost functions of interest for the problems covered in the survey.

The fundamental theorems for the p-median problem are those of Hakimi [48, 49], and their extensions by Goldman [42], Levy [73], Hakimi and Maheshwari [50] and Wendell and Hurter [102]. These results were reviewed, and this was followed by a survey of exact and heuristic solution methods currently available to solve the p-median problem.

Although remarkable theoretical progress has been made in relation to the p-median and other minisum network location problems, much remains to be done in the computational side. This is particularly true for the pure p-median problem.

For this problem, branch-and-bound algorithms were developed, but the lack of efficient lower bounds only allow them to solve the problem for medium-size networks. There are yet unsolved problems in both formulations of the LP relaxation of the p-median problem. Existing heuristic procedures can be further extended.

The following chapters attempt to overcome these difficulties. New ideas and solution procedures are developed, and they represent a contribution towards solving the computational difficulties of the p-median problem.

CHAPTER THREE

LINEAR PROGRAMMING FORMULATIONS OF THE RELAXED p-MEDIAN PROBLEM

3.1 Introduction

The integer programming formulation of the p-median problem - and its corresponding LP relaxation - have been introduced in Chapter 2. Garfinkel et al. [41] solved the LP relaxation by decomposition, thus considerably reducing the size of the linear programme. Due to the very degenerate nature of the master problem, however, serious difficulties with convergence prevent the relaxed p-median problem from being solved by decomposition in many cases.

The importance of the linear programming formulations stems from the fact that in the majority of the cases the solution to the linear programme is all-integer, thus also being a solution to the p-median problem. It is true that fractional LP solutions do occur, but these occurrences are rare. Fractional solutions generally occur for highly contrived cost matrices, difficult to represent in terms of an actual network. The data in these contrived matrices follow the pattern of the cost counter-cycles mentioned by Revelle and Swain [90].

In the present chapter both the general formulation of Revelle and Swain and the decomposition formulation of Garfinkel et al. are studied in detail. Some computational experience is reported for the general formulation. An example of a contrived cost matrix is also presented.

The decomposition formulation is studied in far greater detail. In order to illustrate the method, a small example is solved by hand. Then computational results show the extent of the difficulties with convergence. Finally comments of a general nature are made in relation to the convergence of the algorithm.

The importance of eventually overcoming the convergence problems of the decomposition formulation explains why the main part of this chapter has been dedicated to this method. If the difficulties arising from the lack of convergence of the algorithm can be solved, then the decomposition formulation, in conjunction with its embedding into branch-and-bound algorithms, can be used to solve the p-median problem for large-scale networks.

3.2 The General Linear Programming Formulation

The integer programming formulation of the p-median problem has already been given in the first two chapters of this thesis. A formal statement of this formulation [12, Chapter 6] is now given in the following.

Let $[\xi_{ij}]$ be a $(n\times n)$ allocation matrix so that $\xi_{ij} = 1$ if vertex x_j is allocated to vertex x_i, $\xi_{ij} = 0$ otherwise.

Further, let $\xi_{ii} = 1$ imply that vertex x_i is a median vertex and let $\xi_{ii} = 0$ otherwise. The p-median problem can be then stated as follows:

Minimize

$$Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, \xi_{ij} \tag{3.1}$$

Subject to

$$\sum_{i=1}^{n} \xi_{ij} = 1, \quad j = 1,\dots,n \tag{3.2}$$

$$\sum_{i=1}^{n} \xi_{ii} = p \tag{3.3}$$

$$\xi_{ij} \leqq \xi_{ii}, \qquad i,j = 1,\dots,n, \quad i\neq j \tag{3.4}$$

and

$$\xi_{ij} = 0 \text{ or } 1, \tag{3.5}$$

where $[d_{ij}]$ is the weighted distance matrix of the network, i.e. the distance matrix of the network with every column $\underline{j}$ multiplied by a weight v_j.

It is worthwhile to discuss the meaning of the constraints of this integer programme. Equation (3.2) ensures that any vertex x_j is allocated to one and only one median vertex x_i. Equation (3.3) guarantees that there are exactly $\underline{p}$ medians, and Equation (3.4) makes sure that $\xi_{ij} = 1$ only if $\xi_{ii} = 1$, i.e. non-median vertices are only allocated to vertices that are in the median set. If $[\overline{\xi}_{ij}]$ is the allocation matrix corresponding to the optimal solution, the optimal p-median is given by

$$\overline{X}_p = \{x_i \mid \xi_{ii} = 1\}. \tag{3.6}$$

As already noted, if Equation (3.5) is replaced by

$$\xi_{ij} \geqq 0, \quad i,j = 1,\ldots,n , \tag{3.7}$$

the resulting problem is the linear programming relaxation of the p-median problem. It has also been already pointed out that in the LP relaxation an upper bound of value 1 on ξ_{ij} is not necessary, since $\xi_{ij} \leqq 1$, $i,j = 1,\ldots,n$, is implied by Equation (3.2).

<u>Solving the linear programme</u>

Revelle and Swain [90] used a standard IBM mathematical programming package (MPS) to solve the formulation given by Equations (3.1) to (3.4) and (3.7). Their experience with this formulation was reported in Chapter 2. The main interest of this research in the general formulation is not in the formulation per se, but in the possibility of embedding it into the branch-and-bound algorithm of Chapter 5. It was decided therefore that a simple computer code should be used to solve the LP, i.e. a code that could be easily adapted to be activated at every node generated by the branch-and-bound algorithm.

The code chosen for this purpose was a Nottingham Algorithms Group (NAG) subroutine. This subroutine is not especially efficient, as it stores all the data for the LP in the central processing unit of the computer. Consequently, due to the size of the linear programmes generated by the general formulation, it was not possible to go beyond a 10-vertex network when using the NAG subroutine to test this formulation of the LP.

The experience with the embedding of the general formulation into the branch-and-bound algorithm of Chapter 5 is described in that chapter. In the present chapter only some computational results of general interest to this approach are given.

Computational experience

It was not easy to find a small network for which the LP relaxation of the p-median problem would yield a fractional solution for a given $1 \leqq p \leqq n$. Confirming the experience of Revelle and Swain [90], non-integer solutions were only obtained for highly contrived matrices with cost counter-cycles. Garfinkel et al., however, do provide in their paper [41, p. 231] a 10-vertex network for which the LP relaxation yields a fractional solution for $p = 3$. This network is shown in Figure 3.1.

In Figure 3.1, the numbers alongside the arcs are distances between vertices. All vertices have unit weights. The LP relaxation of the problem was solved and the following solution was obtained for $p = 3$:

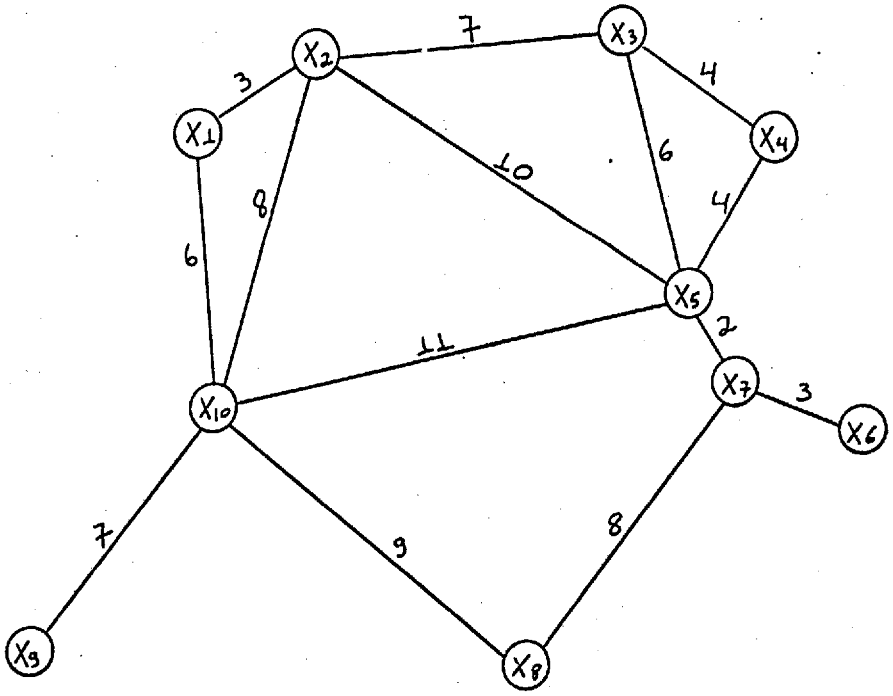


Figure 3.1

10-vertex network of Garfinkel et al. [41, p.231]

$$\xi_{11} = \xi_{12} = \xi_{1,10} = 0.5$$

$$\xi_{21} = \xi_{22} = \xi_{23} = 0.5$$

$$\xi_{53} = \xi_{54} = \xi_{55} = \xi_{56} = \xi_{57} = 0.5$$

$$\xi_{74} = \xi_{75} = \xi_{76} = \xi_{77} = \xi_{78} = 0.5$$

$$\xi_{99} = 0.5$$

$$\xi_{10,8} = \xi_{10,9} = \xi_{10,10} = 0.5 .$$

The value of the objective function for the solution above is 35.5. The solution to the optimal 3-median of the network of Figure 3.1 consists in fact of six different 3-vertex sets, all with an objective function equal to 36. It is interesting to note that for this network, for all other possible values of p the solution to the LP is all-integer, and therefore also a solution to the corresponding p-median problem.

It took 153 iterations of the simplex method and 70.69 CDC 6400 seconds for the LP to converge to the fractional solution shown above. Thus, the solution to the general LP formulation is not particularly fast.

It has been pointed out that non-integer solutions to the LP relaxation of the p-median problem are often obtained for highly contrived matrices with cost counter-cycles. An example of this type of matrix is given in Figure 3.2.

3.3 The Decomposition Formulation [41]

Consider the LP relaxation of the IP formulation of the p-median problem (Equations (3.1) to (3.4) and (3.7) of Section 3.2). It is possible to decompose the LP on the index i. The linking constraints will be (3.2) and (3.3), which together with the objective function will constitute the master problem, the basis of which contains only n+2 rows. Rewriting the LP in a form suitable for decomposition, the following results:

TO

FROM	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
x_1	0	0.5	N	N	6.0	N	N	N	N	N
x_2	7.0	0	1.5	N	N	N	N	N	N	N
x_3	N	12.0	0	2.5	N	N	N	N	N	N
x_4	N	N	10.0	0	2.0	N	N	N	N	N
x_5	1.0	N	N	8.0	0	N	N	N	N	N
x_6	N	N	N	N	N	0	1.5	N	N	6.0
x_7	N	N	N	N	N	7.0	0	2.5	N	N
x_8	N	N	N	N	N	N	8.0	0	3.0	N
x_9	N	N	N	N	N	N	N	9.0	0	3.5
x_{10}	N	N	N	N	N	4.0	N	N	10.0	0

FIGURE 3.2

Contrived Cost Matrix

N = Large Number

Minimize

$$Z = \sum_{i=1}^{n} D_i X_i \tag{3.8}$$

Subject to

$$\sum_{i=1}^{n} A_i X_i = b_0 \tag{3.9}$$

$$B_i X_i \leqq 0 , \quad i = 1,\ldots,n \tag{3.10}$$

$$X_i \geqq 0, \quad i = 1,\ldots,n \tag{3.11}$$

where*

$$D_i = (d_{i1},\ldots,d_{in})$$

$$X_i = (\xi_{i1},\ldots,\xi_{in})^T$$

$$A_i = \left(\begin{array}{c} I_n \\ \hline e_i^T \end{array}\right) .$$

$$b_0 = [1,\ldots,1, p]^T$$

$$B_i = [e_1, e_2,\ldots,e_{i-1}, - p_{n-1}^T, e_i,\ldots,e_{n-1}]$$

$$0 = [0,\ldots,0]^T .$$

I_n is the n^{th} identity matrix, e_i is the i^{th} unit column vector of appropriate dimension, p_k is a row vector containing k 1's, and T denotes transpose.

Note that the constraint set

$$S_i = \{X_i \mid B_i X_i \leqq 0, \quad X_i \geqq 0\} \tag{3.12}$$

has one extreme point for each subproblem defined by (3.10) and (3.11), namely $X_i = 0$. Then, if the null vector is considered to be a degenerate extreme ray, at any iteration one of the extreme rays of S_i will be found. Thus the usual convexity constraints can be omitted from the master problem.

* ( ) denotes a row vector, and [ ] a column vector.

Now let y_i^k, $k = 1,\ldots,T_i$ be the extreme rays of S_i. Then, using the notation of Hadley [47], the modified master problem is

Minimize

$$Z = \sum_{i=1}^{n} \sum_{k=1}^{T_i} \rho_i^k D_i y_i^k \tag{3.13}$$

Subject to

$$\sum_{i=1}^{n} \sum_{k=1}^{T_i} \rho_i^k A_i y_i^k = b_0 \tag{3.14}$$

$$\rho_i^k \geqq 0 \text{ for all } i,k \tag{3.15}$$

Given a feasible basis B to the master problem above, <u>n</u> subproblems of the form

Maximize

$$\theta_i = F_i y_i \tag{3.16}$$

Subject to

$$B_i y_i \leqq 0 \tag{3.17}$$

$$y_i \geqq 0 \tag{3.18}$$

must be solved. An optimal solution to the subproblem above will be an extreme ray of S_i, and $F_i = \sigma A_i - D_i$ is the row vector of dual slack variables associated with the basis B. Because of the simple structure of A_i, given σ it is trivial to calculate F_i without matrix multiplication, since

$$F_i = \bar{\sigma}_i - D_i \, , \tag{3.19}$$

where

$$\bar{\sigma}_i = (\sigma_1,\ldots,\sigma_{i-1}, \sigma_i + \sigma_{n+1}, \sigma_{i+1},\ldots,\sigma_n) \, .$$

If y_i^* is an optimal solution to subproblem <u>i</u> with value θ_i^* for a given $\bar{\sigma}_i$, then

$$\theta = \text{Max}_{1 \leqq i \leqq n} \; \theta_i^* \tag{3.20}$$

If $\theta = 0$, an optimal solution to the LP has been found. If $\theta > 0$, that vector y_i^* which yields θ is brought into the basis.

The subproblems can be solved by inspection. Letting $F_i = (f_{i1},\ldots,f_{in})$ and $y_i = (y_{i1},\ldots,y_{in})$, the subproblems are of the form

Maximize

$$\theta_i = \sum_{j=1}^{n} f_{ij}\, y_{ij} \tag{3.21}$$

Subject to

$$y_{ij} - y_{ii} \leqq 0, \qquad j = 1,\ldots,n\,,\ i \neq j \tag{3.22}$$

$$y_{ij} \geqq 0, \qquad j = 1,\ldots,n \tag{3.23}$$

In order to solve the subproblems above, calculate

$$t_i = f_{ii} + \sum_{i \neq j} \text{Max}\,(0,\ f_{ij}) \tag{3.24}$$

If $t_i \leqq 0$, then $y_{ij}^* = 0$, $j = 1,\ldots,n$. If $t_i > 0$, then $y_{ii}^* = 1$ and, for all $i \neq j$

$$y_{ij}^* = \begin{cases} 1 & \text{if } f_{ij} > 0 \\ 0 & \text{if } f_{ij} \leqq 0 \end{cases} \tag{3.25}$$

It should be noted here that θ_i will also be maximized if, for $t_i > 0$, $y_{ii}^* = 1$, and, for all $i \neq j$

$$y_{ij}^* = \begin{cases} 1 & \text{if } f_{ij} \geqq 0 \\ 0 & \text{if } f_{ij} < 0 \end{cases} \tag{3.26}$$

This second possibility is not mentioned in [41], and although apparently not very significant, it has proved to be of some importance, especially when the convergence of the algorithm is studied. This point will be brought to attention again in Section 3.4.3.

Thus if $t_i > 0$, y_i^* is a binary n-vector with a one in the i^{th}

position. The column introduced into the basis is easily seen to be

$$H_i^* = A_i \, y_i^* = \begin{bmatrix} y_i^* \\ \\ 1 \end{bmatrix} \tag{3.27}$$

Since H_i^* is a binary vector, premultiplication by B^{-1} involves nothing more than addition of the columns of B^{-1} corresponding to the 1's in H_i^*. Thus multiplications and divisions are not needed until the pivot step.

Initial basic feasible solution

Because of the simple structure of the constraint matrix, initial basic feasible solutions are readily obtained without a 'Phase I' procedure. Two such possibilities are as follows.

Initial Solution A

Since $d_{ij} \geqq 0$ for all _i_ and _j_, it is clear that (3.2) can be replaced by

$$\sum_{i=1}^{n} \xi_{ij} \geqq 1 \;, \qquad j = 1,\ldots,n \tag{3.2a}$$

without loss of any optimal solution. It would also be desirable to replace (3.3) by

$$\sum_{i=1}^{n} \xi_{ii} \geqq p. \tag{3.3a}$$

However, since $d_{ii} = 0$ for all _i_, this would result in a median being located at every vertex. In order to avoid this, it is necessary to alter the distance (cost) structure so that

$$\begin{cases} d'_{ii} = d_{ii} + W \\ d'_{ij} = d_{ij} \quad \text{for } i \neq j \;, \end{cases} \tag{3.28}$$

where W is an arbitrarily large positive constant. This has the effect of forcing equality in (3.3a), while adding the constant $\underline{pW}$ to the objective function.

Now let $X'_p = \{x'_i \mid \xi'_{ii} = 1\}$ be any feasible solution to the p-median problem. Generating X'_p is a simple matter. One can easily choose $\underline{p}$ vertices to be medians, and then assign nonmedian vertex x_j to median x^*_i if $d'_{i^*j} = \underset{i}{\text{Min}}\ d'_{ij}$, x_i a median, where ties may be broken arbitrarily. Without loss of generality, assume $\xi'_{11} = \ldots = \xi'_{pp} = 1$. This can always be achieved by renumbering the vertices.

Now, in order to construct an initial basic feasible solution to the master problem, note that X'_p generates solutions to each of the $\underline{p}$ subproblems defined from (3.21) to (3.23). These solutions are y-vectors of the form

$$y_i = \left[\begin{array}{c} e_i \\ \hdashline q_i \end{array}\right], \tag{3.29}$$

where e_i is the i^{th} unit p-vector, and q_i is an (n-p)-vector whose j^{th} component is one if vertex x_{p+j} is allocated to median x_i, and zero otherwise. Thus $\underline{p}$ vectors of the form $\begin{bmatrix} y_i \\ 1 \end{bmatrix}$ can be placed in the basis, where the 1 is from constraint (3.3a). The basis is then filled out with (n-p) surplus variables from (3.2a) and one from (3.3a).

Thus

$$B_0 = \left[\begin{array}{c|c|c|c|c|c} e_1 & \text{---} & e_p & 0 & \text{---} & 0 \\ \hline q_1 & \text{---} & q_p & -e_1 & \text{---} & -e_{n-p+1} \\ \hline 1 & \text{---} & 1 & & & \end{array}\right], \tag{3.30}$$

or

$$B_0 = \left[\begin{array}{c|c} I_p & 0 \\ \hline Q & -I_{n-p+1} \\ \cline{1-1} P_p & \end{array}\right], \tag{3.31}$$

where P_p is the sum p-vector (1,...,1). The basis B_0 has the desirable property of being involutory ($B_0 = B_0^{-1}$), so that the dual variables are readily computed as

$$\sigma = D_B\, B_0^{-1} = D_B\, B_0 , \tag{3.32}$$

and the initial LP solution is $B_0^{-1}\, b_0 = [\, P_p,\ 0]^T$. Note that none of the last n-p+1 variables in B_0 will be in the optimal LP basis at a positive level.

Initial Solution B

Another easily invertable basis that has the advantage over B_0 of containing only one surplus variable is

$$B_1 = \left[\begin{array}{c|c|c} I_p & 0 & 0 \\ \hline Q & I_{n-p} & 0 \\ \hline P_p & P_{n-p} & -1 \end{array}\right], \tag{3.33}$$

where Q is defined in (3.31). The matrix I_{n-p} corresponds to allocating vertices p+1 through n to themselves. The inverse of B_1 is

$$B_1^{-1} = \left[\begin{array}{c|c|c} I_p & 0 & 0 \\ \hline -Q & I_{n-p} & 0 \\ \hline (-p_{n-p}Q)+P_p & P_{n-p} & -1 \end{array}\right], \tag{3.34}$$

and the initial LP solution is $B_1^{-1}b_0 = [P_p,\ 0]^T$.

If B_1 is used (and assuming $n-p \geq 2$) it is possible to remove, on the first pivot, the last column corresponding to the surplus variable. This is done by introducing into B_1 a column H_1 corresponding to vertex n (or (n-1)) being a median, with vertex (n-1) (or n) allocated to it.

Column H_i contains a 1 in the last three rows and zeros elsewhere. Thus,

$$B_1^{-1} H_i = H_i \ , \tag{3.35}$$

and H_i can be introduced into B_1 and the last column dropped. .

Basis B_1 was the one actually used when the LP decomposition algorithm was coded, and for which computational results are given in a later section of this chapter.

3.4 The Decomposition Formulation Studied in Detail

The mathematical derivation of the decomposition formulation was given in Section 3.3. This formulation is now studied in detail. The decomposition formulation is initially illustrated by means of a small example solved by hand. Then computational results are given in 3.4.2. Finally degeneracy and the problems with convergence are discussed in Section 3.4.3.

3.4.1 A small example solved by hand

It was thought that the best way to illustrate the decomposition formulation would be to solve a small example by hand.

The initial basic feasible solution used for this purpose was Initial Solution A, despite the fact that Initial Solution B was used when the algorithm was programmed for the computer. Initial Solution A was chosen for the illustration because this was considered the best way to give a 'physical' interpretation to this formulation of the problem.

Consider the complete directed 4-vertex network whose distance matrix is given in Figure 3.3, and for which the optimal 2-median must be found. If the structure of the distance matrix is altered as per Equation (3.28), the matrix of Figure 3.4 is obtained.

$$\begin{array}{cc|cccc|}
 & & \multicolumn{4}{c}{\underline{\text{TO}}} \\
 & & X_1 & X_2 & X_3 & X_4 \\
\hline
 & X_1 & 0 & 1 & 2 & 3 \\
\underline{\text{FROM}} & X_2 & 3 & 0 & 1 & 2 \\
 & X_3 & 4 & 3 & 0 & 1 \\
 & X_4 & 3 & 2 & 2 & 0 \\
\hline
\end{array}$$

FIGURE 3.3

Distance Matrix of Illustrative Example

$$\begin{array}{cc|cccc|}
 & & \multicolumn{4}{c}{\underline{\text{TO}}} \\
 & & X_1 & X_2 & X_3 & X_4 \\
\hline
 & X_1 & W & 1 & 2 & 3 \\
\underline{\text{FROM}} & X_2 & 3 & W & 1 & 2 \\
 & X_3 & 4 & 3 & W & 1 \\
 & X_4 & 3 & 2 & 2 & W \\
\hline
\end{array}$$

FIGURE 3.4

Modified Distance Matrix

The data for the LP decomposition formulation of Equations (3.8) to (3.11) are then

$$D_1 = (W, 1, 2, 3) \quad ; \quad X_1 = (\xi_{11}, \xi_{12}, \xi_{13}, \xi_{14})^T$$
$$D_2 = (3, W, 1, 2) \quad ; \quad X_2 = (\xi_{21}, \xi_{22}, \xi_{23}, \xi_{24})^T$$
$$D_3 = (4, 3, W, 1) \quad ; \quad X_3 = (\xi_{31}, \xi_{32}, \xi_{33}, \xi_{34})^T$$
$$D_4 = (3, 2, 2, W) \quad ; \quad X_4 = (\xi_{41}, \xi_{42}, \xi_{43}, \xi_{44})^T$$

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad ; \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad ; \quad A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$b_0 = [1, 1, 1, 1, 2]^T$$

$$B_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \quad ; \quad B_2 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$B_3 = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad ; \quad B_4 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Then, if Equations (3.8) to (3.11) are written in full, the following is obtained:

1. Equation (3.8)

Minimize

$$Z = \sum_{i=1}^{4} D_i X_i = (W,1,2,3)(\xi_{11},\xi_{12},\xi_{13},\xi_{14})^T + \dots +$$

$$+ (3,2,2,W)(\xi_{41},\xi_{42},\xi_{43},\xi_{44})^T =$$

$$= W\,\xi_{11} + 1\times\xi_{12} + \dots + 2\times\xi_{43} + W\,\xi_{44} ,$$

Subject to

2. Equation (3.9)

$$\sum_{i=1}^{4} A_i X_i = b_0 \text{ , or}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_{11} \\ \xi_{12} \\ \xi_{13} \\ \xi_{14} \end{bmatrix} + \dots + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_{41} \\ \xi_{42} \\ \xi_{43} \\ \xi_{44} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{bmatrix} ,$$

or, finally,

$$\begin{aligned} \xi_{11} + \xi_{21} + \xi_{31} + \xi_{41} &= 1 \\ \xi_{12} + \xi_{22} + \xi_{32} + \xi_{42} &= 1 \\ \xi_{13} + \xi_{23} + \xi_{33} + \xi_{43} &= 1 \\ \xi_{14} + \xi_{24} + \xi_{34} + \xi_{44} &= 1 \\ \xi_{11} + \xi_{22} + \xi_{33} + \xi_{44} &= 2 \end{aligned}$$

Equations (3.8) and (3.9) correspond to the master problem. Turning now to the subproblems, given by Equations (3.10) and (3.11):

3. Subproblem 1 (i=1)

$$B_1 X_1 \leqq 0 \;, \qquad X_1 \geqq 0 \;,$$

or

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi_{11} \\ \xi_{12} \\ \xi_{13} \\ \xi_{14} \end{bmatrix} \leqq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} ,$$

or, finally,

$$-\xi_{11} + \xi_{12} \leqq 0$$
$$-\xi_{11} + \xi_{13} \leqq 0$$
$$-\xi_{11} + \xi_{14} \leqq 0 ,$$

and

$$\xi_{1j} \geqq 0 , \quad j = 1,\dots,4,$$

and similarly for the other three subproblems.

It is now possible to understand more easily the meaning of the vectors y_i^k, $k = 1,\dots,T_i$, described in Section 3.3 as the extreme rays of S_i. They are vectors for which either

A. $y_{ij} = \xi_{ij} = 0$ for all $\underline{j}$,

or

B. $y_{ii} = \xi_{ii} = 1$ and $y_{ij} = \xi_{ij} = 0$ or 1 for all $j \neq i$.

The value of y_{ii} in the above is $y_{ii} = 1$ if vertex x_i is assigned as a median, and $y_{ii} = 0$ otherwise. When a vertex x_i is assigned as a median, $y_{ij} = 1$ indicates that vertex x_j is allocated to median x_i and $y_{ij} = 0$ otherwise.

Any of the subproblems can generate 2^{n-1} vectors that satisfy Equations (3.10) and (3.11) for a specific $\underline{i}$. Not all these vectors, however, can be considered as candidates to enter the basis of the master problem, since at every iteration the number of medians must be equal to $\underline{p}$ in order that feasibility is maintained in the master problem. The problem then is, all subproblems considered, to enter the basis of the master problem as few vectors as possible before the optimal solution to the LP is obtained. This explains the procedure developed to choose the vector to enter the basis of the master problem at every iteration of the algorithm.

In the following the decomposition formulation is applied to find the optimal 2-median of the network whose distance matrix is shown in Figure 3.3.

Initial basic feasible solution

In order that an initial basic feasible solution is generated for the problem, let x_1 and x_2 be assigned as medians, i.e. let $X_2' = \{x_1, x_2\}$. If x_3 and x_4 are then allocated to the two medians above in the best possible way, the following is obtained:

$$\xi_{11} = 1$$

$$\xi_{22} = \xi_{23} = \xi_{24} = 1 \text{ ,}$$

and

$$\sigma(X_2') = 3$$

The next step is to generate B_0, given by (3.31):

$$B_0 = B_0^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 1 & 1 & 0 & 0 & -1 \end{bmatrix} .$$

The initial LP solution is

$$\beta = B_0^{-1} b_0 = [P_p, 0]^T = \begin{bmatrix} V_1^{i=1} \\ V_2^{i=2} \\ S_3 \\ S_4 \\ S_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} ,$$

where $V_{\ell}^{i=k}$ means the ℓ^{th} vector to enter the basis, with $i = k$ implying that vertex x_k is the assigned median in this vector. S_3, S_4 and S_5 are the vectors corresponding to the surplus variables. The initial ordered list of basic variables is then $V_1^{i=1}$, $V_2^{i=2}$, S_3, S_4 and S_5.

The vector D_B corresponding to the initial basis is $D_B = (W, W+3, 0, 0, 0)$. Now σ can be readily calculated from Equation (3.32):

$$\sigma = D_B B_0^{-1} = D_B B_0 = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (W, W+3, 0, 0, 0)$$

It should be noted that the components of vector D_B are the cost of the allocations of vertices to medians implied by the corresponding column vectors in the basis.

Solving the problem

The 2-median problem is solved in the following. An optimal solution was obtained after six iterations. The interpretation of this optimal solution is given after the completion of the algorithm. Because of the nature of the decomposition formulation, the inverse matrix method, in the product form, was used to solve the master problem. For details concerning this method see Beale [6, Chapter 7].

First iteration

The first step is to solve each of the four subproblems, so that θ and the vector to enter the basis can be determined:

$$\bar{\sigma}_1 = (\sigma_1+\sigma_5, \sigma_2, \sigma_3, \sigma_4) = (W, W+3, 0, 0)$$
$$\bar{\sigma}_2 = (\sigma_1, \sigma_2+\sigma_5, \sigma_3, \sigma_4) = (W, W+3, 0, 0)$$
$$\bar{\sigma}_3 = (\sigma_1, \sigma_2, \sigma_3+\sigma_5, \sigma_4) = (W, W+3, 0, 0)$$
$$\bar{\sigma}_4 = (\sigma_1, \sigma_2, \sigma_3, \sigma_4+\sigma_5) = (W, W+3, 0, 0)$$

Subproblem 1

$$F_1 = \bar{\sigma}_1 - D_1 = (0,\ W+2,\ -2,\ -3)$$

$$\theta_1^* = t_1 = W+2\ (>0)$$

$$y_1^* = (1,\ 1,\ 0,\ 0)^T$$

Subproblem 2

$$F_2 = \bar{\sigma}_2 - D_2 = (W-3,\ 3,\ -1,\ -2)$$

$$\theta_2^* = t_2 = W\ (>0)$$

$$y_2^* = (1,\ 1,\ 0,\ 0)^T$$

Subproblem 3

$$F_3 = \bar{\sigma}_3 - D_3 = (W-4,\ W,\ -W,\ -1)$$

$$\theta_3^* = t_3 = W-4\ (>0)$$

$$y_3^* = (1,\ 1,\ 1,\ 0)^T$$

Subproblem 4

$$F_4 = \bar{\sigma}_4 - D_4 = (W-3,\ W+1,\ -2,\ -W)$$

$$\theta_4^* = t_4 = W-2\ (>0)$$

$$y_4^* = (1,\ 1,\ 0,\ 1)^T$$

$$\theta = \underset{i}{\mathrm{Max}}\ \theta_i^* = \theta_1^*$$

The vector to enter the basis is y_1^*. The corresponding column to be introduced into the basis is given by Equation (3.27):

$$H_1^* = \begin{bmatrix} y_1^* \\ \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The vector to leave the basis must now be determined.

$$\alpha = B_0^{-1} \; H_1^* = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Now, calculate

$$\underset{i}{\text{Min}} \; \beta_i/\alpha_i, \; \alpha_i > 0 = \text{Min} \; (1/1, \; 1/1, \; 0/1, \; 0/1, \; 0/1) = 0$$

Any of three vectors - S_3, S_4 or S_5 - can be chosen to leave the basis. If S_3 is chosen, the new ordered list of basic variables is then: $V_1^{i=1}$, $V_2^{i=2}$, $V_3^{i=1}$, S_4 and S_5.

The inverse of the new basis, $\hat{B}_0^{-1}$, must be now calculated. Elementary matrices [6] are used for this purpose.*

$$T_1 = \begin{bmatrix} 1 & & -1 & & \\ & 1 & -1 & & \\ & & 1 & & \\ & & -1 & 1 & \\ & & -1 & & 1 \end{bmatrix}$$

$$\hat{B}_0^{-1} = T_1 \; B_0^{-1} = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 0 & -1 \end{bmatrix}.$$

The new β vector is then

$$\hat{\beta} = T_1 \beta = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

* Throughout this example, blank entries in the elementary matrices correspond to zeros.

and, finally,

$$\hat{D}_B = (W,\ W+3,\ W+1,\ 0,\ 0)$$
$$\hat{\sigma} = \hat{D}_B\ \hat{B}_0^{-1} = (W,\ 1,\ W+2,\ 0,\ 0)$$

The next four iterations proceed in a similar fashion. At the end of the fifth iteration the situation is the following:

Ordered list of basic variables : $V_6^{i=3},\ V_7^{i=1},\ V_3^{i=1},\ V_4^{i=1},\ V_5^{i=1}$

$$T_5 = \begin{bmatrix} 1 & 1 & & & \\ & 1/2 & & & \\ & 1/2 & 1 & & \\ & -1/2 & & 1 & \\ & -1/2 & & & 1 \end{bmatrix}$$

$$\hat{B}_0^{-1} = T_5 B_0^{-1} = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 1/2 & 1/2 & 1/2 & 1/2 & -1 \\ -1/2 & 1/2 & -1/2 & -1/2 & 1 \\ 1/2 & -1/2 & 1/2 & -1/2 & 0 \\ 1/2 & -1/2 & -1/2 & 1/2 & 0 \end{bmatrix}$$

$$\hat{\beta} = T_5 \beta = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{D}_B = (W+1,\ W+6,\ W+1,\ W+2,\ W+3)$$
$$\hat{\sigma} = \hat{D}_B\ \hat{B}_0^{-1} = (4,\ 1,\ 2,\ 3,\ W-4)$$

Sixth iteration

$$\bar{\sigma}_1 = (W, 1, 2, 3)$$
$$\bar{\sigma}_2 = (4, W-3, 2, 3)$$
$$\bar{\sigma}_3 = (4, 1, W-2, 3)$$
$$\bar{\sigma}_4 = (4, 1, 2, W-1)$$

Subproblem 1

$$F_1 = \bar{\sigma}_1 - D_1 = (0, 0, 0, 0)$$

$$\theta_1^* = t_1 = 0$$
$$y_1^* = (0, 0, 0, 0)^T$$

Subproblem 2

$$F_2 = \bar{\sigma}_2 - D_2 = (1, -3, 1, 1)$$

$$\theta_2^* = t_2 = -3 + 1 + 1 + 1 = 0$$
$$y_2^* = (0, 0, 0, 0)^T$$

Subproblem 3

$$F_3 = \bar{\sigma}_3 - D_3 = (0, -2, -2, 2)$$

$$\theta_3^* = t_3 = -2 + 0 + 0 + 2 = 0$$
$$y_3^* = (0, 0, 0, 0)^T$$

Subproblem 4

$$F_4 = \bar{\sigma}_4 - D_4 = (1, -1, 0, -1)$$

$$\theta_4^* = t_4 = -1 + 1 + 0 + 0 = 0$$
$$y_4^* = (0, 0, 0, 0)^T$$

Then: $\theta = \underset{i}{\text{Max}}\ \theta_i^* = 0$.

The optimal solution has been found. The value of this solution is

$$\beta = \begin{bmatrix} V_6^{i=3} \\ V_7^{i=1} \\ V_3^{i=1} \\ V_4^{i=1} \\ V_5^{i=1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The interpretation of this solution is a fairly simple matter. As $V_6^{i=3} = V_3^{i=1} = 1$, the y_i^* vectors generated in the fourth ($V_6^{i=3}$) and first ($V_3^{i=1}$) iterations respectively provide the solution to the problem. Therefore,

$$V_6^{i=3} = \begin{bmatrix} \xi_{31} \\ \xi_{32} \\ \xi_{33} \\ \xi_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} ,$$

and

$$V_3^{i=1} = \begin{bmatrix} \xi_{11} \\ \xi_{12} \\ \xi_{13} \\ \xi_{14} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} ,$$

or

$$\xi_{11} = \xi_{12} = 1$$
$$\xi_{33} = \xi_{34} = 1 \ .$$

The solution to the LP is integer and therefore also a solution to the 2-median problem. The optimal 2-median is then $\overline{X}_2 = \{x_1, x_3\}$, with vertex x_2 allocated to median x_1 and vertex x_4 allocated to median x_3. The cost of this optimal solution is $\sigma(\overline{X}_2) = 2$, as it can be easily verified from the distance matrix of Figure 3.3. It is important to note that this matrix is not symmetrical and that the solution given above is optimal only for the direction i(row) → j(column). That is, this is an optimal solution only if customers are served from facilities, as in the case of depots supplying customers in a distribution network. The optimal solution for the direction j→i - if, for example, the facilities are schools to which students must travel - is entirely different from the one above and can be obtained from the transpose of the matrix of Figure 3.3.

3.4.2 Computational results

Some computational experience with the decomposition algorithm is reported in [41]. The algorithm is also independently assessed in the present section of this thesis. The examples used for this purpose are mainly from networks whose data were randomly generated, with unit weights given to all vertices. A description of how the data were generated, and the actual data corresponding to each of the randomly generated networks used to test the algorithm are given in the appendix. Where examples taken from other sources were used, their origin is clearly indicated in the appropriate table.

The computational results of this section are shown in Tables 3.1 to 3.3. The decomposition algorithm was tested in a CDC 6400 computer, and initially single and double precision versions of the code were used. In addition, the effect of using a random initial basic feasible solution was compared with the possibility of using the Teitz and Bart heuristic method [98] for obtaining the initial basic solution for the algorithm.

It was initially thought that the lack of convergence of the decomposition algorithm might be overcome through greater accuracy in the computations. Double precision and reinvertion techniques (see

Orchard-Hays [84]) are possible ways to obtain greater accuracy for this particular algorithm. Unfortunately the results produced when double precision was used were very discouraging. Consequently, attempts to solve the algorithm's convergence problems through greater accuracy were subsequently dropped. The results shown in Tables 3.1 to 3.3 correspond to the single precision version of the code.

In Table 3.1, results are shown for networks ranging from 5 to 33 vertices, and for a wide range of values of p. Some of the problems were also tested with the initial basic solution obtained from the Teitz and Bart heuristic method, and the corresponding results are shown in Table 3.2. Finally, in Table 3.3 the no heuristics option is compared to the heuristics one.

The examination of Tables 3.1 to 3.3 clearly shows that as the size of the network increases (and more often for the smaller values of p) the algorithm fails to converge after the maximum allowed 1000 iterations. It can be also observed from these tables that when the algorithm converges to an optimal solution, this solution is integer in the vast majority of the cases. This is in line with the fact that the LP relaxation of the p-median problem usually produces all-integer solutions.

The lack of convergence is the only drawback of the decomposition formulation, but it is unfortunately a very serious one. This is not made very clear in [41]. The results shown in this section, however, indicate that the lack of convergence prevents the algorithm from being used as a standard technique to solve the p-median problem.

When the algorithm converges the method is very fast and requires less computer core when compared, for example, with the general LP formulation. Whereas it took 70.69 CDC 6400 seconds to find the fractional LP solution (for p = 3) of the network of Figure 3.1 through the general formulation, the same example was solved in only 0.38 seconds when the decomposition formulation was used.

The lack of convergence of this formulation is due to its very degenerate nature. This is discussed in Section 3.4.3. It is interesting to note that sometimes the optimal solution is reached but not "recognized" as such by the algorithm. Refer for example to the p = 1 and p = 4 test cases of the 30-vertex network of Table 3.1. In both cases the solution obtained at some iteration before the 1000th was optimal, but the algorithm failed to recognize the optimality of these solutions.

The use of a heuristic initial basic feasible solution

In Table 3.3 the performance of the algorithm is compared for two different initial basic feasible solutions. It is perhaps surprising that convergence was obtained more consistently when a random (and usually worse) initial solution was used. This was the case for p = 2 and p = 3 in the 15-vertex network, p = 2, p = 8 and p = 9 in the 20-vertex network and p = 1 in the 25-vertex network. The reverse never occurred.

From the data of Table 3.3 it looks as though the closer the initial solution is to the optimal, the less likely is the algorithm to converge. On the other hand, when convergence occurs in both cases, the number of iterations it takes to reach the optimal solution does not follow a discernible trend. Sometimes convergence is quicker when the heuristic initial solution is used, sometimes the opposite is true. The use of a random initial solution appears, therefore, to be the best option concerning the choice of an initial basic feasible solution for the algorithm.

Computing times

Computing times increase with _n_, but for a given value of _n_ the number of seconds per iteration remains practically unchanged as _p_ increases, decreasing only slightly as _p_ approaches _n_ for the

Table 3.1 - Random Initial Solution

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in+ Seconds
5++	2	Yes	2	I	3.0	Yes	0.03
6++	2	Yes	10	F	14.0	Yes	0.07
9++	3	Yes	26	I	6.0	Yes	0.23
10*	1	Yes	31	I	3446.0	Yes	0.35
10	2	Yes	42	I	2049.0	Yes	0.47
10	3	Yes	32	I	1524.0	Yes	0.33
10	4	Yes	26	I	1187.0	Yes	0.29
10	5	Yes	23	I	882.0	Yes	0.25
10	6	Yes	24	I	579.0	Yes	0.26
10	7	Yes	11	I	294.0	Yes	0.12
10	8	Yes	7	I	163.0	Yes	0.09
10	9	Yes	2	I	75.0	Yes	0.06
10	10	Yes	2	I	0.0	Yes	0.06
10**	1	Yes	17	I	79.0	Yes	0.20
10	2	Yes	28	I	47.0	Yes	0.32
10	3	Yes	36	F	35.5	Yes	0.38
10	4	Yes	9	I	26.0	Yes	0.10
10	5	Yes	11	I	18.0	Yes	0.12
10	6	Yes	9	I	12.0	Yes	0.11
10	7	Yes	6	I	8.0	Yes	0.09
10	8	Yes	6	I	5.0	Yes	0.09
10	9	Yes	5	I	2.0	Yes	0.08
10	10	Yes	4	I	0.0	Yes	0.08

+ CPU Time, in CDC 6400 seconds

++ Test case provided by A.W. Neebe (see Appendix)

* Example from Revelle and Swain [90, p. 38]

** Example from Garfinkel et al. [41, p. 231]

Table 3.1 (cont'ed) - Random Initial Solution

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in Seconds+
15	1	Yes	34	I	809.0	Yes	0.69
15	2	Yes	222	I	412.0	Yes	4.75
15	3	Yes	171	I	294.0	Yes	3.82
15	4	Yes	172	I	215.0	Yes	3.65
15	5	Yes	160	I	150.0	Yes	3.31
15	6	Yes	483	I	113.0	Yes	10.10
15	7	Yes	63	I	93.0	Yes	1.23
15	8	Yes	36	I	74.0	Yes	0.65
15	9	Yes	26	I	57.0	Yes	0.50
15	10	Yes	18	I	41.0	Yes	0.39
20	1	Yes	41	I	1159.0	Yes	1.31
20	2	Yes	387	I	724.0	Yes	13.86
20	3	No	1000	-	523.0	No	34.69
20	4	No	1000	-	511.0	No	34.75
20	5	No	1000	-	476.0	No	34.92
20	6	No	1000	-	392.0	No	35.72
20	7	No	1000	-	356.0	No	34.96
20	8	Yes	465	I	199.0	Yes	15.79
20	9	Yes	132	I	175.0	Yes	4.03
20	10	Yes	129	I	151.0	Yes	4.20

+ CPU Time, in CDC 6400 seconds

Table 3.1 (cont'ed) - Random Initial Solution

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in Seconds$^+$
25	1	Yes	77	I	1352.0	Yes	3.41
25	2	No	1000	F	980.50	No	53.02
25	3	No	1000	-	732.0	No	52.16
25	4	No	1000	-	790.0	No	52.68
25	5	No	1000	-	763.0	No	52.59
25	6	No	1000	F	411.33	No	52.07
25	7	No	1000	-	533.0	No	52.32
25	8	No	1000	-	415.0	No	52.23
25	9	No	1000	-	393.0	No	52.17
25	10	No	1000	-	354.0	No	51.72
25	15	No	1000	-	184.0	-	50.84
25	20	Yes	32	I	51.0	Yes	1.35
30	1	No	1000	-	1432.0	Yes	74.67
30	2	No	1000	-	987.0	No	73.95
30	3	No	1000	F	767.0	-	72.43
30	4	No	1000	-	610.0	Yes	72.62
30	5	Yes	692	I	516.0	Yes	49.52
30	6	Yes	403	I	438.0	Yes	27.75
30	7	No	1000	-	663.0	No	72.79
30	8	No	1000	-	641.0	No	72.76
30	9	No	1000	-	455.0	No	66.45
30	10	Yes	196	I	265.0	Yes	12.49
30	15	No	1000	-	286.0	No	67.74
30	20	Yes	520	I	93.0	Yes	32.15
30	25	Yes	16	I	41.0	Yes	0.81

+ CPU Time, in CDC 6400 seconds

Table 3.1 (cont'ed) - Random Initial Solution

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in[+] Seconds
33[++]	1	No	1000	-	37993.0	No	88.56
33	2	No	1000	-	17592.0	No	88.67
33	3	No	1000	-	14627.0	Yes	85.15
33	4	Yes	830	I	12363.0	Yes	70.19
33	5	Yes	699	I	10398.0	Yes	58.58
33	6	No	1000	-	8862.0	No	83.17
33	7	Yes	423	I	8119.0	Yes	35.12
33	8	Yes	408	F	7460.0	Yes	33.39
33	9	Yes	354	F	6846.0	Yes	26.78
33	10	Yes	454	I	6267.0	Yes	36.14
33	15	Yes	121	I	4314.0	Yes	8.28
33	20	Yes	45	I	2786.0	Yes	2.49
33	25	Yes	23	I	1564.0	Yes	1.27

+ CPU Time, in CDC 6400 seconds

++ Karg and Thompson 33 City Data [57, p. 244]

Table 3.2 - Initial Solution from Heuristics

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in^{+} Seconds
5^{++}	2	Yes	1	I	3.0	Yes	0.04
6^{++}	2	Yes	5	I	14.0	Yes	0.06
9^{++}	3	Yes	14	I	6.0	Yes	0.26
10*	1	Yes	41	I	3446.0	Yes	0.49
10	2	Yes	26	I	2049.0	Yes	0.35
10	3	Yes	10	I	1524.0	Yes	0.21
10	4	Yes	11	I	1187.0	Yes	0.21
10	5	Yes	13	I	882.0	Yes	0.22
10	6	Yes	27	I	579.0	Yes	0.32
10	7	Yes	14	I	294.0	Yes	0.21
10	8	Yes	6	I	163.0	Yes	0.10
10	9	Yes	4	I	75.0	Yes	0.09
10	10	Yes	4	I	0.0	Yes	0.11
10**	1	Yes	34	I	79.0	Yes	0.42
10	2	Yes	23	I	47.0	Yes	0.31
10	3	Yes	28	F	35.5	Yes	0.38
10	4	Yes	6	I	26.0	Yes	0.20
10	5	Yes	7	I	18.0	Yes	0.17
10	6	Yes	12	I	12.0	Yes	0.25
10	7	Yes	8	I	8.0	Yes	0.15
10	8	Yes	5	I	5.0	Yes	0.11
10	9	Yes	1	I	2.0	Yes	0.07
10	10	Yes	1	I	0.0	Yes	0.07

+ CPU Time, in CDC 6400 seconds (inclusive of time to perform heuristics)
++ Test case provided by A.W. Neebe (see Appendix)

* Example from Revelle and Swain [90,p.38]
** Example from Garfinkel et al. [41,p.231]

Table 3.2 (cont'ed) - Initial Solution from Heuristics

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in Seconds+
15	1	Yes	201	I	809.0	Yes	4.25
15	2	No	1000	-	412.0	Yes	21.21
15	3	No	1000	-	294.0	Yes	22.00
15	4	Yes	293	I	215.0	Yes	6.66
15	5	Yes	704	I	150.0	Yes	15.92
15	6	Yes	256	I	113.0	Yes	5.97
15	7	Yes	148	I	93.0	Yes	3.47
15	8	Yes	60	I	74.0	Yes	1.52
15	9	Yes	24	I	57.0	Yes	0.77
15	10	Yes	31	I	41.0	Yes	0.79
20	1	Yes	620	I	1159.0	Yes	23.03
20	2	No	1000	-	724.0	Yes	37.68
20	3	No	1000	-	518.0	Yes	37.51
20	4	No	1000	-	414.0	Yes	36.98
20	5	No	1000	-	353.0	No	37.32
20	6	No	1000	-	259.0	Yes	36.58
20	7	No	1000	-	230.0	No	37.42
20	8	No	1000	-	202.0	No	37.98
20	9	No	1000	-	175.0	Yes	35.50
20	10	Yes	275	I	151.0	Yes	10.31

+ CPU time, in CDC 6400 seconds (inclusive of time to perform heuristics)

Table 3.2 (cont'ed) - Initial Solution from Heuristics

Problem Size		Convergence		Nature of Solution	Objective Function		
n	p	Yes or No	No. of Iterations	[Integer (I) or Fractional (F)]	Value	Optimal?	Time in seconds +
25	1	No	1000	-	1352.0	Yes	54.60
25	2	No	1000	-	1027.0	No	56.59
25	3	No	1000	-	777.0	No	59.17
25	4	No	1000	-	556.0	Yes	56.41
25	5	No	1000	-	468.0	Yes	55.93
25	6	No	1000	-	387.0	Yes	50.79
25	7	No	1000	-	341.0	Yes	51.47
25	8	No	1000	-	303.0	No	51.88
25	9	No	1000	-	266.0	Yes	54.13
25	10	No	1000	-	237.0	No	53.76
25	15	No	1000	-	128.0	-	51.82
25	20	Yes	5	I	51.0	Yes	1.27

+ CPU time, in CDC 6400 seconds (inclusive of time to perform heuristics)

Table 3.3 - Comparison of random vs. heuristic initial solution

Problem Size		Convergence				Objective Function						Time in Seconds^{+}	
		No Heuristics		Heuristics		No Heuristics			Heuristics			No Heuristics	Heuristics^{++}
n	p	Yes or No	No. of Iterations	Yes or No	No. of Iterations	I or F	Value	Optimal?	I or F	Value	Optimal?		
5^{+++}	2	Yes	2	Yes	1	I	3.0	Yes	I	3.0	Yes	0.03	0.04
6^{+++}	2	Yes	10	Yes	5	F	14.0	Yes	I	14.0	Yes	0.07	0.06
9^{+++}	3	Yes	26	Yes	14	I	6.0	Yes	I	6.0	Yes	0.23	0.26
10*	1	Yes	31	Yes	41	I	3446.0	Yes	I	3446.0	Yes	0.35	0.49
10	2	Yes	42	Yes	26	I	2049.0	Yes	I	2049.0	Yes	0.47	0.35
10	3	Yes	32	Yes	10	I	1524.0	Yes	I	1524.0	Yes	0.33	0.21
10	4	Yes	26	Yes	11	I	1187.0	Yes	I	1187.0	Yes	0.29	0.21
10	5	Yes	23	Yes	13	I	882.0	Yes	I	882.0	Yes	0.25	0.22
10	6	Yes	24	Yes	27	I	579.0	Yes	I	579.0	Yes	0.26	0.32
10	7	Yes	11	Yes	14	I	294.0	Yes	I	294.0	Yes	0.12	0.21
10	8	Yes	7	Yes	6	I	163.0	Yes	I	163.0	Yes	0.09	0.10
10	9	Yes	2	Yes	4	I	75.0	Yes	I	75.0	Yes	0.06	0.09
10	10	Yes	2	Yes	4	I	0.0	Yes	I	0.0	Yes	0.06	0.11
10**	1	Yes	17	Yes	34	I	79.0	Yes	I	79.0	Yes	0.20	0.42
10	2	Yes	28	Yes	23	I	47.0	Yes	I	47.0	Yes	0.32	0.31
10	3	Yes	36	Yes	28	F	35.5	Yes	F	35.5	Yes	0.38	0.38
10	4	Yes	9	Yes	6	I	26.0	Yes	I	26.0	Yes	0.10	0.20
10	5	Yes	11	Yes	7	I	18.0	Yes	I	18.0	Yes	0.12	0.17
10	6	Yes	9	Yes	12	I	12.0	Yes	I	12.0	Yes	0.11	0.25
10	7	Yes	6	Yes	8	I	8.0	Yes	I	8.0	Yes	0.09	0.15
10	8	Yes	6	Yes	5	I	5.0	Yes	I	5.0	Yes	0.09	0.11
10	9	Yes	5	Yes	1	I	2.0	Yes	I	2.0	Yes	0.08	0.07
10	10	Yes	4	Yes	1	I	0.0	Yes	I	0.0	Yes	0.08	0.07

+ CPU time, in CDC 6400 seconds ++ Inclusive of time to perform heuristics +++ Test case provided by A.W. Neebe (see Appendix)

* Example from Revelle and Swain [90, p.38] ** Example from Garfinkel et al. [41, p.231]

Table 3.3 (cont'ed) - Comparison of random vs. heuristic initial solution

Problem Size		Convergence				Objective Function						Time in Seconds+	
		No heuristics		Heuristics		No Heuristics			Heuristics			No Heuristics	Heuristics++
n	p	Yes or No	No. of Iterations	Yes or No	No. of Iterations	I or F	Value	Optimal?	I or F	Value	Optimal?		
15	1	Yes	34	Yes	201	I	809.0	Yes	I	809.0	Yes	0.69	4.25
15	2	Yes	222	No	1000	I	412.0	Yes	-	412.0	Yes	4.75	21.21
15	3	Yes	171	No	1000	I	294.0	Yes	-	294.0	Yes	3.82	22.00
15	4	Yes	172	Yes	293	I	215.0	Yes	I	215.0	Yes	3.65	6.66
15	5	Yes	160	Yes	704	I	150.0	Yes	I	150.0	Yes	3.31	15.92
15	6	Yes	483	Yes	256	I	113.0	Yes	I	113.0	Yes	10.10	5.97
15	7	Yes	63	Yes	148	I	93.0	Yes	I	93.0	Yes	1.23	3.47
15	8	Yes	36	Yes	60	I	74.0	Yes	I	74.0	Yes	0.65	1.52
15	9	Yes	26	Yes	24	I	57.0	Yes	I	57.0	Yes	0.50	0.77
15	10	Yes	18	Yes	31	I	41.0	Yes	I	41.0	Yes	0.39	0.79
20	1	Yes	41	Yes	620	I	1159.0	Yes	I	1159.0	Yes	1.31	23.03
20	2	Yes	387	No	1000	I	724.0	Yes	-	724.0	Yes	13.86	37.68
20	3	No	1000	No	1000	-	523.0	No	-	518.0	Yes	34.69	37.51
20	4	No	1000	No	1000	-	511.0	No	-	414.0	Yes	34.75	36.98
20	5	No	1000	No	1000	-	476.0	No	-	353.0	No	34.92	37.32
20	6	No	1000	No	1000	-	392.0	No	-	259.0	Yes	35.72	36.58
20	7	No	1000	No	1000	-	356.0	No	-	230.0	No	34.96	37.42
20	8	Yes	465	No	1000	I	199.0	Yes	-	202.0	No	15.79	37.98
20	9	Yes	132	No	1000	I	175.0	Yes	-	175.0	Yes	4.03	35.50
20	10	Yes	129	Yes	275	I	151.0	Yes	I	151.0	Yes	4.20	10.31

+ CPU time, in CDC 6400 seconds

++ Inclusive of time to perform heuristics

Table 3.3 (cont'ed) - Comparison of random vs. heuristic initial solution

Problem Size		Convergence				Objective Function						Time in Seconds[+]	
		No heuristics		Heuristics		No Heuristics			Heuristics			No Heuristics	Heuristics[++]
n	p	Yes or No	No. of Iterations	Yes or No	No. of Iterations	I or F	Value	Optimal?	I or F	Value	Optimal?		
25	1	Yes	77	No	1000	I	1352.0	Yes	-	1352.0	Yes	3.41	54.60
25	2	No	1000	No	1000	F	980.50	No	-	1027.0	No	53.02	56.59
25	3	No	1000	No	1000	-	732.0	No	-	777.0	No	52.16	59.17
25	4	No	1000	No	1000	-	790.0	No	-	556.0	Yes	52.68	56.41
25	5	No	1000	No	1000	-	763.0	No	-	468.0	Yes	52.59	55.93
25	6	No	1000	No	1000	F	411.33	No	-	387.0	Yes	52.07	50.79
25	7	No	1000	No	1000	-	533.0	No	-	341.0	Yes	52.32	51.47
25	8	No	1000	No	1000	-	415.0	No	-	303.0	No	52.23	51.88
25	9	No	1000	No	1000	-	393.0	No	-	266.0	Yes	52.17	54.13
25	10	No	1000	No	1000	-	354.0	No	-	237.0	No	51.72	53.76
25	15	No	1000	No	1000	-	184.0	-	-	128.0	-	50.84	51.82
25	20	Yes	32	Yes	5	I	51.0	Yes	I	51.0	Yes	1.35	1.27

[+] CPU time, in CDC 6400 seconds

[++] Inclusive of time to perform heuristics larger values of n. Table 3.4 below shows the number of CDC 6400 seconds per iteration for several values of n. The data of Table 3.4 were obtained from Table 3.1.

Table 3.4 - CDC 6400 Seconds per iteration

Number of Vertices (n)	CDC 6400 Seconds per iteration
10	0.011
15	0.021
20	0.035
25	0.052
30	0.070
33	0.080

The obvious conclusion to be drawn from the table above is that if the convergence problems of the algorithm are solved, the decomposition formulation can be used to solve the LP relaxation of the p-median problem for practically any size of network, within a reasonable amount of computer time.

3.4.3 Degeneracy and the problems with convergence

The serious convergence problems experienced in the previous section are due to the very degenerate nature of the decompositon formulation. This is more intensely felt for p small in relation to n and n large, although the convergence of the algorithm is data dependent to some extent. This data dependency can be best observed in the 33-vertex network of Karg and Thompson [57], the computational results of which are shown in Table 3.1. For this particular network convergence occurred much more frequently than for the 20, 25 and 30-vertex networks shown in the same table.

The degenerate nature of the decomposition formulation can be readily understood from the nature of the initial LP solution, defined by the vector β_0 below. β_0 is given by

$$\beta_0 = B^{-1}b_0 = [P_p, 0]^T , \qquad (3.36)$$

where B is the initial basis of the master problem - either B_0 of Equation (3.31) or B_1 of Equation (3.33).

In the vector β_0 above p of its components are equal to one, and (n-p) are equal to zero. Exactly (n-p) basic variables are therefore equal to zero at the first iteration of the algorithm. This initial degeneracy is in fact maintained throughout the solution procedure, as shown in the next few paragraphs. It is degeneracy on such large scale that is responsible for the lack of convergence reported in 3.4.2.

In order to show how the algorithm progressses from an initial basic feasible solution to optimality, successive values of the β solution vector are shown in the following for a particular application of the algorithm. This application was to find the optimal 3-median of the network of Figure 3.1, after the variables ξ_{11} and ξ_{12} had been fixed to one. All other variables in the problem were free to assume any value between zero and one. $\xi_{11} = \xi_{12} = 1$ is in fact part of one of the six optimal solutions to the 3-median problem of Figure 3.1.

Recall that the LP solution of the original problem was fractional for p = 3. After making $\xi_{11} = \xi_{12} = 1$, the initial basic feasible solution for this problem was $X_3' = \{x_1, x_3, x_4\}$, with $\sigma(X_3') = 55$. It took then 15 iterations for the LP to converge to an all-integer solution with $\sigma(\overline{X}_3) = 36$.

Successive β_j' vectors, j = 1, ..., 15, are given below. It should be noted that vector β_j' differs from vector β_j in that its top entry corresponds to the value of the objective function at the end of the iteration. This value is omitted in vector β_j.

The initial β'_j (β'_0) corresponds to the initial basic feasible solution. It is given by

$$\beta'_0 = \begin{bmatrix} 55 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Except for the iteration in which the surplus variable was driven out of the initial basis B_1, the successive β'_j vectors were:

$$\beta'_1 = \beta'_2 = \beta'_3 = \beta'_4 = \beta'_5 = \beta'_6 = \beta'_7 = \beta'_8 = \begin{bmatrix} 39 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\beta'_9 = \begin{bmatrix} 38.00 \\ 0.29 \\ 0.29 \\ 0.14 \\ 0 \\ 0.14 \\ 0.14 \\ 0.29 \\ 0.29 \\ 0.71 \\ 0.71 \\ 0 \end{bmatrix} \qquad \beta'_{10} = \begin{bmatrix} 37.56 \\ 0.22 \\ 0.44 \\ 0.11 \\ 0 \\ 0.11 \\ 0.11 \\ 0.22 \\ 0.22 \\ 0.56 \\ 0.78 \\ 0.22 \end{bmatrix}$$

$$\beta'_{11} = \beta'_{12} = \beta'_{13} = \beta'_{14} = \beta'_{15} = \begin{bmatrix} 36 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Regarding the β'_j vectors shown above, a few points are worth mentioning:

1. Except for β'_9 and β'_{10}, the solution vectors remain very degenerate throughout the solution procedure;
2. Notice the very "stationary" nature of the algorithm, i.e. it usually takes a very large number of iterations for the solution vector to change;
3. Although the optimal solution was attained at the end of iteration 11, optimality was only "recognized" by the algorithm at the end of iteration 15.

Some comments regarding the convergence of the algorithm

It has already been said that the convergence problems experienced by the decomposition formulation are due to its very degenerate nature, and that the use of greater accuracy in the computations does not improve the algorithm's convergence.

An approach suggested by Beale* consists in calculating the vector of dual variables using the following expression:

$$\hat{\sigma} = (1 - \alpha)\sigma^{\phi} + \alpha\sigma^{N} , \tag{3.37}$$

where σ^N is given by Equation (3.32) ($\sigma^N = D_B B^{-1}$), σ^{ϕ} is the vector of dual variables computed in the previous iteration of the algorithm, and $\underline{\alpha}$ is a smoothing constant ($0 \leqq \alpha \leqq 1$).

It is now worthwhile to take up a point made in Section 3.3, when the maximization of the objective function θ_i of subproblem $\underline{i}$ was being discussed. At that opportunity it was mentioned that, for $t_i > 0$, θ_i is maximized if $y^*_{ii} = 1$, and, for all i ≠ j, if either

$$y^*_{ij} = \begin{cases} 1 & \text{if} \quad f_{ij} > 0 \\ 0 & \text{if} \quad f_{ij} \leqq 0 \end{cases} \tag{3.25}$$

or if

$$y^*_{ij} = \begin{cases} 1 & \text{if} \quad f_{ij} \geqq 0 \\ 0 & \text{if} \quad f_{ij} < 0 \end{cases} \tag{3.26}$$

In other words, when $f_{ij} = 0$ the expression

$$\theta_i = \sum_{j=1}^{n} f_{ij} y_{ij} \tag{3.21}$$

is maximized for any corresponding value of y_{ij}. The setting of y_{ij} to zero or one is therefore entirely arbitrary when $f_{ij} = 0$. The question that arises then is whether this property can be used to improve

* Private Communication

the convergence of the algorithm.

Refer back to the example solved by hand in 3.4.1 and suppose that at some stage

$$\mathrm{Max}_{1 \leq i \leq 4}\ \theta_i = \theta_1 ,$$

and that $f_{12} = f_{13} = f_{14} = 0$. The vector y_1^* to enter the basis can then be any of eight possibilities, in each of which the top entry is equal to one. The three remaining entries can be any of the 2^3 possible combinations of zeros and ones. It is possible to represent this vector y_1^* by

$$y_1^* = \begin{bmatrix} 1 \\ \delta \\ \delta \\ \delta \end{bmatrix} ,$$

with the δ's to be replaced by one of the possible eight combinations mentioned above.

It is important to emphasize that the use of a particular vector can improve the convergence of the algorithm if a situation similar to the one described above develops at a given stage of the solution procedure. The practical difficulty, however, is how to use opportunities of multiple choice in a consistent way so as to improve the convergence of the algorithm.

It should be finally said that, when the algorithm converges, the choice of one particular vector may have an influence on the number of iterations it takes for the LP to converge. Furthermore, if the problem has more than one optimal solution, the optimal solution actually obtained may be affected by the choice of the vector.

The facts described above were confirmed in practice when Equations (3.25) and (3.26) were used independently in separate runs of the decomposition formulation. It was then observed that convergence was obtained in different number of iterations when (3.26) was used instead of (3.25). Furthermore, for problems with multiple optimal solutions, the solutions produced when (3.26) was used were generally different from the solutions obtained through the use of (3.25).

3.5 Conclusions

In the vast majority of cases the linear programming relaxation produces integer solutions that are optimal solutions to the p-median problem itself.

Two formulations of the linear programming relaxation were studied in the present chapter, and both were found to have their limitations. The general formulation produces very large linear programmes and is therefore unsuitable for use in large-scale networks. The decomposition formulation often does not converge because of its very degenerate nature. The problems with convergence become particularly serious as the size of the network increases, and for values of $\underline{p}$ small in relation to $\underline{n}$.

Regarding the difficulties mentioned above it is felt that, while not much can be done in relation to the general formulation, there is room for improvement in the decomposition formulation. If the difficulties arising from the lack of convergence can be tackled, then this formulation, together with its embedding into branch-and-bound algorithms, can be used to solve the p-median problem for large-scale networks.

CHAPTER FOUR

BOUNDS FOR THE p-MEDIAN PROBLEM

4.1 Introduction

It is well known that the quality of the bounds used in tree search methods is a factor of vital importance in the efficiency of the method. Branch-and-bound algorithms so far developed for the p-median problem suffer from a lack of stronglower bounds, and for this reason they are not very efficient. On the other hand, although both formulations of the LP relaxation discussed in Chapter 3 can be embedded into branch-and-bound algorithms and used as bounds for the problem, their limitations prevent them from being effectively used in this context.

After a brief review of earlier work on bounds for the p-median problem, two new lower bounds are developed in the present chapter. One of the bounds is a graph-theoretical bound, based on shortest spanning trees and arborescences and other graphical properties of the p-median problem. The other bound is based on the dual of the LP relaxation of the problem, and a heuristic procedure has been developed to compute an exact bound.

Both the graph-theoretical and the dual bound perform substantially better than a third bound developed in [12](call this bound the shortest distance bound). It is in fact shown in a later section that the graph-theoretical bound dominates the shortest distance bound. As for the dual bound, it outperforms the graph-theoretical bound very consistently, especially for values of $\underline{p}$ small in relation to $\underline{n}$.

Computational results that allow a comparison of the three bounds mentioned above are presented at the end of the chapter.

4.2 Earlier work on bounds for the p-median problem

The first lower bounds developed for the p-median problem appeared in papers by Järvinen, Rajala and Sinervo [55] and El-Shaieb [30]. Christofides [12] developed the shortest distance bound, for a direct tree search algorithm he designed for the problem.

The bounds described in [12], [30] and [55] can be considered to belong to the same family of bounds. They use the same basic principles, but differ in details that take advantage of the type of search for which they were designed. A few words are said below on each of these bounds.

The branch-and-bound algorithm of Järvinen et al. is a "drop" algorithm. It starts with all facilities "open", and facilities are successively "closed" until exactly p facilities are left "open". The iterative process continues until all feasible solutions have been implicitly enumerated.

For the computation of the lower bound, assume that at a given stage r facilities, corresponding to vertices $x_{k1}, x_{k2}, \ldots, x_{kr}$, have been "closed" ($1 \leqq r < n-p$). There are then $(n-r)$ vertices left, from which p vertices must be chosen. It is possible to define two sets of vertices:

$$V^r = \{x_{k1}, x_{k2}, \ldots, x_{kr}\} \text{ and } V^{n-r} = \{x_{\ell 1}, x_{\ell 2}, \ldots, x_{\ell n-r}\} = \\ = V - V^r,$$

where V is the set of all vertices of the network. The corresponding sets of indices are

$$K = \{k_1, k_2, \ldots, k_r\} \text{ and } L = \{\ell_1, \ell_2, \ldots, \ell_{n-r}\}.$$

Now let v_j be the weight of vertex x_j, and define $D_{ij} = v_j d_{ij}$ to be the weighted shortest distance between vertices x_i and x_j. For every column $k \in K$ of matrix $D = [D_{ij}]$ it is possible to compute

$$s_k = \min_{i \in L} D_{ik}. \qquad (4.1)$$

On the other hand, for every column $\ell \in L$ of D,

$$s_\ell = \min_{\substack{i \in L \\ i \neq j}} D_{ij} \qquad (4.2)$$

can be calculated. The lower bound is then given by

$$LB(K) = S_K + S_L , \qquad (4.3)$$

where $S_K = \sum_{k \in K} s_k$, and S_L is the sum of the (n-r-p) smallest s_ℓ, $\ell \in L$.

The branch-and-bound algorithm developed by El-Shaieb uses a different concept. In his algorithm the tree branches represent assignments of sources (facilities) and destinations. Locations are added one at a time to either the source or the destination set to form the next branches. Each set of branches consists of two branches. One of the branches corresponds to adding a location to the source set, while the other branch corresponds to adding the same location to the destination set. At the end of each branch there is a node that contains the corresponding source and destination sets.

El-Shaieb developed two different lower bounds for his algorithm. If the first bound is used an optimal solution is produced after a larger number of iterations than if the second bound is used. The first bound, however, needs a small amount of computation per iteration and is reported to be more efficient for small values of _p_. The second bound is more efficient for the larger values of _p_.

The bounds proposed by El-Shaieb and Christofides can only be properly understood after a detailed description of the corresponding branch-and-bound algorithms. The algorithm of El-Shaieb will not be described here. The algorithm developed by Christofides is given in Chapter 5; a detailed description of the corresponding bound is therefore left for that chapter. However, as in a later section this bound is compared with the two bounds developed in the present chapter, its computation before the beginning of the tree search is described below.

Let $d = [d_{ij}]$ be the distance matrix of a n-vertex network whose vertices x_j have weights equal to v_j. Now set up a matrix $M = [m_{kj}]$, the j^{th} column of which contains all the vertices of the network arranged in ascending order of their shortest distance from vertex x_j. The first entry of column $\underline{j}$ corresponds to vertex x_j itself. Call $m_{\beta_j j}$ the second entry of column $\underline{j}$. A lower bound for the p-median problem is the sum of the (n-p) smallest products:

$$v_j \times d(x_j, m_{\beta_j j}) \tag{4.4}$$

over all vertices x_j of the network. In the product above $d(x_j, m_{\beta_j j})$ is the shortest distance between vertices x_j and $m_{\beta_j j}$.

Finally, a word should be said about the LP relaxation of Chapter 3. In addition to providing an optimal solution to the p-median problem when the procedure converges and the solution is all-integer, non-integer solutions to the LP can obviously be used as lower bounds for the p-median problem. The use of the two formulations of Chapter 3 as bounds in branch-and-bound algorithms is discussed in Chapter 5.

4.3 A Graph-Theoretical Bound

A graph-theoretical lower bound for the p-median problem is now developed. Shortest spanning trees and arborescences form the basis for the computation of this bound. For nonweighted networks further graph-theoretical properties are used to strengthen the bound.

The graph-theoretical bound has been developed for both nondirected and complete symmetrical (directed) networks. In most applications of the p-median problem, nondirected networks are sufficient to adequately represent the problem. The association of weights with the vertices of a nondirected network, however, is equivalent to transforming this network into a complete symmetrical one. As this thesis addresses itself to the more general case in which a weight v_j is associated with every vertex x_j of the network, complete symmetrical networks have been considered in the development of the graph-theoretical bound.

Trees, arborescences, shortest spanning trees and shortest spanning arborescences are defined in the next section. This is followed by the development of the graph-theoretical bound for nondirected, nonweighted networks. The bound is then generalized for weighted networks. Finally, the graph-theoretical bound is shown to dominate the shortest distance bound.

4.3.1 Trees, Arborescences, Shortest Spanning Trees and Shortest Spanning Arborescences

One of the most important concepts of graph theory is that of a tree. A tree can be either nondirected or directed, depending on the nature of the underlying graph. A nondirected tree is defined as follows [12].

Definition: A nondirected tree is a connected graph of n vertices and (n-1) links.

A directed tree is called an arborescence. It can be defined as follows [12]:

Definition: A directed tree is a directed graph without a circuit, for which the indegree of every vertex is equal to unity, except for one vertex (called the root of the tree), for which the indegree is zero.

If G = (X,A) is a nondirected graph of n vertices, then a spanning tree of G is defined as a partial graph of G which forms a tree that spans every vertex of the graph. A spanning arborescence rooted at r of a directed graph $G' = (X',A')$ is a spanning tree of the underlying nondirected graph $\overline{G}' = (X',\overline{A}')$, having the following properties [38]:

(i) Each vertex of G' other than r has just one arc of the arborescence directed toward it; and

(ii) No arc of the arborescence is directed towards r.

The shortest spanning tree of a graph is defined for a nondirected graph G when costs c_{ij} are associated with its links. It has obvious applications in cases where roads (gas pipelines, electric power lines, etc.) are to be used to connect n points in such a way as to minimize the total length of the road that has to be constructed. Several algorithms [64, 86] have been designed to construct the shortest spanning tree of a graph (network); the length of the shortest spanning tree is independent of the vertex at which its construction starts.

The corresponding concept for directed networks is called the shortest spanning arborescence. Unlike shortest spanning trees, shortest spanning arborescences depend on the root under consideration.

In [10], [24] and [104] general algorithms for the construction of the minimum shortest spanning arborescence of a network are given. Besides producing the minimum shortest arborescence, these algorithms may also be used to produce shortest spanning arborescences for any specified root. The method used in [10] is similar in several respects to the Hungarian method for the classical assignment problem [68, 69].

4.3.2 Shortest spanning trees as lower bounds for the p-median problem

It is now shown that, for nondirected, nonweighted networks, shortest spanning trees can be used to compute a lower bound for the p-median problem. This is done by stating a lemma and demonstrating a theorem, although the final result could have been derived from

Kruskal's algorithm to construct shortest spanning trees. The lemma and the theorem have been used because they lend themselves to an easier generalization of the results to weighted networks.

The lemma is very general, being valid for both nondirected and directed networks, and even when the costs associated with the arcs of the network do not conform to the triangularity condition of metric space. The theorem only applies to nondirected, nonweighted networks, but is later extended to weighted networks.

Before the theorem is proved, it is necessary to derive a relationship that arises when a network is divided into a number of subnetworks. This relationship is of fundamental importance for the demonstration of the theorem. For the sake of clarity it will be derived within the context of the p-median problem.

Suppose that the optimal p-median of a network has been found. The original network can be then divided into p subnetworks $N_j = (X_j, A_j)$. X_j is the set of vertices of subnetwork j, and comprises the j^{th} assigned median and the nonmedian vertices allocated to it. A_j is the corresponding set of arcs, comprising all arcs of the original network interconnecting the vertices in X_j. The only arcs of the original network not present in any of the sets A_j are the arcs of the original network that interconnect the newly formed subnetworks.

If the lengths of:

(i) The shortest spanning tree of the original network (call this length SST_{ON}), and of

(ii) The shortest spanning trees of each of the p subnetworks N_j (call these lengths SST_{Oj}, j = 1, ..., p) are computed, the following relationship holds:

$$SST_{ON} \leq \sum_{j=1}^{p} SST_{Oj} + \sum_{j=1}^{p-1} SL_j \quad , \qquad (4.5)$$

where $\sum_{j=1}^{p-1} SL_j$ is the sum of the (p-1) smallest arcs of the original network not in $\bigcup_{j=1}^{p} A_j$ that will transform the p newly formed subnetworks into a connected network.

Now let G' = (X',A') be a graph (directed or not), every vertex of which is defined to be either a source or a sink. Allocate each sink vertex $x \in X'$ to a unique source vertex $y(x) \in X'$. Form partial subgraphs T' of G' by adding every arc on the shortest path from x to y(x), for all sink vertices x. If more than one shortest path from a given sink x to the corresponding source y(x) exists, choose only one such path. Then

Lemma - There is always a choice of a shortest path for each sink vertex $x \in X'$ for which the corresponding T' is a tree [12].

It is important to note that the lemma is valid for both directed and nondirected graphs. Furthermore, as no relationship related to metric spaces is assumed, the lemma is valid even for graphs whose arcs do not conform to the triangularity condition of metric space.

Corollary - Let SST_{0j} be the length of the shortest spanning tree of one of the p subnetworks into which a nondirected, nonweighted network N = (X,A) can be divided once the optimal p-median is known. Then SST_{0j} is a lower bound on the sum of shortest distances from the median x_{0j} of N_j to the vertices allocated to it.

This follows immediately from the lemma above. The lemma guarantees that the subnetwork, formed when nonmedian vertices of N_j are connected to the median x_{0j} through the corresponding shortest paths, can be constructed so that a tree is formed. Call the length of this tree ST_j. The shortest spanning tree of N_j has a length that is, by definition, shorter than or equal to the length of any other spanning tree of N_j. Therefore

$$ST_j \geqq SST_{0j} \; . \qquad (4.6)$$

On the other hand, the sum of the shortest distances from x_{0j} to the vertices allocated to it is greater than or equal to ST_j (some arcs can be counted twice or more when the sum of shortest distances is computed). It follows then that

$$\sum_i d(x_i, x_{0j}) \geqq ST_j \ , \qquad (4.7)$$

where the x_i's are the nonmedian vertices of N_j. If (4.6) and (4.7) are combined, it is possible to write

$$\sum_i d(x_i, x_{0j}) - SST_{0j} \geqq 0 \ . \qquad (4.8)$$

The theorem can now be proved.

Theorem - Let N = (X,A) be a nondirected, nonweighted network for which the optimal p-median must be found. A lower bound on the value of the objective function of the problem is the length of the shortest spanning tree of the network, minus the shortest spanning tree's (p-1) longest links.

Proof - Suppose the optimal p-median was found and that the original network was divided into the p subnetworks $N_j = (X_j, A_j)$ defined above. Equation (4.8) can be applied to each of the p subnetworks:

$$\left.\begin{array}{l} \displaystyle\sum_{\substack{i1 \in \\ \text{Subnetwork (1)}}} d(x_{i1}, x_{01}) - SST_{01} \geqq 0 \\ \displaystyle\sum_{\substack{i2 \in \\ \text{Subnetwork (2)}}} d(x_{i2}, x_{02}) - SST_{02} \geqq 0 \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \displaystyle\sum_{\substack{ip \in \\ \text{Subnetwork (p)}}} d(x_{ip}, x_{0p}) - SST_{0p} \geqq 0 \end{array}\right\} \qquad (4.9)$$

Adding the p inequalities above it follows that

$$\sum_{j=1}^{p} \sum_{\substack{ij \in \\ \text{subnetwork (j)}}} d(x_{ij}, x_{0j}) - \sum_{j=1}^{p} SST_{0j} \geqq 0 \qquad (4.10)$$

Now refer back to Equation (4.5). It can be re-arranged as

$$\sum_{j=1}^{p} SST_{0j} \geqq SST_{0N} - \sum_{j=1}^{p-1} SL_j \qquad (4.11)$$

Substituting (4.11) into (4.10) it follows that

$$\sum_{j=1}^{p} \sum_{\substack{ij\in \\ \text{Subnetwork }(j)}} d(x_{ij}, x_{0j}) \geqq SST_{ON} - \sum_{j=1}^{p-1} SL_j \ . \tag{4.12}$$

Before the optimal p-median is found, however, it is not known which are the (p-1) SL_j's that satisfy equations (4.11) and (4.12). For the computation of the lower bound, the worst possible case is that these (p-1) SL_j's are the (p-1) longest arcs of the shortest spanning tree of the original network. Let LL_j be the arcs of this shortest spanning tree, ranked in order of decreasing arc length. Since $\sum_{j=1}^{p-1} LL_j \geqq \sum_{j=1}^{p-1} SL_j$, it is finally possible to write

$$\sum_{j=1}^{p} \sum_{\substack{ij\in \\ \text{subnetwork }(j)}} d(x_{ij}, x_{0j}) \geqq SST_{ON} - \sum_{j=1}^{p-1} LL_j \ . \tag{4.13}$$

The left-hand side of Equation (4.13) is the value of the objective function of the p-median problem. The theorem is thus proved.

Since shortest spanning trees are only defined for nondirected networks, the theorem is only valid for nondirected, nonweighted networks. Its extension to weighted networks, through the use of shortest spanning arborescences, is given in 4.3.4.

4.3.3 Further graph-theoretical properties and a stronger lower bound

In the previous section it was shown that shortest spanning trees can provide a lower bound for the p-median problem. This bound can be improved, as shown in the remainder of this section.

Consider again the p subnetworks N_j defined in the previous section. For each of the subnetworks construct the spanning tree defined in the lemma of that section. If δ_j is the degree of median x_{0j} of spanning tree ST_j, it is easy to see that

$$\sum_{j=1}^{p} \delta_j \leqq n - p \tag{4.14}$$

In general it can be said that, when $\sum_{j=1}^{p} \delta_j < n - p$, the lengths of a number of arcs (say β) are counted at least twice when the value of the objective function of the p-median problem is computed. The value of β is given by

$$\beta = (n-p) - \sum_{j=1}^{p} \delta_j \, . \qquad (4.15)$$

Refer, for example, to Figure 4.1 , in which the optimal 3-median of a 10-vertex network is shown through the spanning trees defined in the lemma of Section 4.3.2.

The 3 medians are vertices x_1, x_{10} and x_5. In Figure 4.1c vertex x_6 is two arcs away from x_5, the median vertex to which it has been allocated. Consequently, the length of the arc $\overline{x_7x_5}$ is counted twice when the value of the objective function of the problem is computed.

For the example of Figure 4.1 $(n-p) = 7$, and $\sum_{j=1}^{3} \delta_j = 1 + 2 + 3 = 6 < 7$. Then $\beta = 7 - 6 = 1$, and the length of one arc ($\overline{x_7x_5}$ in this particular case) is counted twice when the value of the objective function of the problem is computed.

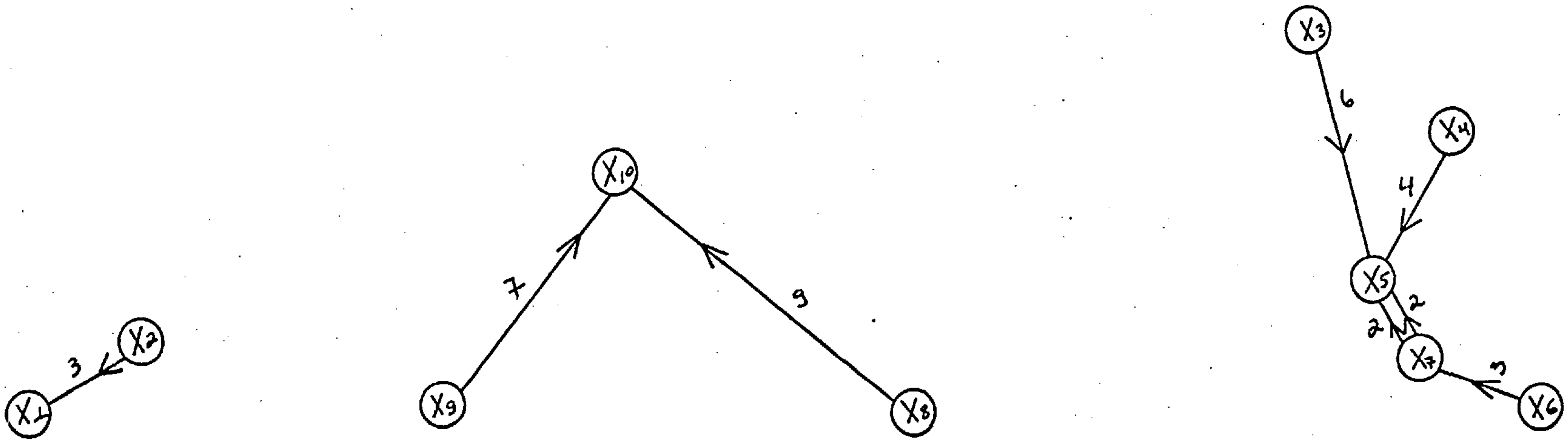
It is possible to use the properties described above to strengthen the bound developed in 4.3.2. If R_j is the sum of the lengths of the arcs of ST_j (subnetwork N_j) that must be added to the spanning tree's length in order to obtain the sum of shortest distances between source and sinks in N_j, it follows that

$$\sum_i d(x_i, x_{0j}) = ST_j + R_j \, . \qquad (4.16)$$

Suppose now that it is possible to know that there are at least β_j arcs whose length ℓ_i is counted twice or more in the computation of the sum of shortest distances. Then

$$R_j \geqq \sum_{i=1}^{\beta_j} \ell_i \quad , \qquad (4.17)$$

and consequently,



(4.1a)

$\delta_1 = 1$

(4.1b)

$\delta_2 = 2$

(4.1c)

$\delta_3 = 3$

Figure 4.1

Optimal 3-median of a 10-vertex network (Garfinkel et al. [41, p.231])

$$\sum_i d(x_i, x_{0j}) \geqq ST_j + \sum_{i=1}^{\beta_j} \ell_i \; . \tag{4.18}$$

Finally, as $SST_{0j} \leqq ST_j$,

$$\sum_i d(x_i, x_{0j}) \geqq SST_{0j} + \sum_{i=1}^{\beta_j} \ell_i \; . \tag{4.19}$$

If Equation (4.19) is now applied to each of the p subnetworks N_j, and the p resulting inequalities are added together, the following is obtained:

$$OF(\overline{p}) = \sum_{j=1}^{p} \sum_{\substack{ij \in \\ \text{subnetwork } (j)}} d(x_{ij}, x_{0j}) \geqq \sum_{j=1}^{p} SST_{0j} + \sum_{j=1}^{p} \sum_{i_j=1}^{\beta_j} \ell_{ij}, \tag{4.20}$$

where $OF(\overline{p})$ is the value of the optimal solution of the p-median problem. Now, by replacing $\sum_{j=1}^{p} SST_{0j}$ in Equation (4.20) by its value in Equation (4.5), it follows that

$$OF(\overline{p}) \geqq (SST_{ON} - \sum_{j=1}^{p-1} LL_j) + \sum_{j=1}^{p} \sum_{i_j=1}^{\beta_j} \ell_{ij} \; . \tag{4.21}$$

Before the optimal p-median is known, however, it is not possible to know exactly how many arcs of the network, if any, are going to be counted more than once when the value of $OF(\overline{p})$ is computed. From an examination of the iegrees of the vertices of the original network it is possible to know, however, the minimum number of arcs that are going to be counted at least twice, and the corresponding minimum total length. The following procedure is thus suggested:

Step 1. Calculate the degree δ_j of each of the vertices of the network for which the optimal p-median is being sought.

Step 2. Rank these degrees in descending order, and call the ranked degrees δ_{Rj}.

Step 3. Compute

$$\alpha = \sum_{j=1}^{p} \delta_{Rj} \; . \tag{4.22}$$

Step 4. (a) If $n - p \leq \alpha$, no improvement can be added to the bound of section 4.3.2;

(b) If $n - p > \alpha$, compute

$$\beta = (n-p) - \alpha \quad . \qquad (4.23)$$

Then add to the lower bound of section 4.3.2 the sum of the lengths of the β shortest arcs in the network. In the worst possible case at least the length of these β arcs will be counted twice in the computation of the value of $OF(\overline{p})$.

4.3.4 Generalization for weighted networks

The extension of the lower bound derived in 4.3.2 and 4.3.3 to weighted networks is straightforward. Recall that in Chapter 1 it was shown that weighted networks must be transformed into complete (directed) symmetrical networks before they can be handled. In this section it will be always assumed that such transformation has taken place.

The theorem of Section 4.3.2 can be readily extended to weighted networks. The theorem for weighted networks is:

Theorem - Let $N' = (X',A')$ be a complete symmetrical network for which the optimal p-median must be found. A lower bound on the value of the objective function of the problem is the length of the minimum shortest spanning arborescence of the network, minus the (p-1) longest arcs in this arborescence.

The proof is analogous to that of the theorem of Section 4.3.2, and for this reason will not be given here.

Note that graph-theoretical properties of the type discussed in 4.3.3 cannot be used to improve the bound of weighted networks. Since the indegree of every vertex of a complete symmetrical network of n vertices is equal to (n-1), it follows that $\beta \leq 0 \ \forall \ p$ (β is defined in the previous section). The bound provided by shortest spanning arborescences cannot therefore be improved in the case of weighted networks.

4.3.5 Dominance over the shortest distance bound

It is not difficult to prove that the graph-theoretical bound dominates the shortest distance bound. The proof will be limited to nondirected, nonweighted networks. The extension to weighted networks is straightforward and will not be given here.

Recall that the shortest distance bound is equal to the sum of the (n-p) smallest products

$$v_j \times d(x_j, m_{\beta_j j}) \qquad (4.24)$$

over all vertices x_j of the network. As only nonweighted networks will be considered, $v_j = 1 \; \forall \; j$ in the present discussion. On the other hand, $d(x_j, m_{\beta_j j})$ is the distance between vertex x_j and the vertex closest to it in the distance matrix of the network. It is obvious that $m_{\beta_j j}$ has to be directly connected to x_j through one of the links of the network.

It is interesting to note that the bound provided by shortest spanning trees is also the sum of (n-p) lengths of links between vertices of the network: The shortest spanning tree of a network has (n-1) links, and if (p-1) links are subtracted from it exactly (n-p) links are left. What remains to be proved is that each link used in the construction of the graph-theoretical bound is at least as long as the corresponding link used in the construction of the shortest distance bound.

In Kruskal's algorithm [64] for the shortest spanning tree, the links of the network must be ordered in ascending order of cost. Then, starting from the top of the list, links must be added to the initially disconnected set of vertices, provided that no circuit is formed when a new link is added to the existing set of links. A bound is obtained for the p-median problem after (n-p) links are selected in this way and their corresponding costs added to form the bound.

Now refer back to the shortest distance bound for nonweighted networks. This bound is also obtained by adding the costs of the (n-p) shortest links of the network. For this bound, however, there are no restrictions on the formation of circuits, and therefore every link used in the computation of the graph-theoretical bound is at least as long as the corresponding link used in the computation of the shortest distance bound. The graph-theoretical bound thus dominates the shortest distance bound.

4.4 A bound based on the dual of the linear programming relaxation of the problem

The dual of the linear programming relaxation of the p-median problem provides a very good lower bound for the problem. The difficulty in obtaining this bound is that, similarly to the primal, the dual is a very large linear programme. Any attempt to obtain the bound by actually solving the dual would lead to difficulties similar to those experienced when the primal was studied in Chapter 3 (see Section 4.4.1).

A heuristic procedure has been developed to generate approximate solutions to the dual LP. This procedure, which produces a True bound to the p-median problem, is a two-phase method. It takes advantage of the simple form of the dual objective function and of the special nature of its variables.

Very good bounds were obtained for the problem through this dual procedure. Computational results given in Section 4.5 compare the dual bound with both the shortest distance bound and the graph-theoretical bound.

The dual formulation is derived in the next section. Then a heuristic procedure to generate approximate solutions to the dual is discussed, and a detailed step-by-step description of the algorithm is given. Finally, computational results that allow the dual bound to be evaluated are given for a wide range of values of n and p.

4.4.1 The Dual Linear Programme

Recall the linear programming relaxation of the p-median problem. For the sake of convenience this formulation is repeated below. The symbols selected for the dual variables are indicated in brackets, alongside the corresponding primal constraints. The LP relaxation is

$$\text{Minimize } Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, \xi_{ij} \tag{4.25}$$

Subject to

$$\sum_{i=1}^{n} \xi_{ij} = 1 \;, \quad j = 1, \ldots, n \qquad [\sigma_i] \tag{4.26}$$

$$\sum_{i=1}^{n} \xi_{ii} = p \qquad [\sigma_{n+1}] \tag{4.27}$$

$$\xi_{ij} - \xi_{ii} \leqq 0 \;, \; i, j = 1, \ldots, n, \; i \neq j \qquad [\pi_{ij}] \tag{4.28}$$

$$\xi_{ij} \geqq 0 \;, \; i, j = 1, \ldots, n \tag{4.29}$$

The dual of this linear programme is

$$\text{Maximize } Z' = \sum_{i=1}^{n} \sigma_i + p\sigma_{n+1} \tag{4.30}$$

Subject to

$$\sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij} \leqq 0 \;\forall\; i \tag{4.31}$$

$$\sigma_j + \pi_{ij} \leqq d_{ij} \;\forall\; i, j, \; j \neq i \tag{4.32}$$

$$\pi_{ij} \leqq 0 \;\forall\; i, j, \; j \neq i \tag{4.33}$$

$$\sigma_i < = > 0 \;, \; i = 1, \ldots, n + 1 \tag{4.34}$$

As indicated by Equation (4.34), the σ_i's are unconstrained variables. This is so because they correspond to equality constraints in the primal LP. A closer examination of the problem, however, makes it possible to determine the true nature of these variables.

Refer to the primal problem. Since $d_{ij} \geqq 0 \; \forall \; i, j$, it is clear that Equation (4.26) can be replaced by

$$\sum_{i=1}^{n} \xi_{ij} \geqq 1 \; , \; j = 1, \ldots, n \tag{4.26a}$$

without loss of any optimal solution. It then follows that

$$\sigma_i \geqq 0 \; , \; i = 1, \ldots, n \; . \tag{4.34a}$$

On the other hand, if Equation (4.31) is re-arranged, the following is obtained:

$$\sigma_{n+1} \leqq - \sigma_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij} \; \forall \; i \; . \tag{4.31a}$$

In view of Equations (4.33) and (4.34a) it follows immediately that

$$\sigma_{n+1} \leqq 0 \; . \tag{4.34b}$$

The dual LP can be now re-written as

$$\text{DLP} \left\{ \begin{array}{lr} \text{Maximize } Z' = \sum_{i=1}^{n} \sigma_i + p\sigma_{n+1} & (4.30) \\ \text{Subject to} & \\ \qquad \sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij} \leqq 0 \; \forall \; i & (4.31) \\ \qquad \sigma_j + \pi_{ij} \leqq d_{ij} \; \forall \; i, j, j \neq i & (4.32) \\ \qquad \pi_{ij} \leqq 0 \; \forall \; i, j, j \neq i & (4.33) \\ \qquad \sigma_i \geqq 0, \; i = 1, \ldots, n & (4.34a) \\ \qquad \sigma_{n+1} \leqq 0 & (4.34b) \end{array} \right.$$

It is easy to see that the optimal value of Z' (and therefore any value below it) is a lower bound for the p-median problem. Since the primal LP is a relaxation of the problem, if $\overline{X}_p$ is the value of the optimal solution of the p-median problem it follows that

$$\overline{X}_p \geqq \text{Min } Z \, . \tag{4.35}$$

On the other hand, Min Z = Max Z' (Theorem of Duality [54]), and

$$\overline{X}_p \geqq \text{Max } Z' \, . \tag{4.36}$$

Consequently, if the dual LP is solved, the optimal value of its objective function (or any value below the optimal) is a lower bound for the p-median problem (call this bound the dual bound).

The only difficulty in computing the dual bound is that, similarly to the primal, the dual is a very large linear programme, with (n^2+1) variables and $\underline{n^2}$ constraints.

However, as the interest in the dual is limited to obtaining a bound for the p-median problem, if a heuristic procedure for solving the dual can be shown to yield solutions close to the optimal in an efficient way, this procedure can be used to compute lower bounds for the problem.

Fortunately, the simple form of the dual objective function (Equation 4.30), plus the special nature of its variables, readily suggest such a procedure. The procedure has proved to be computationally efficient, and can therefore be embedded into branch-and-bound algorithms designed to solve the problem.

4.4.2 A heuristic method to solve the DLP

In the dual linear programme given by equations (4.30) through (4.34b), since $\sigma_i \geqq 0$, $i = 1, \ldots, n$, and $\sigma_{n+1} \leqq 0$, Z' can be maximized if the positive σ_i's are chosen as large as possible, while the absolute value of σ_{n+1} is kept as small as possible, provided that equations (4.31), (4.32) and (4.33) are always satisfied, i.e. that the dual remains feasible throughout the procedure.

Now recall Equation (4.31a) given by

$$\sigma_{n+1} \leqq - \sigma_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij} \ \forall \ i. \tag{4.31a}$$

For the dual to remain feasible throughout the procedure, given a set of values for the σ_i's and π_{ij}'s, σ_{n+1} must take a value that satisfies the <u>n</u> constraints of (4.31a). That is

$$\sigma_{n+1} \leqq \underset{i}{\text{Min}} \ (- \sigma_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij}). \tag{4.37}$$

On the other hand, if the objective is to maximize Z', for a given set of σ_i's the absolute values of the π_{ij}'s must be the smallest possible values that will satisfy equations (4. 32) and (4.33). This can be achieved by making

$$\begin{cases} \pi_{ij} = 0 \text{ if } \sigma_j \leqq d_{ij} \\ \pi_{ij} = - (\sigma_j - d_{ij}) \text{ if } \sigma_j > d_{ij} \end{cases}$$

Or, combining the two conditions above into one equation

$$\pi_{ij} = - \text{Max} \ (0, \sigma_j - d_{ij}) \ \forall \ i, j, j \neq i \ . \tag{4.38}$$

From the above it can be seen that, given the distance matrix $[d_{ij}]$ of the network, and a set of values for the positive σ_i's, both the π_{ij}'s and σ_{n+1} can be determined in an optimal way with respect to maximizing Z'. The problem that remains is how to determine the initial σ_i's and, subsequently, how to modify these values in a stepwise fashion, so as to increase Z' to a value as close to the optimal solution of the dual as possible. An algorithm that performs these tasks is given below.

The detailed steps of the algorithm

The heuristic procedure given below is a two-phase method. The first phase is iterative, whereas the second phase is a one-pass algorithm.

The first phase of the procedure starts from a given set of σ_i's, and attempts to decrease the absolute value of σ_{n+1} by suitably decreasing some of the σ_i's. This is done in such a way that an increase is obtained in the value of Z' from one iteration of this phase to the following iteration. When such increase is no longer possible, the second phase of the algorithm is activated.

The second phase of the method is a one-pass algorithm, in which an attempt is made to increase each of the positive σ_i's individually, but without altering as a consequence the value of σ_{n+1} obtained at the end of phase 1. This second phase starts from the values of the σ_i's at the end of phase 1, and terminates after all n σ_i's have been tentatively increased.

The detailed steps of the algorithm are now given.

Phase 1

Step 1. Choose initial values for each of the n positive σ_i's (the choice of these initial values is discussed in 4.4.3). Then make k = 1 and go to Step 2 below.

Step 2. For each i, i = 1, ..., n, compute

$$\tau_i = \sigma_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} \pi_{ij} , \qquad (4.39)$$

with the π_{ij}'s computed as per Equation (4.38). Then compute the initial value of the bound (Z') and go to Step 3 below.

Step 3. Find the largest and next-to-largest values of τ_i. Call these values τ_{MX} and τ_{NMX} respectively. Set

$$\sigma_{n+1} = - \tau_{MX} .$$

Step 4. Attempt to increase σ_{n+1} from $- \tau_{MX}$ to $- \tau_{NMX}$ by suitably decreasing the necessary σ_i's.* Start by computing, for the _i_ corresponding to $\tau_i = \tau_{MX}$ (ties broken arbitrarily), the set J defined by

$$J = \{\sigma_j | (\sigma_j - d_{ij}) \geqq (\tau_{MX} - \tau_{NMX})\} . \qquad (4.40)$$

Step 5. If J = ∅ go to Phase 2 of the algorithm, as Z' cannot be increased any further in Phase 1. Otherwise compute

$$S_{Max} = \underset{\sigma_j \in J}{MAX} \sum_{i=1}^{n} Max(0, \sigma_j - d_{ij}). \qquad (4.41)$$

Step 6. Decrease the σ_j corresponding to S_{MAX} by $(\tau_{MX} - \tau_{NMX})$. Then recalculate the τ_i's of Equation (4.39), given the decrease in σ_j defined in the present step.

Step 7. After recalculating the τ_i's check if, for any _i_, $\tau_i = \tau_{MX}$ (This is only possible if a tie occurred in the computation of τ_{MX} in Step 3). If so, repeat Steps 4 through to 7 in an attempt to change this τ_i to τ_{NMX}. Otherwise go to Step 8 below.

Step 8. Compute G_k, the gain of iteration _k_:

$$G_k = p(\tau_{MX} - \tau_{NMX}) - \sum_{\substack{\sigma_j\text{'s decreased} \\ \text{in Step 6}}} (\sigma_j^O - \sigma_j^N) , \qquad (4.42)$$

* Note that the above defined increase in σ_{n+1} is only worthwhile if the sum of the necessary decreases in the σ_i's is offset by a corresponding increase in the value of the product $p(\tau_{MX} - \tau_{NMX})$.

where σ_j^O is the value of σ_j before the start of iteration <u>k</u> and σ_j^N its value at the end of the iteration. Then go to Step 9 below.

<u>Step 9</u>. (a) If $G_k > 0$ increase Z' by G_k, make $k = k + 1$ and go to Step 3 for a new iteration of Phase 1 of the algorithm;

(b) If $G_k \leqq 0$, go to Phase 2 of the algorithm.

<u>Phase 2</u>

<u>Step 10</u>. Compute the difference $(\tau_{MX}^1 - \tau_{NMX}^1)$, where τ_{MX}^1 and τ_{NMX}^1 are the values of τ_{MX} and τ_{NMX} at the end of Phase 1. Then make $j = 1$ and go to Step 11 below.

<u>Step 11</u>. Make

$$\sigma_j^N = \sigma_j^O + (\tau_{MX}^1 - \tau_{NMX}^1) \ . \qquad (4.43)$$

Then compute $\tau_i \ \forall \ i$ (Equation 4.39) , given the change in σ_j defined in the present step.

<u>Step 12</u>. (a) If, for any <u>i</u>, $\tau_i > \tau_{MX}^1$, make σ_j^O the permanent value of σ_j. Then make $j = j + 1$ and go to Step 13;

(b) If $\tau_i \leqq \tau_{MX}^1 \ \forall \ i$, make σ_j^N the permanent value of σ_j and increase the value of Z' by $(\tau_{MX}^1 - \tau_{NMX}^1)$. Then make $j = j + 1$ and go to Step 13 below.

<u>Step 13</u>. If $j \leqq n$ go to Step 11. Otherwise, terminate the algorithm. The final value of Z' is a lower bound for the p-median problem.

<u>4.4.3 The initialization of the heuristic procedure</u>

The final value of the dual bound depends to some extent on the initial values of the positive σ_i's. While there is great freedom of choice for these initial values when the bound is computed for the overall optimal solution to the problem, the choice is very restricted when some of the variables of the corresponding primal problem have known values. This is of special relevance when the bound is embedded into branch-and-bound algorithms, but a detailed discussion of the subject is left for the next chapter. The initialization discussed in the present section is therefore important only if interest is centred in obtaining a lower bound on the overall optimal solution of the p-median problem.

Three different starting rules are discussed below. Computational experience shows that while unreasonable starting values for the positive σ_i's may lead to useless bounds, none of the investigated starting rules always yield the best value for the bound. The corresponding results are summarized at the end of this section.

The procedure described in 4.4.2 is one in which the initial values of the positive σ_i's are decreased throughout the iterative phase of the algorithm. The algorithm must therefore start from values of σ_i expected to be larger than their respective values at the termination of the procedure.

An alternative procedure would be to start from small values for the σ_i's and build the bound by increasing these values in a stepwise fashion. This alternative procedure has not been investigated experimentally, given the satisfactory results obtained with the procedure described in 4.4.2, and its better suitability for embedding the bound into branch-and-bound algorithms.

<u>Starting Rules 1 and 2</u>

These two starting rules take advantage of a relationship developed by Diehr [22], which gives an approximate value for the summation of the positive σ_i's:

$$\sum_{i=1}^{n} \sigma_i = \overline{X}'_p + p[\overline{X}'_{p-1} - \overline{X}'_p] \, , \qquad (4.44)$$

where $\overline{X}'_p$ and $\overline{X}'_{p-1}$ are approximate solutions to the $\underline{p}$ and (p-1)-median problems respectively. Equation (4.44) is an approximation biased towards giving a value above the exact value of the summation, satisfying therefore the condition of providing initial values for the σ_i's above their expected final values.

The only difference between starting rules 1 and 2 is the way by which the individual σ_i's are obtained from the summation given by Equation (4.44). In starting Rule 1 the individual σ_i's are weighted according to the sum of distances in the corresponding column of the distance matrix of the network. In starting Rule 2 all initial σ_i's are equal to the result of the division of the summation of Equation (4.44) by the number of vertices of the network.

Starting Rule 3

Starting Rule 3 assumes that the final values of the σ_i's used to calculate the bound for the (p-1)-median problem are available prior to starting the procedure to find the dual bound for the p-median problem. The initial values of the σ_i's for calculating the dual p-median bound are then made equal to the final values of the σ_i's used to calculate the bound for the (p-1)-median problem. For $p = 1$ each initial σ_i is made equal to the average of the distances in the corresponding column of the distance matrix of the network.

Note that with starting Rule 3 the bound for the (p-1)-median problem must be available before the bound for the p-median problem can be calculated. It is therefore necessary to start by computing the bound from $p = 1$ if one wishes to use starting Rule 3 to compute lower bounds for successive values of p, starting from $p = p_{min}$ and going up to $p = p_{max}$. The computation of bounds for values of $p < p_{min}$ is a necessary "starting-up" procedure if Rule 3 is to be used.

Computational results

The results obtained with the dual bound are given in Table 4.1. Results for networks ranging from 10 to 50 vertices are shown in this table. As with the examples used to produce the computational results of Chapter 3, the data describing the randomly generated networks of Table 4.1 are given in the appendix of the thesis. Where examples taken from the literature were used, their origin is explicitly indicated.

In Table 4.1, for each of the starting rules the final value of the bound is given, together with the number of Phase 1 iterations and the total time taken to calculate the bound (CDC 7600 seconds). The bound corresponding to the best starting rule is indicated for each case, and its value compared with the best available solution for the problem.

The percentage deviations shown in Table 4.1 confirm that the dual bound is a very good lower bound for the p-median problem. For networks of up to 15 vertices the average percentage deviation from the best available solution was only 0.96%, whereas for networks ranging from 20 to 50 vertices the corresponding value was 3.53%.

It should be clear from Table 4.1 that the results are inconclusive as to which is the best starting rule for the σ_i's - for the great majority of the cases the bounds produced by the three rules are not very different from each other. It is important to note, however, that Rule 2 prevailed in 40% of the cases, whereas the corresponding percentages for Rules 1 and 3 were 31% and 29% respectively.

Starting Rule 3 appears to be the more unreliable of the three: for two of the networks of Table 4.1 (n = 10 and n = 25) this rule produced meaningless bounds for a number of values of p.

Table 4.1 - The Dual Bound: Computational Results

Problem Size		Starting Rule 1			Starting Rule 2			Starting Rule 3			Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p	Bound	No. iter	Time*	Bound	No. iter	Time*	Bound	No. iter	Time*	Rule***	Value		
10**	1	79	1	0.01	79	1	0.01	74	1	0.01	2	79	79	0.00
10	2	45	2	0.01	46	11	0.01	45	2	0.01	2	46	47	2.13
10	3	34	3	0.01	31	8	0.01	33	9	0.01	1	34	36	5.56
10	4	24	2	0.01	25	12	0.01	22	3	0.01	2	25	26	3.85
10	5	18	5	0.01	18	6	0.01	17	4	0.01	1	18	18	0.00
10	6	12	4	0.01	12	6	0.01	12	3	0.01	3	12	12	0.00
10	7	8	3	0.01	8	4	0.01	8	3	0.01	1	8	8	0.00
10	8	5	2	0.01	5	2	0.01	5	2	0.01	2	5	5	0.00
10	9	2	3	0.01	2	2	0.01	2	1	0.01	3	2	2	0.00
10	10	0	2	0.01	0	1	0.01	-1	1	0.01	2	0	0	0.00
10	1	400	1	0.01	400	1	0.01	395	1	0.01	1	400	400	0.00
10	2	250	13	0.01	236	28	0.02	258	14	0.01	3	258	273	5.49
10	3	195	24	0.02	192	20	0.02	185	24	0.02	1	195	195	0.00
10	4	148	4	0.01	140	33	0.02	144	19	0.02	1	148	149	0.67
10	5	105	6	0.01	107	8	0.01	107	6	0.01	3	107	107	0.00
10	6	69	6	0.01	69	3	0.01	75	9	0.02	3	75	75	0.00
10	7	42	6	0.01	42	3	0.01	43	1	0.01	3	43	43	0.00
10	8	15	5	0.01	15	1	0.01	15	5	0.01	2	15	15	0.00
10	9	2	4	0.01	2	2	0.01	N+	-	-	2	2	2	0.00
10	10	0	3	0.01	0	1	0.01	N	-	-	2	0	0	0.00

+ N = Large Negative Bound

* CPU time, in CDC 7600 seconds

** Example from Garfinkel et al. [41, p.231]

*** Where ties occurred in the value of the bound, no. of iterations and computing time, in this order, were used to determine the rule yielding the best bount

Table 4.1 (cont'ed) - The Dual Bound: Computational Results

Problem Size		Starting Rule 1			Starting Rule 2			Starting Rule 3			Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p	Bound	No. iter	Time*	Bound	No. iter	Time*	Bound	No. iter	Time*	Rule**	Value		
15	1	809	1	0.01	809	1	0.01	655	1	0.01	2	809	809	0.00
15	2	371	11	0.02	383	15	0.03	411	2	0.02	3	411	412	0.24
15	3	285	10	0.03	276	8	0.02	262	38	0.07	1	285	294	3.06
15	4	206	12	0.03	207	10	0.03	206	15	0.04	2	207	215	3.72
15	5	141	12	0.03	144	8	0.03	137	7	0.03	2	144	150	4.00
15	6	113	22	0.06	113	12	0.04	112	15	0.05	2	113	113	0.00
15	7	86	15	0.04	89	5	0.02	93	12	0.05	3	93	93	0.00
15	8	69	8	0.03	71	5	0.02	74	3	0.02	3	74	74	0.00
15	9	54	8	0.03	55	5	0.02	57	2	0.02	3	57	57	0.00
15	10	41	7	0.03	41	5	0.02	41	2	0.02	3	41	41	0.00
20	1	1159	1	0.03	1159	1	0.03	1136	1	0.03	1	1159	1159	0.00
20	2	711	4	0.04	704	16	0.05	720	3	0.03	3	720	724	0.56
20	3	488	7	0.04	480	5	0.04	462	20	0.08	1	488	518	5.79
20	4	382	21	0.09	383	10	0.05	406	28	0.11	3	406	414	1.93
20	5	302	28	0.10	316	19	0.08	305	2	0.04	2	316	338	6.51
20	6	255	32	0.14	258	27	0.11	203	1	0.03	2	258	259	0.39
20	7	202	15	0.08	217	17	0.10	218	45	0.25	3	218	227	3.96
20	8	185	17	0.09	191	13	0.08	177	1	0.04	2	191	199	4.02
20	9	161	20	0.10	168	10	0.07	158	10	0.09	2	168	175	4.00
20	10	142	20	0.10	148	10	0.07	150	12	0.12	3	150	151	0.66

* CPU time, in CDC 7600 seconds

** Where ties occurred in the value of the bound, no. of iterations and computing time, in this order, were used to determine the rule yielding the best bound

Table 4.1 (cont'ed) - The Dual Bound: Computational Results

Problem Size		Starting Rule 1			Starting Rule 2			Starting Rule 3			Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p	Bound	No. iter	Time*	Bound	No. iter	Time*	Bound	No. iter	Time*	Rule**	Value		
25	1	1352	1	0.06	1352	1	0.06	1329	1	0.06	1	1352	1352	0.00
25	2	856	9	0.07	935	30	0.12	790	1	0.05	2	935	956	2.20
25	3	708	7	0.07	700	10	0.08	251	1	0.05	1	708	722	1.94
25	4	551	9	0.09	543	21	0.14	N[+]	-	-	1	551	556	0.90
25	5	452	11	0.11	446	15	0.13	N	-	-	1	452	468	3.42
25	6	377	12	0.12	377	11	0.11	N	-	-	2	377	387	2.58
25	7	309	8	0.09	317	12	0.12	N	-	-	2	317	341	7.04
25	8	275	21	0.21	279	20	0.20	N	-	-	2	279	298	6.38
25	9	244	19	0.20	248	26	0.22	N	-	-	2	248	266	6.77
25	10	223	18	0.19	226	26	0.21	N	-	-	2	226	235	3.83
30	1	1432	1	0.10	1432	1	0.09	1361	1	0.09	2	1432	1432	0.00
30	2	924	8	0.12	909	13	0.12	926	8	0.12	3	926	936	1.07
30	3	718	3	0.11	721	10	0.13	737	50	0.33	3	737	777	5.15
30	4	610	26	0.25	578	14	0.16	566	9	0.15	1	610	610	0.00
30	5	500	14	0.18	496	16	0.18	486	33	0.35	1	500	516	3.10
30	6	424	20	0.24	409	19	0.20	408	21	0.28	1	424	438	3.20
30	7	361	26	0.29	355	16	0.21	348	22	0.33	1	361	386	6.48
30	8	320	27	0.33	313	18	0.28	320	28	0.44	1	320	337	5.04
30	9	281	17	0.27	278	24	0.36	280	4	0.15	1	281	294	4.42
30	10	250	14	0.23	248	14	0.29	250	8	0.22	3	250	265	5.66

[+] N = large negative bound

* CPU time, in CDC 7600 seconds

** Where ties occurred in the value of the bound, no. of iterations and computing time, in this order, were used to determine the rule yielding the best bound

Table 4.1 (cont'ed) - The Dual Bound: Computational Results

Problem Size		Starting Rule 1			Starting Rule 2			Starting Rule 3			Best Bound		Best avail.+ Solution	% Deviation from Best Avail. sol.
n	p	Bound	No. iter	Time*	Bound	No. iter	Time*	Bound	No. iter	Time*	Rule**	Value		
40	1	80634	1	0.22	80634	1	0.21	74955	1	0.21	2	80634	80634	0.00
40	2	43142	5	0.23	38874	12	0.26	43638	5	0.24	3	43638	45862	4.85
40	3	34728	99	1.03	33281	21	0.34	33423	56	0.65	1	34728	35946	3.39
40	4	26073	41	0.57	26374	32	0.47	23654	61	0.81	2	26374	26899	1.95
40	5	21443	40	0.70	21457	51	0.60	21210	75	1.20	2	21457	22396	4.19
40	6	18128	26	0.55	18471	91	0.81	17257	17	0.45	2	18471	18775	1.62
40	7	15145	35	0.64	15321	228	3.24	16402	101	1.99	3	16402	17426	5.88
40	8	14122	42	0.64	14349	75	1.23	14488	13	0.50	3	14488	16155	10.32
40	9	12897	190	3.99	13806	118	2.29	13212	47	1.29	2	13806	14539	5.04
40	10	12033	75	1.58	12371	74	1.44	12063	28	0.86	2	12371	13436	7.93
50	1	128548	1	0.44	128548	1	0.42	126560	1	0.42	2	128548	128548	0.00
50	2	70685	25	0.62	72128	182	1.87	71153	15	0.53	2	72128	72168	0.06
50	3	52615	108	1.71	50695	747	6.59	32931	569	8.38	1	52615	52708	0.18
50	4	39629	55	1.18	40313	241	2.97	34931	266	4.50	2	40313	42228	4.53
50	5	33335	104	2.03	32760	66	1.16	31325	489	10.47	1	33335	35677	6.56
50	6	29850	51	1.38	28708	80	1.68	29307	175	4.34	1	29850	31853	6.29
50	7	27240	58	1.57	26215	93	2.21	25392	60	2.07	1	27240	28300	3.75
50	8	24361	57	1.66	23440	382	6.89	20519	5	0.59	1	24361	25624	4.93
50	9	21125	30	1.29	20344	37	1.13	23403	331	11.17	3	23403	24129	3.01
50	10	19158	37	1.54	18895	76	2.29	21514	38	2.01	3	21514	22668	5.09

+ Best of two available heuristic solutions

* CPU time, in CDC 7600 seconds

** Where ties occurred in the value of the bound, no. of iterations and computing time, in this order, were used to determine the rule yielding the best bound

4.5 Comparison of Bounds

Three lower bounds for the p-median problem - the shortest distance bound, the graph-theoretical bound and the dual bound - are compared in Table 4.2. The examples for which these bounds are compared are the same as those used in Table 4.1. The bounds shown in Table 4.2 are lower bounds on the overall optimal solution to the corresponding p-median problem. The dual bound corresponds to the best starting rule of Table 4.1.

Table 4.2 is self-explanatory. The value of each of the three bounds is shown in its initial columns, and the best of the three bounds is singled out for comparison with the best available solution to the problem. This solution is the optimal solution for networks of up to 30 vertices, but for the 40- and 50-vertex networks the best available solution is a heuristic solution.

Under the 'Best Bound' heading, both the type of the best bound and its corresponding value are indicated. The dominance of the graph-theoretical bound over the shortest distance bound is confirmed by the numerical values shown in the table. On the other hand, there was not a single example for which the dual bound was dominated by the graph-theoretical bound.

Due to the very nature of the graph-theoretical bound, its performance improves as the value of p increases. The results shown in Table 4.2 suggest that the graph-theoretical bound becomes competitive with the dual bound for values of $p/n > 0.4$. As for the larger networks ($n \geqq 30$) the values of the ratio for which results are shown do not exceed 0.3, it is not surprising that for the examples presented the dual bound was the dominant bound.

The percentage deviation of the best bound from the best available solution is shown in the last column of the table. This

Table 4.2 - Comparison of Bounds

Problem Size		Shortest Distance Bound	Graph-theoretical Bound	Dual Bound	Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p				Type+	Value		
10++	1	34	56	79	D	79	79	0.00
10	2	27	36	46	D	46	47	2.13
10	3	21	29	34	D	34	36	5.56
10	4	17	22	25	D	25	26	3.85
10	5	13	16	18	D	18	18	0.00
10	6	10	12	12	D,GT	12	12	0.00
10	7	7	8	8	D,GT	8	8	0.00
10	8	4	5	5	D,GT	5	5	0.00
10	9	2	2	2	D,GT,S	2	2	0.00
10	10	0	0	0	D,GT,S	0	0	0.00
10	1	201	318	400	D	400	400	0.00
10	2	157	229	258	D	258	273	5.49
10	3	122	185	195	D	195	195	0.00
10	4	90	142	148	D	148	149	0.67
10	5	58	107	107	D,GT	107	107	0.00
10	6	30	75	75	D,GT	75	75	0.00
10	7	17	43	43	D,GT	43	43	0.00
10	8	4	15	15	D,GT	15	15	0.00
10	9	2	2	2	D,GT,S	2	2	0.00
10	10	0	0	0	D,GT,S	0	0	0.00

+ D = Dual bound

GT = Graph-theoretical bound

S = Shortest distance bound

++ Example from Garfinkel et al. [41, p.231]

Table 4.2 (cont'ed) - Comparison of Bounds

Problem size		Shortest Distance Bound	Graph-Theoretical Bound	Dual Bound	Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p				Type+	Value		
15	1	153	341	809	D	809	809	0.00
15	2	134	237	411	D	411	412	0.24
15	3	115	189	285	D	285	294	3.06
15	4	98	163	207	D	207	215	3.72
15	5	82	138	144	D	144	150	4.00
15	6	70	113	113	D,GT	113	113	0.00
15	7	58	93	93	D,GT	93	93	0.00
15	8	48	74	74	D,GT	74	74	0.00
15	9	38	57	57	D,GT	57	57	0.00
15	10	28	41	41	D,GT	41	41	0.00
20	1	295	596	1159	D	1159	1159	0.00
20	2	271	459	720	D	720	724	0.56
20	3	250	349	488	D	488	518	5.79
20	4	229	278	406	D	406	414	1.93
20	5	208	250	316	D	316	338	6.51
20	6	187	226	258	D	258	259	0.39
20	7	167	205	218	D	218	227	3.96
20	8	147	184	191	D	191	199	4.02
20	9	129	163	168	D	168	175	4.00
20	10	113	142	150	D	150	151	0.66

+ D = Dual Bound
GT = Graph-theoretical bound

Table 4.2 (cont'ed) - Comparison of Bounds

Problem size		Shortest Distance Bound	Graph-Theoretical Bound	Dual Bound	Best Bound		Optimal Solution	% Deviation from Optimal Solution
n	p				Type+	Value		
25	1	349	717	1352	D	1352	1352	0.00
25	2	325	577	935	D	935	956	2.20
25	3	303	467	708	D	708	722	1.94
25	4	282	378	551	D	551	556	0.90
25	5	261	325	452	D	452	468	3.42
25	6	241	301	377	D	377	387	2.58
25	7	223	279	317	D	317	341	7.04
25	8	206	257	279	D	279	298	6.38
25	9	190	236	248	D	248	266	6.77
25	10	174	215	226	D	226	235	3.83
30	1	337	720	1432	D	1432	1432	0.00
30	2	315	612	926	D	926	936	1.07
30	3	298	518	737	D	737	777	5.15
30	4	281	445	610	D	610	610	0.00
30	5	264	380	500	D	500	516	3.10
30	6	248	323	424	D	424	438	3.20
30	7	234	282	361	D	361	386	6.48
30	8	221	265	320	D	320	337	5.04
30	9	208	248	281	D	281	294	4.42
30	10	195	231	250	D	250	265	5.66

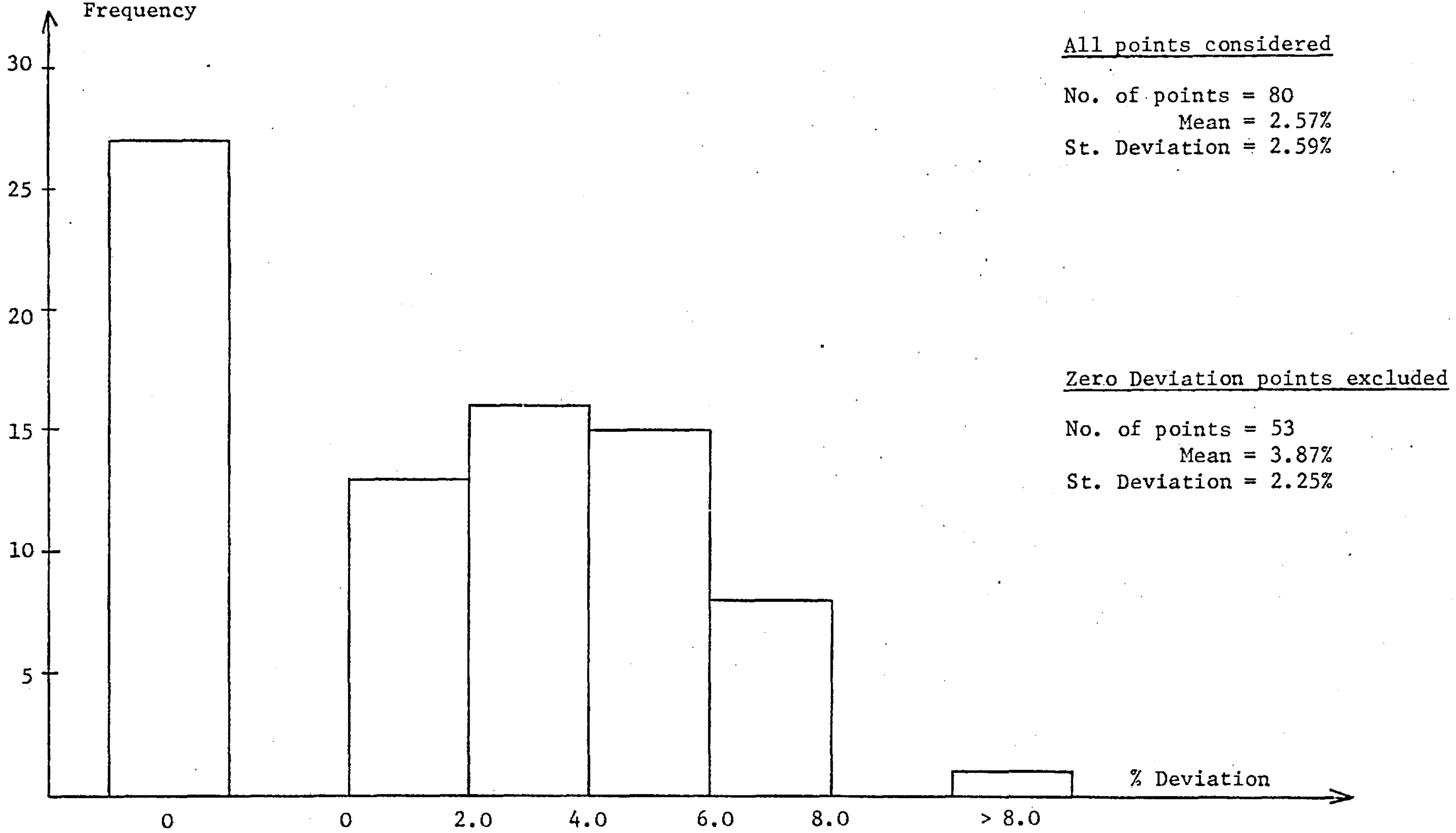
+ D = Dual bound

Table 4.2 (cont'ed) - Comparison of Bounds

Problem size		Shortest Distance Bound	Graph-Theoretical Bound	Dual Bound	Best Bound		Best avail.+ Solution	% Deviation from best avail. sol.
n	p				Type++	Value		
40	1	14029	30094	80634	D	80634	80634	0.00
40	2	13378	26636	43638	D	43638	45862	4.85
40	3	12767	23951	34728	D	34728	35946	3.39
40	4	12175	21450	26374	D	26374	26899	1.95
40	5	11608	19059	21457	D	21457	22396	4.19
40	6	11056	16809	18471	D	18471	18775	1.62
40	7	10517	14872	16402	D	16402	17426	5.88
40	8	9980	13157	14488	D	14488	16155	10.32
40	9	9449	11599	13806	D	13806	14539	5.04
40	10	8965	10878	12371	D	12371	13436	7.93
50	1	17687	41104	128548	D	128548	128548	0.00
50	2	16828	37393	72128	D	72128	72168	0.06
50	3	16028	33894	52615	D	52615	52708	0.18
50	4	15228	30614	40313	D	40313	42228	4.53
50	5	14456	27644	33335	D	33335	35677	6.56
50	6	13754	24820	29850	D	29850	31853	6.29
50	7	13060	22362	27240	D	27240	28300	3.75
50	8	12378	20059	24361	D	24361	25624	4.93
50	9	11749	18066	23403	D	23403	24129	3.01
50	10	11134	16406	21514	D	21514	22668	5.09

+ Best of two available heuristic solutions

++ D = Dual bound



All points considered

No. of points = 80
Mean = 2.57%
St. Deviation = 2.59%

Zero Deviation points excluded

No. of points = 53
Mean = 3.87%
St. Deviation = 2.25%

Figure 4.2

% Deviation of Best Bound from Best Available Solution column confirms that the dual bound - the dominant bound for all examples in the table - is a very good lower bound for the p-median problem. Although the percentage deviation can only increase when the best available solution is of heuristic nature, the maximum observed deviation was 10.32%.

A histogram of the percentage deviations defined above is shown in Figure 4.2. In this figure the mean and the standard deviation are shown:

(i) For all points of Table 4.2;

(ii) Only for the points corresponding to non-zero deviations, since the zero deviations mainly correspond to the smaller networks.

4.6 Conclusions

The quality of bounds used in tree search methods is a factor of vital importance in the efficiency of the method. Branch-and-bound algorithms so far developed for the p-median problem suffer from a lack of strong lower bounds, and for this reason are not very efficient.

Two new lower bounds for the p-median problem were developed in the present chapter, namely the graph-theoretical bound and the dual bound. The graph-theoretical bound is based on shortest spanning trees and arborescences and other graphical properties of the problem. The dual bound is based on the dual of the linear programming relaxation of the p-median problem. A heuristic procedure was developed to compute an exact value for this bound.

The graph-theoretical bound was shown to dominate the shortest distance bound, a bound developed in [12] to be used in a branch-and-bound algorithm. The performance of the graph-theoretical bound is poor for small values of p, but improves considerably as the value of p increases.

The dual bound has proved to be a very good lower bound for the p-median problem. For 80 test problems its average deviation from the best available solution was only 2.57%. The dual bound can be easily embedded into branch-and-bound algorithms, as shown in the next chapter.

CHAPTER FIVE

A BRANCH-AND-BOUND ALGORITHM

5.1 Introduction

Branch-and-bound algorithms, additive algorithms and direct search algorithms are some of the variations around the same basic idea, having common features which offer both advantages and disadvantages in relation to other solution procedures. On the positive side tree search methods are easy to understand and to program for the computer. They lack, however, mathematical structure, and the upper bound on the number of steps needed to complete the algorithm is of the order of $O(K^m)$, where $\underline{K}$ is a constant and $\underline{m}$ is a function of the problem variables.

It is not felt that this thesis is the appropriate place to discuss in any depth the principles, types and properties of tree search methods. A very good introduction to the subject can be found in [40], and the subject is dealt with in great detail in the literature [2, 3, 7, 72, 75]. The basic principle upon which these methods are based is outlined in very short form in the next paragraph.

The basic principle involved in tree search methods is the partition of an initial problem P_0 into a number of subproblems, $P_1, P_2, \ldots, P_k$, whose totality represent P_0 and which are easier to solve than P_0. If however after the initial partition it is still impossible to resolve* a subproblem P_i, this subproblem is further partitioned into yet smaller subproblems $P_{i1}, P_{i2}, \ldots, P_{ik}$. This partitioning (also

* To resolve a subproblem means:

either (i) find an optimal solution,
or (ii) show that the value of the optimal solution of the subproblem is worse than the best solution obtained so far,
or (iii) show that the subproblem is infeasible.

called branching) is repeated for every subproblem which cannot be resolved.

A direct tree search algorithm for the p-median problem is described in the present chapter. This algorithm is a depth-first tree search approach to the problem, the basic principles of which are outlined in [12]. These basic principles had to undergo considerable development before they could be applied to solve the p-median problem for networks of meaningful size. In its final form the algorithm cascades through the shortest distance bound and the dual bound described in Chapter 4.

The branch-and-bound algorithm is fully described in Section 5.2: its basic principles are outlined, the embedding of each of the bounds is described in detail, and the detailed steps of the procedure are given. A small example is solved to illustrate the procedure and the importance of tight bounds in determining the efficiency of the algorithm.

Computational results are then presented. The algorithm is shown to guarantee an optimal solution for 30-vertex networks in less than 2 minutes in a CDC 7600 computer, for any value of p, $1 \leq p \leq n$, where n is the number of vertices of the network. Except for small values of p, computing times become prohibitive for $n > 30$.

The computational experience reported above represents a substantial advancement in the field of exact solution procedures for the p-median problem. While other methods available in the literature may occasionally provide optimal solutions for problems in which n is larger than 30, no previously available method guarantees an optimal solution for any value of p for networks with more than 20 vertices. The branch-and-bound algorithm described in the present chapter also proved to be computationally faster and more efficient than previous tree search approaches to solve the problem [30, 55], especially for the larger networks ($n > 20$).

One of the methods that may on occasion provide an optimal solution for large networks is the decomposition formulation of Garfinkel et al. [41], discussed in Chapter 3. The results obtained when this formulation was embedded into the branch-and-bound algorithm are reported in Section 5.4. Unfortunately the embedding did not improve the convergence of the decomposition formulation, and computing times become prohibitive even for 20-vertex networks.

5.2 The Branch-and-bound algorithm

A direct tree search algorithm for the p-median problem is now described. This algorithm is a depth-first tree search approach, in which each subproblem generated by the branching from a tree node is produced by setting - for a given vertex x_j - a variable ξ_{ij} to 1 for some vertex x_i. The setting of $\xi_{ij} = 1$ implies that vertex x_j is allocated to vertex x_i, which, obviously, also implies that x_i is a median vertex.

The vertices are randomly numbered from x_1 to x_n. The search proceeds by allocating sequentially - starting from x_1 and finishing with x_n - all the vertices x_j of the network. A vertex x_j is initially allocated to itself (thus becoming a median), then to its nearest, second nearest, third nearest vertices, and so on, until all possibilities are implicitly enumerated.

Two lower bounds are used to limit the search. The first bound to be activated is the shortest distance bound, as this is the faster of the two bounds. If this bound fails to cause backtracting, the dual bound is then activated. The cascading through the two bounds combines the best feature of each of them - fastness in the case of the shortest distance bound and tightness in the case of the dual bound, and has proved to be efficient. Observations relevant to the type of search being carried out, and fully described in 5.2.1, limit the size of the tree search further by reducing the number of alternative possible allocations of a vertex x_j at any one stage.

In the introduction of this thesis it has been pointed out that the p-median problem is a combinatorial problem characterized by a large number of feasible solutions. It is therefore not surprising that problems are often found for which multiple optimal solutions exist. The algorithm has been programmed so that multiple optimal solutions may be generated (if desired) for any given problem. The search for possible multiple optimal solutions is costly, however. The effect that seeking multiple optimal solutions has in the size of the search is reported in Section 5.3.

Finally, it should be pointed out that the search can be considerably reduced if an upper bound on the value of the optimal solution is available prior to the start of the tree search. This upper bound can be calculated by a simple heuristic such as the 1-optimal substitution method of Teitz and Bart. The advantage that the availability of an upper bound has over starting from $z^* = \infty$ is also demonstrated in 5.3.

5.2.1 Description of the algorithm

An overview of the tree search algorithm is given in the present section. A more detailed description of the procedure is left for the remaining subsections of 5.2.

The tree search can be carried out as follows: set up a matrix $M = [m_{kj}]$, the j^{th} column of which contains all the vertices of the network N arranged in ascending order of their shortest distance from vertex x_j. Matrix M can be set up only after the distance matrix of the graph, $D = [d_{ij}]$, has been calculated. Assume now that matrix M is available. Then, if $m_{kj} = x_i$, vertex x_i is the k^{th} nearest vertex to x_j. Obviously the nearest vertex to x_j is x_j itself, i.e. $m_{1j} = x_j$.

The search proceeds by allocating sequentially - starting from x_1 and finishing with x_n - all the vertices x_j of the graph. A vertex x_j is initially allocated to vertex m_{1j}, then m_{2j}, m_{3j} and so on, until all possibilities are implicitly enumerated. The following observations can now be made.

1. Since in the optimal solution there are p median vertices, each allocation of a vertex in this solution must be the best of the p possible allocations to the median vertices, i.e. there are at least (p-1) more costly ways of allocating any one vertex. The last (p-1) rows of matrix M can therefore be permanently removed without any possibility of the optimal solution to the p-median problem being affected.

2. Suppose that vertex $x_{j'}$ has been allocated to vertex $m_{k'j'}$ $(= x_i)$. For a vertex x_j not yet allocated, corresponding to column j of matrix M $(j' < j \leqq n)$, let x_i be the k^{th} nearest vertex to x_j, i.e. let $m_{kj} = x_i$. Then all entries $m_{\ell j}$ of this column, $\ell > k$, can be neglected (marked), since the allocation of $x_{j'}$ to x_i implied that x_i is a median vertex. Vertex x_j can therefore be allocated at lower cost to x_i than to any of the vertices $m_{\ell j}$, $\ell > k$. Clearly, if at some backtracking step during the search the allocation of vertex $x_{j'}$ to x_i is altered, then the entries $m_{\ell j}$ have to be reconsidered (unmarked).

3. Let vertex $x_{j'}$ be allocated to vertex $m_{k'j'}$. Then all vertices $m_{1j'}, m_{2j'}, \ldots, m_{(k'-1)j'}$ are not median vertices, for, if they were, $x_{j'}$ could have been allocated to any of them instead at a lower cost. These vertices can therefore be marked in all columns $j > j'$. Once more the marking of these vertices is temporary and must be removed whenever the allocation of $x_{j'}$ to $m_{k'j'}$ is changed.

4. If the top t entries of a column j of matrix M corresponding to an unallocated vertex x_j are marked and the following entry - t+1 -

is a median vertex (i.e. if $m_{(t+1)j}$ is a median vertex), then x_j must be allocated to median $m_{(t+1)j}$ and no other alternative need be considered until some of the top t entries are unmarked. This is a direct consequence of observations 2 and 3 above.

5. If at some stage q of the tree search a total of p median vertices are implied by the allocations already made, then the remaining unallocated vertices must be allocated to their nearest median vertex. This is obviously the optimal completion of the partial solution corresponding to the allocations up to that stage, and the next backtracking step must necessarily involve a change in the allocation at stage q.

Observations 1 to 5 above can be used to limit the size of the tree search by reducing the number of alternative possible allocations of a vertex x_j at any one stage. They are not, however, the decisive element in the tree search. Lower bounds on the overall optimal solution to the problem, calculated given the allocations already made at some stage q, are of primary importance in determining the efficiency of the search and the size of problems it can solve.

Details are given in the next two sections of the embedding into the branch-and-bound algorithm of two of the bounds described in Chapter 4 - the shortest distance bound and the dual bound. In its final form the algorithm cascades through these two bounds, and this has proved decisive in securing optimal solutions for networks of up to 30 vertices, in a reasonable amount of computing time and for every possible value of p ($1 \leqq p \leqq n$).

5.2.2 The embedding of the shortest distance bound

The embedding of the shortest distance bound into the branch-and-bound algorithm is now described.

Using the notation of 5.2.1, suppose that the allocations of vertices made so far (up to and including the allocation of vertex $x_{j'}$) imply a total of p', p'<p, median vertices. The remaining allocations must then imply a total of a further (p-p') median vertices. Let J be the set of indices of the as yet unallocated vertices. In general J is the set of indices j, where $j' < j \leqq n$, but excluding the indices of those vertices whose allocation might have already been forced by the allocation of the first j' vertices $x_1, x_2, \ldots, x_{j'}$ - see Observation 4 of Section 5.2.1.

Let $m_{\alpha_j j}$ and $m_{\beta_j j}$ be the topmost and second topmost unmarked entries in column j. The best possible allocation of vertex x_j is then to vertex $m_{\alpha_j j}$. If the number of distinct vertices m_{α_j} for $j \in J$ is h and $h = p-p'$, then all these best possible allocations for the as yet unallocated vertices are feasible (i.e. they produce a total of p median vertices). These allocations then constitute the optimal completion of the partial solution implied by the current allocations of the vertices $x_1, x_2, \ldots, x_{j'}$. In such event the result of the optimal completion should be noted and backtracking can take place from the current partial solution.

If, however, $h > (p-p')$, then at least (h-p+p') of the best allocations must be changed to second best or worse in order to produce a total of p median vertices. Now let $J' \subseteq J$ be the union of subsets $J'_1 \cup J'_2 \cup \ldots \cup J'_r$. Each of the subsets J'_k, $1 \leqq k \leqq r$, comprise the columns $j \in J$ of matrix M which have the same m_{α_j} $(= x_k)$ vertex as their topmost unmarked entry. Thus, when there are columns $j \in J$ of M which have as their topmost unmarked entry the same vertex x_k, for each such vertex x_k a subset J'_k comprising the corresponding columns $j \in J$ is defined. For each subset J'_k the sum

$$S_j = \sum_{j \in J'_k} v_j \, [\, d(x_j, m_{\beta_j j}) - d(x_j, m_{\alpha_j j})] \tag{5.1}$$

can be defined.

The idea behind Equation (5.1) is that (i) when there is more than one column $j \in J$ which has the same vertex x_k as its topmost unmarked entry, and (ii) vertex x_k is not to be a median vertex in the completion of the partial solution for which p' median vertices have already been defined, then, for all such columns, the allocation of the corresponding vertices x_j will have to be made to the second topmost (or worse) unmarked entry.

Now, in the event that $J' \subset J$ (i.e. if some columns $j \in J$ have as their topmost entry a vertex x_ℓ that does not appear as the topmost entry in any other column $j \in J$), let $J'' = J - J'$. For each column $j \in J''$ the sum S_j of Equation (5.1) becomes

$$S_j = v_j \, [d(x_j, m_{\beta_j j}) - d(x_j, m_{\alpha_j j})] \tag{5.2}$$

Then, for allocations that must be changed to second best or worse, the minimum additional cost of allocation is the sum of the (h-p+p') smallest S_j, $j \in J = J' + J''$.

A lower bound on the cost of the overall optimal solution, given the current partial solution, is the sum of the costs of the allocations already made, plus the sum

$$\sum_{j \in J} v_j \, d(x_j, m_{\alpha_j j}) \ , \tag{5.3}$$

plus the sum of the (h-p+p') smallest S_j, the S_j being given either by Equation (5.1) (for j ∈ J'), or by Equation (5.2) (for j ∈ J").

It is also possible that h < (p-p'), in which case the best completion of the current partial solution leads to less than p median vertices. However, since it is quite apparent that the transmission $\sigma(\overline{X}_p)$ of the optimal p-median $\overline{X}_p$ monotonically decreases as p increases, it follows that when h < (p-p') the current partial solution is certainly not part of the optimal p-median solution and backtracking can then take place.

5.2.3 The embedding of the dual bound

When embedding the dual bound, it is important to determine what effect the setting of the ξ_{ij} variables along the tree search has on the corresponding dual formulation of the relaxed problem. Two separate cases will be considered: the setting of variables that imply the assignment of a median vertex (the ξ_{ii} variables), and the setting of variables that imply the allocation of vertices to medians (the ξ_{ij} variables, i≠j). Then these two cases will be combined, and the generalized procedure for the embedding of the dual bound described.

It is worthwhile to recall both the primal and the dual formulations of the relaxed problem. The primal LP is

$$\text{Minimize} \quad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\, \xi_{ij} \tag{5.4}$$

Subject to

$$\sum_{i=1}^{n} \xi_{ij} = 1 \quad \forall \ j \tag{5.5}$$

$$\sum_{i=1}^{n} \xi_{ii} = p \tag{5.6}$$

$$\xi_{ij} - \xi_{ii} \leqq 0 \quad \forall \quad i,j, \ i \neq j \tag{5.7}$$

$$\xi_{ij} \geqq 0 \quad \forall \quad i,j \tag{5.8}$$

The dual of this linear programme has already been developed in Chapter 4. The DLP is

$$\text{Maximize} \quad Z' = \sum_{i=1}^{n} \sigma_i + p\,\sigma_{n+1} \tag{5.9}$$

Subject to

$$\sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j\neq i}}^{n} \pi_{ij} \leqq 0 \quad \forall \quad i \tag{5.10}$$

$$\sigma_j + \pi_{ij} \leqq d_{ij} \quad \forall \quad i,j,\ j\neq i \tag{5.11}$$

$$\pi_{ij} \leqq 0 \quad \forall \quad i,j,\ j\neq i \tag{5.12}$$

$$\sigma_i \geqq 0, \qquad i = 1,\ldots,n \tag{5.13a}$$

$$\sigma_{n+1} \leqq 0 \tag{5.13b}$$

The setting of the ξ_{ii} variables

The setting of a ξ_{ii} variable to 1 corresponds to assigning vertex x_i to be a median vertex. Suppose that vertex x_{i_1} is assigned as a median. It follows that $\xi_{i_1 i_1} = 1$, and this in turn implies the following:

(a) The primal constraint corresponding to $j = i_1$ in Equation (5.5) is

$$\sum_{i=1}^{n} \xi_{ii_1} = 1 \ .$$

Since $\xi_{i_1 i_1} = 1$, it follows that $\xi_{ii_1} = 0 \ \forall \ i\neq i_1$.

(b) The number of medians yet to be assigned is reduced by 1 to (p-1). Equation (5.6) becomes

$$\sum_{\substack{i=1\\ i\neq i_1}}^{n} \xi_{ii} = p-1$$

(c.1) The constraints corresponding to $i = i_1$ in Equation (5.7) become

$$\xi_{i_1 j} - \xi_{i_1 i_1} \leqq 0 \quad \forall \quad j \neq i_1 ,$$

or, since $\xi_{i_1 i_1} = 1$,

$$\xi_{i_1 j} \leqq 1 \quad \forall \quad j \neq i_1 .$$

The above is already implied by Equation (5.5). These constraints can therefore be dropped from the primal LP.

(c.2) The constraints corresponding to $j = i_1$, $i \neq i_1$ in Equation (5.7) become

$$\xi_{ii_1} - \xi_{ii} \leqq 0 \qquad \forall \quad i \neq i_1 .$$

Since $\xi_{ii_1} = 0 \; \forall \; i \neq i_1$ (see (a) above), it follows that the above reduces to

$$\xi_{ii} \geqq 0 \quad \forall \quad i \neq i_1,$$

which is already covered by Equation (5.8).

The observations (a) to (c.2) above allow the reduced primal (after $\xi_{i_1 i_1}$ has been set to 1) to be written as

Minimize $$Z = \sum_{i=1}^{n} \sum_{\substack{j=1\\ j\neq i_1}}^{n} d_{ij}\, \xi_{ij} \tag{5.14}$$

Subject to

$$\sum_{i=1}^{n} \xi_{ij} = 1 \qquad \forall \quad j \neq i_1 \tag{5.15}$$

$$\sum_{\substack{i=1 \\ i \neq i_1}}^{n} \xi_{ii} = p-1 \tag{5.16}$$

$$\xi_{ij} - \xi_{ii} \leqq 0 \ \forall \ i \neq i_1, \ j \neq i_1, \ i \neq j \tag{5.17}$$

$$\xi_{ij} \geqq 0 \ \forall \ i, \ j \neq i_1 \tag{5.18}$$

The dual corresponding to the above reduced primal is then

$$\text{Maximize} \quad Z' = \sum_{\substack{i=1 \\ i \neq i_1}}^{n} \sigma_i + (p-1)\,\sigma_{n+1} \tag{5.19}$$

Subject to

$$\sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j \neq i_1 \\ j \neq i}}^{n} \pi_{ij} \leqq 0 \ \forall \ i \neq i_1 \tag{5.20}$$

$$\sigma_j + \pi_{ij} \leqq d_{ij} \ \forall \ i \neq i_1, \ j \neq i_1, \ j \neq i \tag{5.21}$$

$$\sigma_j \leqq d_{i_1 j} \ \forall \ j \neq i_1 \tag{5.22}$$

$$\sigma_i \geqq 0 \ \forall \ i \neq i_1 \tag{5.23a}$$

$$\sigma_{n+1} \leqq 0 \tag{5.23b}$$

$$\pi_{ij} \leqq 0 \quad \forall \quad i \neq i_1 \, , \ j \neq j_1 \, , \ i \neq j \tag{5.23c}$$

Note that in the above formulation $\sigma_{i_1} = 0$ and $\pi_{ii_1} = 0 \ \forall \ i$.

The Setting of the ξ_{ij} Variables ($i \neq j$)

The setting of a ξ_{ij} variable to 1 ($i \neq j$), corresponds to allocating a nonmedian vertex x_j to a median vertex x_i. Evidently ξ_{ij} can be set to 1 only if $\xi_{ii} = 1$, a condition expressed by Equation (5.7) of the primal LP.

Suppose now that vertex x_{j_1} has been allocated to median vertex x_{i_1}. The setting of $\xi_{i_1 j_1}$ to 1 has very similar implications to the setting of $\xi_{i_1 i_1}$ to 1, explained above in detail. The only differences are that (i) the right-hand side of Equation (5.6) is not affected this time (since no new median has been assigned by setting $\xi_{i_1 j_1}$ to 1), and (ii) no new upper bounds are imposed on the remaining positive dual variables (a consequence of the fact that no vertex can be allocated to a nonmedian vertex).

When $\xi_{i_1 j_1}$ is set to 1 (following the setting of $\xi_{i_1 i_1}$ to 1), the remaining dual LP is

$$\text{Maximize} \quad Z' = \sum_{\substack{i=1 \\ i \neq i_1, j_1}}^{n} \sigma_i + (p-1)\,\sigma_{n+1} \tag{5.24}$$

Subject to

$$\sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j \neq i_1, j_1 \\ j \neq i}}^{n} \pi_{ij} \leqq 0 \ \forall\ i \neq i_1, j_1 \tag{5.25}$$

$$\sigma_j + \pi_{ij} \leqq d_{ij} \ \forall\ i \neq i_1, j_1,\ j \neq i_1, j_1,\ j \neq i \tag{5.26}$$

$$\sigma_j \leqq d_{i_1 j} \ \forall\ j \neq i_1, j_1 \tag{5.27}$$

$$\sigma_i \geqq 0 \ \forall\ i \neq i_1, j_1 \tag{5.28a}$$

$$\sigma_{n+1} \leqq 0 \tag{5.28b}$$

$$\pi_{ij} \leqq 0 \quad \forall \quad i \neq i_1, j_1,\ j \neq i_1, j_1,\ i \neq j \tag{5.28c}$$

The Embedding Generalized

In the tree search described in the present chapter, at a given stage $\underline{q}$ of the search several ξ_{ii} and ξ_{ij} $(i \neq j)$ variables will have been set to 1. The dual formulation is now given for the general case of the embedding.

Suppose that at stage $\underline{q}$ of the tree search $\underline{r}$ variables ξ_{ii}, $r < p$, and $\underline{s}$ variables ξ_{ij} $(i \neq j)$, $s < (n-p)$, have been set to 1. The dual formulation for this general case is a straightforward generalization of equations (5.24) to (5.28c) above. The only additional dual constraints stem from the fact that each variable ξ_{ii} set to 1 establishes new upper bounds on the values of the remaining positive dual variables.

Define the sets $i_q = \{i_1, i_2, \ldots, i_r\}$, and $j_q = \{j_1, j_2, \ldots, j_s\}$, where i_q is the set of indices of the ξ_{ii} variables set to 1 at stage $\underline{q}$ of the search, and j_q is the set of indices of the ξ_{ij} $(i \neq j)$ variables set to 1 at the same stage $\underline{q}$ of the tree search. The generalized embedded dual formulation (at stage $\underline{q}$ of the tree search) can be written as

$$\text{Maximize} \quad Z' = \sum_{\substack{i=1 \\ i \notin i_q, j_q}}^{n} \sigma_i + (p-r)\, \sigma_{n+1} \tag{5.29}$$

Subject to

$$\sigma_i + \sigma_{n+1} - \sum_{\substack{j=1 \\ j \notin i_q, j_q \\ j \neq i}}^{n} \pi_{ij} \leqq 0 \;\forall\; i \notin i_q, j_q \tag{5.30}$$

$$\sigma_j + \pi_{ij} \leqq d_{ij} \;\forall\; i \notin i_q, j_q,\; j \notin i_q, j_q,\; j \neq i \tag{5.31}$$

$$\sigma_j \leqq \operatorname*{Min}_{i \in i_q} d_{ij} \;\forall\; j \notin i_q, j_q \tag{5.32}$$

$$\sigma_i \geqq 0 \;\forall\; i \notin i_q, j_q \tag{5.33a}$$

$$\sigma_{n+1} \leqq 0 \tag{5.33b}$$

$$\pi_{ij} \leqq 0 \;\forall\; i \notin i_q, j_q,\; j \notin i_q, j_q,\; i \neq j \tag{5.33c}$$

Fortunately, a simple algorithm based on the procedure of 4.4.2 can find approximate solutions to this seemingly complicated formulation.

The heuristic procedure described in Section 4.4.2 must be applied to a reduced distance matrix of the network for which the p-median problem must be solved. This reduced distance matrix is always obtained from the original (n×n) distance matrix of the network, by crossing out the rows and columns corresponding to each vertex that has been either assigned as a median or allocated to an existing median when the embedded dual bound is activated.

The only variation in relation to the heuristic procedure of 4.4.2 lies in the fact that the initial positive σ's cannot be freely chosen as in Chapter 4. Each of the positive σ's to be determined at stage q of the tree search has an upper bound defined by Equation (5.32). The initial positive σ's for the computation of the dual bound at stage q of the search can be chosen to be the upper bounds of Equation (5.32), in which case they are given by

$$\sigma_j = \underset{i \in i_q}{\text{Min}} \ d_{ij} \ \forall \ j \notin i_q, j_q \qquad (5.34)$$

The heuristic procedure of Section 4.4.2 applies exactly as described there, after the distance matrix of the network is appropriately reduced to take into account the assignment of median vertices and the allocation of nonmedian vertices up to stage q of the tree search. The initial values of the positive σ's are given by Equation (5.34).

An illustrative example might help to clarify the general case of the embedding. Suppose that the optimal 3-median is being sought for a 5-vertex network, and that at stage q of the search vertices x_1 and x_3 have been assigned as medians, and nonmedian vertex x_2 allocated to median x_3. This implies $\xi_{11} = \xi_{33} = 1$, and $\xi_{32} = 1$.

The distance matrix D of this 5-vertex network is shown in Figure 5.1, together with the σ_p vector of positive dual variables. The i_q and j_q sets are respectively $i_q = \{1, 3\}$, and $j_q = \{2\}$.

If the rows and columns corresponding to i_q and j_q are crossed out, as indicated in Figure 5.1, it can be easily seen that the dual LP at this stage can be written as

$$\text{Maximize} \quad Z' = \sigma_4 + \sigma_5 + (3-2)\sigma_6$$

Subject to

$$\sigma_4 + \sigma_6 - \pi_{45} \leqq 0$$

$$\sigma_5 + \sigma_6 - \pi_{54} \leqq 0$$

$$D = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} \\ d_{41} & d_{42} & d_{43} & d_{44} & d_{45} \\ d_{51} & d_{52} & d_{53} & d_{54} & d_{55} \end{bmatrix}$$

$$\sigma_p = (\sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \sigma_4 \quad \sigma_5)$$

Figure 5.1

D Matrix and Corresponding σ_p Vector for a 5-Vertex Network

$$\sigma_4 + \pi_{54} \leqq d_{54}$$

$$\sigma_5 + \pi_{45} \leqq d_{45}$$

$$\left.\begin{array}{l}\sigma_4 \leqq d_{14}\\ \sigma_4 \leqq d_{34}\end{array}\right\} \Rightarrow \sigma_4 \leqq \text{Min}\ (d_{14},\ d_{34})$$

$$\left.\begin{array}{l}\sigma_5 \leqq d_{15}\\ \sigma_5 \leqq d_{35}\end{array}\right\} \Rightarrow \sigma_5 \leqq \text{Min}\ (d_{15},\ d_{35})$$

$$\sigma_4,\ \sigma_5 \geqq 0$$

$$\sigma_6 \leqq 0\ ,\quad \pi_{45} \leqq 0\ ,\quad \pi_{54} \leqq 0\ ,$$

which coincides with the general formulation given by Equations (5.29) through (5.33c).

Note that in Figure 5.1 the d_{ij} elements crossed out by a vertical line, or by both a vertical and a horizontal line, do not appear in the formulation above. On the other hand, the d_{ij} elements crossed out only by horizontal lines corresponding to median vertices, provide upper bounds on the remaining σ_i variables. Finally, the d_{ij} elements not crossed out appear in the constraints corresponding to Equations (5.30) and (5.31) [The right-hand side of Equation (5.30) is in reality d_{ii}, always equal to zero in the p-median problem] .

5.2.4 The Detailed Steps of the Algorithm

The detailed steps of the branch-and-bound algorithm described in 5.2.1 are now given below.

Step 1. Calculate D, the distance matrix of the network. Floyd's algorithm [34, 83] can be used to calculate D.

Step 2. Set up the matrix M = $[m_{kj}]$, the j^{th} column of which contains all the vertices of the network arranged in ascending order of their shortest distance from vertex x_j.

Step 3. Cross out the last (p-1) rows of matrix M (see Observation 1 of 5.2.1).

Step 4. Note the best available solution before the beginning of the tree search as Z* (such solution can be provided by a simple heuristic). If no solution is available make $Z^* = \infty$.

Step 5. Make $j' = 1$.

Step 6. Allocate vertex $x_{j'}$ to the topmost unmarked entry - $m_{k'j'} = x_i$ - of column j'. x_i becomes then a median vertex.

Step 7. For all columns j, $j' < j \leq n$, mark all unmarked entries which appear after x_i as $MT1_i$.

Step 8. If $x_i = x_{j'}$, go to Step 9. Otherwise assume there are $(k'-1)$ vertices preceding vertex x_i in column j'. These vertices correspond to entries $m_{1j'}, m_{2j'}, \ldots, m_{(k'-1)j'}$. For all columns j, $j' < j \leq n$, mark all unmarked entries corresponding to vertices $m_{1j'}, m_{2j'}, \ldots, m_{(k'-1)j'}$ as $MT2_{j'}$.

Step 9. For all columns j, $j' < j \leq n$, make, whenever the case, the forced allocations of nonmedian vertices to the medians assigned so far (see Observation 4 of 5.2.1).

Step 10. Calculate the cost of the allocations made so far. Call this cost C_a. Then if $C_a \geq Z^*$ go to Step 15 (backtrack). Otherwise (if $C_a < Z^*$) go to Step 11 below.*

Step 11. Test whether p', the number of medians assigned so far, is equal to p. If so, or if $h = (p-p')$ [see Section 5.2.2], go to Step 12 below. Otherwise go to Step 13.

* The detailed steps given above are for The case when only one optimal solution is desired. If multiple optimal solutions are desired, then in Step 10, and in Steps 13 and 14, bactracking can take place only if Z* is strictly less than: (i) C_a in Step 10, and (ii) LB and DB in Steps 13 and 14, respectively.

Step 12. Allocate each of the remaining non-allocated vertices to their closest median. These allocations then constitute the optimal completion of the partial solution implied by the current allocations of the vertices x_1, x_2, ..., $x_{j'}$. Calculate C_T, the cost corresponding to the solution just completed. Then:

(a) If $C_T < Z^*$ note the solution, make $Z^* = C_T$ and go to Step 15 (backtrack);

(b) If $C_T \geqq Z^*$, go to Step 15 (simply backtrack).

Step 13. Calculate the shortest distance bound, given the allocations made so far. Call this bound LB. Then:

(a) If LB $\geqq Z^*$, or if $h < (p-p')$ (see Section 5.2.2), go to Step 15 (backtrack);

(b) Otherwise go to Step 14 below.

Step 14. Calculate the dual bound, given the allocations made so far. Call this bound DB. Then:

(a) If DB $\geqq Z^*$, go to Step 15 (backtrack);

(b) Otherwise (DB $< Z^*$) make $j' = j' + 1$ and return to Step 6.

Step 15. Backtrack. Mark entry $m_{k'j'}$ (= x_i, the latest median vertex to which $x_{j'}$ had been allocated before the backtracking step) as MT3. Then:

(a) If the bactracking step is from within the first column of M and all entries are now marked in this column, stop. The tree search has been completed and the solution corresponding to the current value of Z* is the optimal solution $\overline{X}_p$ of the p-median problem.

(b) Otherwise:

(i) Unmark, for all columns j, $j' < j \leqq n$, all entries marked either as $MT2_{j'}$ or as MT3;

(ii) Discard all previously forced allocations (see Observation 4 of 5.2.1) for all columns j, $j' < j \leqq n$;

(iii) Furthermore, if the current backtracking step un-assigns one median (i.e. if no other vertices x_j corresponding to columns $j < j'$ are allocated to vertex $x_i = m_{k',j'}$), for all columns $\underline{j}$, $j' < j \leq n$, unmark all entries marked as $MT1_i$.

After (i), (ii) and (iii) above go to Step 16 below.

Step 16. If any entry in column j' remains unmarked return to Step 6. Otherwise make j' = j' - 1 and return to Step 15 (backtrack).

5.2.5 The Algorithm Illustrated

The branch-and-bound algorithm is now illustrated. The actual tree resulting from the application of the method to find the optimal 2-median of a 10-vertex network is shown for 4 different cases. The purpose of this illustration is twofold: to give a pictorial view of the search, and to show how the efficiency of the method can be improved by (i) the use of strong lower bounds, and (ii) the availability of an upper bound prior to the start of the search.

The network used is the 10-vertex example of Garfinkel et al. [41, p.231]. This network has already been used for illustrative purposes in previous chapters of this thesis and elsewhere in the literature. For the sake of convenience it is repeated in Figure 5.2. The numbers on the links represent distance between vertices, and all vertices are equally weighted.

The optimal 2-median of the network of Figure 5.2 is $\overline{X}_2 = \{X_5, X_{10}\}$. This is shown in Figure 5.3, where the allocations of vertices to medians are clearly indicated. The cost of the optimal solution is 47.

In order to emphasize the importance of strong bounds on the efficiency of the algorithm, the effect that each of the 3 bounds

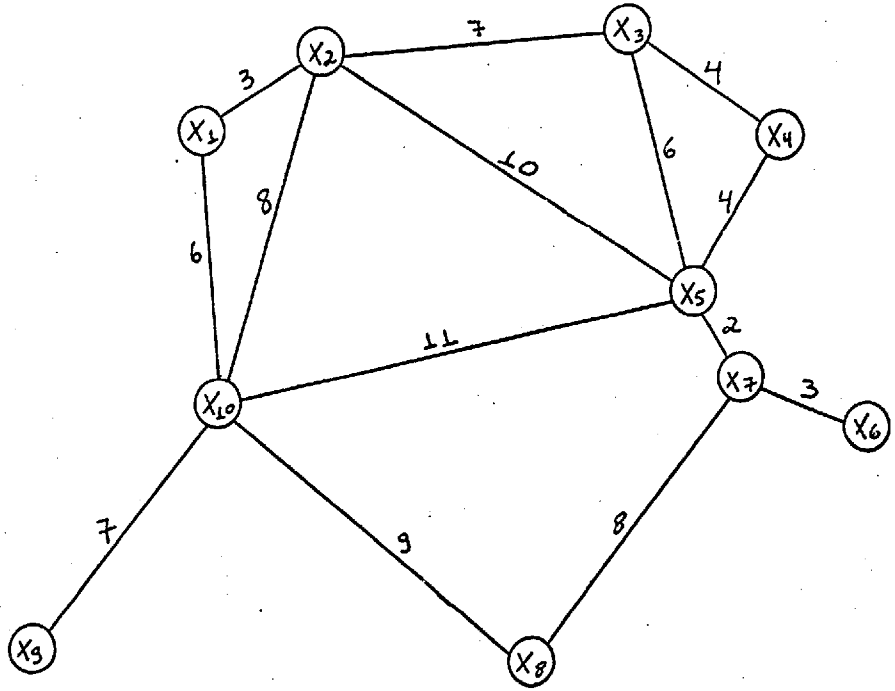
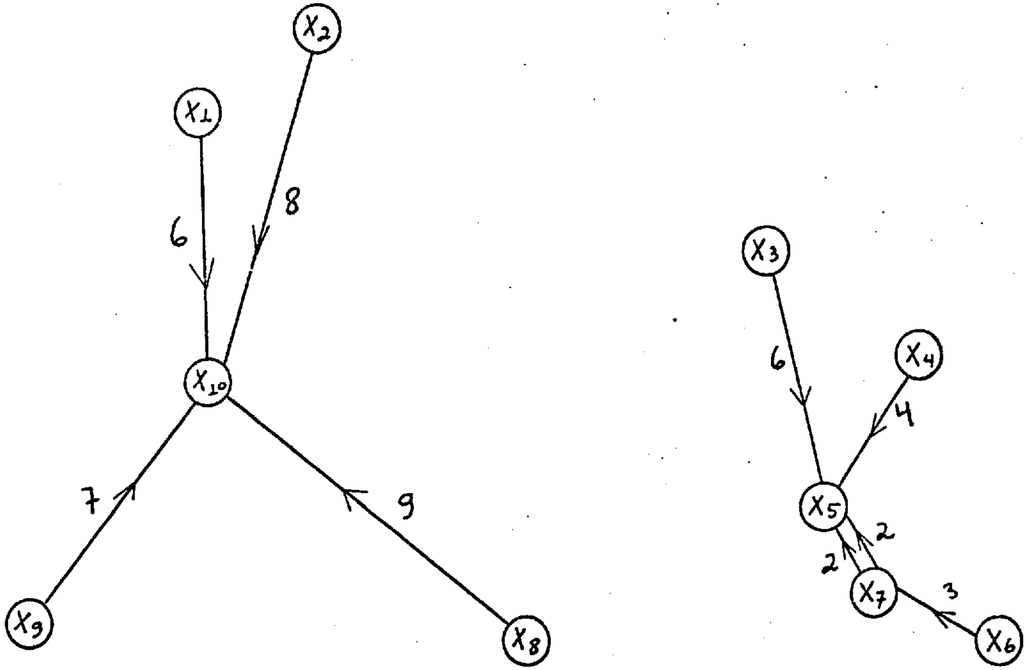


Figure 5.2

10-vertex network of Garfinkel et al. [41, p.231]



(5.3a)

Allocations to Median X_{10}

(5.3b)

Allocations to Median X_5

Figure 5.3

Allocations of Vertices to Medians

described in detail in Chapter 4 has on the size of the search is shown separately. In addition, the tree that results when an upper bound is used in conjunction with the tightest of the 3 bounds (the dual bound), is shown in a separate figure.

The tree search resulting from the use of the shortest distance bound is shown in Figure 5.5. Figures 5.6 and 5.7 show the trees corresponding to the use of the graph-theoretical bound and the dual bound, respectively. In all these 3 cases it was assumed that no upper bound was available prior to the start of the search, and the variable Z*, denoting the best available solution at stage _q_ of the tree search, was consequently set to infinity at the initial node I of each of the corresponding figures. Figure 5.8 corresponds to an upper bound being available at node I, and in this case Z* was initially set to 48, a value obtained through the vertex substitution heuristic method of Teitz and Bart [98].

Figures 5.5 to 5.8 speak for themselves, and the bounds performed in the way expected from the analysis presented in Chapter 4. The dual bound is by far the strongest of the three, and the shortest distance bound the weakest. Although the embedding of the graph-theoretical bound was not coded, the tree resulting from its application to the network of Figure 5.2 is shown in Figure 5.6.

The embedding of the graph-theoretical bound presents no special problems. It involves the transformation of the original network into smaller networks as the search develops from one level of the tree to the next and nonmedian vertices are allocated to assigned medians. Conversely, the network must be restored to its shape at higher levels of the tree every time backtracking takes place.

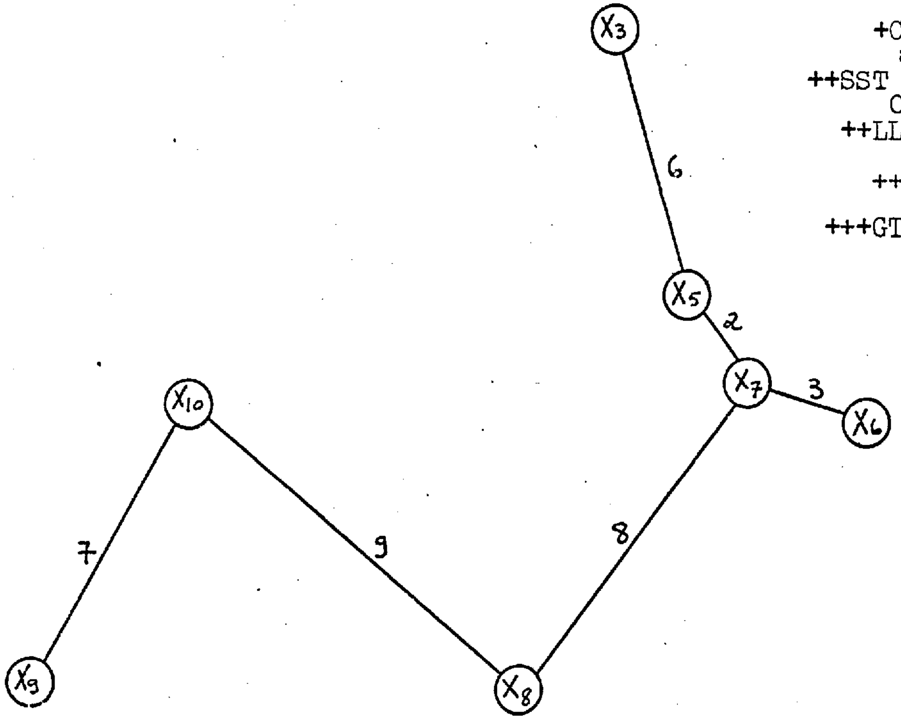
Each and every time the bound is calculated the procedure described in Section 4.3 must be applied to the complete network, described by its distance matrix at the appropriate level of the

+ C_a is the cost of the allocation of nonmedian vertices of the partial solution at node 27 of Figure 5.6

++ The meaning of SST_{ON} and LL_1 are given in 4.3.2.

The meaning of β is given in 4.3.3.

+++ GTB means the value of the graph-theoretical bound



+C_a = $d_{31}+d_{32}+d_{34}$ = 10+7+4 = 21

++SST_{ON} = 35

++LL_1 = 9

++β = 0

+++GTB = $C_a+SST_{ON}-LL_1$ = 21+35-9 = 47

Figure 5.4

The graph-theoretical bound computed at node 27 of Figure 5.6

+LB = Shortest distance bound

++FS = Feasible solution not better than Z*

+++Z_{op} = Value of optimal solution

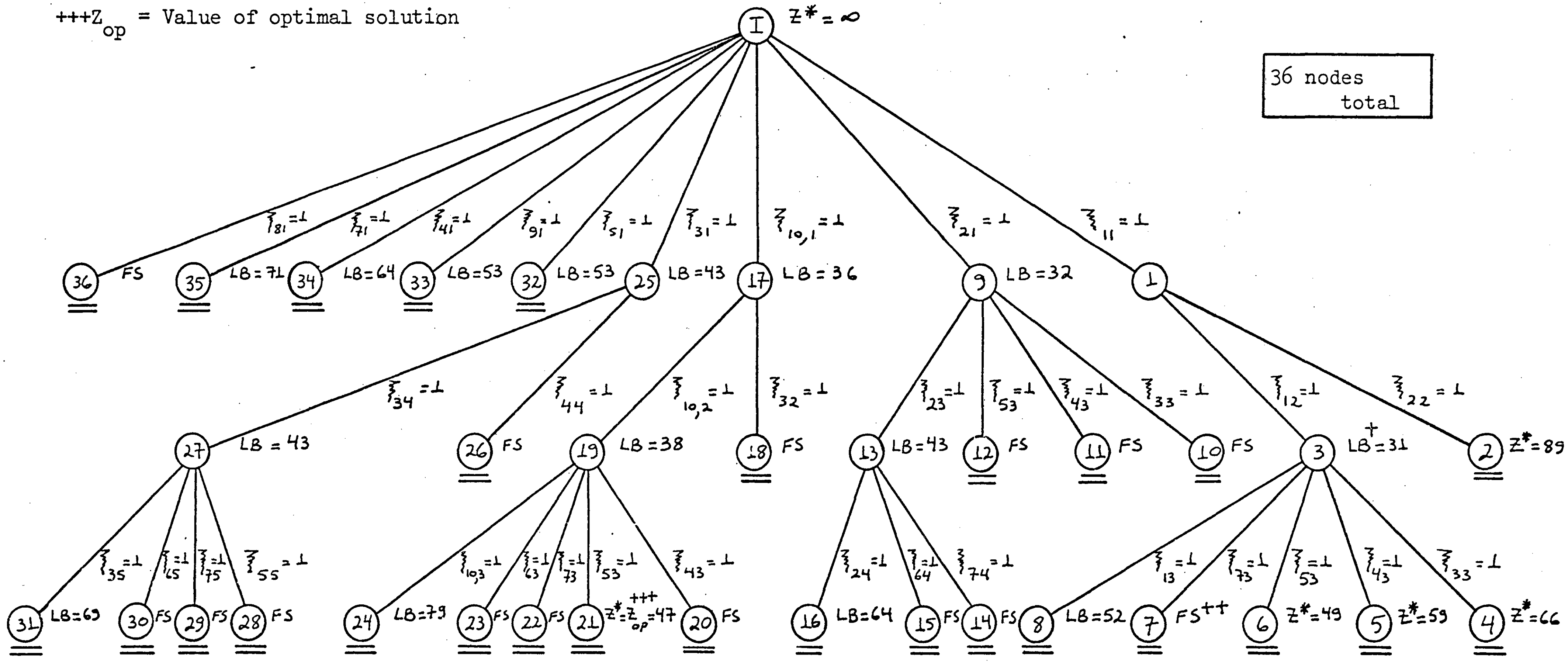


Figure 5.5

The tree search: shortest distance bound used

+GTB = Graph-theoretical bound

++FS = Feasible solution not better than Z*

+++Z_{op} = Value of optimal solution

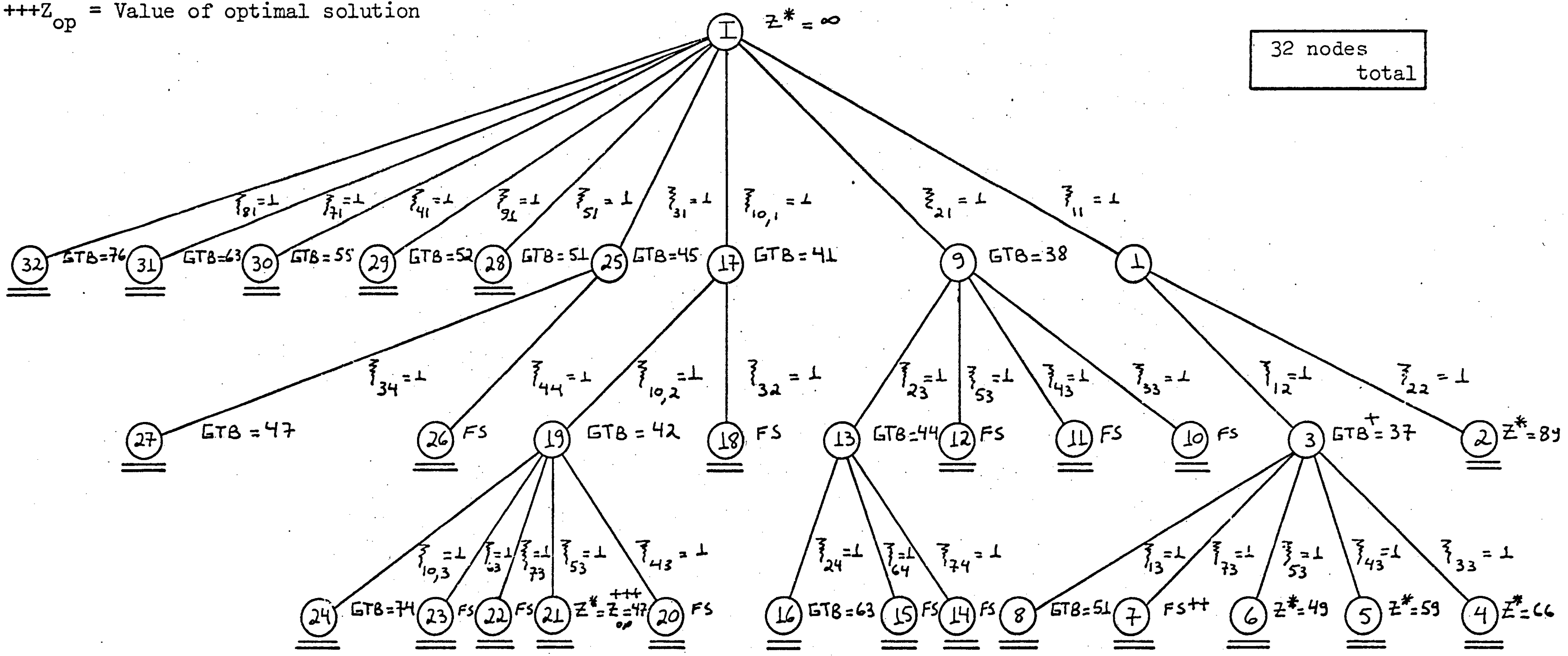
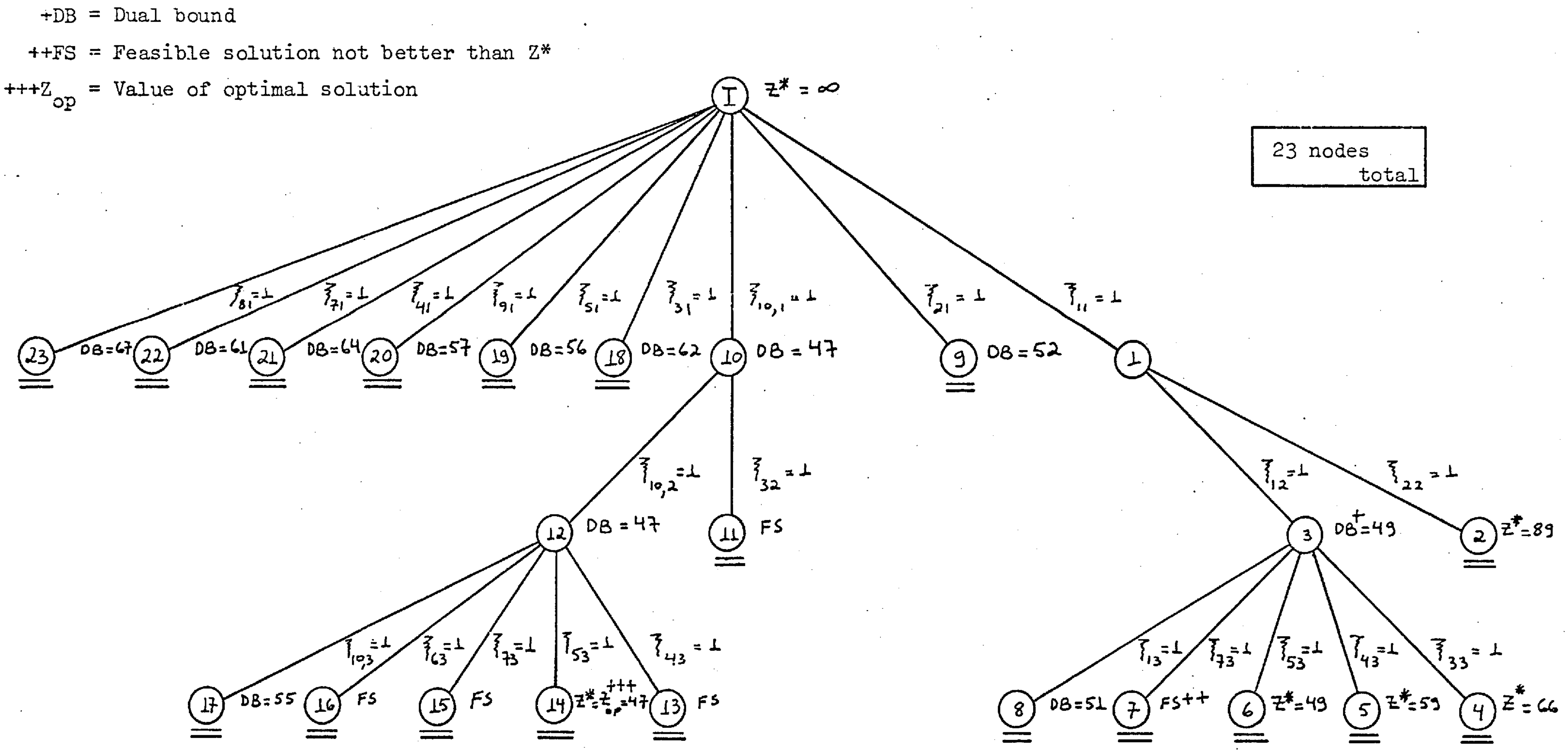


Figure 5.6

The tree search: graph-theoretical bound used



+DB = Dual bound

++FS = Feasible solution not better than Z*

+++Z_{op} = Value of optimal solution

Figure 5.7

The tree search: dual bound used

+DB = Dual bound

++FS = Feasible solution not better than Z*

+++Z_{op} = Value of optimal solution

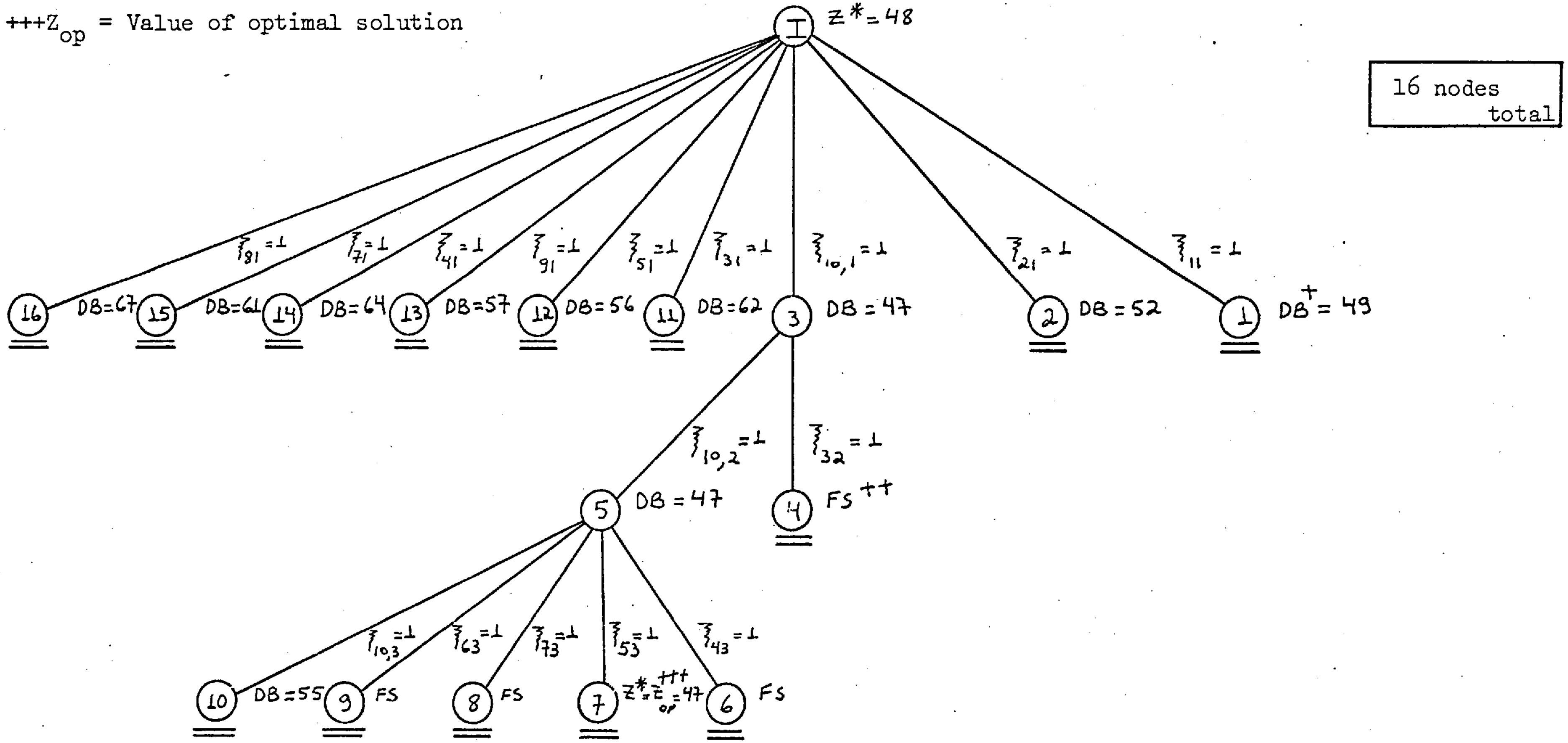


Figure 5.8

The tree search: dual bound plus initial upper bound used tree. The marked entries of matrix M (see Section 5.2.1) must be used to prevent arcs being allowed between a vertex x_j and any vertex corresponding to a marked entry in column j of M. The detailed computation of the graph-theoretical bound at node 27 of the tree of Figure 5.6 is shown in Figure 5.4.

Important parameters of the searches of Figures 5.5 to 5.8 are shown in Table 5.1 below. A substantial improvement in efficiency is noted when the searches of Figures 5.5 and 5.8 are compared. The number of nodes examined dropped from 36 to 16, due to a marked improvement in the performance of the respective bounds.

Table 5.1 - The 4 Searches Compared

Bound Used	No. Nodes Examined	No. times bound was computed	No. complete solutions evaluated
1. Shortest Distance Bound	36	16	20
2. Graph-Theoretical Bound	32	15	16
3. Dual Bound	23	12	10
4. Dual Bound plus initial upper bound	16	11	5

It is important to note that the dominance of the graph-theoretical bound over the shortest distance bound, proved in 4.3.5 for the initial node I of the tree, does not hold for its lower levels. This is caused by the fact that the marking of entries in matrix M strengthens the shortest distance bound more than it strengthens the graph-theoretical bound, especially at the lower levels of the tree.

Finally, as a matter of interest, it is worth mentioning that if complete enumeration were used to find the optimal 2-median of the network of Figure 5.2, a total of $\binom{10}{2} = 45$ feasible solutions would have had to be examined for this particular example. Quite clearly (and in common with tree search methods used to solve other combinatorial problems), the relative differences in efficiency between algorithms that result from the use of different bounds increase with the size of the problem.

5.3 Computational Results

Computational results obtained for networks ranging from 10 to 30 vertices, and for a wide range of values of p, are shown in Tables 5.2 to 5.5. The networks to which these results correspond are the same used to produce the dual bound data of Table 4.1. Except for the 10-vertex network of Garfinkel et al. (see Figure 5.2), all other networks used to produce the results of Tables 5.2 to 5.5 are described in the appendix.

The results shown in Table 5.2 correspond to the algorithm described in Section 5.2.4. An upper bound on the overall optimal solution of the problem was always obtained prior to the start of the search, and cascading through the shortest distance and dual bounds was used in all cases. The upper bound was obtained through the vertex substitution method of Teitz and Bart.

This combination of bounds has proved to be quite efficient for this particular algorithm. The advantage of using the two lower bounds is not a reduction in the number of nodes that needs to be examined before the completion of the algorithm. Although it has not been possible to prove dominance of the dual bound over the shortest distance bound, there are grounds to believe that in the vast majority of cases the number of nodes examined would be the same if the dual bound had been used on its own.

The advantage of cascading through the two lower bounds is a substantial reduction in computing times, given that the shortest distance bound is much faster to compute than the dual bound. In these circumstances, any backtracking caused by the shortest distance bound avoids a corresponding computation of the dual bound, saving computing time. This saving is substantial,especially for the larger values of n and p. This can be easily verified from the data shown in Table 5.2.

In Table 5.2 data is provided on the main parameters of the tree search, with detailed information provided on each of the two lower bounds used. Most of the data provided are self-explanatory, but the number of nodes examined and the time spent on the dual bound need a closer examination. They are the two factors that determine the total computer time needed to find the optimal solution to any given problem, and as a consequence they ultimately limit the size of problems that can be solved through the present branch-and-bound algorithm.

The p-median problem belongs to the NP class of combinatorial problems [58], and consequently the number of steps needed before completion of the present branch-and-bound algorithm is of the order of $O(K^m)$, where K is a constant and m is a function of the number of vertices of the network (n) and of the number of medians being sought (p), i.e. $m = f(n,p)$. Of the two n is the main determining factor. On the other hand, for any given n the number of nodes examined increases with p up to a certain point ($\simeq n/3$ for the larger networks), and decreases thereafter down to 1 node when $p = n$. The relationship between number of nodes examined and p, however, does not always follow a very rigid pattern, a very good example of which are the results obtained for the 30-vertex network of Table 5.2.

Given the number of steps needed to obtain the optimal solution, the computer time spent on the dual bound is the main limiting factor in the branch-and-bound algorithm. For $n \geq 15$ the number of nodes examined is only a very small fraction of the total enumeration for the problem, but the computer time spent on the dual bound represents a substantial percentage of the total time needed to complete the tree search. These are facts of special relevance for the large values of n and p, as demonstrated in Table 5.3.

The data of Table 5.3 strongly substantiates the point that the time needed to compute the dual bound is the algorithm's bottleneck, preventing it from solving the p-median problem for networks with more than 30 vertices, except for small values of p. Any improvement in the performance of the algorithm is therefore dependent on the ability to improve the computational performance of the dual bound.

Two additional tables complete the set of computational results. In Table 5.4 the effect that the use of an initial upper bound has on the efficiency of the algorithm is clearly indicated. For many of the examples included in this table (and especially for the larger networks), a significant reduction both in number of nodes examined and in total computing time is shown when an upper bound is available at the initial node I of the tree.

Finally, in Table 5.5 the cost of seeking possible multiple solutions for the p-median problem is shown for a number of examples. A significantly larger number of iterations may be required when possible multiple solutions are being investigated, especially for large values of n and p. Table 5.5 indicates that, contrary to other combinatorial problems, multiple optimal solutions are not a rare occurrence in the p-median problem.

In summary, the computational results of the present section support the claim made in Section 5.1, that the branch-and-bound algorithm developed in this thesis represents a substantial advancement in the field of exact solution procedures for the p-median problem. The algorithm guarantees an optimal solution for any value of p

Table 5.2 - The Branch-and-bound algorithm: computational results*

Problem size				Shortest distance bound			Dual Bound			Time in Seconds+		
n	p	Value of optimal solution	Number of nodes examined	No. of calls	No. backtrack steps caused	Computing time+ spent on bound	No. of calls	No. backtrack steps caused	Computing time+ spent on bound	Upper bound	Tree search	Total time
10++	1	79	10	0	0	0.00	0	0	0.00	0.01	0.06	0.07
10	2	47	16	11	6	0.01	5	3	0.03	0.01	0.05	0.06
10	3	36	25	19	4	0.01	15	10	0.06	0.01	0.09	0.10
10	4	26	33	26	7	0.01	19	10	0.08	0.02	0.11	0.13
10	5	18	29	21	9	0.01	12	4	0.06	0.02	0.09	0.11
10	6	12	25	19	9	0.01	10	3	0.07	0.02	0.10	0.12
10	7	8	23	20	10	0.01	10	3	0.08	0.01	0.10	0.11
10	8	5	15	13	7	0.01	6	1	0.05	0.01	0.07	0.08
10	9	2	16	8	2	0.00	6	0	0.05	0.01	0.06	0.07
10	10	0	1	1	1	0.00	0	0	0.00	0.00	0.01	0.01
10	1	400	10	0	0	0.00	0	0	0.00	0.01	0.04	0.05
10	2	273	18	7	3	0.00	4	3	0.03	0.01	0.04	0.05
10	3	195	26	17	7	0.01	10	4	0.12	0.01	0.15	0.16
10	4	149	22	18	9	0.01	9	4	0.16	0.02	0.19	0.21
10	5	107	24	18	6	0.01	12	6	0.20	0.02	0.23	0.25
10	6	75	22	17	5	0.01	12	6	0.23	0.02	0.27	0.29
10	7	43	25	15	2	0.00	13	5	0.19	0.01	0.22	0.23
10	8	15	22	10	1	0.00	9	1	0.13	0.01	0.15	0.16
10	9	2	18	8	0	0.00	8	0	0.13	0.01	0.15	0.16
10	10	0	1	1	1	0.00	0	0	0.00	0.00	0.01	0.01

+ CPU time, in CDC 7600 seconds

++ Garfinkel et al. example (see Figure 5.2)

* Only one optimal solution sought

Table 5.2 (cont'ed) - The Branch-and-bound algorithm: computational results*

Problem size				Shortest distance bound			Dual Bound			Time in Seconds$^+$		
n	p	Value of optimal solution	Number of nodes examined	No. of calls	No. backtrack steps caused	Computing time$^+$ spent on bound	No. of calls	No. backtrack steps caused	Computing time$^+$ spent on bound	Upper bound	Tree search	Total time
15	1	809	15	0	0	0.00	0	0	0.00	0.02	0.10	0.12
15	2	412	22	13	3	0.01	10	9	0.07	0.03	0.13	0.16
15	3	294	36	28	9	0.02	19	16	0.16	0.05	0.28	0.33
15	4	215	87	69	22	0.04	47	32	0.65	0.08	0.82	0.90
15	5	150	41	31	14	0.02	17	10	0.40	0.08	0.47	0.55
15	6	113	41	28	10	0.01	18	10	0.54	0.09	0.64	0.73
15	7	93	45	33	11	0.01	22	13	0.69	0.07	0.79	0.86
15	8	74	82	57	23	0.02	34	13	0.97	0.03	1.09	1.12
15	9	57	42	27	13	0.01	14	4	0.57	0.03	0.65	0.68
15	10	41	46	31	15	0.01	16	4	0.56	0.02	0.65	0.67
20	1	1159	20	0	0	0.00	0	0	0.00	0.04	0.25	0.29
20	2	724	33	17	7	0.02	10	8	0.10	0.10	0.33	0.43
20	3	518	118	88	22	0.08	66	54	1.13	0.15	1.86	2.01
20	4	414	188	150	59	0.14	91	75	2.13	0.13	3.33	3.46
20	5	338	238	195	82	0.17	113	85	3.10	0.19	4.35	4.54
20	6	259	167	116	62	0.10	54	34	2.63	0.25	3.43	3.68
20	7	227	137	92	37	0.07	55	30	2.69	0.20	3.19	3.39
20	8	199	155	120	47	0.08	73	41	3.03	0.25	3.55	3.80
20	9	175	156	120	52	0.08	68	37	3.22	0.19	3.68	3.87
20	10	151	191	156	66	0.09	90	44	3.87	0.13	4.32	4.45

+ CPU time, in CDC 7600 seconds

* Only one optimal solution sought

Table 5.2 (cont'ed) - The Branch-and-bound algorithm: computational results*

Problem size		Value of	Number	Shortest distance bound			Dual bound			Time in Seconds+		
		optimal	of nodes	No. of	No. backtrack	Computing time+	No. of	No. backtrack	Computing time+	Upper	Tree	Total
n	p	solution	examined	calls	steps caused	spent on bound	calls	steps caused	spent on bound	bound	search	time
25	1	1352	25	0	0	0.00	0	0	0.00	0.08	0.53	0.61
25	2	956	33	24	7	0.04	17	16	0.19	0.17	0.75	0.92
25	3	722	139	112	19	0.16	93	80	2.08	0.38	3.64	4.02
25	4	556	230	192	45	0.27	147	124	4.07	0.37	6.31	6.68
25	5	468	241	210	63	0.27	147	121	5.63	0.42	7.74	8.16
25	6	387	381	292	118	0.35	174	133	7.60	0.45	10.69	11.14
25	7	341	731	627	242	0.63	385	288	13.90	0.59	18.76	19.35
25	8	298	922	837	294	0.92	543	399	25.46	0.84	31.32	32.16
25	9	266	1012	904	323	0.95	581	416	31.33	0.72	37.04	37.76
25	10	235	715	643	241	0.66	402	264	24.07	0.59	27.59	28.18
25	11	210	613	533	236	0.53	297	178	21.45	0.98	24.23	25.21
25	12	188	726	619	287	0.57	332	186	21.12	0.62	24.06	24.68
30	1	1432	30	0	0	0.00	0	0	0.00	0.11	0.95	1.06
30	2	936	46	28	8	0.07	20	19	0.27	0.30	1.32	1.62
30	3	777	237	193	60	0.04	133	119	4.04	0.69	8.72	9.41
30	4	610	262	219	70	0.05	149	131	6.95	1.01	11.90	12.91
30	5	516	833	715	180	1.32	535	442	24.57	1.35	36.71	38.06
30	6	438	675	564	198	0.98	366	292	28.87	1.92	38.20	40.12
30	7	386	1304	1131	448	1.68	683	512	47.94	1.59	62.73	64.32
30	8	337	794	664	276	1.10	388	301	52.06	2.14	61.65	63.79
30	9	294	388	322	144	0.57	178	133	37.89	1.91	42.64	44.55
30	10	265	1028	922	444	1.44	478	323	71.35	1.67	81.20	82.87

+ CPU time, in CDC 7600 seconds

* Only one optimal solution sought

Table 5.3 - No. of nodes examined and time spent on the dual bound*

Problem size		Total[+] Enumeration (No. of solutions)	No. of Nodes Examined	% of Total Enumeration	% of total search time spent on the dual bound
n	p				
15	4	1,365	87	6.37	79
15	5	3,003	41	1.37	85
15	6	5,005	41	0.82	84
15	7	6,435	45	0.70	87
15	8	6,435	82	1.27	89
20	4	4,845	188	3.88	64
20	5	15,504	238	1.54	71
20	6	38,760	167	0.43	77
20	7	77,520	137	0.18	84
20	8	125,970	155	0.12	85
25	8	1,081,575	922	0.085	81
25	9	2,042,975	1012	0.050	85
25	10	3,268,760	715	0.022	87
25	11	4,457,400	613	0.014	89
25	12	5,200,300	726	0.014	88
30	6	593,775	675	0.1100	76
30	7	2,035,800	1304	0.0640	76
30	8	5,852,925	794	0.0140	84
30	9	14,307,150	388	0.0027	89
30	10	30,045,015	1028	0.0034	88

* Only one optimal solution sought, cascading and upper bound always used.

+ Total enumeration = $\binom{n}{p} = \frac{n!}{(n-p)!p!}$ solutions, each requiring $p(n-p)$ comparisons and $(n-p)$ additions in order to be evaluated.

Table 5.4 - No upper bound vs. initial upper bound*

Problem size		Value of Optimal solution	Number of nodes examined		Total time in seconds+	
n	p		No upper bound used	Upper bound used	No upper bound used	Upper** bound used
10++	1	79	10	10	0.03	0.07
10	2	47	23	16	0.06	0.06
10	3	36	38	25	0.10	0.10
10	4	26	65	33	0.13	0.13
10	5	18	74	29	0.11	0.11
10	6	12	77	25	0.12	0.12
10	7	8	59	23	0.10	0.11
10	8	5	30	15	0.04	0.08
10	9	2	20	16	0.04	0.07
10	10	0	1	1	0.01	0.01
10	1	400	10	10	0.06	0.05
10	2	273	18	18	0.07	0.05
10	3	195	41	26	0.14	0.16
10	4	149	43	22	0.21	0.21
10	5	107	48	24	0.22	0.25
10	6	75	45	22	0.25	0.29
10	7	43	50	25	0.20	0.23
10	8	15	39	22	0.12	0.16
10	9	2	18	18	0.12	0.16
10	10	0	1	1	0.01	0.01

* Only one optimal solution sought, cascading always used

+ CPU time, in CDC 7600 seconds

++ Garfinkel et al. example (see Figure 5.2)

** Includes time spent on the computation of the upper bound

Table 5.4 (cont'ed) - No upper bound vs. initial upper bound*

Problem size		Value of	Number of nodes examined		Total time in seconds+	
		Optimal	No upper	Upper	No upper	Upper**
n	p	solution	bound used	bound used	bound used	bound used
15	1	809	15	15	0.11	0.12
15	2	412	55	22	0.29	0.16
15	3	294	47	36	0.33	0.33
15	4	215	129	87	0.88	0.90
15	5	150	150	41	0.87	0.55
15	6	113	128	41	0.99	0.73
15	7	93	134	45	1.14	0.86
15	8	74	96	82	0.86	1.12
15	9	57	60	42	0.39	0.68
15	10	41	52	46	0.28	0.67
20	1	1159	20	20	0.26	0.29
20	2	724	108	33	0.92	0.43
20	3	518	183	118	2.36	2.01
20	4	414	211	188	3.34	3.46
20	5	338	323	238	4.68	4.54
20	6	259	376	167	4.17	3.68
20	7	227	278	137	3.13	3.39
20	8	199	295	155	3.16	3.80
20	9	175	277	156	3.30	3.87
20	10	151	290	191	3.68	4.45

* Only one optimal solution sought, cascading always used

+ CPU time, in CDC 7600 seconds

** Includes time spent on the computation of the upper bound

Table 5.4 (cont'ed) - No upper bound vs. initial upper bound*

Problem size		Value of Optimal solution	Number of nodes examined		Total time in seconds+	
n	p		No upper bound used	Upper bound used	No upper bound used	Upper** bound used
25	1	1352	25	25	0.54	0.61
25	2	956	51	33	1.33	0.92
25	3	722	195	139	5.36	4.02
25	4	556	386	230	10.96	6.68
25	5	468	506	241	15.61	8.16
25	6	387	716	381	20.53	11.14
25	7	341	1161	731	30.75	19.35
25	8	298	1638	922	44.60	32.16
30	1	1432	30	30	1.00	1.06
30	2	936	160	46	4.59	1.62
30	3	777	337	237	13.86	9.41
30	4	610	545	262	23.75	12.91
30	5	516	1524	833	62.25	38.06

* Only one optimal solution sought, cascading always used

+ CPU time, in CDC 7600 seconds

** Includes time spent on the computation of the upper bound

Table 5.5 - Single vs. multiple optimal solutions[+]

Problem size		Value of optimal solution	Number of nodes examined		Number of multiple solutions
n	p		One solution	Multiple solutions	
10	2	273	18	18	1
10	3	195	41	41	1
10	4	149	43	53	1
10	5	107	48	62	1
10	6	75	45	81	3
10	7	43	50	69	2
10	8	15	39	84	4
10	9	2	18	25	2
10	10	0	1	10	1
15	2	412	55	55	1
15	3	294	47	55	2
15	4	215	129	148	2
15	5	150	150	169	1
15	6	113	128	202	8
15	7	93	134	311	16

[+] No upper bound used

Table 5.5 (cont'ed) - Single vs. multiple optimal solutions[+]

Problem size		Value of Optimal solution	Number of nodes examined		Number of multiple solutions
n	p		One solution	Multiple solutions	
20	2	724	108	108	1
20	3	518	183	200	1
20	4	414	211	211	1
20	5	338	323	366	1
20	6	259	376	388	1
20	7	227	278	331	2
20	8	199	295	545	4
20	9	175	277	713	10
20	10	151	290	563	4
25	2	956	51	51	1
25	3	722	195	205	2
25	4	556	386	434	2
25	5	468	506	653	8
25	6	387	716	1118	16
30	2	936	160	160	1
30	3	777	337	365	1
30	4	610	545	607	1

[+] No upper bound used

for networks with up to 30 vertices, within a reasonable amount of computer time. This is not matched by any other exact procedure available in the literature [30, 41, 55, 78].

5.4 The LP relaxation and the Branch-and-bound algorithm

The possibility of using the LP relaxation of the p-median problem to provide bounds for branch-and-bound algorithms has already been mentioned in Section 4.2. When the LP relaxation was investigated, both the general and the decomposition formulations of this relaxation were embedded into the branch-and-bound algorithm of the present chapter. The experience with these embeddings is now reported.

The LP relaxation must be used to solve the complete problem before the branch-and-bound algorithm is activated. Very often the solution produced for the relaxed problem is all-integer, being therefore the optimal solution for the p-median problem. Only if the LP relaxation produces a fractional solution at this initial stage of the procedure, should the tree search be activated.

When the branch-and-bound algorithm is activated, the subproblems, generated by the setting of some of the ξ_{ij} variables to zero or 1, can be solved by either of the formulations of the relaxed problem. At a given stage _q_ of the search backtracking occurs if:

either (i) The solution to the LP relaxation of the subproblem is all-integer. In such cases this solution is obviously the optimal completion of the partial solution corresponding to the allocations made up to stage _q_ of the tree search;

or (ii) The solution is fractional, but the value of its objective function, plus the cost of the allocations made up to stage _q_ of the search, provide a lower bound that is greater than or equal to the best available solution at stage _q_ (Z^*).

Due to the very large linear programmes produced by the general formulation, it soon became evident that the embedding of this formulation could not produce results of any significance. Besides the fact that very large LP's are already produced for 20-vertex networks, the approach proved not to be practical even for a 10-vertex network.

In this respect, the search for the optimal 3-median of the network of Figure 5.2 provided the following results:* Although the number of nodes examined was reduced from 25 (see Table 5.2) to only 8 when this embedding was used, the total computing time increased from 0.10 to 8.32 CDC 7600 seconds. This was due to the long time taken to solve the 5 LP's that were needed to terminate the search.

The embedding of the decomposition formulation, however, provided better grounds for hope. This is a fast algorithm, requiring in addition little computer core. There was also hope that some of the convergence problems reported in Chapter 3 could be solved after some of the ξ_{ij}'s had been fixed along the tree search. The experience with the embedding of this formulation is reported in the next two sections.

5.4.1 The Embedding of the LP Decomposition Formulation

The embedding of the LP decomposition formulation involves solving the linear programme for the ξ_{ij} variables not yet fixed to either zero or 1 when the LP is activated. For this formulation, the setting of variables is taken care of in each of the n sub-problems of Section 3.3. It influences therefore the vector to enter the basis at each iteration of the LP.

* As already pointed out in Chapter 3, for all values of $p \neq 3$ the LP solution was all-integer for this particular example.

The procedure is better explained by means of an example. Refer back to Figure 5.2, and assume the method is being used to determine the optimal 3-median of the corresponding network. Assume further that the LP must be solved at the node of the tree corresponding to vertices x_1 and x_2 having been assigned as medians, with vertex x_3 allocated to median x_2. Vertices x_4 to x_{10} are not as yet allocated to any median vertex.*

The first point that must be made is that vertex x_3 cannot be assigned as a median in any branch emanating from the node described above. Vertex x_3, therefore, should never be brought into the basis of the corresponding master problem. To make sure this will not happen, the vector corresponding to subproblem 3 must be a zero vector, i.e. $y_3^* = (0, \ldots, 0)^T$ in every iteration of the algorithm.

Secondly, in all other possible candidate vectors y_i^* to enter the basis, $i \neq 3$, the top 3 entries are already determined by the allocations of vertices x_1, x_2 and x_3. These allocations imply:

$$\text{(i)}\quad y_{11}^* = y_{22}^* = y_{23}^* = 1\ , \qquad \text{(ii)}\quad y_{12}^* = y_{21}^* = y_{13}^* = 0,$$

and $\quad$ (iii) $\quad y_{i1}^* = y_{i2}^* = y_{i3}^* = 0, \quad i = 4,5,\ldots,10$.

In summary, at this particular stage of the tree search the candidate vectors to enter the basis of the master problem are given by

$$y_1^* = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \end{bmatrix}, \qquad y_2^* = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \end{bmatrix}$$

* The situation described above never happened for this particular example, and the development that follows is only for illustrative purposes. The actual tree search for this example is shown in Figure 5.9.

$$y_4^* = y_5^* = y_6^* = y_7^* = y_8^* = y_9^* = y_{10}^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \\ \delta \end{bmatrix},$$

where the δ's are to be replaced by either zero or 1, in the way described in Section 3.3 [see equations (3.25) and (3.26)] . The decomposition formulation can then procede as described in Chapter Three.

5.4.2 Computational Experience with the Embedding

Computational results corresponding to the embedding of the decomposition formulation are shown in Table 5.6. In this table these results are also compared with data taken from Table 5.2. In addition, the tree search corresponding to the 10-vertex network appearing in this table is illustrated in Figure 5.9.

Except for the search shown in Figure 5.9, the smallest network for which it was possible to test the embedding of the decomposition formulation was the 20-vertex network of Table 5.2. For smaller networks all-integer solutions were obtained for the complete problem, and no tree search was needed. For the 20-vertex and larger networks of Table 5.2, however, the decomposition formulation failed to converge after 1,000 iterations for several values of <u>p</u> (see Table 3.1), thus enabling the embedding of this LP formulation to be tested.

Four different examples are shown in Table 5.6. For 3 of them it was possible to complete the search within 150 seconds of computer time in the CDC 7600, but for the n = 25, p = 4 example this was not possible.

Table 5.6 - The embedding of the LP decomposition

Problem size						Convergence of the LP					
n	p	Sol. Method[+]	Value of optimal solution	No. of nodes examined	No. of LP calls	No. of calls that converged	No. of calls not converging	Max. No.[*] iter./call	Search[+++] completed?	Avg. time[++] to solve LP given convergence	Total[++] time in seconds
10	3	1	36	25	-	-	-	-	-	-	0.10
10	3	2	36	8	5	5	0	40	Yes	0.07	0.25
20	6	1	259	167	-	-	-	-	-	-	3.68
20	6	2	259	72	23	15	8	700	Yes	2.04	65.56
20	7	1	227	137	-	-	-	-	-	-	3.39
20	7	2	227	128	51	34	17	700	Yes	1.33	137.32
25	4	1	556	230	-	-	-	-	-	-	6.68
25	4	2	-	-	25	7	18	650	No	3.65	>150.00

[+] Solution method 1: same as in Table 5.2
Solution method 2: tree search with LP decomposition embedded

[++] CPU time, in CDC 7600 seconds

[+++] "No" means search not completed within 150 CDC 7600 seconds

[*] Input value. Further branching takes place if a subproblem does not converge within this number of iterations.

+LP = Value of the objective function of the linear programme

++Z_{op} = Value of optimal solution

*LB = Shortest distance bound

**C_a = Cost of allocations at this node

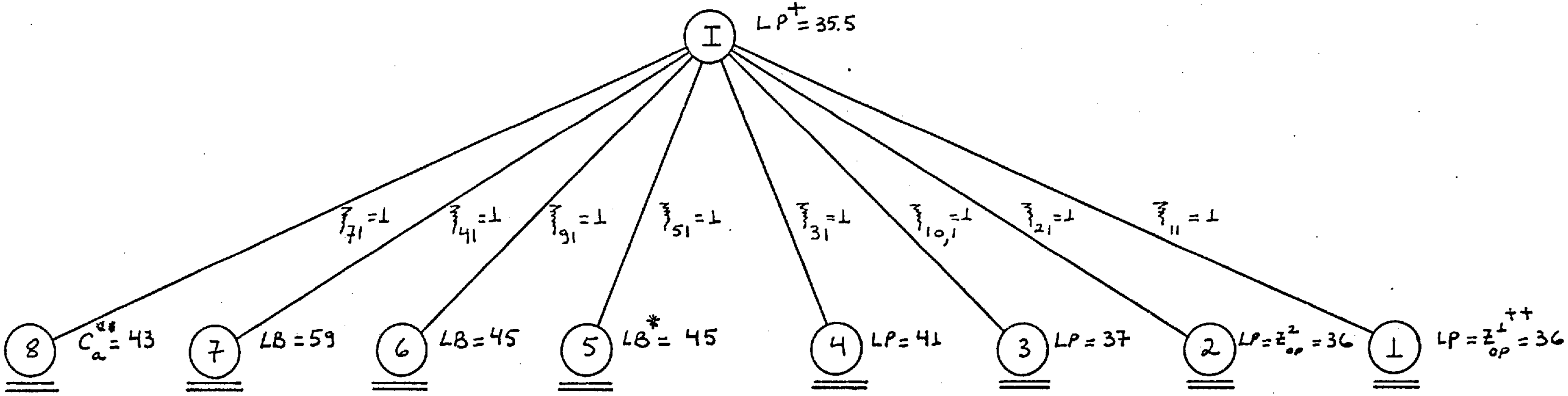


Figure 5.9

The tree search with the LP decomposition embedded (optimal 3-median of the 10-vertex network of Figure 5.2)

The results presented in Table 5.6 not only show that the embedding of the decomposition formulation is by no means competitive with the algorithm described in 5.2.4, but also that this LP formulation fails to provide a valid alternative for solving the p-median problem.

Lack of convergence continues to be a major stumbling block in the decomposition formulation, even after some of the ξ_{ij}'s have been fixed. The percentage of subproblems that do not converge when the LP is activated increases with the value of n, and this prevented the search for the n = 25, p = 4 test case from being completed within the time limit of 150 seconds.

It can be safely concluded that, if the basic problems of the decomposition formulation cannot be solved, this will remain a method that may solve the p-median problem only on occasion. The embedding of this formulation into a branch-and-bound algorithm did little to improve its potentiality, and it is felt that little can be done until the basic problem of convergence, caused by large-scale degeneracy, is solved.

5.5 Conclusions

A direct tree search algorithm for the p-median problem was developed in the present chapter. Two lower bounds were embedded into the search, and cascading through them proved very efficient. The shortest distance bound is weak but fast to compute. It saves computing time when it forces backtracking, as it then avoids a corresponding computation of the dual bound. The latter is a very strong bound, but relatively slow to compute. An initial upper bound obtained through heuristics helped to reduce the search further.

The computational experience reported in the present chapter represents a substantial improvement over existing exact solution procedures for the p-median problem. It produces optimal solutions for 30-vertex networks in less than 2 minutes of computer time in the CDC 7600, for any possible value of p ($1 \leq p \leq n$). The algorithm is both faster and more efficient (in terms of the number of nodes generated) than other branch-and-bound procedures available in the literature [30, 55].

Additional exact solution methods for the p-median problem, such as the LP decomposition of Garfinkel et al. [41], or the Lagrange multiplier approach of Marsten [78], may on occasion solve problems of similar size. Both methods, however, cannot guarantee an optimal solution for every possible value of p, and may fail on much smaller problems.

The relatively long computing times required for the calculation of the dual bound is the bottleneck of the present algorithm. Any improvement obtained in the computational efficiency of this bound should increase the size of problems for which an optimal solution can be found.

CHAPTER SIX

HEURISTIC METHODS

6.1 Introduction

Maranzana [76] and Teitz and Bart [98] pioneered in proposing heuristic methods for the p-median problem. Except for the work of Surkis [96] and Diehr [22], which are limited extensions of the methods of Maranzana and Teitz and Bart, respectively, not much has been done in this area.

The heuristic methods of Maranzana and Teitz and Bart have already been briefly described in Chapter 2. They are discussed in greater detail later in this chapter.

The vertex substitution method of Teitz and Bart is in fact only one of a family of algorithms based on local optimization and the idea of λ-optimality. The idea of λ-optimality was first introduced by Lin [74] for the travelling salesman problem, and subsequently extended by others [13,14,59] for a variety of combinatorial problems.

After a brief review of the earlier work mentioned above, the important theoretical contribution of Cornuejols, Fisher and Nemhauser [19] to the study of heuristics and relaxations for the uncapacitated location problem is discussed in terms of its application to the p-median problem. Then the vertex substitution method of Teitz and Bart is extended, and λ-optimal substitution methods for the p-median problem are introduced.

It will be shown later in the chapter that the number of iterations needed to ensure λ-optimality for the p-median problem increases very rapidly with λ. Hence practical algorithms cannot use values of λ much above 2 or 3. The computational experience reported in this thesis is therefore limited to the special cases of $\lambda = 1$ (the Teitz and Bart algorithm) and $\lambda = 2$.

A simple vertex addition ("greedy") heuristic, and its use as a 'pre-processor' to λ-optimal substitution algorithms, complete the work on heuristics in this thesis. Computational experience with what is described as 'the combined approach' is reported for $\lambda = 1$ and $\lambda = 2$, and these results are compared with the ones corresponding to the "pure" $\lambda = 1$ and $\lambda = 2$ optimal substitution methods.

6.2 A review of earlier work on heuristics for the p-median and related problems

In studying certain classes of location-allocation problems Cooper [17] pointed out a fortunate property of many of these problems: the lack of a sharp optimum, i.e. the existence of many alternative optimal or near-optimal solutions. This property is fortunate in that, for these problems, a well constructed heuristic has a reasonably high probability of finding one of these optimal or near-optimal solutions.

The p-median problem belongs to the set of problems having the above property, and heuristic methods designed for this problem take full advantage of this fact.

6.2.1 The partition method of Maranzana [76]

Maranzana's partition method is in some respects a discrete version of the alternate location and allocation algorithm devised by Cooper [17] for the continuous case. Let $d(x_i,x_j)$ be the length of the shortest path between vertices x_i and x_j of a network $N = (X,A)$. A formal statement of Maranzana's algorithm is as follows.

Step 1. Arbitrarily select p distinct points from the set X of all vertices of N to form the set X_p.

Step 2. Associated with the set X_p of p points (p_{x_1}, p_{x_2}, ..., p_{x_p}), determine a corresponding partition of X, P_{x_1},..., P_{x_p}, by putting

$$P_{x_i} = \left\{ p_{x_k} \middle| d(x_k,x_i) \leqq d(x_k,x_j) \text{ for all } j \neq i,\ x_i,x_j \in X_p,\ x_k \notin X_p \right\} .$$

Step 3. Determine a "centre" c_{x_i} for each P_{x_i}.

Step 4. If $c_{x_i} = p_{x_i}$ for all i, computation is stopped and the current values of p_{x_i} and P_{x_i} constitute the desired solution. Otherwise set $p_{x_i} = c_{x_i}$ for all i and return to Step 2.

In Step 2, if a point is equidistant from more than one source, this point may be arbitrarily placed in the set associated with the source p_{x_i} having the smallest i. If the "centre" is non-unique in Step 3, a likewise arbitrary decision can be made, and the point with the smallest subscript selected for "centre".

Maranzana proves that his algorithm is monotonic, i.e. that the total weighted distance value cannot increase from one iteration of the algorithm to the next. He also identifies certain conditions under which the algorithm will fail to converge to an optimal solution, but claims that with several initial choices of the p distinct points a solution close to the optimum is likely.

6.2.2 The vertex substitution method of Teitz and Bart [12,98]

A general description of the vertex substitution method has already been given in Chapter 2. A formal statement of the method is now given.

Let $\sigma(S)$ be the transmission number* for a subset S of the set X

* For the definition of the transmission number $\sigma(S)$ of a subset S of vertices of a network, refer to equations (2.19) to (2.22) of Chapter 2.

of all vertices of a network $N = (X,A)$. The algorithm is then [12]:

Step 1. Select a set S of p vertices to form the initial approximation to the optimal p-median set $\bar{X}_p$. Call all vertices $x_j \notin S$ "untried".

Step 2. Select some "untried" vertex $x_j \notin S$, and for each vertex $x_i \in S$ compute the "reduction" Δ_{ij} in the set transmission if x_j is substituted for x_i, i.e. compute

$$\Delta_{ij} = \sigma(S) - \sigma(S \cup \{x_j\} - \{x_i\}) \quad .$$

Step 3. Find $\Delta_{i_oj} = \underset{x_i \in S}{\text{Max}} [\Delta_{ij}]$. Then:

(i) If $\Delta_{i_oj} \leqq 0$ call x_j "tried" and go to Step 2.

(ii) If $\Delta_{i_oj} > 0$ set $S \leftarrow S \cup \{x_j\} - \{x_i\}$, call x_j "tried" and go to Step 2.

Step 4. Repeat steps 2 and 3 until all vertices in $(X-S)$ have been tried. This is referred to as a cycle. If during the last cycle no vertex substitution at all has been made at Step 3, go to Step 5. If some vertex substitution has been made, call all the vertices $x_j \notin S$ "untried" and return to Step 2.

Step 5. Stop. The current set S is the estimated p-median set $\bar{X}_p$.

Teitz and Bart tested their algorithm against the partition method of Maranzana. They say that the performance of the partition method may be quite erratic, and claim that their method is a preferable heuristic because it exhibits considerably less variation in performance. Teitz and Bart conclude by saying that if the partition method is used, the high variance of its error suggests that great caution in the selection of the initial locations is necessary. This apparent difficulty may be overcome by performing the computations for several initial choices of the distinct p points, as suggested by Maranzana.

6.2.3 The work of Cornuejols, Fisher and Nemhauser [19]

In a recent paper, Cornuejols et al. [19] make an analysis of heuristics and relaxations for the uncapacitated location problem. The main interest for this thesis lies on the analysis of heuristics and relaxations for the p-median problem, easily obtainable from the more general results presented in [19].

Let Z be the optimal value of the objective function of the uncapacitated location problem, $\bar{\bar{Z}}$ and $\underline{Z}$ upper and lower bounds for the problem, and Z_R a suitably chosen reference value such that

$$Z_R \geq \bar{\bar{Z}} \geq Z \geq \underline{Z} . \qquad (6.1)$$

Cornuejols et al. define

$$G = (\bar{\bar{Z}} - Z) / (Z_R - Z) \qquad (6.2)$$

for measuring the quality of heuristics (upper bounds for minimization problems), and

$$H = (Z - \underline{Z}) / (Z_R - \underline{Z}) \qquad (6.3)$$

for measuring the quality of lower bounds $\underline{Z}$.

Ideally, the reference Z_R should be equal to the maximum objective function value of the uncapacitated location problem P being studied, but, in any event, Z_R should be an upper bound on this maximum value that is sensitive to significant data changes such as the addition of a constant to every element of a row of the cost of matrix of problem P.

Consider, for example, a network $N = (X,A)$ of $\underline{n}$ vertices, with a weight v_j associated with every vertex $x_j \in X$. Let $D_{ij} = v_j \; d_{ij}$ and define the matrix $D = [D_{ij}]$. Then, for the p-median problem, Z_R is defined as the sum of the $(n-p)$ largest values

$$\underset{j}{\text{Max}} \, [D_{ij}] \quad , \tag{6.4}$$

over all rows $\underline{i}$ of D.

Given the above definition of Z_R, $(Z_R - Z)$ and $(Z_R - \underline{Z})$ may be thought of as the worst possible deviations that could be achieved by a given heuristic on lower bound, respectively. Then G measures the deviation for a particular heuristic relative to the worst possible deviation, and H the deviation for a particular lower bound relative to the worst possible deviation.

According to Cornuejols et al., a heuristic is "good" if

$$\underset{\text{All Problems P}}{\text{Lim}} \quad G < 1 \quad , \tag{6.5}$$

and "not good" if

$$\underset{\text{All Problems P}}{\text{Lim}} \quad G = 1 \quad . \tag{6.6}$$

Similarly, a relaxation is "good" if

$$\underset{\text{All Problems P}}{\text{Lim}} \quad H < 1 \quad , \tag{6.7}$$

and "not good" if

$$\underset{\text{All Problems P}}{\text{Lim}} \quad H = 1 \quad . \tag{6.8}$$

Cornuejols et al. specifically study heuristics for the uncapacitated location problem that correspond to:

(i) The vertex substitution heuristic described in 6.2.2;

(ii) The vertex addition ("greedy") heuristic, described in Section 6.4 for the p-median problem, and

(iii) The combination of the two above methods, also described in 6.4.

For the "greedy" heuristic they prove that

$$G_g = (Z_g-Z)/(Z_R-Z) \leqq [(p-1)/p]^p < 1/e \quad , \qquad (6.9)$$

where Z_g is the "greedy" heuristic solution and p the maximum allowed number of open facilities in the final solution. They also prove that, if Z_D is the optimum value of the strong linear programming relaxation of the uncapacitated location problem*,

$$H_D = (Z-Z_D)/(Z_R-Z_D) \leqq [(p-1)/p]^p < 1/e \quad . \qquad (6.10)$$

Finally, they show that the bounds of equations (6.9) and (6.10) are the best possible bounds, that is

$$\lim_{\text{All Problems P}} G_g = \lim_{\text{All Problems P}} H_D = 1/e \quad .$$

<u>The analysis applied to the p-median problem</u>

The above analysis, as well as the ones that follow for the two other heuristics, obviously apply to the p-median problem, in which exactly p facilities must be open in the final solution.

Consider now the integer programming (IP) formulation of the p-median problem, and its corresponding linear programming (LP) relaxation, given in Chapter 3. This LP relaxation corresponds to the strong LP relaxation of the uncapacitated location problem. If in Equation (6.10) H_D is replaced by H_{LP}, Z_D by Z_{LP} and Z by Z_{IP} it follows that

* For a definition of the strong and weak linear programming relaxations of the uncapacitated location problem, refer to [19], pp.1-4.

$$H_{LP} = (Z_{IP} - Z_{LP}) / (Z_R - Z_{LP}) < 1/e. \quad (6.11)$$

From equations (6.9) to (6.11) it is possible to conclude that the "greedy" heuristic is a "good" heuristic for the p-median problem, and that the LP relaxation of the IP formulation of the problem constitutes a "good" lower bound, in the sense defined by Cornuejols et al.

A worst case analysis is also carried out in [19] for both (i) the vertex substitution heuristic, and (ii) its combination with the "greedy" heuristic. Let Z_I be the solution to the vertex substitution heuristic and let

$$G_I = (Z_I - Z)/(Z_R - Z) \quad . \quad (6.12)$$

Cornuejols et al. prove that for all uncapacitated location problems

$$G_I \leqq (p-1)/(2p-1) \quad , \quad (6.13)$$

and that there exist problems P for which

$$G_I = (p-1)/(2p-1) \quad . \quad (6.14)$$

The p-median problem is among the problems P for which the equality may hold.

Now compare equations (6.13) and (6.14) with equation (6.9). It is possible to conclude that, in terms of worst case analysis, for every possible value of p for which the "greedy" heuristic can be used (p > 1), the vertex substitution heuristic does not perform as well as the "greedy" heuristic. This is a surprising result of some significance, especially because the "greedy" heuristic is by far the fastest to compute of the two.

The "greedy" and the vertex substitution heuristics combined

The idea of combining these two heuristic methods stems from the fact that, since the starting set of p locations for the vertex substitution heuristic is arbitrary, it might be advantageous to obtain this set of cardinality p by applying the "greedy" heuristic.

Let Z_{gI} be the value of the solution produced by the combination of the two heuristics, and let

$$G_{gI} = (Z_{gI} - Z) / (Z_R - Z) \quad . \qquad (6.15)$$

Cornuejols et al. prove that, for a well defined family of problems, the combination of the two heuristics fails to improve the solution obtained by the "greedy" heuristic. That is,

$$G_{gI} = G_g = [(p-1)/p]^p \qquad (6.16)$$

for a well defined family of uncapacitated location problems, in which case no interchange yields an improvement over the "greedy" heuristic.

6.3 λ-optimal substitution methods for the p-median problem

It has already been pointed out that the vertex substitution method is only one of a family of algorithms based on local optimization and the idea of λ-optimality. In the p-median problem a set S of p vertices is called λ-optimal ($\lambda \leq p$) if the replacement of any λ vertices in S by any other λ vertices of the set X of all vertices of the network N = (X,A) cannot produce a new set with transmission less than $\sigma(S)$. The replacement set of λ vertices chosen from X must obviously satisfy the condition that at least one of its elements belongs to the set (X-S). Within this context the answer produced by the vertex substitution algorithm of Teitz and Bart may be called 1-optimal.

From the definition of λ-optimality given above it is not difficult to see that in order to ensure that a given set is λ-optimal a total T_λ of

$$T_\lambda = \sum_{\lambda'=1}^{\lambda} \binom{p}{\lambda'} \binom{n-p}{\lambda'} \qquad (6.17)$$

potential substitutions (and hence calculations of transmissions σ) must be performed in each cycle of the algorithm. This number T_λ increases rapidly with λ, and hence practical algorithms cannot use values of λ much above 2 or 3.

Note that if S is the optimal p-median set $\bar{X}_p$ of a network, then S is p-optimal. It should also be pointed out that a λ-optimal substitution algorithm cannot be used when $p < \lambda$. A 2-optimal algorithm, for example, can only be used for $p \geq 2$.

A formal statement of λ-optimal substitution methods for the p-median problem is now given. This is a straightforward extension of the algorithm described in 6.2.2, and the same notation is used here. The algorithm is:

Step 1. Given a network $N = (X,A)$, select a set S of p vertices to form the initial approximation to the optimal p-median set $\bar{X}_p$. Call all sets of λ vertices $\{x_{j1},\ldots,x_{j\lambda}\}$, in which at least one element belongs to (X-S), "untried".

Step 2. Select some "untried" set of λ vertices $\{x_{j1},\ldots,x_{j\lambda}\}$ defined in Step 1, and for each of the $\binom{p}{\lambda}$ sets of λ vertices $\{x_{i1},\ldots,x_{i\lambda}\} \in S$ compute the "reduction" Δ_{ij} in the set transmission if $\{x_{j1},\ldots,x_{j\lambda}\}$ is substituted for $\{x_{i1},\ldots,x_{i\lambda}\}$, i.e. compute

$$\Delta_{ij} = \sigma(S) - \sigma(S \cup \{x_{j1},\ldots,x_{j\lambda}\} - \{x_{i1},\ldots,x_{i\lambda}\})$$

Step 3. Find $\Delta_{i_o j} = \underset{\{x_{i1},\ldots,x_{i\lambda}\} \in S}{\text{Max}} [\Delta_{ij}]$. Then:

(i) If $\Delta_{i_o j} \leqq 0$ call the set $\{x_{j1},\ldots,x_{j\lambda}\}$ "tried" and go to Step 2.

(ii) If $\Delta_{i_o j} > 0$ set $S \leftarrow S \cup \{x_{j1},\ldots,x_{j\lambda}\} - \{x_{i1},\ldots,x_{i\lambda}\}$, call $\{x_{j1},\ldots,x_{j\lambda}\}$ "tried" and go to Step 2.

Step 4. Repeat steps 2 and 3 until all sets of λ vertices $\{x_{j1},\ldots,x_{j\lambda}\}$ defined in Step 1 have been tried. This is a cycle of the algorithm. If during the last cycle no substitution of sets of λ vertices has been made in Step 3, go to Step 5. If some substitution has been made, call all sets of λ vertices $\{x_{j1},\ldots,x_{j\lambda}\}$ in which at least one element of the set belongs to (X-S) "untried" and return to Step 2.

Step 5. Stop. The current set S is the estimated p-median set $\bar{X}_p$.

The algorithm described above has been coded and tested for $\lambda = 2$. Computational results are given in Table 6.1, where these results are also compared with corresponding results obtained through the 1-optimal substitution method first introduced by Teitz and Bart.

Computational Results

Computational results for the $\lambda = 1$ and $\lambda = 2$ optimal substitution methods are shown in Table 6.1 for networks ranging from 10 to 30 vertices. Results for the 1-optimal method are then shown in Table 6.2 for the 33-city example of Karg and Thompson [57], and in Table 6.3 for 40 and 50 - vertex networks. For each of the networks the values shown in these tables range from $p = 1$ to $p = 10$.

The networks used to produce the results of Tables 6.1 to 6.3 are the same as those used in the previous three chapters. The data for the randomly generated networks used in these tables are given in the appendix.

In Table 6.1 the heuristic solution obtained by each of the two methods is shown, together with the number of cycles needed to reach the local optimum in each case. In addition, the random solution corresponding to the initial set S of p vertices is given in a separate column. For both methods this solution corresponds to using the first p vertices of each network to form the initial set S, with the vertices ranked by vertex index.

The optimal solution obtained by branch-and-bound is also shown in Table 6.1, so that the quality of the solutions produced by each of the two methods can be evaluated. Finally computing times are given in the last two columns of the table.

The 1-optimal substitution method has proved to be a satisfactory heuristic, and the corresponding percentage deviations from the optimal solution are summarized in Figure 6.1. In this figure the high frequency of 1-optimal solutions coinciding with the global optimum can be easily observed, and the average percentage error for heuristic solutions that are not optimal is shown to be low. In fact non-zero deviations from the optimal occurred more often for the larger networks ($n \geq 20$), although the maximum deviation (9.3%) occurred for the randomly generated 10-vertex network of Table 6.1 (for $p = 7$). This maximum deviation is well below the worst-case analysis result of Cornuejols et al. [See Equation (6.13)].

For the 2-optimal method, only in one case ($n = 25$, $p = 9$) the 2-optimal solution did not coincide with the global optimum (see Table 6.1).

6.4 The vertex addition heuristic and its use as a 'pre-processor' for λ-optimal substitution algorithms

The vertex addition heuristic described in the present section was initially developed as a procedure to provide upper bounds for

Table 6.1 - Computational Results for the λ=1 and λ=2 Optimal Substitution Methods

Problem Size			1-Optimal Substitution		2-Optimal Substitution			Time in Seconds*	
n	p	Random Initial Solution	Value of Solution	No. of Cycles	Value of Solution	No. of Cycles	Optimal Solution	1-Optimal Substitution	2-Optimal Substitution
10**	1	107	79	2	-	-	79	0.01	-
10	2	89	47	3	47	2	47	0.01	0.02
10	3	63	36	3	36	2	36	0.01	0.04
10	4	52	26	3	26	2	26	0.02	0.04
10	5	36	19	4	18	3	18	0.02	0.06
10	6	31	12	5	12	3	12	0.02	0.04
10	7	27	8	4	8	3	8	0.01	0.03
10	8	19	5	3	5	2	5	0.01	0.01
10	9	6	2	2	2	2	2	0.01	0.01
10	10	0	0	1	0	1	0	0.00	0.01
10	1	556	400	2	-	-	400	0.01	-
10	2	361	273	2	273	2	273	0.01	0.02
10	3	327	195	3	195	2	195	0.01	0.04
10	4	314	149	4	149	3	149	0.02	0.06
10	5	187	107	4	107	3	107	0.02	0.05
10	6	131	75	5	75	3	75	0.02	0.04
10	7	92	47	4	43	2	43	0.01	0.02
10	8	46	15	2	15	2	15	0.01	0.01
10	9	2	2	1	2	1	2	0.01	0.01
10	10	0	0	1	0	1	0	0.00	0.01

* CPU time, in CDC 7600 seconds

** Garfinkel et al. example [41, p.231]

Table 6.1 (cont'd.) - Computational Results for the λ=1 and λ=2 Optimal Substitution Methods

Problem Size			1-Optimal Substitution		2-Optimal Substitution			Time in Seconds*	
n	p	Random Initial Solution	Value of Solution	No. of Cycles	Value of Solution	No. of Cycles	Optimal Solution	1-Optimal Substitution	2-Optimal Substitution
15	1	846	809	2	-	-	809	0.02	-
15	2	573	412	2	412	2	412	0.03	0.09
15	3	533	294	3	294	2	294	0.05	0.18
15	4	310	215	4	215	3	215	0.08	0.39
15	5	265	150	4	150	3	150	0.08	0.48
15	6	208	113	5	113	3	113	0.09	0.52
15	7	148	93	4	93	3	93	0.07	0.50
15	8	87	77	2	74	3	74	0.03	0.44
15	9	67	58	2	57	3	57	0.03	0.36
15	10	50	41	2	41	2	41	0.02	0.17
20	1	1694	1159	2	-	-	1159	0.04	-
20	2	1227	724	3	724	2	724	0.10	0.29
20	3	732	523	4	518	2	518	0.15	0.66
20	4	511	414	3	414	2	414	0.13	1.04
20	5	476	338	4	338	2	338	0.19	1.38
20	6	392	259	5	259	3	259	0.25	2.55
20	7	356	241	4	227	5	227	0.20	4.62
20	8	332	209	5	199	4	199	0.25	3.82
20	9	275	181	4	175	3	175	0.19	2.90
20	10	239	157	3	151	4	151	0.13	3.60

* CPU time, in CDC 7600 seconds

Table 6.1 (cont'd.) - Computational Results for the λ=1 and λ=2 Optimal Substitution Methods

Problem Size			1-Optimal Substitution		2-Optimal Substitution			Time in Seconds*	
n	p	Random Initial Solution	Value of Solution	No. of Cycles	Value of Solution	No. of Cycles	Optimal Solution	1-Optimal Substitution	2-Optimal Substitution
25	1	1747	1352	2	-	-	1352	0.08	-
25	2	1551	956	3	956	2	956	0.17	0.73
25	3	1161	722	5	722	3	722	0.38	2.58
25	4	790	556	4	556	3	556	0.37	4.37
25	5	763	468	4	468	3	468	0.42	6.11
25	6	706	387	4	387	3	387	0.45	7.77
25	7	533	341	5	341	3	341	0.59	9.22
25	8	415	305	7	298	4	298	0.84	13.72
25	9	393	278	6	267	5	266	0.72	17.92
25	10	354	253	5	235	6	235	0.59	21.92
30	1	2400	1432	2	-	-	1432	0.11	-
30	2	1610	936	3	936	2	936	0.30	1.54
30	3	1168	796	5	777	3	777	0.69	5.75
30	4	1040	610	6	610	4	610	1.01	13.06
30	5	883	530	7	516	4	516	1.35	19.16
30	6	688	438	9	438	5	438	1.92	31.63
30	7	663	386	7	386	4	386	1.59	30.94
30	8	641	337	9	-	-	337	2.14	-
30	9	455	294	8	-	-	294	1.91	-
30	10	438	265	7	-	-	265	1.67	-

* CPU time, in CDC 7600 seconds.

Table 6.2 - Computational Results for the 1-Optimal Substitution Method

Problem Size		Random Initial Solution	Value of Heuristic Solution	No. of Cycles	Optimal Solution	Time in* Seconds
n	p					
33**	1	37993	32072	2	32072	0.14
33	2	35800	17474	2	17474	0.26
33	3	35145	14627	4	14627	0.70
33	4	34806	12625	5	12363	1.09
33	5	34453	10727	6	10398	1.52
33	6	34200	8832	11	8832	3.10
33	7	32855	8261	8	8119	2.42
33	8	31898	7561	8	7472	2.56
33	9	31651	6848	9	6848	2.98
33	10	31236	6295	9	6267	3.05

* CPU time, in CDC 7600 seconds

** Karg and Thompson 33 City Example [57, p.244]

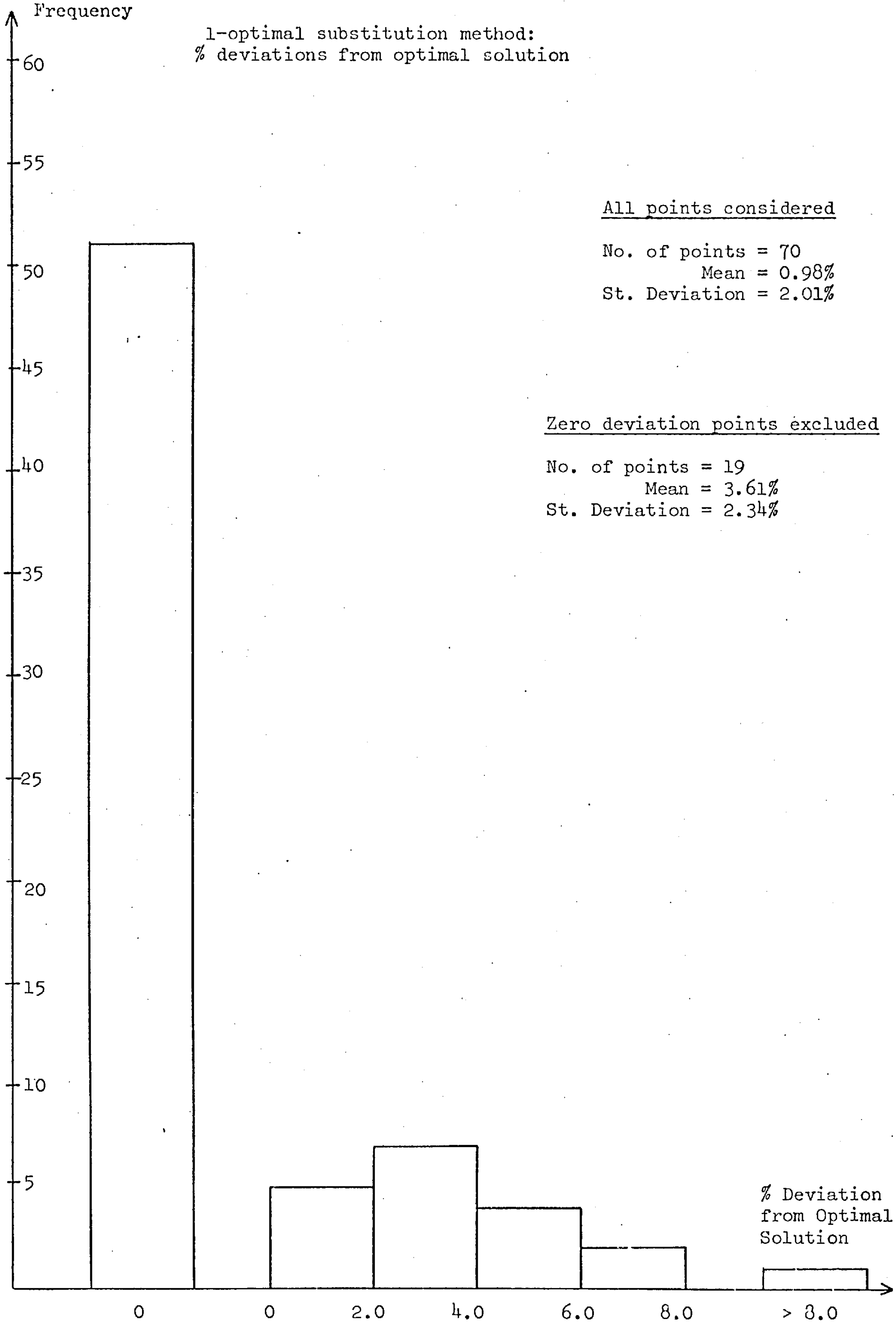
Table 6.3 - Computational Results for the 1-Optimal Substitution Method

Problem Size					
n	p	Random Initial Solution	Value of Heuristic Solution	No. of Cycles	Time in* Seconds
40	1	84954	80634	2	0.27
40	2	81794	45862	3	0.74
40	3	76951	35946	4	1.35
40	4	74632	26899	6	2.49
40	5	73828	22396	6	2.94
40	6	71504	18775	7	3.87
40	7	68954	17426	9	5.49
40	8	68525	16251	10	6.55
40	9	67709	14980	10	6.99
40	10	62957	13443	10	7.26
50	1	292916	128548	2	0.51
50	2	273599	72168	4	1.90
50	3	231943	52708	6	4.03
50	4	205945	42228	4	3.42
50	5	179107	35677	7	7.13
50	6	151690	31853	6	7.11
50	7	141360	28300	5	6.47
50	8	122640	25624	9	12.73
50	9	100287	24580	8	12.14
50	10	86463	22796	10	16.27

* CPU time, in CDC 7600 seconds

Figure 6.1

1-optimal substitution method:
% deviations from optimal solution



All points considered

No. of points = 70
Mean = 0.98%
St. Deviation = 2.01%

Zero deviation points excluded

No. of points = 19
Mean = 3.61%
St. Deviation = 2.34%

the branch-and-bound algorithm of Chapter 5. A generalization of the method for uncapacitated location problems is described in [19] where it is referred to as the "greedy" heuristic. Variations of this heuristic also appear elsewhere in the literature [55,92].

Even though the vertex addition heuristic does not perform badly on its own, especially for the larger values of p, the main interest in the present section is in its use as a 'pre-processor' to λ-optimal substitution methods. The idea behind the 'combined approach', described and tested in the remaining of this section, is that since λ-optimal substitution algorithms must start from a set S of p vertices, some advantage might be gained by starting from a "good" set of p vertices.

The main advantage gained from the combined approach was a substantial reduction in computing times. Although it could be claimed from the available data that for $\lambda = 1$ some precision was gained when the vertex addition heuristic was used as a 'pre-processor', the justification for using the combined approach lies in the substantial drop observed in the number of cyles needed to find the local optimum in λ-optimal substitution algorithms. The corresponding drop in computing times is especially remarkable for the larger networks ($n \geq 20$), as shown in Tables 6.4 to 6.6.

Used on its own the vertex addition heuristic starts from an available solution to the (p-1)-median problem and adds to this solution the vertex that produces the maximum possible decrease in the objective function as the number of medians is increased from (p-1) to p. The surprisingly good results obtained through this simple procedure appear to derive from the relative 'stability' of the solutions* of the problem as p is increased, and from the already

* It has been observed that in the majority of cases most of the vertices present in the optimal (p-1)-median set are also present in the optimal p-median set.

mentioned existence of many optimal or near-optimal solutions to the p-median problem.

The combined approach is initialized with the 1-median solution. It then proceeds in a stepwise fashion, with the appropriate λ-optimal substitution method being applied to the set of cardinality p generated by the vertex addition heuristic. The procedure terminates after solutions are produced for the desired range of values of p. The stepwise nature of the combined approach explains why, for any given problem, the initial solutions provided by the vertex addition heuristic do not necessarily coincide for different values of λ (see Table 6.4).

Computational experience

The computational experience with the combined approach is shown in Table 6.4 for λ = 1 and λ = 2, and in Tables 6.5 and 6.6 for λ = 1 only.

In Table 6.4 the initial solution provided by the vertex addition heuristic, together with the final heuristic solution and the number of cycles needed to reach the local optimum are shown for both λ = 1 and λ = 2. The optimal solution obtained by branch-and-bound is given in a separate column, and finally computing times for each of the four different heuristic methods analysed in the present chapter are shown in the last columns of the table.

The computing times corresponding to the "pure" λ-optimal substitution methods are repeated in Table 6.4 in order to facilitate the comparison of the four methods. The sums of computing times for the several values of p within each network and heuristic method of Tables 6.4 to 6.6 are also provided.

The combined approach has proved to be a good heuristic for λ = 1, and the corresponding percentage deviations from the optimal solution are shown in Figure 6.2. If the results shown in this figure are

Table 6.4 - Computational Results for the Combined Approach ($\lambda=1$ and $\lambda=2$)

Problem Size		Combined Approach, $\lambda=1$			Combined Approach, $\lambda=2$				Time in Seconds*			
n	p	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Optimal Solution	1-Opt, Random	1-Opt, V. Add.	2-Opt, Random	2-Opt, V. Add.
10**	1	-	79	2	-	-	-	79	0.01	0.01	-	-
10	2	47	47	1	47	47	1	47	0.01	0.01	0.02	0.02
10	3	36	36	1	36	36	1	36	0.01	0.01	0.04	0.02
10	4	27	26	2	27	26	2	26	0.02	0.01	0.04	0.04
10	5	20	18	2	20	18	2	18	0.02	0.01	0.06	0.04
10	6	12	12	1	12	12	1	12	0.02	0.01	0.04	0.02
10	7	8	8	1	8	8	1	8	0.01	0.01	0.03	0.01
10	8	5	5	1	5	5	1	5	0.01	0.01	0.01	0.01
10	9	2	2	1	2	2	1	2	0.01	0.01	0.01	0.01
10	10	0	0	1	0	0	1	0	0.00	0.01	0.01	0.01
								ΣTime†=	0.12	0.10	0.26	0.18
10	1	-	400	2	-	-	-	400	0.01	0.01	-	-
10	2	276	273	2	276	273	2	273	0.01	0.01	0.02	0.03
10	3	195	195	1	195	195	1	195	0.01	0.01	0.04	0.02
10	4	149	149	1	149	149	1	149	0.02	0.01	0.06	0.03
10	5	107	107	1	107	107	1	107	0.02	0.01	0.05	0.02
10	6	75	75	1	75	75	1	75	0.02	0.01	0.04	0.02
10	7	43	43	1	43	43	1	43	0.01	0.01	0.02	0.01
10	8	15	15	1	15	15	1	15	0.01	0.01	0.01	0.01
10	9	2	2	1	2	2	1	2	0.01	0.01	0.01	0.01
10	10	0	0	1	0	0	1	0	0.00	0.01	0.01	0.01
								ΣTime†=	0.12	0.10	0.26	0.16

* CPU time, in CDC 7600 seconds

** Garfinkel et al. Example [41, p.231]

† Σ Time is the sum of computing times for the several values of p for which a solution is available.

Table 6.4 (cont'd.) - Computational Results for the Combined Approach (λ=1 and λ=2)

Problem Size		Combined Approach, λ=1			Combined Approach, λ=2				Time in Seconds*			
n	p	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Optimal Solution	1-Opt, Random	1-Opt, V. Add.	2-Opt, Random	2-Opt, V.Add.
15	1	-	809	2	-	-	-	809	0.02	0.02	-	-
15	2	484	412	3	484	412	2	412	0.03	0.05	0.09	0.10
15	3	294	294	1	294	294	1	294	0.05	0.03	0.18	0.10
15	4	215	215	1	215	215	1	215	0.08	0.03	0.39	0.14
15	5	170	150	2	170	150	2	150	0.08	0.05	0.48	0.33
15	6	113	113	1	113	113	1	113	0.09	0.03	0.52	0.18
15	7	93	93	1	93	93	1	93	0.07	0.03	0.50	0.17
15	8	74	74	1	74	74	1	74	0.03	0.03	0.44	0.15
15	9	57	57	1	57	57	1	57	0.03	0.02	0.36	0.13
15	10	41	41	1	41	41	1	41	0.02	0.02	0.17	0.09
								ΣTime† =	0.50	0.31	3.13	1.39
20	1	-	1159	2	-	-	-	1159	0.04	0.04	-	-
20	2	847	724	2	847	724	2	724	0.10	0.09	0.29	0.31
20	3	552	523	3	552	518	3	518	0.15	0.13	0.66	1.00
20	4	438	431	3	433	414	2	414	0.13	0.15	1.04	1.04
20	5	340	340	1	353	338	2	338	0.19	0.07	1.38	1.36
20	6	281	259	3	277	259	2	259	0.25	0.17	2.55	1.65
20	7	230	230	1	230	227	2	227	0.20	0.08	4.62	1.83
20	8	202	202	1	199	199	1	199	0.25	0.07	3.82	0.99
20	9	175	175	1	175	175	1	175	0.19	0.07	2.90	0.98
20	10	151	151	1	154	151	2	151	0.13	0.07	3.60	1.81
								ΣTime† =	1.63	0.94	20.86	10.97

* CPU time, in CDC 7600 seconds

† Σ Time is the sum of computing times for the several values of p for which a solution is available.

Table 6.4 (cont'd.) - Computational Results for the Combined Approach (λ=1 and λ=2)

Problem Size		Combined Approach, λ=1			Combined Approach, λ=2				Time in Seconds[*]			
n	p	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Vertex Add. Initial Solution	Heuristic Solution	No.of Cycles	Optimal Solution	1-Opt, Random	1-Opt, V. Add.	2-Opt, Random	2-Opt, V.Add.
25	1	-	1352	2	-	-	-	1352	0.08	0.08	-	-
25	2	1027	1027	1	1027	956	2	956	0.17	0.10	0.73	0.75
25	3	807	722	4	722	722	1	722	0.38	0.33	2.58	0.90
25	4	556	556	1	556	556	1	556	0.37	0.12	4.37	1.49
25	5	468	468	1	468	468	1	468	0.42	0.13	6.11	2.10
25	6	387	387	1	387	387	1	387	0.45	0.14	7.77	2.65
25	7	341	341	1	341	341	1	341	0.59	0.14	9.22	3.12
25	8	298	298	1	298	298	1	298	0.84	0.14	13.72	3.48
25	9	269	269	1	269	267	2	266	0.72	0.15	17.92	7.37
25	10	244	244	1	242	242	1	235	0.59	0.15	21.92	3.79
								ΣTime[†]=	4.61	1.48	84.34	25.65
30	1	-	1432	2	-	-	-	1432	0.11	0.11	-	-
30	2	1029	936	2	1029	936	2	936	0.30	0.26	1.54	1.58
30	3	796	796	1	796	777	2	777	0.69	0.19	5.75	3.85
30	4	676	610	3	660	610	3	610	1.01	0.57	13.06	9.84
30	5	533	533	1	533	516	3	516	1.35	0.25	19.16	14.34
30	6	467	438	4	443	438	2	438	1.92	0.95	31.63	12.68
30	7	389	389	1	389	386	2	386	1.59	0.28	30.94	15.31
30	8	341	341	1	337	337	1	337	2.14	0.28	-	8.86
30	9	294	294	1	307	294	3	294	1.91	0.29	-	29.36
30	10	266	266	1	266	265	2	265	1.67	0.29	-	21.17

[*] CPU time, in CDC 7600 seconds.

[†] Σ Time is the sum of computing times for the several values of p for which a solution is available.

Table 6.5 - Computational Results for the Combined Approach (λ=1)

Problem Size						Time in Seconds[*]	
n	p	Vertex Addition Initial Solution	Heuristic Solution	No. of Cycles	Optimal Solution	1-Optimal Random	1-Optimal Vertex Add.
33[**]	1	-	32072	2	32072	0.14	0.14
33	2	19196	17474	2	17474	0.26	0.31
33	3	14962	14627	3	14627	0.70	0.57
33	4	12509	12363	3	12363	1.09	0.70
33	5	10797	10797	1	10398	1.52	0.31
33	6	9287	8832	3	8832	3.10	0.89
33	7	8213	8119	2	8119	2.42	0.65
33	8	7538	7538	1	7472	2.56	0.37
33	9	7055	7055	1	6848	2.98	0.38
33	10	6592	6408	2	6267	3.05	0.72
					Σ Time[†] =	17.82	5.04

[*] CPU time, in CDC 7600 seconds

[**] Karg and Thompson 33 City Example [57, p.244]

[†] Σ Time is the sum of computing times for the several values of p for which a solution is available

Table 6.6 - Computational Results for the Combined Approach (λ=1)

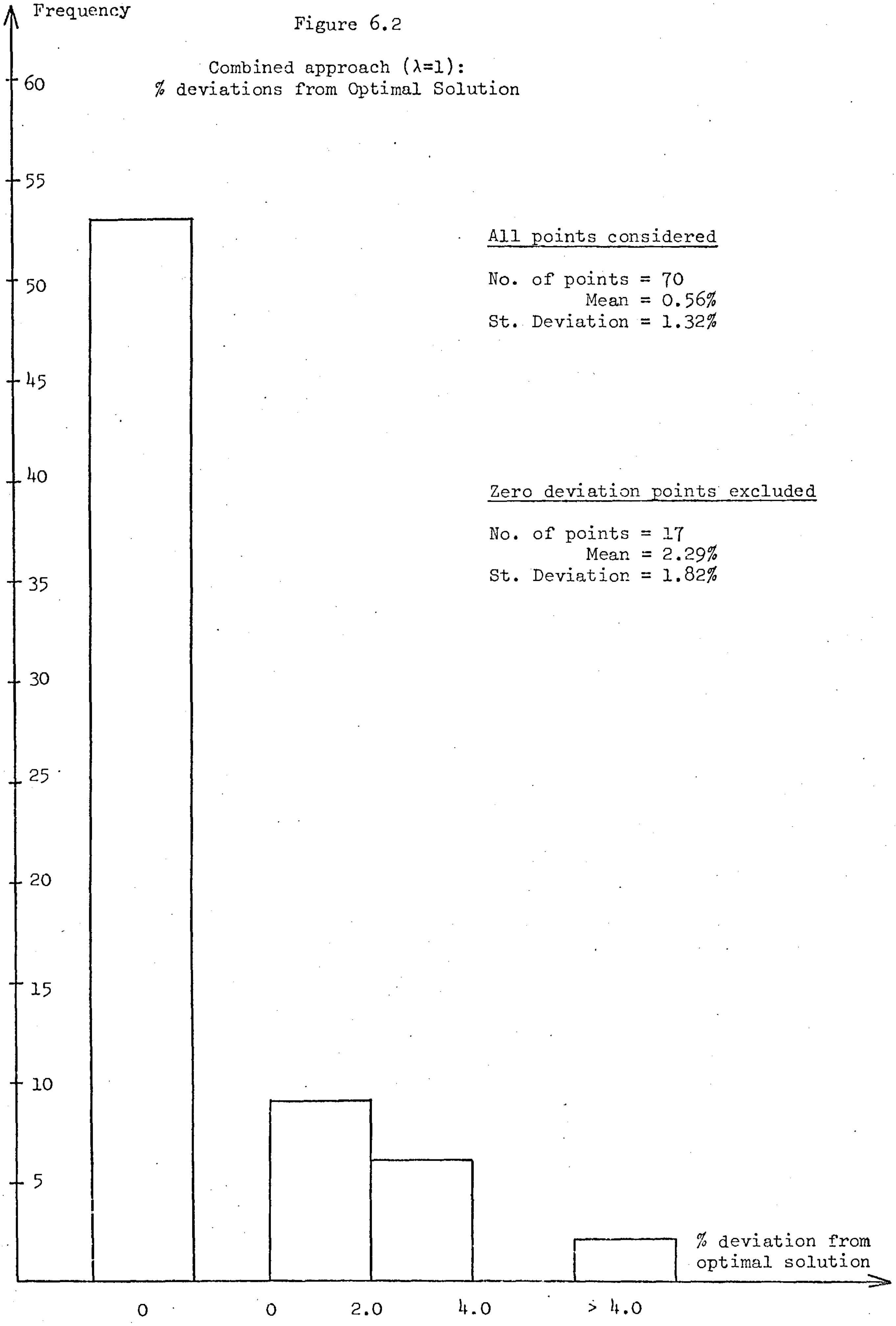
Problem Size		1-Optimal Substitution		Combined Approach, λ = 1				Time in Seconds*	
n	p	Heuristic Solution	No. of Cycles	Vertex Add. Initial Solution	Heuristic Solution	No. of Cycles	Best Avail. Solution	1-Optimal Random	1-Optimal Vertex Add.
40	1	80634	2	-	80634	2	80634	0.27	0.27
40	2	45862	3	45862	45862	1	45862	0.74	0.35
40	3	35946	4	35946	35946	1	35946	1.35	0.43
40	4	26899	6	28897	26899	3	26899	2.49	1.34
40	5	22396	6	23278	23278	1	22396	2.94	0.58
40	6	18775	7	20594	18775	4	18775	3.87	2.31
40	7	17426	9	17426	17426	1	17426	5.49	0.69
40	8	16251	10	16155	16155	1	16155	6.55	0.74
40	9	14980	10	15095	14539	2	14539	6.99	1.47
40	10	13443	10	13484	13436	2	13436	7.26	1.52
							ΣTime† =	37.95	9.70
50	1	128548	2	-	128548	2	128548	0.51	0.51
50	2	72168	4	83910	72168	4	72168	1.90	2.07
50	3	52708	6	54959	52708	3	52708	4.03	2.17
50	4	42228	4	44274	42228	3	42228	3.42	2.69
50	5	35677	7	36710	35677	3	35677	7.13	3.17
50	6	31853	6	32406	31853	2	31853	7.11	2.46
50	7	28300	5	29177	29177	1	28300	6.47	1.43
50	8	25624	9	26569	25624	4	25624	12.73	5.79
50	9	24580	8	24129	24129	1	24129	12.14	1.65
50	10	22796	10	22668	22668	1	22668	16.27	1.74
							ΣTime† =	71.71	23.68

* CPU time, in CDC 7600 seconds

† ΣTime is the sum of computing times for the several values of p for which a solution is available.

Figure 6.2

Combined approach (λ=1):
% deviations from Optimal Solution



All points considered

No. of points = 70
Mean = 0.56%
St. Deviation = 1.32%

Zero deviation points excluded

No. of points = 17
Mean = 2.29%
St. Deviation = 1.82%

compared with those of Figure 6.1, it is possible to say that for $\lambda = 1$ the combined approach is a better heuristic than the "pure" 1-optimal substitution method. Similarly to Figure 6.1, in Figure 6.2 the non-zero deviations from the optimal occurred exclusively for the larger networks ($n \geq 20$). The maximum deviation from the optimal was 7.43% (for $n = 25$, $p = 2$), again well below the maximum possible deviation of 1/e derived by Cornuejols et al. [See Equations (6.9) and (6.16)].

The combined approach for $\lambda = 2$ can be said to be as precise as its "pure" 2-optimal counterpart: of all points shown in Table 6.4 on only two occasions (for $n = 25$, $p = 9$ and $p = 10$) did the heuristic solution fail to coincide with the corresponding global optimum.

Finally, computing times were related to both <u>n</u> and <u>p</u> for the λ-optimal substitution methods described in the present chapter. When these times were plotted against <u>n</u> and <u>p</u>, it became evident that the equation describing the total time needed to reach a local optimum in these algorithms is of the form

$$CT_\lambda = K\, n^{k_1}\, p^{k_2} \quad , \tag{6.18}$$

where CT_λ is the total computing time, and K, k_1 and k_2 are constants. In the above formula <u>n</u> is of more importance than <u>p</u> in determining the final value of CT_λ.

The data of Table 6.4 were used to find the values of K, k_1 and k_2 for $\lambda = 1$ and $\lambda = 2$. The resulting equations are

$$CT_\lambda = 1.87 \times 10^{-6}\ n^{3.33}\ p^{0.25} \tag{6.19}$$

for $\lambda = 1$, and

$$CT_\lambda = 5.56 \times 10^{-9}\ n^{5.56}\ p^{1.13} \tag{6.20}$$

for $\lambda = 2$.

Figure 6.3

CDC 7600 seconds vs. n for λ=1

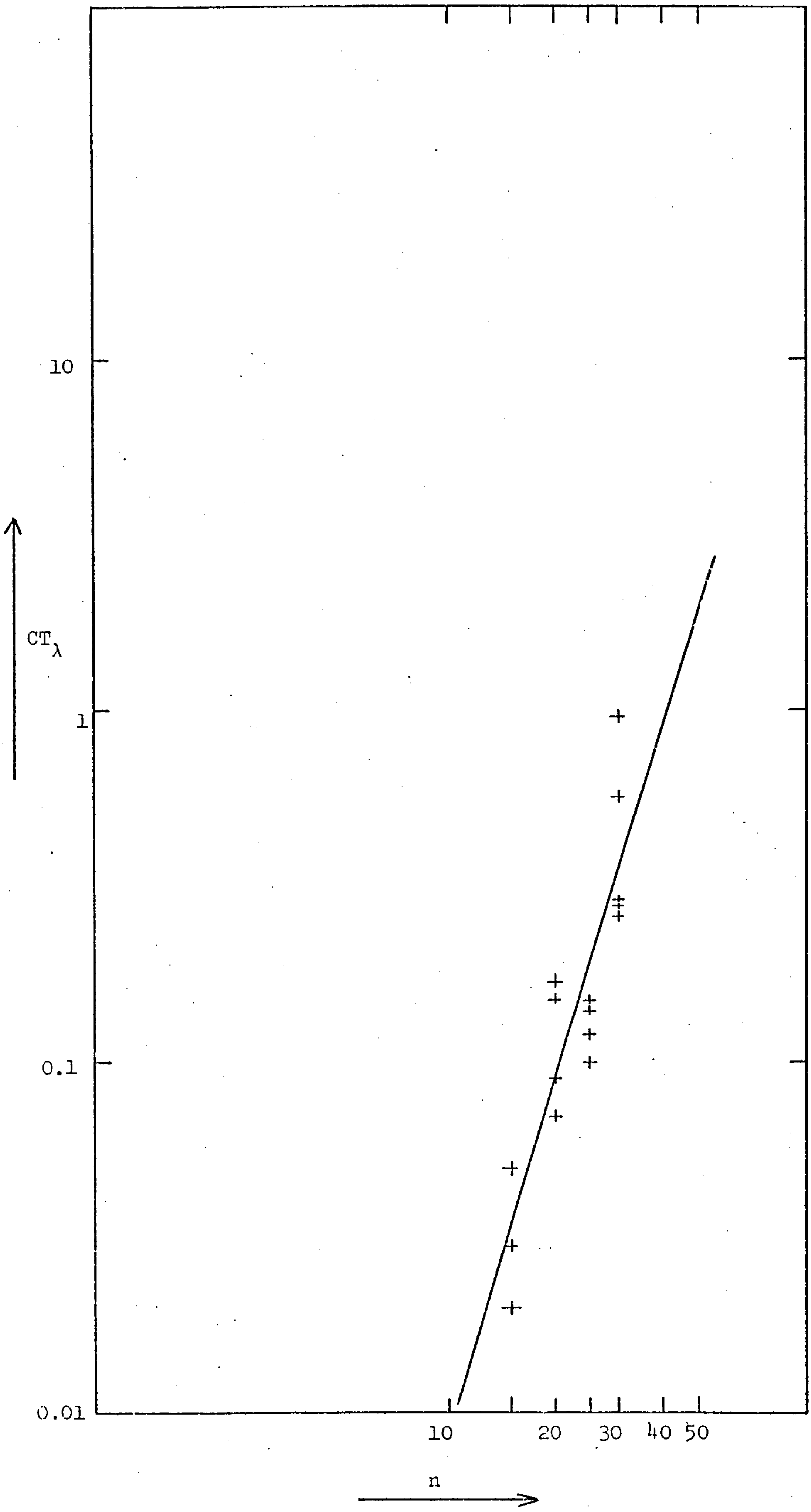
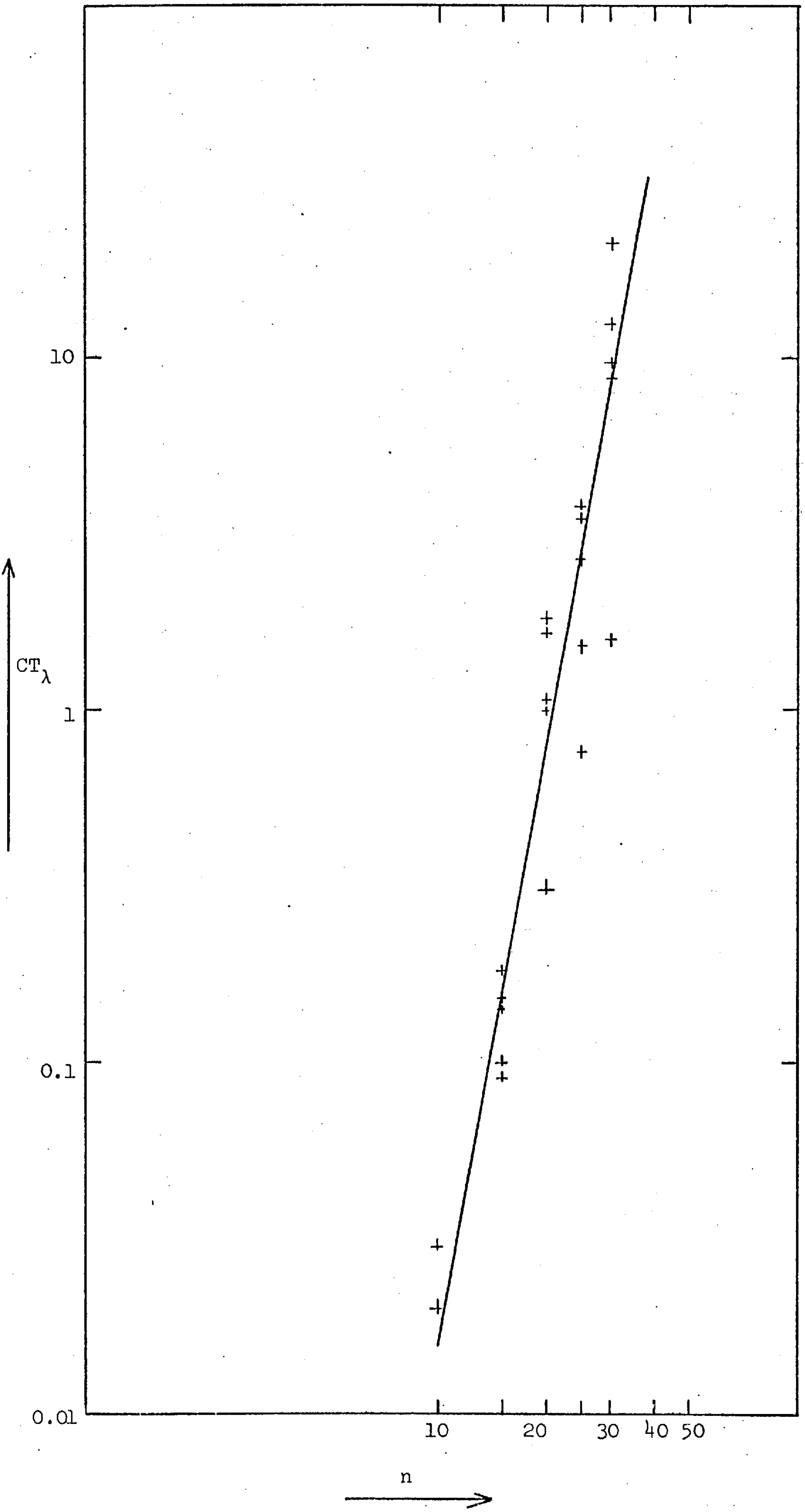


Figure 6.4

CDC 7600 seconds vs. n for λ=2



CT_λ is plotted against $\underline{n}$ for $p = 8$ in Figures 6.3 and 6.4 for $\lambda = 1$ and $\lambda = 2$ respectively. The correlation coefficients obtained when a least squares line was fitted through the points in each of these figures were 0.92 for $\lambda = 1$ and 0.95 for $\lambda = 2$.

6.5 Conclusions

Four interrelated heuristic methods for the p-median problem were studied in the present chapter. The vertex substitution method of Teitz and Bart was extended, and λ-optimal substitution methods, based on local optimization and the idea of λ-optimality, were described. Computational experience was reported for the particular cases of $\lambda = 1$ and $\lambda = 2$.

A simple vertex addition heuristic was introduced, and used as a 'pre-processor' to λ-optimal substitution algorithms. Computational results were again reported for $\lambda = 1$ and $\lambda = 2$. The four resulting heuristics were then evaluated on the basis of the quality of the solutions produced and the computing times required to reach these solutions.

The precision of the heuristic solutions naturally increases with λ. This increased precision, however, is obtained at substantially higher computing costs. From the data available for $\lambda = 2$ it can be safely concluded that it is not practical to use λ-optimal substitution methods for $\lambda > 2$.

The introduction of the vertex addition heuristic as a 'pre-processor' to λ-optimal substitution algorithms substantially reduced computing times. This is so because the number of iterations required to reach the local optimum is sharply reduced when the combined approach is used, especially for $\lambda = 1$. This is accomplished without loss of quality in the solutions produced by the combined approach.

From a cost-effectiveness point of view, the combination of the vertex addition heuristic with the 1-optimal substitution method appears to be the best of the four methods studied. It produces solutions that are on average of better quality than the solutions produced by its "pure" 1-optimal substitution counterpart, and not much worse than the solutions produced by the 2-optimal methods. The corresponding computing times are the lowest of the four methods studied.

CHAPTER SEVEN

CONCLUSIONS

7.1 General Summary

This thesis studied the p-median problem, concentrating on exact solution procedures for the problem. New methods of solution were developed in the course of the work. These include the development of two lower bounds, and the use of one of them in a direct tree search algorithm especially designed for the problem. The resulting procedure represents a substantial advancement in the area of exact solution methods for the p-median problem.

Due to the fact that in the vast majority of cases the LP relaxation of the integer programming formulation of the problem produces integer solutions that are optimal solutions to the p-median problem itself, two formulations of this relaxation were initially studied. The general formulation produces very large linear programmes, and is therefore unsuitable for use in large-scale networks. The decomposition formulation often does not converge because of its very degenerate nature. The problems with convergence become particularly serious as the size of the network increases, and for values of p small in relation to n.

Branch-and-bound algorithms available in the literature suffer from a lack of strong lower bounds and for this reason are not very efficient in solving the p-median problem. In this thesis two new lower bounds were developed, namely the graph-theoretical bound and the dual bound. The graph-theoretical bound is not very good for small values of p, but improves considerably as the value of p increases. The dual bound has proved to be a very good lower bound. When tested in 80 different problems, its average deviation from the best available solution was only 2.57% (See Figure 4.2).

The dual bound was embedded into a direct tree search algorithm especially designed for the p-median problem. This algorithm also used a weaker bound and cascaded through both bounds in order to reduce computing times. An upper bound obtained from heuristics contributed to further reduce the size of the tree search. The use of LP decomposition to solve the subproblems was also investigated.

The branch-and-bound algorithm produces optimal solutions for networks of up to 30 vertices in less than 2 minutes of computer time in a CDC 7600 computer, for every possible value of p ($1 \leqq p \leqq n$). Besides guaranteeing optimal solutions for larger problems than any other existing exact procedure, the algorithm is both faster (in terms of time) and more efficient (in terms of number of nodes) than other branch-and-bound algorithms available in the literature for the p-median problem.

Finally, heuristic methods were investigated and tested in a number of problems. The vertex substitution method of Teitz and Bart was extended into a family of heuristics, the λ-optimal substitution heuristic methods. Then a simple vertex addition heuristic was introduced, and used as a 'pre-processor' to λ-optimal substitution methods, thus considerably reducing computing times. The particular cases of $\lambda = 1$ and $\lambda = 2$ were coded, and computational experience reported on the resulting heuristic methods.

From the data available on heuristic methods it is safe to conclude that 2-optimal substitution methods are too expensive for networks with more than 20 vertices, and that, from a cost-effectiveness point of view, the combination of the vertex addition heuristic with the 1-optimal substitution method is the best of the four methods studied. It produces solutions that are on average of better quality than the solutions produced by its "pure" 1-optimal counterpart, and not much worse than the solutions produced by 2-optimal methods. The corresponding computing times are the lowest of the four methods studied.

7.2 Possible areas for further research

One main area for further research on the p-median problem arises naturally from the work done in this thesis, and is related to solving the convergence problems of the LP decomposition algorithm. Progress in this area would allow guaranteed optimal solutions to be found for large-scale networks within a reasonable amount of computer time.

Regarding the convergence problems of the LP decomposition algorithm, it is worth noting the approach suggested by Beale, and reported in Section 3.4.3. If the difficulties arising from the lack of convergence of this algorithm can be solved, then LP decomposition, and its use to solve subproblems in branch-and-bound algorithms, can be used to provide optimal solutions to the p-median problem for large-scale networks.

Beyond the pure p-median problem, there remain the several variations of the generalized p-median problem mentioned in 2.4.2. It was then stated that the main difficulty in solving minisum network location problems rests with the pure p-median problem studied in this thesis. Any progress in solving the pure p-median problem necessarily means, therefore, progress in solving generalized p-median problems.

APPENDIX

DATA FOR THE TEST PROBLEMS USED IN THE THESIS

Except for

(i) The test cases provided by A.W. Neebe (see Tables 3.1 to 3.3),

(ii) the 10-vertex networks of Garfinkel et al. [41, p.231] and Revelle and Swain [90, p.38], and

(iii) the 33 city example of Karg and Thompson [57, p.244],

all other networks used as test problems in Chapters 3 to 6 of this thesis were randomly generated.

The data for the randomly generated networks were obtained as follows. The Cartesian coordinates of the vertices were generated randomly from a discrete uniform distribution over two different intervals: (0,100) for networks of up to 30 vertices, and (0,1000) for the 40 and 50-vertex networks. The vertices thus generated were connected by choosing links at random until a tree was formed. Finally additional links were added to this basic connected network. The number of additional links used in each network, and the pair of vertices each of these links were to connect, were also randomly generated.

The length of the links in each of the randomly generated networks was calculated using the Euclidean distance formula. All randomly generated networks are nondirected, nonweighted graphs.

The data for the test cases provided by A.W. Neebe are given in matrix format. This is followed by the data describing the randomly generated networks. In the latter set of data each pair of vertices connected by a link is listed alongside the corresponding link length. This is the format of the input data for all computer programmes listed in [39]. For each randomly generated network the average vertex degree

$$\bar{d} = (\sum_{i=1}^{n} d_i) / n$$

is also given.

1. NEEBE'S TEST PROBLEMS

(A) 5-Vertex Network

TO

FROM	x_1	x_2	x_3	x_4	x_5
x_1	0	1	1	2	2
x_2	1	0	1	1	2
x_3	1	1	0	2	1
x_4	2	1	2	0	1
x_5	2	2	1	1	0

(B) 6-Vertex Network

TO

FROM	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	5	4	8	7	12
x_2	5	0	3	3	6	7
x_3	4	3	0	6	3	8
x_4	8	3	6	0	3	4
x_5	7	6	3	3	0	5
x_6	12	7	8	4	5	0

(C) 9-Vertex Network

						TO				
		x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
FROM	x_1	0	14	2	2	3	14	13	14	1
	x_2	14	0	14	14	11	2	1	2	13
	x_3	2	14	0	2	3	14	13	14	1
	x_4	2	14	2	0	3	14	13	14	1
	x_5	3	11	3	3	0	11	10	11	2
	x_6	14	2	14	14	11	0	1	2	13
	x_7	13	1	13	13	10	1	0	1	12
	x_8	14	2	14	14	11	2	1	0	13
	x_9	1	13	1	1	2	13	12	13	0

2. RANDOMLY GENERATED NETWORKS

(A) 10-Vertex Network ($\bar{d} = 3.4$)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	3	32
2	1	4	13
3	1	6	28
4	2	3	81
5	2	7	35
6	2	10	42
7	3	5	43
8	3	10	44
9	4	5	43
10	5	6	54
11	5	7	34
12	5	9	44
13	5	10	2
14	6	8	51
15	7	8	46
16	7	9	55
17	7	10	32

(B) 15-Vertex Network ($\bar{d}$ = 3.2)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	3	25
2	1	9	10
3	2	3	20
4	2	5	25
5	2	7	30
6	2	15	32
7	3	13	4
8	4	6	19
9	4	10	29
10	4	11	16
11	5	7	30
12	5	15	12
13	6	9	58
14	6	11	5
15	6	14	32
16	7	9	67
17	7	12	19
18	8	11	34
19	8	12	26
20	8	14	10
21	9	10	17
22	9	12	76
23	9	13	36
24	11	14	32

(C) 20-Vertex Network ($\bar{d}$ = 2.9)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	5	16
2	1	12	11
3	2	7	33
4	2	9	21
5	2	11	52
6	3	8	38
7	3	10	29
8	3	18	18
9	3	19	30
10	4	8	24
11	4	17	13
12	5	6	45
13	5	15	35
14	6	7	36
15	6	14	38
16	6	16	51
17	6	19	28
18	7	18	11
19	9	16	8
20	10	17	30
21	10	19	21
22	10	20	45
23	11	12	18
24	11	15	15
25	13	14	21
26	13	20	20
27	14	19	64
28	15	16	12
29	17	20	21

(D) 25-Vertex Network ($\bar{d}$ = 2.8)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	9	39
2	1	10	42
3	1	19	20
4	1	23	41
5	2	5	18
6	2	15	37
7	3	15	9
8	3	17	26
9	3	24	18
10	4	6	26
11	4	9	22
12	4	21	15
13	5	15	44
14	5	19	11
15	6	9	38
16	6	25	11
17	7	14	22
18	7	22	21
19	8	9	36
20	8	10	23
21	8	11	25
22	8	14	34
23	8	17	21
24	8	21	45
25	10	20	13
26	11	16	24
27	12	13	19
28	12	18	17
29	13	18	25
30	13	24	16
31	14	21	14
32	16	22	14
33	17	18	12
34	17	20	8
35	23	25	25

(E) 30-Vertex Network ($\bar{d}$ = 2.7)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	10	24
2	1	23	40
3	1	25	22
4	2	17	8
5	2	20	58
6	3	5	22
7	3	8	17
8	4	7	13
9	4	14	17
10	5	10	17
11	5	21	21
12	5	26	16
13	6	11	13
14	6	14	36
15	6	22	20
16	7	8	11
17	8	15	9
18	9	13	25
19	9	24	16
20	9	29	7
21	11	14	40
22	11	18	11
23	12	16	22
24	12	18	10
25	13	20	28
26	13	23	13
27	13	27	34
28	15	26	18
29	16	17	13
30	16	19	20
31	19	20	23
32	19	29	11
33	21	25	8
34	21	28	9
35	22	24	14
36	22	30	13
37	23	28	15
38	24	30	14
39	26	27	12
40	26	28	21
41	27	30	9

(F) 40-Vertex Network ($\bar{d} = 3.0$)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	2	395
2	1	10	216
3	1	40	552
4	2	3	799
5	2	39	458
6	3	4	395
7	3	38	493
8	4	5	402
9	4	37	136
10	5	6	980
11	5	36	956
12	6	7	651
13	6	35	956
14	7	8	289
15	7	34	445
16	8	9	408
17	8	33	742
18	9	10	262
19	9	32	226
20	10	11	437
21	10	31	441
22	11	12	667
23	11	30	908
24	12	13	549
25	12	29	531
26	13	14	262
27	13	28	266
28	14	15	271
29	14	27	330
30	15	16	720
31	15	26	849
32	16	17	391
33	16	25	402
34	17	18	439
35	17	24	718
36	18	19	537
37	18	23	611
38	19	20	705
39	19	22	741
40	20	21	307
41	20	22	500
42	21	22	319
43	22	23	758
44	23	24	647
45	24	25	567
46	25	26	719
47	26	27	484
48	27	28	330
49	28	29	278
50	29	30	592

(F) 40-Vertex Network (cont'ed.)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
51	30	31	839
52	31	32	503
53	32	33	489
54	33	34	18
55	34	35	715
56	35	36	907
57	36	37	539
58	37	38	442
59	38	39	936
60	39	40	641

(G) 50-Vertex Network ($\bar{d}$ = 2.3)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
1	1	2	411
2	1	50	531
3	2	3	935
4	2	49	865
5	3	4	596
6	3	48	697
7	4	5	639
8	4	47	680
9	5	6	703
10	5	46	335
11	6	7	276
12	6	45	209
13	7	8	520
14	7	44	760
15	8	9	667
16	8	43	826
17	9	10	432
18	9	42	629
19	10	11	63
20	10	41	852
21	11	12	426
22	11	40	654
23	12	13	430
24	12	39	433
25	13	14	335
26	13	38	216
27	14	15	460
28	14	37	518
29	15	16	349
30	15	36	614
31	16	17	110
32	16	35	1091
33	17	18	277
34	17	34	66
35	18	19	112
36	18	33	615
37	19	20	184
38	19	32	772
39	20	21	556
40	20	31	206
41	21	22	485
42	21	30	694
43	22	23	149
44	22	29	529
45	23	24	402
46	23	28	702
47	24	25	936
48	24	26	110
49	25	27	859
50	28	30	866

(G) 50-Vertex Network (cont'ed.)

LINK	SOURCE(x_i)	SINK(x_j)	DISTANCE
51	29	31	540
52	34	36	265
53	35	37	164
54	38	40	362
55	41	43	800
56	44	46	543
57	47	49	143
58	48	50	682

REFERENCES*

1. Akinc, U. and Khumawala, B.M. (1977) "An efficient branch and bound algorithm for the capacitated warehouse location problem", Man. Sci., Vol.23, pp.585-594.

2. Balas, E. (1965) "An additive algorithm for solving linear programs with zero-one variables", Opns. Res., Vol.13, pp.517-546.

3. Balinski, M.L. (1965) "Integer programming: methods, uses, computation", Man. Sci., Vol.12, pp.253-313.

4. Balinski, M.L. and Mills, H. (1960) "A warehouse problem", Mathematica, Princeton, New Jersey.

5. Baumol, W.J. and Wolfe, P. (1958) " A warehouse location problem", Opns. Res., Vol.6, pp.252-263.

6. Beale, E.M.L. (1968) Mathematical Programming in Practice, Sir Isaac Pitman & Sons, London.

7. Beale, E.M.L. and Small, R.E. (1965) "Mixed integer programming by a branch and bound technique", Proc. IFIP Congress, New York.

8. Benders J.F. (1962) "Partitioning procedures for solving mixed-variables programming problems", Numerische Mathematik, Vol.4, pp.238-252.

9. Berge, C. (1973) Graphs and Hypergraphs (Translated by Edward Minieka), North-Holland Publishing Co., Amsterdam and London.

10. Bock, F. (1971) "An algorithm to construct a minimum directed spanning tree in a directed network", Developments in Operations Research (Edited by B. Avi-Itzhak), Gordon and Breach, pp.29-44.

11. Cabot, A.V., Francis, R.L. and Stary, M.A. (1970) "A network flow solution to a rectilinear distance facility location problem", AIIE Transactions, Vol.2, pp.132-141.

[*] The journal name abbreviations used in this list of references are identical to those used in the International Abstracts in Operations Research.

12. Christofides, N. (1975) Graph Theory: An Algorithmic Approach, Academic Press, London, New York and San Francisco.

13. Christofides, N. and Eilon, S. (1969) "An algorithm for the vehicle dispatching problem", Operat. Res. Quart., Vol.20, pp.309-318.

14. Christofides, N. and Eilon, S. (1972) "Algorithms for large scale travelling salesman problems", Operat. Res. Quart., Vol.23, pp.511-518.

15. Christofides, N. and Viola, P. (1971) "The optimum location of multi-centres on a graph", Operat. Res. Quart., Vol.22, pp.145-154.

16. Cooper, L. (1963) "Location-Allocation problems", Opns. Res., Vol.11, pp.331-343.

17. Cooper, L. (1964) "Heuristic methods for location-allocation problems", SIAM Review, Vol.6, pp.37-53.

18. Cooper, L. (1967) "Solutions of generalized locational equilibrium models", J. Reg. Sci., Vol.7, pp.1-18.

19. Cornuejols, G., Fisher, M.L. and Nemhauser,G.L. (1976) "An analysis of heuristics and relaxations for the uncapacitated location problem", Core Discussion Papers 7602, Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium.

20. Davis, P.S. and Ray, T.L. (1969) "A branch-and-bound algorithm for the capacitated facilities location problem", NRLQ, Vol.16, pp.331-344.

21. Dearing, P.M. and Francis, R.L. (1974) "A network flow solution to a multifacility minimax location problem involving rectilinear distances", Trans. Sci., Vol.8, pp.126-141.

22. Diehr, G. (1972) "An algorithm for the p-median problem", W.P. No.191, Western Management Science Institute, UCLA.

23. Domschke, W. (1975) "Modelle und verfahren zur bestimmung betrieblicher und innerbetrieblicher standorte - Ein überblick" ["Models and methods for determination of plant location and plant layout - A survey"], Zeitschrift für Operations Research, Vol.19, pp.B13-B41.

24. Edmonds, J. (1967) "Optimum branchings", Bureau of Standards J. of Res., Vol.71B, pp.233-240.

25. Efroymson, E. and Ray, T.L. (1966) "A branch-and-bound algorithm for plant location", Opns. Res., Vol.14, pp.361-368.

26. Eilon, S. (1972) "Goals and constraints in decision making", Operat. Res. Quart., Vol.23, pp.3-15.

27. Eilon, S., Watson-Gandy, C.D.T. and Christofides, N. (1971) Distribution Management: Mathematical Modelling & Practical Analysis, Griffin, London.

28. Elshafei, A.N. (1974) "Studies in facility location: survey of some methodologies and case studies in Great Britain", Memo 404, The Institute of National Planning, Cairo.

29. Elshafei, A.N. (1975) "An approach to locational analysis", Operat. Res. Quart., Vol.26, pp.167-181.

30. El-Shaieb, A.M. (1973) "A new algorithm for locating sources among destinations", Man. Sci., Vol.20, pp.221-231.

31. Elson, D.G. (1972) "Site location via mixed-integer programming", Operat. Res. Quart., Vol.23, pp.31-43.

32. Elzinga, J., Hearn, D. and Randolph, W.D. (1976) "Minimax multifacility location with Euclidean distances", Trans. Sci., Vol.10, pp.321-336.

33. Feldman, E.,Lehrer, F.A. and Ray, T.L. (1966) "Warehouse location under continuous economies of scale", Man. Sci., Vol.12, pp.670-684.

34. Floyd, R.W. (1962) "Algorithm 97-shortest path", Comm. of ACM, Vol.5, p.345.

35. Francis, R.L. and Goldstein, J.M. (1974) "Location theory: A selective bibliography", Opns. Res., Vol.22, pp.400-410.

36. Frank, H. (1966) "Optimum locations on a graph with probabilistic demands", Opns. Res., Vol.14, pp.409-421.

37. Frank, H. (1967) "Optimum locations on graphs with correlated normal demands", Opns. Res., Vol.15, pp.552-557.

38. Fulkerson, D.R. (1974) "Packing rooted directed cuts in a weighted directed graph", Math. Prog., Vol.6, pp.1-13.

39. Galvaŏ, R.D. (1977) "Fortran codes for the p-median problem", Internal Report, Dept. of Management Science, Imperial College of Science and Technology, London.

40. Garfinkel, R.S. and Nemhauser, G.L. (1972) Integer Programming, John Wiley and Sons.

41. Garfinkel, R.S., Neebe, A.W. and Rao, M.R. (1974) "An algorithm for the m-median plant location problem", Trans. Sci., Vol.8, pp.217-236.

42. Goldman, A.J. (1969) "Optimal locations for centers in a network", Trans. Sci., Vol.3, pp.352-360.

43. Goldman, A.J. (1971) "Optimal center location in simple networks", Trans. Sci., Vol.5, pp.212-221.

44. Goldman, A.J. (1972) "Minimax location of a facility in a network", Trans. Sci., Vol.6, pp.407-418.

45. Goldstein, A.J. (1962) Private Communication, Bell Telephone Laboratories, Murray Hill, N.J.

46. Gray, P. (1970) "Exact solutions of the site selection problem by mixed integer programming", Applications of Mathematical Programming Techniques (Edited by E.M.L. Beale), American Elsevier Publishing Co., New York.

47. Hadley, G. (1962) Linear Programming, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, Palo Alto and London.

48. Hakimi, S.L. (1964) "Optimum location of switching centers and the absolute centers and medians of a graph", Opns. Res., Vol.12, pp.450-459.

49. Hakimi, S.L. (1965) "Optimum distribution of switching centers in a communication network and some related graph theoretic problems", Opns. Res., Vol.13, pp.462-475.

50. Hakimi, S.L. and Maheshwari, S.N. (1972) "Optimum location of centers in networks", Opns. Res., Vol.20, pp.967-973.

51. Haley, K.B. (1963) "Siting of depots", IJPR, Vol.2, pp.41-45.

52. Handler, G.Y. (1974) "Minimax network location - theory and algorithms", Report FTL-R74-4, Flight Transportation Laboratory, Massachusetts Institute of Technology.

53. Hansen, P. (1976) "The simple plant location problem", OMEGA, Vol.4, pp.347-349.

54. Hu, T.C. (1969) Integer Programming and Network Flows, Addison-Wesley Publishing Co., Reading, Massachusetts.

55. Järvinen, P., Rajala, J. and Sinervo, H. (1972) "A branch-and-bound algorithm for seeking the p-median", Opns. Res., Vol.20, pp.173-178.

56. Juel, H. and Love, R.F. (1976) "An efficient computational procedure for solving the multifacility rectilinear facilities location problem", Operat. Res. Quart., Vol.27, pp.697-703.

57. Karg, R.L. and Thompson, G.L. (1964) "A heuristic approach to solving travelling salesman problems", Man. Sci., Vol.10, pp.225-248.

58. Karp, R.M. (1975) "On the computational complexity of combinatorial problems", Networks, Vol.5, pp.45-68.

59. Kerningan, B.W. and Lin, S. (1970) "An efficient heuristic procedure for partitioning graphs", Bell System Tech. J., Vol.49, pp.291-307.

60. Khumawala, B.M. (1972) "An efficient branch-and-bound algorithm for the warehouse location problem", Man. Sci., Vol.18, pp.B718-B731.

61. Khumawala, B.M. (1973) "An efficient heuristic procedure for the uncapacitated warehouse location problem", NRLQ, Vol.20, pp.109-121.

62. Khumawala, B.M. (1974) "An efficient heuristic procedure for the capacitated warehouse location problem", NRLQ, Vol.21, pp.609-623.

63. Khumawala, B.M., Neebe, A.W. and Dannenbring, D.G. (1974) "A note on El-Shaieb's new algorithm for locating sources among destinations", Man. Sci., Vol.21, pp.230-233.

64. Kruskal, J.B. Jr. (1956) "On the shortest spanning subtree of a graph and the travelling salesman problem", Proc. American Math. Soc., Vol.7, pp.48-50.

65. Kuehn, A.A. and Hamburger, M. (1963) "A heuristic program for locating warehouses", Man. Sci., Vol.9, pp.643-666.

66. Kuenne, R.E. and Soland, R.M. (1971) "The multisource Weber problem", IDA Economic Papers.

67. Kuenne, R.E. and Soland, R.M. (1972) "Exact and approximate solutions to the multisource Weber problem", Math. Prog., Vol.3, pp.193-209.

68. Kuhn, H.W. (1955) "The hungarian method for the assignment problem", NRLQ, Vol.2, pp.83-97.

69. Kuhn, H.W. (1956) "Variants of the hungarian method for assignment problems", NRLQ, Vol.3, pp.253-258.

70. Kuhn, H.W. and Kuenne, R.E. (1962) "An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics", J. Reg. Sci., Vol.4, pp.21-33.

71. Lawler, E.L. and Wood, D.E. (1966) "Branch-and-bound methods: a survey", Opns. Res., Vol.14, pp.699-719.

72. Lemke, C.E. and Spielberg, K. (1967) "Direct search algorithms for zero-one and mixed-integer programming", *Opns. Res.*, Vol.15, pp.892-914.

73. Levy, J. (1967) "An extended theorem for location on a network", *Operat. Res. Quart.*, Vol.18, pp.433-442.

74. Lin, S. (1965) "Computer solutions of the travelling salesman problem", *Bell System Tech. J.*, Vol.44, pp.2245-2269.

75. Little, J.D.C., Murty, K.G., Sweeney, D.W. and Karel, C. (1963) "An algorithm for the travelling salesman problem", *Opns. Res.*, Vol.11, pp.972-989.

76. Maranzana, F.E. (1964) "On the location of supply points to minimize transport costs", *Operat. Res. Quart.*, Vol.15, pp.261-270.

77. Marks, D.H. (1969) "Facility location and routing models in solid waste collection systems", *Ph.D. Thesis*, The John Hopkins University.

78. Marsten, R.E. (1972) "An algorithm for finding almost all of the medians of a network", *Discussion Paper No.23*, Northwestern University.

79. Miehle, W. (1958) "Link-length minimization in networks", *Opns. Res.*, Vol.6, pp.232-243.

80. Minieka, E. (1970) "The m-center problem", *SIAM Review*, Vol.12, pp.138-139.

81. Morris, J.G. (1974) "On linear programming solutions to a class of location-allocation problems", Kent State University, Kent, Ohio.

82. Morris, J.G. (1975) "A linear programming solution to the generalized rectangular distance Weber problem", *NRLQ*, Vol.22, pp.155-164.

83. Murchland, J.D. (1965) "A new method for finding all elementary paths in a complete directed graph", *Report LSE-TNT-22*, London School of Economics.

84. Orchard-Hays, W. (1968) *Advanced Linear Programming Techniques*, McGraw-Hill Book Company.

85. Palermo, F.P. (1961) "A network minimization problem", IBM Journal of Research and Development, Vol.5, pp.335-337.

86. Prim, R.C. (1957) "Shortest connection networks and some generalizations", Bell System Tech. J., Vol.36, pp.1389-1401.

87. Rand, G.K. (1976) "Methodological choices in depot location studies", Operat. Res. Quart., Vol.27, pp.241-249.

88. Rao, M.R. (1972) "The rectilinear facilities location problem", Working Paper no. F7215, The Graduate School of Management, University of Rochester, Rochester, New York.

89. Revelle, C.,Marks, D. and Liebman, J.C. (1970) "An analysis of private and public sector location models", Man. Sci., Vol.16, pp.692-707.

90. Revelle, C.S. and Swain, R.W. (1970) "Central facilities location", Geogr. Anal., Vol.2, pp.30-42.

91. Robers, P.D. (1971) "Some comments concerning Revelle, Marks and Liebman's article on facility location", Man. Sci., Vol.18, pp.109-111.

92. Sá, G. (1969) "Branch-and-bound and approximate solutions to the capacitated plant location problem", Opns. Res., Vol.17, pp.1005-1016.

93. Soland, R.M. (1974) "Optimal facility location with concave costs", Opns. Res., Vol.22, pp.373-382.

94. Spielberg, K. (1969) "Algorithms for the simple plant location problem with some side conditions", Opns. Res., Vol.17, pp.85-111.

95. Spielberg,, K. (1969) "Plant location with generalized search origin", Man. Sci., Vol.16, pp.165-178.

96. Surkis, J. (1967) "Optimal warehouse location", XIV International TIMS Conference, Mexico City.

97. Swain, R.W. (1974) "A parametric decomposition approach for the solution of uncapacitated location problems", Man. Sci., Vol.21, pp.189-198.

98. Teitz, M.B. and Bart, P. (1968) "Heuristic methods for estimating the generalized vertex median of a weighted graph", *Opns. Res.*, Vol.16, pp.955-961.

99. Toregas, C., Swain, R., Revelle, C. and Bergman, L. (1971) "The location of emergency service facilities", *Opns. Res.*, Vol.19, pp.1363-1373.

100. Watson-Gandy, C.D.T. and Eilon, S. (1972) "The depot siting problem with discontinuous delivery cost", *Operat. Res. Quart.*, Vol.23, pp.277-287.

101. Weber, A. (1909) *Uber den Standort der Industrien*, Tübingen. Translated as *Alfred Weber's Theory of the Location of Industries*, by Friedrich, C.J. (1929), University of Chicago Press.

102. Wendell, R.E. and Hurter, A.P. (1973) "Optimal locations on a network", *Trans. Sci.*, Vol.7, pp.18-33.

103. Wesolowsky, G.O. (1972) "Rectangular distance location under the minimax optimality criterion", *Trans. Sci.*, Vol.6, pp.103-113.

104. Yoeng-Jin, C. and Tseng-Hong, L. (1965) "On the shortest arborescence of a directed graph", *Scientia Sinica*, Vol. XIV, pp.1396-1400.