SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK

TECHNICAL REPORT NO. 605

August 1983
(Revised September 1984)

The Uncapacitated Facility Location Problem[+,++]

by

Gerard Cornuejols[*]
George L. Nemhauser
Laurence A. Wolsey[**]

[+] This paper was prepared as Chapter 3 for the forthcoming book Discrete Location Theory, R.L. Francis and P. Mirchandini (eds.), Wiley-Interscience.

[++]This work was supported in part by National Science Foundation grants ECS-8005350, ECS-8307473 and ECS-8205425.

[*] Graduate School of Industrial Administration, Carnegie-Mellon University

[**]Center for Operations Research and Econometrics, Universite Catholique de Louvain

Typed by Anne T. Kline

Chapter 3

The Uncapacitated Facility Location Problem

Table of Contents

3.1. Formulation and Applications

An economic problem of great practical importance is to choose the location of facilities, such as industrial plants or warehouses, in order to minimize the cost (or maximize the profit) of satisfying the demand for some commodity. In general there are fixed costs for locating the facilities and transportation costs for distributing the commodities between the facilities and the clients. This problem has been extensively studied in the literature and is commonly referred to as the plant location problem, or facility location problem. When each potential facility has a capacity, which is the maximum demand that it can supply, the problem is called the capacitated facility location problem. When the capacity hypothesis is not needed, we have the simple or uncapacitated facility location problem, or, for short, the UFL Problem.

The mathematical formulation of these problems as integer programs has proven very fruitful in the derivation of solution methods. To formalize the UFL Problem, we consider a set of clients $I = \{1,\ldots,m\}$ with a given demand for a single commodity, and a set of sites $J = \{1,\ldots,n\}$ where facilities can be located. In the literature it has been traditional to use the phrase "facility j is open" to mean that a facility is actually established at site j. Let f_j be the given fixed cost of opening facility j and assume there is a known profit c_{ij} that is made by satisfying the demand of client i from facility j. Typically, c_{ij} is a function of the production costs at facility j, the transportation costs from facility j to client i, the demand of client i and the selling price to client i. For example, $c_{ij} = d_i(p_i - q_j - t_{ij})$ where d_i is the demand, p_i the price per unit, q_j the production cost per unit and t_{ij} the transportation cost per unit.

The UFL Problem is to open a subset of facilities in order to maximize total profit, given that all demand has to be satisfied. An instance of the problem is specified by integers m and n, an mxn matrix $C = \{c_{ij}\}$ and a $1xn$ matrix $f = \{f_j\}$. Note that there is no loss of generality in assuming $f_j \geq 0$ for all $j \in J$ since if $f_k < 0$ every optimal solution contains facility k.

For any given set S of open facilities, it is optimal to serve client i from a facility j for which c_{ij} is maximum over $j \in S$. So, given S, the profit is $z(S) = \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j$. The problem is to find a set S that yields the maximum profit Z, that is $Z = \max_{S \subseteq J} z(S)$. This can be viewed as a combinatorial formulation of the problem.

An integer linear programming formulation is obtained by introducing the following variables. Let $x_j = 1$ if facility j is open and $x_j = 0$ otherwise; $y_{ij} = 1$ if the demand of client i is satisfied from facility j and $y_{ij} = 0$ otherwise. The integer program is

$$Z = \max \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j \tag{1.1}$$

$$\sum_{j \in J} y_{ij} = 1 \qquad \text{all } i \in I \tag{1.2}$$

$$y_j \leq x_{ij} \qquad \text{all } i \in I,\ j \in J \tag{1.3}$$

$$x_j,\ y_{ij} \in \{0,1\} \quad \text{all } i \in I,\ j \in J. \tag{1.4}$$

The constraints (1.2) guarantee that the demand of every client is satisfied, whereas (1.3) guarantees that the clients are supplied only from open facilities.

Note that because of (1.2) if c_{ij} is replaced by $c'_{ij} = c_{ij} + c_i$ for all $j \in J$, then the profit of each feasible solution is changed by c_i. Hence we can add a constant to any row of matrix C and the set of optimal decisions remains the same. In particular, the selling price of the commodity to client i is a constant added to row i. Therefore, only the costs (production, transportation and fixed operating costs) are relevant for the decision. This is why in the literature the UFL Problem is often presented as

$$\min \sum_{i\in I} \sum_{j\in J} d_{ij}y_{ij} + \sum_{j\in J} f_j x_j \qquad (1.1')$$

subject to (1.2)-(1.4) where d_{ij} is the cost (production plus transportation) of serving client i from facility j. This formulation is mathematically equivalent to (1.1)-(1.4) since (1.1') becomes -(1.1) by setting $c_{ij} = -d_{ij}$. In other words, costs can simply be regarded as negative profits.

In the integer program (1.1)-(1.4), the x_j's are the strategic variables. Once a set of x_j's that satisfy (1.4) are specified, it is simple to determine a set of y_{ij}'s that solves the integer program for the fixed x_j's. Even if we drop the integrality requirement on the y_{ij}'s, an optimal set of y_{ij}'s is given by $y_{ij^*} = 1$ and $y_{ij} = 0$, $j \neq j^*$ where $c_{ij^*} = \max\{c_{ij}: x_j = 1, j \in J\}$. Thus (1.4) can be replaced by

$$x_j \in \{0,1\},\ y_{ij} \geq 0 \quad \text{all} \quad i \in I,\ j \in J. \qquad (1.4')$$

The formulation (1.1)-(1.3), (1.4') is a mixed integer linear program.

An integer linear program equivalent to (1.1)-(1.4) is obtained by replacing the constraints (1.3) by the more compact set of constraints

$$\sum_{i \in I} y_{ij} \leq m x_j \quad \text{all} \quad j \in J. \tag{1.3'}$$

To see that (1.3) can be replaced by (1.3'), note that when $x_j = 0$ both (1.3) and (1.3') imply $y_{ij} = 0$ for all $i \in I$, and when $x_j = 1$ both (1.3) and (1.3') are satisfied for all choices of y_{ij} that satisfy the constraints (1.2). However, the two formulations are equivalent only for 0-1 values of the variables x_j. For each j, (1.3') is obtained by summing (1.3) over all $i \in I$; hence any solution to (1.2), (1.3) is also a solution to (1.2) and (1.3'). But the converse is false when $0 < x_j < 1$; i.e. the feasible region defined by (1.2), (1.3') and

$$0 \leq x_j \leq 1, \; y_{ij} \geq 0 \quad \text{all} \quad i \in I, \; j \in J \tag{1.5}$$

strictly contains the region defined by (1.2), (1.3) and (1.5).

In the UFL Problem, the number of facilities that are open in an optimal solution is not specified; it is determined by a solution of (1.1)-(1.4). From a practitioner's standpoint, it might be useful to consider a formulation where the number p of open facilities is a parameter of the problem. This is realized by adding one of the following constraints to the program (1.1)-(1.4)

$$\sum_{j \in J} x_j = p \tag{1.6}$$

or

$$\sum_{j \in J} x_j \leq p \tag{1.7}$$

where p is some given integer, $1 \leq p \leq n$. The formulation (1.1)-(1.4), (1.6) is called the p-facility location problem. When $f_j = 0$ for all $j \in J$, (1.1)-(1.4), (1.6) is known as the p-median problem. This model is the topic of Chapter 2.

In addition to the classical interpretation of the UFL Problem given above, the mathematical model (1.1)-(1.4) has several other applications. We conclude this section by giving two such applications. First we interpret (1.1)-(1.4) as as bank account location problem, (see Cornuejols, Fisher, Nemhauser, 1977b). The second example occurs in clustering analysis, (see Mulvey and Crowder, 1979). Other applications arise in lock-box location (Kraus, Janssen and McAdams, 1970), location of offshore drilling platforms (Balas and Padberg, 1976), economic lot sizing (Krarup and Bilde, 1977), machine scheduling and information retrieval (Hansen and Kaufman, 1972) and portfolio management (Beck and Mulvey, 1982). The formulation (1.1)-(1.4) also arises as a subproblem in several contexts, such as some problems in network design, vehicle routing and, of course, location theory when additional constraints, such as capacity constraints, are present.

A Bank Account Location Problem

The number of days required to clear a check drawn on a bank in city j depends on the city i in which the check is cashed. Thus, to maximize its available funds a company that pays bills to clients in various locations may

find it advantageous to maintain accounts in several strategically located banks. It would then pay bills to clients in city i from a bank in city j that had the largest clearing time. The economic significance to large corporations of such a strategy is discussed in an article in Businessweek (1974).

To formalize the problem of selecting an optimal set of account locations, let $I = \{1,\ldots,m\}$ be the set of client locations, $J = \{1,\ldots,n\}$ the set of potential account locations, f_j the fixed cost of maintaining an account in city j, d_i the dollar volume of checks paid in city i, and ϕ_{ij} the number of days (translated into monetary value) to clear a check issued in city j and cashed in city i. All this information is assumed to be known and $c_{ij} = d_i \phi_{ij}$ represents the value of paying clients in city i from an account in city j. Let $x_j = 1$ if an account is maintained in city j, and $x_j = 0$ otherwise; $y_{ij} = 1$ if clients in city i are paid from account j, and $y_{ij} = 0$ otherwise. Then the account location problem can be stated as (1.1)-(1.4).

Besides desiring to delay payments for as long as possible, corporations also want to collect funds due to them as quickly as possible. This can be done by situating check collection centers or "lock-boxes" at optimal locations. Since (1.1) and (1.1') are mathematically equivalent, (1.1)-(1.4) is also a model for the lock-box problem.

A Clustering Problem

Cluster analysis consists of partitioning objects into classes, known as clusters, in such a way that the elements within a cluster have a high degree of natural association among themselves while the clusters are relatively distinct from one another. Cluster analysis is used in biology, psychology,

medicine, artificial intelligence, pattern recognition, marketing research, automatic classification, statistics, and other areas.

Let $I = \{1,\ldots,m\}$ be the set of objects to be clustered. The clustering is done around objects that "represent" the clusters. Let $J = \{1,\ldots,n\}$ be the set of eligible "cluster representatives". In many applications $J = I$. However in some applications $|J| < |I|$, that is only objects with certain characteristics can qualify as representatives (e.g. survey papers and books in the automatic classification of technical material in some field). In other cases $|I| < |J|$, (e.g. in the automatic classification of technical material, an alternative to the above policy is to represent a cluster by a list of key words. This list does not need to match exactly that of any single paper in the cluster).

The clustering problem is defined relative to a matrix of parameters c_{ij} that represent the similarity between objects i and j. For example c_{ij} could be the number of common key words associated with technical papers i and j. Anderberg (1973) gives several ways of calculating the similarity matrix C. In many applications there are no natural fixed charges, rather, the constraint (1.6) is added to the formulation (1.1)-(1.4) and the problem is solved as a p-median problem. As an alternative, artificial fixed charges can be introduced and then the problem is solved as a UFL problem. It is interesting to note that the method which consists of introducing artificial fixed charges and varying their value has been proposed by some authors as a way of solving the p-median problem for some values of p (Marsten, 1972) and Mavrides, 1979). In general, however, the p-median problem cannot be solved for all values of p in this way.

3.2. Brief Historical Overview

There is a vast literature on the UFL Problem. Krarup and Pruzan (1983) give an up-to-date survey. We will not repeat their effort here. However, to set the stage for the ensuing sections, we will mention the main solution approaches and cite some basic references.

The first approaches were heuristic. One of the earliest of these is due to Kuehn and Hamburger (1963). Their heuristic approach is actually designed for a wider class of location problems. It consists of two routines. The first routine opens facilities sequentially in an order that maximizes the increase of the objective function value at each step. It stops when adding a new facility would decrease the value. Kuehn and Hamburger called it the "add routine"; in the modern literature such a procedure is called a greedy heuristic because of its appetite for maximum improvement at each step. Their second routine is the "bump and shift routine". It eliminates (bumps) any facility that has become uneconomical because of the presence of other facilities chosen subsequently by the greedy heuristic. Then, starting from this feasible solution, it considers interchanging an open facility with a closed facility. Such a pairwise interchange is performed if it improves the current feasible solution and the procedure stops when a solution has been found that cannot be improved by such interchanges. In the remainder of this chapter, the shifting procedure will be referred to as an interchange heuristic.

The greedy and interchange heuristics are the basis of numerous approximation algorithms. They can, of course, be helpful in exact algorithms that require feasible initial solutions or use feasible solutions in other ways, (Spielberg, 1969b; and Hansen and Kaufman, 1972). However,

when a heuristic is used to obtain a final solution, it is very important to have an upper bound as well so that the user can be confident that the heuristic solution value is not too far from the optimal value. Cornuejols, Fisher and Nemhauser (1977b) gave such bounds for the greedy and interchange heuristics, both a priori worst-case bounds and a posteriori bounds on a particular solution constructed by the heuristic. In Section 3.5, heuristic approaches are developed in more detail. In Section 3.9, we mention that some of the heuristic results can be generalized to the maximization of submodular set functions (see Nemhauser, Wolsey and Fisher, 1978).

General solution techniques for finding optimal solutions to integer programs have been specialized to the UFL Problem. The mixed integer linear programming formulation can be solved by Benders decomposition (see Benders, 1962). This approach was proposed by Balinski and Wolfe (1963) and appears to have been the first attempt to solve the UFL Problem to optimality. The computational experiments were discouraging (see Balinski, 1965), and this method was abandoned until recently. Magnanti and Wong (1981) develop techniques to accelerate the convergence of Benders decomposition. They generate strong cuts from the set of feasible Benders cuts and by so doing they are able to reduce the number of integer programs to be solved. Nemhauser and Wolsey (1981) consider the Benders cuts in the more general framework of maximizing a submodular set function.

Branch-and-bound algorithms for the UFL Problem use the fact that it is not necessary to constrain the variables y_{ij} to be integral. Branching is done on a binary enumeration tree with respect to the variables x_j. Bounds are obtained from one of the two linear programming relaxations - (1.1), (1.2), (1.5) and (1.3) or (1.3'). The earlier algorithms used the linear

program with (1.3'), which we call the weak linear programming relaxation. Efroymson and Ray (1966) showed that this linear program can be solved analytically so that the bound at each node of the enumeration tree could be computed very quickly in constant time. Improvements to Efroymson's and Ray's algorithm were made by Spielberg (1969a), Khumawala (1972) and Hansen (1972). However, the bounds obtained from the weak linear programming relaxation are generally not sufficiently strong to curtail the enumeration adequately.

As we observed previously, the constraints (1.3) imply (1.3') but not conversely. The linear programming relaxation that uses (1.3) is called the strong linear programming relaxation (SLPR). ReVelle and Swain (1970), among others, observed that SLPR is so effective that its solution is very often integral. Thus a branch-and-bound algorithm that uses SLPR to compute bounds is very likely to perform well in the sense that very little (if any) enumeration will be required. However, because of its size, it is not efficient to solve SLPR directly by the simplex method.

Much of the recent research on the UFL Problem has involved the development of special purpose algorithms for solving SLPR. Marsten (1972) used parametric linear programming and a special implementation of the simplex method. Garfinkel, Neebe and Rao (1974) used Dantzig-Wolfe decomposition. Schrage (1975) devised a generalized simplex method to treat the variable upper bounds (1.3). Guignard and Spielberg (1977) suggested a version of the simplex method that pivots only to integral vertices of the polytope of feasible solutions to (1.2), (1.3), and (1.6). Cornuejols and Thizy (1982b) used a primal subgradient algorithm.

Dual algorithms or algorithms that solve the dual of the SLPR have the advantage that upper bounds are obtained from any dual feasible solution. Thus a sufficiently good bound may be obtained to fathom a node of the enumeration tree prior to solving the dual to optimality. Duality is discussed in Section 3.4.

Bilde and Krarup (1977) and Erlenkotter (1978) used heuristic methods to obtain a near-optimal solution of the dual. Erlenkotter went a step further by using the complementarity slackness conditions of linear programming to improve this bound. His procedure was so effective that in 45 out of the 48 problems that he tested, optimality was reached at the first node of the branch-and-bound algorithm. His DUALOC code appears to outperform all existing algorithms.

A Lagrangian dual of the formulation (1.1)-(1.4), proposed by Geoffrion (1974), is obtained by weighting the constraints (1.2) by multipliers and placing them in the objective function. It can be solved using subgradient optimization, (see Held, Wolfe and Crowder, 1974). The Lagrangian approach can also be used for the p-median location problem. Some computational results are reported by Narula, Ogbu and Samuelsson (1977), Cornuejols, Fisher, Nemhauser (1977b) and Mulvey and Crowder (1979). Krarup and Pruzan (1983) mention a different Lagrangian dual obtained when constraints (1.3) (instead of (1.2)) are weighted by multipliers and placed in the objective function.

In Section 3.6, we present and relate some of the methods for solving SLPR.

3.3. Computational Complexity

An algorithm is said to be a polynomial-time algorithm for problem P, if for all instances of P (possible data sets), the computing time of the algorithm can be bounded by a polynomial function of the data size. If L measures the data size and k is the order of the polynomial, we say that the computing time of the algorithm is $O(L^k)$. Sometimes it is more convenient to express the computing time as a function of basic data parameters, such as the dimension of a matrix or the number of nodes in a graph. Then, but only then, it is assumed that all arithmetic operations and comparisons are performed in unit time.

A fundamental theoretical question, also of some practical importance, is whether a given combinatorial optimization problem can be solved by some polynomial-time algorithm. Denote by P the class of problems that can be solved in polynomial-time, that is by some polynomial-time algorithm. For most combinatorial optimization problems of practical interest, the question - "Are they in P?" - has not been answered. A significant step was made by Cook (1971) and Karp (1972) who introduced the notion of NP-complete problems. This is a class of combinatorial problems that are equivalent in the sense that either all or none of these problems can be solved by a polynomial-time algorithm.

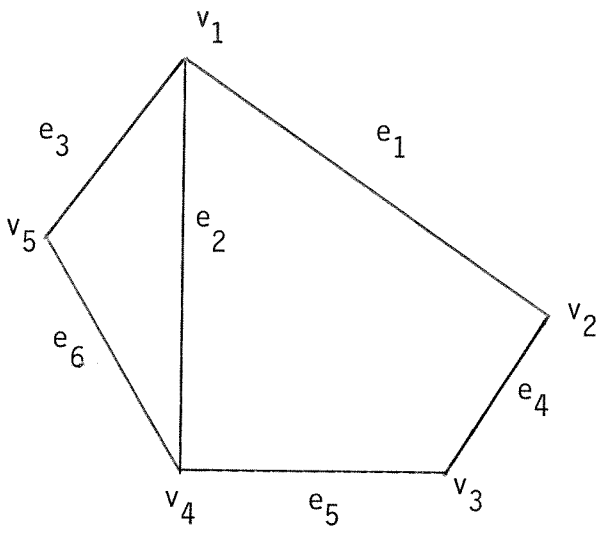
At present no polynomial-time algorithm is known for solving any NP-complete problem and it has been widely conjectured that none exists. A problem is said to be NP-hard if the existence of a polynomial-time algorithm to solve it would imply that all NP-complete problems can be solved by a polynomial-time algorithm. Thus to show that a problem (P) is NP-hard it suffices to find a polynomial transformation that reduces a known NP-complete

problem, (see, e.g., the comprehensive list given by Garey and Johnson, 1979), to the problem (P).

Theorem 3.1 The UFL Problem is NP-hard.

Proof: First we need to introduce the node cover problem. Given a graph G and an integer k, does there exist a subset of k nodes of G that cover all the arcs of G? (Node v is said to cover arc e if v is an endpoint of e.) The node cover problem is NP-complete, (see Karp, 1972 or Garey and Johnson, 1979).

We now reduce the node cover problem to the UFL Problem. Consider a graph $G = (V,E)$ with node set V and arc set E. Construct an instance of the UFL Problem with the set of potential facilities $J = V$ and set of clients $I = E$. Let $c_{ij} = 2$ if $v_j \in V$ is an endpoint of $e_i \in E$ and let $c_{ij} = 0$ otherwise. Also let $f_j = 1$ for all $v_j \in V$. This transformation is polynomial in the size of the graph. A small example of the transformation is shown in Figure 3.1.



$$f = (1 \quad 1 \quad 1 \quad 1 \quad 1)$$

$$C = \begin{pmatrix} 2 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 2 & 0 \\ 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

Figure 3.1.

An instance of the UFL Problem defined in this way consists of covering all the arcs of the graph G with the minimum number of nodes. Thus an optimal solution of the UFL Problem provides the answer to the node cover problem. This proves that the UFL Problem is NP-hard. (In the example, $x_1 = x_3 = x_4 = 1$, $x_j = 0$ otherwise is an optimal solution to the instance of UFL. Hence 3 nodes are needed to cover all of the arcs of G.) □

A polynomial transformation that reduces a known NP-hard problem to a problem (Q) shows that (Q) is also NP-hard. An immediate corollary of Theorem 3.1 is that the p-facility location problem is NP-hard since solving it for every $p = 1,\ldots,n$ provides a solution to the UFL Problem.

Although the UFL Problem is NP-hard, some special cases can be solved in polynomial-time. Kolen (1982) has shown that the UFL Problem is solvable in time $O(r^3)$ when the problem is defined on a tree with r nodes and some other assumptions are imposed. These are (i) the clients as well as the facilities are located at nodes of the tree, (ii) a positive length is associated with each edge of the tree, and (iii) d_{ij} is the length of the path between nodes i and j. Kolen solves the formulation (1.1)', (1.2)-(1.4) and also shows that SLPR always has an integral optimal solution in this case.

Another interesting case of the UFL Problem that can be solved in polynomial time was discovered by Krarup and Bilde (1977). The conditions required by Krarup and Bilde generalize those obtained when a clasical economic lot size problem is formaulated as a UFL Problem. In this problem too, it is very significant that SLPR always has an integral optimal solution.

Finally, Barany, Edmonds and Wolsey (1984) have given a polynomial-time algorithm for a tree partitioning problem that contains both Kolen's and Krarup's and Bilde's problems. These problems are described and studied in Section 3.8.

3.4. Duality

Suppose we are given a feasible solution to the UFL Problem that is claimed to be optimal or nearly optimal (within a specified absolute or relative tolerance). We know of only two ways to verify this claim:

a. enumeration: compare, perhaps implicitly, the value of this feasible solution to all others, and

b. bounding: determine an upper bound on the optimal value of all feasible solutions that is sharp enough to verify the claim.

Enumeration is useful algorithmically only when it can be done implicitly. Generally, this means that the enumerative approach, as in a branch-and-bound algorithm, uses upper bounds to curtail the enumeration. Conversely, an algorithm whose primary thrust is bounding may need to resort to some enumeration to verify the claim.

The point is that good upper bounds, as well as good feasible solutions, are crucial in solving the UFL Problem, as for that matter, any hard combinatorial optimization problem. We will see, however, that the UFL Problem has many features that make it a relatively easy NP-hard problem.

Duality plays a key role in the determination of upper bounds. The dual of the strong linear programming relaxation given by (1.1)-(1.3), (1.5) is

$$W = \min \sum_{i \in I} u_i + \sum_{j \in J} t_j \tag{4.1}$$

$$u_i + w_{ij} \geq c_{ij} \quad \text{all } i \in I, j \in J \tag{4.2}$$

$$-\sum_{i \in I} w_{ij} + t_j \geq -f_j \quad \text{all } j \in J \tag{4.3}$$

$$w_{ij}, t_j \geq 0 \quad \text{all } i \in I, j \in J. \tag{4.4}$$

We can eliminate variables and constraints from this formulation by noting that:

(a.) because of (4.1), we would like to make t_j as small as possible. Thus, for given w_{ij}, (4.3) and (4.4) imply that $t_j = (\sum_{i\in I} w_{ij} - f_j)^+$, where $(\alpha)^+ = \max(0,\alpha)$.

(b.) hence, for given u_i, to make t_j as small as possible, we would like to make w_{ij} as small as possible. Thus (4.2) and (4.4) imply that $w_{ij} = (c_{ij} - u_i)^+$ so that

$$t_j = [\sum_{i\in I} (c_{ij} - u_i)^+ - f_j]^+ \quad \text{all } j \in J. \tag{4.5}$$

If we think of u_i as the price of the ith client, we can interpret t_j as the profits from the jth facility relative to these prices.

Substituting (4.5) into (4.1) yields the condensed dual

$$W = \min_u \{\sum_{i\in I} u_i + \sum_{j\in J} [\sum_{i\in I} (c_{ij} - u_i)^+ - f_j]^+\}. \tag{4.6}$$

The first term in (4.6) is the value of the clients and the second term is the value of the facilities. The dual problem (4.6) is to determine values or prices for the clients to minimize the total value of the resources.

If $\sum_{i\in I} [(c_{ij} - u_i)^+ - f_j]^+ > 0$, then there is a k such that $c_{kj} - u_k > 0$. Hence if u_k is increased by a small amount, then t_j will be decreased by the same amount so that the objective function (4.6) does not increase. Hence there is an optimal solution with $\sum_{i\in I} (c_{ij} - u_i)^+ - f_j \leq 0$ for all $j \in J$. Also if $u_i > \max_{j\in J} c_{ij}$, then u_i can be decreased without increasing the

objective function (4.6). These observations yield a second condensed dual

$$W = \min_{u} \sum_{i \in I} u_i \tag{4.7}$$

$$\sum_{i \in I} (c_{ij} - u_i)^+ - f_j \leq 0 \quad \text{all} \quad j \in J \tag{4.8}$$

$$u_i \leq \max_{j \in J} c_{ij} \quad \text{all} \quad i \in I. \tag{4.9}$$

Although both condensed dual formulations are nonlinear, they are important because they contain only m variables $u = (u_1, u_2, \ldots, u_m)$. Furthermore, since u is unconstrained in (4.6), by duality, we obtain an upper bound on the primal objective function given by

$$W(u) = \sum_{i \in I} u_i + \sum_{j \in J} \left[\sum_{i \in I} (c_{ij} - u_i)^+ - f_j \right]^+. \tag{4.10}$$

When (4.8) holds, the upper bound reduces to

$$W(u) = \sum_{i \in I} u_i. \tag{4.11}$$

A Lagrangian dual of (1.1)-(1.4) is obtained by weighting the constraints (1.2) by multipliers u_i and placing them in the objective function. Let

$$L(u) = \max\left[\sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j + \sum_{i \in I} u_i \left(1 - \sum_{j \in J} y_{ij}\right) \right] \tag{4.12}$$

subject to (1.3) and (1.4).

Frequently, a Lagrangian dual provides a tighter upper bound than a linear programming dual. But this is not the case here.

Proposition 3.2 $L(u) = W(u)$ for all u.

Proof: $L(u) = \max \sum_{i\in I} \sum_{j\in J} (c_{ij} - u_i)y_{ij} - \sum_{j\in J} f_j x_j + \sum_{i\in I} u_i$ subject to (1.3) and (1.4). Hence

$$y_{ij} = \begin{cases} x_j & \text{if } c_{ij} - u_i > 0 \\ 0 & \text{if } c_{ij} - u_i < 0 \\ 0 \text{ or } x_j & \text{if } c_{ij} - u_i = 0. \end{cases} \tag{4.13}$$

Thus

$$L(u) = \max_{x_j\in\{0,1\}} \sum_{j\in J} [\sum_{i\in I} (c_{ij} - u_i)^+ - f_j] x_j + \sum_{i\in I} u_i.$$

Hence

$$x_j = \begin{cases} 1 & \text{if } \sum_{i\in I} (c_{ij} - u_i)^+ - f_j > 0 \\ 0 & \text{if } \sum_{i\in I} (c_{ij} - u_i)^+ - f_j < 0 \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases} \tag{4.14}$$

Thus

$$L(u) = \sum_{j\in J} t_j + \sum_{i\in I} u_i = W(u)$$

where t_j is given by (4.5). □

An immediate consequence of Proposition 3.2 is

Corollary 3.3 $L = \min_u L(u) = W.$

When u satisfies the constraints (4.8), the solution (4.14) satisfies the complementarity conditions

$$\left(\sum_{i \in I} (c_{ij} - u_i)^+ - f_j\right)x_j = 0 \quad \text{all} \quad j \in J. \tag{4.15}$$

Equation (4.15) suggests that if u satisfies (4.8), then $x_j = 0$ unless $\sum_{i \in I} (c_{ij} - u_i)^+ - f_j = 0$. In other words, to find a good primal solution, we should only consider opening those facilities for which $\sum_{i \in I} (c_{ij} - u_i)^+ - f_j = 0$.

The results of this section will be used in the next two sections in the development of solution techniques for the UFL Problem.

3.5. Heuristics

The combinatiorial formulation of the UFL Problem $\max_{S \subseteq J} z(S)$, where $z(S) = \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j$ can be viewed as a condensed, nonlinear primal that depends only on the values of the sets $S \subseteq J$. Based on this observation numerous authors have proposed heuristics that iterate on the set S of open locations and avoid an explicit integer programming formulation. Primal heuristics of this type are described in Section 3.5.1 and a dual descent heuristic is given in Section 3.5.2.

3.5.1 Primal Heuristics

The Greedy Heuristic.

The greedy heuristic starts with no facilities open. Given a set S of open facilities, the greedy heuristic adds to S that facility $j \notin S$ whose incremental value $\rho_j(S) = z(S \cup \{j\}) - z(S)$ is as large as possible, and is positive. If no such facility exists, it stops with the set S of open facilities. Formally, the procedure is stated as follows:

Initialization $S^0 = \emptyset$, $\rho_j(\phi) = \sum_{i \in I} c_{ij} - f_j$, $t = 1$.

Iteration t Find $j_t = \arg\max_{j \notin S^{t-1}} \rho_j(S^{t-1})$.

If $\rho_{j_t}(S^{t-1}) \leq 0$, stop. The set S^{t-1} is the greedy solution with value $Z^G = z(S^{t-1})$ if $t > 1$. If $t = 1$, the greedy solution is $S^1 = \{j_1\}$. (Note that when $t = 1$, some facility must be opened since $S = \emptyset$ is not feasible).

If $\rho_{j_t}(S^{t-1}) > 0$, $S^t = S^{t-1} \cup \{j_t\}$.

Set $t \leftarrow t + 1$.

The greedy heuristic requires at most n iterations and each iteration requires $O(nm)$ calculations. Thus the overall running time is $O(n^2m)$.

An Example

We will use the following small example to introduce and motivate the ideas developed subsequently. Real-world problems are typically much larger (e.g. $m = n = 100$). Consider the instance of the UFL Problem defined by the data:

$$m = 4,\ n = 6,\ f = (3,2,2,2,3,3) \text{ and}$$

$$C = \begin{pmatrix} 6 & 6 & 8 & 6 & 0 & 6 \\ 6 & 8 & 6 & 0 & 6 & 6 \\ 5 & 0 & 3 & 6 & 3 & 0 \\ 2 & 3 & 0 & 2 & 4 & 4 \end{pmatrix}.$$

Applying the greedy heuristic to the example yields

iteration 1: $(\rho_1(\phi),\ldots,\rho_6(\phi)) = (16,15,15,12,10,13)$.

Hence $j_1 = 1$ and $S^1 = \{1\}$.

iteration 2: $(\rho_2(\{1\}),\ldots,\rho_6(\{1\})) = (1,0,-1,-1,-1)$.

Hence $j_2 = 2$ and $S^2 = \{1,2\}$.

iteration 3: $(\rho_3(\{1,2\}),\ldots,\rho_6(\{1,2\})) = (0,-1,-2,-2)$.

The set S^2 of value $Z^G = z(S^2) = 17$ is the greedy solution.

We can now use (4.10) to obtain upper bounds on the values of the feasible solutions produced by the greedy heuristic. In fact, such an upper bound can be associated with any $S \subseteq J$.

Define

$$\bar{u}_i(S) = \max_{j \in S} c_{ij} \quad \text{all} \quad i \in I. \tag{5.1}$$

Note that

$$z(S) = \sum_{i \in I} \bar{u}_i(S) - \sum_{j \in S} f_j \tag{5.2}$$

and

$$\rho_j(S) = \sum_{i \in I} (c_{ij} - \bar{u}_i(S))^+ - f_j \quad \text{all} \quad j \notin S.$$

Thus, from (4.10)

$$W(\bar{u}(S)) = \sum_{i \in I} \bar{u}_i(S) + \sum_{j \notin S} [\rho_j(S)]^+$$

since $c_{ij} - \bar{u}_i(S) \leq 0$ for all $i \in I$ and $j \in S$.

In particular if S^G is the final set chosen by the greedy heuristic then, by the stopping criterion, $\rho_j(S^G) \leq 0$, $j \notin S^G$, so that $W(\bar{u}(S^G)) = \sum_{i \in I} \bar{u}_i(S^G)$. Hence by (5.2), the greedy solution S^G deviates from optimality by at most $\sum_{j \in S^G} f_j$. This suggests that it will yield a small error when the fixed costs are small in comparison to the profits. Furthermore, we may obtain a better bound by considering all of the sets produced by the greedy algorithm. Let $\bar{u}_i^0 = \min_{j \in J} c_{ij}$ all $i \in I$ and $\bar{u}^k = \bar{u}(S^k)$, $k = 1,\ldots,t-1$. Define a dual greedy value by $W^G = \min_k W(\bar{u}^k)$. In the example, $\bar{u}^0 = (0\ 0\ 0\ 0)$, $W(\bar{u}^0) = \sum_{j \in J} \rho_j(\phi) = 81$, $\bar{u}^1 = (6,6,5,2)$,

$W(\bar{u}^1) = \sum_{i\in I} \bar{u}_i^1 + 1 = 20$, $\bar{u}^2 = (6,8,5,3)$ and $W(\bar{u}^2) = \sum_{i\in I} \bar{u}_i^2 = 22$. Hence $W^G = 20$.

The bound we have given so far is an a posteriori bound for a particular instance of the UFL Problem. In fact, a general relationship between Z^G and W^G that a priori applies to all instances of the UFL Problem is given by

Theorem 3.4 [Cornuejols, Fisher and Nemhauser (1977b)]. Let $R = \sum_{i\in I} \min_{j\in J} c_{ij} - \sum_{j\in J} f_j$ and e be the base of the natural logarithm.

Then

$$Z^G \geq (\frac{e-1}{e})W^G + (\frac{1}{e})R \ .$$

A proof of Theorem 3.4 that uses linear programming duality is given in Fisher, Nemhauser and Wolsey [1978].

For the p-median problem with $c_{ij} \geq 0$ for all i and j, we have $R \geq 0$. Thus we achieve a simple data independent statement of Theorem 3.4.

Corollary 3.5 [Cornuejols, Fisher and Nemhauser (1977)] For the p-median problem with $c_{ij} \geq 0$ for all i and j,

$$\frac{Z^G}{W^G} \geq \frac{e-1}{e} \simeq 0.63.$$

There are families of p-median problems for which this bound is achieved asymptotically. Furthermore, since $Z^G \leq Z \leq W \leq W^G$

$$\max\left\{\frac{Z^G}{Z}, \frac{Z}{W}, \frac{W}{W^G}\right\} \geq \left(\frac{e-1}{e}\right)^{1/3} \simeq 0.86.$$

There are several variations and generalizations of the greedy heuristic for which bounds similar to those of Theorem 3.4 and Corollary 3.5 are known, see Cornuejols, Fisher and Nemhauser [1977b] and Nemhauser, Wolsey and Fisher [1978]. For example, we can begin with all facilities open and at each iteration close a facility that gives the largest improvement in the objective function so long as such a facility exists. A generalization of the greedy heuristic is to start with the family consisting of all sets of cardinality k, for some fixed k, and apply greedy to each of these $\binom{n}{k}$ initial sets separately; we then choose the best of the resulting $\binom{n}{k}$ solutions.

None of these variations or generalizations, however, improve the worst-case bound of the greedy heuristic. In fact, what is remarkable about the bound on the greedy heuristic is not its value, but that no polynomial-time procedure of any degree whatsoever is known for p-median problems that has a better worst-case performance.

The salient feature of the greedy heuristic is that the maximum possible improvement is made at each step. If this is not done, worst-case performance deteriorates, even if a broader choice of improvements is considered.

An example of such a heuristic is <u>generalized interchange</u>, see Nemhauser, Wolsey and Fisher (1978). Here we begin with an arbitrary set S^0. Given S^{t-1}, at iteration t we select any set S^t such that $z(S^t) > z(S^{t-1})$, $|S^t \backslash S^{t-1}| \leq 1$ and $|S^{t-1} \backslash S^t| \leq 1$ or stop if no such S^t exists. Thus, at each iteration, we are allowed to open a facility, close a facility or do both so long as an improvement is made.

The worst-case bound of generalized interchange is weaker than that of greedy. For example, under the conditions of Corollary 3.5, this heuristic can guarantee only to find a solution of value at least half of the optimum value. Of course, by starting with a greedy solution, the bounds of Theorem 3.4 and Corollary 3.5 are obtained. Nevertheless, they are not strengthened by applying generalized interchange. On the other hand, greedy followed by generalized interchange seems to give good empirical performance, (see Hansen, 1972, and Cornuejols, Fisher and Nemhauser, 1977b).

The dual solution $\overline{u}(S)$ given by (5.1) yields a bound on the value of the primal solution S. Conversely, given a dual solution that satisfies (4.8), the complementarity conditions (4.15) suggest considering a primal solution in which $x_j = 0$ if $\sum_{i \in I} (c_{ij} - u_i)^+ - f_i < 0$.

Define

$$J(u) = \{j: \sum_{i \in I} (c_{ij} - u_i)^+ - f_j = 0\}.$$

An optimal solution with $x_j = 0$ for $j \notin J(u)$ is obtained by solving

$$\max_{S \subseteq J(u)} \{ \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j \},$$

but this problem may not be much easier to solve than the original problem. Instead, we take any minimal set $K(u) \subseteq J(u)$ that satisfies

$$\max_{j \in K(u)} c_{ij} = \max_{j \in J(u)} c_{ij} \quad \text{all } i \in I. \tag{5.3}$$

Proposition 3.6 Given a u that satisfies (4.8) and $u_i \leq \max_{j \in J(u)} c_{ij}$ all $i \in I$, and a primal solution $K(u)$ defined by (5.3), let $k_i = \left|\{j \in K(u): c_{ij} > u_i\}\right|$. If $k_i \leq 1$ all $i \in I$, then $K(u)$ is an optimal set of open facilities.

Proof:

$$z(K(u)) = \sum_{i \in I} \max_{j \in K(u)} c_{ij} - \sum_{j \in K(u)} f_j.$$

If $k_i = 0$

$$\max_{j \in K(u)} c_{ij} = u_i = u_i + \sum_{j \in K(u)} (c_{ij} - u_i)^+$$

and if $k_i = 1$

$$\max_{j \in K(u)} c_{ij} = u_i + \sum_{j \in K(u)} (c_{ij} - u_i)^+.$$

Hence, if $k_i \leq 1$ all $i \in I$,

$$z(K(u)) = \sum_{i \in I} \sum_{j \in K(u)} (c_{ij} - u_i)^+ - \sum_{j \in K(u)} f_j + \sum_{i \in I} u_i$$

$$= \sum_{j \in K(u)} \left(\sum_{i \in I} (c_{ij} - u_i)^+ - f_j \right) + \sum_{i \in I} u_i$$

$$= \sum_{i \in I} u_i = W(u) \quad \text{by (4.11).} \quad \square$$

In our example, with $u = (6\ 6\ 4\ 3)$, we obtain $J(u) = K(u) = \{2,3,4\}$ and $(k_1,k_2,k_3,k_4) = (1\ 1\ 1\ 0)$. Hence these are optimal primal and dual solutions of value $\sum_{i\in I} u_i = 19$.

While Proposition 3.6 may permit us to recognize an optimal solution, it is limited to those cases in which $\min_u W(u) = Z$ and even then it is still necessary to find an appropriate u and $K(u)$.

3.5.2 Dual Descent

Dual descent is a heuristic that begins with a u satisfying (4.8) and then attempts to decrease the u_i's one-at-a-time while maintaining (4.8), (see Erlenkotter, 1978, and Bilde and Krarup, 1977). It is surprisingly effective, but not fail safe, in finding a u that satisfies the conditions of Proposition 3.6. This descent approach, with some embellishments, is the inner loop of Erlenkotter's DUALOC algorithm. The basic descent method procedes as follows:

Begin with $u_i^0 = \max_{j\in J} c_{ij}$ all $i \in I$. Cycle through the indices $i \in I$ attempting to decrease u_i while satisfying the constraints (4.8). If u_i cannot be decreased, then consider u_{i+1}. If u_i can be decreased, then decrease it to $\max\{c_{ij}: c_{ij} < u_i\}$ if this change satisfies (4.8). If not, then decrease u_i to the minimum value allowed by (4.8). When all of the u_i's are blocked from further decreases, the procedure terminates. The reason for decreasing u_i only to the next smaller c_{ij}, rather than to the smallest permissible value, is to keep the k_i of Proposition 3.6 as small as possible.

Applying dual descent to our example yields the results shown in Table 3.1. For the first four steps, each

Step	u^T				$f_j - \sum_{i \in I} (c_{ij} - u_i)^+$					
0	8	8	6	4	3	2	2	2	3	3
1	6	8	6	4	3	2	0	2	3	3
2	6	6	6	4	3	0	0	2	3	3
3	6	6	5	4	3	0	0	1	3	3
4	6	6	5	3	3	0	0	1	2	2
5	6	6	4	3	2	0	0	0	2	2

Table 3.1

of the u_i's is decreased to the second max in the row. Now u_1 is considered again, but cannot be decreased because the constraints (4.8) for $j = 3$ would be violated. Similarly, a decrease of u_2 would violate (4.8) for $j = 2$. However, u_3 can be decreased; but it is decreased only to 4 because (4.8) becomes active for $j = 4$ when $u_3 = 4$. Finally u_4 cannot be decreased because of (4.8) for $j = 2$. This completes the dual descent with $u = (6\ 6\ 4\ 3)$ and $W(u) = \sum_{i \in I} u_i = 19$. Now, as noted above, $J(u) = K(u) = \{2,3,4\}$ and we also obtain a primal solution of value 19.

A possible improvement of dual descent, which is likelier to produce a primal and dual pair for which Proposition 3.6 applies, is obtained by modifying the order in which the u_i's are considered as candidates to decrease. In particular, rather than just cycling through the u_i's, let $Q_i(u) = \{j: c_{ij} - u_i \geq 0\}$. Then if u_i is decreased and descent terminates, $k_i \leq |Q_i(u)|$. Hence we choose u_s next if $|Q_s(u)| \leq |Q_i(u)|$ for all $i \in I$.

Suppose dual descent terminates with the dual solution u^* and we determine a primal solution given by $K(u^*)$ such that Proposition 3.6 fails to verify optimality. Then there exists an i such that $k_i > 1$. By increasing u_i to its previous value and reapplying dual descent, it may be possible to improve the dual solution further.

We have sketched the basic ideas of the dual descent procedure. An example in which it does not find an optimal solution is given by

$$f = (2\ 2\ 2), \qquad C = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix}. \tag{5.4}$$

In this instance, beginning with $u^0 = (2\ 2\ 2)$, dual descent terminates with $u = (0\ 2\ 2)$, which is not optimal. Moreover, the embellishments do not give an optimal solution either. An optimal u is $(1\ 1\ 1)$ yielding $W = 3$ and $Z = 2$. Nevertheless, dual descent has performed extremely well on the problems that Erlenkotter and others have considered. Erlenkotter reports that in 45 of 48 problems tested, the heuristic found an optimal solution. To provide the capability of finding an optimal solution and proving optimality, the heuristic is imbedded in a branch-and-bound algorithm. The whole procedure is called DUALOC. Given its simplicity, speed and availability, DUALOC may be the most efficient way to solve the UFL Problem. However, it may perform poorly on hard problems in which the heuristic bound is not as good as the linear programming bound. Thus one is motivated to develop efficient algorithms for solving the strong linear programming relaxation. This is the subject of the next section.

3.6. Algorithms and Reformulations of the Strong Linear Programming Relaxation

The strong linear programming relaxation (SLPR) of the UFL Problem is

$$Z_{LP} = \max \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} - \sum_{j \in J} f_j x_j \tag{6.1}$$

$$\sum_{j \in J} y_{ij} = 1 \qquad \text{all } i \in I \tag{6.2}$$

$$y_{ij} - x_j \leq 0 \qquad \text{all } i \in I,\ j \in J \tag{6.3}$$

$$y_{ij} \geq 0,\ x_j \geq 0 \qquad \text{all } i \in I,\ j \in J. \tag{6.4}$$

The polyhedron of feasible solutions to (6.2), (6.3) and (6.4) has fractional extreme points. In the example given by (5.4), the unique optimal solution to SLPR is $x_j = 1/2$ for $j = 1,2,3$, $y_{ij} = 1/2$ for $i \neq j$ and $y_{ij} = 0$ for $i = j$. The fractional extreme points of this polyhedron will be characterized in the next section. However, for reasons that are barely understood, very frequently, SLPR has an optimal integral solution. This observation is supported by results in the literature on random problems and some applications. Thus an efficient method for solving SLPR will also be an efficient method for solving many instances of the UFL Problem. Even when SLPR has a fractional optimal solution, its optimal value generally provides a very good upper bound on the optimal value of UFL so that a branch-and-bound algorithm should terminate rapidly.

On the basis of these remarks, one might conclude that the UFL Problem can be solved using SLPR and a standard mixed integer programming package. However, this is not true because of the large number of constraints (6.3).

For example, an instance of the UFL Problem with $m = n = 100$ has more than 10,000 constraints. Hence any approach to the UFL Problem that uses the SLPR must be capable of solving very large linear programs.

In this section, we consider several approaches for solving very large structured linear programs. We begin the section by briefly mentioning two direct approaches to eliminating the difficulty caused by the large number of constraints (6.3). Then we will apply some well-known reformulations and algorithms including Lagrangian duality and subgradient optimization, Dantzig-Wolfe decomposition, Benders decomposition, and subgradient optimization on the primal. Connections among those approaches will be noted. Finally, we will consider a reformulation that involves the aggregation and disaggregation of clients so that the matrix C reduces to a useful canonical form.

3.6.1 Direct Approaches

The constraints $y_{ij} \leq x_j$ are generalizations of simple upper bound constraints in which the upper bounds themselves are variables. It is well-known how to handle fixed upper bound constraints in the simplex method without expanding the dimension of the basis to include them. Schrage (1975) has generalized this idea to incorporate variable upper bounds and has reported computational results obtained by applying the method to SLPR.

An alternative is to generate the constraints $y_{ij} \leq x_j$ as cuts only when they are violated in an optimal solution to the weak linear programming relaxation. This idea has been tested by Morris (1978). In the example of the last section, an optimal solution to the weak linear programming relaxation is $x_2 = x_3 = x_4 = x_5 = 1/4$ and $y_{13} = y_{22} = y_{34} = y_{45} = 1$. We could

then add the four violated variable upper bound constraints and continue to solve the linear program.

The direct approaches are primal methods. Dual methods, however, may be superior for two reasons. First, if SLPR is incorporated in a branch-and-bound algorithm, it may not be necessary to solve SLPR to optimality at every node of the enumeration tree, since at some of the nodes, a dual feasible solution may suffice to bound the subproblem. Second, as we have already shown in the last section, one may easily generate an integral primal solution from a given dual solution.

3.6.2 Lagrangian Duality and Subgradient Optimization

The Lagrangian $L(u)$ of (4.12) forms the basis of a subgradient algorithm for solving the dual of SLPR. The subgradient algorithm solves the problem $\min_u L(u)$.

The function $L(u)$ given by (4.12) is the maximum of a finite number of linear functions. Therefore $L(u)$ is piecewise linear and convex. Subgradient optimization (Held, Wolfe and Crowder, 1974) has proved to be a useful method for minimizing unconstrained piecewise linear convex functions. This approach is a generalization of the gradient method for minimizing smooth, nonlinear convex functions. Since gradients do not exist at non-differentiable points of $L(u)$, the gradient direction is replaced by a "subgradient direction", which will be explained below.

Given u^t, an iteration of the subgradient algorithm generates a new dual solution by the formula

$$u^{t+1} = u^t - \gamma^t \partial L(u^t) \qquad (6.2.1)$$

where $\partial L(u^t)$ is a subgradient at u^t and γ^t is the stepsize. If $\partial L(u^t) = 0$, then u^t is an optimal dual solution.

Suppose

$$L(u) = \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij}^* - \sum_{j \in J} f_j x_j^* + \sum_{i \in I} u_i (1 - \sum_{j \in J} y_{ij}^*)$$

where $\{x_j^*, y_{ij}^*\}$ are defined by (4.13) and (4.14). If the $\{y_{ij}^*\}$ are unique then

$$\partial L(u)_i = 1 - \sum_{j \in J} y_{ij}^* \quad \text{all} \quad i \in I \qquad (6.2.2)$$

is the gradient of $L(u)$ at u. However if the $\{y_{ij}^*\}$ are not unique, then any direction given by (6.2.2) or convex combinations of such directions is a subgradient direction. Although a step in a subgradient direction does not guarantee a decrease in $L(u)$, it can be proved that with an appropriate choice of stepsize the iterates given by (6.2.1) converge to an optimal solution (Polyak, 1969). Cornuejols, Fisher and Nemhauser (1977b) have reported computational results on solving the Lagrangian dual by subgradient optimization.

In our example, if we start with $u^0 = (8\ 8\ 6\ 4)$, then $L(u^0) = 26$ and $\partial L(u) = (1\ 1\ 1\ 1)$. With a stepsize of $\gamma^0 = 2$, we obtain $u^1 = (6\ 6\ 4\ 2)$ and $L(u^1) = 19$. With $u = u^1$, the solution (4.13), (4.14) is not unique; however, the optimal solution $x_2 = x_4 = 1$, $x_j = 0$ otherwise, and $y_{12} = y_{22} = y_{34} = y_{42} = 1$, $y_{ij} = 0$ otherwise, yields $\partial L(u^1) = (0\ 0\ 0\ 0)$ and verifies the optimality of u^1.

3.6.3 Dantzig-Wolfe Decomposition

For all non-empty $R \subseteq I$, let $\lambda_j^R = 1$ if facility j serves only these clients in the set R and $\lambda_j^R = 0$ otherwise. If $\lambda_j^R = 1$, facility j yields a profit of $\sum_{i \in R} c_{ij} - f_j$. Thus the UFL Problem can be reformulated as the integer program

$$Z = \max \sum_{j \in J} \sum_{R \subseteq I} \left(\sum_{i \in R} c_{ij} - f_j \right) \lambda_j^R \tag{6.3.1}$$

$$\sum_{j \in J} \sum_{R \ni i} \lambda_j^R = 1 \qquad \text{all } i \in I \tag{6.3.2}$$

$$\sum_{R \subseteq I} \lambda_j^R \leq 1 \qquad \text{all } j \in J \tag{6.3.3}$$

$$\lambda_j^R \in \{0,1\} \quad \text{all } R \subseteq I,\ j \in J. \tag{6.3.4}$$

The equations (6.3.2) state that each client is served by exactly one facility and the inequalities (6.3.3) state that each facility can serve only one set of clients.

We consider the linear programming relaxation of this integer program obtained by replacing (6.3.4) by

$$\lambda_j^R \geq 0 \quad \text{all } R \subseteq I,\ j \in J. \tag{6.3.5}$$

Proposition 3.7: Let $\overline{Z}$ be the optimal value of the linear program (6.3.1), (6.3.2), (6.3.3) and (6.3.5). Then $\overline{Z} = Z_{LP}$.

Proof: Apply Dantzig-Wolfe decomposition to the linear program (6.1)-(6.4) with master constraints (6.2) and subproblem constraints (6.3), (6.4) and $x_j \leq 1$ all $j \in J$. Substituting the subproblem extreme points into (6.2) yields (6.3.2), and (6.3.3) are the convexity constraints for the subproblems. □

Some simplifications can be made in the integer program (6.3.1)-(6.3.4) and its linear programming relaxation. First observe that the constraints (6.3.3) are unnecessary. To see this, suppose $\lambda_j^R = \lambda_j^{R'} = 1$ where $R \neq R'$. If $R \cap R' \neq \emptyset$, then (6.3.2) will be violated. If $R \cap R' = \emptyset$, then since $f_j \geq 0$, we have

$$\sum_{i \in R \cup R'} c_{ij} - f_j \geq \sum_{i \in R} c_{ij} - f_j + \sum_{i \in R'} c_{ij} - f_j.$$

Hence the alternate solution with $\lambda_j^{R \cup R'} = 1$ and $\lambda_j^R = \lambda_j^{R'} = 0$ is at least as good.

A further simplification is obtained by noting that if $\sum_{i \in R} c_{ij} - f_j > \sum_{i \in R} c_{ik} - f_k$, then $\lambda_k^R = 0$ in every optimal solution. Hence for each R, we need only one variable, say λ_R, whose profit is given by

$$d_R = \max_{j \in J} \left(\sum_{i \in R} c_{ij} - f_j \right).$$

Thus we can restate the integer program as

$$\begin{aligned} Z = \max \quad & \sum_{R \subseteq I} d_R \lambda_R \\ & \sum_{R \ni i} \lambda_R = 1 \qquad \text{all } i \in I \qquad (6.3.6) \\ & \lambda_R \in \{0,1\} \quad \text{all } R \subseteq I. \end{aligned}$$

This formulation has 2^{m-1} variables and m constraints. Since n does not enter into the size of this formulation, for fixed m, it can be solved in polynomial-time.

The linear program obtained from (6.3.6) by replacing the integrality requirement by $\lambda_R \geq 0$ all $R \subseteq I$ can be solved by column generation. Suppose we begin with m columns, say those for $R = \{i\}$, $i = 1,\ldots,m$. Then the primal solution is $\lambda_R^0 = 1$ for $R = \{i\}$, $i = 1,\ldots,m$, and the dual solution is $u_i^0 = \max_{j \in J}(c_{ij} - f_j)$ all $i \in I$.

We now need to see if any of the nonbasic columns have a positive price. This can be done at iteration p by solving for each $j \in J$ the subproblem

$$t_j^p = \max \sum_{i \in I} (c_{ij} - u_i^p)y_{ij} - f_j x_j$$

subject to (1.3) and (1.4), since $\sum_{i \in R} (c_{ij} - u_i^p) - f_j$ is the price of variable λ_j^R. The solution of the subproblem is $t_j^p = (\sum_{i \in I} (c_{ij} - u_i^p)^+ - f_j)^+$. If $t_j^p = 0$ all $j \in J$ then the current solution is optimal. For each k such that $t_k^p > 0$, let R_k be any set satisfying $\{i: c_{ik} - u_i^p > 0\} \subseteq R_k \subseteq \{i: c_{ik} - u_i^p \geq 0\}$. We now add to the linear program variables λ_{R_k} for all k such that $t_k^p > 0$.

Garfinkel, Neebe and Rao (1974) have obtained computational experience with this type of algorithm.

An important feature of this approach is that both lower and upper bounds on Z_{LP} are obtained at each iteration. By primal feasibility, $Z_{LP} \geq \sum_{i \in I} u_i^p$; and from (4.12) we see that $Z_{LP} \leq \sum_{j \in J} t_j^p + \sum_{i \in I} u_i^p = W(u^p)$. Moreover, as well as obviously being a primal method, it also can be viewed as a dual method and compared with the method that uses Lagrangian duality and subgradient

optimization. Here the u_i's at each iteration are determined by solving a linear program, while in the previous method the dual variables are determined by moving in the direction of a subgradient of the function $L(u)$.

In our example, we start with an initial basis consisting of the four unit columns $R_i = \{i\}$, $i = 1,\ldots,4$ with objective coefficients $d_{R_1} = d_{R_2} = 6$, $d_{R_3} = 4$ and $d_{R_4} = 1$. This yields the dual solution $u^0 = (6\ 6\ 4\ 1)$. Solving the subproblems, we obtain $t^0 = (0\ 2\ 0\ 1\ 0\ 0)$ and thus $17 \leq Z_{LP} \leq 20$. We generate two new columns $R_5 = \{1\ 2\ 4\}$ and $R_6 = \{3\ 4\}$ with objective coefficients $d_5 = 15$ and $d_6 = 6$. The next master linear program yields the primal solution $\lambda^1_{R_5} = \lambda^1_{R_3} = 1$, $\lambda^1_{R_i} = 0$ otherwise and the dual solution $u^1 = (6\ 6\ 4\ 3)$. Now $t^1_j = 0$ all $j \in J$ so that the primal has been solved. In terms of the original variables, we have $x_2 = x_4 = 1$, $x_j = 0$ otherwise.

3.6.4 Primal Subgradient Algorithm

For a given x, the profit from the ith client is

$$
\begin{aligned}
V_i(x) = \max \quad & \sum_{j \in J} c_{ij} y_{ij} \\
& \sum_{j \in J} y_{ij} = 1 \qquad (6.4.1) \\
& y_{ij} \leq x_j \quad \text{all } j \in J \\
& y_{ij} \geq 0 \quad \text{all } j \in J.
\end{aligned}
$$

It is known from the theory of parametric linear programming that the function $V_i(x)$ is piecewise linear and concave. Hence we can formulate SLPR as the maximization of a piecewise linear concave function subject to simple constraints. In particular,

$$Z_{LP} = \max \sum_{i \in I} V_i(x) - \sum_{j \in J} f_i x_j$$
$$0 \leq x_j \leq 1 \quad \text{all} \quad j \in J. \tag{6.4.2}$$

Technically, (6.4.2) also requires the constraint $\sum_{j \in J} x_j \geq 1$, but this constraint can be omitted by adding a suitably large constant to any row of the profit matrix C. In the remainder of this chapter, we assume that $\sum_{j \in J} x_j \geq 1$ is satisfied by every optimal solution to (6.4.2).

The formulation given by (6.4.2) can be solved by subgradient optimization. Let $\partial Z(x)_j$ be the jth component of a subgradient of the objective function at x. We have $\partial Z(x)_j = \sum_{i \in I} \partial V_i(x)_j - f_j$ where $\partial V_i(x)_j$ is the jth component of a subgradient of $V_i(x)$.

To obtain a closed form expression for $\partial V_i(x)_j$, we consider the linear program (6.4.1), which can be solved by a greedy method. In particular, suppose $c_{ij_1} \geq c_{ij_2} \geq \ldots \geq c_{ij_n}$ and define p_i by $\sum_{k=1}^{p_i - 1} x_{j_k} < 1 \leq \sum_{k=1}^{p_i} x_{jk}$. Then an optimal solution to (6.4.1) is given by $y_{ij_k} = x_{j_k}$ for $k = 1, \ldots, p_i - 1$, $y_{ij_{p_i}} = 1 - \sum_{k=1}^{p_i} x_{j_k}$ and $y_{ij_k} = 0$ otherwise. Thus we can take $\partial V_i(x)_j = (c_{ij} - c_{ip_i})^+$ and

$$\partial Z(x)_j = \sum_{i \in I} (c_{ij} - c_{ip_i})^+ - f_j \quad \text{all} \quad j \in J. \tag{6.4.3}$$

Given x^t, an iteration of the subgradient algorithm generates a new point $x^{t+1} = x^t - \gamma^t \partial Z(x^t)$. If x^{t+1} does not lie in the cube $0 \leq x_j \leq 1$, all $j \in J$, and therefore violates some of the constraints of (6.4.2), we modify it by simply projecting it on the cube, that is we replace x_j^{t+1} by 0 if $x_j^{t+1} < 0$ and by 1 if $x_j^{t+1} > 1$.

Cornuejols and Thizy (1982b) report some computational experience in solving (6.4.2) by a subgradient algorithm.

The results of attempting to solve (6.4.2) by subgradient optimization for our example are summarized in Table 3.2. An optimal solution to SLPR is obtained at iteration 3, but the expression (6.4.3) for the subgradient is not adequate to verify optimality.

iteration	x						V(x)				$Z_{LP}(x)$	∂V(x)						step size
0	0	1	1	1	1	0	8	8	6	4	17	-3	-2	-2	-2	-3	-3	1/4
1	0	1/2	1/2	1/2	1/4	0	7	7	9/2	3	71/4	-1	1	0	1	-1	-1	1/4
2	0	3/4	1/2	3/4	0	0	7	15/2	21/4	11/4	37/2	-1	1	0	1	-1	-1	1/4
3	0	1	1/2	1	0	0	7	8	6	3	19	-3	-2	0	-2	-2	-2	

Table 3.2

3.6.5 Benders Decomposition

We can state (6.4.2) as

$$Z_{LP} = \max \sum_{i \in I} V_i - \sum_{j \in J} f_j x_j$$

$$V_i \leq V_i(x) \quad \text{all } i \in I \text{ and all feasible } x \qquad (6.5.1)$$

$$0 \leq x_j \leq 1 \qquad \text{all } j \in J.$$

Now we observe that

$$V_i(x) = \min u_i + \sum_{j \in J} x_j w_{ij}$$

$$u_i + w_{ij} \geq c_{ij} \quad \text{all } j \in J \qquad (6.5.2)$$

$$w_{ij} \geq 0$$

since (6.5.2) is the dual of (6.4.1). We have $w_{ij} = (c_{ij} - u_i)^+$ so that

$$V_i(x) = \min(u_i + \sum_{j \in J} x_j(c_{ij} - u_i)^+) \tag{6.5.3}$$

The right-hand side of (6.5.3) is a piecewise linear function of u_i and the function changes slope when $u_i = c_{ij}$ for some j. Hence the minimum is attained when $u_i = c_{ik}$ for some $k \in J$. Hence

$$V_i(x) = \min_{k \in J}(c_{ik} + \sum_{j \in J} x_j(c_{ij} - c_{ik})^+)$$

and (6.5.1) can be stated as the linear program

$$\begin{aligned} Z_{LP} = \max & \sum_{i \in I} V_i - \sum_{j \in J} f_j x_j \\ & V_i - \sum_{j \in J} x_j(c_{ij} - c_{ik})^+ \leq c_{ik} \quad \text{all } i \in I,\ k \in J \\ & 0 \leq x_j \leq 1. \end{aligned} \tag{6.5.4}$$

This is precisely the linear program that arises when applying Benders decomposition to SLPR. Although we have mn constraints of the form (6.5.4), we can think of these as cutting planes and generate them only as they are needed.

In particular, suppose we have only a proper subset of the constraints (6.5.4). We solve the relaxed linear program and determine an optimal solution (x^q, V^q). Now we use x^q in (6.4.1) to determine $V_i(x^q)$ all $i \in I$. If

$$V_i(x^q) \leq V_i^q \quad \text{all } i \in I, \tag{6.5.5}$$

then (x^q, V^q) satisfies all of the constraints (6.5.4) and x^q is an optimal

solution. If not, each i for which (6.5.5) is violated specifies a violated constraint of the form (6.5.4). These are added to the linear program and we continue.

In our example, we begin with the constraints (6.5.4) determined by the second maximum in each row, i.e., $V_1 - 2x_3 \leq 6$, $V_2 - 2x_2 \leq 6$, $V_3 - x_4 \leq 5$, and $V_4 \leq 4$. A solution to the linear program is $V^1 = (6\ 8\ 5\ 4)$ and $x^1 = (0\ 1\ 0\ 0\ 0\ 0)$. Then by solving (6.4.1) we obtain $V(x^1) = (6\ 8\ 0\ 3)$ and generate the constraints

$$V_3 - 5x_1 - 3x_3 - 6x_4 - 3x_5 \leq 0$$

$$V_4 - x_5 - x_6 \leq 3.$$

Now we obtain the solution $V^2 = (6\ 8\ 6\ 3)$ and $x^2 = (0\ 1\ 0\ 1\ 0\ 0)$. Since $V(x^2) = (6\ 8\ 6\ 3)$, all of the constraints of (6.5.4) are satisfied and x^2 is an optimal solution.

Magnanti and Wong (1981) have used a variation of this approach and have developed stronger inequalities in an attempt to accelerate the convergence of the algorithm.

3.6.6 Canonical Reduction

We now present a formulation that involves the disaggregation and aggregation of clients, (see Cornuejols, Nemhauser and Wolsey, 1980). The aggregation of two clients i_1 and i_2 means to replace clients i_1 and i_2 by a single client i such that

$$c_{ij} = c_{i_1 j} + c_{i_2 j} \quad \text{all} \quad j \in J . \tag{6.6.1}$$

The disaggregation of client i into two clients i_1 and i_2 means to replace i by two clients i_1 and i_2 such that (6.6.1) holds. While aggregation is uniquely defined, disaggregation is not.

In general, aggregation yields an underestimation of profit and disaggregation overestimates. This is true, because if (6.6.1) is satisfied the greedy solution to the linear program (6.4.1) implies $V_i(x) \leq V_{i_1}(x) + V_{i_2}(x)$ for all x with $0 \leq x_j \leq 1$. We say that aggregation or disaggregation is valid when

$$V_i(x) = V_{i_1}(x) + V_{i_2}(x) \tag{6.6.2}$$

for all x with $0 \leq x_j \leq 1$.

For $k \in I$, let $s_k = (s_k(1), s_k(2), \ldots, s_k(n))$ be any permutation of $\{1,2,\ldots,n\}$ such that $c_{ks_k(1)} \geq c_{ks_k(2)} \geq \cdots \geq c_{ks_k(n)}$.

Proposition 3.8 For $i_1, i_2 \in I$, if there exists s_{i_1} and s_{i_2} such that $s_{i_1}(j) = s_{i_2}(j)$, $j = 1,\ldots,n$, then (6.6.1) is a valid aggregation of rows i_1 and i_2.

Proof: If $s_{i_1}(j) = s_{i_2}(j)$ for all j, and rows i_1 and i_2 are aggregated by (6.6.1), then $s_{i(j)} = s_{i_1}(j) = s_{i_2}(j)$, $j = 1,\ldots,n$. Thus the greedy solution to (6.4.1) implies that (6.6.2) holds. □

Corollary 3.9 If row $i \in I$ is disaggregated into rows i_1 and i_2 and (6.6.1) is satisfied and there exists s_i, s_{i_1} and s_{i_2} such that $s_{i(j)} = s_{i_1}(j) = s_{i_1}(j)$ for all j, then the disaggregation is valid.

The following proposition shows how any row with at least two unequal elements can be disaggregated.

Proposition 3.10 Suppose $c_{is_i(1)} \geq \cdots \geq c_{is_i(p-1)} > c_{is_i(p)} \geq \cdots \geq c_{is_i(n)}$. Then for $j = 1,\ldots,p-1$

$$c_{i_1 s_{i_1}(j)} = c_{is_i(p)} \quad \text{and} \quad c_{i_2 s_{i_2}(j)} = c_{is_i(j)} - c_{is_i(p)}$$

and for $j = p,\ldots,n$

$$c_{i_1 s_{i_1}(j)} = c_{is_i(j)} \quad \text{and} \quad c_{i_2 s_{i_2}(j)} = 0$$

is a valid disaggregation of row i.

Proof: The condition of Corollary 3.9 holds. □

We can apply Proposition 3.10 recursively to disaggregate a row i into at most n rows, $i_1,\ldots,i_n$ with the following properties: (i) $c_{i_t j} \in \{0, r_{i_t}\}$ for $t = 1,\ldots,n$, (ii) $r_{i_t} > 0$ for $t > 1$, (iii) $c_{i_1 j} = r_{i_1}$ for $j = 1,\ldots,n$, (iv) $c_{i_t j} = 0$ implies $c_{i_{t+1} j} = 0$ for $t = 2,\ldots,n-1$.

To do this, suppose $c_{is_i(q)} = \cdots = c_{is_i(n)}$ for some $q \leq n$.

Apply Proposition 3.10 with $p = q$. This yields

$$c_{i_1 s_{i_1}(j)} = c_{is_i(q)}, \qquad j = 1,\ldots,n$$

$$c_{i_2 s_{i_2}(j)} = \begin{cases} c_{is_i(j)} - c_{is_i(q)} & j = 1,\ldots,q-1 \\ 0 & j = q,\ldots,n. \end{cases}$$

Row i_1 is in the desired form and if $c_{is_i(j)} = c_{is_i(j+1)}$, $j = 1,\ldots,q-2$

so is row i_2. Otherwise let ℓ be the largest value of j for which $c_{is_i(\ell-1)} > c_{is_i(\ell)}$. Now apply Proposition 3.10 with $p = \ell$ to disaggregate row i_2. Here we refer to the two new rows as i_2 and i_3. Thus

$$c_{i_2 s_{i_2}(j)} = \begin{cases} c_{is_i(\ell)} - c_{is_i(q)} & j = 1,\ldots,q-1 \\ \\ 0 & j = q,\ldots,n \end{cases}$$

$$c_{i_3 s_{i_3}(j)} = \begin{cases} c_{is_i(j)} - c_{is_i(\ell)} & j = 1,\ldots,\ell-1 \\ \\ 0 & j = \ell,\ldots,n. \end{cases}$$

Row i_2 is now in the desired form and we now disaggregate row i_3 if necessary. Since at each step, the row to be disaggregated has at least one more zero than the previous row, the procedure takes at most $n - 1$ steps and yields at most n rows each having the desired property.

Consider

$$(c_{i1}\ c_{i2}\ c_{i3}\ c_{i4}) = (4\ 2\ 2\ -1).$$

We obtain

$$(c_{i_1 1}\ c_{i_1 2}\ c_{i_1 3}\ c_{i_1 4}) = (-1\ -1\ -1\ -1)$$

and

$$(c_{i_2 1}\ c_{i_2 2}\ c_{i_2 3}\ c_{i_2 4}) = (5\ 3\ 3\ 0).$$

Disaggregating row i_2 yields (3 3 3 0) and (2 0 0 0). Thus a client represented by the row (4 2 2 -1) can be replaced by 3 equivalent clients: (-1 -1 -1 -1), (3 3 3 0) and (2 0 0 0).

Suppose each of the clients $i \in I$ is disaggregated in this way. Now we may obtain pairs of clients say i_t and k_ℓ such that for $j \in J$ $c_{i_t j} \neq 0$ if and only if $c_{k_\ell j} \neq 0$. By Proposition 3.8, these two clients can be aggregated into a single client with profit $c_{i_t j} + c_{k_\ell j}$ for $j \in T$ and 0 profit for $j \notin T$. Finally, a client whose profit is constant for all $j \in J$ can be eliminated from the problem since this client produces the same profit for all feasible x.

To summarize this procedure, we can transform the matrix C into an equivalent <u>canonical</u> matrix R containing at most $\min(m(n-1), 2^n-2)$ rows. Each row of R represents a set $T \subset J$, $T \neq \phi$, and there is a profit r_T for all $j \in T$ and a profit of zero for $j \notin T$.

In our example,

$$R = \begin{pmatrix} 6 & 6 & 6 & 6 & 0 & 6 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 6 & 6 & 6 & 0 & 6 & 6 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 3 & 3 & 3 & 0 \\ 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 0 & 2 & 2 & 2 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{client 1} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{client 2} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{client 3} \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{client 4} \end{matrix} .$$

We will now use this transformation to obtain a reformulation of SLPR. Given x, $0 \leq x_j \leq 1$ all $j \in J$, the profit from the client represented by the set T is

$$V_T(x) = r_T(\min(\sum_{j \in T} x_j, 1)) = r_T - r_T(1 - \sum_{j \in T} x_j)^+.$$

If $\sum_{j \in T} x_j \geq 1$, we say that the client represented by the set T is fully served. The quantity $\pi_T = (1 - \sum_{j \in T} x_j)^+$ is the fraction of client T not served. Thus

$$V_T(x) = r_T - \min(r_T \pi_T)$$

$$\pi_T \geq 1 - \sum_{j \in T} x_j$$

$$\pi_T \geq 0.$$

Let $\mathcal{T}$ be the collection of subsets of J that are represented in the profit matrix R. Then

$$Z_{LP} = \sum_{T \in \mathcal{T}} r_T - \min(\sum_{T \in \mathcal{T}} r_T \pi_T + \sum_{j \in J} f_j x_j)$$

$$\pi_T + \sum_{j \in T} x_j \geq 1 \quad \text{all } T \in \mathcal{T} \tag{6.6.3}$$

$$\pi_T \geq 0, \; x_j \geq 0 \quad \text{all } T \in \mathcal{T}, \text{ all } j \in J.$$

The dual of (6.6.3) is

$$
\begin{aligned}
Z^*_{LP} = - \max \ & \sum_{T \in \mathcal{T}} u_T \\
& \sum_{T \ni j} u_T \leq f_j \qquad \text{all } j \in J \qquad (6.6.4) \\
& 0 \leq u_T \leq r_T \qquad \text{all } T \in \mathcal{T} .
\end{aligned}
$$

The linear program (6.6.4) has at most $m(n-1)$ variables and n constraints plus upper bounds on the variables. It is the most compact linear programming formulation we know and has the structure of the linear programming relaxation of a set packing problem. In our example, see the matrix R given above, there are only 10 variables and 6 constraints other than upper bounds. In comparison, the original linear programming formulation of SLPR ((6.1)-(6.4)) has 32 variables and 28 constraints. In experimenting with some k-median problems, Cornuejols, Nemhauser and Wolsey (1980) have observed that when the simplex method (with upper bounds treated implicitly) is applied to the various linear programming formulations, the formulation (6.6.4) was by far the best one in terms of simplex pivots and running time. In addition, the formulation (6.6.4) provides a nice interpretation for the dual descent heuristic given in Section 3.5. We leave this as an exercise.

3.6.7 Summary

Two types of decompositions have been considered in this section. Lagrangian duality and Dantzig-Wolfe decomposition assign prices to the clients and for a fixed set of prices, the problem decomposes into independent, single problems for the facilities. The two methods differ only

in their schemes for adjusting prices. Benders decomposition and the formulation of Section 3.6.4 choose a facilities vextor x and for a given x, the problem decomposes into independent, simple problems for the clients. The two methods differ only in their schemes for adjusting x.

Any method for solving the UFL Problem that uses exact or approximate solutions of SLPR as a subroutine, must be imbedded into a branch-and-bound algorithm. One approach is to solve SLPR by a special purpose algorithm. Another approach is to use a general purpose mixed-integer programming (MIP) system.

Among the special purpose algorithms, DUALOC, which uses approximate dual solutions to SLPR, is a strong candidate. It is generally available as a FORTRAN program, is very fast on most problems and is not limited by size unless quite a lot of enumeration is necessary.

Difficult instances of the UFL problem may require the capability of finding optimal solutions of SLPR in order to curtail the enumerative phase of the algorithm. Here solving the Lagrangian dual by subgradient optimization provides an easily programmable and a relatively fast algorithm. Another method worth considering is the linear programming formulation (6.6.4), which has a significant advantage in size over the other linear programming formulations and can be solved directly by the simplex method for instances that are not too large. Schrage's adaptation of the simplex method to handle generalized upper bounds is still another possibility.

If we choose to use a MIP system, the original formulation (1.1)-(1.4), the Benders formulation with $x_j \in \{0,1\}$ for all $j \in J$ and the canonical formulation (6.6.3) with $x_j \in \{0,1\}$ are candidates. However, each of these formulations is limited by problem size since they involve $O(mn)$ constraints

and/or variables. The formulation (6.6.4), which is the dual of (6.6.3) is the most compact linear programming formulation, but to use it in a conventional MIP system would require a modification that permits branching on fractional dual variables.

The main advantage of using a general MIP system is that additional constraints create no difficulties, as they may for special purpose codes. There is work in progress on MIP systems that will be capable of working with the weak linear programming relaxation and will generate violated variable upper bound constraints as needed (see Martin and Schrage, 1982, and Van Roy and Wolsey, 1983.) Codes of this type should make it feasible to solve medium-sized instances of the UFL Problem as general mixed integer programs.

3.7. Polyhedral Results

In this section, we study the polytope of feasible solution to the SLPR constraints

$$\begin{cases} \sum_{j=1}^{n} y_{ij} = 1 & i = 1,\ldots,m \\ 0 \leq y_{ij} \leq x_j \leq 1 & i = 1,\ldots,m \text{ and } j = 1,\ldots,n, \end{cases} \tag{7.1}$$

and the polytope defined by the convex hull of integral solutions to (7.1).

3.7.1 The SLPR Polytope

Let $Q_{m,n}$ be the polytope of feasible solutions to (7.1) When $m \leq 2$ or $n \leq 2$, it has been shown by Muckendi (1975), Krarup and Pruzan (1983), and Cho, Johnson, Padberg and Rao (1983) that all the extreme points of (7.1) are integral. In fact, the constraint matrix is totally unimodular in that case. However, for values as small as $m = n = 3$, $Q_{m,n}$ has fractional extreme points. For example, when $C = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $f_j = 1$ for $j = 1,2,3$, we have remarked previously that $x_j = 1/2$ for $j = 1,2,3$ and $y_{ij} = 1/2$ for $i \neq j$, $y_{ij} = 0$ for $i = j$ is the unique optimal solution.

The fractional extreme points of $Q_{m,n}$ are completely characterized by the next theorem. For a given fractional solution (x,y) of (7.1) let

$J_1 = \{j \in J: 0 < x_j < 1\}$ and

$I_1 = \{i \in I: y_{ij} = 0$ or x_j for all j and y_{ij} is fractional for at least one $j\}$.

Let $a_{ij} = 1$ if $y_{ij} > 0$ and 0 otherwise, and denote by A the $|I_1| \times |J_1|$ matrix whose elements are a_{ij} for $i \in I_1$, $j \in J_1$.

Theorem 3.11 (Cornuejols, Fisher and Nemhauser, 1977b). A fractional solution (x,y) of (7.1) is an extreme point of $Q_{m,n}$ if and only if

(i) $x_j = \max_{i \in I} y_{ij}$ for all $j \in J_1$

(ii) for each $i \in I$, there is at most one $j \in J$ with $0 < y_{ij} < x_j$

(iii) the rank of A equals $|J_1|$.

The three conditions of this theorem are easily verified for the example given above.

Since $Q_{m,n}$ has many fractional extreme points, the type of objective function that is optimized over this polyhedron must play an important role in the attainment of integral optimal solutions. Frequently, $C = \{c_{ij}\}$ is defined over a network with the property that the further node v_i is from node v_j in the network the smaller is c_{ij}. (E.g., if v_i is on the shortest path from v_i to v_j in the network then $c_{ij} \leq c_{i'j}$). This property is important in showing that SLPR always has an integral optimal solution for tree networks (see Section 3.8). A similar result is known for the p-median problem defined on a path (Wong, Ward, Oudjit and Lemke, 1984). However, for more general networks, appropriate conditions on C for attaining an integral optimal solution are not known.

In some cases SLPR often has fractional optimal solutions. An example is the linear programming relaxation of the set covering problem. The set covering problem is the special case of the UFL Problem where $C = \{c_{ij}\}$ is a general 0,1 matrix and $f_j = 1$ all $j \in J$.

Some valid inequalities for the UFL Problem that are violated by fractional extreme points of $Q_{m,n}$ are given in the next theorem.

Theorem 3.12 (Cho et al., 1983). Let B be a $k \times k$ nonsingular 0,1 matrix such that $B^{-1} e \geq 0$, where e is a column vector of ones. Index the rows and columns of B by $I_k \subseteq I$ and $J_k \subseteq J$ where $|I_k| = |J_k| = k$. Then

$$\sum_{i \in I_k} \sum_{j \in J_k} b_{ij} y_{ij} - \sum_{j \in J_k} x_j \leq \lfloor k - e^T B^{-1} e \rfloor$$

is a valid inequality for the UFL Problem. It cuts off at least one fractional extreme point of $Q_{m,n}$ if $e^T B^{-1} e$ is not integral.

For example, if $B = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ we generate the constraint $y_{12} + y_{13} + y_{21} + y_{23} + y_{31} + y_{32} - x_1 - x_2 - x_3 \leq 1$, which cuts off the fractional extreme point given in the beginning of this section.

Moreover, it is easy to show that the family of valid inequalities defined in Theorem 3.12 cuts off all the fractional extreme points of $Q_{m,n}$. However, in general new fractional extreme points arise.

3.7.2 The UFL Polytope

Let $P_{m,n}$ be the polytope defined as the convex hull of the integer solutions to the system (7.1). Thus the extreme points of $P_{m,n}$ are the feasible solutions to the UFL Problem. Here we consider the identification of valid inequalities for $P_{m,n}$ that define facets. To explain these results, we need some definitions from linear algebra and polyhedral theory.

A set of $k + 1$ points $w^0, w^1, \ldots, w^k$ are affinely independent if the k vectors $w^1 - w^0, \ldots, w^k - w^0$ are linearly independent. A polytope has dimension k if it contains $k + 1$ affinely independent points but not more. An affine space is the intersection of hyperplanes. The smallest

affine space which contains a polytope P is called its affine hull. The polytope $P_{m,n}$ has dimension $mn + n - m$ and affine hull given by $\{(x,y) \in R^n \times R^{mn} : \sum_{j=1}^{n} y_{ij} = 1 \text{ for } i = 1,\ldots,m\}$.

A face of a polytope P is a set $F = P \cap \{x: ax = b\}$ where $ax \leq b$ is satisfied by every $x \in P$ and $ax < b$ for at least one $x \in P$. The inequality $ax \leq b$ is said to define the face F. Any face of a polytope is itself a polytope. When its dimension is one less than that of the polytope P, the face F is called a facet. To describe the polytope P by a linear system, it suffices to have a description of its affine hull and one defining inequality for each facet of P.

If such a description of $P_{m,n}$ were known then, in principle, the UFL Problem could be solved as a linear program. Cho et al (1983) give a complete description of $P_{m,n}$ when $n \leq 3$ or $m \leq 3$. However, when $n \geq 4$ and $m \geq 4$, a complete linear system defining $P_{m,n}$ is not known explicitly. Even if one were, only relevant portions of it need be used to solve an instance of the UFL Problem. That is, some of the facets of $P_{m,n}$ can be used as cutting planes in the spirit of Padberg and Hong's (1980) and Grotschel's (1980) work on the traveling salesman problem.

Whatever the algorithmic use of a partial linear description of $P_{m,n}$, the first step is to identify some of its facets.

Theorem 3.13 The following inequalities define distinct facets of $P_{m,n}$

(i) $y_{ij} \leq x_j$ for all $i \in I$, $j \in J$

(ii) $y_{ij} \geq 0$ for all $i \in I$, $j \in J$

(iii) $x_j \leq 1$ for all $j \in J$.

These facets are called the elementary facets of $P_{m,n}$. The next theorem provides a necessary and sufficient condition for an inequality with coefficients of 0 or 1 to define a facet of $P_{m,n}$.

Assume that $I' \subseteq I$ and $J' \subseteq J$ are two nonempty sets and that $B = (b_{ij})$ $i \in I'$, $j \in J'$ is a 0,1 matrix with no zero row. Consider the inequality

$$\sum_{i \in I'} \sum_{j \in J'} b_{ij} y_{ij} - \sum_{j \in J'} x_j \leq r. \qquad (7.2.1)$$

Define the graph G as follows. It has a node associated with each variable y_{ij}, $i \in I$, $j \in J$, and x_j, $j \in J$. We will use the same notation for a node and its associated variable. For all $i \in I$ and $j \in J$, the node y_{ij} is joined by an arc to the node x_j and to every node y_{ik} for $k \neq j$.

Let N' be the set of nodes $\{\{y_{ij}: i \in I', j \in J'\} \cup \{x_j: j \in J'\}\}$ and let G' be the subgraph of G induced by the node set N'. Given a graph H we denote by $\alpha(H)$ the maximum size of a stable set in H (a stable set is a set of mutually nonadjacent nodes). Finally, an arc e of H is critical if $\alpha(H - e) > \alpha(H)$, where $H - e$ denotes the graph obtained from H by removing the arc e.

Theorem 3.14 (Cornuejols and Thizy, 1982a) The inequality (7.2.1) is a facet of $P_{m,n}$ if and only if the following set of conditions is satisfied

(i) $r = \alpha(G') - |J|$

(ii) G' is connected,

(iii) for every $i \in I'$, $j \in J'$ such that $b_{ij} = 1$, the arc (x_j, y_{ij}) is critical,

(iv) for every $j, k \in J'$, there exists a sequence of critical arcs

$$(y_{i_1 j}, y_{i_1 \ell_1}), (y_{i_2 \ell_1}, y_{i_2 \ell_2}), \ldots, (y_{i_{s-1} \ell_{s-2}}, y_{i_{s-1} \ell_{s-1}}), (y_{i_s \ell_{s-1}}, y_{i_s k}).$$

(v) for every $i \in I$, $j \in J$ such that $y_{ij} \notin N'$, the inequality $\alpha(G') < \alpha(G'')$ is strict, where G'' denotes the subgraph of G induced by $N' \cup \{y_{ij}\}$.

These necessary and sufficient conditions can be used to prove the next theorem, which provides constructively a large class of facets for the UFL Problem.

For $2 \leq t < \ell \leq n$, define $B^{\ell t}_{ij} = (b^{\ell t}_{ij})$ as a matrix with $\binom{\ell}{t}$ rows and ℓ columns whose rows consist of all distinct 0,1 vectors with t ones and $\ell - t$ zeros.

Theorem 3.15 (Cornuejols and Thizy, 1982a) For any pair of integers ℓ and t such that $2 \leq t < \ell \leq n$ and $\binom{\ell}{t} \leq m$, and any sets $I' \subseteq I$, $J' \subseteq J$ such that $|I'| = \binom{\ell}{t}$ and $|J'| = \ell$, the inequality

$$\sum_{i \in I'} \sum_{j \in J'} b^{\ell t}_{ij} y_{ij} - \sum_{j \in J'} x_j \leq \binom{\ell}{t} + t - \ell - 1$$

defines a facet of $P_{m,n}$.

For example, take $t = 2$, $\ell = 3$, $B^{23} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, and $I' = J' = \{1,2,3\}$. According to Theorem 3.15 we get the facet

$$y_{12} + y_{13} + y_{21} + y_{23} + y_{31} + y_{32} - x_1 - x_2 - x_3 \leq 1,$$

which is identical to the valid inequality of Theorem 3.12 that we obtained above with $B = B^{23}$.

Additional material on facets of the UFL polytope can be found in Guignard (1980), Cornuejols and Thizy (1982a), and Cho, Padberg and Rao (1983).

3.8. Polynomially Solvable Cases

Formally, by a special case of the UFL Problem, we mean a problem of the form (1.1)-(1.4) all of whose instances are described by a subfamily of objective functions (C,f). In this section, we consider two special cases that have the following significant properties.

1. SLPR always has an integral optimal solution
2. The problem can be solved in polynomial-time by a combinatorial algorithm.

The Economic Lot Size Problem

There is a demand d_i in period i, $i = 1,\ldots,n$. The fixed cost of producing in period j is $f_j \geq 0$. The variable production cost is p_j. The variable storage and backorder costs are $c_j^+ \geq 0$ and $c_j^- \geq 0$ respectively. Let y_{ij} represent the fraction of the demand of period i produced in period j, and $x_j = 1$ if and only if there is production in period j. Then the UFL formulation can be used to minimize production cost where

$$c_{ij} = -(p_j + c_j^+ + \ldots + c_{i-1}^+)d_i \quad \text{if} \quad i \geq j$$

and

$$c_{ij} = -(p_j + c_j^- + \ldots + c_{i+1}^-)d_i \quad \text{if} \quad i < j.$$

The Tree Location Problem

Let $G = (V,E)$ be a graph with node set V and arc set E and suppose that G is a tree, (that is, there is a unique path in G between each pair of nodes). Here the nodes represent both clients and facilities. The cost of opening the jth facility is $f_j \geq 0$ all $v_j \in V$. Associated with each arc $e \in E$, there is a given non-negative distance. The distance d_{ij} between any

pair of nodes v_i and v_j is the sum of the edge distances along the unique path between v_i and v_j for all $v_i, v_j \in V$. There is also a non-negative weight w_i associated with each node v_i. Let $c_{ij} = -w_i d_{ij}$, $v_i, v_j \in V$, $i \neq j$ and $c_{ii} = 0$ all $v_i \in V$.

The induced subgraph of G generated by $V_j \subseteq V$ is the graph $G_j = (V_j, E_j)$ where $E_j = \{e \in E$: both end nodes of e are in $V_j\}$. G_j is said to be a subtree of G if G_j itself is a tree.

Theorem 3.16 (Kolen, 1982) There is an optimal solution to the tree location problem in which the set of open facilities $S \subseteq V$ is such that for each $v_j \in S$, $V_j = \{v_i \in V$: node v_i is served by node $v_j\}$ induces a subtree. Moreover, this solution is also optimal to the linear programming relaxation of the tree network problem.

A similar result applies to the lot sizing problem. Consider the tree $G = (V,E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_{i+1})$: $i = 1, \ldots, n-1\}$. Here G is simply a path from node v_1 to node v_n so that $V' \subset V$ is a subtree or a path if and only if $V' = \{v_i, v_{i+1}, \ldots, v_k\}$ for some i and k, $1 \leq i \leq n$ and $k \geq i$.

Theorem 3.17 (Krarup and Bilde, 1977) There is an optimal solution to the lot sizing problem in which the set of periods having positive production $S \subseteq V$ is such that for each $j \in S$, $V_j = \{v_i \in V$: period i is served by production in period $j\}$ induces a path. Moreover, this solution is also optimal to the linear programming relaxation of the tree network problem.

The fact that an optimal solution to these problems induces subtrees that partition V is not surprising. In the tree location problem, suppose that

node v_i is on the path joining nodes v_j and v_k and node v_j serves node v_k but not node v_i. Hence v_j does not induce a subtree. Suppose node v_i is served by node $v_{j'} \neq v_j$. Then the from this part of the solution is $d_{jk} + f_j + d_{j'1} + f_{j'}$. If instead, node v_i is served by node v_j, the cost is $d_{ji} + d_{jk} + f_j + f_{j'}$.

Thus if $d_{ji} \leq d_{j'i}$, then v_j can serve v_i and all nodes after v_i that are being served by $v_{j'}$ without increasing the cost. Otherwise $d_{ji} > d_{j'i}$, which implies

$$d_{jk} = d_{ji} + d_{ik} > d_{j'i} + d_{ik} = d_{j'k}.$$

This inequality implies that the solution in which $v_{j'}$ serves v_k and only nodes after v_k that are currently being served by v_j costs less than the original. A recursive application of this argument proves the first statement of Theorem 3.16.

A similar argument proves this result for the economic lot sizing problem.

Both of these results suggest that the ability to partition the solution into subtrees is crucial and leads us to consider the following generalization.

The Tree Partitioning Problem

Given a tree graph $G = (V,E)$ and a node by node matrix with elements γ_{ij} all $v_i, v_j \in V$, let the weight of a subtree $G_j = (V_j,E_j)$ be $w(G_j) = \max_{v_k \in V_j} \left(\sum_{v_i \in V_j} \gamma_{ik} \right)$. Find a partition of G into subtrees such that the sum of the weights over all subtrees in the solution is maximum.

To model the lot sizing problem as a tree partitioning problem we take

$$\gamma_{jj} = -(f_j + p_j d_j)$$

$$\gamma_{ij} = -(p_j + c_j^+ + \ldots + c_{i-1}^+)d_i \quad \text{if} \quad i > j$$

$$\gamma_{ij} = -(p_j + c_j^- + \ldots + c_{i+1}^-)d_i \quad \text{if} \quad i < j.$$

To model the tree location problem as a tree partitioning problem we take $\gamma_{jj} = -f_j$ and $\gamma_{ij} = -w_i d_{ij}$ if $i \neq j$.

We now formulate the tree partitioning problem as an integer program in a manner that establishes its connection to the UFL Problem. If $v_{j*} \in V_j$ is such that $\arg\max_{v_k \in V_j} (\sum_{v_i \in V_j} \gamma_{ik}) = v_{j*}$, we say that v_{j*} is the root of subtree G_j. Let $y_{ij} = 1$ if $v_i \in V$ is in a subtree rooted at v_j and $y_{ij} = 0$ otherwise. Then the tree partitioning problem can be formulated as

$$\max \sum_i \sum_j \gamma_{ij} y_{ij} \tag{8.1}$$

$$\sum_j y_{ij} = 1 \qquad \text{all } v_i \in V \tag{8.2}$$

$$y_{ij} - y_{i'j} \leq 0 \qquad \text{all } v_i, v_{i'}, v_j \in V \text{ such that } v_{i'} \text{ precedes } v_i \text{ on a path from } v_j \text{ to } v_i \tag{8.3}$$

$$y_{ij} \in \{0,1\} \qquad \text{all } v_i, v_j \in V. \tag{8.4}$$

Constraint (8.3) guarantees that if v_j is the root of a tree that contains v_i then the tree must also contain $v_{i'}$. Constraint (8.2) guarantees that each node must be in exactly one tree.

The linear programming relaxation of this integer program is obtained by replacing (8.4) by

$$y_{ij} \geq 0 \quad \text{all} \quad v_i, v_j \in V. \tag{8.5}$$

Theorem 3.18 (Barany, Edmonds and Wolsey, 1984) The polyhedron defined by (8.2), (8.3) (8.5) has only integral extreme points. Hence for any objective function (8.1), the solution to the linear programming relaxation is integral.

This model for the tree partitioning problem resembles the model (1.1)-(1.4) for the UFL Problem if we think of the y_{jj}'s as x_j's and replace (8.3) by (1.3). In fact, we can represent solutions of (1.1)-(1.4) as a collection of subtrees that partition a graph G, but unfortunately G is not necessarily a tree. In the graph of Figure 3.2, V_1 represents the set of clients and V_2 the set of facilities.

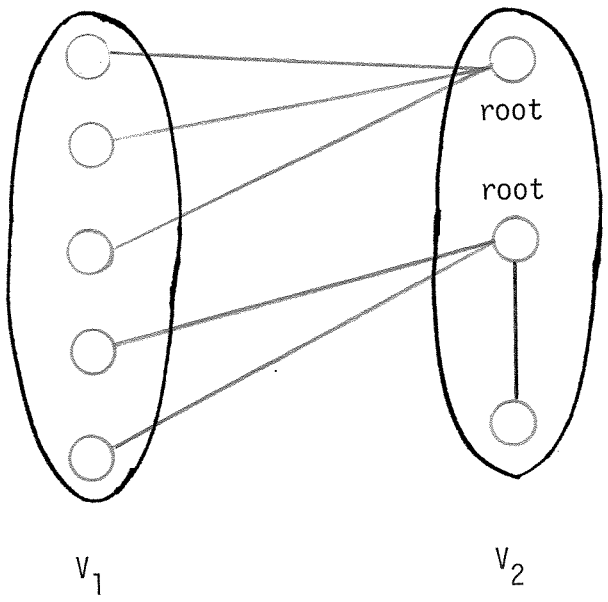


Figure 3.2

A solution to the problem is a set of subtrees, each of which is rooted at a node in V_2. The arcs from the root of a tree to nodes in V_1 show the clients being served by the facility that corresponds to the root. A node in V_2 that is not a root corresponds to an unused facility. We set $\gamma_{ij} = c_{ij}$ for $v_i \in V_1$ and $v_j \in V_2$. To eliminate the possibility of nodes in V_1 being roots we set $\gamma_{jj} = -M$ for $v_j \in V_1$ (M is a large positive number) and to represent the fixed costs we set $\gamma_{jj} = -f_j$ for $v_j \in V_2$. Finally to accommodate unused facilities we set $\gamma_{jk} = 0$ if v_j and v_k are in V_2.

We close this section by giving an $O(|V^2|)$ dynamic programming algorithm for solving the tree partitioning problem. Another very general dynamic programming algorithm for location problems on trees can be found in Megiddo, Zemel and Hakimi (1983).

Given the tree $G = (V,E)$ we choose an arbitrary root node r. This induces a partial order on V. For all $v \in V$, let $V(v) = \{u: v \text{ is on the unique path between } r \text{ and } u\}$ and $S(v) = \{u: u \in V(v) \text{ and } (v,u) \in E\}$. Let T_v be the tree induced by $V(v)$ and $g(v)$ be the maximum weight of a partition for the tree T_v.

The idea of the algorithm is to calculate $g(r)$ recursively by determining $g(v)$ from $\{g(u)\}$ for all $u \in S(v)$. To develop the general recursion equations, we need another function. Let $g_u(v)$ be the maximum weight of a partition of T_v when v is served by node u. If $u \notin V(v)$, then $g_u(v)$ includes the terms γ_{uw} for all $w \in V(v)$ that are served by u.

Then

$$g(v) = \max_{u \in V(v)} g_u(v). \tag{8.6}$$

Now suppose we are given $g_u(w)$ for all $w \in S(v)$ and all $u \in V$. The calculation of $g_u(v)$ divides into two cases as shown in Figure 3.3

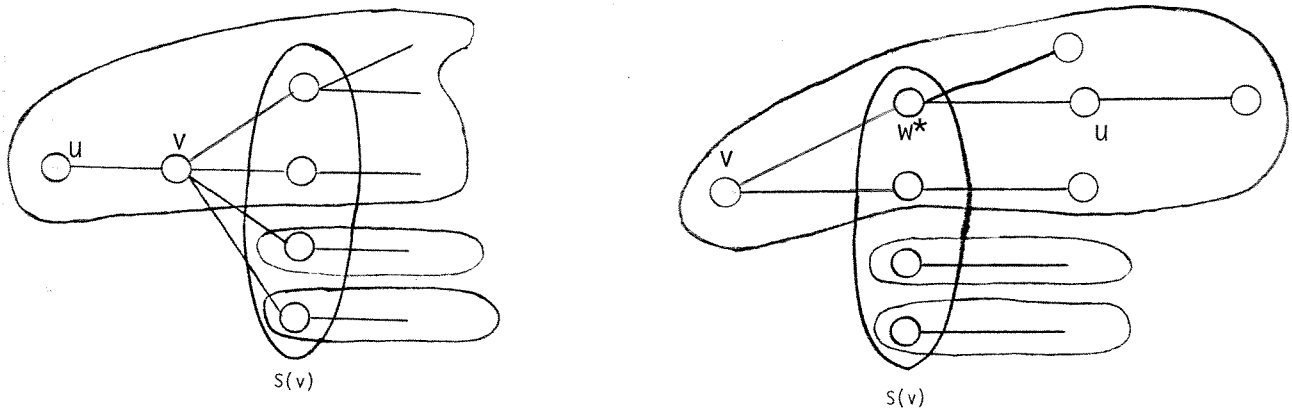


Figure 3.3

If $u \notin V(v)$ or $u = v$, and u serves v, then $w \in S(u)$ is either served by u or some node in $V(w)$ because of the tree structure. Thus w is served by u when v is served by u if and only if $g_u(w) > g(w)$. Hence

$$g_u(v) = \gamma_{vu} + \sum_{w \in S(v)} \max\{g_u(w), g(w)\}. \tag{8.7}$$

If $u \in V(v)\backslash\{v\}$ serves v, then the tree structure implies that the node $w^* \in S(v)$ on the path joining v and u must also be served by u. Hence

$$g_u(v) = \gamma_{vu} + \sum_{w \in S(v)\backslash\{w^*\}} \max\{g_u(w), g(w)\} + g_u(w^*). \tag{8.8}$$

A node $v \in V$ is said to be a <u>leaf</u> of T_r if $S(v) = \emptyset$. The recursion is begun with the leaves. For all leaves $v \in V$, we have

$$g(v) = g_v(v) = \gamma_{vv} \quad \text{and} \quad g_u(v) = \gamma_{vu} \quad \text{for} \quad u \neq v. \tag{8.9}$$

<u>An Example</u>

We consider a tree location problem on the graph of Figure 3.4. The numbers on the arcs are the

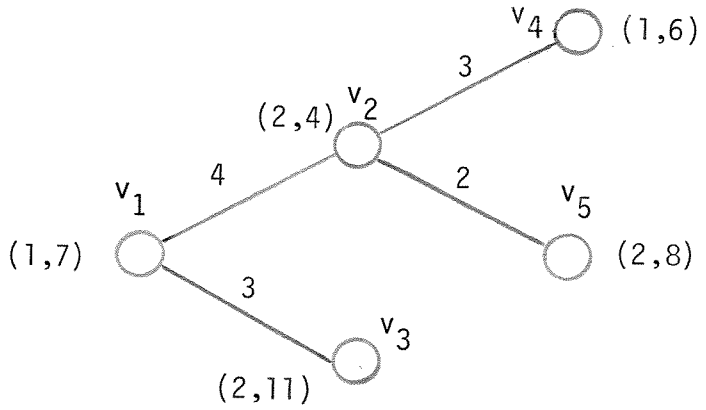


Figure 8.3

d_{ij}'s and the pair of numbers adjacent to the nodes are (w_i,f_i) all $v_i \in V$. Let $\gamma_{jj} = -f_j$ all $v_j \in V$ and $\gamma_{ij} = -w_i d_{ij}$ all $v_i \neq v_j \in V$. Hence we obtain the matrix

$$\gamma = \begin{pmatrix} -7 & -4 & -3 & -7 & -6 \\ -8 & -4 & -14 & -6 & -4 \\ -6 & -14 & -11 & -20 & -18 \\ -7 & -3 & -10 & -6 & -5 \\ -12 & -4 & -18 & -10 & -8 \end{pmatrix}.$$

For the leaves v_3, v_4 and v_5, the bottom 3 rows of the matrix γ give g_u, see (8.9).

For node v_2, (8.7) and (8.8) yield

$$g_1(2) = \gamma_{21} + \max\{g_1(4),g(4)\} + \max\{g_1(5),g(5)\}$$
$$-8 + \max(-7,-6) + \max(-12,-8) = -22$$

$$g_2(2) = -4 + \max(-3,-6) + \max(-4,-8) = -11$$

$$g_3(2) = -14 + \max(-10,-6) + \max(-18,-8) = -28$$

$$g_4(2) = \gamma_{24} + \max\{g_4(5),g(5)\} + g(4)$$
$$= -6 + \max\{-10,-8\} + (-6) = -20$$

$$g_5(2) = -4 + \max\{-5,-6\} + (-8) = -17.$$

Hence

$$g(2) = \max\{g_2(2),g_4(2),g_5(2)\} = \max\{-11,-20,-17\} = g_2(2) = -11.$$

For node v_1 we obtain

$$g_1(1) = -7 + \max\{-22,-11\} + \max\{-6,-11\} = -24$$

$$g_2(1) = -4 + \max\{-14,-11\} + (-11) = -26$$

$$g_3(1) = -3 + \max\{-28,-11\} + (-11) = -25$$

$$g_4(1) = -7 + \max\{-20,-11\} + (-20) = -38$$

$$g_5(1) = -6 + \max\{-18,-11\} + (-17) = -34 .$$

Hence $g(1) = g_1(1) = -24$, where $g_1(1) = \gamma_{11} + g(2) + g_1(3)$.

Thus node v_1 serves itself and node v_3. Since $g(2) = g_2(2) = \gamma_{22} + g_2(4) + g_2(5)$, node v_2 serves itself and nodes v_4 and v_5.

3.9. Submodularity

As defined at the outset of this chapter, the UFL is the combinatorial optimization problem $\max_{S \subseteq J} z(S)$ where

$$z(S) = \sum_{i=1}^{m} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j \qquad (9.1)$$

is the profit made when the set S of facilities is open. A very important property of the set function z is its submodularity, A function w defined on the subsets of a finite set J is submodular if

$$w(S \cup \{k\}) - w(S) \leq w(R \cup \{k\}) - w(R) \quad \text{for all} \quad k \notin S \quad \text{and} \quad R \subseteq S \subseteq J - \{k\}.$$

The fact that the profit function z is submodular was observed by Spielberg (1969a), Babayev (1974), Frieze (1974) and Fisher, Nemhauser and Wolsey (1978). It means that the additional profit that can be made by opening a facility in location k when a set S is already open in other locations is a nonincreasing function of S with respect to set inclusion. The larger S, the smaller the profit of establishing a new facility. This is proved formally in the next theorem.

Theorem 3.19 The profit function z given by (9.1) is submodular.

Proof: Let $R \subseteq S \subseteq J - \{k\}$. For all $i = 1,\ldots,m$

$$\max_{j\in S\cup\{k\}} c_{ij} - \max_{j\in S} c_{ij} = \max(0,\ c_{ik} - \max_{j\in S} c_{ij})$$

$$\leq \max(0,\ c_{ik} - \max_{j\in R} c_{ij}) = \max_{j\in R\cup\{k\}} c_{ij} - \max_{j\in R} c_{ij}$$

where the inequality follows from $\max_{j\in S} c_{ij} \geq \max_{j\in R} c_{ij}$.

By summing these inequalities for all i, we obtain

$$\sum_{i=1}^{m} \max_{j\in S\cup\{k\}} c_{ij} - \sum_{i=1}^{m} \max_{j\in S} c_{ij} \leq \sum_{i=1}^{m} \max_{j\in R\cup\{k\}} c_{ij} - \sum_{i=1}^{m} \max_{j\in R} c_{ij}.$$

Hence

$$z(S\cup\{k\}) - z(S) \leq z(R\cup\{k\}) - z(R). \quad \square$$

Thus the UFL Problem is a special case of the problem

$$\max_{S\subseteq J} \{z(S):\ z \text{ submodular}\}. \tag{9.2}$$

We can apply the greedy and interchange heuristics to (9.2), we can formulate (9.2) as an integer program and many of the results that we have given for the UFL Problem extend to (9.2) and, in particular, to the capacitated location problem, which is another special case of (9.2). We will not elaborate on these results here, but refer the interested reader to Fisher, Nemhauser and Wolsey (1978), Nemhauser and Wolsey (1978), Nemhauser, Wolsey and Fisher (1978), Cornuejols, Nemhauser and Wolsey (1980), Nemhauser and Wolsey (1981) and Conforti and Cornuejols (1984).

3.10. Probabilistic Results

In Section 3.4 the point was made that good feasible solutions as well as good upper bounds are crucial in solving the UFL Problem. Accordingly, the algorithmic tools presented in this chapter focused on heuristics (Section 3.5) and on the solution of SLPR (Section 3.6). The computational experience accumulated over the years on this problem has shown that several algorithms produce very good lower and upper bounds. Can these observations be explained formally? The first formal probabilistic analysis of a location algorithm was performed by Fisher and Hochbaum (1980). Their main result is for the p-median problem but it is also useful for the UFL Problem.

This type of analysis requires that we assume an underlying probability distribution of problem instances. Since these assumptions can be somewhat arbitrary, different models can emerge, each giving a different insight. To illustrate the approach, we will consider one specific model, the so-called Euclidean model.

Here n points, $x_1,\ldots,x_n$, are chosen independently and uniformly at random in the unit square $[0,1]^2$. Denote by $||x_i - x_j||$ the Euclidean distance between points x_i and x_j, and let f be a fixed cost. The Euclidean UFL Problem is to choose $S \subseteq \{x_1,\ldots,x_n\}$, $S \neq \emptyset$, such that $\sum_{x_i \notin S} \min_{x_j \in S} ||x_i - x_j|| + f|S|$ is minimum.

The results of Fisher and Hochbaum were improved by Papadimitriou (1981). Papadimitriou's paper is about the p-median problem but his results can also be applied to the Euclidean UFL Problem defined above. We will state them for this problem. Assume that

$$\text{for some } \epsilon > 0,\ n^{-1/2+\epsilon} \leq f \leq n^{1-\epsilon}. \qquad (10.1)$$

Theorem 3.20 (Papadimitriou, 1981) Under assumption (10.1), the optimum value of the Euclidean UFL Problem is

$$Z \sim (.986612\ldots)f^{1/3}n^{2/3} \quad \text{almost surely.}$$

Equivalently, this theorem states that, for any $\alpha > 0$,

$$\Pr\left[1 - \alpha \leq \frac{Z}{(.986612\ldots)f^{1/3}n^{2/3}} \leq 1 + \alpha\right] \to 1 \quad \text{as} \quad n \to \infty.$$

Papadimitriou also proposes the following heuristic.

(a) Let $K = \left[(.328871\ldots)f^{-2/3}n^{2/3}\right]$

(b) Tile the plane with hexagons $H_1,\ldots,H_K$ each of area $1/K$. Choose those hexagons H_j for which $H_j \leq [0,1]^2$. Let the set of their centers be $H = \{h_1,\ldots,h_p\}$, $p \leq K$.

(c) Define the set $S = \{s_1,\ldots,s_p\}$ of open facilities as follows. For $j = 1,\ldots,p$, let $s_j \in \{x_1,\ldots,x_n\}$ be a point such that $\left|\left|s_j - h_j\right|\right| \leq \left|\left|x_i - h_j\right|\right|$ for all $i = 1,\ldots,n$.

(d) $Z^H = \sum_{x_i \notin S} \min_{x_j \in S} \left|\left|x_i - x_j\right|\right| + f|S|$.

Theorem 3.21 (Papadimitriou, 1981) Under assumption (10.1), the value Z^H of the solution produced by the heuristic for the Euclidean UFL Problem is

$$Z^H \sim (.986612\ldots)f^{1/3}n^{2/3} \quad \text{almost surely.}$$

In other words, Theorems 3.20 and 3.21 assert that the error $(Z^H - Z)$ made by using the heuristic is of an order smaller than the optimal value Z, that is

$$\frac{Z^H - Z}{Z} \to 0 \quad \text{as} \quad n \to \infty.$$

The bound W provided by the SLPR can be analyzed in a similar fashion.

Theorem 3.22 (Ahn, Cornuejols and Frieze, 1984) Under assumption (10.1), the value of the SLPR for the Euclidean UFL Problem is

$$W \sim (.984745...)f^{1/3}n^{2/3} \quad \text{almost surely.}$$

Corollary 3.23 $\frac{Z - W}{Z} \sim .00189...$ almost surely.

The probabilistic results obtained for this model have some interesting consequences. Any branch and bound algorithm where the branching is done on the variables x_j and the bounding is based on the SLPR, will almost surely enumerate an exponential number of subproblems before it can produce an optimum solution. That is if we insist on having an optimum solution. On the other hand, a very good solution and a proof that the solution is within two tenth of one percent of the optimum can almost surely be found very quickly. (Details can be found in Ahn, Cornuejols and Frieze, 1984).

3.11. Exercises

1. Sometimes a p-median problem can be solved by removing the condition $\sum_{j \in J} x_j = p$, introducing an artificial fixed charge of $f_j = f$ for each $j \in J$, and solving the resulting UFL Problem for different value of f until a value f^* is found for which an optimal solution of the UFL Problem has p open facilities.

(a) Show that this method does not always work by giving an instance of a 2-median problem for which no value of f provides an optimal solution with 2 open facilities in the associated UFL Problem.

(b) For each $n \geq 3$, find an integer m and an $m \times n$ matrix C such that the instance of the UFL Problem defined by C and $f_j = f$ for $j = 1,\ldots,n$, has only two possible optimal solutions when f varies: when $f < f^*$, there is a unique optimal solution and it has one open facility; when $f > f^*$ there is a unique optimal solution and it has $n - 1$ open facilities; when $f = f^*$, both of these solutions are optimum.

2. The set covering problem is defined as follows and is known to be NP-hard. Given a finite set E and a family of subsets $A_i \subseteq E$, $i = 1,\ldots,q$, find a subfamily $\{A_i\}_{i \in H}$, where $H \subseteq \{1,\ldots,q\}$ such that $\bigcup_{i \in H} A_i = E$ and $|H|$ is minimum.
Show that the UFL Problem is NP-hard by a polynomial transformation of the set covering problem.

3. The strong linear programming relaxation of the p-facility location problem is given by (1.1)-(1.3), (1.5), (1.6).

(a) Write its dual.

(b) Write a condensed dual similar to (4.6).

(c) Write a condensed dual similar to (4.7)-(4.9).

(d) Write a Lagrangian dual similar to (4.12).

(e) Can you prove a proposition similar to Proposition 3.2 for the p-facility location problem?

4. Consider an instance of the UFL Problem where the optimum value W of SLPR equals the optimum value Z of UFL. Let u be a dual solution satisfying (4.8), (4.9). Define $J(u) = \{j: \sum_{i \in I} (c_{ij} - u_i)^+ - f_j = 0\}$ and for, any set $K(u) \subseteq J(u)$, let $k_i = \left|\{j \in K(u) = c_{ij} > u_i\}\right|$. Show that a necessary condition for u to be an optimal dual solution is that there exists a set $K(u) \subseteq J(u)$ such that $k_i \leq 1$ for all $i \in I$.

5. Consider the greedy algorithm applied to the p-facility location problem. Show that, if $\rho_j(S^t) = \rho_j(\emptyset)$ for $t = 1,\ldots,p-1$ and all $j \in J - S^t$, then the greedy algorithm is optimum.

6. Let $j_1,\ldots,j_p$ be a solution obtained by applying the greedy algorithm to the p-facility location problem, and let A be an optimum solution. Denote by $S^t = \{j_1,\ldots,j_t\}$ the partial greedy solution obtained at iteration t.
Show that, if $\rho_{j_t}(S^{t-1}) = \rho_{j_t}(S^{t-1} \cup A)$ for all $t = 1,\ldots,p$, then the greedy algorithm is optimum.

7. Consider the instance of the UFL Problem defined by $m = 5$, $n = 8$, $f_j = 2$ for $j = 1,\ldots,8$ and

$$C = \begin{pmatrix} 3 & 6 & 3 & 5 & 6 & 4 & 3 & 0 \\ 4 & 4 & 5 & 3 & 5 & 2 & 0 & 5 \\ 4 & 3 & 4 & 3 & 4 & 0 & 4 & 5 \\ 5 & 3 & 4 & 6 & 0 & 4 & 6 & 3 \\ 5 & 5 & 4 & 0 & 3 & 6 & 5 & 3 \end{pmatrix} .$$

(a) Use the greedy heuristic to find a solution. Give the greedy value Z^G and the dual greedy value W^G.

(b) Give the value W' of the weak linear programming relaxation.

(c) Apply the dual descent procedure, cycling through the indices i. Give the value $W(u)$ found by this procedure. Is the set $K(u)$ given by (5.3) an optimal set of open facilities for this problem instance?

8. Let P denote the set of nine integral points in the square $0 \leq x \leq 2$, $0 \leq y \leq 2$. For any point $j \in P$, let x_j and y_j be its coordinates. Consider the instance of the UFL Problem defined by

$$m = n = 9,\ f_j = 2 \text{ for all } j \in P \text{ and}$$

$$c_{ij} = -\left|x_i - x_j\right| - \left|y_i - y_j\right| \text{ for all } i,j \in P.$$

(a) Use the greedy heuristic to find values Z^G and W^G.

(b) Apply the variation of the dual descent procedure based on the sets $Q_i(u)$. Is the set $K(u)$ given by (5.3) an optimal set of open facilities for this problem instance?

9. Find an instance of the UFL Problem having the following properties: (i) $Z = W$, (ii) c_{ij} is integer for every $i \in I$, $j \in J$ and f_j is integer for every $j \in J$; (iii) any optimum dual solution $(u_1,\ldots,u_m)$ has at least one fractional coordinate u_i.

Can the dual descent procedure solve the SLPR for this problem instance?

10. Consider the instance of the UFL Problem defined in Exercise 7.

(a) Starting from the point u^0 defined by $u_i^0 = \max_{j=1,\ldots,8} c_{ij}$, perform two iterations of the subgradient algorithm applied to the Lagrangian dual. Take $\gamma^0 = 1.2$ and $\gamma^1 = .4$ as the respective step sizes.

(b) Solve the problem using the Dantzig-Wolfe approach. Start with $R_i = \{i\}$, $i = 1,\ldots,5$ and, in the iterative step, take $R_k = \{i: c_{ik} - u_i^p > 0\}$ whenever there is a tie for the choice of R_k.

(c) Starting from the point $x^0 = (0,0,0,0,1,1,1,1)$, perform three iterations of the primal subgradient algorithm using the step sizes $\gamma^0 = \gamma^1 = 1/4$ and $\gamma^2 = 1/16$.

(d) Solve the problem using Benders decomposition. Start from the five constraints (6.5.4), one for each i, determined by the index k such that c_{ik} is the second maximum c_{ij}. Then choose an optimal solution x^1 to this linear program which is not identically zero.

11. Let p and q be two positive integers and consider a $p \times q$ matrix A with elements $a_{ij} = 0$ or 1 for all $i = 1,\ldots,p$ and $j = 1,\ldots,q$. Let b be a p-vector and c be a q-vector. Given any set S of columns of A, let $I(S) = \{i: a_{ij} = 1 \text{ for at least one } j \in S\}$.

We define the following problem.

Find a set S of columns of A which maximizes $\sum_{i \in I(S)} b_i - \sum_{j \in S} c_j$.

(a) Formulate this problem as an integer program.

(b) How does this relate to the canonical formulation (6.6.3)?

12. (a) Show that the constraint matrix of the UFL Problem is totally unimodular when $m \leq 2$.

(b) Show that it is totally unimodular when $n \leq 2$.

13. (a) Show that the conditions (i)-(iii) of Theorem 3.11 are necessary.

(b) Prove Theorem 3.12.

14. Prove Theorem 3.13.

15. Show that the inequality of Theorem 3.15 satisfies all the conditions of Theorem 3.14.

16. Prove Theorem 3.17.

17. (a) Prove that, if a set function w is submodular and nondecreasing, then

$$w(T) \leq w(S) + \sum_{j \in T-S} \rho_j(S) \quad \text{for all} \quad S,T.$$

(b) Consider a p-median problem with $c_{ij} \geq 0$. Let ρ_t be the increase in the profit value achieved at the t^{th} step of the greedy heuristic, $t = 1,\ldots,p$. Show that the optimal solution Z of the p-median problem satisfies

$$Z \leq \sum_{t=1}^{k-1} \rho_t + p\rho_k \quad \text{for} \quad k = 1,\ldots,p.$$

(c) Use the inequalities found in (b) and the fact that the greedy value is

$$Z^G = \sum_{t=1}^{p} \rho_t \quad \text{to prove that} \quad \frac{Z^G}{Z} \geq 1 - \left(\frac{p-1}{p}\right)^p .$$

References

S. Ahn, G. Cornuejols and A.M. Frieze (1984), to appear.

M.R. Anderberg (1973) Cluster Analysis for Applications, Academic Press, New York.

D.A. Babayev (1974) "Comments on a Note of Frieze," Mathematical Programming 7, 249-252.

E. Balas and M.W. Padberg (1976) "Set Partitioning: A Survey," SIAM Review 18, 710-760.

M.L. Balinski (1965) "Integer Programming: Methods, Uses, Computation," Management Science 12, 253-313.

M.L. Balinski and P. Wolfe (1963) "On Benders Decomposition and a Plant Location Problem," Working Paper ARO-27, Mathematica.

I. Barany, J. Edmonds and L.A. Wolsey (1984) "Packing and Covering a Tree by Subtrees," Discussion Paper 8434, Center for Operations Research and Econometrics, Universite Catholique de Louvain.

M.P. Beck and J.M. Mulvey (1982) "Constructing Optimal Index Funds," Report EES-82-1, School of Engineering and Applied Science, Princeton University.

J.F. Benders (1962) "Partitioning Procedures for Solving Mixed Variables Programming Problems," Numerische Mathematik 4, 238-252.

O. Bilde and J. Krarup (1977) "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem," Annals of Discrete Mathematics 1, 79-97.

Businessweek (1974) "Making Millions by Stretching the Float," November 23, 89-90.

D.C. Cho, E.L. Johnson, M.W. Padberg and M.R. Rao (1983) "On the Uncapacitated Plant Location Problem I: Valid Inequalities and Facets," Mathematics of Operations Research.

D.C. Cho, M.W. Padberg and M.R. Rao (1983) "On the Uncapacitated Plant Location Problem II: Facets and Lifting Theorems," Mathematics of Operations Research.

M. Conforti and G. Cornuejols (1984) "Submodular Set Functions, Matroids and the Greedy Algorithm: Tight Worst-Case Bounds and Some Generalizations of the Rado-Edmonds Theorem," Discrete Applied Mathematics 7, 251-274.

S.A. Cook (1971) "The Complexity of Theorem-Proving Procedures," Proceedings 3rd Annual ACM Symposium on the Theory of Computing, 151-158.

G. Cornuejols, M.L. Fisher and G.L. Nemhauser (1977a) "On the Uncapacitated Location Problem," Annals of Discrete Mathematics 1, 163-177.

G. Cornuejols, M.L. Fisher and G.L. Nemhauser (1977b) "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms," Management Science 23, 789-810.

G. Cornuejols, G.L. Nemhauser and L.A. Wolsey (1980) "A Canonical Representation of Simple Plant Location Problems and its Applications," SIAM Journal on Algebraic and Discrete Methods 1, 261-272.

G. Cornuejols and J.M. Thizy (1982a) "Some Facets of the Simple Plant Location Polytope," Mathematical Programming 23, 50-74.

G. Cornuejols and J.M. Thizy (1982b) "A Primal Approach to the Simple Plant Location Problem," SIAM Journal on Algebraic and Discrete Methods 3, 504-510.

M.A. Efroymson and T.L. Ray (1966) "A Branch and Bound Algorithm for Plant Location," Operations Research 14, 361-368.

D. Erlenkotter (1978) "A Dual-based Procedure for Uncapacitated Facility Location," Operations Research 26, 992-1009.

M.L. Fisher and D.S. Hochbaum (1980) "Probabilistic Analysis of the Planar K-Median Problem," Mathematics of Operations Research 5, 27-34.

M.L. Fisher, G.L. Nemhauser and L.A. Wolsey (1978) "An Analysis of Approximations for Maximizing Submodular Set Functions II," Mathematical Programming Study 8, 73-87.

A.M. Frieze (1974) "A Cost Function Property for Plant Location Problems," Mathematical Programming 7, 245-248.

M.R. Garey and D.S. Johnson (1979) Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco.

R.S. Garfinkel, A.W. Neebe and M.R. Rao (1974) "An Algorithm for the M-median Plant Location Problem," Transportation Science 8, 217-236.

A.M. Geoffrion (1974) "Lagrangian Relaxation for Integer Programming," Mathematical Programming Study 2, 82-114.

M. Grötschel (1980) "On the Symmetric Travelling Salesman Problem: Solution of a 120-city Problem," Mathematical Programming Study 12, 61-77.

M. Guignard (1980) "Fractional Vertices, Cuts and Facets of the Simple Plant Location Problem," Mathematical Programming Study 12, 150-162.

M. Guignard and K. Spielberg (1977) "Algorithms for Exploiting the Structure of the Simple Plant Location Problem," Annals of Discrete Mathematics 1, 247-271.

P. Hansen (1972) "Two Algorithms for the Simple Plant Location Problem Using Additive Penalties," presented at the European Congress of the Econometric Society, Budapest.

P. Hansen and L. Kaufman (1972) "An Algorithm for Central Facilities Location under an Investment Constraint," in P. Van Moeseke ed., Mathematical Programs for Activity Analysis, North Holland, Amsterdam.

M. Held, P. Wolfe and H.P. Crowder (1974) "Validation of Subgradient Optimization," Mathematical Programming 6, 62-88.

R.M. Karp (1972) "Reducibility among Combinatorial Problems," in R.E. Miller and J.W. Thatcher eds., Complexity of Computer Computations, Plenum Press, New York, 85-104.

B.M. Khumawala (1972) "An Efficient Branch and Bound Algorithm for the Warehouse Location Problem," Management Science 18, 718-731.

A. Kolen (1982) "Location Problems on Trees and in the Rectilinear Plane," Mathematisch Centrum, Amsterdam.

J. Krarup and O. Bilde (1977) "Plant Location, Set Covering and Economic Lot Sizing: An O(mn) Algorithm for Structured Problems," in L. Collatz et al eds., Optimierung bei graphentheoretishen and ganzzahligen probleme, Birkhauser Verlag, Basel, 155-180.

J. Krarup and P.M. Pruzan (1983) "The Simple Plant Location Problem: Survey and Synthesis," European Journal of Operations Research 12, 36-81.

A. Kraus, C. Janssen and A. McAdams (1970) "The Lock-box Location Problem, a Class of Fixed Charge Transportation Problems," Journal of Bank Research 1, 51-58.

A.A. Kuehn and M.J. Hamburger (1963) "A Heuristic Program for Locating Warehouses," Management Science 9, 643-666.

T.L. Magnanti and R.T. Wong (1981) "Accelerated Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria," Operations Research 29, 464-484.

R.E. Marsten (1972) "An Algorithm for Finding Almost All the Medians of a Network," Discussion Paper 23, The Center for Mathematical Studies in Econometrics and Management Science, Northwestern University.

K. Martin and L. Shrage (1982) "Some Cuts for 0/1 Mixed Integer Programming," Mimeo, University of Chicago.

L.P. Mavrides (1979) "An Indirect Method for the Generalized K-Median Problem Applied to Lock-Box Location," Management Science 25, 990-996.

N. Megiddo, E. Zemel and S.L. Hakimi (1983) "Maximum Coverage Location Problem," SIAM Journal on Algebraic and Discrete Methods 4, 253-261.

J.G. Morris (1978) "On the Extent to which Certain Fixed-Charge Depot Location Problems can be Solved by LP," Journal of the Operational Research Society 29, 71-76.

C. Mukendi (1975) "Sur l'Implantation d' Equipement dans un Reseau: le Probleme de m-centre," University of Grenoble, France.

J.M. Mulvey and H.L. Crowder (1979) "Cluster Analysis: An Application of Lagrangian Relaxation," Management Science 25, 329-340.

S.C. Narula, U.I. Ogbu and H.M. Samuelson (1977) "An Algorithm for the p-median Problem," Operations Research 25, 709-713.

G.L. Nemhauser and L.A. Wolsey (1978) "Best Algorithms for Approximating the Maximum of a Submodular Set Function," Mathematics of Operations Research 3, 177-188.

G.L. Nemhauser and L.A. Wolsey (1981) "Maximizing Submodular Set Functions: Formulations and Analysis of Algorithms," Annals of Discrete Mathematics 11, 279-301.

G.L. Nemhauser, L.A. Wolsey and M.L. Fisher (1978) "An Analysis of Approximations for Maximizing Submodular Set Functions I," Mathematical Programming 14, 265-294.

M. Padberg and S. Hong (1980) "On the Symmetric Travelling Salesman Problem: A Study," Mathematical Programming Study 12, 78-107.

C.H. Papadimitriou (1981) "Worst-Case and Probabilistic Analysis of a Geometric Location Problem," SIAM Journal on Computing 10, 542-557.

B.T. Polyak (1969) "Minimization of Unsmooth Functionals," U.S.S.R. Computational Mathematics and Mathematical Physics, 14-29.

C.S. ReVelle and R.S. Swain (1970) "Central Facilities Location," Geographical Analysis 2, 30-42.

L. Schrage (1975) "Implicit Representation of Variable Upper Bounds in Linear Programming," Mathematical Programming Study 4, 118-132.

K. Spielberg (1969a) "Plant Location with Generalized Search Origin," Management Science 16, 165-178.

K. Spielberg (1969b) "Algorithms for the Simple Plant Location Problem with Some Side Conditions," Operations Research 17, 85-111.

T.J. Van Roy and L.A. Wolsey (1983) "Valid Inequalities for Mixed 0-1 Programs," CORE Discussion Paper 8316, University of Louvain, Belgium.

R. Wong, J. Ward, A. Oudjit and P. Lemke (1984) "Linear Programming Models of the K-Median Location Problem on Special Structure", presented at ORSA/TIMS Meeting, Dallas.