LOCATION PROBLEMS ON TREES AND IN THE RECTILINEAR PLANE

Antoon Kolen

LOCATION PROBLEMS ON TREES AND IN THE RECTILINEAR PLANE

LOCATION PROBLEMS ON TREES AND IN THE RECTILINEAR PLANE

ACADEMISCH PROEFSCHRIFT

TER VERKRIJGING VAN DE GRAAD VAN
DOCTOR IN DE WISKUNDE EN NATUURWETENSCHAPPEN
AAN DE UNIVERSITEIT VAN AMSTERDAM
OP GEZAG VAN DE RECTOR MAGNIFICUS
DR. D.W. BRESTERS
HOOGLERAAR IN DE FACULTEIT
DER WISKUNDE EN NATUURWETENSCHAPPEN
IN HET OPENBAAR TE VERDEDIGEN
IN DE AULA DER UNIVERSITEIT
(TIJDELIJK IN DE LUTHERSE KERK, INGANG SINGEL 411, HOEK SPUI)
OP WOENSDAG 3 MAART 1982 DES NAMIDDAGS TE 3.00 UUR PRECIES

DOOR

ANTHONIUS WILHELMUS JOHANNES KOLEN

GEBOREN TE TILBURG

1982
MATHEMATISCH CENTRUM, AMSTERDAM

PROMOTOR: PROF.DR. G. DE LEVE

COPROMOTOR: DR. J.K. LENSTRA

COREFERENT: PROF.DR. A.H.G. RINNOOY KAN

PREFACE

This study was written at the Department of Operations Research of the Mathematisch Centrum in Amsterdam under the supervision of prof.dr. G. de Leve and dr. J.K. Lenstra. I am very grateful to prof.dr. A.H.G. Rinnooy Kan and dr. J.K. Lenstra for suggesting location theory as an area for research and to the Mathematisch Centrum for the opportunity to study combinatorial optimization.

I have benefited from many inspiring discussions with Lex Schrijver and Jan Karel Lenstra on the first versions of this manuscript and on the research reports I have written over the years.

I want to acknowledge the useful cooperation with Andries Brouwer which made it possible to prove the main result with respect to totally-balanced matrices on which Chapter 2 is based.

With respect to the technical realization of this book, I would like to thank Susan Carolan for typing the manuscript, Tobias Baanders for the cover design, D. Zwarst, J. Schipper, J. Suiker, F. Swenneker and J. van der Werf for the printing.

Antoon Kolen

Amsterdam, January 1982

CONTENTS

If you want to improve your mathematics,
cut down on your errors.

Dedicated to my parents and Miriam

CHAPTER 0

INTRODUCTION

Location theory dates back to the beginning of the seventeenth century, when Fermat posed his famous problem: given three points in a plane, find a fourth point such that the sum of its distances to the three given points is minimum. This problem was generalized by Simpson in his "Doctrine and Application of Fluxious" (London 1750), who asked for a point with minimum weighted sum of distances to three given points. Almost all the work on location theory, however, has taken place between 1957 and the present. Papers on the subject appear in a remarkably diverse collection of periodicals in the field of operations research, management science, industrial and civil engineering, transportation and regional science, geography, economics and planning. As a consequence, many results have been rediscovered over the years and results obtained by researchers in one field were not known to those in other fields. The latter is especially true for results in graph theory and combinatorial optimization. As we will show in this monograph, results on perfect graphs and totally-balanced matrices have only very recently found their way into location theory and provide a uniform framework in which old as well as new results can be derived. Communication in the field of location theory has increased since ISOLDE I, the First International Symposium On Locational Decisions, held in Banff, Canada in April 1978, and ISOLDE II, held in Skodsborg, Denmark in June 1981.

The class of location problems is characterized by the following features: it is desired to locate a number of new facilities in a given space so as to provide goods or services to a specified set of existing facilities and to optimize one or several criteria, subject to a number of constraints. The quality of the service is typically measured in terms of the distances among the facilities. We will encounter several variations of the general location problem depending upon the type of location space (a network or the plane), the distance function (shortest path length or rectilinear distance), the number of facilities (fixed or variable), the type of interaction between facilities (between a new facility and all existing facilities within a certain distance, or between all pairs of facilities), and the optimality criterion (minimize the weighted sum of distances, minimize the maximum distance, minimize the total cost associated with establishing new facilities).

The theory of computational complexity provides the tools to make a formal distinction between *well-solved* problems, which are solvable by an algorithm whose running time is bounded by a polynomial function of problem size, and NP-*hard* problems, for which the existence of such an algorithm is highly unlikely. The exposition below is adapted from LAWLER and LENSTRA [1981]. Computational complexity theory deals primarily with *decision* problems, which require a yes/no answer, rather than with optimization problems. By way of example consider the following decision problem that will occur in Chapters 3 and 4:

CLIQUE
Instance: A graph $G = (V,E)$ and an integer c.
Question: Does there exist a subset $C \subset V$ of cardinality c such that $\{j,k\} \in E$ if $\{j,k\} \subset C$?

An instance of a decision problem is said to be *feasible* if the question can be answered affirmatively. Feasibility is usually characterized by the existence of an associated *structure* which satisfies a certain property. E.g., in the case of CLIQUE the structure is a set of c pairwise adjacent vertices.

Two important problem classes are defined as follows. A decision problem is in the class P if, for any instance, one can determine its feasibility or infeasibility in polynomial time. It is in the class NP if, for any instance, one can determine in polynomial time whether a given structure affirms its feasibility. E.g., CLIQUE is a member of NP, since for any clique C one can verify that all its vertices are pairwise adjacent in $O(c^2)$ time.

A problem R is *reducible* to a problem Q if for any instance of problem R a corresponding instance of problem Q can be constructed in polynomial time such that solving the latter instance will solve the instance of R as well. In the class NP, a class of NP-*complete* problems is identified which have the property that: (i) they belong to NP; and (ii) every other problem in NP is reducible to them. A polynomial-time algorithm for an NP-complete problem Q could be used to solve any problem in NP in polynomial time by reducing it to Q. Since the class NP contains many notorious combinatorial problems for which, in spite of considerable research efforts, no polynomial-time algorithms have been found so far, it is very unlikely that $Q \in P$ for any NP-complete Q.

In dealing with the computational complexity of *optimization* problems, one usually reformulates the problem of finding a feasible solution of, say, minimum value as the problem of deciding whether there exists a feasible solution with value at most equal to a given threshold. If this decision problem is NP-complete, then the optimization problem is said to be NP-*hard* in the sense that the existence of a polynomial-time algorithm for its solution would imply that $P = NP$.

Most location problems on graphs, where the distance between two points is defined as the length of a shortest path between these two points, have been proved to be NP-hard. This justifies the investigation of restricted versions of these problems. We will make the assumption that the graph does not contain cycles, i.e., that it is a tree. It turns out that most location problems on trees can be solved in polynomial time. It is not well understood what it is about tree location problems that makes them so tractable. We will partially answer this question by relating tree location problems to convexity, perfect graphs and totally-balanced matrices.

The research on location problems has frequently been justified as having direct applicability to a wide variety of real world location and logistical problems. Yet very few applications of location models are discussed or even mentioned in the literature. Although there are indications that the techniques are incidentally being applied both in the private and public sector, it seems clear that if models and solution algorithms were commonly used there would be more evidence of this in the literature. There are, of course, built-in biases against the publication of applications. Most researchers are theoreticians and not practitioners, and that practitioners have a lesser propensity to publish. There is also a lag effect between invention and adoption of algorithms. Although tree location problems have a nice mathematical structure, it remains to be tested how useful the developed algorithms are in more complex real world problems. They could be used for example in branch-and-bound algorithms as well as in heuristics for more general problems. Not much work in this direction has been done.

At this point we give a preview of what follows. In Chapter 1 we give some basic properties of trees and also introduce the concept of a chordal graph and a totally-balanced matrix. We indicate how these concepts relate to tree location problems.

In Chapter 2 we give a polynomial-time algorithm to solve the set covering problem on a totally-balanced matrix. It is shown that a totally-balanced matrix can be recognized in polynomial time. The simple plant location problem on a tree is given as an example of a tree location problem in which totally-balanced matrices appear. This problem is well solved.

In Chapters 3 and 4 we mention problems with mutual communication, i.e., problems in which there is interaction between all pairs of facilities, both new and existing. We treat the p-median problem in Chapter 3 and the p-center problem in Chapter 4.

In Chapter 5 we consider the nonlinear round-trip p-center and covering problems. The solution procedure developed for these problems is typical for the class of center and covering location problems on trees.

In Chapter 6 we show how Farkas' Lemma can be used to solve location problems in the plane using rectilinear distances.

We assume the reader of this monograph to be familiar with the basic principles of mathematical optimization including network flow theory and linear programming. Although concepts of graph theory are defined when needed, some familiarity with graph theory will be useful.

CHAPTER 1

TREES

In this chapter we will study properties of trees and subtrees that will be used in later chapters. Many location problems which are NP-hard when defined on an arbitrary graph become solvable in polynomial time when the graph is a tree. DEARING, FRANCIS & LOWE [1976] ask the question what it is about tree location problems that makes them so tractable. They were not able to answer this question but indicated that for some problems the answer is related to convexity properties of distance functions defined on trees; we will be more precise in Section 1.1. We will provide further insight into the answer to the question by relating covering problems on a tree to the problem of finding a minimum clique cover of a chordal graph in Section 1.2 and by relating minimum cost location problems on a tree to the set covering problem on a totally-balanced matrix in Section 1.3. How these relations can be explored to solve tree location problems is demonstrated by the cover and p-center problems considered in Section 1.2 and by the minimum cost covering problems dealt with in Section 1.3.

1.1. Tree properties

A *graph* is defined to be a pair (V,E) where V is a set of elements called *vertices*, and E is a set of unordered pairs of vertices, called *edges*. Two vertices v_i and v_j are *adjacent* if $\{v_i,v_j\} \in E$; v_i and v_j are the *endpoints* of the edge $\{v_i,v_j\}$. A *q-chain* is a sequence $(e_1,e_2,\ldots,e_q)$ of edges such that each edge e_i $(i = 2,\ldots,q-1)$ in the sequence has one endpoint in common with its predecessor in the sequence and its other endpoint with its successor in the sequence. The endpoint of $e_1(e_q)$ which is not an endpoint of $e_2(e_{q-1})$ is an *endpoint of the chain*. A *cycle* is a chain such that: (1) no edge appears twice in the sequence; and (2) the two endpoints of the chain are the same vertex. An *induced subgraph* of the graph $G = (V,E)$ is defined by a subset $V_1 \subseteq V$ and all edges of E with both endpoints in V_1. This subgraph is denoted by G_{V_1}.

A *directed graph* is defined to be a pair (V,A) where V is a set of elements called *vertices*, and A is a set of ordered pairs of vertices, called *arcs*. If $(v_i,v_j) \in A$, then v_i is a *predecessor* of v_j and arc (v_i,v_j), and v_j is a *successor* of v_i and arc (v_i,v_j). A *directed* q-*cycle* is a sequence $(a_1,a_2,\dots,a_q)$ of arcs such that: (1) no arc occurs twice in the cycle; and (2) the successor of arc a_i is the predecessor of a_{i+1} $(i = 1,\dots,q)$, where $a_{q+1} = a_1$.

A *tree* is a graph without cycles. Let T = (V,E) be a tree. Each edge $e \in E$ has a positive length $\ell(e)$. An edge $e = \{v_i,v_j\}$ is identified with a line segment of length $\ell(e)$ so that we can talk about any point on e at a distance λ from v_i and $\ell(e)-\lambda$ from v_j $(0 \le \lambda \le \ell(e))$. A *point on the tree* can be a vertex or a point anywhere along the edge. When X is a set of points on the tree T we use the notation $X \subset T$. The *distance* d(x,y) between two points x and y on T is defined to be the length of the shortest path between x and y; we denote this path by P(x,y). A *subtree* of the tree is a connected subset of the tree, not necessarily an induced subgraph. An example of a subtree is the shortest path between two points on the tree. A subtree T_1 is a *neighbourhood subtree* if there exists a fixed point x on T (called *center*) and a nonnegative number r (called *radius*) such that $T_1 = \{y \in T \mid d(y,x) \le r\}$.

LEMMA 1.1.1. *Let* x, y *and* z *be three points on a tree* T. *Then there is a point* $u \in T$ *such that* $u \in P(x,y) \cap P(y,z) \cap P(z,x)$.

PROOF. If two of the three points x, y and z are equal, then the result is trivial. So assume x, y and z are three different points. Define u to be the point on $P(x,y) \cap P(x,z)$ at largest distance from x. If $u \notin P(y,z)$, then define v to be the point on $P(x,z) \cap P(z,y)$ at largest distance from z, and w to be the point on $P(x,y) \cap P(y,z)$ at largest distance from y. Then P(u,v), P(v,w) and P(w,u) form a cycle, which contradicts the tree assumption. We conclude that $u \in P(y,z)$. □

COROLLARY 1.1.2. *Let* x, y, z *and* t *be points on* T *such that* $t \in P(x,y)$. *Then* $t \in P(x,z)$ *or* $t \in P(y,z)$.

PROOF. There exists a point $u \in T$ such that $u \in P(x,y) \cap P(y,z) \cap P(z,x)$. Since $t \in P(x,y)$, we have $t \in P(x,u)$ or $t \in P(u,y)$. If $t \in P(x,u)$, then $t \in P(x,z)$. If $t \in P(u,y)$, then $t \in P(y,z)$. □

<u>LEMMA 1.1.3.</u> *Let* T_1 *be a subtree of a tree* T *and let* x, y *be two points on* T_1. *Then* $P(x,y) \subseteq T_1$.

<u>PROOF.</u> Since T_1 is connected there is a shortest path in T_1 connecting x and y. If this path is not equal to $P(x,y)$, then this would introduce a cycle in the tree. □

<u>COROLLARY 1.1.4.</u> *The intersection of two subtrees* T_1 *and* T_2 *is a subtree.*

<u>PROOF.</u> Let $x,y \in T_1 \cap T_2$. Then $P(x,y) \subseteq T_1 \cap T_2$. This proves that $T_1 \cap T_2$ is connected. □

<u>LEMMA 1.1.5.</u> (Helly Property). *Let* $T_1, T_2, \ldots, T_m$ *be subtrees such that* $T_i \cap T_j \neq \emptyset$ *for all* $i,j = 1,2,\ldots,m$. *Then* $\cap_{i=1}^{m} T_i \neq \emptyset$.

<u>PROOF.</u> The proof proceeds by induction on m. Consider $m = 3$. Let $x \in T_1 \cap T_2$, $y \in T_1 \cap T_3$ and $z \in T_2 \cap T_3$. It follows from Lemma 1.1.3 that $P(x,y) \subseteq T_1$, $P(x,z) \subseteq T_2$ and $P(y,z) \subseteq T_3$, and from Lemma 1.1.1 that there is a point $u \in P(x,y) \cap P(y,z) \cap P(z,x)$. Hence $u \in T_1 \cap T_2 \cap T_3$. Suppose the lemma is true for less than m subtrees ($m \geq 4$) and consider $T_1, T_2, \ldots, T_m$. By induction we have $\cap_{i=1}^{m-2} T_i \neq \emptyset$. It follows from Corollary 1.1.4 that $\cap_{i=1}^{m-2} T_i$ is a subtree. Consider the three subtrees $\cap_{i=1}^{m-2} T_i$, T_{m-1} and T_m. By induction we have $(\cap_{i=1}^{m-2} T_i) \cap T_{m-1} \neq \emptyset$ and $(\cap_{i=1}^{m-2} T_i) \cap T_m \neq \emptyset$. Since $T_m \cap T_{m-1} \neq \emptyset$ we have by induction $\cap_{i=1}^{m} T_i \neq \emptyset$. □

<u>COROLLARY 1.1.6.</u> *Given a tree* $T = (V,E)$ *with vertex set* $V = \{v_1, v_2, \ldots, v_n\}$ *and nonnegative numbers* r_i $(i = 1,2,\ldots,n)$. *There exists a point* $x \in T$ *such that* $d(v_i,x) \leq r_i$ *for all* $i = 1,2,\ldots,n$ *if and only if* $d(v_i,v_j) \leq r_i + r_j$ *for all* $i,j = 1,2,\ldots,n$, $i \neq j$.

<u>PROOF.</u> Define the neighbourhood subtrees $T_i = \{y \in T \mid d(v_i,y) \leq r_i\}$, $i = 1,2,\ldots,n$. According to Lemma 1.1.5 it is sufficient to show that $T_i \cap T_j \neq \emptyset$ if and only if $d(v_i,v_j) \leq r_i + r_j$, $i,j = 1,2,\ldots,n$, $i \neq j$.

If $T_i \cap T_j \neq \emptyset$, then there is an $x \in T_i \cap T_j$. Hence $d(v_i,v_j) \leq d(v_i,x) + d(x,v_j) \leq r_i + r_j$.

Conversely, let $d(v_i,v_j) \leq r_i + r_j$. If $r_i \geq d(v_i,v_j)$, then $v_j \in T_i \cap T_j$. If $r_i < d(v_i,v_j)$, then define x to be the point on $P(v_i,v_j)$ at distance r_i from v_i. We have $x \in T_i$ and $d(v_j,x) = d(v_i,v_j) - d(v_i,x) = d(v_i,v_j) - r_i \leq r_j$ so that $x \in T_j$. We conclude that $T_i \cap T_j \neq \emptyset$. □

We will use Corollary 1.1.6 in Chapter 4 to derive necessary and sufficient conditions for the existence of points $x_1,x_2,\ldots,x_p$ on T such that
(1) $d(v_i,x_j) \le c_{ij}$, $i = 1,2,\ldots,n$, $j = 1,2,\ldots,p$, and
(2) $d(x_j,x_k) \le b_{jk}$, $j,k = 1,2,\ldots,p$,
where c_{ij} $(i = 1,2,\ldots,n,\ j = 1,2,\ldots,p)$ and b_{jk} $(j,k = 1,2,\ldots,p)$ are non-negative numbers. Note that Corollary 1.1.6 solves this problem for $p = 1$. These necessary and sufficient conditions will be the main result needed in solving the p-*center problem with mutual communication,* which is given by

$$\begin{aligned}\min_{x_1,x_2,\ldots,x_p}\{\max\{&\max_{i=1,\ldots,n,j=1,\ldots,p}\{\alpha_{ij}d(v_i,x_j)\},\\ &\max_{j,k=1,\ldots,p}\{\beta_{jk}d(x_j,x_k)\}\}\},\end{aligned}$$

where α_{ij} $(i = 1,2,\ldots,n,\ j = 1,\ldots,p)$ and β_{jk} $(j,k = 1,2,\ldots,p)$ are non-negative numbers.

Let x and y be two different points on the tree. The point $\lambda x + (1-\lambda)y$ $(0 \le \lambda \le 1)$ is defined to be the point on $P(x,y)$ at distance $\lambda d(x,y)$ from x. Let v, x_i and y_i $(i = 1,2)$ be fixed points on the tree. DEARING, FRANCIS & LOWE [1976] proved that $d(v,\lambda x + (1-\lambda)y) \le \lambda d(v,x) + (1-\lambda)d(v,y)$ and $d(\lambda x_1 + (1-\lambda)y_1,\ \lambda x_2 + (1-\lambda)y_2) \le \lambda d(x_1,x_2) + (1-\lambda)d(y_1,y_2)$. Due to a continuity argument it is sufficient to show that these relations hold for $\lambda = \frac{1}{2}$.

<u>LEMMA 1.1.7</u>. *Given two points* $x,y \in T$, *let* $z = \frac{1}{2}x + \frac{1}{2}y$. *Then* $d(v,z) \le \frac{1}{2}d(v,x) + \frac{1}{2}d(v,y)$ *for all* $v \in T$.

<u>PROOF</u>. We have $z \in P(x,v)$ or $z \in P(y,v)$. Without loss of generality assume $z \in P(x,v)$. Then

$$\begin{aligned}d(v,z) &= d(v,x) - d(x,z)\\ &= d(v,x) - \tfrac{1}{2}d(x,y)\\ &= \tfrac{1}{2}d(v,x) + \tfrac{1}{2}(d(v,x) - d(x,y))\\ &\le \tfrac{1}{2}d(v,x) + \tfrac{1}{2}d(v,y). \quad \square\end{aligned}$$

LEMMA 1.1.8. *Given four points* $x_1, x_2, y_1, y_2 \in T$, *let* $z_i = \frac{1}{2} x_i + \frac{1}{2} y_i$ $(i = 1,2)$. *Then* $d(z_1,z_2) \le \frac{1}{2} d(x_1,x_2) + \frac{1}{2} d(y_1,y_2)$.

PROOF. We have $z_1 \in P(x_1,z_2)$ or $z_1 \in P(y_1,z_2)$. Without loss of generality assume $z_1 \in P(x_1,z_2)$. Then

$$\begin{aligned}
d(z_1,z_2) &= d(z_2,x_1) - d(x_1,z_1) \\
&\le \frac{1}{2} d(x_1,x_2) + \frac{1}{2} d(x_1,y_2) - d(x_1,z_1) \qquad \text{(Lemma 1.1.7)} \\
&= \frac{1}{2} d(x_1,x_2) + \frac{1}{2} (d(x_1,y_2) - d(x_1,z_1)) - \frac{1}{2} d(y_1,z_1) \\
&\qquad\qquad (d(x_1,z_1) = d(y_1,z_1)) \\
&\le \frac{1}{2} d(x_1,x_2) + \frac{1}{2} d(y_2,z_1) - \frac{1}{2} d(y_1,z_1) \\
&\le \frac{1}{2} d(x_1,x_2) + \frac{1}{2} d(y_1,y_2). \quad \square
\end{aligned}$$

Define the function F by

$$F(X) = \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij} d(v_i,x_j) + \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{p} \beta_{jk} d(x_j,x_k),$$

where α_{ij} $(i = 1,2,\ldots,n,\ j = 1,2,\ldots,p)$ and β_{jk} $(j,k = 1,2,\ldots,p)$ are non-negative numbers. Finding the vector X which minimizes F(X) is known as the p-*median problem with mutual communication*. Lemmas 1.1.7 and 1.1.8 lead to the following theorem stating that the function F is convex.

THEOREM 1.1.9. $F(\lambda X + (1-\lambda)Y) \le \lambda F(X) + (1-\lambda)F(Y)$ *for* $0 \le \lambda \le 1$.

We will explore this convexity property in Chapter 3 and derive necessary and sufficient conditions for optimality as well as an $O(np^3)$ algorithm to solve this problem.

1.2. Chordal graphs

A *chordal graph* is a graph with the property that any cycle of length at least four contains a *chord*, i.e., an edge connecting two vertices of the cycle which are not adjacent in the cycle. Chordal graphs are sometimes called *triangulated graphs* or *rigid circuit graphs*. A *clique* is a subset of pairwise adjacent vertices. A *simplicial vertex* is a vertex with the property that all vertices adjacent to it form a clique. Let $G = (W,F)$ be a graph. A subset $S \subset W$ is a *vertex separator* for nonadjacent vertices b and c (or a b-c separator) if after the removal of S from the graph b and c belong to distinct connected components. If no proper subset of S is a b-c separator, then S is a *minimal vertex separator* for b and c.

EXAMPLE 1.2.1. The graph $G = (W,F)$ given in Figure 1.2.2 is chordal. The vertex w_1 is a simplicial vertex since $\{w_1,w_2,w_5\}$ induces a clique. A minimal vertex separator for w_1 and w_3 is $\{w_2,w_5\}$. Note that $\{w_2,w_5\}$ induces a clique. This is not a coincidence as is demonstrated by the following well-known lemma.

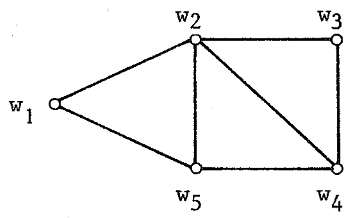

Figure 1.2.2. The graph $G = (W,F)$. □

LEMMA 1.2.3. *If* $G = (W,F)$ *is a chordal graph, then every minimal vertex separator induces a clique of* G.

PROOF. Suppose S is a minimal b-c separator. Let B and C be subsets of W inducing the connected components of $G_{W\backslash S}$ containing b and c, respectively. Since S is minimal each $x \in S$ is adjacent to some vertex in B and some vertex in C. Therefore, for any pair $x,y \in S$ there exist paths $[x,b_1,\dots,b_r,y]$ and $[y,c_1,\dots,c_t,x]$ where each $b_i \in B$ and $c_i \in C$, such that these paths are chosen to be of smallest possible length. It follows that $[x,b_1,\dots,b_r,y, c_1,\dots,c_t,x]$ is a cycle whose length is at least four, implying that it must have a chord. But $\{b_i,c_j\} \notin F$ by the definition of vertex separator, and $\{b_i,b_j\} \notin F$ and $\{c_i,c_j\} \notin F$ by the minimality of r and t. Thus, the only possible chord is $\{x,y\} \in F$. □

The following lemma will be needed in Section 2.3.

LEMMA 1.2.4. *Let* $G = (W,F)$ *be a chordal graph, let* S *be a minimal* b-c *separator and let* G_B *and* G_C *be the connected components of* $G_{W\setminus S}$ *containing* b *and* c, *respectively. Then there exists a vertex* d *in* G_B *such that* $\{d\} \cup S$ *induces a clique.*

PROOF. We shall prove that for each subset $X \subseteq S$ there exists a vertex in G_B which together with X induces a clique. Since S is a minimal vertex separator this is true for all subsets of S of cardinality one. Assume the result holds for all subsets of S of cardinality at most k $(k \geq 1)$ and let $X \subseteq S$ be a set of cardinality $k+1$. Choose $y \in X$. There is a vertex $z \in B$ such that $\{x,z\} \in F$ for all $x \in X\setminus\{y\}$ and there is a vertex $t \in B$ such that $\{y,t\} \in F$. If $\{x,t\} \in F$ for all $x \in X\setminus\{y\}$, then the result follows. So assume there is a vertex $x \in X\setminus\{y\}$ such that $\{x,t\} \notin F$. Let $[b_1,\ldots,b_r]$ with $b_1 = z$, $b_r = t$ be the path in G_B connecting z and t of smallest possible length. Then $[y,x,b_1,\ldots,b_r,y]$ is a cycle of length at least four, implying that it must have a chord. But $\{x,t\} \notin F$ by assumption and $\{b_i,b_j\} \notin F$ by minimality of r. Thus, the only possible chord is $\{y,z\}$. It follows that $\{z\} \cup X$ induces a clique. □

The next theorem due to DIRAC [1961] plays a major role in the theory on chordal graphs.

THEOREM 1.2.5. *Every chordal graph* $G = (W,F)$ *has a simplicial vertex. Moreover, if* G *is not a clique, then it has two non-adjacent simplicial vertices.*

PROOF. The theorem is trivial if G is complete. Assume G has two non-adjacent vertices b and c and that the theorem is true for all graphs with fewer vertices than G. Let S be a minimal vertex separator for b and c with G_B and G_C being the connected components of $G_{W\setminus S}$ containing b and c, respectively.

By induction, either the subgraph $G_{B\cup S}$ has two non-adjacent simplicial vertices one of which must be in B (since S induces a clique), or $G_{B\cup S}$ is itself complete and any vertex in B is simplicial in $G_{B\cup S}$. Furthermore since the set of all vertices connected to a vertex in B is contained in $B \cup S$, a simplicial vertex of $G_{B\cup S}$ in B is a simplicial vertex in all of G. Similarly C contains a simplicial vertex of G. This proves the theorem. □

A *perfect scheme* of a graph $G = (W,F)$ is a mapping σ: $W \to \{1,2,\ldots,|W|\}$ such that for all vertices $v \in W$ the set of vertices $\{w|\{v,w\} \in F, \sigma(v) < \sigma(w)\}$ induces a clique. Since an induced subgraph of a chordal graph is a chordal graph it follows from Theorem 1.2.5 that a perfect scheme exists for a chordal graph. A perfect scheme σ can be obtained by defining $\sigma(w_1) = 1$ for a simplicial vertex w_1 and iteratively defining $\sigma(w_i) = i$, where w_i is a simplicial vertex of the subgraph induced by $W\backslash\{w_j \mid j = 1,\ldots,i-1\}$ $(i = 2,\ldots,|W|)$. An *orientation* of the graph $G = (W,F)$ transforms the graph into a directed graph by transforming each unordered pair of vertices corresponding to an edge into an ordered pair. An R-*orientation* of $G = (W,F)$ is an orientation of the edges such that: (1) the resulting digraph $H = (W,A)$ has no directed cycles; and (2) if $(w,v) \in A$ and $(u,v) \in A$, then $(u,w) \in A$ or $(w,u) \in A$.

EXAMPLE 1.2.6. Consider the graph $G = (W,F)$ of Example 1.2.1. The mapping σ defined by $\sigma(w_1) = 1$, $\sigma(w_5) = 2$, $\sigma(w_2) = 3$, $\sigma(w_4) = 4$, and $\sigma(w_3) = 5$ is a perfect scheme. An R-orientation of $G = (W,F)$ is given in Figure 1.2.7. Let $H = (W,A)$ denote the resulting digraph. Note that $A = \{(w_i,w_j) \mid \{w_i,w_j\} \in F$ and $\sigma(w_i) > \sigma(w_j)\}$.

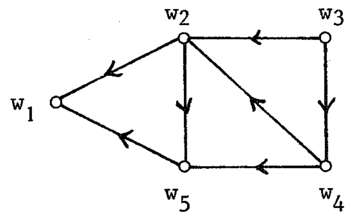

Figure 1.2.7. The digraph $H = (W,A)$. □

The following lemma of ROSE [1970] is a corollary of Dirac's Theorem.

LEMMA 1.2.8. *A graph* $G = (W,F)$ *is chordal if and only if an* R-*orientation exists.*

PROOF. Let $G = (W,F)$ be a chordal graph. There exists a perfect scheme σ. Define the set of arcs A by $A = \{(v,w) \mid \{v,w\} \in F$ and $\sigma(v) > \sigma(w)\}$. We claim that the orientation defined by A is an R-orientation. Clearly there are no directed cycles in $H = (W,A)$. Let $(u,v) \in A$ and $(w,v) \in A$. By definition of σ it follows that $\{u,w\} \in F$ so that $(u,w) \in A$ or $(w,u) \in A$.

Conversely assume that an R-orientation exists for $G = (W,F)$ and let

$H = (W,A)$ be the resulting digraph. Let C be a q-cycle ($q \geq 4$) in G. Since there are no directed cycles in H there is a vertex v which has no successors in the cycle C. Let $\{u,v\}$ and $\{w,v\}$ be the two edges in the cycle with v as endpoint. Then $(u,v) \in A$ and $(w,v) \in A$. By definition of an R-orientation we have $(u,w) \in A$ or $(w,u) \in A$ so that $\{u,w\} \in F$, i.e., $\{v,w\}$ is a chord of the cycle. □

Let $T = (V,E)$ be a tree with vertex set $V = \{v_1,v_2,\ldots,v_n\}$. Let $T_1,T_2,\ldots,T_m$ be subtrees of the tree T. The *intersection graph* of $T_1,\ldots,T_m$ is defined by the vertex set $\{1,2,\ldots,m\}$ and edge set $\{\{i,j\} \mid T_i \cap T_j \neq \emptyset\}$. The following theorem is due to BUNEMAN [1974] and GAVRIL [1974].

<u>THEOREM 1.2.9</u>. *A graph is a chordal graph if and only if it is the intersection graph of a number of subtrees of a tree.*

We will only use the fact that the intersection graph of a number of subtrees of a tree is a chordal graph. In order to prove this we renumber the vertices of the tree T from $1,2,\ldots,n$ as follows. Fix one vertex r of the tree T and call this the *root* of the tree. We define the *level* $\ell(v)$ of a vertex v to be the number of edges on the path $P(v,r)$. We renumber the vertices such that vertices at a certain level have a smaller number than vertices at a lower level. Let $T_1,T_2,\ldots,T_m$ be subtrees of T. We assume without loss of generality that the subtrees are induced subgraphs (if a subtree contains only part of an edge, then we can add the tip on this edge to the vertex set). With subtree T_i we associate its largest vertex with respect to $\{1,2,\ldots,n\}$, say r_i; we call r_i the *root* of T_i ($i = 1,2,\ldots,n$). If $r_i \leq r_j$, then $T_i \cap T_j \neq \emptyset$ if and only if $r_i \in T_j$.

<u>LEMMA 1.2.10</u>. *The intersection graph* $G = (\{1,2,\ldots,m\},F)$ *of subtrees* $T_1,T_2,\ldots,T_m$ *of the tree* T *is a chordal graph.*

<u>PROOF</u>. Define the set of arcs A by $A = \{(j,i) \mid \{i,j\} \in F \text{ and } r_i < r_j\} \cup \{(j,i) \mid \{i,j\} \in F,\ r_i = r_j \text{ and } i < j\}$. We claim that A defines an R-orientation. Clearly there are no directed cycles in the digraph $H = (\{1,2,\ldots,m\}, A)$. If $(j,i) \in A$ and $(k,i) \in A$, then $r_i \in T_j \cap T_k$ so that $\{j,k\} \in F$. It follows from Lemma 1.2.8 that G is chordal. □

An *independent set* of a graph is a subset of pairwise non-adjacent vertices. The cardinality of a maximum independent set of a graph G is denoted by $\alpha(G)$. A *covering by cliques* of a graph is a set of cliques such that each vertex is in at least one clique. The minimum cardinality of a clique cover is denoted by $\theta(G)$. It is a trivial observation that $\alpha(G) \leq \theta(G)$ for all graphs G. Chordal graphs belong to the family of *perfect graphs* (BERGE [1976]) which have several nice properties, two of which we mention here:

(1) $\alpha(G) = \theta(G)$ for a perfect graph G.

(2) Both a maximum independent set and a minimum clique cover can be calculated in polynomial time (GRÖTSCHEL et al. [1981]).

It is a simple matter to prove both of these results for chordal graphs. Assume the vertices are numbered according to a perfect scheme. A perfect scheme for a chordal graph $G = (W,F)$ as well as a maximum independent set and a minimum clique covering can be found in $O(|W|+|F|)$ time. For theory and algorithms on chordal graphs we refer to the book by GOLUMBIC [1980]. The following algorithm due to GAVRIL [1972] produces an independent set I and a clique cover C such that $|I| = |C|$. Therefore I is a maximum independent set and C is a minimum clique cover.

ALGORITHM. Add the vertex with smallest number to I. Let c be the set containing this vertex and all its adjacent vertices. By definition of a perfect scheme it follows that c is a clique. Add c to C and delete from the graph all vertices in c and all edges having at least one endpoint in c. Continue until there are no vertices left. □

Let us no indicate how the problem of finding a minimum clique cover relates to tree location problems. We mention three examples here and another one in Chapter 5 (also see CHANDRASEKARAN & TAMIR [1981]).

Suppose we want to locate facilities on the tree such that each vertex of the tree is within distance r from the closest facility. The *minimum covering problem* is to find the minimum number of facilities for which this is possible. A vertex $v \in T$ is *covered within distance* r when there is a facility located at point $x \in T$ such that $d(v,x) \leq r$. Define $T_i = \{y \in T \mid d(y,v_i) \leq r\}$ $(i = 1,2,\ldots,n)$. In order to cover v_i within distance r there must be a facility located at T_i $(i = 1,2,\ldots,n)$. It follows from Corollary 1.1.6 that $T_i \cap T_j \neq \emptyset$ if and only if $d(v_i,v_j) \leq 2r$. Construct the intersection graph $G = (\{1,2,\ldots,n\},F)$ of $T_1,T_2,\ldots,T_n$, i.e., $\{i,j\} \in F$ if and only if $d(v_i,v_j) \leq 2r$. It follows from Corollary 1.1.6 that if C is a clique of G, then there exists a facility location $x \in T$ such that all vertices v_i $(i \in C)$ can be covered within distance r from x. Conversely, let $\{v_i \mid i \in C \subseteq \{1,2,\ldots,n\}\}$ be the set of all vertices covered within distance r from a given facility location $x \in T$. Then C is a clique of G. This one-to-one correspondence between cliques of G and vertices which can be covered within distance r from the same facility location shows that the minimum covering problem is equivalent to finding a minimum clique cover in the intersection graph G. Let $d(y,X)$ denote the distance from the point $y \in T$ to the closest point in the set $X \subset T$. Due to the duality result $\alpha(G) = \theta(G)$ for the chordal graph G we have

(1.2.11) $$\max\{|I| \mid I \subseteq \{1,2,\ldots,n\},\ d(v_i,v_j) > 2r,\ i,j \in I,\ i \neq j\}$$

$$= \min\{|X| \mid X \subset T,\ d(v_i,X) \leq r,\ i = 1,2,\ldots,n\}.$$

The p-*center problem* is to find the minimum distance r such that each vertex is within distance r of the closest facility given the fact that we may locate p facilities. This problem can be formulated as

$$\min_{\substack{X \subset T \\ |X|=p}} \max_{1 \leq i \leq n} \{d(v_i,X)\}.$$

The following lemma states a duality result for the p-center problem.

LEMMA 1.2.12.

$$\max_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=p+1}} \min_{\substack{i,j \in I \\ i \neq j}} \{\tfrac{1}{2} d(v_i,v_j)\} = \min_{\substack{X \subset T \\ |X|=p}} \max_{1 \leq i \leq n} \{d(v_i,X)\}.$$

PROOF.

$$\min_{|X|=p} \{ \max_{1 \leq i \leq n} \{d(v_i,X)\}\} \leq r \Longleftrightarrow \min\{|X| \mid \max_{1 \leq i \leq n} \{d(v_i,X)\} \leq r\} \leq p$$

$$\Longleftrightarrow \max\{|I| \mid I \subseteq \{1,2,\ldots,n\},\ d(v_i,v_j) > 2r,\ i,j \in I,\ i \neq j\} \leq p \qquad (1.2.11)$$

$$\Longleftrightarrow \max_{\substack{I \subseteq \{1,2,\ldots,n\} \\ |I|=p+1}} \{ \min_{\substack{i,j \in I \\ i \neq j}} \{\tfrac{1}{2} d(v_i,v_j)\}\} \leq r. \quad \square$$

Lemma 1.2.12 shows that in order to solve the problem one only has to solve the minimum covering problem for values of r belonging to the set $\{\frac{1}{2} d(v_i,v_j) \mid i,j = 1,2,\ldots,n,\ i \neq j\}$. The solution to the p-center problem is the smallest value in this set for which the value of the covering problem is equal to p. Performing a bisection search on this set solves the p-center problem in $O(n^2 \log n)$ time. A minimum covering problem can be solved in $O(n)$ time. This method was followed by HAMIKI & KARIV [1979]. One can improve this bound to $O(n^2)$ as indicated by CHANDRASEKARAN & TAMIR [1981]. If one allows parallel processors the problem can be solved in $O(n \log^2 n \log \log n)$ time (MEGIDDO & TAMIR [1981]).

In the p-center problem defined above we assume that clients to be served by facilities are located at vertices. In case clients may be located anywhere along an edge we have the *continuous* p-*center problem* which can be formulated as

$$\min_{\substack{X \subset T \\ |X|=p}} \max_{y \in T} \{d(y,X)\}.$$

We have a duality result for the continuous p-center problem similar to the one given for the p-center problem:

$$(1.2.13) \qquad \max_{\substack{Y \subset T \\ |Y|=p+1}} \min_{\substack{y_i,y_j \in Y \\ i \neq j}} \{\tfrac{1}{2}\, d(y_i,y_j)\} = \min_{\substack{X \subset T \\ |X|=p}} \max_{y \in T} \{d(y,X)\}.$$

This result follows from a duality result due to SHIER [1977] in exactly the same way as we proved the duality result given by Lemma 1.2.12. SHIER [1977] proved that

$$(1.2.14) \qquad \max\{|Y| \mid Y \subset T,\ d(y_i,y_j) > 2r,\ y_i,y_j \in Y,\ i \neq j\} =$$
$$= \min\{|X| \mid X \subset T,\ d(y,X) \leq r,\ \forall y \in T\}.$$

The continuous p-center problem was solved by CHANDRASEKARAN & TAMIR [1981] by first identifying a finite set of values to which the optimum value of the continuous p-center problem belongs, and then solving a sequence of covering problems. Their algorithm requires $O((n \log p)^2)$ time. If one allows parallel processors the problem can be solved in $O(n \log^3 n)$ time (MEGIDDO & TAMIR [1981]).

1.3 Neighbourhood subtrees

This section will be devoted to neighbourhood subtrees and their properties. Neighbourhood subtrees can be regarded as generalizations of intervals. There are some properties of subintervals of a fixed interval which also hold for neighbourhood subtrees of a tree. Two of these properties are given in Lemma 1.3.1 and Theorem 1.3.2. Lemma 1.3.1 is well known (FRANCIS, LOWE & RATLIFF [1978]) and states that the intersection of two neighbourhood subtrees is again a neighbourhood subtree. Theorem 1.3.2 (KOLEN [1981]) states that neighbourhood subtrees containing an endpoint of a longest path in the tree can be totally ordered by inclusion. This property is very useful in solving minimum cost tree location problems as will be shown in Chapter 2.

LEMMA 1.3.1. *Let* $E_1 = \{y \in T \mid d(y,x_1) \le r_1\}$ *and* $E_2 = \{y \in T \mid d(y,x_2) \le r_2\}$ *be two neighbourhood subtrees of a tree* T *with* $E_1 \cap E_2 \ne \emptyset$. *Then* $E_1 \cap E_2$ *is a neighbourhood subtree.*

PROOF. Corollary 1.1.6 states that $E_1 \cap E_2 \ne \emptyset$ if and only if $d(x_1,x_2) \le r_1 + r_2$. Without loss of generality assume $r_1 \le r_2$.

If $r_2 \ge r_1 + d(x_1,x_2)$, then $E_1 \subseteq E_2$. This can be seen as follows. Let $y \in E_1$. Then $d(y,x_2) \le d(y,x_1) + d(x_1,x_2) \le r_1 + d(x_1,x_2) \le r_2$.

If $r_2 < r_1 + d(x_1,x_2)$, then define x_{12} to be the point on $P(x_1,x_2)$ at distance $\frac{1}{2}(r_1 - r_2 + d(x_1,x_2))$ from x_1 and define $r_{12} = \frac{1}{2}(r_1 + r_2 - d(x_1,x_2))$. We claim that $E_1 \cap E_2 = \{y \in T \mid d(y,x_{12}) \le r_{12}\}$. This can be proved as follows.

Let $y \in E_1 \cap E_2$. We have $x_{12} \in P(x_1,y)$ or $x_{12} \in P(x_2,y)$. Assume $x_{12} \in P(x_2,y)$ (the case $x_{12} \in P(x_1,y)$ can be treated analogously). Then

$$\begin{aligned} d(y,x_{12}) &= d(x_2,y) - d(x_2,x_{12}) \le r_2 - d(x_2,x_{12}) \\ &= r_2 - \frac{1}{2}(r_2 - r_1 + d(x_1,x_2)) \\ &= \frac{1}{2}(r_1 + r_2 - d(x_1,x_2)) = r_{12}. \end{aligned}$$

Conversely, let $d(y,x_{12}) \le r_{12}$. It follows that

$$d(y,x_1) \le d(y,x_{12}) + d(x_{12},x_1) \le r_{12} + \frac{1}{2}(r_1 - r_2 + d(x_2,x_2)) = r_1.$$

Analogously we prove that $d(y,x_2) \le r_2$. Hence $y \in E_1 \cap E_2$. □

THEOREM 1.3.2. *Let* $t_1, t_2 \in T$ *such that* $P(t_1,t_2)$ *is a longest path in the tree* T *and let* $E_1 = \{y \in T \mid d(y,x_1) \le r_1\}$ *and* $E_2 = \{y \in T \mid d(y,x_2) \le r_2\}$ *be neighbourhood subtrees such that* $t_1 \in E_1 \cap E_2$. *Then* $E_1 \subseteq E_2$ *or* $E_2 \subseteq E_1$.

PROOF. Define s_i $(i = 1,2)$ to be the point on E_i closest to t_2. Without loss of generality assume $d(s_2,t_2) \le d(s_1,t_2)$. We shall prove $E_1 \subseteq E_2$. If $s_2 = t_2$, then E_2 is equal to the entire tree. Therefore assume $s_2 \ne t_2$. Let $p \in E_1$. If $p \in P(t_1,s_1)$, then $p \in E_2$. Let p be such that $p \notin P(t_1,s_1)$ and define q to be the point on $P(t_1,s_1)$ closest to p. We shall prove in Lemma 1.3.3 that $d(p,q) \le d(t_1,q)$ and $d(p,q) \le d(s_1,q)$. We have $q \in P(x_2,t_1)$ or $q \in P(x_2,t_2)$. If $q \in P(x_2,t_1)$, then

$$d(p,x_2) \le d(p,q) + d(q,x_2) \le d(t_1,q) + d(q,x_2) = d(t_1,x_2) \le r_2.$$

Hence $p \in E_2$. If $q \in P(x_2,t_2)$ and $q \notin P(x_2,t_1)$, then

$$d(p,x_2) \le d(p,q) + d(q,x_2) \le d(s_1,q) + d(q,x_2)$$

$$= d(x_2,s_1) \le d(x_2,s_2) \le r_2.$$

Hence $p \in E_2$. It follows that $E_1 \subseteq E_2$. □

LEMMA 1.3.3. *Let* p, q, s_1, s_2, t_1, t_2, x_1 *and* x_2 *be the points defined in the proof of Theorem* 1.3.2. *Then* $d(p,q) \le d(t_1,q)$ *and* $d(p,q) \le d(q,s_1)$.

PROOF. Since $P(t_1,t_2)$ is a longest path we have $d(t_1,t_2) \ge d(p,t_2)$. From $d(p,t_2) = d(p,q) + d(q,t_2)$ and $d(t_1,t_2) = d(t_1,q) + d(q,t_2)$ it follows that $d(p,q) \le d(t_1,q)$.

In order to prove $d(p,q) \le d(s_1,q)$ we distinguish between $q \in P(x_1,t_1)$ or $q \in P(x_1,t_2)$. If $q \in P(x_1,t_1)$, then by definition of s_1 and since $s_1 \ne t_2$, we have $d(x_1,t_1) \le d(x_1,s_1)$. Hence

$$d(t_1,q) = d(x_1,t_1) - d(q,x_1) \le d(x_1,s_1) - d(q,x_1) \le d(q,s_1).$$

Since $d(p,q) \le d(t_1,q)$ it follows that $d(p,q) \le d(s_1,q)$. If $q \in P(x_1,t_2)$ and $q \notin P(x_1,t_1)$, then by definition of s_1 and since $s_1 \ne t_2$ we have $d(x_1,p) \le d(x_1,s_1)$. From $d(x_1,p) = d(x_1,q) + d(q,p)$ and $d(x_1,s_1) = d(x_1,q) + d(q,s_1)$ it follows that $d(p,q) \le d(q,s_1)$. □

Choose a vertex v, calculate the distances from v to all other vertices, and order them in nonincreasing order, say, $d(t_1,v) \geq d(t_2,v) \geq \ldots \geq d(t_n,v)$, where $t_n = v$. The subgraph T_i induced by $\{t_i,t_{i+1},\ldots,t_n\}$ is a subtree of the tree $T = T_1$. Lemma 1.3.4 states that t_i is an endpoint of a longest path in T_i, $i = 1,2,\ldots,n-1$. Let $E_1,\ldots,E_m$ be neighbourhood subtrees of T. Lemma 1.3.5 states that $E_j \cap T_i$ is a neighbourhood subtree, $j = 1,\ldots,m$, $i = 1,\ldots,n$. Define $\sigma\colon V \to \{1,\ldots,n\}$ by $\sigma(t_i) = i$, $i = 1,2,\ldots,n$. Combining Lemmas 1.3.4, 1.3.5 and Theorem 1.3.2 we find that σ has the property that all subtrees E_j containing t_i with $\sigma(t_i) = i$ can be totally ordered by inclusion when restricted to T_i, i.e., restricted to all vertices w with $\sigma(w) \geq i$. We call a mapping σ with this property a *nest ordering with respect to* $E_1,\ldots,E_m$.

LEMMA 1.3.4. *Let* v *be a fixed vertex and let* t_1 *be a vertex at largest distance from* v. *Then* t_1 *is an endpoint of a longest path.*

PROOF. Define t_2 to be the point at largest distance from t_1. We shall prove that $P(t_1,t_2)$ is a longest path. Let x be the point on $P(t_1,t_2)$ at equal distance from t_1 and t_2. We shall prove that for all $y \in T$, $d(y,x) \leq \frac{1}{2}d(t_1,t_2) = d(x,t_1) = d(x,t_2)$. Then for all $s_1,s_2 \in T$ we have $d(s_1,s_2) \leq d(s_1,x) + d(x,s_2) \leq d(t_1,t_2)$ which proves that $P(t_1,t_2)$ is a longest path.

Let $y \in T$. We have $x \in P(y,t_1)$ or $x \in P(y,t_2)$. If $x \in P(y,t_1)$, then

$$\begin{aligned} d(y,x) &= d(y,t_1) - d(t_1,x) \\ &\leq d(t_1,t_2) - d(t_1,x) \qquad \text{(by definition of } t_2\text{)} \\ &\leq d(x,t_2). \end{aligned}$$

If $x \in P(y,t_2)$ and $x \notin P(y,t_1)$, then, since t_1 is a vertex with largest distance from v we have $x \in P(v,t_1)$ and $d(v,y) \leq d(v,t_1)$. From $d(v,t_1) = d(v,x) + d(x,t_1)$ and $d(v,y) = d(v,x) + d(x,y)$ it follows that $d(x,y) \leq d(x,t_1)$. □

A (0,1)-matrix is *totally-balanced* (LOVÁSZ [1979]) if it does not contain a square submatrix of size at least three which has no identical columns, and its row and column sums equal to two. Let $E_1,E_2,\ldots,E_m$ be neighbourhood subtrees of T. Define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $v_i \in E_j$. We will show in Theorem 1.3.6 that A is totally-balanced. This result was first proved by GILES [1978]. TAMIR [1980] proved the following generalization. Let $R_1,R_2,\ldots,R_n$, $Q_1,Q_2,\ldots,Q_m$ be neighbourhood subtrees of T. Define the (0,1)-matrix $B = (b_{ij})$ by $b_{ij} = 1$ if and only if $R_i \cap Q_j \neq \emptyset$. Then B is totally-balanced. Taking R_i to be a single point gives us Theorem 1.3.6. To prove this theorem we need a preliminary lemma.

LEMMA 1.3.5. *Let* $E = \{y \in T \mid d(y,x_1) \leq r_1\}$ *be a neighbourhood subtree of* T *and let* $T_1,T_2,\ldots,T_n$ *be the subtrees defined previously to Lemma* 1.3.4 *Then* $E \cap T_i$ *is a neighbourhood subtree* $(i = 1,2,\ldots,n)$.

PROOF. We use induction on n. In case n = 1 the lemma is true. Suppose $E \cap T_i = \{y \in T_i \mid d(y,x_i) \leq r_i\}$ with $x_i \in T_i$ $(1 \leq i \leq n)$. Let $a(t_i)$ be the vertex in T_i adjacent to t_i. Define $x_{i+1} \in T_{i+1}$ and r_{i+1} as follows. If $x_i \in (t_i,a(t_i))$, then $x_{i+1} = a(t_i)$ and $r_{i+1} = r_i - d(x_i,a(t_i))$, else $x_{i+1} = x_i$ and $r_{i+1} = r_i$. It follows that $E \cap T_{i+1} = \{y \in T_{i+1} \mid d(y,x_{i+1}) \leq r_{i+1}\}$. □

THEOREM 1.3.6. Let $E_1,E_2,\ldots,E_m$ be neighbourhood subtrees of T. Define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $v_i \in E_j$. Then A is totally-balanced.

PROOF. The total balancedness of the matrix is not affected if we reorder the rows of the matrix. Suppose the rows of the matrix are ordered such that row i corresponds to t_i as defined above. Let A_1 be a square submatrix of size $k \geq 3$ which has no identical columns, and its row and column sums equal to two. Let row i be the first row of A_1 and let E_j and E_k be the neighbourhood subtrees corresponding to the two columns of A_1 which have a one in row i. Since by Lemma 1.3.4 t_i is an endpoint of a longest path in T_i and, by Lemma 1.3.5, $(E_j \cap T_i)$ and $(E_k \cap T_i)$ are neighbourhood subtrees Theorem 1.3.2 implies that $(E_j \cap T_i) \subseteq (E_k \cap T_i)$ or $(E_k \cap T_i) \subseteq (E_j \cap T_i)$. Without loss of generality assume $(E_j \cap T_i) \subseteq (E_k \cap T_i)$. Since A_1 has no identical columns it follows that the column corresponding to E_k contains at least three ones in the rows of A_1, which contradicts the assumption about A_1. Hence there is no square submatrix of size $k \geq 3$ which has no identical columns, and its row and column sums equal to two. □

Totally-balanced matrices belong to the class of balanced matrices. A (0,1)-matrix is *balanced* if it does not contain an odd square submatrix of size $k \geq 3$ with all row and column sums equal to two. Balanced matrices have been studied by BERGE [1972] and FULKERSON et al. [1974]. Let A be an n×m balanced matrix and let c_j $(j = 1,2,\ldots,m)$ be nonnegative integers. Consider the following *set covering problem:*

$$\text{(SCP)} \qquad \text{minimize} \sum_{j=1}^{m} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1,2,\ldots,n,$$

$$x_j \in \{0,1\}, \qquad j = 1,2,\ldots,m.$$

It is well known that (SCP) when solved as a linear programming problem gives an integral solution (BERGE [1972]), FULKERSON et al. [1974]). Due to KHACHIAN [1979] we know that linear programming problems can be solved in polynomial time. In Chapter 2 we give an efficient algorithm to solve the set covering problem on a totally-balanced matrix.

Given a tree $T = (V,E)$. Let $D \subseteq V$ be a set of demand points and $S \subseteq V$ a set of supply points. With each supply point v_j we associate a distance r_j and a cost c_j. Supply point v_j can serve demand points which are located within distance r_j from v_j, and using (opening) supply point v_j yields a cost of c_j. The *minimum cost covering problem* is to serve all clients at minimum cost. Define the (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $d(v_i,v_j) \leq r_j$. Then the minimum cost covering problem can be formulated as a set covering problem on the totally-balanced matrix A. We can apply the technique of the next chapter to solve this problem.

CHAPTER 2

TOTALLY-BALANCED MATRICES

In this chapter we investigate problems involving totally-balanced matrices. In Section 1.3 we already showed how the minimum cost covering problem on a tree can be formulated as a set covering problem on a totally-balanced matrix. A (0,1)-matrix $A = (a_{ij})$ is in *standard form* if $a_{ik} = a_{i\ell} = a_{jk} = 1$ implies that $a_{j\ell} = 1$ for all i,j,k,ℓ with $i < j$, $k < \ell$. We emphasize that the terminology used does not imply that every (0,1)-matrix can be transformed into standard form. As a matter of fact we shall prove in Section 2.2 that a matrix can be transformed into standard form if and only if the matrix is totally-balanced. In Section 2.1 we solve the set covering problem on a matrix in standard form. In Section 2.2 we present polynomial-time algorithms to transform a totally-balanced matrix into standard form as well as to recognize a totally-balanced matrix. Polynomial-time algorithms can also be obtained from the result of BROUWER & KOLEN [1980] which we prove in Section 2.3. These algorithms differ from the ones discussed in Section 2.2 in that they are less efficient. In Section 2.3 we also discuss the relation between totally-balanced matrices and chordal bipartite graphs. In Section 2.4 we solve the simple plant location problem on a tree. Throughout this chapter we discuss the relation between our work and work done independently by HOFFMAN & SAKAROVITCH [1981].

2.1. The set covering problem on a matrix in standard form

In this section we assume that the underlying matrix of the set covering problem we investigate is in standard form. Let $A = (a_{ij})$ be an $n \times m$ (0,1)-matrix in standard form. The *set covering problem* is defined by

$$\text{(SCP)} \qquad \text{minimize} \sum_{j=1}^{m} c_j x_j$$

$$\text{subject to} \sum_{j=1}^{m} a_{ij} x_j \geq 1, \qquad i = 1,2,\ldots,n,$$

$$x_j \in \{0,1\}, \; j = 1,2,\ldots,m,$$

where c_j $(j = 1,2,\dots,m)$ are nonnegative integers. The *dual problem* of (SCP) is defined by

$$\text{(DSCP)} \qquad \text{maximize} \sum_{i=1}^{n} y_i$$

$$\text{subject to} \sum_{i=1}^{n} y_i a_{ij} \le c_j, \quad j = 1,2,\dots,m,$$

$$y_i \ge 0, \quad i = 1,2,\dots,n.$$

We shall construct a feasible dual solution y and feasible primal solution x such that when $I = \{i \mid y_i > 0\}$ and $J = \{j \mid x_j = 1\}$ the following complementary slackness relations hold:

$$\text{(2.1.1)} \qquad \sum_{i=1}^{n} y_i a_{ij} = c_j \text{ for all } j \in J,$$

(2.1.2) for all $i \in I$ there is at most one $j \in J$ such that $a_{ij} = 1$.

Note that since x is assumed to be a feasible solution we can replace at most one by exactly one in (2.1.2). It follows from (2.1.1) and (2.1.2) that the value of the primal and dual solutions are equal. Therefore both solutions are optimal.

The feasible dual solution is constructed by a greedy approach. The value of y_i is determined according to increasing index and taken to be as large as possible. This procedure is formulated in the Dual Algorithm:

<u>DUAL ALGORITHM</u>.

```
         for i := 1 step 1 to n
(2.1.3)  do $y_i := \min_{j:a_{ij}=1} [c_j - \sum_{k=1}^{i-1} y_k a_{kj}]$ od. □
```

<u>EXAMPLE 2.1.4</u>. The matrix of the example as well as the costs and results of the Dual Algorithm are given in Figure 2.1.5.

j =	1	2	3	4	5	6	7	
i = 1	1	1	0	0	0	0	0	$y_1 = 2$
2	1	1	0	0	0	0	0	$y_2 = 0$
3	1	1	0	0	1	0	0	$y_3 = 0$
4	0	0	1	0	0	1	0	$y_4 = 2$
5	0	0	1	0	0	1	0	$y_5 = 0$
6	0	0	0	1	0	0	1	$y_6 = 1$
7	0	1	0	0	1	0	1	$y_7 = 1$
8	0	0	1	0	0	1	1	$y_8 = 0$
9	0	0	0	1	1	1	1	$y_9 = 0$
	2	3	2	1	2	3	2	cost c_j
	0	1	2	1	2	3	2	cost after subtracting y_1
	0	1	0	1	2	1	2	" " " $y_1 + y_4$
	0	1	0	0	2	1	1	" " " $y_1 + y_4 + y_6$
	0	0	0	0	1	1	0	" " " $y_1 + y_4 + y_6 + y_7$

Figure 2.1.5. Illustration of the Dual Algorithm. □

The Primal Algorithm has as input the set $I = \{i | y_i > 0\}$ and the matrix A and as output a subset of columns J which will define an optimal primal solution. We say that a row i is *covered by* column j in the matrix $A = (a_{ij})$ if and only if $a_{ij} = 1$.

PRIMAL ALGORITHM.

```
Delete all columns which do not correspond to a tight constraint; J := ∅;
while there are still columns left
do add last column to J and delete all columns from the matrix that
    cover a row i ∈ I which is also covered by the chosen last column
od. □
```

EXAMPLE 2.1.6. Apply the Primal Algorithm to the results of Example 2.1.4. We have $I = \{1,4,6,7\}$. The Primal Algorithm starts by deleting the columns 5 and 6 which do not correspond to a tight constraint, $J := \emptyset$.

Iteration 1: $J := \{7\}$, delete column 4 (columns 4 and 7 cover row $6 \in I$) and column 2 (columns 2 and 7 cover row $7 \in I$).

Iteration 2: $J := \{7,3\}$.

Iteration 3: $J := \{7,3,1\}$.

The value of the primal solution defined by $x_j = 1$ if and only if $j \in J$ is equal to the value of the dual solution, namely 6. □

With respect to the set I constructed by the Dual Algorithm we call a column k a *blocking column* for row i if row i is covered by column k and for all rows j $(j > i)$ which are covered by column k we have $j \notin I$. If $j(i)$ denotes the index of the constraint for which the minimum is attained in (2.1.3) during iteration i of the Dual Algorithm, then constraint $j(i)$ is a tight constraint and column $j(i)$ is a blocking column for row i.

THEOREM 2.1.7. *The* (0,1)-*solution* x *defined by* $x_j = 1$ *if and only if* $j \in J$ *is an optimal primal solution.*

PROOF. It is clear that the feasible dual solution y and the primal solution x satisfy (2.1.1) and (2.1.2). So in order to show that x is an optimal primal solution we have to show that x is feasible, i.e., the set of columns J cover all rows of the matrix. We shall prove this by induction on the number of columns of the matrix. The induction hypothesis is that all rows for which there is a blocking column are covered by the set of columns found by the algorithm. Note that at the beginning of the while statement in the Primal Algorithm each row is covered by a blocking column. Let ℓ be the last column and delete all columns that cover a row $i \in I$ which is also covered by column ℓ. We shall prove that for a row j which is not covered by column ℓ none of the deleted columns is a blocking column. This observation and the induction hypothesis then prove that J covers all rows. Suppose row j is covered by column k but not by column ℓ and columns k and ℓ both cover a row $i \in I$. If $j > i$, then since the matrix is in standard form this would imply that row j is covered by column ℓ. Therefore $j < i$. Since $i \in I$ it follows that column k is not a blocking column for row j. □

Both the Dual and Primal Algorithm can be executed in $O(nm)$ time.

We now discuss the relation with the work of HOFFMAN & SAKAROVITCH [1981]. Call a (0,1)-matrix *box-greedy* if, for every c, successive maximization of y_1, then $y_2, \ldots$ on the polytope $\{y \mid Ay \le c,\ y \ge 0\}$ simultaneously maximizes $y_1, y_1+y_2, y_1+y_2+y_3$, etc. HOFFMAN & SAKAROVITCH [1981] showed that A is Box-greedy if and only if the rows of A can be permuted so that A is in standard form.

2.2. Transforming a totally-balanced matrix into standard form

In this section we prove that a matrix can be transformed into standard form if and only if the matrix is totally-balanced. We present polynomial-time algorithms to transform a totally-balanced matrix into standard form as well as to recognize a totally-balanced matrix.

Let $A = (a_{ij})$ be an $n \times m$ (0,1)-matrix. We consider column j of A as a subset of rows, namely those rows which are covered by column j. Let us denote column j by $E_j = \{i | a_{ij} = 1\}$. A permutation σ: $\{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ is a *nest ordering with respect to* $E_1, E_2, \ldots, E_m$ if for all $i = 1,2,\ldots,n$, all columns covering the row j defined by $\sigma(j) = i$ can be totally ordered by inclusion when restricted to the rows k of the matrix with $\sigma(k) \geq i$. By a *lexicographic ordering* of subsets $E_1, E_2, \ldots, E_m$ of $\{1,2,\ldots,n\}$ the following is meant. With each set E we associate a vector b_E of length $|E|$. The first component of b_E is the largest element of E, the second component is the second largest element, and so on. E_i is lexicographically smaller than E_j if b_{E_i} is lexicographically smaller then b_{E_j}. Ties are broken arbitrarily (they only occur when two subsets contain the same elements). The following lemma shows why the definitions of a nest and lexicographic ordering are useful.

LEMMA 2.2.1. *Let $A = (a_{ij})$ be a matrix such that the rows form a nest ordering with respect to the columns, and the columns are ordered in lexicographically increasing order. Then the matrix A is in standard form.*

PROOF. Suppose $a_{ik} = a_{i\ell} = a_{jk} = 1$, $i < j$, $k < \ell$. Define $E_f^i = E_f \setminus \{1,2,\ldots,i-1\}$ ($f = 1,2,\ldots,m$). Since $i \in E_k \cap E_\ell$ and the rows form a nest ordering, we have either $E_k^i \subseteq E_\ell^i$ or $E_\ell^i \subseteq E_k^i$. Since the columns are in lexicographically increasing order we have $E_k^i \subseteq E_\ell^i$. Hence $a_{jk} = 1$ implies that $a_{j\ell} = 1$. □

It was shown in Section 1.3 that if the rows of the matrix correspond to the vertices $v_1, v_2, \ldots, v_n$ of a tree and the columns correspond to neighbourhood subtrees, then there exists a nest ordering. This nest ordering σ can be found by fixing a vertex v, ordering the vertices according to nonincreasing distances to v, say $d(t_1,v) \geq d(t_2,v) \geq \ldots \geq d(t_n,v)$, where $t_n = v$, and setting the σ-value of the row corresponding to t_i equal to i for $i = 1,2,\ldots,n$.

Ordering m sets of at most n elements lexicographically requires $O(mn)$ time. It follows that a matrix A with rows corresponding to the vertices of a tree and columns corresponding to neighbourhood subtrees can be transformed into standard form in $O(\max(n \log n, nm))$ time.

In case of an arbitrary totally-balanced matrix it was shown by BROUWER & KOLEN [1980] that there is a row of a totally-balanced matrix such that all columns covering this row can be totally ordered by inclusion. By inspection we can find this row in $O(nm^2)$ time. Give this row number 1 and delete the row from the matrix. Let A_i be the matrix we get from A by deleting the rows with number $1,2,\ldots,i$ from the matrix A. Then since A_i is still totally-balanced there is a row with the property that all columns covering this row in A_i can be totally ordered by inclusion. Give this row number $i+1$ $(i = 1,2,\ldots,n-1)$. In this way we find a number for each row in $O(n^2m^2)$ time. Clearly this mapping is a nest ordering. This result shows that we can transform an $n \times m$ totally-balanced matrix into standard form in $O(n^2m^2)$ time. Instead of proving this result we shall describe an algorithm which transforms an $n \times m$ totally-balanced matrix into standard form in $O(nm^2)$ time. Since a matrix is in standard form if and only if its transpose is in standard form the algorithm can be used to transform an $n \times m$ totally-balanced matrix into standard form in $O((\min\{n,m\})^2\max\{n,m\})$ time.

The algorithm we will describe for transforming an $n \times m$ totally-balanced matrix into standard form constructs a mapping $\sigma:\{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ corresponding to a transformation of the rows ($\sigma(i) = j$ denotes that row i will be row j in the transformed matrix) and a mapping τ: $\{E_1,E_2,\ldots,E_m\} \to \{1,2,\ldots,m\}$ corresponding to a transformation of the columns ($\tau(E_i) = j$ denotes that column i will be column j in the transformed matrix). The mapping σ will be a nest ordering with respect to the columns $E_1,E_2,\ldots,E_m$. The mapping τ will be a lexicographic ordering with respect to the matrix A where the rows have been reordered according to σ. Lemma 2.2.1 states that the matrix is in standard form after it has been transformed according to σ and τ.

Let us now describe the algorithm. The algorithm consists of m iterations. In iteration i we select the column E for which we define $\tau(E) = m-i+1$

($1 \le i \le m$). We say that a column E has been *determined* if $\tau(E)$ has been defined. At the beginning of each iteration the rows are partitioned into a number of groups, say $G_r,\ldots,G_1$. If $i < j$, then for all $k \in G_i$ and $\ell \in G_j$ we have $\sigma(k) < \sigma(\ell)$, i.e., rows belonging to G_i precede rows belonging to G_j in the transformed matrix. Rows b and c occur in the same group G at the beginning of iteration i if and only if for all columns E we have determined so far, i.e., all columns E for which $\tau(E) \ge m-i+2$, we cannot distinguish between the rows b and c, i.e., $b \in E$ if and only if $c \in E$. At the beginning of iteration 1 all rows occur in the same group. Let $G_r,\ldots,$ G_1 be the partitioning into groups at the beginning of iteration i ($1 \le i \le m$). For each column E not yet determined we calculate the vector d_E of length r, where $d_E(j) = |G_{r-j+1} \cap E|$ ($j = 1,2,\ldots,r$). A column E for which d_E is a lexicographically largest vector is the column determined in iteration i and $\tau(E) = m-i+1$. Ties can be broken arbitrarily. After we have determined E we can distinguish between some elements in the same group G if $1 \le |G \cap E| < |G|$. If this is the case we shall take rows in $G \backslash E$ to precede rows in $G \cap E$ in the transformed matrix. This can be expressed by adjusting the partitioning into groups in the following way. For $j = r,$ $r-1,\ldots,1$ respectively we check if the intersection of G_j and E is nonempty and not equal to G_j. If this is the case we increase the index of all groups with index greater than j by one and partition the group G_j into two groups called G_j and G_{j+1} with $G_{j+1} = G_j \cap E$ and $G_j = G_j \backslash E$. The algorithm ends after m iterations with a partitioning into groups, say $G_r,\ldots,G_1$. The permutation σ has to satisfy $\sigma(i) < \sigma(j)$ if $i \in G_k$, $j \in G_\ell$ and $k < \ell$, i.e., for $i = 1,2,\ldots,r$ we assign the values $\Sigma_{j=1}^{i-1} |G_j|+1,\ldots,\Sigma_{j=1}^{i} |G_j|$ in an arbitrary way to the elements in group G_i. The number of calculations at each iteration is $O(mn)$. Therefore the overall time complexity of the algorithm is $O(nm^2)$.

EXAMPLE 2.2.2. The 9×7 (0,1)-matrix A is given by its columns $E_1 = \{1,2,3\}$, $E_2 = \{1,2,3,5\}$, $E_3 = \{4,5\}$, $E_4 = \{3,4,5,9\}$, $E_5 = \{5,8,9\}$, $E_6 = \{6,7,8,9\}$, $E_7 = \{6,7,8\}$.

Iteration 1: $G_1 = (1,2,3,4,5,6,7,8,9)$.

$d_{E_i} = (|E_i|)$, choose E_4, $\tau(E_4) = 7$.

Iteration 2: $G_2 = (3,4,5,9)$, $G_1 = (1,2,6,7,8)$.

E	E_1	E_2	E_3	E_5	E_6	E_7
d_E	(1,2)	(2,2)	(2,0)	(2,1)	(1,3)	(0,3)

Choose E_2, $\tau(E_2) = 6$.

Iteration 3: $G_4 = (3,5)$, $G_3 = (4,9)$, $G_2 = (1,2)$, $G_1 = (6,7,8)$.

E	E_1	E_3	E_5	E_6	E_7
d_E	(1,0,2,0)	(1,1,0,0)	(1,1,0,1)	(0,1,0,3)	(0,0,0,3)

Choose E_5, $\tau(E_5) = 5$.

Iteration 4: $G_7=(5)$, $G_6=(3)$, $G_5=(9)$, $G_4=(4)$, $G_3=(1,2)$, $G_2=(8)$, $G_1=(6,7)$.

E	d_E
E_1	(0,1,0,0,2,0,0)
E_3	(1,0,0,1,0,0,0)
E_6	(0,0,1,0,0,1,2)
E_7	(0,0,0,0,0,1,2)

Choose E_3, $\tau(E_3) = 4$.

From now on the groups do not change. Therefore $\tau(E_1) = 3$, $\tau(E_6) = 2$, $\tau(E_7) = 1$. A mapping σ is given by σ: $(6,7,8,1,2,4,9,3,5) \to (1,2,3,4,5,6,7,8,9)$. The mapping τ is given by $\tau: (E_7,E_6,E_1,E_3,E_5,E_2,E_4) \to (1,2,3,4,5,6,7)$. The transformed matrix is the one used in Example 2.1.4. □

We now proceed to prove that the mapping σ constructed by the algorithm applied to a totally-balanced matrix is a nest ordering with respect to the columns and the mapping τ is a lexicographic ordering of the columns where the rows of the matrix have been reordered according to σ. Let E_1,E_2 be two columns. We call E_1,E_2 *comparable* if $E_1 \subseteq E_2$ or $E_2 \subseteq E_1$. E_1,E_2 are *incomparable* if they are not comparable. The mappings σ and τ constructed by the algorithm have the following properties.

<u>PROPERTY 2.2.3</u>. Let E_1,E_2 be two incomparable columns such that $\tau(E_1) < \tau(E_2)$, let $i_1 \in E_1 \backslash E_2$, and let i_2 be the largest row with respect to σ in $E_2 \backslash E_1$, i.e., there is no $i \in E_2\backslash E_1$ such that $\sigma(i) > \sigma(i_2)$. Then $\sigma(i_2) > \sigma(i_1)$.

<u>PROOF</u>. Consider the iteration in which E_2 was determined. Let $G_r,\ldots,G_1$ be the partitioning into groups at the beginning of this iteration and let k be the largest index for which $G_k \cap E_1 \neq G_k \cap E_2$. Then $i_2 \in G_k$. If $i_1 \in G_j$ with $j < k$, then $\sigma(i_1) < \sigma(i_2)$. If $i_1 \in G_k$, then after E_2 is determined the group G_k is partitioned into two groups $G_k \cap E_2$ and $G_k \backslash E_2$ where rows in $G_k\backslash E_2$ precede rows in $G_k \cap E_2$. Since $i_2 \in G_k \cap E_2$ and $i_1 \in G_k \backslash E_2$ we have $\sigma(i_1) < \sigma(i_2)$. □

<u>COROLLARY 2.2.4</u>. *The mapping τ is a lexicographic ordering of the columns of the matrix where the rows are reordered according to σ.*

<u>PROOF</u>. Let E_1, E_2 be two columns such that $\tau(E_1) < \tau(E_2)$. If E_1 and E_2 are comparable, then $E_1 \subseteq E_2$. If E_1 and E_2 are incomparable, then according to Property 2.2.3 the largest row with respect to σ in $E_2 \setminus E_1$ is greater than any row in $E_1 \setminus E_2$, i.e., E_2 is lexicographically greater than E_1. □

A column E_1 *separates* i *from* j if $i \in E_1$, $j \notin E_1$ and for all columns E_2 with $\tau(E_2) > \tau(E_1)$ we have $i \in E_2$ if and only if $j \in E_2$.

<u>PROPERTY 2.2.5</u>. If $\sigma(i) > \sigma(j)$ and i,j are not covered by the same columns, then there is a column E_1 which separates i from j.

<u>PROOF</u>. Let E_1 be the column such that we can distinguish between i and j, i.e. $i \in E_1$, $j \notin E_1$ or $j \in E_1$ and $i \notin E_1$, and for all columns E_2 with $\tau(E_2) > \tau(E_1)$ we cannot distinguish between i and j. At the beginning of the iteration in which E_1 is determined there is a group G which contains both i and j. After E_1 is determined this group is partitioned into $G \cap E_1$ and $G \setminus E_1$. Since $\sigma(i) > \sigma(j)$ we have $i \in G \cap E_1$ and $j \in G \setminus E_1$, i.e. E_1 separates i from j. □

<u>PROPERTY 2.2.6</u>. Let i_0 be the row defined by $\sigma(i_0) = 1$. Let $G_r, \ldots, G_1$ be the partitioning into groups at the beginning of the iteration in which the column E was determined. If $i_0 \in E$, then $G_1 \cap E = G_1$.

<u>PROOF</u>. Let $i_0 \in E$. If $G_1 \cap E \neq G_1$, then since $i_0 \in G_1 \cap E$ we have that group G_1 is partitioned into $G_1 \cap E$ and $G_1 \setminus E$. This would mean that $i_0 \in G_2$ at the end of the iteration, which contradicts that $\sigma(i_0) = 1$. □

We still have to prove that the mapping σ constructed by the algorithm applied to a totally-balanced matrix is a nest ordering with respect to the columns. The proof proceeds by induction on the number of rows. Assume that the mapping constructed by the algorithm applied to a totally-balanced matrix with less than n rows is a nest ordering with respect to the columns (Note that this is true for n=2). Apply the algorithm to the $n \times m$ totally-balanced matrix A. Let σ be the mapping constructed by the algorithm applied to A, let i_0 be the row defined by $\sigma(i_0) = 1$ and let A_1 be the matrix we get from A by deleting row i_0. We shall prove that

(2.2.7) there exists a mapping σ_1 constructed by the algorithm applied to A_1 such that $\sigma_1(i) = \sigma(i)-1$ for all $i \neq i_0$, and

(2.2.8) all columns covering row i_0 can be totally ordered by inclusion.

Since by the induction hypothesis σ_1 is a nest ordering with respect to the columns of A_1 we have from (2.2.7) and (2.2.8) that σ is a nest ordering with respect to the columns of the matrix A.

Note that (2.2.8) holds in Example 2.2.2 since all columns covering row 6 with $\sigma(6) = 1$ can be totally ordered by inclusion. In Example 2.2.9 we will apply the algorithm to the matrix A of Example 2.2.2 with row 6 deleted.

EXAMPLE 2.2.9. The 8×7 (0,1)-matrix A_1 is given by $E_1 = \{1,2,3\}$, $E_2 = \{1,2,3,5\}$, $E_3 = \{4,5\}$, $E_4 = \{3,4,5,9\}$, $E_5 = \{5,8,9\}$, $E_6 = \{7,8,9\}$, $E_7 = \{7,8\}$.

Iteration 1: $G_1 = (1,2,3,4,5,7,8,9)$.

$d_{E_i} = (|E_i|)$, choose E_4, $\tau_1(E_4) = 7$.

Iteration 2: $G_2 = (3,4,5,9)$, $G_1 = (1,2,7,8)$.

E	E_1	E_2	E_3	E_5	E_6	E_7
d_E	(1,2)	(2,2)	(2,0)	(2,1)	(1,2)	(0,2)

Choose E_2, $\tau_1(E_2) = 6$.

Iteration 3: $G_4 = (3,5)$, $G_3 = (4,9)$, $G_2 = (1,2)$, $G_1 = (7,8)$.

E	E_1	E_3	E_5	E_6	E_7
d_E	(1,0,2,0)	(1,1,0,0)	(1,1,0,1)	(0,1,0,2)	(0,0,0,2)

Choose E_5, $\tau_1(E_5) = 5$.

Iteration 4: $G_7 = (5), G_6 = (3), G_5 = (9), G_4 = (4), G_3 = (1,2), G_2 = (8), G_1 = (7)$.

E	d_E
E_1	(0,1,0,0,2,0,0)
E_3	(1,0,0,1,0,0,0)
E_6	(0,0,1,0,0,1,1)
E_7	(0,0,0,0,0,1,1)

Choose E_3, $\tau_1(E_3) = 4$.

From now on the groups do not change. Therefore $\tau_1(E_1) = 3$, $\tau_1(E_6) = 2$ and $\tau_1(E_7) = 1$. The mapping τ_1 is given by $\tau_1 : (E_7,E_6,E_1,E_3,E_5,E_2,E_4) \to (1,2,3,4,5,6,7)$. A mapping σ_1 is given by σ_1: $(7,8,1,2,4,9,3,5) \to (1,2,3,4,5,6,7,8)$. □

Note that $\sigma_1(i) = \sigma(i)-1$ for all $i \neq 6$ so that (2.2.7) holds for the matrix A of Example 2.2.2.

Let us first prove (2.2.7). We shall prove that statement S(i) holds for all $i = 0,1,\dots,m$, where S(i) is given by:

at each of the first i iterations of the algorithm applied to both A and A_1 we have determined the same column; if at the beginning of iteration i+1 $G_r,\dots,G_2,G_1$ is the partitioning into groups of the algorithm applied to A, then the partitioning into groups of the algorithm applied to A_1 is given by $G_r,\dots,G_2,G_1\backslash\{i_0\}$ if $|G_1| > 1$, and else by $G_r,\dots,G_2$.

Note that in Example 2.2.9 S(i) holds for all $i = 0,1,\dots,m$. We shall prove that if S(i-1) holds, then also S(i) holds ($i = 1,2,\dots,m$). Since S(0) holds we conclude that S(m) holds and we can take $\sigma_1(i) = \sigma(i)-1$ for all $i \neq i_0$. Let S(i-1) hold and let E_1 be the column determined in iteration i of the algorithm applied to A. Assume we can choose E_1 to be the column determined in iteration i of the algorithm applied to A_1. When we calculate the new partitioning into groups each group G_i ($i\geq 2$) is partitioned in exactly the same way. In case $|G_1| = 1$ this shows that S(i) holds. We still have to consider the case that $|G_1| > 1$. If $i_0 \in E_1$, then according to Property 2.2.6 $G_1 \cap E_1 = G_1$ so that $G_1(G_1\backslash\{i_0\})$ remain unchanged in the algorithm applied to $A(A_1)$. If $i_0 \notin E_1$ and $G_1 \cap E_1 \neq \emptyset$, then $G_1(G_1\backslash\{i_0\})$ is partitioned into $G_1 \cap E_1$ $((G_1\backslash\{i_0\}) \cap E_1)$ and $G_1\backslash E_1$ $((G_1\backslash\{i_0\})\backslash E_1)$ in the algorithm applied to $A(A_1)$. Since $G_1 \cap E_1 = (G_1\backslash\{i_0\}) \cap E_1$ and $(G_1\backslash\{i_0\})\backslash E_1 = (G_1\backslash E_1)\backslash\{i_0\}$ we have that S(i) holds. In order to show that we can choose E_1 to be the column we determine in iteration i of the algorithm applied to A_1 we have to prove that the vector d_{E_1} associated with the partitioning into groups is lexicographically greater than or equal to d_{E_2} for all E_2 not yet determined. If $|G_1| = 1$, then this follows from the choice of E_1. If $|G_1| > 1$, then it is sufficient to prove that

$$\text{(2.2.10)} \qquad \text{for all } E_2\left[|G_1\cap E_1| \geq |G_1\cap E_2| \text{ implies that } |(G_1\backslash\{i_0\})\cap E_1| \geq \right.$$
$$\left. \geq |(G_1\backslash\{i_0\})\cap E_2|\right].$$

If $i_0 \in E_1$, then according to Property 2.2.6 $G_1 \cap E_1 = G_1$ so that (2.2.10) holds. If $i_0 \notin E_1$, then $|(G_1\backslash\{i_0\})\cap E_1| = |G_1\cap E_1| \geq |G_1\cap E_2| \geq |(G_1\backslash\{i_0\})\cap E_2|$.

This completes the proof of (2.2.7).

Let us turn to the proof of (2.2.8). Suppose there are two incomparable columns E_1 and E_2 covering i_0. Without loss of generality assume $\tau(E_1) < \tau(E_2)$. Let $i_1 (i_2)$ be the largest row with respect to σ in $E_1 \backslash E_2$ $(E_2 \backslash E_1)$. According to Property 2.2.3 we have $\sigma(i_1) < \sigma(i_2)$. We call $(i_0, i_1, \ldots, i_m, E_1, E_2, \ldots, E_m)$ an *m-chain* $(m \geq 2)$ if

(1) $i_j \in E_k \Leftrightarrow j=k$ or $j=k-2$ $(k=1,2,\ldots,m)$, where $i_{-1}=i_0$,

(2) $\sigma(i_{j+1}) > \sigma(i_j)$ $(j = 0,1,\ldots,m-1)$,

(3) $\tau(E_{j+1}) > \tau(E_j)$ $(j = 1,2,\ldots,m-1)$,

(4) E_j separates i_{j-2} from i_{j-3} $(j = 3,\ldots,m)$,

(5) i_j is the largest row with respect to σ in $E_j \backslash E_{j-1}$ $(j=1,2,\ldots,m)$, where $E_0 = E_2$.

Note that under the assumption made (i.e., (2.2.8) does not hold) a 2-chain exists. We shall prove in the next theorem that we can extend an m-chain to an m+1-chain. Since the number of rows of the matrix is finite this leads to a contradiction. We conclude that there does not exist a 2-chain, i.e., all columns covering i_0 can be totally ordered by inclusion. Figure 2.2.11 gives a schematic representation of a 4-chain

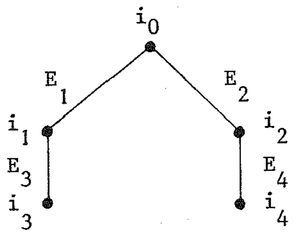

Figure 2.2.11. Schematic representation of a 4-chain.

THEOREM 2.2.12. *An* m-*chain can be extended to an* m+1-*chain* $(m \geq 2)$.

PROOF. Since $\sigma(i_{m-1}) > \sigma(i_{m-2})$ and i_{m-1}, i_{m-2} are not covered by the same columns $(i_{m-1} \notin E_m)$ it follows from Property 2.2.5 that there exists a column E_{m+1} which separates i_{m-1} from i_{m-2}. Note that $i_{m-2} \notin E_{m+1}$ and $\tau(E_{m+1}) > \tau(E_m)$. Repeating the following argument for $k = m, m-1, \ldots, 3$ respectively proves that $i_{m-2}, i_{m-3}, \ldots, i_0 \notin E_{m+1}$: E_k separates i_{k-2} from i_{k-3}, $i_{k-2} \notin E_{m+1}$ and $\tau(E_{m+1}) > \tau(E_k)$ imply that $i_{k-3} \notin E_{m+1}$. If $i_m \in E_{m+1}$, then the rows $i_0, i_1, \ldots, i_m$ and columns $E_1, E_2, \ldots, E_{m+1}$ define a square submatrix of A of size $m+1 \geq 3$ with no identical columns and with row and column sums equal to two. This contradicts the total balancedness of the matrix A. Hence $i_m \notin E_{m+1}$. Since $i_m \in E_m \backslash E_{m+1}$ and $i_{m-1} \in E_{m+1} \backslash E_m$ we have that E_m and E_{m+1} are incomparable. Let i_{m+1} be the largest row with respect to σ in $E_{m+1} \backslash E_m$. It follows from Property 2.2.3 that $\sigma(i_{m+1}) > \sigma(i_m)$. We claim that the m-chain extended with i_{m+1} and E_{m+1} is an m+1-chain. We have to prove that $i_{m+1} \notin E_k$ for $k = 1,2,\ldots,m$. Assume there is a k such that $i_{m+1} \in E_k$. Since $i_{m+1} \notin E_m$ we can take k to be the largest index such that $i_{m+1} \in E_k$ and $i_{m+1} \notin E_{k+1}$. Since $\sigma(i_{m+1}) > \sigma(i_k)$ and i_k is the largest row with respect to σ in $E_k \backslash E_{k-1}$ it follows that $i_{m+1} \in E_{k-1}$. If $k = 1$, then this contradicts the fact that $i_{m+1} \notin E_2 = E_0$. If $k > 1$, then we have $i_{m+1} \in E_{k-1} \backslash E_{k+1}$, $i_{k+1} \in E_{k+1} \backslash E_{k-1}$ and $i_{k-1} \in E_{k-1} \cap E_{k+1}$ which contradicts the fact that according to (2.2.7) E_{k-1} and E_{k+1} should be comparable when restricted to rows i with $\sigma(i) \geq \sigma(i_{k-1})$. We conclude that $i_{m+1} \notin E_k$ for $k = 1,2,\ldots,m$. □

This completes the correctness proof of the algorithm. Since a matrix in standard form clearly is totally-balanced we have established that a matrix can be transformed into standard form if and only if it is totally-balanced. This result was also known to HOFFMAN & SAKAROVITCH [1981].

The last part of this section is devoted to the question how to recognize a totally-balanced matrix. The following theorem shows how an $n \times m$ totally-balanced matrix ($m \leq n$) can be recognized in $O(nm^2)$ time.

THEOREM 2.2.13. *A* (0,1)*-matrix* A *is totally-balanced if and only if the permutation* σ *constructed by the algorithm is a nest ordering with respect to the columns.*

PROOF. We know already that if A is totally-balanced, then σ is a nest ordering. If A is not totally-balanced, then there is a square submatrix A_1 of size $k \geq 3$ with no identical columns and all row and column sums equal to two. Let i_1 be the smallest row with respect to σ in A_1. Let E_j and E_k be the columns of A_1 covering row i_1, and let $i_2(i_3)$ be the other row of A_1 covered by $E_j(E_k)$. We have $i_2 \in E_j \backslash E_k$ and $i_3 \in E_k \backslash E_j$, i.e. E_j and E_k are incomparable when restricted to rows i with $\sigma(i) \geq \sigma(i_1)$. Hence σ is not a nest ordering. □

One can find σ in $O(nm^2)$ for any matrix A. Checking whether σ is a nest ordering can be done by comparing all columns covering the row j defined by $\sigma(j) = i$ for $i = 1,2,\ldots,n$ respectively. Columns which have been compared because they cover a row j with $\sigma(j) = i$ do not have to be considered again in any other iteration k with $k > i$. So each pair has to be checked at most once. This can be done in $O(nm^2)$ time.

2.3. Existence of a nest row

Let $A = (a_{ij})$ be an $n \times m$ totally-balanced matrix. In this section we shall give a theoretical proof of the fact that there is a row of A such that all columns covering this row can be totally ordered by inclusion. A row with this property is called a *nest row*. The *row intersection graph* $G = (\{1,2,\ldots,n\},F)$ of A is defined by $\{i,j\} \in F$ if and only if $a_{ik} = a_{jk} = 1$ for some column E_k $(1 \le k \le m)$. Note that by definition of total balancedness G is a chordal graph.

LEMMA 2.3.1. *For every clique in the row intersection graph of a totally-balanced matrix* A *there exists a column covering all rows in the clique.*

PROOF. The proof proceeds by induction on the size of the clique. In case of only two rows in the clique the result holds by definition of the row intersection graph. Assume the result holds for all cliques of size at most $k-1$ $(k\ge 3)$. Let $C = \{i_1,i_2,\ldots,i_k\}$ be a clique of size k. By the induction hypothesis there is a column E_j covering all rows i_r, $r = 1,2,\ldots,k$, $r \ne j$, for $j = 1,2,\ldots,k$. If $i_j \in E_j$, then E_j covers all rows in C. So we may assume that $i_j \notin E_j$ for all $j = 1,2,\ldots,k$. But then the 3×3 submatrix A defined by rows i_1,i_2,i_3 and columns E_1,E_2 and E_3 has row and column sums equal to two, which leads to a contradiction. We conclude that there is a column E_j $(1\le j\le k)$ covering all rows in C. □

THEOREM 2.3.2. *Each totally-balanced matrix with more than one row has at least two nest rows. Moreover if the row intersection graph is not complete it has two nest rows which are nonadjacent in this graph.*

PROOF. The proof proceeds by induction on the number of rows of the totally-balanced matrix. In case of two rows the theorem is trivial. Assume the result holds for all totally-balanced matrices with less than n rows and let A be a totally-balanced matrix with n rows $(n\ge 3)$. Let G be the intersection graph of A. If G is complete, then it follows from Lemma 2.3.1 that there is a column which covers all rows. We delete this column from the matrix. Nest rows of the remaining matrix are also nest rows of A. Therefore assume without loss of generality that G is not a clique. Let b,c be two nonadjacent vertices and let S be a minimal vertex separator of b and c with

G_B and G_C being the connected components of $G_{\{1,2,\ldots,n\}\setminus S}$ containing b and c, respectively.

Consider the submatrix $A_{B\cup S}$ of A defined by the rows of $B \cup S$ and all columns covering at least one of those rows. According to Lemma 1.2.4 there is a vertex d in G_C such that $\{d\} \cup S$ induces a clique. According to Lemma 2.3.1 there is a column E which covers the rows belonging to $\{d\} \cup S$. Since S is a minimal vertex separator the column E does not cover a row belonging to B and columns covering a row in B do not cover a row belonging to C. Consider the row intersection graph of $A_{B\cup S}$. If this graph is not complete, then $A_{B\cup S}$ contains two nonadjacent nest rows. At least one of these nest rows does belong to B (S is a clique in the intersection graph of $A_{B\cup S}$ because column E covers each row of S). This row is also a nest row of A. If the intersection graph is complete, then we delete all columns which cover all rows of $A_{B\cup S}$. The row intersection graph of the remaining matrix is not complete and contains two nonadjacent nest rows. At least one of those rows is in B (S is still a clique, because each row in S is covered by E). This row is also a nest row of A. Similarly one proves that there exists a nest row in C. □

A *bipartite graph* $H = (V_1,V_2,E)$ is a graph with vertex set $V = V_1 \cup V_2$ $(V_1 \cap V_2 = \emptyset)$ and edge set $E \subseteq \{\{i,j\} \mid i \in V_1,\ j \in V_2\}$. If $E = \{\{i,j\} \mid i \in V_1,\ j \in V_2\}$, then the bipartite graph is *complete*. A *chordal bipartite graph* is a bipartite graph for which every cycle of length strictly greater than four has a chord. Chordal bipartite graphs and totally-balanced matrices are equivalent in the following sense:

(2.3.3) Given a chordal bipartite graph $H = (\{1,2,\ldots,n\},\{1,2,\ldots,m\},E)$ define the $n \times m$ (0,1)-matrix $A = (a_{ij})$ by $a_{ij} = 1$ if and only if $\{i,j\} \in E$. Then A is totally-balanced. Given an $n \times m$ totally-balanced matrix $A = (a_{ij})$ define the bipartite graph $H = (\{1,2,\ldots,n\},\{1,2,\ldots,m\},E)$ by $E = \{\{i,j\} \mid a_{ij} = 1\}$. Then H is a chordal bipartite graph.

Chordal bipartite graphs were studied by GOLUMBIC [1980] in relation with perfect Gaussian elimination for nonsymmetric matrices. An edge $\{i,j\}$ of a bipartite graph is *bisimplicial* if the subgraph induced by all vertices adjacent to i or j is a complete bipartite graph. Let $\sigma = [e_1,\ldots,e_n]$ be a sequence of pairwise nonadjacent edges of a bipartite graph $H = (V_1,V_2,E)$. Denote by S_i the set of endpoints of the edges $e_1,e_2,\ldots,e_i$ and let $S_0 = \emptyset$. We say that σ is a *perfect edge elimination scheme* for H if each edge e_i

is bisimplicial in the induced subgraph $H_{(V_1 \cup V_2)\setminus S_{i-1}}$ and $H_{(V_1 \cup V_2)\setminus S_n}$ has no edge. GOLUMBIC [1980] proved that a chordal bipartite graph has a perfect edge elimination scheme. This immediately follows from our results as follows. Construct the totally-balanced matrix corresponding to the chordal bipartite graph as indicated in (2.3.3). Let i_0 be a nest row and let E_0 be such that for all columns E covering i_0 we have $E_0 \subseteq E$. Then the edge of the chordal bipartite graph corresponding to i_0 and E_0 is a bisimplicial edge. This shows that the algorithm we gave in Section 2.2 can also be used to construct a perfect edge elimination scheme for a chordal bipartite graph.

2.4. The simple plant location problem

Given a set of possible locations for establishing new facilities (plants, warehouses etcetera) or expanding already existing facilities. The simple plant location problem deals with the supply of a single commodity from a subset of these to a set of clients with prescribed demands for the commodity. We assume unlimited capacity of each facility such that in principle any facility can satisfy all demands. Given the cost structure we seek a minimum cost production/transportation plan (in terms of the number of facilities established, their location and the amount shipped from each facility to each client) satisfying all demands. The constituents of the simple plant location problem are:

m:	the number of potential facilities indexed by j, $j \in J = \{1,2,\ldots,m\}$;
n:	the number of clients indexed by i, $i \in I = \{1,2,\ldots,n\}$;
f_j:	the nonnegative fixed cost for establishing facility j;
p_j:	the per unit cost of operating facility j;
b_i:	the number of units demand by client i;
t_{ij}:	the transportation cost of shipping one unit from facility j to client i.

We introduce the following decision variables:

z_j:	$z_j = 1$ if facility j is open, and 0 otherwise.
s_{ij}:	number of units produced at facility j and shipped to client i.

The *simple plant location problem* can be formulated as

$$\text{minimize } \textstyle\sum_{i\in I} \sum_{j\in J} (p_j+t_{ij})s_{ij} + \sum_{j\in J} f_j z_j$$

$$\begin{aligned}\text{subject to } \textstyle\sum_{j\in J} s_{ij} &= b_i, && i \in I,\\ s_{ij} &\le b_i z_j, && i \in I,\ j \in J,\\ s_{ij} &\ge 0, && i \in I,\ j \in J,\\ z_j &\in \{0,1\}, && j \in J.\end{aligned}$$

If we define $x_{ij} = s_{ij}/b_i$ and $c_{ij} = b_i(p_j+t_{ij})$, then the problem is in the form in which it is usually studied:

$$\text{minimize} \quad \textstyle\sum_{i\in I} \sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j z_j$$

$$\begin{aligned}\text{subject to } \textstyle\sum_{j\in J} x_{ij} &= 1, && i \in I,\\ x_{ij} &\le z_j, && i \in I,\ j \in J,\\ x_{ij} &\ge 0, && i \in I,\ j \in J,\\ z_j &\in \{0,1\}, && j \in J.\end{aligned}$$

For a historical review of this problem we refer to KRARUP & PRUZAN [1977]. A very efficient algorithm for its solution was developed by ERLENKOTTER [1978].

We want to study the case in which all clients and potential facilities are located at vertices of a tree, and the cost c_{ij} is linear with respect to the distance d_{ij} between vertex i of client i and vertex j of facility location j, say, $c_{ij} = w_i d_{ij}$. We assume that the tree has vertex set $\{1,2,\ldots,n\}$ numbered in such a way that it forms a nest ordering, i.e., i is endpoint of a longest path in the subtree induced by $\{i,i+1,\ldots,n\}$ for all $i = 1,\ldots,n$. It follows from Lemma 1.3.4 that such a numbering can be obtained in $O(n \log n)$ time. So the problem we will study can be formulated as

$$\text{(P)} \qquad \text{minimize} \quad \textstyle\sum_{i=1}^{n} \sum_{j=1}^{n} w_i d_{ij} x_{ij} + \sum_{j=1}^{n} f_j z_j$$

$$\begin{aligned}\text{subject to } \textstyle\sum_{j=1}^{n} x_{ij} &= 1, && i = 1,2,\ldots,n,\\ x_{ij} &\le z_j, && i,j = 1,2,\ldots,n,\\ x_{ij} &\ge 0, && i,j = 1,2,\ldots,n,\\ z_j &\in \{0,1\}, && j = 1,2,\ldots,n.\end{aligned}$$

Consider the dual problem associated with the linear programming relaxation of (P) with the constraints $z_j \in \{0,1\}$ replaced by $z_j \ge 0$, $j = 1,2,\ldots,n$:

(D') maximize $\sum_{i=1}^{n} y_i$

subject to $\sum_{i=1}^{n} \mu_{ij} \le f_j, \qquad j = 1,2,\ldots,n,$

$$y_i - \mu_{ij} \le w_i d_{ij}, \quad i,j = 1,2,\ldots,n,$$

$$\mu_{ij} \ge 0, \quad i,j = 1,2,\ldots,n.$$

Setting $\mu_{ij} = \max(0, y_i - w_i d_{ij})$ gives the more condensed dual problem

(D) maximize $\sum_{i=1}^{n} y_i$

subject to $\sum_{i=1}^{n} \max(0, y_i - w_i d_{ij}) \le f_j, \qquad j = 1,2,\ldots,n.$

In order to solve problem (P) we first construct a feasible dual solution y and a set $J \subseteq \{1,2,\ldots,n\}$ such that

(2.4.1) $y_i \ge w_i d_{ij}$ for at least one $j \in J$,

(2.4.2) $y_i > w_i d_{ij}$ for at most one $j \in J$, and

(2.4.3) $\sum_{i=1}^{n} \max(0, y_i - w_i d_{ij}) = f_j$ for all $j \in J$.

We then define

$z_j = 1$ if and only if $j \in J$,
$x_{ij} = 1$ if and only if $j = j(i) \in J$, where $j(i)$ is a closest vertex in J to i.

The fact that both these feasible dual and primal solutions are optimal follows from the following theorem given by ERLENKOTTER [1978].

THEOREM 2.4.4. *The values* V(D) *and* V(P) *of the above solutions to problems* D *and* P *are equal.*

PROOF. Due to (2.4.1) and (2.4.2) we have $y_i \ge w_i d_{ij(i)}$ and $y_i \le w_i d_{ij}$ for all $j \in J$, $j \ne j(i)$. Therefore

$$\begin{aligned} V(D) &= V(D) + \textstyle\sum_{j\in J}(f_j - \sum_{i=1}^n \max(0,y_i - w_i d_{ij})) \\ &= \textstyle\sum_{i=1}^n y_i + \sum_{j\in J} f_j - \sum_{i=1}^n (y_i - w_i d_{ij(i)}) - \\ &\qquad\qquad \textstyle\sum_{i=1}^n \sum_{j\in J: j\neq j(i)} \max(0,y_i - w_i d_{ij}) \\ &= \textstyle\sum_{i=1}^n w_i d_{ij(i)} + \sum_{j\in J} f_j \\ &= V(P). \quad \square \end{aligned}$$

We now proceed with the procedure to obtain the dual feasible solution y and the set $J \subseteq \{1,2,\ldots,n\}$ satisfying (2.4.1), (2.4.2) and (2.4.3). The procedure will be described in an informal way and demonstrated by an example. We consider all sets U of potential facilities j with $f_j < \infty$ for which there is a client i and a radius r such that $U = \{j \mid f_j < \infty,\ d_{ij} \leq r\}$. We can find these sets by calculating for each client i the set of distances to all potential facilities and order the elements of this set in increasing order, say $r_{i1} < r_{i2} < \ldots < r_{ik(i)}$, $1 \leq k(i) \leq n$. Each $r_{ij}(1 \leq j \leq k(i))$ corresponds to a set of facilities which are within distance r_{ij} from client i. If $r_{ij} > 0$, then also the empty set belongs to these sets. Define $r_{i0} = 0$ in case $r_{i1} > 0$ and also define $r_{i,k(i)+1} = \infty$ $(i = 1,2,\ldots,n)$. We order all these sets in lexicographically increasing order, say $U_1,U_2,\ldots,U_k$; note that $k \leq n^2$. With each of these sets U_i we associate a list L_i of pairs $(j,r_{j,\ell+1})$, where $(j,r_{j,\ell+1}) \in L_i$ if and only if $U_i = \{k \mid f_k < \infty,\ d_{jk} \leq r_{j\ell}\}$. All pairs in one list are ordered according to their first component, i.e., (i,a) occurs above (j,b) on the list if $i < j$. The Dual Algorithm has as input these lists. The algorithm considers the list according to increasind index and within each list works from top to bottom. When we consider the pair $(j,r_{j,\ell+1})$ in L_i the current value of y_j is equal to $w_j r_{j\ell}$. We try to increase y_j to $w_j r_{j,\ell+1}$. If this is possible with respect to the dual constraints, then we increase y_j to $w_j r_{j,\ell+1}$ and postpone increasing y_j and further until we reach the list containing the pair $(j,r_{j,\ell+2})$. Note that this list has a larger index than i, the index of the list currently under consideration. If it is not possible to increase y_j to $w_j r_{j,\ell+1}$, then this would violate a constraint $\sum_{i=1}^n \max(0,y_i - w_i d_{ik}) \leq f_k$ for some k $(1 \leq k \leq n)$. In this case we increase y_j as far as possible, i.e., until a constraint gets tight. At this moment we regard y_j as *determined* and we call a constraint which prevented y_j from increasing any further a *determining constraint* for y_j. We delete all pairs with first component j from all lists and continue with the algorithm until all y-values are determined.

EXAMPLE 2.4.5. The tree we consider is given by Figure 2.6, the length of each edge is given between brackets. We have $w_i = 1$, $i = 1,2,3,4,5$, $f_1 = 5$, $f_3 = 6$, $f_4 = 4$, $f_2 = f_5 = \infty$.

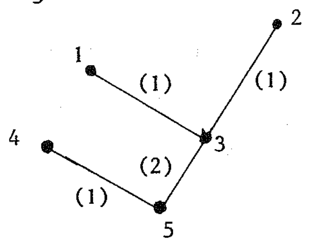

Figure 2.4.6. Tree of Example 2.4.5.

The clients i, their distances r_{ij} and the set of facilities reached within r_{ij} are given in Table 2.4.7.

vertex i	distance r_{ij}	facilities reached within r_{ij}
1	0	{1}
	1	{1,3}
	4	{1,3,4}
2	0	$\emptyset$
	1	{3}
	2	{3,1}
	4	{3,1,4}
3	0	{3}
	1	{3,1}
	3	{3,1,4}
4	0	{4}
	3	{4,3}
	4	{4,3,1}
5	0	$\emptyset$
	1	{4}
	2	{4,3}
	3	{4,3,1}

Table 2.4.7. Calculating sets U for Example 2.4.5.

Ordering the sets in lexicographically increasing order gives $\emptyset$, $\{1\}$, $\{3\}$, $\{3,1\}$, $\{4\}$, $\{4,3\}$ and $\{4,3,1\}$. The corresponding lists are given by:

L_1: $(2,1)$, $(5,1)$;
L_2: $(1,1)$;
L_3: $(2,2)$, $(3,1)$;
L_4: $(1,4)$, $(2,4)$, $(3,3)$;
L_5: $(4,3)$, $(5,2)$;
L_6: $(4,4)$, $(5,3)$;
L_7: $(1,\infty)$, $(2,\infty)$, $(3,\infty)$, $(4,\infty)$, $(5,\infty)$.

The Dual Algorithm works as follows:

L_1: $y_2 = 1$, $y_5 = 1$;
L_2: $y_1 = 1$ $(f_1 = 4)$;
L_3: $y_2 = 2$, $y_3 = 1$ $(f_3 = 4)$;
L_4: $y_1 = 4$, $y_2 = 3$, $y_3 = 1$ $(f_1 = 0$, $f_3 = 0)$, y_2 is determined, y_3 is determined;
L_5: $y_4 = 3$, $y_5 = 2$ $(f_4 = 0)$;
L_6: $y_4 = 3$, $y_5 = 2$, y_4 is determined, y_5 is determined;
L_7: $y_1 = 4$, y_1 is determined;

The dual value is equal to 13. □

Define $I(j) = \{i \mid y_i > w_i d_{ij}\}$ $(j = 1,2,\ldots,n,\ f_j < \infty)$. The Primal Algorithm has as input the sets $I(j)$ and as output a subset $J \subseteq \{j \mid f_j < \infty\}$ which will define the optimal set of facilities which should be opened.

PRIMAL ALGORITHM.

```
Delete all j which do not correspond to a tight constraint; J := ∅;
while there are still vertices left
do add the largest vertex, say ℓ, to J and delete all vertices k for
   which I(k) ∩ I(ℓ) ≠ ∅
od □
```

Note that the dual solution y and the set J satisfy (2.4.2) and (2.4.3). It remains to be shows that $y_i \geq w_i d_{ij}$ for at least one $j \in J$. Define the (0,1)-matrix $A = (a_{ij})$ as follows. Columns correspond to facilities and are ordered in increasing order with respect to the corresponding vertices where they are located. The rows correspond to clients and are ordered in the same order as the y-values are determined, i.e., the row corresponding to client i precedes the row corresponding to client j if y_i was determined before y_j. There is a one in the column corresponding to facility j and the row corresponding to client i if and only if $d_{ij} \le y_i/w_i$. According to the way the y-values were determined we know that the rows of the matrix A are ordered in lexicographically nondecreasing order. It follows from Lemma 2.2.1 that the transpose of A and therefore also A itself is a matrix in standard form. Delete from A all columns which do not correspond to a tight constraint. We know that in the resulting matrix each row is covered by at least one column, namely the column corresponding to a determining constraint. Showing that $y_i \ge w_i d_{ij}$ for at least one $j \in J$ is equivalent to showing that the subset of columns J covers every row of the matrix A.

THEOREM 2.4.8. *Let* A *be the matrix defined above. Then the set of columns* J *found by the Primal Algorithm covers all rows of* A.

PROOF. First of all we delete from A all columns which do not correspond to a tight constraint. The proof that J covers all rows will be by induction on the number of columns. The induction hypothesis is that all rows which are covered by a determining column (i.e., a column corresponding to a determining constraint) are covered by the subset of columns found by the Primal Algorithm. Note that at the beginning each row is covered by a determining column.

Let ℓ be the column chosen by the Primal Algorithm and let column k be deleted because $I(k) \cap I(\ell) \neq \emptyset$. We will show that for all rows which are not covered by column ℓ the deleted columns are not determining columns. Then using the induction hypothesis this proves that J covers all rows of A. Let $i \in I(k) \cap I(\ell)$. Suppose the row corresponding to client j is covered by column k and not by column ℓ. First of all we know that y_j was determined before y_i. If y_j was determined after y_i, then since the matrix A is in standard form column ℓ would also cover the row corresponding to client j. We show that column k is not a determining column for the row corresponding to client j by proving that after y_j was determined the value of y_i was increased with a positive amount above $w_i d_{ik}$. This shows that constraint k was not tight when y_j was determined, i.e., column k is not a determining column for client j.

Let $w_i r_{i,m+1}$ $(0 \le m \le k(i))$ be the value of y_i after the list in which y_j was determined was completed. This means that the set of facilities which can be reached from client i within r_{im} is lexicographically smaller than or equal to the set corresponding to the list in which y_j was determined. If $d_{ik} \le r_{im}$ and $d_{i\ell} \le r_{im}$, then since the row corresponding to client j is covered by column k the standard form of the matrix implies that this row is also covered by column ℓ. This contradicts our starting assumption. If $d_{ik} > r_{im}$, then the first time facility k will occur in a set of facilities which can be reached from client i within a distance r is when r = $= d_{ik}$. Since $d_{ik} \ge r_{i,m+1}$ this set corresponds to a list with larger index than the one in which y_j was determined. Since $y_i > w_i d_{ik}$ $(i \in I(k))$ for the final value of y_i we have that after y_j was determined we increased y_j with a positive amount above $w_i d_{ik}$. If $d_{ik} \le r_{im}$ and $d_{i\ell} > r_{im}$, then since $y_i > w_i d_{i\ell}$ $(i \in I(\ell))$ we can repeat the argument for the case $d_{ik} > r_{im}$ to prove that y_i was increased with a positive amount above $w_i d_{i\ell}$ after y_j was determined. Since $d_{ik} < d_{i\ell}$ this also shows that y_i was increased with a positive amount above $w_i d_{ik}$ after y_j was determined. □

EXAMPLE 2.4.9. Consider Example 2.4.5. All constraints are tight. We have $I(1) = \{1,2\}$, $I(3) = \{1,2,3\}$ and $I(4) = \{4,5\}$. The Primal Algorithm has as output $J = \{3,4\}$. The associated value of the primal solution is equal to 13. □

Let us now look at the complexity of this algorithm. Finding all sets of facilities requires $O(n^2 \log n)$ time. There are $O(n^2)$ sets of $O(n)$ size, so a lexicographical ordering can be found in $O(n^3)$ time. The calculation of the y-values takes place in $O(n^2)$ iterations, one for each list. In each list L_i we calculate $\min_{j \in U_i} \{f_j - \sum_{k=1}^{n} \max(0, y_k - w_k d_{kj})\}$ which we have to adjust each time a y-value changes. Since there are at most n pairs in each list the total work done in one iteration is $O(n)$. Therefore the y-values can be found in $O(n^3)$ time. Each set $I(j)$ can be found in $O(n)$ time in such a way that the elements of $I(j)$ are ordered in increasing order. The intersection of two of those sets can be found in $O(n)$ time. So finding the set J requires at most n iterations of $O(n^2)$ calculations, i.e., a total of $O(n^3)$ calculations. We conclude that the simple plant location problem on a tree with n vertices can be solved in $O(n^3)$ time.

The dual problem (D) has a nice interpretation which was first observed by TAMIR [1980] in the context of a location problem similar to the minimum cost covering problem defined in Section 1.3. Let $V(U)$ $(U \subseteq N = \{1,2,\ldots,n\})$ be the optimum value of the primal problem when we only want to serve clients at vertices $i \in U$. A situation like this may occur when clients $i \in U$ decide to form a coalition. Suppose we want to distribute the cost $V(N)$ of serving all clients among them such that no subset U of N can benefit from forming a coalition, i.e., we try to find a $\lambda \in \mathbb{R}_+^n$ such that $\sum_{i\in N} \lambda_i = V(N)$ and $\sum_{i\in U} \lambda_i \le V(U)$ for all $U \subseteq N$.

The set of all vectors λ satisfying these conditions is defined to be the *core* C, i.e., $C = \{\lambda \in \mathbb{R}_+^n \mid \sum_{i\in N} \lambda_i = V(N),\ \sum_{i\in U} \lambda_i \le V(U)$ for all $U \subseteq N\}$.

TAMIR [1980] gave an example of a location problem for which the core is empty. A *core allocation* is a distribution of the cost according to a vector belonging to the core. It is clear that a core allocation possesses a desirable stability property which seems to be necessary for a cost allocation to be acceptable by all clients. We shall prove that for the simple plant location problem on a tree the core is equal to the set of optimal dual variables. This follows from the following theorem and the fact that we proved strong duality.

Let us define $B = \{\lambda \in \mathbb{R}_+^n \mid \sum_{i\in N} \lambda_i = V(N),\ \sum_{i=1}^n \max(0,\lambda_i - w_i d_{ij}) \le f_j$ for all $j = 1,\dots,n\}$.

<u>THEOREM 2.4.10</u>. $C = B$.

<u>PROOF</u>. Let $\lambda \in C$. Define $U_j = \{i \mid \lambda_i \ge w_i d_{ij}\}$. Then

$$\sum_{i\in U_j} \lambda_i \le V(U_j) \le f_j + \sum_{i\in U_j} w_i d_{ij},$$

or equivalently

$$\sum_{i\in U_j} (\lambda_i - w_i d_{ij}) = \sum_{i=1}^n \max(0,\lambda_i - w_i d_{ij}) \le f_j, \text{ i.e., } \lambda \in B.$$

Let $\lambda \in B$ and let U be a subset of N. Suppose $V(U) = \sum_{k=1}^t f_k + {}$ $+ \sum_{i\in U} w_i d_{ij(i)}$, where $j(i) \in \{1,2,\dots,k\}$ is the closest vertex in $\{1,\dots,k\}$ to i. Then

$$\begin{aligned}\sum_{i\in U} (\lambda_i - w_i d_{ij(i)}) &\le \sum_{i\in U} \max(0,\lambda_i - w_i d_{ij(i)}) \\ &\le \sum_{k=1}^t \sum_{i=1}^n \max(0,\lambda_i - w_i d_{ij}) \\ &\le \sum_{k=1}^t f_k,\end{aligned}$$

or equivalently

$$\sum_{i\in U} \lambda_i \le \sum_{i\in U} w_i d_{ij(i)} + \sum_{k=1}^{t} f_k = V(U), \quad \text{i.e.,} \quad \lambda \in C. \quad \square$$

CHAPTER 3

THE p-MEDIAN PROBLEM WITH MUTUAL COMMUNICATION

Let T = (V,E) be a tree and let existing facilities be located at the vertices. We want to locate p new facilities on the tree such that the sum of the weighted distances between existing and new facilities as well as between new facilities is minimized. This problem is known as the p-*median problem with mutual communication* and can be formulated as

$$\text{minimize}_{X = \{x_1,x_2,\ldots,x_p\}:X\subseteq T} F(X) = \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij} d(v_i,x_j) + \\ + \tfrac{1}{2}\sum_{j=1}^{p} \sum_{k=1}^{p} \beta_{jk} d(x_j,x_k),$$

where α_{ij} $(i=1,2,\ldots,n, j=1,2,\ldots,p)$ and $\beta_{jk} = \beta_{kj}$ $(j,k = 1,2,\ldots,p)$ are nonnegative integers, called *weights*.

The p-median problem with mutual communication has been studied extensively in the plane using rectilinear distances. The *rectilinear distance* d(A,C) between points A = (a,b) and C = (c,d) in the plane is defined by $d(A,C) = |a-c| + |b-d|$. This problem can be separated into two independent problems on a line, one for each coordinate. Let $a_1,a_2,\ldots,a_n$, $a_i < a_{i+1}$, $i = 1,2,\ldots,n-1$, be existing facility locations on a line. The p-median problem with mutual communication on a line can be formulated as

$$\text{minimize}_{X = \{x_1,\ldots,x_p\}} G(X) = \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij} |a_i-x_j| + \\ + \tfrac{1}{2}\sum_{j=1}^{p} \sum_{k=1}^{p} \beta_{jk} |x_j-x_k|.$$

There are essentially two approaches to solve the problem on a line.

The first one is the direct search approach by PRITSKER & GHARE [1970], RAO [1973], JUEL & LOVE [1976], SHERALI & SHETTY [1978], and KOLEN [1981]. The direct search approach is as follows. Start with some solution X^1 with new facilities located at existing facility locations. Consider the set N_q

of new facilities located at some existing facility location a_q ($1 \leq q \leq n$). If there is a subset $S \subseteq N_q$ that can be moved to an adjacent existing facility location (a_{q-1} or a_{q+1}) such that the new solution X^2 satisfies $G(X^2) < G(X^1)$, move that subset to the corresponding adjacent facility location. Then repeat this procedure with X^2. If no such subset exists at any location we have found an optimal solution. A justification of the algorithm can be found in the linear programming formulation of the problem. RAO [1973] has proved through the negation of various alternatives that a single non-degenerate simplex pivot can only result in the movement of a subset of new facilities at a given existing facility location to an adjacent facility location. We will show in Section 3.1 that the same result holds for the problem on a tree. In all algorithms proposed, the crucial point is to find a subset of new facilities located at an existing facility which, when moved to an adjacent facility location, gives the largest reduction in the value of G. None of these algorithms, with a single exception, has a polynomial running time; it was shown in KOLEN [1981] how the direct search algorithms could be adjusted to run in polynomial time. This algorithm will be generalized to the case of a tree in Section 3.2. In contrast, the problem on an arbitrary graph is NP-hard, as will be shown in Section 3.3.

The second approach is the cut approach due to PICARD & RATLIFF [1978]. The complexity of this algorithm is $O(np^3)$, which is the same as that of the direct search algorithm given by KOLEN [1981]. This is not surprising since it was shown in the latter reference that the two approaches are equivalent.

3.1. Necessary and sufficient conditions for optimality

In this section we give necessary and sufficient conditions for a solution X to be an optimal solution to the p-median problem with mutual communication on a tree. These conditions will be explored in Section 3.2 to derive a polynomial algorithm to solve this problem.

Let X^1 be a given solution and let a,b be two points on the same edge such that there are no facilities lying on the edge between a and b. Deleting the part of the edge between a and b from the tree results in two subtrees. Let $T_a(T_b)$ denote the subtree containing a(b). Let X^2 be the solution we get from X^1 be moving a subset S of new facilities located at a to b. Then the distance between new facilities in S and facilities, both existing and new, which in both solutions are lying on $T_a(T_b)$ is increased (decreased) with $d(a,b)$. So the difference between $F(X^2)$ and $F(X^1)$ is a constant times $d(a,b)$, where the constant does not depend on the exact location of a facility, but only on whether the facility is on T_a or T_b. If before moving the subset of new facilities S from a to b we move a subset of new facilities, disjoint from S in T_a or T_b, then this does not affect the change in the objective value when we move S. We call two movements *independent* if the change in the objective value when one movement is performed is the same whether the other movement has been performed first or not.

Let $X = \{x_1,x_2,\ldots,x_p\}$ be a given solution with $x_j \in V$ for all $j = 1,2,\ldots,p$. A necessary condition for optimality is that no subset of facilities can be moved to an adjacent vertex such that the objective value decreases. We shall prove that this condition is sufficient as well. Suppose X is not optimal. Then there is a solution Y such that $F(Y) < F(X)$. It was proved in Theorem 1.1.9 that F is a convex function, i.e.,

$$F(\lambda X + (1-\lambda)Y) \le \lambda F(X) + (1-\lambda)F(Y), \quad 0 \le \lambda \le 1.$$

It follows that $F(\lambda X + (1-\lambda)Y) < F(X)$, $0 \le \lambda < 1$.

Define ℓ, $\ell > 0$ to be the length of the shortest edge. Then there is a λ_1, $0 \le \lambda_1 < 1$ such that for all λ, $\lambda_1 \le \lambda < 1$ we have

$$d(x_i,\lambda x_i + (1-\lambda)y_i) = (1-\lambda)d(x_i,y_i) < \tfrac{1}{2}\ell, \; i = 1,2,\ldots,p.$$

Let Z be the solution $\lambda X + (1-\lambda)Y$ for some λ, $\lambda_1 \le \lambda < 1$. In order to obtain Z from X we only have to move subsets of new facilities along parts of the edges. All movements starting from different vertices as well as movements starting from the same vertex along different edges are independent. Since $F(Z) < F(X)$ we know that there must be a vertex v_s and an edge $\{v_s,v_t\}$ such that performing all movements from v_s on this edge reduces the objective value. Let in Z subsets of new facilities S_i be located at the point u_i on the edge $\{v_s,v_t\}$, $i = 1,2,\ldots,k$, where $0 < d(v_s,u_i) < d(v_s,u_{i+1}) < \tfrac{1}{2}\ell$, $i = 1,2,\ldots,k-1$. We can perform the movements in k steps. In step i we move the subset of new facilities $\cup_{j=i}^{k} S_j$ from u_{i-1} to u_i, $i = 1,2,\ldots,k$, where $u_0 = v_s$. Since the objective value is decreased after these movements at least one of the movements gives a decrease in the objective value. Without loss of generality let this be the movement performed at step i ($1 \le i \le k$). The objective value increases with $d(u_{i-1},u_i)$ times a constant, where the constant only depends on which facilities not belonging to $\bigcup_{j=i}^{k} S_j$ are located in $T_{u_{i-1}}$ or T_{u_i}. We know by assumption that this constant is negative. We claim that when we move the subset $\bigcup_{j=i}^{k} S_j$ of new facilities located at v_s in X to v_t, the objective value decreases. This proves that the necessary conditions are also sufficient for optimality. When we move the subset $\bigcup_{j=i}^{k} S_j$ from v_s to v_t the value of objective value increases with $d(v_s,v_t)$ times the same constant we had before when we moved the same subset in the solution Z from u_{i-1} to u_i. The two constants are the same since the facilities not belonging to $\bigcup_{j=i}^{k} S_j$ located at $T_{u_{i-1}}(T_{u_i})$ in Z are the same as the facilities located at $T_s(T_t)$ in X.

3.2. Polynomial-time algorithm

In this section we present a polynomial-time algorithm to solve the p-median problem with mutual communication on a tree. This algorithm generalizes the 1-median algorithm of GOLDMAN [1971].

ALGORITHM.

```
P: = {1,2,...,p}.
while the tree consists of at least one edge
do select a tip vertex $v_s$ of the tree. Locate all new facilities be-
   longing to P at $v_s$ and determine the subset S of P which when
   moved to the adjacent vertex $v_t$ gives the largest positive decrease
   in the objective value. If no such subset exists, then the current
   solution is optimal, else define $x_j = v_s$ for all $j \in P\backslash S$, $\alpha_{tk}$ :=
   := $\alpha_{tk} + \alpha_{sk} + \sum_{j \in P\backslash S} \beta_{jk}$ for all $k \in S$, P := S and delete the
   edge $\{v_s,v_t\}$ from the tree
od. □
```

The proof that the algorithm constructs an optimal solution is by induction on the number of edges. The induction hypothesis is that for all k-median problems with mutual communication on a tree ($k \le p$) the solution constructed by the algorithm has the property that no subset of facilities can be moved to an adjacent vertex such that the objective value decreases.

Consider the case where there is only one edge $\{v_s,v_t\}$. We locate new facilities belonging to P at v_s. The subset S which gives the largest positive reduction in the objective value when moved to v_t is moved to v_t. Let $S_1(S_2)$ be a subset of new facilities located at $v_s(v_t)$ after we moved

S, which gives a reduction in the objective value when moved to $v_t(v_s)$. Then moving $S \cup S_1(S\backslash S_2)$ in the solution with all facilities located at v_s would give a larger reduction than moving S. This contradicts the definition of S. Hence the solution constructed by the algorithm on a tree with one edge satisfies the sufficient conditions for optimality.

Assume the induction hypothesis is true for all trees with less than n vertices ($n \geq 3$). Consider the p-median problem with mutual communication on the tree T with n vertices. We choose a tip vertex v_s and determine the subset S of P which when moved to the adjacent vertex v_t gives a largest positive reduction in the objective value. An argument similar to the one in the previous paragraph shows that in the solution constructed by the algorithm on T no subset of facilities located at $v_s(v_t)$ can be moved to $v_t(v_s)$ such that the objective value decreases. It remains to be shown that no subset of facilities located in the tree T_t we obtain from T by deleting the edge $\{v_s,v_t\}$ can be moved from a vertex in T_t to an adjacent vertex in T_t such that the objective value decreases. If we move a subset of S in T_t from v_k to an adjacent vertex v_ℓ, then the change in the objective value depends on which facilities are located in T_k and which are located in T_ℓ and does not depend on their exact location. Since v_s and v_t always occur in the same subtree we might as well add the weights of the facilities located at v_s to the weight of v_t, i.e., $\alpha_{tk} := \alpha_{tk} + \alpha_{sk} + \sum_{j\in P\backslash S} \beta_{jk}$ for all $k \in S$. Using the induction hypothesis on T_t we know that the solution constructed by the algorithm on T_t for the problem with new adjusted weights α_{tk} ($k\in S$) satisfies the property that no subset of S can be moved from a vertex in T_t to an adjacent vertex in T_t such that the objective value decreases. The previous observation shows that this is equivalent to the property that no subset of S can be moved in the original problem such that the objective value decreases. This proves that the solution constructed by the algorithm satisfies the sufficient condition for optimality and hence is optimal.

Let us now return to the problem of finding a subset $S \subseteq P$ which when moved from v_s to v_t gives the largest reduction in the objective value. When we move the subset S from v_s to v_t the objective value is increased by

$$d(v_s,v_t)\left[\sum_{j\in S} \alpha_{sj} + \sum_{j\in S} \sum_{k\in P\backslash S} \beta_{jk} - \sum_{j\in S} \sum_{i:v_i\in T_t} \alpha_{ij}\right] .$$

The problem we have to solve is given by

$$(3.2.1)\qquad \min_{S\subseteq P}\{\sum_{j\in S}\alpha_{sj}+\sum_{j\in S}\sum_{k\in P\setminus S}\beta_{jk}-\sum_{j\in S}\sum_{i:v_i\in T_t}\alpha_{ij}\}.$$

If this minimum is nonnegative, then the current solution is optimal, else we move the subset for which the minimum is attained. We transform problem (3.2.1) into an equivalent problem by adding the constant $\sum_{j\in P}\sum_{i:v_i\in T_t}\alpha_{ij}$, which does not depend on S. This gives the problem

$$(3.2.2)\qquad \min_{S\subseteq P}\{\sum_{j\in S}\alpha_{sj}+\sum_{j\in S}\sum_{k\in P\setminus S}\beta_{jk}+\sum_{j\in P\setminus S}\sum_{i:v_i\in T_t}\alpha_{ij}\}.$$

In order to solve (3.2.2) we need some results from network flow theory.

Consider the directed graph G (see Figure 3.2.3) with vertex set $\{s,t,1,2,\dots,p\}$ and arc set $\{(s,j)|j=1,2,\dots,p\}\cup\{(j,t)|j=1,2,\dots,p\}\cup\{(i,j)|i,j=1,2,\dots,p,i\neq j\}$. Associated with each arc is a capacity. The capacity of arc (s,j) ((j,t)) is given by a_j (b_j), $j = 1,2,\dots,p$, the capacity of (i,j) is given by c_{ij}, $i,j = 1,2,\dots,p$, $i \neq j$.

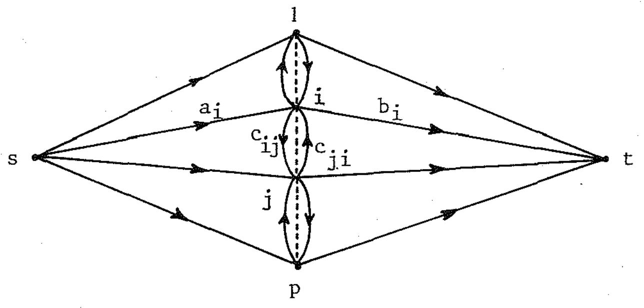

Figure 3.2.3. The digraph G.

The problem of finding a maximum flow from a *source* s to a *sink* t is known as the *maximum flow problem* and can be formulated as

maximize z

$$\text{subject to } \sum_j x_{ji} - \sum_j x_{ij} = \begin{cases} -z & i = s, & (3.2.4)\\ 0 & i \neq s,t, & (3.2.5)\\ z & i = t, & (3.2.6)\end{cases}$$

$$0 \le x_{ij} \le c_{ij},\quad i,j = 1,2,\dots,p,\ i \neq j,$$
$$0 \le x_{sj} \le a_j,\quad j = 1,2,\dots,p,$$
$$0 \le x_{jt} \le b_j,\quad j = 1,2,\dots,p.$$

Constraint (3.2.4) ((3.2.6)) says that the flow out of s (into t) is equal to z. Constraints (3.2.5) say that for each vertex $i \neq s,t$ the flow into the vertex is equal to the flow out of it.

An (s,t)-*cutset* is identified by a pair (S,T) of subsets of vertices with $s \in S$, $t \in T$ which partition the vertex set. The *capacity* of a cutset (S,T) is the sum of the capacities of all arcs with the predecessor in S and the successor in T.

A well-known result in the theory of maximum flows states that the maximum flow from s to t is equal to the minimum capacity of an (s,t)-cutset. The algorithm of KARZANOV [1974] finds a maximum flow and a minimum cutset in $O(p^3)$ time.

The (s,t) cutsets of the digraph G are defined by a subset $S \subseteq P$. The cutset defined by $S \subseteq P$ is given by $(\{s\} \cup (P \backslash S), \{t\} \cup S)$ and its capacity is equal to

$$\sum_{j \in S} a_j + \sum_{j \in P \backslash S} b_j + \sum_{j \in P \backslash S} \sum_{k \in S} c_{jk}.$$

The problem of finding a minimum capacity cutset can be formulated as

$$(3.2.7) \qquad \min_{S \subseteq P} \{ \sum_{j \in S} a_j + \sum_{j \in P \backslash S} b_j + \sum_{j \in P \backslash S} \sum_{k \in S} c_{jk} \}.$$

Note that problem (3.2.7) is equivalent to problem (3.2.2) if we define $a_j := \alpha_{sj}$, $b_j := \sum_{i: v_i \in T_t} \alpha_{ij}$ and $c_{jk} := \beta_{jk}$. Therefore problem (3.2.2) can be solved in $O(p^3)$ time.

The total complexity of the algorithm is $O(np^3)$ since we have to solve a minimum cut problem for each of the n-1 edges of the tree with n vertices.

3.3. NP-Hardness

In this section we prove that the p-median problem with mutual communication on an arbitrary graph is NP-hard. It is sufficient to establish NP-completeness for the following decision problem:

MMC

Instance: A graph $G' = (V',E')$ with $V' = \{v_1', v_2', \dots, v_n'\}$, $p \in \mathbb{N}$, weights $\alpha(v_i',j) \in \mathbb{Z}^+$, $i = 1,2,\dots,n$, $j = 1,2,\dots,p$, $\beta(j,k) \in \mathbb{Z}^+$, $j,k = 1,2,\dots,p$, lenghts $\ell(e) \in \mathbb{Z}^+$, $e \in E'$, $B \in \mathbb{Q}^+$.

Question: Is there a set $X = \{x_1, x_2, \dots, x_p\}$ on G' such that

$$\sum_{i=1}^{n}\sum_{j=1}^{p} \alpha(v_i',j)d(v_i',x_j) + \tfrac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} \beta(j,k)d(x_j,x_k) \le B?$$

<u>THEOREM 3.3.1</u>. MMC *is* NP-*complete.*

<u>PROOF</u>. The problem belongs to the class $\mathcal{NP}$ since by using standard shortest path techniques one can test whether a given solution satisfies the bound B in polynomial time.

We shall reduce the problem CLIQUE defined in Chapter 0 to MMC. Example 3.3.6 shows the reduction for a special case. Let an instance of CLIQUE be given by $G = (V,E)$ with $V = \{v_1,v_2,\ldots,v_n\}$ and c. The corresponding instance of MMC is defined by

$V' = V \cup \bar{V} \cup \bar{\bar{V}} \cup \{s_1,s_2\} \cup \{\bar{s}_1,\bar{s}_2\} \cup \{\bar{\bar{s}}_1,\bar{\bar{s}}_2\}$, where $\bar{V} = \{\bar{v}|v \in V\}$ and $\bar{\bar{V}} = \{\bar{\bar{v}}|v \in V\}$,

$E' = \{\{s_1,v_i\},\{s_2,v_i\},\{\bar{s}_1,\bar{v}_i\},\{\bar{s}_2,\bar{v}_i\},\{\bar{\bar{s}}_1,\bar{\bar{v}}_i\},\{\bar{\bar{s}}_2,\bar{\bar{v}}_i\},\{v_i,\bar{v}_i\},\{\bar{v}_i,\bar{\bar{v}}_i\}|$
$i = 1,2,\ldots,n\} \cup \{\{v_i,\bar{\bar{v}}_j\}|(v_i,v_j) \in E,\ i < j\}$,

$p = 3c$,

$\alpha(s_1,j) = \alpha(s_2,j) = c-j+1, \alpha(\bar{s}_1,j+c) = \alpha(\bar{s}_2,j+c) = 2, \alpha(\bar{\bar{s}}_1,j+2c) = \alpha(\bar{\bar{s}}_2,j+2c) = j,\ j = 1,2,\ldots,c,\ \beta(j,j+c) = \beta(j+c,j+2c) = 2,\ j = 1,2,\ldots,c,\ \beta(i,j+2c) = 2,\ i,j = 1,2,\ldots,c,\ i < j$, all other weights are zero,

$\mathcal{L}(e) = 1,\ e \in E'$,

$B = 3(c^2+3c)$.

We claim that $G = (V,E)$ contains a clique of size c if and only if there exist points $x_1,x_2,\ldots,x_{3c}$ on $G' = (V',E')$ such that

$$\sum_{j=1}^{c} [(c-j+1)(d(s_1,x_j)+d(s_2,x_j)) + 2(d(\bar{s}_1,x_{j+c}) + d(\bar{s}_2,x_{j+c})) + j(d(\bar{\bar{s}}_1,x_{j+2c}) + d(\bar{\bar{s}}_2,x_{j+2c})) + 2d(x_j,x_{j+c}) + 2d(x_{j+c},x_{j+2c})] + \sum_{i=1}^{c}\sum_{j=i+1}^{c} 2d(x_i,x_{j+2c}) \le 3(c^2+3c). \quad (3.3.2)$$

Let $x_1,x_2,\ldots,x_{3c}$ be points on $G' = (V',E')$ such that (3.3.2) holds. We have the following inequalities:

$$d(s_k,\bar{s}_k) \le d(s_k,x_j) + d(x_j,x_{j+c}) + d(x_{j+c},\bar{s}_k),\ j=1,2,\ldots,c, k=1,2, \quad (3.3.3)$$

$$d(\bar{s}_k,\bar{\bar{s}}_k) \le d(\bar{s}_k,x_{j+c}) + d(x_{j+c},x_{j+2c}) + d(x_{j+2c},\bar{\bar{s}}_k),\ j=1,2,\ldots,c,\ k=1,2, \quad (3.3.4)$$

(3.3.5) $\quad d(s_k,\bar{\bar{s}}_k) \le d(s_k,x_i) + d(x_i,x_{j+2c}) + d(x_{j+2c},\bar{\bar{s}}_k),\ i,j=1,2,\ldots,c,$
$i < j,\ k=1,2.$

By adding all these inequalities (3.3.3), (3.3.4) and (3.3.5) together we find that the lefthandside of (3.3.2) is greater than or equal to $c[d(s_1,\bar{s}_1) + d(\bar{s}_1,\bar{\bar{s}}_1) + d(s_2,\bar{s}_2) + d(\bar{s}_2,\bar{\bar{s}}_2)] + c(c-1)[d(s_1,\bar{\bar{s}}_1) + d(s_2,\bar{\bar{s}}_2)]/2 = 3(c^2+3c)$. It follows that equality holds in (3.3.2). Therefore equality holds for all inequalities (3.3.3), (3.3.4) and (3.3.5). It follows that $x_j \in V$, say $x_j = v(j)$, then $x_{j+c} = \bar{v}(j)$ and $x_{j+2c} = \bar{\bar{v}}(j)$, $j=1,2,\ldots,c$. It follows from (3.3.5) that $(v(i),\bar{\bar{v}}(j)) \in E'$, i.e. $(v(i),v(j)) \in E$, $i < j$. Hence $\{v(j)\,|\,j = 1,2,\ldots,c\}$ is a clique in $G = (V,E)$.

Let $\{v(j)\,|\,j=1,2,\ldots,c\}$ be a clique of size c in $G = (V,E)$. Define $x_j = v(j)$, $x_{j+c} = \bar{v}(j)$ and $x_{j+2c} = \bar{\bar{v}}(j)$. Then $x_1,x_2,\ldots,x_{3c}$ satisfies (3.3.2) with equality. □

EXAMPLE 3.3.6. Let the graph $G = (V,E)$ be as given in Figure 3.3.7.

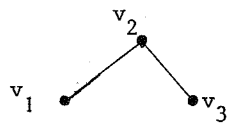

Figure 3.3.7. The graph $G = (V,E)$.

The corresponding graph $G' = (V',E')$ of MMC is given by Figure 3.3.8.

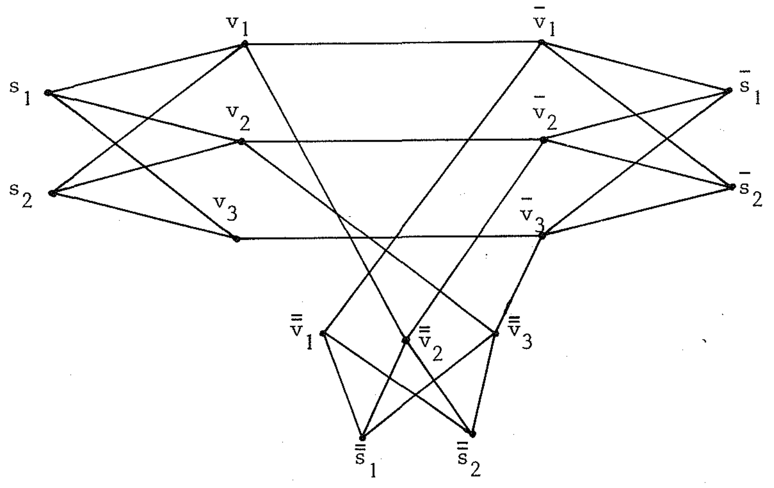

Figure 3.3.8. The graph $G' = (V',E')$.

CHAPTER 4

THE p-CENTER PROBLEM WITH MUTUAL COMMUNICATION

Let $T = (V,E)$ be a tree with $V = \{v_1,v_2,\ldots,v_n\}$ and let existing facilities be located at the vertices. We want to locate p new facilities on the tree such that the maximum over the weighted distances between existing and new facilities as well as between new facilities is minimized. This problem is known as the *p-center problem with mutual communication* and can be formulated as

$$\min_{x_1,x_2,\ldots,x_p}\{\max\{\max_{i=1,\ldots,n,j=1,\ldots,p}\{\alpha_{ij}d(v_i,x_j)\},$$
$$\max_{j,k=1,\ldots,p}\{\beta_{jk}d(x_j,x_k)\}\}\},$$

where α_{ij}, $i = 1,2,\ldots,n$, $j = 1,2,\ldots,p$ and $\beta_{jk} = \beta_{kj}, j,k = 1,2,\ldots,p$ are positive integers.

We can reformulate the problem as

(P) minimize z

$$\text{subject to } d(v_i,x_j) \le z/\alpha_{ij},\ i = 1,2,\ldots,n, j = 1,\ldots,p, \quad (4.1)$$
$$d(x_j,x_k) \le z/\beta_{jk},\ j,k = 1,\ldots,p. \quad (4.2)$$

Problem (P) was solved by FRANCIS, LOWE & RATLIFF [1978]. They first give necessary and sufficient conditions for the existence of points $x_1,x_2,\ldots,x_p$ on the tree such that the following distance constraints are satisfied:

(DC) $$d(v_i,x_j) \le c_{ij},\ i = 1,2,\ldots,n, j = 1,\ldots,p, \quad (4.3)$$
$$d(x_j,x_k) \le b_{jk},\ j,k = 1,\ldots,p, \quad (4.4)$$

where c_{ij}, $i = 1,2,\ldots,n$, $j = 1,\ldots,p$ and $b_{jk} = b_{kj}$, $j,k = 1,\ldots,p$

are nonnegative numbers. These necessary and sufficient conditions for the existence of points $x_1,\dots,x_p$ satisfying (DC), called separation conditions, lead to a closed form expression for the optimum value z^* of problem (P). They also give an algorithm to construct a feasible solution to (4.3) and (4.4) if one exists. Substituting z^* in (4.1) and (4.2) and using this algorithm gives an optimal solution to the p-center problem with mutual communication.

In Section 4.1 we will derive the separation conditions in a way which differs from the proof given by FRANCIS, LOWE and RATLIFF [1978]. In Section 4.2 we show that the problem on an arbitrary graph is NP-hard.

4.1. Separation conditions

In Section 4.1 we will derive the separation conditions in a way which differs from the proof given by FRANCIS, LOWE & RATLIFF [1978]. In Section 4.2 we show that the problem on an arbitrary graph is NP-hard.

Construct the graph BC as follows. We have vertices E_i corresponding to v_i, $i = 1,2,\dots,n$, and N_j corresponding to x_j, $j = 1,2,\dots,p$, edges $\{E_i,N_j\}$ of length c_{ij}, $i = 1,2,\dots,n$, $j = 1,2,\dots,p$, and $\{N_j,N_k\}$ of length b_{jk}, $j,k = 1,\dots,p$, $j < k$. The graph BC for the case we have 3 vertices and $p = 3$ is given in Figure 4.1.1.

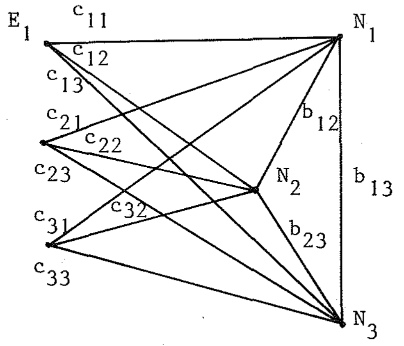

Figure 4.1.1. The graph BC.

Let $L(E_i,E_j)$ denote the length of a shortest path in BC from E_i to E_j. The separation conditions are given by

$$d(v_i,v_j) \le L(E_i,E_j), \quad i,j = 1,2,\dots,n,\ i \neq j.$$

<u>THEOREM 4.1.2</u>. *There are points* $x_1,x_2,\ldots,x_p$ *on* T *satisfying* (DC) *if and only if the separation conditions hold.*

<u>PROOF</u>. We use induction on the number p. In the case that p = 1, the theorem is equivalent to Corollary 1.1.6. Suppose the theorem is true for distance constraints with less than p points ($p\geq 2$). Consider the distance constraints (DC). Suppose we have found points $x_1,\ldots,x_{p-1}$ satisfying

$$d(v_i,x_j) \leq c_{ij}, \quad i = 1,2,\ldots,n,\ j = 1,2,\ldots,p-1,$$
$$d(x_j,x_k) \leq b_{jk},\ j,k = 1,\ldots,p-1.$$

According to the induction hypothesis there exists a point x_p such that

$$d(v_i,x_p) \leq c_{ip}, \quad i = 1,2,\ldots,n,$$
$$d(x_j,x_p) \leq b_{jp}, \quad j = 1,\ldots,p-1,$$

if and only if

$$d(v_i,v_j) \leq c_{ip} + c_{jp}, \quad i,j = 1,2,\ldots,n, i \neq j,$$
$$d(v_i,x_j) \leq c_{ip} + b_{jp}, \quad i = 1,2,\ldots,n,\ j = 1,\ldots,p-1,$$
$$d(x_j,x_k) \leq b_{jp} + b_{kp}, \quad j,k = 1,\ldots,p-1.$$

We conclude that there exist points $x_1,x_2,\ldots,x_p$ satisfying (DC) if and only if

$$d(v_i,v_j) \leq c_{ip} + c_{jp},\ i,j = 1,2,\ldots,n,\ i \neq j,$$

and there exist points $x_1,x_2,\ldots,x_{p-1}$ satisfying

(4.1.3) $$d(v_i,x_j) \leq \min\{c_{ij},c_{ip} + b_{jp}\}, \quad i = 1,2,\ldots,n,\ j = 1,\ldots,p-1,$$

(4.1.4) $$d(x_j,x_k) \leq \min\{b_{jk},b_{jp} + b_{kp}\}, \quad j,k = 1,\ldots,p-1.$$

Let BC' be the graph with vertices E'_i, $i = 1,2,\ldots,n$, N'_j, $j = 1,\ldots,p$ and edges $\{E'_i,N'_j\}$ of length $\min\{c_{ij},b_{jp} + c_{ip}\}$, $i = 1,2,\ldots,n$, $j = 1,\ldots,p-1$, $\{E'_i,N'_p\}$ of length c_{ip}, $i = 1,2,\ldots,n$, and $\{N'_j,N'_k\}$ of length

$\min\{b_{jk}, b_{jp} + b_{kp}\}$, $j,k = 1,\dots,p-1$, $j < k$. The graph BC' for the case $n = p = 3$ is given in Figure 4.1.5.

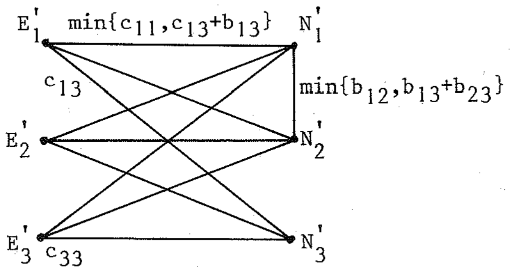
E_1'
$\min\{c_{11}, c_{13}+b_{13}\}$
N_1'
c_{13}
$\min\{b_{12}, b_{13}+b_{23}\}$
E_2'
N_2'
E_3'
c_{33}
N_3'

Figure 4.1.5. The graph BC'.

Let $L(E_i', E_j')$ denote the length of a shortest path from E_i' to E_j' in BC'. Then using the induction hypothesis on (4.1.3) and (4.1.4) and taking into account the constraints

$$d(v_i, v_j) \le c_{ip} + c_{jp}, \quad i,j = 1,2,\dots,n, \ i \ne j,$$

it follows that necessary and sufficient conditions for the existence of $x_1,\dots,x_p$ satisfying (DC) are

$$d(v_i, v_j) \le L(E_i', E_j'), \ i,j = 1,2,\dots,n, \ i \ne j.$$

We shall prove in Lemma 4.1.6 that $L(E_i, E_j) = L(E_i', E_j')$, which completes the proof. □

LEMMA 4.1.6. $L(E_i, E_j) = L(E_i', E_j')$ *for all* $i,j = 1,2,\dots,n$, $i \ne j$.

PROOF. Consider a path from E_i' to E_j' in BC'. We construct a path in BC of the same length. This proves $L(E_i, E_j) \le L(E_i', E_j')$. An edge $\{E_i', N_j'\}$, $j \ne p$ is replaced by the edges $\{E_i, N_p\}\{N_p, N_j\}$ if $c_{ip} + b_{jp} < c_{ij}$. Similarly we replace $\{E_j', N_k'\}$, $k \ne p$ by $\{E_j, N_p\}\{N_p, N_k\}$ if $b_{jp} + b_{kp} < b_{jk}$. All other edges are replaced by the corresponding edges. In this way we have constructed a path in BC of the same length.

Consider the shortest path from E_i to E_j in BC. We construct a path in BC' of the same length. This proves that $L(E_i', E_j') \le L(E_i, E_j)$. If the path contains an edge $\{N_j,N_k\}$ $(j,k \neq p)$, then we know that $b_{jk} \leq b_{jp} + b_{kp}$ and we take the corresponding edge in BC'. If the path contains an edge $\{E_i,N_j\}$ $(j \neq p)$, then $c_{ij} \leq c_{ip} + b_{jp}$ and we take the corresponding edge in BC'. If $\{E_i,N_p\}\{N_p,N_j\}$ occurs in the path, then $c_{ip} + b_{jp} \leq c_{ij}$ and these two edges are replaced by $\{E_i',N_j'\}$. If $\{N_j,N_p\}\{N_p,N_k\}$ occurs in the path, then $b_{jp} + b_{kp} \leq b_{jk}$ and these two edges are replaced by $\{N_j',N_k'\}$. The constructed path in BC' has the same length as the shortest path in BC. □

Let us return to problem (P). We construct the graph BC with $c_{ij} = 1/\alpha_{ij}$, $i = 1,2,\ldots,n$, $j = 1,\ldots,p$ and $b_{jk} = 1/\beta_{jk}$, $j,k = 1,\ldots,p$, $j < k$. It follows from the separation conditions that (4.1) and (4.2) are feasible if and only if

$$d(v_i,v_j) \leq zL(E_i,E_j),\ i,j = 1,2,\ldots,n,\ i \neq j.$$

We conclude that $z^* = \max_{i,j=1,2,\ldots,n,\ i \neq j} \left\{\frac{d(v_i,v_j)}{L(E_i,E_j)}\right\}$.

4.2. NP-Hardness

In this section we prove that the p-center problem with mutual communication on an arbitrary graph is NP-hard. It is sufficient to establish NP-completeness for the following decision problem:

CMC

Instance: A graph $G' = (V',E')$ with $V' = \{v_1',v_2',\ldots,v_n'\}$, $p \in \mathbb{N}$, weights $\alpha(v_i',j) \in \mathbb{Z}^+$, $i = 1,2,\ldots,n$, $j = 1,\ldots,p$, $\beta(j,k) \in \mathbb{Z}^+$, $j,k = 1,\ldots,p$, lengths $\ell(e) \in \mathbb{Z}^+$, $e \in E'$, $B \in \mathbb{Q}^+$.

Question: Is there a set $X = \{x_1,x_2,\ldots,x_p\}$ on G' such that

$$\alpha(v_i',j)d(v_i',x_j) \leq B, \qquad i = 1,\ldots,n,\ j = 1,\ldots,p,$$

$$\beta(j,k)\ d(x_j,x_k) \leq B, \qquad j,k = 1,\ldots,p\ ?$$

THEOREM 4.2.1. CMC *is* NP-*complete*.

PROOF. The problem belongs to the class *NP* since by using standard shortest path techniques one can test whether a given solution satisfies the bound B in polynomial time.

We shall reduce the problem CLIQUE to CMC. Example 4.2.5 shows the reduction for a special case. Let an instance of CLIQUE be given by $G = (V,E)$ with $V = \{v_1,v_2,\dots,v_n\}$ and c. The corresponding instance of CMC is defined by

$$V' = V \cup \bar{V} \cup \bar{\bar{V}} \cup \{s\} \cup \{\bar{s}\} \cup \{\bar{\bar{s}}\}, \text{ where } \bar{V} = \{\bar{v} | v \in V\} \text{ and } \bar{\bar{V}} = \{\bar{\bar{v}} | v \in V\},$$
$$E' = \{\{s,v_i\},\{\bar{s},\bar{v}_i\},\{\bar{\bar{s}},\bar{\bar{v}}_i\},\{v_i,\bar{v}_i\},\{\bar{v}_i,\bar{\bar{v}}_i\} | i = 1,\dots,n\} \cup$$
$$\{\{v_i,\bar{\bar{v}}_j\} | \{v_i,v_j\} \in E,\ i < j\},$$
$$p = 3c,$$
$$\alpha(s,j) = \alpha(\bar{s},j+c) = \alpha(\bar{\bar{s}},j+2c) = 1,\ j = 1,\dots,c,$$
$$\beta(j,j+c) = \beta(j+c,j+2c) = 1,\ j = 1,\dots,c,\ \beta(i,j+2c) = 1,\ i,j = 1,\dots,c,$$
$$i < j, \text{ all other weights are zero},$$
$$\ell(e) = 1,\ e \in E',$$
$$B = 1.$$

We claim that $G = (V,E)$ contains a clique of size c if and only if there exist $x_1,x_2,\dots,x_{3c}$ on $G' = (V',E')$ such that

(4.2.2) $\quad d(s,x_j) \le 1,\ d(\bar{s},x_{j+c}) \le 1,\ d(\bar{\bar{s}},x_{j+2c}) \le 1,\ j = 1,\dots,c,$

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

(4.2.4) $\quad d(x_i,x_{j+2c}) \le 1,\quad i,j = 1,\dots,c,\quad i < j.$

Let $x_1,x_2,\dots,x_{3c}$ be points on $G' = (V',E')$ satisfying (4.2.2), (4.2.3) and (4.2.4). Since $d(s,\bar{s}) = d(\bar{s},\bar{\bar{s}}) = 3$ we have from (4.2.2) and (4.2.3) that $x_j \in V$, say $x_j = v(j)$, $x_{j+c} = \bar{v}(j)$ and $x_{j+2c} = \bar{\bar{v}}(j)$. It follows from (4.2.4) that $\{v(i),\bar{\bar{v}}(j)\} \in E'$, i.e. $\{v(i),v(j)\} \in E$, $i < j$. Hence $\{v(1),\dots,v(c)\}$ is a clique of $G = (V,E)$.

Let $\{v(1),\dots,v(c)\}$ be a clique of size c in $G = (V,E)$. Define $x_j = v(j)$, $x_{j+c} = \bar{v}(j)$ and $x_{j+2c} = \bar{\bar{v}}(j)$, $j = 1,\dots,c$. Then $x_1,x_2,\dots,x_{3c}$ satisfy (4.2.2), (4.2.3) and (4.2.4). $\square$

<u>EXAMPLE 4.2.5</u>. Let the graph $G = (V,E)$ be the graph given in Figure 4.2.6.

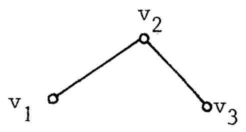

Figure 4.2.6. The graph $G = (V,E)$.

The corresponding graph $G' = (V',E')$ of CMC is given by Figure 4.2.7.

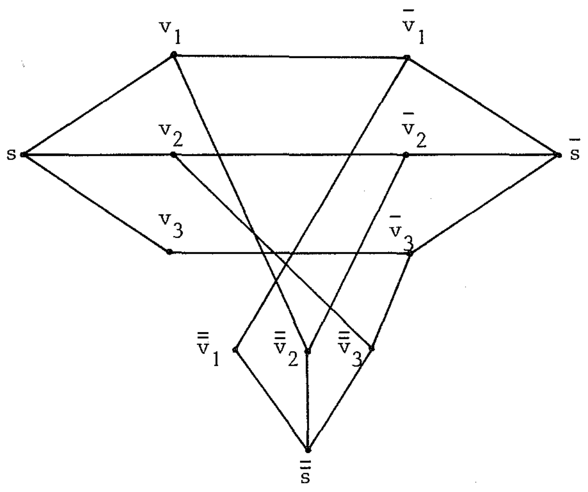

Figure 4.2.7. The graph $G' = (V',E')$.

CHAPTER 5

THE NONLINEAR ROUND-TRIP p-CENTER AND COVERING PROBLEMS

Let $T = (V,E)$ be a tree. Consider the problem of a transportation service which has to transport goods from point a_i to point b_i on the tree; let us call this job J_i, $i = 1,2,\ldots,m$. The distance a vehicle located at a depot at point x has to travel to execute job J_i and return to its depot is given by the *round-trip distance* $d(x,a_i) + d(a_i,b_i) + d(b_i,x)$. We assume that the cost associated with executing job J_i is measured by a strictly increasing continuous function f_i of the travelled distance, $i = 1,2,\ldots,m$. Due to budget constraints the transportation service can only build p vehicle depots to execute the jobs. The *nonlinear round-trip p-center problem* is to locate p vehicle depots on the tree so as to minimize the maximum round-trip cost.

For a set of point $X = \{x_1,x_2,\ldots,x_p\}$ we define $D(X,J_i) = \min_{x\in X}\{d(a_i,x) + d(x,b_i)\}$. The round-trip p-center problem can be formulated as

$$\min_{X\subset T:|X| = p}\{\max_{1\le i\le m}\{f_i(D(X,J_i))\}\}.$$

The round-trip 1-center problem with linear functions f_i was solved by CHAN & FRANCIS [1976]. The round-trip 1-center problem in the rectilinear plane was solved by CHAN & HEARN [1977]. We will return to this rectilinear problem in Chapter 6.

The *round-trip covering problem* plays a similar role in connection with the round-trip p-center problem as the covering problem in connection with the p-center problem as was indicated in Section 1.2. Given a distance r, $r \ge \max_{1\le i\le m}\{d(a_i,b_i)\}$ the round-trip covering problem is given by

$$\min\{|X|\,\big|\,X \subset T, D(X,J_i) \le r,\ i = 1,2,\ldots,m\}.$$

In Section 5.1 we develop a polynomial-time algorithm to solve the round-trip covering problem. We also define a dual problem to the round-trip covering problem and give a constructive and a theoretical proof of strong duality. In Section 5.2 we formulate a dual problem to the round-trip p-center problem and with the help of the duality proved in Section 5.1 we will show that strong duality holds. Just as in the p-center problem this duality result provides a polynomially bounded set of values to which the optimal value belongs. This means that we can solve the round-trip p-center problem as a sequence of round-trip covering problems.

5.1. The round-trip covering problem

Given m pairs of vertices $J_i = \{a_i,b_i\}$ and m positive numbers r_i satisfying $d(a_i,b_i) \le r_i$, $i = 1,2,\dots,m$, a more general round-trip covering problem than defined previously is the problem to find the minimum number of points on the tree such that each pair $\{a_i,b_i\}$ is *covered within distance* r_i, i.e.,

$$\min\{|X|\,|X \subset T, D(X,J_i) \le r_i,\ i = 1,2,\dots,m\}.$$

During each iteration of the algorithm to solve this problem the original tree is partitioned into two subgraphs: one green, the other brown. The green subgraph is always a tree, denoted by GT, while the brown subgraph consists of one or more subtrees of the original tree T, each of which is "rooted" at a vertex of the green tree. By convention a root t will be in both GT and the associated brown subtree, denoted by BT(t). Given a brown subtree BT(t) we have the following sets and values assigned to the root t:

- I_t^1 denotes the set of all indices i such that either a_i or b_i belongs to BT(t), and such that $\{a_i,b_i\}$ is not covered within distance r_i by any facility x located so far at BT(t), i.e. $d(a_i,x) + d(b_i,x) > r_i$.
- I_t^2 denotes the set of all indices i such that both a_i and b_i belong to BT(t), and such that $\{a_i,b_i\}$ is not covered within distance r_i by any facility x located so far at BT(t).
- $c_i(t) = \frac{1}{2}[r_i-d(a_i,t)-d(b_i,t)]$ for every $i \in I_t^2$.

The following observation clarifies the meaning of the nonnegative value $c_i(t)$. Since $\{a_i,b_i\}$ is not covered by a facility located at BT(t), it must be covered by a facility x not located at BT(t), i.e., $d(a_i,x) + d(b_i,x) \le r_i$. Since $a_i,b_i \in BT(t)$ we have $d(a_i,x) + d(b_i,x) =$

$d(a_i,t) + d(b_i,t) + 2d(t,x)$. It follows that x covers $\{a_i,b_i\}$ within distance r_i if and only if $d(t,x) \le c_i(t)$.

- $c_0(t) = \min_{i \in I_t^2}\{c_i(t)\}$, $c_0(t) = \infty$ if $I_t^2 = \emptyset$.
- $d(t)$ denotes the distance from t to the nearest facility located at BT(t), $d(t) = \infty$ if there is no facility located at BT(t).

Let k denote the number of facilities located so far and let $I \subseteq \{1,2,\ldots,m\}$ denote the set of indices for which we have not yet determined whether $\{a_i,b_i\}$ is covered within distance r_i by a facility located so far.

At each iteration we select a tip vertex t of GT. If GT = {t}, then $k := k+1$, $x_k = t$ and we stop. If GT $\neq$ {t}, then let ad(t) be the vertex of GT adjacent to t. We delete {t,ad(t)} from GT and add this edge together with BT(t) to BT(ad(t)) to form the new BT(ad(t)). If $c_0(t) < d(t,ad(t))$, then $k := k+1$ and x_k is defined to be the point on {t,ad(t)} at distance $c_0(t)$ from t. We calculate the new sets and values $I^1_{ad(t)}$, $I^2_{ad(t)}$, $d(ad(t))$, $c_i(ad(t))$ for all $i \in I^2_{ad(t)}$ and the set I. If $I = \emptyset$, then we stop. Otherwise we continue with the next iteration.

Let us now consider the problem of finding the sets and values assigned to ad(t) during an iteration.

Case A Assume $c_0(t) < d(t,ad(t))$.

Case A.1 Suppose $\{a_i,b_i\}$ is included in BT(ad(t)). There are three possibilities.

Case A.1.1 Both points are in BT(t), i.e., $i \in I_t^2$. Since

$$\begin{aligned} d(a_i,x_k) + d(b_i,x_k) &= d(a_i,t) + d(b_i,t) + 2d(t,x_k) \\ &= d(a_i,t) + d(b_i,t) + 2c_0(t) \\ &\le d(a_i,t) + d(b_i,t) + 2c_i(t) = r_i, \end{aligned}$$

$\{a_i,b_i\}$ is covered by x_k within distance r_i, and i can be deleted from I.

Case A.1.2 Both points are in the old BT(ad(t)), i.e., $i \in I^2_{ad(t)}$. $\{a_i,b_i\}$ is not covered by a facility located at the old BT(ad(t)), but it is covered by a facility located at the new BT(ad(t)) if and only if it is covered by x_k. Since

$$\begin{aligned} d(a_i,x_k) + d(b_i,x_k) &= d(a_i,ad(t)) + d(b_i,ad(t)) + 2d(ad(t),x_k) \\ &= d(a_i,ad(t)) + d(b_i,ad(t)) + 2d(t,ad(t)) - \\ &\quad 2c_0(t) \\ &= r_i + 2[d(t,ad(t)) - c_0(t) - c_i(ad(t))], \end{aligned}$$

we have $d(a_i,x_k) + d(b_i,x_k) \le r_i$ if and only if $d(t,ad(t)) - c_0(t) \le c_i(ad(t))$.
If this condition holds, then i can be deleted from I, else $i \in I^2_{ad(t)}$.

Case A.1.3 One point is in BT(t), the other point is in the old BT(ad(t)), i.e., $i \in I^1_t \cap I^1_{ad(t)}$. Since

$$d(a_i,x_k) + d(b_i,x_k) = d(a_i,b_i) \le r_i,$$

the pair $\{a_i,b_i\}$ is covered within distance r_i by x_k and i can be deleted from I.

The values of $c_i(ad(t))$ for those i which are not deleted remain the same. We have $d(ad(t)) = \min\{d(ad(t)),d(t,ad(t))-c_0(t)\}$.

Case A.2 Suppose either a_i or b_i is contained in BT(ad(t)). There are two possibilities.

Case A.2.1 The point is in BT(t), i.e., $i \in I^1_t \backslash I^1_{ad(t)}$.
Since

$$d(a_i,x_k) + d(b_i,x_k) = d(a_i,b_i) \le r_i,$$

i can be deleted from I.

Case A.2.2 The point is in the old BT(ad(t)), i.e., $i \in I^1_{ad(t)} \backslash I^1_t$.
$\{a_i,b_i\}$ is not covered by any facility in the old BT(ad(t)) but is covered by a facility in the new BT(ad(t)) if and only if it is covered by x_k. Since

$$\begin{aligned} d(a_i,x_k) + d(b_i,x_k) &= d(a_i,ad(t)) + d(b_i,ad(t)) + 2d(ad(t),x_k) \\ &= d(a_i,b_i) + 2[d(ad(t),t) - c_0(t)] \end{aligned}$$

we have $d(a_i,x_k) + d(b_i,x_k) \le r_i$ if and only if $d(ad(t),t) - c_0(t) \le \frac{1}{2}[r_i-d(a_i,b_i)]$.
If this condition holds, then i can be deleted from I, else $i \in I^1_{ad(t)}$.

Case B Assume $c_0(t) \ge d(t,ad(t))$.

Case B.1 Suppose $\{a_i,b_i\}$ is included in BT(ad(t)). There are three possibilities.

Case B.1.1 Both points are in BT(t), i.e., $i \in I^2_t$.
$\{a_i,b_i\}$ is not covered by any facility located at BT(t), but is covered by a facility at the new BT(ad(t)) if and only if is

is covered by the facility x located closest to ad(t) at the old BT(ad(t)). Since

$$\begin{aligned} d(a_i,x) + d(b_i,x) &= d(a_i,t) + d(b_i,t) + 2[d(t,ad(t)) + \\ &\quad + d(ad(t),x)] \\ &= r_i + 2[d(t,ad(t)) + d(ad(t)) - c_i(t)], \end{aligned}$$

we have $d(a_i,x) + d(b_i,x) \le r_i$ if and only if $d(t,ad(t)) + d(ad(t)) \le c_i(t)$. If this condition holds, then i can be deleted from I, else $i \in I^2_{ad(t)}$ and $c_i(ad(t)) = c_i(t) - d(t,ad(t))$.

Case B.1.2 Both points are in the old BT(ad(t)), i.e., $i \in I^2_{ad(t)}$. Equivalent to Case B.1.1 we find that if $d(t,ad(t)) + d(t) \le$ $\le c_i(ad(t))$, then i can be deleted from I, else $i \in I^2_{ad(t)}$ and $c_i(ad(t))$ remains the same.

Case B.1.3 One point is in BT(t), the other point is in the old BT(ad(t)), i.e., $i \in I^1_t \cap I^1_{ad(t)}$. $\{a_i,b_i\}$ is not covered by a facility located at the new BT(ad(t)). Hence $i \in I^2_{ad(t)}$ and $c_i(ad(t)) = \frac{1}{2}[r_i-d(a_i,ad(t))-d(b_i,ad(t))] = \frac{1}{2}[r_i-d(a_i,b_i)]$.

We have $d(ad(t)) = \min\{d(ad(t)),d(t,ad(t))+d(t)\}$.

Case B.2 Suppose either a_i or b_i is contained in BT(ad(t)). There are two possibilities.

Case B.2.1 The point is in BT(t), i.e., $i \in I^1_t \backslash I^1_{ad(t)}$. $\{a_i,b_i\}$ is not covered by a facility at BT(t) and is covered by a facility at BT(ad(t)) if and only if it is covered by the closest facility x to ad(t). Since

$$\begin{aligned} d(a_i,x) + d(b_i,x) &= d(a_i,ad(t)) + d(b_i,ad(t)) + 2d(ad(t),x) \\ &= d(a_i,b_i) + 2d(ad(t)), \end{aligned}$$

we have $d(a_i,x) + d(b_i,x) \le r_i$ if and only if $d(ad(t)) \le \frac{1}{2}[r_i-d(a_i,b_i)]$. If this condition holds, then i can be deleted from I, else $i \in I^1_{ad(t)}$.

Case B.2.2 The point is in the old BT(ad(t)), i.e., $i \in I^1_{ad(t)} \backslash I^1_t$. Equivalent to case B.2.1 we find that if $d(t) + d(t,ad(t)) \le$ $\le \frac{1}{2}[r_i-d(a_i,b_i)]$, then i can be deleted from I, else $i \in I^1_{ad(t)}$.

Each iteration requires O(m) calculations. Therefore the total complexity is O(mn), where n is the number of vertices in the tree. We now proceed to prove the correctness of the algorithm.

LEMMA 5.1.1. *The algorithm constructs a feasible solution* X *to the round-trip covering problem with* $|X| \le m$.

PROOF. An index i is deleted from $\{1,2,\dots,m\}$ during an iteration whenever $\{a_i,b_i\}$ is covered within distance r_i by a facility located so far. The algorithm stops when all indices have been deleted or when the green tree consists of a single point t. In the former case all pairs are covered. In the latter case, the set I of pairs which have not been deleted is equal to I_t^2. Since $c_i(t) = \frac{1}{2}[r_i - d(a_i,t) - d(b_i,t)] \ge 0$ for all $i \in I_t^2$, the pair $\{a_i,b_i\}$ is covered by t, $i \in I_t^2$. □

Let the algorithm construct a feasible solution $X = \{x_1,x_2,\dots,x_p\}$. We define t_i to be the tip vertex of GT chosen by the algorithm in the iteration during which x_i was located. Note that after this iteration all pairs $\{a_k,b_k\}$ which have at least one point in $BT(t_i)$ are covered. If x_i is located at distance $c_0(t)$ from t_i on $\{t_i,ad(t_i)\}$, then without loss of generality we assume for notational convenience that $c_i(t_i) = c_0(t_i)$. If x_p is located when the green tree consists of a single point t_p, then we assume without loss of generality that $p \in I_{t_p}^2$.

LEMMA 5.1.2. *Under the assumption made above we have* $d(a_i,b_j) + d(a_j,b_i) > r_i + r_j$ *for all* $i,j = 1,\dots,p$, $i < j$.

PROOF. First assume that x_p is located when the green tree consists of a single point t_p. Since $p \in I_{t_p}^2$ we know that $\{a_p,b_p\}$ is not covered by any point x_i, $i = 1,2,\dots,p-1$. This implies:

(1) Both a_p and b_p are not in $BT(t_i)$.

(2) $d(a_p,x_i) + d(b_p,x_i) > r_p$, $i = 1,2,\dots,p-1$.

Since both a_i and b_i are in $BT(t_i)$ and $d(a_i,x_i) + d(b_i,x_i) = r_i$, we have

$$\begin{aligned} d(a_i,b_p) + d(b_i,a_p) &= d(a_i,x_i) + d(b_i,x_i) + d(a_p,x_i) + d(b_p,x_i) \\ &= r_i + d(a_p,x_i) + d(b_p,x_i) > r_i + r_p. \end{aligned}$$

If both facilities x_i and x_j $(i<j)$ are located in the case that GT consists of at least one edge we consider two cases

Case 1. If $BT(t_i) \subseteq BT(t_j)$, then $a_j,b_j \notin BT(t_i)$ and $\{a_j,b_j\}$ is not covered by x_i, i.e., $d(a_j,x_i) + d(b_j,x_i) > r_j$. We have

$$\begin{aligned} d(a_i,b_j) + d(a_j,b_i) &= d(a_i,x_i) + d(b_i,x_i) + d(a_j,x_i) + d(b_j,x_i) \\ &= r_i + d(b_j,x_i) + d(a_j,x_i) > r_i + r_j. \end{aligned}$$

<u>Case</u> 2. If $BT(t_i) \nsubseteq BT(t_j)$, then we have

$$\begin{aligned} d(a_i,b_j) + d(a_j,b_i) &= d(a_i,x_i) + d(x_i,x_j) + d(x_j,b_j) + d(a_j,x_j) + \\ &\quad + d(x_j,x_i) + d(x_i,b_i) \\ &= r_i + r_j + 2d(x_i,x_j) > r_i + r_j. \quad \square \end{aligned}$$

At first sight Lemma 5.1.2 may seem strange because the condition depends on which point is called $a_i(a_j)$ and which point is called $b_i(b_j)$. This seems to conflict with the fact that if we interchange the role of a_i and b_i the problem does not change. We can prove however that for $i,j \in \{1,2,\ldots,p\}$, $i \neq j$ the shortest paths $P(a_i,b_i)$ and $P(a_j,b_j)$ do not intersect so that $d(a_i,b_j) + d(a_j,b_i) = d(a_i,a_j) + d(b_i,b_j)$.

The following lemma indicates when two pairs can be covered by the same facility.

<u>LEMMA 5.1.3</u>. *There is an* $x \in T$ *such that* $d(a_i,x) + d(x,b_i) \le r_i$ *and* $d(a_j,x) + d(x,b_j) \le r_j$, *where* $r_i \ge d(a_i,b_i)$ *and* $r_j \ge d(a_j,b_j)$, *if and only if* $d(a_j,b_i) + d(a_i,b_j) \le r_i + r_j$.

<u>PROOF</u>. If $d(a_i,x) + d(x,b_i) \le r_i$ and $d(a_j,x) + d(x,b_j) \le r_j$, then $d(a_i,b_j) + d(a_j,b_i) \le d(a_i,x) + d(x,b_j) + d(a_j,x) + d(x,b_i) \le r_i + r_j$.

Let $d(a_i,b_j) + d(a_j,b_i) \le r_i + r_j$. If $P(a_i,b_i) \cap P(a_j,b_j) \neq \emptyset$, then any point x in the intersection satisfies $d(a_i,x) + d(x,b_i) = d(a_i,b_i) \le r_i$ and $d(a_j,x) + d(x,b_j) = d(a_j,b_j) \le r_j$. If $P(a_i,b_i) \cap P(a_j,b_j) = \emptyset$, then define $c_i(c_j)$ to be the point on $P(a_i,b_i)(P(a_j,b_j))$ closest to $P(a_j,b_j)(P(a_i,b_i))$. We have

$$\begin{aligned} d(c_i,c_j) &= \tfrac{1}{2}[d(a_i,b_j) + d(a_j,b_i) - d(a_i,b_i) - d(a_j,b_j)] \\ &\le \tfrac{1}{2}[r_i - d(a_i,b_i)] + \tfrac{1}{2}[r_j - d(a_j,b_j)]. \end{aligned}$$

It follows from Corollary 1.1.6 that there exists a point $x \in T$ such that $d(c_i,x) \le \frac{1}{2}[r_i - d(a_i,b_i)]$ and $d(c_j,x) \le \frac{1}{2}[r_j - d(a_j,b_j)]$. Then

$$\begin{aligned} d(a_i,x) + d(x,b_i) &\le d(a_i,c_i) + d(c_i,b_i) + 2d(c_i,x) \\ &= d(a_i,b_i) + 2d(c_i,x) \le r_i, \end{aligned}$$

and similarly we prove $d(a_j,x) + d(x,b_j) \le r_j$. $\square$

THEOREM 5.1.4. *The solution* $X = \{x_1, x_2, \ldots, x_p\}$ *constructed by the algorithm is an optimal solution.*

PROOF. We know from Lemma 5.1.2 that $d(a_i,b_j) + d(a_j,b_i) > r_i + r_j$, $i = 1,2,\ldots,p$, $i < j$. It follows from Lemma 5.1.3 that any feasible solution to the covering problem needs at least p points. Since X is a feasible solution and $|X| = p$, X is an optimal solution. □

Define the dual problem to the round-trip covering problem by

$$\max\{|I| \,\big|\, I \subseteq \{1,2,\ldots,m\},\ d(a_i,b_j) + d(a_j,b_i) > r_i + r_j,\ i,j \in I,\ i \neq j\}.$$

We have given a constructive proof of strong duality. Strong duality can also be derived as follows.

Define the subtree T_i by $T_i = \{x \in T \,|\, d(a_i,x) + d(b_i,x) \leq r_i\}$, $i = 1,2,\ldots,m$. Since $d(a_i,b_i) \leq r_i$ the subtree T_i is nonempty. Construct the subtree intersection graph G. It follows from Lemma 5.1.3 that the round-trip covering problem is equivalent to finding a minimum clique cover in the chordal graph G. In Section 1.2 we showed that the cardinality of a minimum clique cover is equal to the cardinality of a maximum independent set. Finding a maximum independent set is precisely the dual problem.

5.2. The round-trip p-center problem

The round-trip p-center problem is given by

$$\min_{X \subset T:\,|X|=p}\{\max_{1\leq i\leq m}\{f_i(D(X,J_i))\}\},$$

or equivalently

$$\min\{r \,|\, f_i(D(X,J_i) \leq r,\ i = 1,2,\ldots,m,\ X \subseteq T,\ |X| = p\},$$

where $f_i : \mathbb{R}^+ \to \mathbb{R}^+$ is a strictly increasing continuous function, $i = 1,2,\ldots,m$. Define $\delta_i = \max\{d(a_i,x) + d(x,b_i) \,|\, x \in T\}$, $\alpha = \max_{1\leq i\leq m}\{f_i(d(a_i,b_i))\}$ and $\eta = \min_{1\leq i\leq m}\{f_i(\delta_i)\}$. We assume without loss of generality that $\alpha < \eta$. If $\alpha = f_s(d(a_s,b_s)) \geq f_t(\delta_t) = \eta$, then $f_s(D(X,J_s)) \geq f_t(D(X,J_t))$ for all X, and the constraint corresponding to f_t can be deleted from the constraint set without changing the optimum.

Define $\varepsilon_{ij} = (f_i^{-1} + f_j^{-1})^{-1}(d(a_i,b_j) + d(a_j,b_i))$. Note that $(f_i^{-1} + f_j^{-1})^{-1}$ is a strictly increasing function. We now formulate the dual problem to the round-trip p-center problem. The dual problem is given by

$$\max_{I\subseteq\{1,2,\ldots,m\}:|I|=p+1}\{\max\{\alpha,\min_{i,j\in I,i\neq j}\{\varepsilon_{ij}\}\}\}.$$

The next theorem states that we have strong duality.

THEOREM 5.2.1. $\max_{I\subseteq\{1,2,\ldots,m\}:|I|=p+1}\{\max\{\alpha,\min_{i,j\in I,i\neq j}\{\varepsilon_{ij}\}\}\} =$

$$\min_{X\subset T:|X|=p}\{\max_{1\le i\le m}\{f_i(D(X,J_i))\}\}.$$

PROOF. Clearly the optimum value of both problems is at least α.

Let $r \ge \alpha$. Then

$$\{\min_{X\subset T:|X|=p}\{\max_{1\le i\le m}\{f_i(D(X,J_i))\}\}\} \le r \ (r\ge\alpha) \Longleftrightarrow$$

$$\{\min\{|X|\,|X \subset T, \{\max_{1\le i\le m}\{f_i(D(X,J_i))\}\} \le r,\ r \ge \alpha\} \le p \Longleftrightarrow$$

$$\{\min\{|X|\,|X \subset T,\ D(X,J_i) \le f_i^{-1}(r),\ i = 1,2,\ldots,m,\ r \ge \alpha\}\} \le p \Longleftrightarrow$$

(5.2.2) $$\{\max\{|I|\,|I \subseteq \{1,2,\ldots,m\},\ d(a_i,b_j) + d(a_j,b_i) > (f_i^{-1} + f_j^{-1})(r),\ i,j \in I,\ i \neq j,\ r \ge \alpha\}\} \le p \Longleftrightarrow$$

(5.2.3) $$\{\max\{|I|\,| \subseteq \{1,2,\ldots,m\},\ \varepsilon_{ij} > r,\ i,j \in I,\ i \neq j,\ r \ge \alpha\}\} \le p \Longleftrightarrow$$

$$\{\max_{I\subseteq\{1,2,\ldots,m\}:|I|=p+1}\{\max\{\alpha,\min_{i,j\in I,i\neq j}\{\varepsilon_{ij}\}\}\}\} \le r.$$

(5.2.2) follows from the duality result proved in Section 5.1, (5.2.3) follows from the fact that $(f_i^{-1} + f_j^{-1})^{-1}$ is a strictly increasing function.
□

The strong duality result for the round-trip p-center problem shows that there are only $O(m^2)$ possible values for the optimum value of the problem. If we assume that inverses can be calculated in unit time, then the round-trip p-center problem can be solved in $O(m \max\{m,n\}\log m)$ time as follows. Calculate the values α and ε_{ij}, $i,j = 1,2,\ldots,m$, $i \neq j$ and arrange these values in a list L in nondecreasing order. This takes

$O(m^2 \log m)$ time. For any $r \geq \alpha$ in L the round-trip covering algorithm can be applied to find the optimum value of the round-trip covering problem with distance r. We perform a bisection search on L to find the minimum value r_p for which the answer to the round-trip covering problem is p. This value r_p which is the optimum value of the round-trip p-center problem can be found in $O(mn \log m)$ time.

CHAPTER 6

FARKAS' LEMMA IN RECTILINEAR LOCATION PROBLEMS

FARKAS [1902] proved what is now known as a fundamental result in linear programming:

THEOREM 6.1. *Let* A *be a matrix, and let* b *be a vector. Then there exists a nonnegative vector* x *such that* $Ax = b$ *if and only if* $yb \geq 0$ *whenever* $yA \geq 0$ *for any vector* y. □

In this chapter we will use an equivalent formulation of Theorem 6.1:

THEOREM 6.2. (Farkas' Lemma) *The system of linear inequalities* $Ax \leq b$ *has a solution* x *if and only if* $yb \geq 0$ *whenever* $yA = 0$ *for any nonnegative vector* y. □

In Section 6.1 we shall show that various rectilinear min-max location problems can be formulated as

$$\text{determine } z^* = \min\{z \mid Ax \leq b(z)\},$$

where A is a matrix and b(z) is a vector each component of which is a function of the variable z. We then use Farkas' Lemma to derive a necessary and sufficient condition for the system $Ax \leq b(z)$ to have a solution x. This condition takes on the form $z \geq c$, where c is a constant depending only on the problem data. It is clear that in this case $z^* = c$.

In Section 6.2 we consider an extension of Farkas' Lemma proved by FOURIER [1826], KUHN [1956], and MOTZKIN [1936] (see also STOER & WITZGALL [1970]):

THEOREM 6.3. *Given are matrices* A_1 *and* A_2 *and vectors* b_1 *and* b_2. *Then there exists a vector* x *such that* $A_1x < b_1$ *and* $A_2x \leq b_2$ *if and only if for all nonnegative vectors* y_1 *and* y_2 *one has: if* $y_1A_1 + y_2A_2 = 0$, *then* $y_1b_1 + y_2b_2 \geq 0$, *and if furthermore* $y_1 \neq 0$, *then* $y_1b_1 + y_2b_2 > 0$. □

Given are n points (a_i,b_i), $i = 1,2,\dots,n$ in the plane. A point (x_1,y_1) is *dominated in position* k if there is a point (x,y) such that

$$|a_i-x| + |b_i-y| \le |a_i-x_1| + |b_i-y_1|, \text{ for } i = 1,2,\dots,n$$

$$|a_k-x| + |b_k-y| < |a_k-x_1| + |b_k-y_1|.$$

A point which can not be dominated in any position is called an *efficient point*. We shall apply Theorem 6.3 to find a characterization of efficient points in the plane. This characterization was used in CHALMET, FRANCIS & KOLEN [1981] to give an $O(n \log n)$ algorithm to find the set of all efficient points in the plane. Efficient points play a role in multiple objective problems (WENDEL, HURTER & LOWE [1977]) and in sensitivity analyses for single objective location (CHALMET, FRANCIS & KOLEN [1981]).

6.1. Min-max location problems

The best way to show how one can use Farkas' Lemma in solving rectilinear min-max location problems is by considering the following simple example, which is known as the 1-*center problem*:

$$\min_{x,y}\{\max_{1\le i\le n}\{w_i(|a_i-x| + |b_i-y|) + h_i\}\},$$

where (a_i,b_i) is a point in the plane, and w_i,h_i are positive integers, $i = 1,2,\dots,n$. We can reformulate this problem as

minimize z

(6.1.1) subject to $w_i(|a_i-x| + |b_i-y|) + h_i \le z$, $i = 1,2,\dots,n$.

Define $(\hat{p},\hat{q}) = (p+q,-p+q)$. Then (6.1.1) is equivalent to $w_i(\max\{|\hat{x}-\hat{a}_i|, |\hat{y}-\hat{b}_i|\}) + h_i \le z$. It follows that we can solve the 1-center problem in the plane using rectilinear distances as two independent problems on the line of the following type:

minimize z

subject to $|x-a_i| \le (z-h_i)/w_i$, $i = 1,2,\dots,n$, or equivalently

$$\text{minimize} \quad z$$

$$\text{subject to } x \le \min_i\{a_i + (z-h_i)/w_i\},$$
$$- x \le \min_j\{-a_j + (z-h_j)/w_j\}.$$

Using Farkas' Lemma we have

$$\exists x \begin{bmatrix} 1 \\ -1 \end{bmatrix} x \le \begin{bmatrix} \min_i\{a_i+(z-h_i)/w_i\} \\ \min_j\{-a_j+(z-h_j)/w_j\} \end{bmatrix} \iff$$

$$\forall y_1,y_2 \ge 0[y_1-y_2 = 0 \Rightarrow y_1\min_i\{a_i+(z-h_i)/w_i\} +$$

$$+ y_2\min_j\{-a_j+(z-h_j)/w_j\} \ge 0] \iff$$

$$\min_i\{a_i + (z-h_i)/w_i\} + \min_j\{-a_j + (z-h_j)/w_j\} \ge 0 \iff$$

$$z \ge (|a_i-a_j| + h_i/w_i + h_j/w_j)w_iw_j/(w_i+w_j), \ i,j = 1,2,\dots,n, \ i < j.$$

It follows that the optimum value z^* of the 1-center problem is given by

$$z^* = \max_{i,j=1,2,\dots,n,i<j}\{(|a_i-a_j| + h_i/w_i + h_j/w_j)w_iw_j/(w_i+w_j)\}.$$

This simple example has all the ingredients we described at the beginning of this chapter: the problem can be formulated as $\min\{z|Ax \le b(z)\}$, and the necessary and sufficient condition for the existence of x satisfying $Ax \le b(z)$ takes on the form $z \ge c$, which leads to the optimum value z^* given by $z^* = c$.

The second problem we consider is the *round-trip 1-center problem* in the rectilinear plane. As indicated in Chapter 5 this problem was solved by CHAN & HEARN [1977]. It is given by

$$\min_{x,y}\{\max_{1\le i\le n}\{w_i(|x-a_i| + |y-b_i| + |x-c_i| + |y-d_i|) + h_i\}\},$$

where (a_i,b_i), (c_i,d_i) are points in the plane and w_i,h_i are positive integers, $i = 1,2,\dots,n$. An equivalent formulation of the problem is given by:

$$\text{minimize} \quad z$$

(6.1.2) $$\text{subject to } w_i(|x-a_i| + |y-b_i| + |x-c_i| + |y-d_i|) + h_i \le z, \quad i = 1,2,\dots,n.$$

Each of the constraints in (6.1.2) can be written as a system of sixteen linear inequalities. This leads to the following formulation of the problem:

$$\text{minimize} \quad z$$

$$\text{subject to } \tfrac{1}{2}\max_i\{a_i+b_i+c_i+d_i-(z-h_i)/w_i \le x+y \le \tfrac{1}{2}\min_j\{a_j+b_j+c_j+d_j+(z-h_j)/w_j\},$$

$$\tfrac{1}{2}\max_i\{a_i+c_i-b_i-d_i-(z-h_i)/w_i\} \le x-y \le \tfrac{1}{2}\min_j\{a_j+c_j-d_j-b_j+(z-h_j)/w_j\},$$

$$\tfrac{1}{2}\max_i\{a_i+c_i+|b_i-d_i|-(z-h_i)/w_i\} \le x \le \tfrac{1}{2}\min_j\{a_j+c_j-|b_j-d_j|+(z-h_j)w_j\},$$

$$\tfrac{1}{2}\max_i\{b_i+d_i+|a_i-c_i|-(z-h_i)/w_i\} \le y \le \tfrac{1}{2}\min_j\{b_j+d_j-|a_j-c_j|+(z-h_j)/w_j\},$$

$$z \ge \max_i\{w_i(|a_i-c_i|+|b_i-d_i|)+h_i\}.$$

Using Farkas' Lemma we find that x,y satisfying the system of inequalities

$$a_1 \le x+y \le a_2,$$
$$b_1 \le x-y \le b_2,$$
$$c_1 \le x \le c_2,$$
$$d_1 \le y \le d_2,$$

exist if and only if the following twelve conditions are satisfied:

$$\begin{array}{lll} a_1 \le a_2, & c_1 \le \tfrac{1}{2}(a_2+b_2), & a_1 \le c_2+d_2, \\ b_1 \le b_2, & c_2 \ge \tfrac{1}{2}(a_1+b_1), & a_2 \ge c_1+d_1, \\ c_1 \le c_2, & d_1 \le \tfrac{1}{2}(a_2-b_1), & b_1 \le c_2-d_1, \\ d_1 \le d_2, & d_2 \ge \tfrac{1}{2}(a_1-b_2), & b_2 \ge c_1-d_2. \end{array}$$

Applying this to the round-trip location problem leads to the following seven necessary and sufficient conditions:

$$z \ge \max_i\{w_i(|a_i-c_i|+|b_i-d_i|)+h_i\},$$

$$z \ge \max_{1\le i\le j\le n}\{(|a_i+c_i-a_j-c_j|+|b_i+d_i-b_j-d_j|+ \\ + h_i/w_i+h_j/w_j)w_iw_j/(w_i+w_j)\},$$

$$z \ge \max_{1\le i\le j\le n}\{|(a_i+c_i-a_j-c_j|+|b_i-d_i|+|b_j-d_j|+ \\ + h_i/w_i+h_j/w_j)w_iw_j/(w_i+w_j)\},$$

$$z \ge \max_{1\le i\le j\le n}\{|b_i+d_i-b_j-d_j|+|a_i-c_i|+|a_j-c_j| + \\ + h_i/w_i+h_j/w_j)w_iw_j/(w_i+w_j)\}$$

$$z \ge \max_{i,j,k,i\le j}\{(|2(a_k+c_k)-a_i-c_i-a_j-c_j|+|b_i+d_i-b_j-d_j|+2|b_k-d_k|+ \\ + h_i/w_i+h_j/w_j+2h_k/w_k)w_iw_jw_k/(2w_iw_j+w_iw_k+w_jw_k)\},$$

$$z \ge \max_{i,j,k,i\le j}\{(|2(b_k+d_k)-b_i-d_i-b_j-d_j|+|a_i+c_i-a_j-c_j|+2|a_k-c_k|+ \\ + h_i/w_i+h_j/w_j+2h_k/w_k)w_iw_jw_k/2w_iw_j+w_iw_k+w_jw_k)\},$$

$$z \ge \max_{i,j,k}\{(|a_i+c_i-a_k-c_k|+|b_j+d_j-b_k-d_k|+|a_j-c_j|+|b_i-d_i|+ \\ + h_i/w_i+h_j/w_j+h_k/w_k)w_iw_jw_k/(w_iw_j+w_iw_k+w_jw_k)\}.$$

The optimum value is equal to the maximum of the righthand sides of these seven inequalities.

Our third example is the p-*center problem with mutual communication* in the rectilinear plane. This problem can be formulated as

$$\min_{x_1,\ldots,x_p,y_1,\ldots,y_p}\{\max\{\max_{1\le i\le n, 1\le j\le p}\{\alpha_{ij}(|a_i-x_j|+|b_i-y_j|)\},$$

$$\max_{1\le j<k\le p}\{\beta_{jk}(|x_j-x_k|+|y_j-y_k|)\}\}\},$$

where (a_i,b_i) are points in the plane, $i = 1,2,\ldots,n$, α_{ij},β_{jk} are positive integers, $i = 1,\ldots,n,j$, $k = 1,\ldots,p$. Using the same transformation as in the case of the 1-center problem the problem can be separated into two independent problems on a line, of the type

$$\text{minimize} \quad z$$

$$\text{subject to } |a_i-x_j| \le z/\alpha_{ij}, \quad i = 1,2,\ldots,n,\ j = 1,2,\ldots,p,$$

$$|x_j-x_k| \le z/\beta_{jk}, \quad j,k = 1,2,\ldots,p.$$

In Chapter 4 we derived necessary and sufficient conditions for the more general case of a tree. Our objective is to show how these conditions can be derived from Farkas' Lemma in this special case of the problem on a line.

We consider the set of inequalities given by

$$|a_i-x_j| \le c_{ij},\ i = 1,2,\ldots,n,\ j = 1,2,\ldots,p,$$

$$|x_j-x_k| \le b_{jk},\ j,k = 1,\ldots,p,$$

where c_{ij} and b_{jk} are nonnegative numbers, $i=1,2,\ldots,n$, $j,k = 1,\ldots,p$.

This set of inequalities is equivalent to

$$\left.\begin{aligned} x_j &\le a_i+c_{ij} \\ -x_j &\le c_{ij}-a_i \end{aligned}\right\}\ i = 1,2,\ldots,n,\ j = 1,\ldots,p,$$

$$\left.\begin{aligned} x_j-x_k &\le b_{jk} \\ -x_j+x_k &\le b_{jk} \end{aligned}\right\}\ j,k = 1,\ldots,p.$$

Let j_k denote a column vector of length k consisting of all ones, and let I_k denote the $k \times k$ unity matrix. Let A be the matrix corresponding to the set of inequalities and let b be the corresponding righthand side, i.e.,

$$A = \begin{bmatrix} j_n & & & & \\ & j_n & & & \\ & & \ddots & & \\ & & & j_n & \\ -j_n & & & & \\ & -j_n & & & \\ & & \ddots & & \\ & & & -j_n & \\ j_{p-1} & & & -I_{p-1} & \\ -j_{p-1} & & & I_{p-1} & \\ & j_{p-2} & & -I_{p-2} & \\ & -j_{p-2} & & I_{p-2} & \\ & & \ddots & & \\ & & & 1 & -1 \\ & & & -1 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \\ e_1 \\ e_2 \\ \vdots \\ e_p \\ f_1 \\ f_1 \\ f_2 \\ f_2 \\ \vdots \\ f_{p-1} \\ f_{p-1} \end{bmatrix},$$

where $d_i = (a_1+c_{1i},\dots,a_n+c_{ni})^T$, $c_i = (-a_1+c_{1i},\dots,-a_n+c_{ni})^T$, $i = 1,2,\dots,p$, and $f_i = (b_{i,i+1},\dots,b_{ip})^T$, $i = 1,2,\dots,p-1$.

According to Farkas' Lemma, there exists a vector x such that $Ax \le b$ if and only if $yb \ge 0$ for all $y \ge 0$ such that $yA = 0$. By scaling we may assume that each component of y is less than or equal to 1. Therefore there exists a vector x such that $Ax \le b$ if and only if $\min\{yb \mid yA = 0, 0 \le y \le 1\} \ge 0$. Since the matrix A is totally unimodular the minimum is achieved by a (0,1)-vector y. The total unimodularity of the matrix A follows from the following result (PADBERG [1976]): a (0,1,-1) matrix is totally unimodular if each collection of columns can be split up into two parts so that the sum of the columns in one part minus the sum in the other part is a vector with entires only 0,1 and -1.

Let y be the vector given by $y = (t_1,\dots,t_p,u_1,\dots,u_p,v_1,w_1,\dots,v_{p-1}, w_{p-1})$, where $t_i = (t_{1i},\dots,t_{ni})$, $u_i = (u_{1i},\dots,u_{ni})$, $i = 1,2,\dots,p$, $v_i = (v_{i+1,i},\dots,v_{pi})$, $w_i = (w_{i+1,i},\dots,w_{pi})$, $i = 1,2,\dots,p-1$. The constraints $yA = 0$ imply

$$(6.1.3) \qquad \sum_{k=1}^{n} (t_{ki}-u_{ki}) - \sum_{j=1}^{i-1} (v_{ij}-w_{ij}) + \sum_{j=i+1}^{p} (v_{ji}-w_{ji}) = 0, \; i = 1,2,\dots,p.$$

Summing these equalities over all i and noting that

$$\sum_{i=1}^{p}\sum_{j=1}^{i-1}(v_{ij}-w_{ij}) = \sum_{i=1}^{p}\sum_{j=i+1}^{p}(v_{ji}-w_{ji})$$

we find that

$$(6.1.4) \qquad \sum_{i=1}^{p}\sum_{k=1}^{n} t_{ki} = \sum_{i=1}^{p}\sum_{k=1}^{n} u_{ki}.$$

Consider the directed graph G with vertices E_i, $i = 1,2,\dots,n$, and N_j, $j = 1,\dots,p$, and arcs (E_i,N_j) of length $a_i + c_{ij}$, (N_j,E_i) of length $-a_i+c_{ij}$, $i = 1,\dots,n$, $j = 1,\dots,p$, arcs (N_j,N_k), (N_k,N_j) of length b_{jk}. Let us associate the following interpretation to the parameters in the vector y:

$t_{ki} = 1$ if and only if vertex N_i is entered through arc (E_k,N_i),
$u_{ki} = 1$ if and only if vertex E_k is entered through arc (N_i,E_k),
$w_{ij} = 1$ if and only if vertex N_i is entered through arc (N_j,N_i) $(i > j)$,
$v_{ij} = 1$ if and only if vertex N_j is entered through arc (N_i,N_j) $(i > j)$.

Constraints (6.1.3) say that the number of times we enter a vertex N_i is equal to the number of times we leave the same vertex, $i = 1,2,\dots,p$. Constraint (6.1.4) says that the number of times we enter the set $\{E_i | i = 1,2,\dots,n\}$ is equal to the number of times we leave the same set. We conclude that we can interpret y as the sum of a number of directed paths starting from and ending in $\{E_i | i = 1,2,\dots,n\}$. The constraint $yb \geq 0$ says

$$(6.1.5) \qquad \sum_{i=1}^{p}\sum_{k=1}^{n}[t_{ki}(a_k+c_{ki})+u_{ki}(-a_k+c_{ki})]+\sum_{i=1}^{p}\sum_{j=i+1}^{p}(v_{ji}+w_{ji})b_{ij} \geq 0.$$

Constraint (6.1.5) says that the sum of the lengths of the paths corresponding to y is nonnegative. If we let $L(E_i,E_j)$ denote the length of the shortest path in G from E_i to E_j, then the necessary and sufficient conditions for the existence of x such that $Ax \leq b$ are given by

$$L(E_i,E_j) \geq 0, \qquad i,j = 1,\dots,n.$$

These are exactly the separation conditions derived in Chapter 4.

6.2. Efficient points

Using Theorem 6.3 we will derive a necessary and sufficient condition for a point to be efficient. This will then be used to develop a polynomial-time algorithm to find the set of all efficient points.

Let $P_i = (a_i,b_i)$, $i = 1,2,\ldots,n$ be given points in the plane. A point (x_1,y_1) is dominated in position k if and only if there is a point (x,y) such that

$$\begin{aligned} |a_i-x|+|b_i-y| &\le q_i, \quad i = 1,2,\ldots,n,\ i \ne k, \\ |a_k-x|+|b_k-y| &< q_k, \end{aligned}$$

where $q_i = |a_i-x_1|+|b_i-y_1|$, $i = 1,2,\ldots,n$. An equivalent formulation of the constraint set is given by

$$\begin{aligned} x+y &\le \min_{i\ne k}\{a_i+b_i+q_i\}, \\ x-y &\le \min_{i\ne k}\{a_i-b_i+q_i\}, \\ -x+y &\le \min_{i\ne k}\{-a_i+b_i+q_i\}, \\ -x-y &\le \min_{i\ne k}\{-a_i-b_i+q_i\}, \\ x+y &< a_k+b_k+q_k, \\ x-y &< a_k-b_k+q_k, \\ -x+y &< -a_k+b_k+q_k, \\ -x-y &< -a_k-b_k+q_k. \end{aligned}$$

Define

$$A_1=A_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}$$

and nonnegative vectors $s = (s_1,s_2,s_3,s_4)$ and $t = (t_1,t_2,t_3,t_4)$. Consider all vectors s and t such that $sA_1 + tA_2 = 0$. We have to prove that s and t satisfy the condition of Theorem 6.3. We have

$$s_1A_1+tA_2 = 0 \iff \begin{bmatrix} s_1+s_2-s_3-s_4+t_1+t_2-t_3-t_4 = 0 \\ s_1-s_2+s_3-s_4+t_1-t_2+t_3-t_4 = 0 \end{bmatrix}$$

$$(6.2.1) \qquad \iff \begin{bmatrix} s_1-s_4+t_1-t_4 = 0 \\ s_2-s_3+t_2-t_3 = 0 \end{bmatrix}.$$

The matrix corresponding to equalities (6.2.1) is totally unimodular. Therefore we can restrict ourselves to (0,1)-vectors $(s_1,s_2,s_3,s_4,t_1,t_2,t_3,t_4)$. There are eight vectors satisfying (6.2.1) which cannot be written as the sum of other vectors satisfying (6.2.1). The eight vectors and the corresponding conditions are:

(1,0,0,1,0,0,0,0): $(a_k+b_k+q_k)+(-a_k-b_k+q_k) > 0,$ (6.2.2)
(1,0,0,0,0,0,0,1): $(a_k+b_k+q_k)+\min_{i\neq k}\{-a_i-b_i+q_i\} > 0,$ (6.2.3)
(0,1,1,0,0,0,0,0): $(a_k-b_k+q_k)+(-a_k+b_k+q_k) > 0,$ (6.2.4)
(0,1,0,0,0,0,1,0): $(+a_k-b_k+q_k)+\min_{i\neq k}\{-a_i+b_i+q_i\} > 0,$ (6.2.5)
(0,0,1,0,0,1,0,0): $(-a_k+b_k+q_k)+\min_{i\neq k}\{a_i-b_i+q_i\} > 0,$ (6.2.6)
(0,0,0,1,1,0,0,0): $(-a_k-b_k+q_k)+\min_{i\neq k}\{a_i+b_i+q_i\} > 0,$ (6.2.7)
(0,0,0,0,1,0,0,1): $\min_{i\neq k}\{a_i+b_i+q_i\}+\min_{j\neq k}\{-a_j-b_j+q_j\} \geq 0,$ (6.2.8)
(0,0,0,0,0,1,1,0): $\min_{i\neq k}\{a_i-b_i+q_i\}+\min_{j\neq k}\{-a_j+b_j+q_j\} \geq 0.$ (6.2.9)

Conditions (6.2.2) and (6.2.4) imply $q_k > 0$, i.e., $(x_1,y_1) \neq (a_k,b_k)$. Conditions (6.2.3), (6.2.5), (6.2.6) and (6.2.7) imply $q_i+q_k > |a_i-a_k| + |b_i-b_k|$, $\forall i \neq k$. Conditions (6.2.8) and (6.2.9) imply $q_i+q_j \geq |a_i-a_j|+|b_i-b_j|$, $\forall_{i\neq k}$ which is always satisfied because of the triangular inequality.

We conclude that a point X is dominated in position k if and only if
(1) $X \neq P_k$,
(2) $d(P_i,P_k) < d(P_i,X) + d(P_k,X)$ $\forall_{i\neq k}$,
where d(A,B) is the rectilinear distance between points A and B in the plane. This result leads to the following characterization of an efficient point:

(6.2.10) A point X is efficient if and only if for every point P_i there exists a point P_j such that $d(P_i,X) + d(X,P_j) = d(P_i,P_j)$

We shall use this characterization of an efficient point to develop an O(n log n) algorithm to find the set of all efficient points.

Through each point P_i construct a horizontal line and a vertical line. The horizontal (vertical) line should extend at least as far right and as far left (as far up and as far down) as every P_i. Subsequently whenever we refer to a *line* we mean one of these constructed lines. Figure 6.2.11 illustrates a number of definitions to follow. For any vertical line we define the set of points *east* (*west*) of the line to be the union of the line with the set of points to the right (left) of the line. Similarly we define the set of points *north*, and the set of points *south* of each horizontal line. (For convenience we use the abbreviations N,E,W, and S for North, East, West, and South respectively). Given any two distinct adjacent horizontal lines H and H' with H N of H', and any two distinct adjacent vertical V and V' with V E of V', we call the set of points lying W of V, E of V', S of H and N of H', a *box*, and denote the box by B. We call the collection of all boxes between any two adjacent vertical (horizontal) lines a *column* (*row*). Each of the four intersections of the box with a line we call an *edge* of B. We say two boxes are *adjacent* if their intersection is an edge of each box. The collection of all points lying S of H' and E of V we call the SE *direction* of B (abbreviated SE(B)); similarly we define SW, NW and NE directions of B, and use the abbreviations SW(B), NW(B) and NE(B) respectively. We say that a direction of B is empty if there is no point P_i in the direction, $i = 1,2,\ldots,n$.

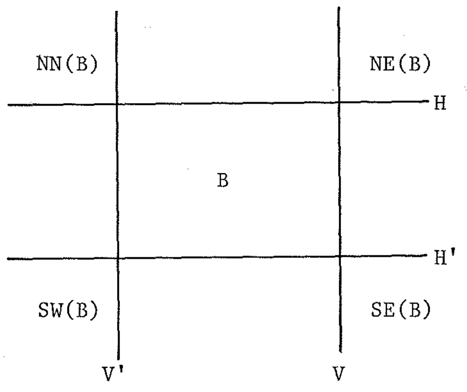

Figure 6.2.11. Directions of B.

We assume that not all the P_i lie on a single vertical, or on a single horizontal line, as in this case the set of efficient points is precisely the line segment joining the two P_i which are farthest apart.

We call a box a *0-box*, *1-box* or *2-box* if 0,1, or 2 directions respectively are empty; it is possible to have only these three types of boxes. For any 1-box B we call the two edges opposite the empty direction the *leading edges* of B. It follows directly from the result (6.2.10) that each point in a 0-box or in a 2-box is an efficient point, and that only leading edges of a 1-box are possible candidates for containing efficient points.

Denote the horizontal lines by $H_1,H_2,\ldots,H_{p+1}$ from N to S and the rows by $R_1,\ldots,R_p$ from N to S. For R_i, $1 \le i \le p$, define $NW_i(SW_i)$ to be the x-coordinate of the westmost point P_j which is N(S) of R_i. Define $NE_i(SE_i)$ to be the x-coordinate of the eastmost point P_j which is N(S) of R_i. For each R_i we define $W_i^* = \max\{NW_i, SW_i\}$ and $E_i^* = \min\{NE_i, SE_i\}$. For $1 \le i \le p+1$ denote by $W_i(E_i)$ the x-coordinate of the westmost (eastmost) point P_j on H_i. Let $V' < B < V$ mean that box B is E of V' and W of V. The following two lemmas are trivial.

LEMMA 6.1.12. *A box* B *in* R_i *is a* 0*-box if and only if* $W_i^* < E_i^*$ *and* $W_i^* < B < E_i^*$. □

LEMMA 6.2.13. *A box* B *in* R_i *is a* 2*-box if and only if* $W_i^* > E_i^*$ *and* $E_i^* < B < W_i^*$. □

The next two lemmas characterize horizontal and vertical leading edges belonging to the set of efficient points.

LEMMA 6.2.14. *A vertical leading edge* e *of a* 1*-box* B *in* R_i*, contained in the vertical line* V *and not an edge of a* 0*-box or* 2*-box, belongs to the set of efficient points if and only if* $W_i^* = V = E_i^*$.

PROOF. If $e \in V_1$ (the westmost vertical line), then we have two possibilities:

1. SE(B) is the only empty direction, i.e., $SW_i = SE_i = NW_i = V_1$ and $NE_i > SE_i$
2. NE(B) is the only empty direction, i.e., $SW_i = NW_i = NE_i = V_1$ and $SE_i > NE_i$.

If $e \in V_{q+1}$ (the eastmost vertical line), then we have two possibilities:

1. NW(B) is the only empty direction, i.e., $NW_i = NE_i = SE_i$ and $SW_i < SE_i$;
2. SW(B) is the only empty direction, i.e., $SW_i = SE_i = NE_i$ and $NW_i < NE_i$.

If $e \in V_j$ $(1<j<q+1)$, then e is not in a 0-box or 2-box if and only if e is a leading edge of a box B' adjacent to B (see Figure 6.2.15)

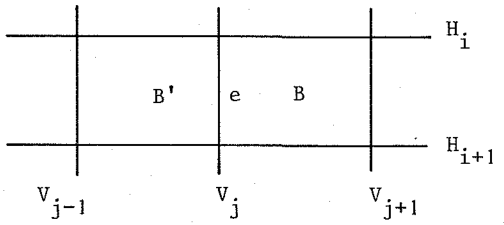

Figure 6.2.15. Leading edge e of B and B'.

There are two possibilities:

1. SW(B') and SE(B) are the only empty directions, i.e., $SW_i = SE_i = V_j$, $NW_i < SW_i$ and $NE_i > SE_i$;
2. NW(B') and NE(B) are the only empty directions, i.e., $NW_i = NE_i = V_j$, $SW_i < NW_i$ and $SE_i > NE_i$.

Combining all these results prove the lemma. □

LEMMA 6.2.16. *A horizontal edge* e *on* H_i *between* V_{j-1} *and* V_j, *which is not contained in a* 2*-box, belongs to the set of efficient solutions if and only if*

$$\max\{NW_i, SW_{i-1}\} \le V_{j-1} \textit{ and } V_j \le \min\{NE_i, SE_{i-1}\}.$$

PROOF. If e is on H_1, then e belongs to the efficient set if and only if there exist a point on H_1 that lies W of V_{j-1} and there exist a point that lies E of V_j. If we define $SW_0 = -\infty$ and $SE_0 = \infty$, then this case can be characterized as indicated by the lemma. If e is on H_{p+1}, then when we define $NW_{p+1} = -\infty$ and $NE_{p+1} = \infty$ this can also be characterized as indicated by the lemma. If e is on H_i, $1<i<p+1$, then let the box B and the box B' be the boxes in R_i, R_{i-1} respectively which have e in common with each other (see Figure 6.2.17).

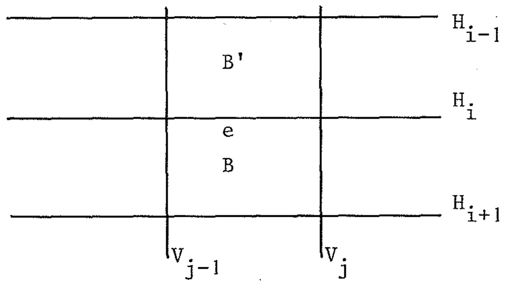

Figure 6.2.17. Leading edge e on H_i.

Box B' and B are not a 2-box, i.e. if SW(B') = ∅, then NE(B') ≠ ∅ and e cannot contain efficient points according to result (6.2.10). The same holds when SE(B') = ∅, NE(B) = ∅, or NW(B) = ∅. If NW(B) ≠ ∅, NE(B) ≠ ∅, SW(B') ≠ ∅ and SE(B') ≠ ∅, then e belongs to the set of efficient points. We conclude that e belongs to the set of efficient points if and only if NW(B) ≠ ∅, NE(B) ≠ ∅, SW(B') ≠ ∅ and SE(B') ≠ ∅, i.e.,

$V_{j-1} \geq \max\{NW_i, SW_{i-1}\}$ and $V_j \leq \min\{NE_i, SE_{i-1}\}$. □

The following algorithm constructs the set of all efficient points. The correctness of the algorithm follows from the previous lemmas.

ALGORITHM. Rank the existing points by their y-coordinates to determine the lines $H_1, \ldots, H_{p+1}$. Compute W_i and E_i, $1 \leq i \leq p+1$, NW_i, NE_i, SW_i, SE_i, $1 \leq i \leq p$ and W_i^* and E_i^*, $1 \leq i \leq p$. If $W_i^* < E_i^*$, then all boxes B for which $W_i^* < B < E_i^*$ belong to the set of efficient points; if $W_i^* = E_i^*$, then the vertical edge at the line V with $W_i^* = V = E_i^*$ belongs to the set of efficient points; if $W_i^* > E_i^*$, then all boxes B for which $W_i^* > B > E_i^*$ belong to the set of efficient points. For the horizontal line H_i compute $u_i = \max\{NW_i, SW_{i-1}\}$ and $t_i = \min\{NE_i, SE_{i-1}\}$, $1 \leq i \leq p+1$. All horizontal edges lying between the vertical line $x = u_i$ and $x = t_i$ belong to the set of efficient points. □

The following recursive relations can be used to compute NW_i, NE_i, SW_i and SE_i.

$$\begin{aligned}
NW_1 &= W_1 , & NE_1 &= E_1 ,\\
NW_i &= \min\{W_i, NW_{i-1}\} , & NE_i &= \max\{E_i, NE_{i-1}\} ,\\
SW_i &= \min\{W_{i+1}, SW_{i+1}\} , & SE_i &= \max\{E_{i+1}, SE_{i+1}\},\\
SW_p &= W_{p+1} , & SE_p &= E_{p+1} ,
\end{aligned}$$

The complexity of the algorithm is $O(n \log n)$.

CHAPTER 7

CONCLUDING REMARKS

In this final chapter we intend to indicate some interesting questions which are still open as well as to express some thoughts about current research in location theory.

The study of tree location problems is related to the theory of totally-balanced matrices. Polynomial algorithms have been given in Chapter 2 to solve a class of integer programming problems defined on this class of matrices. An interesting question for future research is to develop similarly efficient algorithms for the same class of integer programming problems defined on balanced matrices. It is known that these problems are in principle solvable in polynomial time using the ellipsoid-method of KHACHIAN [1979].

In Theorem 1.3.6 we mentioned the result that the incidence matrix of vertices and neighbourhood subtrees of a given tree is a totally-balanced matrix. We conjecture that the reverse is true as well: Given any $n \times m$ totally-balanced matrix A, one can construct a tree on n vertices and m neighbourhood subtrees such that the corresponding incidence matrix is given by A.

The use of data structures in solving problems efficiently is nicely demonstrated by PQ-trees invented by LUEKER [1975]. PQ-trees are used to solve the *consecutive arrangement problem*: Given an $n \times m$ (0,1)-matrix, does there exist a permutation of the rows such that the ones in each column occur consecutively? This problem was solved in $O(m+n+f)$ time, where f is the number of nonzero entries. We conjecture that an $n \times m$ totally-balanced matrix can be put into standard form in $O(m+n+f)$ time, where f is the number of nonzero elements, using a suitable data structure. The algorithm given in Section 2.2 requires $O(nm^2)$ time.

In their recent survey of location on networks, TANSEL, FRANCIS & LOWE [1981] give 118 references, at least 40 of which are devoted to tree location problems. It is our belief that tree location problems have attracted so many researchers mainly because, due to their nice mathematical structure, polynomial algorithms for their solution are likely to exist. The work reported in this monograph is no exception to this rule. By now we regard the class of location problems on trees as well solved. The time has come to move forward to more complex real-world problems and to use the knowledge obtained by studying tree location problems in solving these complex problems. In the future researchers should concentrate on the design and analysis of approximation algorithms which are of polynomial complexity and exhibit a favorable (worst-case or probabilistic) behaviour.

REFERENCES

AHO, A.V., J.E. HOPCROFT & J.D. ULLMAN (1974), *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA).

BERGE, C. (1976), *Graphs and Hypergraphs* (North-Holland, Amsterdam).

BERGE, C. (1972), Balanced Matrices *Math. Programming 2*, 19-31.

BROUWER, A.E. & A. KOLEN (1980), A super-balanced hypergraph has a nest-point. Report ZW 146, Mathematisch Centrum, Amsterdam.

BUNEMAN, P. (1974), A characterization of rigid circuit graphs. *Discrete Math. 9*, 205-212.

CHALMET, L., R.L. FRANCIS & A. KOLEN (1981), Finding efficient solutions for rectilinear distance location problems efficiently *European J. Oper. Res. 6*, 117-124.

CHAN, A. & R.L. FRANCIS (1976), A round-trip location problem on a tree graph. *Transportation Sci. 10*, 35-51.

CHAN, A. & D.W. HEARN, (1977), A rectilinear distance minimax round-trip location problem. *Transportation Sci. 11*, 107-123.

CHANDRASEKARAN, R. & A. TAMIR (1981), An $O((n\log p)^2)$ algorithm for the continuous p-center problem on a tree. *SIAM J. Algebra Discrete Math. 1*, 370-375.

DEARING, P.M., R.L. FRANCIS & T.J. LOWE (1976), Convex location problems on tree networks. *Oper. Res. 24*, 628-642.

DIRAC, G.A. (1961), On rigid circuit graphs. *Abh. Math. Sem. Univ. Hamburg 25*, 71-76.

ERLENKOTTER, D. (1978), A dual-based procedure for uncapacitated facility location. *Oper. Res. 26*, 992-1009.

FARKAS, J. (1902), Ueber die Theorie der einfachen Ungleichungen. *J. rein. angew. Math. 124,* 1-27.

FOURIER, J.B.J. (1826), Solution d'une question particulière du calcul des inéqualités. *Oeuvres II,* 317-328.

FRANCIS, R.L., T.J. LOWE, & H.D. RATLIFF (1978), Distance constraints for tree network multifacility location problems *Oper. Res. 26,* 570-596.

FULKERSON, D.R., A.J. HOFFMAN & R. OPPENHEIM (1974), On balanced matrices. *Math. Programming Stud. 1,* 120-132.

GAVRIL, F. (1972), Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput. 1,* 180-187.

GAVRIL, F. (1974), The intersection graphs of subtrees are exactly the chordal graphs. *J. Combin. Theory Ser. B 16,* 47-56.

GILES, R. (1978), A balanced hypergraph defined by subtrees of a tree. *Ars Combin. 6,* 179-183.

GOLDMAN, A.J. (1971), Optimal center location in simple networks. *Transportation Sci. 5,* 212-221.

GOLUMBIC, M.C. (1980), *Algorithmic Graph Theory and Perfect Graphs* (Academic Press, New York).

GRÖTSCHEL, M., L. LOVÁSZ & A. SCHRIJVER (1981), Polynomial algorithms for perfect graphs. Report No 81176, Institut für Operations Research, Universität Bonn.

HAKIMI, S.L. & O. KARIV (1979), An algorithmic approach to network location problems. Part I: The p-centers *SIAM J. Appl. Math. 37,* 539-560.

HOFFMAN, A.J. & M. SAKAROVITCH (1981), Personal communication.

JUEL, H. & R.F. LOVE (1976), An efficient computational procedure for solving the multifacility rectilinear facilities location problem. *Oper. Res. Quart. 27,* 697-703.

KHACHIAN, L.G. (1979), A polynomial algorithm in linear programming. *Soviet Math. Dokl 20,* 191-194.

KARZANOV, A.V. (1974), Determining the maximal flow in a network by the method of preflows. *Soviet Math. Dokl 15,* 434-437.

KOLEN, A. (1981), Equivalence between the direct search approach and the cut approach to the rectilinear distance location problem. *Oper. Res. 29*, 616-620.

KOLEN, A. (1981), Minimum cost tree location problems (presented at ISOLDE II, Skodsborg, Denmark).

KRARUP, J. & P. PRUZAN (1977), Selected families of discrete location problems. Part III: The plant location family. Working Paper WP-12-77, University of Calgary.

KUHN, H.W. (1956), Solvability and consistency for linear equations and inequalities. *Amer. Math. Monthly 63*, 217-232.

LAWLER, E.L. & J.K. LENSTRA (1981), Machine scheduling with precedence constraints. Report BW 148/81, Mathematisch Centrum, Amsterdam.

LUEKER, G.S. (1975), Interval graph algorithms, Ph. D. Thesis, Princeton University.

LOVÁSZ, L. (1979), *Combinatorial Problems and Exercises*, 528 (Akadémia Kiadó, Budapest).

MEGIDDO, N. & A. TAMIR (1981), New results on the complexity of p-center problems. Report Department of Statistics. Tel-Aviv University.

MOTZKIN, T.S. (1936), Beiträge zur Theorie der linearen Ungleichungen. Inaugural Dissertation Basel, Azriel, Jerusalem.

PADBERG, M.W. (1976), A note on the total unimodularity of matrices. *Discrete Math. 14*, 273-278.

PICARD, J.C. & H.D. RATLIFF (1978), A cut approach to the rectilinear distance location problem. *Oper. Res. 26*, 422-434.

PRITSKER, A.A.B. & P.M. GHARE (1970), Locating new facilities with respect to existing facilities. *AIIE Trans. 2*, 290-297.

RAO, M.R. (1973), On the direct search approach to the rectilinear facilities location problem. *AIIE Trans. 5*, 256-264.

ROSE, D.J. (1970), Triangulated graphs and the elimination process, *J. Math. Anal. Appl. 32*, 597-609.

SHERALI, H.D. & C.M. SHETTY (1978), A primal simplex based solution procedure for the rectilinear distance multifacility location problem. *J. Oper. Res. Soc. 29*, 373-381.

SHIER, D.R. (1977), A min-max theorem for p-center problems on a tree. *Transportation Sci. 11*, 243-252.

STOER, J. & C. WITZGALL (1970), *Convexity and Optimization in Finite Dimensions I* (Springer, Berlin).

TAMIR, A. (1980), On the core of cost allocation games defined on location problems. Report Department of Statistics, Tel-Aviv University.

TAMIR, A. (1980), A class of balanced matrices arising from location problems. Report Department of Statistics, Tel-Aviv University.

TANCEL, B.C., R.L. FRANCIS & T.J. LOWE (1981), Location on networks: a survey. Research Report No. 81-12, Industrial and Systems Engineering Department, University of Florida, Gainesville, FL 32611.

WENDEL, R.E., A.P. HURTER & T.J. LOWE (1977), Efficient points in location problems. *AIIE Trans. 9*, 238-246.

SAMENVATTING

Locatietheorie dateert uit het begin van de zeventiende eeuw toen Fermat zijn beroemd geworden probleem formuleerde: gegeven drie punten in het platte vlak, vind een vierde punt zodanig dat de som van de afstanden tot de drie gegeven punten minimaal is. In zijn "Doctrine and Application of Fluxious" (Londen 1750) generaliseerde Simpson dit probleem tot het vinden van een punt met minimale som van de gewogen afstanden tot de drie gegeven punten. Het meeste werk aan de locatietheorie heeft echter plaatsgevonden sinds 1957. Er is een opmerkelijke verscheidenheid aan tijdschriften waarin artikelen over het onderwerp verschijnen. Genoemd kunnen worden tijdschriften op het gebied van de mathematische besliskunde, bedrijfskunde, civiele techniek, geografie, economie en planning. De communicatie tussen deze gebieden laat veel te wensen over met als gevolg dat vele resultaten herontdekt zijn in de loop der jaren. Resultaten op een bepaald gebied waren niet bekend bij onderzoekers op een ander gebied. Dit geldt vooral voor resultaten uit de grafentheorie en uit de combinatorische optimalisering. In dit proefschrift laten we zien dat resultaten betreffende perfecte grafen en totaal-gebalanceerde matrices slechts onlangs hun weg hebben gevonden in de locatietheorie. Vele oude en nieuwe resultaten kunnen uit deze algemene theorie worden afgeleid. De communicatie is verbeterd sedert ISOLDE (International Symposium on Locational Decisions) I en II, gehouden in respectievelijk Banff, Canada in april 1978 en Skodsborg, Denemarken in juni 1981.

De klasse van locatieproblemen wordt als volgt gekarakteriseerd: plaats een aantal faciliteiten in een gegeven ruimte om bepaalde goederen of diensten te leveren aan een gespecificeerde verzameling van bestaande faciliteiten (klanten) zodanig dat een of meer criteria worden geoptimaliseerd onder een aantal voorwaarden. De kwaliteit van de dienstverlening wordt gewoonlijk gemeten in de afstand tussen de faciliteiten. We komen verschillende soorten locatieproblemen tegen afhankelijk van de gegeven

ruimte (een netwerk of het platte vlak), de afstandsfunctie (kortste-pad-lengte of rectilineaire afstand), het aantal faciliteiten (constant of variabel), de interactie tussen faciliteiten (tussen een nieuwe faciliteit en alle bestaande faciliteiten of tussen alle paren faciliteiten), en het optimaliteitscriterium (minimaliseer de gewogen som van de afstanden, minimaliseer de maximale afstand, minimaliseer de totale kosten verbonden aan het plaatsen van nieuwe faciliteiten).

De complexiteitstheorie maakt het mogelijk onderscheid te maken tussen *goed oplosbare* problemen, die oplosbaar zijn door een algoritme met looptijd begrensd door een polynoom in de probleemgrootte, en NP-*moeilijke* problemen, waarvoor het bestaan van dergelijke algoritmen erg onwaarschijnlijk is.

Van vele locatieproblemen op grafen waarbij de anstand tussen punten gedefinieerd wordt als de lengte van een kortste pad tussen deze twee punten, heeft men bewezen dat het NP-moeilijke problemen zijn. Dit rechtvaardigt het onderzoek naar beperktere versies van die problemen. We zullen aannemen dat de graaf geen cykels bevat, d.w.z., een boom is. Het blijkt dat de meeste locatieproblemen op bomen opgelost kunnen worden in polynomiale tijd. Men begrijpt nog steeds niet helemaal waarom dit zo is. Gedeeltelijk kunnen we dit raadsel oplossen door locatieproblemen op bomen in verband te brengen met convexiteit, perfecte grafen en totaal-gebalanceerde matrices. Alhoewel locatieproblemen op bomen een mooie wiskundige structuur hebben moet nog getoetst worden hoe bruikbaar de ontwikkelde algoritmen zijn in alledaagse complexe problemen. Ze kunnen bijvoorbeeld nuttig zijn in aftellingsmethoden en benaderingsalgoritmen.

In hoofstuk 1 geven we eigenschappen van bomen en we introduceren zowel de koordengraaf als de totaal-gebalanceerde matrix. Aan de hand van een voorbeeld laten we het verband zien tussen min-max locatieproblemen op bomen en het vinden van een minimale overdekking van een koordengraaf met volledige deelgrafen. Tevens laten we zien hoe een locatieprobleem op een boom waarbij het er om gaat een configuratie van faciliteiten te vinden die tegen minimale vestigingskosten alle klanten bedient, geformuleerd kan worden als een verzamelingsoverdekkingsprobleem op een totaal-gebalanceerde matrix.

In hoofdstuk 2 geven we een polynomiale algoritme om het verzamelingsoverdekkingsprobleem op een totaal-gebalanceerde matrix op te lossen. We laten zien dat een totaal-gebalanceerde matrix als zodanig herkend kan worden in polynomiale tijd. Tenslotte geven we een polynomiale algoritme om het locatieprobleem op een boom op te lossen waarbij het er om gaat een configuratie van faciliteiten te vinden die tegen minimale vestigings- en transportkosten alle klanten bedient, waarbij de transportkosten lineair toenemen met de afstand tussen de klant en de faciliteit die hem de goederen levert.

In hoofdstukken 3 en 4 beschrijven we locatieproblemen met wederzijdse communicatie, d.w.z., er is communicatie tussen alle paren faciliteiten. In hoofdstuk 3 behandelen we het probleem waarin we een gegeven aantal nieuwe faciliteiten willen plaatsen zodanig dat de som van de gewogen afstanden tussen alle paren faciliteiten geminimaliseerd wordt. In hoofstuk 4 willen we, gegeven een aantal nieuwe faciliteiten, het maximum van de gewogen afstanden tussen paren faciliteiten minimaliseren.

In hoofdstuk 5 bestuderen we het probleem van een transportfirma die een gegeven aantal vrachtwagendepots wil bouwen op een boomnetwerk, waarbij de vrachtwagens bepaalde taken moeten uitvoeren die er uit bestaan goederen op te halen van een punt en deze te brengen naar een ander punt. De kosten verbonden met het uitvoeren van een taak worden gegeven door een continue stijgende functie van de afgelegde afstand. Het probleem dat opgelost wordt is het vinden van de locaties voor de depots zodanig dat de maximale kosten om een bepaalde taak uit te voeren minimaal zijn.

In hoofdstuk 6 laten we zien hoe het Lemma van Farkas gebruikt kan worden om locatieproblemen in het platte vlak op te lossen waarbij de gebruikte metriek de rectilineaire afstand is.

LOCATION PROBLEMS ON TREES AND IN THE RECTILINEAR PLANE

Antoon Kolen

STELLINGEN

1

Iedere binaire code met de eigenschap dat ieder tweetal codewoorden afstand precies 12 heeft bestaat uit ten hoogste 32 woorden. Er bestaat een dergelijke code met 32 codewoorden en woordlengte 67.

J.I. HALL, A.J.E.M. JANSEN, A.W.J. KOLEN en J.H. VAN LINT (1977) Equidistant codes with distance 12, *Discrete Math.* 17,71-83.

2

Gegeven zijn een groep van personen en een verzameling van eigenschappen; van ieder persoon is bekend welke van de eigenschappen hij bezit. Er wordt gezocht naar een deelverzameling van eigenschappen van minimale cardinaliteit zodanig dat ieder tweetal personen in tenminste één ervan verschilt. Dit probleem is NP-moeilijk, zelfs in het geval dat iedere eigenschap bij niet meer dan twee personen voorkomt. Deze uitspraak is strijdig met een bewering van M.R. Garey en D.S. Johnson.

M.R. GAREY en D.S. JOHNSON (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, p.222.

3

Beschouw het in de voorgaande stelling geformuleerde probleem. Indien iedere eigenschap bij niet meer dan twee personen voorkomt kan in lineaire tijd een benaderingsoplossing worden verkregen die in cardinaliteit ten hoogste 50% van een optimale oplossing afwijkt. De benaderingsoplossing wordt gegeven door een opspannend bos in een met de probleeminstantie corresponderende graaf.

4

Gegeven zijn een boom $T = (\{v_1,v_2,...,v_n\},E)$ met de op blz. 8 gedefinieerde metriek en een stelsel consistente afstandsbeperkingen

(DC) $d(v_i,x_j) \leq c_{ij}$, $\quad i = 1,...,n,\ j = 1,...,p,$

$\quad d(x_j,x_k) \leq b_{jk}$, $\quad j,k = 1,...,p.$

Noem een toegelaten oplossing $X = \{x_1,...,x_p\}$ voor (DC) *efficiënt* als er geen toegelaten oplossing $Y = \{y_1,...,y_p\}$ bestaat zodanig dat $d(v_i,y_j) \leq d(v_i,x_j)$ $\forall i,j$ en $d(y_j,y_k) \leq d(x_j,x_k)$ $\forall j,k$ en stricte ongelijkheid optreedt in tenminste één vergelijking. Laat $Z = \{z_1,...,z_p\}$ een toegelaten oplossing zijn. Construeer de graaf BC met punten $E_1,...,E_n,N_1,...,N_p$ en kanten $\{E_i,N_j\}$ ter lengte $d(v_i,z_j)$ $\forall i,j$ en $\{N_j,N_k\}$ ter lengte $d(z_j,z_k)$ $\forall j,k$. Z is een efficiënte oplossing dan en slechts dan als er voor iedere k, $1 \leq k \leq p$, een kortste pad $L(E_i,E_j)$ in BC bestaat dat N_k bevat en waarvoor $d(v_i,v_j) = L(E_i,E_j)$.

5

Het in (3.2.1) geschetste probleem kan geformuleerd worden als een kwadratisch programmeringsprobleem van de vorm

$$\min\{bx-x^TCx \mid x \text{ binair}\},$$

waarbij b een willekeurige vector is en C een niet-negatieve matrix. Vanuit deze observatie had Kolen [1979] moeten constateren dat ieder kwadratisch programmeringsprobleem van deze vorm te formuleren is als een minimaal snedeprobleem.

A. KOLEN (1979) On "A cut approach to the rectilinear distance facility location problem" by J.C. Picard and H.D. Ratliff, Rapport BW 103/79, Mathematisch Centrum, Amsterdam.

J.C. PICARD en M. QUEYRANNE (1979) Selected applications of maximum flows and minimum cuts in networks, Rapport Ecole Polytechnique de Montréal.

6

Uit het steeds weer zonder commentaar verwijzen naar de incorrecte resultaten van Pritsker en Ghare betreffende het p-mediaan probleem met wederzijdse communicatie, blijkt dat slechts weinigen hun artikel gelezen hebben.

A.A.B. PRITSKER en P.M. GHARE (1970) Locating new facilities with respect to existing facilities, *AIIE Trans.* 2,290-297.

7

Wanneer complexiteitsresultaten zijn gebaseerd op het gebruik van parallelle processoren, dient dit vooralsnog in de titel of samenvatting van het artikel tot uitdrukking te komen. Dit is niet altijd het geval.

N. MEGIDDO en A. TAMIR (1981) New results on the complexity of p-center problems, Rapport Department of Statistics, Tel-Aviv University.

8

In Zuid-Limburg zijn door de gemeentelijke herindeling de vroegere gemeenten Bingelrade, Jabeek, Merkelbeek en Schinveld samengevoegd tot de nieuwe gemeente Onderbanken. De naam Onderbanken is meer te danken aan aversie van de drie eerstgenoemde plaatsen tegen de naam Schinveld dan aan historische overwegingen.

9

Het opnemen van de naam van een variabele in de naam gegeven aan een probleemtype dient vermeden te worden en is alleen dan te rechtvaardigen wanneer het als een onuitroeibaar gebruik wordt beschouwd.

Dit proefschrift, hoofdstukken 3, 4 en 5.

10

Wanneer in een natuurlijk bridgesysteem een aansluiting in een kleur is vastgesteld verdient het aanbeveling het eerstvolgende bod als een vraagbod naar kracht en verdeling te beschouwen. Dit komt de kwaliteit van het systeem ten goede.

11

```
           S: 5
           H: 85
           R: AH7
           K: AH85432
S: H107   ┌─────N─────┐ S: 86432
H: 9      │W         O│ H: V762
R: V1083  │           │ R: B62
K: VB1097 └─────Z─────┘ K: 6
           S: AVB9
           H: AHB1043
           R: 954
           K: -
```

Op het gegeven spel kan zuid een contract van 7 harten winnen bij uitkomst van klaveren vrouw en bij optimaal tegenspel van oost en west.