A COST OPERATOR APPROACH TO MULTISTAGE LOCATICW-ALLOCATION

by

Robert V. Nagelhout & Gerald L* Thompson

DRC-70-6-79

September 1979

Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, PA 15213

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract N00014-75-C-0621 NR 047-048 with the U.S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

1. Introduction

In this paper we present a cost operator algorithm for solving multistage factory warehouse location-allocation problems. The decision variables correspond to the warehouse locations and the shipping amounts from the factories through the warehouses and into the demand centers. The problem is to minimize the total fixed costs of locating warehouses plus the total variable shipping cost, subject to possible capacity restrictions at the factories and the warehouses, plus the demand requirements at the demand centers.

The algorithm takes advantage of the network structure of the supply and demand constraints and the submodularity of the objective function. We use cost operators [22], to facilitate the movement up and down the search tree. This greatly reduces the amount of time spent solving transportation subproblems, which frequently comprises up to 90% of the computational burden.

In Section 2 we give a problem formulation and we discuss some of the more recent research related to the multistage location problem. In Section 3 we point out lower bounds and fathoming rules obtainable from submodular set functions. In Section 4 we transform the multistage location problem into a transportation problem format, and we show how cost operators can be used to generate feasible solutions. Section 5 contains a description of the cost operator branch and bound algorithm. In Section 6 we give an example, and in Section 7 we provide extensive computational experience on problems from the literature plus some multistage problems of our own.

2. Problem Formulation

We describe the location problem to be studied in this paper as well as similar models which have been presented in the literature. We use the following notation:

q = number of factories; $I' = \{1,\ldots,q\}$

ℓ = number of warehouses; $I = \{q+1,\ldots,m\}$

r = number of demand centers

$m = q + \ell + 1$; $n = \ell + r + 1$; $J' = \{1,\ldots,\ell\}$, $J = \{\ell+1,\ldots,n\}$

f_i = fixed cost of opening warehouse i

c_{ij} = cost per distance per unit of shipping from
- factory i to warehouse j : $i \in I'$, $j \in J'$
- factory i to demand center j : $i \in I'$, $j \in J$
- warehouse i to demand center j : $i \in I$, $j \in J$

A_i = capacity of factory i

S_i = capacity of warehouse i

d_j = demand at location j

x_{ij} = amount shipped from location i to location j

y_i = $\begin{cases} 1 & \text{if warehouse } i \text{ is open} \\ 0 & \text{otherwise} \end{cases}$

The multistage or intermediate location problem, which we will call Problem P, can be formulated as:

$$\underset{T \subseteq I}{\text{Minimize}}\ Z(T) = \sum_{i \in I' \cup T} \sum_{j \in J' \cup J} c_{ij} x_{ij} + \sum_{i \in T} f_i y_i \qquad (1)$$

subject to

$$\sum_{j \in J' \cup J} x_{ij} \leq A_i \qquad i \in I' \qquad (2)$$

$$\sum_{j \in J} x_{ij} - S_i y_i \leq 0 \qquad i \in T \qquad (3)$$

$$\sum_{i \in I' \cup T} x_{ij} \geq d_j \qquad j \in J \qquad (4)$$

$$\sum_{h \in I} x_{hi} - \sum_{j \in J} x_{ij} = 0 \qquad i \in T \qquad (5)$$

$$x_{ij} \geq 0 \qquad i \in I, \; j \in J \qquad (6)$$

$$y_i = \begin{cases} 1 & \text{if warehouse } i \text{ is opened} \\ 0 & \text{otherwise} \end{cases} \qquad i \in I \qquad (7)$$

Constraints (2) and (3) ensure that the amount shipped out of a factory or a warehouse should not exceed its capacity. Constraints (4) require that the demand at each demand center be satisfied. Constraints (5) are the standard "conservation of flow" constraints which require that the amount shipped into each warehouse equals the amount shipped out. Constraints (6) and (7) are the nonnegativity and integrality constraints respectively. The objective in problem P is to minimize the total transportation costs from shipping plus the total fixed costs of opening warehouses, while satisfying the customer demand. When

$$S_i < \sum_{j \in J} d_j \qquad (8)$$

we say that warehouse i is <u>capacitated</u> since it cannot satisfy all of the demand by itself. The factories can also be either capacitated or uncapacitated depending upon the size of A_i. Note that in the multistage formulation when $i' > 0$, the total warehouse capacity need not exceed the total demand since units can also be shipped directly from the factories to the demand centers.

A considerable amount of research has been performed on uncapacitated, capacitated, and mixed (partly capacitated) location problems. It seems that each problem which is studied, depending upon the objective function and the capacity assumption, exhibits its own characteristics and yields a different algorithmic approach. Thus the literature on the many different location

problems is enormous. In this section we will discuss recent literature only for those problems which are closely related to P.

Research on multistage location problems has been limited. Geoffrion and Graves [12] described and tested an algorithm for solving multistage multicommodity distribution systems using Benders' decomposition. Ellwin and Gray [8] described and tested an algorithm for solving single stage (i' * 0) location problems and proposed, but did not test, an algorithm for solving multistage problems. Most of the papers in the literature have concentrated on the single stage version of P for which i' • 0. This has been called the simple (capacitated or uncapacitated) warehouse location problem which we denote by PI. Some of the models which have been studied impose added configuration constraints on PI which restrict the total number, and different combinations of warehouses which can be opened. For example, a constraint frequently used is:

$$\sum_{i \in I} y_i \leq K \qquad (9)$$

where K is an upper limit on the total number of warehouses which can be opened. In the special case where $f_i^1 \cdot 0$ for $i \in I$ and each warehouse is uncapacitated, PI with constraint (9) ammended is called the K median problem. Both heuristic [14] and exact methods [5], [16], [21], have been proposed for solving the K median problem.

Earlier attempts at solving the uncapacitated PI concentrated upon the relaxation, Pi', of PI where (7) is replaced by:

$$0 \leq y_i \leq 1 \quad i \in I \qquad (10)$$

(see [6], [7]). The basic idea was to solve PI by imbedding Pi' into an

implicit enumeration scheme. Since then Pi' has been called the weak linear programming relaxation of PI. It is termed "weak" because there is another linear programming relaxation of PI which has proven to be much stronger than PI'. To describe the latter, we add to problem PI the constraints:

$$x_{ij} \leq \min \{s_i, d_j\} y_i \qquad i \in I, j \in J \qquad (11)$$

Let us denote by Pi" the problem in which the constraints (11) are added to PI '. For uncapacitated location problems Pi" is called the strong linear programming relaxation of PI. (Note that for uncapacitated location problems $\min\{s_i, d_j\} = \min\{\infty, 1\} * 1$). Pi" is stronger than PI' in the sense that the gap between the optimal values of PI and Pi" is normally much smaller than the gap between the optimal values of PI and PI'. Also, it is often the case for uncapacitated location problems, that a solution to Pi" will satisfy, or almost satisfy (7). That is, after solving 'PI" almost all of the fixed charge variables will be naturally integer. However, even though optimal solutions to PI" tend to be close approximations to an optimal solution for PI, researchers have attempted to avoid solving PI" directly, because it has an enormous number of constraints (there are mxn constraints of the type (11)). Schrage [20], has proposed a method for solving linear programs which handles constraints of the type (11) implicitly, thereby reducing storage requirements. Still the time required to solve PI" by the simplex method can be excessive and could cause difficulties in an implicit enumeration scheme.

Instead of solving PI" directly, heuristic methods have been proposed which find feasible solutions to the dual of PI". To describe these approaches let us denote by $X^{\wedge}_{-1}$ a feasible, and by X^{*}_{-1} an optimal solution to problem PI.

Let Z(X) represent the objective function value for solution X. Also let D be the dual problem to problem Pi" . Then by duality theory we know that:

$$Z(Pi^{*}) \geq Z(X_{D}^{*}) \geq Z(V^{*}) \geq Z(V) \qquad (12)$$

where X_D^* is an optimal and X_D a feasible solution to D. Thus a feasible solution to D provides a valid lower bound on the optimal objective function value of PI. Subgradient [4] and dual ascent methods [3] , [9] have been used to find good feasible solutions to D. In the case where PI is uncapacitated one can easily compute a low cost primal solution to PI after having found the feasible solution to D. Furthermore it is often the case when Pi is uncapacitated that the gap, $Z(X_{pl}^{*}) - Z(X_{D}^{*})$ is very small or even zero, which makes the additional work to get optimal solutions small. Thus the relaxation Pi" has been very effective in solving the simple uncapacitated warehouse location problems.

Several algorithms also exist for solving capacitated location problems. Akinc and Khumawala [1]> proposed and tested an implicit enumeration algorithm which uses Pi^{7} as a relaxation. Ellwin and Gray [8] described and tested a branch and bound algorithm which uses duality properties of PI and bounds obtainable from the submodularity of the objective function (1) for fathoming. Guignard and Spielberg [13] have generalized the dual ascent method of Bilde-Krarup [3] and Erlenkotter [9] to the capacitated version of PI. They use a relaxation very similar to Pi" which contains the constraints (11). They solved some randomly generated problems and found that the zero gap phenomenon between $Z(X_{I}^{*})$ and $Z(X_{J})$ occurs less frequently with capacitated or mixed problems, than it does with uncapacitated problems. Other relaxations besides PI^{7} and Pi" have also been used to solve PI. Geoffrion and McBride [11]

have considered a location model for which PI is a special case and have used a Lagrangian relaxation combined with implicit enumeration to solve it. Recently Nauss [17] used a Lagrangian relaxation to solve PI with excellent computational results.

In the following sections we will describe an algorithm for solving P. The general approach is similar to the one used in [8] to solve PI. We do not solve a relaxation of P. Instead we make use of some lower bounds obtainable from the submodular property of Z, together with some other fathoming rules, to enumerate explicitly a subset of the solutions to P. The movement in the search tree from one solution to another is facilitated by applying cost operators [21] to P. This significantly reduces the amount of effort required to solve P, since most researchers have found that the majority of the time involved in solving capacitated location problems is spent solving transportation subproblems. Ellwin and Gray [8] have tested and shown that in many cases over 90% of the time required to solve a test problem is spent solving transportation subproblems. In the next section we discuss some lower bounds on the value of Z, some of which are utilized in the algorithm described in Section 5*

3. Objective Function Lower Bounds

Many of the lower bound properties which we present here have been discussed in [2] and [10], in the context of the simple uncapacitated location problem. These lower bounds and their properties are also useful in solving other kinds of mathematical programming problems.

There are many properties which can be used to define submodular set functions. For a discussion of them see [18], To define one such function let Z be a real valued function defined on the finite set of subsets of I.

For notational convenience we define:

$$AZ_B(A) \equiv Z(A \cup B) - Z(A) \quad A \subseteq I,\ B \subseteq I \tag{13}$$

Then any of the following properties can be used to define a submodular function:

(i) $Z(A \cup B) + Z(A \cap B) \geq Z(A) + Z(B)$ for all subsets $A, B \subseteq I$.

(ii) $AZ_i(A) \leq AZ_t(B) \quad \forall A \subseteq B \subseteq I,\ i \in I$ (14)

(iii) Let $\{B_1, \ldots, B_r\}$ be a partition of $B - A$; then

$$Z(A) \geq Z(B) - \sum_{k=1}^{r} AZ_k(B - B_k) \quad A \subseteq B \subseteq I. \tag{15}$$

In most expositions property (i) is taken to be the definition of submodularity, and properties (ii) and (iii) are shown to be equivalent. Details are omitted.

In the context of this paper, $Z(A)$ represents (1). The fact that $Z(A)$ is submodular was proved in [15]. Note that in P, $Z(0)$ represents the value of the solution where all of the shipments originate from the factories. When $l' \neq 0$, as in Pi, one must be careful in defining $Z(A)$ if A happens to be infeasible set of warehouses, i.e., when $\sum_{i \in A} S_i < \sum_{j \in J} d_j$. In this case we we define a "dummy factory" which is always available to service the demand centers but at a high shipping cost. This makes $Z(A)$ large enough so that the value of $Z(A)$ for any infeasible A is at least as large as the value of any feasible solution. The dummy factory approach preserves the submodularity of Z and also yields a different value of Z for solutions having different degrees of infeasibility. As we will see later, this is helpful in deciding which warehouses to open when we are working with infeasible sets of warehouses.

Properties (14) and (15) can be interpreted as adding or subtracting warehouses from a given set A of open warehouses. For example property (14)

says that the addition of warehouse i to the set A decreases the total cost by at least as much as the addition of warehouse i to the set B when A ⊆ B. Thus it is similar to the "decreasing returns to scale" condition in economics.

Properties (14) and (15) can be used to characterize solutions to P and to derive lower bounds on the value of any solution to P. The algorithm to be described in Section 5 searches for sets T ⊂ I which have the following two properties:

(i) $AZ_{\pm}(T) \geq 0 \quad \forall\, i \in I-T$

(ii) $AZ_{\pm}(T-\{i\}) \leq 0 \quad \forall\, i \in T$

Sets T ⊆ I which satisfy (i) and (ii) are such that the addition to T or the deletion from T of a single facility does not cause Z to decrease. Clearly any optimal solution to P satisfies (i) and (ii). In a sense properties (i) and (ii) characterize the set of all "locally optimal" solutions to P. The globally optimal solution is the best locally optimal solution, which must be found by a search process. One of the factors which make it hard to find the globally optimal solution is that there are many locally optimal solutions which have nearly optimal objective function values. Thus in any enumeration procedure, a considerable amount of effort is normally required to eliminate these nearly optimal solutions from consideration. In a practical sense however, it may be true that the nearly optimal solutions, say those within 1% of optimality may indeed be as valuable or "as optimal" to a decision maker as a globally optimal solution. Given the inaccuracies in the cost data and other environmental and polilical factors which must be taken into consideration, it would be desirable to have not only an optimal solution to P, but in addition a list of

solutions which are nearly optimal. We will point out how this may be accomplished using the algorithm of Section 5.

In the initialization phase of any location algorithm, two rules can be applied in order to permanently open or close warehouses.

RULE 1. If $\Delta Z_i(I - \{i\}) \leq 0$ for any $i \in I$ then the warehouse i will be open in some optimal solution to P.

To see this suppose that T is an optimal set of warehouses and $i \notin T$. By property (14)

$$\Delta Z_i(T) = Z(T \cup \{i\}) - Z(T) \leq \Delta Z_i(I - \{i\}) \leq 0,$$

thus $T \cup \{i\}$ is also an optimal solution.

RULE 2. If $\Delta Z_i(I^*) \geq 0$ for any $i \in I$, where I^* is the set of warehouses opened by application of RULE 1, then warehouse i will be closed in some optimal solution to P1.

The justification for RULE 2 is similar to that for RULE 1. We will show, in the next section how the testing of RULE 1 and RULE 2 requires only the application of two cost operators for each warehouse. In many cases, as will be seen in Section 7, RULES 1 and 2 can be used to fix open and closed a large portion of the warehouses in an optimal solution.

Another property of submodular set functions which can be derived from (14) and (15) is the following: Let $\{A_1,\ldots,A_r\}$ and $\{Q_1,\ldots,Q_t\}$ be partitions of $A - T$ such that $Q_i \subseteq A_j$ for each $i = 1,\ldots,t$ and some $j = 1,\ldots,r$. Then

$$Z(T) \geq Z(A) - \sum_{k=1}^{r} \Delta Z_{A_k}(A-A_k) \geq Z(A) - \sum_{k=1}^{t} \Delta Z_{Q_k}(A-Q_k) \qquad (16)$$

for all subsets T satisfying $T \subseteq A \subseteq I$. The quantity on the right hand side

of (15) provides a valid lower bound on the value of Z(T), for any T C I. Property (6) says when we use a more refined partition of A - T, we get a weaker lower bound. Suppose we let A = I. Then the most refined partition of I - T is clearly {{i,},...,U 13 where i, e I - T and t- |l - T|. Then for this partition (16) yields:

$$Z(T) \geq Z(I) - \sum_{k=1}^{t} bZ_{\cdot}\ (I - (i_k 3)) \quad \bullet \qquad (17)$$

Among the class of lower bounds (15), (17) is the weakest. Notice that once Rule 1 has been applied all of the terms on the right hand side of (17) have been calculated. Thus (17) can be applied at any time after Rule 1 has been tested without any extra computational effort. A stronger lower bound than (17) could be obtained with some added computational effort by partitioning I into sets, A_k, of size two and calculating $AZ_{\cdot A_k}(I - A_k)$. We have not yet tested this idea.

Another fathoming device which provides an upper bound on the maximum number of warehouses in an optimal solution to P can be obtained as follows: Let t« Ji - I*J. Let $f_{k_1},\ldots,f_{k_c}$, k_i e I - I*, be a nondecreasing ordering of the fixed charges not in I*. Define

$$DV_{j\ i} - AZ_{\cdot}(I - \{J.\}) - f_{\cdot}, \quad j_{\cdot} \text{ e } I - I^* \qquad (18)$$

and let $DV_{j_l},\ldots,DV_{j_h}$ be nonincreasing ordering of the DV_{j_i}. (Note that $DV_{j_i} < 0$.) The quantity $-DV_{j_i}$, represents the smallest possible incremental savings on the total shipping cost when warehouse j_i is opened. Let Z^u be any upper bound on the optimal objective function value for P. Then an upper bound on the number of warehouses in an optimal solution to P (containing I*) is 11 1 + k where,

$$Z(D-\sum_{U}^{k^*} f_{\cdot} + \sum_{t-1}^{t-k^*} f_{\cdot} + \sum_{8-1} f_{\cdot} \le Z^U < Z(I) - \sum_{t \in I} f_{\cdot} + \sum_{*-l}^{k^*+l} f_{\cdot} + \sum_{8-1}^{t-k^*-l} f_{\cdot} \quad (19)$$

Note that the first two terms on the right hand side of (19) give the smallest possible shipping cost, the next term is the smallest sum of $k + 1$ fixed charges, and the last term is the smallest possible cost of closing down $t-(k^* + 1)$ warehouses. Therefore the right hand side of (19) is a lower bound on the value of an optimal solution to P containing $|I| + k$ -Hi warehouses. If this lower bound exceeds Z^U then any optimal solution will contain at most $i * t * 11\ I + k$ open warehouses.

In the algorithm of Section 5 we use as fathoming devices bounds obtained from (17) on the value of an optimal solution, and bounds obtained from (19) on the number of warehouses in an optimal solution.

4. Use of Cost Operators to Solve Problem P.

Given the choice of a subset $T \subset I$, problem P becomes an ordinary transshipment problem. Rather than resolving this problem each time T changes, we use the operator theory of parametric programming [22] to change the problem and derive the new optimal solution simultaneously.

We shall say that cell (i,j) has been fixed out of the basis when a cell cost operator has been applied to the problem and its solution so that the cost c_{ij} has been driven to 4M, where M is so large that $x_{ij} - 0$ in any optimal solution to the new problem.

We also say that cell (i,j) has been fixed in the basis when a cell cost operator has been applied to the problem and its solution so that the cost c_{ij} has been driven to -M, where M is so large that $x_{ij} \ll \text{Min}(S_i, d_j)$ in any optimal solution to the new problem.

Figure 1 shows an example of a 2X3X4 multistage location problem. Rows 1 and 2 correspond to the factories, rows 3, 4, and 5 to the warehouses, and row 6 represents the dummy factory. Columns 1, 2 and 3 correspond to the warehouses, columns 4 through 8 to the demand centers, and column 9 is a slack column. Rows 1 and 2 contain the costs of shipping from the factories to the warehouses and from the factories directly to the demand centers. Cells (3,1), (4,2), and (5,3) contain the fixed charge of the corresponding warehouse, divided by its capacity. In any solution to P, these cells will contain the unused warehouse capacity. Cells (3,2), (3,3), (4,1), (4,3), (5,1) and (5,2) cannot be used because of their large costs; in effect these arcs have been removed from the problem. The cells in rows 3 through 5, and columns 4 through 8, contain the costs of shipping from each of the warehouses to each of the demand centers plus the proportional fixed charges. Notice that some of the cells in dummy factory row 6 and slack column 9 contain two costs. This can be explained in the following manner. To obtain the solution where:

$y_l \cdot 0$ set $c_{in} * -M$ and $c_{m,i-q} \cdot -M$

$y_i = 1$ set $c_{in} \ll M$ and $c_{m,i-q} \gg M$

(in Figure 1, $m \ll 6$, $n - 8$, $q - 2$, and $i \in \{3,4,5\}$).

To see this consider the problem shown in Figure 2. By solving the transportation problem in Figure 2 we would obtain an optimal solution to P when $T = \{2,3\}$, i.e., when warehouse 1 is closed and warehouses 2 and 3 are open. Notice that cell (3,8) has a cost of -M, thus in the optimal solution $x_{3g} * S_3$, which has the effect of closing down warehouse 1. Cell (6,1) has a cost of -M which causes all of the demand in column 1 to be satisfied by the dummy factory i.e., $x_{61} \cdot S_3$. Thus setting $c_{3g} \ll -M$ and $c_{61} = -M$

effectively closes down warehouse 1. On the other hand for warehouse 2, $c_{,o} * M$ and $c._o \gg M$, so that $x._Q * x_{,o} * 0$. This causes the demand in column 2 to be satisfied from rows 1, 2, or 4. The shipments from rows 1 and 2 represent units being shipped from factories 1 and 2 to warehouse 2. The shipment from row 4 in column 2 represents the unused warehouse capacity, and it is charged at the proportional fixed charge rate. Notice that if cell (4,2) contains x_{4_2} units then exactly $S_{,4} - x_{4,2}$ units can be used to ship from warehouse 2 to the demand centers. This is exactly the amount which is made available to warehouse 2 from the factories. Thus the conservation of flow equations (5) have been satisfied. Because a proportional amount of the fixed charge is assigned to both the used and unused parts of the warehouse capacity, the total fixed charge is covered in any feasible solution.

Figure 5 contains an optimal solution to the 2x4x5 examples solved in Section 6. In Figure 5, T • £l,3} so that warehouses 1 and 3 are opened, and warehouses 2, 4, and 5 are closed* Factory 1 is not used at all so that $x_{1,1^Q} * 61$, Factory 2 ships 21 units to warehouse 1, 40 units to warehouse 3, and 16 units directly to demand center 1. Warehouse 1 ships all 21 of the units it received from Factory 2 to demand center 3. The remaining unused 4 units of capacity at warehouse 1 are in cell (3,1) and are charged at the proportional fixed charge rate. All of the flow for warehouses 2, 4, and 5 is in the last column, and their demand is satisfied entirely by the dummy factory. Warehouse 3 is used to capacity shipping 22 units to demand center 2, and 18 units to demand center 4. In this example each of the demand centers is supplied from a single warehouse. This is not the case in general.

The idea behind the cost operator approach to solving ?(T) is to open the warehouses in T and close those in I - T by fixing in or out of the

basis each of the cells (m,l) through $(m_y l)$ and (q+1, n) through q-ht, n). For example the problem P(I), in which all of the warehouses are open, can be obtained by solving the transportation problem shown in Figure 3. Then to apply Rule 1 in Section 3 we use two cost operators for each i e I. For example we would calculate $AZ_3(I - (3\}) \text{ « } Z(I) - Z(I - [3))$ by fixing in cells (3,8) and (6,1). Doing this would yield a solution to the problem shown in Figure$_v$2, and permits the evaluation of Rule 1. In general, the addition of a warehouse to a given set or the deletion of a warehouse from a given set requires the fixing in or out of two cells. The amount of work needed to fix two cells in or out using cost operators is much less than the computational effort of solving a transshipment problem from scratch. Since the branch and bound search algorithm to be described in the next section requires the solution of transshipment problems for many sets T, the total computational effort saved by the fixing in and fixing out procedure is very large.

Also we should mention that in the case where $I^7 \cdot 0$, as in single stage problems, only one cost operator is needed to open or close a warehouse. In the single stage formulation columns 1 through *K,* and rows 1 through 2 are absent. Thus in order to fix in or out warehouse i we need only apply a cost operator to cell (i,n).

5. The Cost Operator Algorithm.

To describe the cost operator branch and bound algorithm we use the following notation:

£ * level of the search tree

Q « set of open warehouses

List(£) « list of warehouses which may be opened on level £

Z^U * current upper bound

X * current best solution.

We begin by setting $I \cdot 1$, $Q - 0$ and $Z^U \gg Z(0)$ which is the (high) cost of supplying all demands from the factories directly. Then Rule 1 (see Section 3) is applied and we let $Q * I$. Then for each $i \in I\text{-}Q$ we apply the lower bound test (17) by setting $T \gg Q \cup (i)$, and checking to see whether

$$Z^U \geq Z(I) - \sum_{k \in T} AZ_{,k}(I - 00). \qquad (20)$$

If the right hand side of (20) (which is a lower bound on the value of $Z(Q \cup \{i\})$) exceeds Z^u, then warehouse i can be permanently closed. For each i for which (20) holds true we apply cost operators to calculate $AZ_i(Q)$. If $AZ_i(Q) \geq 0$ then we permanently close warehouse i. Then we let List(l) be the set of all i such that $AZ_i(Q) < 0$. (We place the warehouses in List (1) in order of non-increasing objective function values.) Next we remove the last warehouse, i, from List (1) and replace Q by $Q \cup (ij$. We let $Z^U * Z(Q)$, update X. If List (1) is empty then we stop. Otherwise we calculate k^* as in (19). If $k^* \ll 1$ then we stop. Otherwise we replace $K\gg$ by $\pounds + 1$ and continue to the next level. At each level after the first, we perform the lower bound test (20) for each warehouse, i, in List (£-1) by letting $T \cdot Q \cup \{i\}$. If warehouse i passes the test (20) we calculate $AZ_i(Q)$. Again we let List (£) be the set of warehouses such that $AZ_i(Q) < 0$ in nonincreasing order. If List (£) is not empty we remove i, the last element of List (<t); let $Z^U * Z(Q \cup \{i\})$ and update X- If List *(I)* is now empty we backtrack by replacing *If* by £-1, removing the last warehouse paced in Q and then continuing as before. Otherwise we calculate k as in (19). If $\pounds \geq k$ we bracktrack

as just described. Otherwise we replace 4 by 4+1; replace Q by Q U {i}, and continue as before. The algorithm terminates when List (1) has been emptied.

Now we formally state the cost operator multi-stage location-allocation algorithm

Step (1). (Initialization). Set 4 - 1; z^U « Z(0). Apply Rule 1. Let Q = I*. For each i € I-Q apply test (20), letting T - Q U {i}. For each i which satisfies (20) calculate AZ^Q). Order those i such that $AZ_{\pm}(Q) < 0$ in nonincreasing order and place them in List (1). If List (1) empty go to Step (5). Otherwise let k be the last warehouse in List (1). Let Z^U - Z(Q U (k}); update X. Remove warehouse k from List (1). If List (1) is empty go to Step (5). Otherwise go to Step (2).

Step (2). (Bound by k^*). Calculate k* as in (19). If $4 \geq k^*$ go to Step (4). Otherwise replace 4 by 4+1; replace Q by Q U {k} and go to Step (3).

Step (3). (Forward branching). For each warehouse i in List (4-1) apply test (20) letting T « Q U {i}. For each i which satisfies (20) calculate AZ^Q). List those i such that $AZ_i(Q) < 0$ in non-increasing order and place them in List (4). If List (4) is empty go to Step (4). Otherwise let k be the last warehouse in List (4). Let Z^U s Z(Q U {k}); update X. Remove warehouse k from List (4). If List (4) is empty go to Step (4). Otherwise go to Step (2).

Step (4). (Backtracking). Replace 4 by 4-1. Remove from Q the last warehouse placed in Q. If 4 « 1 and List (1) contains only one warehouse go to Step (5). Otherwise let k be the last warehouse in List (4).

Remove k from List (ℓ). If List (ℓ) empty go to Step (4). Otherwise replace Q by $Q \cup \{k\}$; replace ℓ by $\ell+1$ and go to Step (3).

Step (5). (Termination) Stop. The current solution X, is optimal.

Notice that each time Step (3) is performed a new list of feasible solutions to P is available. Many times these solutions have nearly optimal objective function values and thus they might be worthwhile to save.

After calculating I^*, the cost operator algorithm proceeds to open, one at a time, those warehouses which cause the largest decrease in the objective function value. This is continued until at level ℓ, no further decrease in the objective function value is possible (i.e. List (ℓ) is empty). Then we move to the backtracking Step (4). The best feasible solution obtained by the cost operator algorithm before the first backtracking is called the greedy solution. We denote by Z^G the objective function value of the greedy solution. In [18], a worst case bound on Z^G was derived for submodular set functions which is,

$$\frac{Z^G - Z^*}{Z(\emptyset) - Z^*} \leq \frac{|I^*| + k^* - 1}{|I^*| + k^*} \tag{21}$$

where Z^* is the optimal objective function value and k^* is obtained from (19). In practice however, the actual percentage error obtained by the greedy solution is much smaller than the worst case bound. In Section 7 we will see that in most cases the greedy solution value is within .5 percent of optimality.

Finally we should point out that it would be trivial to "reverse" the cost operator algorithm so that instead of starting with all of the warehouses closed, and opening them one at a time, we could start with all of the warehouses open and close them one at a time. This could be done by defining $Z'(T) = Z(I-T)$

in (1). $Z'(T)$ is also submodular, and results from Section 3 would remain valid for z'. The reverse algorithm might be better suited to handle problems where an optimal number of warehouses tends to be close to the total number of potential warehouse sites.

6. Example.

Figure 4 shows the cost tableau for a 2x4x5 multistage location problem. Fixed charges for warehouses 1-5 are: 150, 217, 200, 264, and 140. For an explanation of how the costs in the tableau are calculated, see Figure 1 and Section 4. The diagram in Figure 6 is a 5-dimensional hypercube whose nodes represent all of the possible open warehouse combinations. The search tree will be a subgraph of this hypercube. The number above each node is the value of an optimal solution to P when only those warehouses marked in circle of the node are open. The number in the parenthesis above a node is the total cost lower bound for that node obtained by calculating (20).

We illustrate the steps of the algorithm applied to this example.

Step	Calculations
(1)	Set 4-1, Q«0. $Z^U = Z(0) = 2107$. Applying Rule 1 we get $AZ_1(I - \{l\})$»2303 - 2201 « 102; $AZ_2(I - \{2\})$ * 217 ; AZ_3<I - {3}) = 316; $AZ_4(I - \{4\})$ « 162; $AZ_5(I - \{5\})$ -• 140. I* - Q - 0. For each i e {1,2,3,4,5} apply test (20). None of the lower bounds exceeds the current upper bound. (See the numbers in parenthesis in Figure 6.) Next calculate ^(0) - 1880 - 2107 * -227; $AZ_2(0)$ - -94; $AZ_3(0)$ = -123; $AZ_4(0)$ - -136; $AZ_5(0)$ = -105. Let List (1) = {2,5,3,4}. Let Q = {l}; Z^U = 1880; 4 * 2 ; go to Step (2).

Step	Calculations
(2)	$Z(I) = 2334$; If. • 1002; the fixed charge ordering is f_{iel^x} = 140, f_x * 150, f4 « 200, f_2 * 217, f_3 = 264. The DV ordering is DV_2 - 0, DV5 » 0, DV_4 « 38, DV_L * 42, DV_3 = 48. Z^U » 1880. Setting k* = 2 we get from (19) 1332 + 290 + 38 ≺ 1880 ≺ 1332 +490 + 0. Since *I* * 1 < 2, let *I* « 2; let Q * {l}; go to Step (3).
(3)	For each i € {2,5,4,3} apply test (20) setting T « {l} U {i}. (See Figure 6.) For each i e {2,5,4,3} calculate AZ^fl}). AZ_3({l}) * -118. AZ_4({1}) » 32; AZ_5({1}) « 22; AZ_2({ll) *-18. Let Z^U » 1762. List (2) * {2}. Go to Step (4).
(2)	Setting k* * 2 we get from (19) 1332 + 290 + 38 ≤ 1762 ≤ 1332 + 490 + 0. Since *I* » 2 ≥ 2 go to Step (4).
(4)	Let *I m* 1. Remove {l} from Q (now Q = 0). Let k * 4. Let List (1) « {2,5,3}. Let Q « {4}. Let *I* » 2; go to Step (3).
(3)	For each i € {2,5,3} apply test (2) setting T « {4} U {i}. All solutions are fathomed (see numbers in parenthesis in Figure 6). List (2) » 0. Go to Step (4).
(4)	Let *I* - 1. Remove {4} from Q (now Q « 0). Let k » 3. Let List (1) » {2,5}. Let Q « {3}. Let *I* * 2. Go to Step (3).
(3)	For each i e {2,5} apply test (20) setting T » {3} U {i}. All solutions are fathomed (see Figure 6). List (2) • 0. Go to Step (4).
(4)	Let *I* » 1. Remove {3} from Q. (now Q » 0). Let k - 5. Let List (1) = {2}. Let Q - {5}. Let *I* » 2; Go to Step (3).

Step	Calculations
(3)	For i e {2} apply test (20) setting T = {5} U {i}. The solution is fathomed (see Figure 6). List (2) « 0. Go to Step (4).
(4)	Let 4 - 1 . Go to Step (5).
(5)	Stop. Z^u * 1762 is optimal for warehouses {l,3}. An optimal solution is shown in Figure 5.

An optimal solution to the example is given in Figure 5 when warehouses 1 and 3 are opened. The greedy solution is optimal and is obtained at the circled vertex {1,3} in Figure 6. The upper bound on an optimal number of warehouses, k • 2, was obtained in the first application of Step (2) and thus none of level three warehouse combinations were examined. In total, 16 of the vertices in Figure 6 were generated. Six of the vertices on level 2 were fathomed using (20). The total number of transportation pivots required to solve the sample problem was 85.

7. Computational Results.

The cost operator algorithm of Section 5 was coded in FORTRAN IV and the runs were made on a DEC 20 time sharing system. Many of the problems were run at different times during the day and thus the execution times may vary up to ten or fifteen percent depending upon the computing load of the machine. The maximum time alloted for solving any problem was 500 seconds.

All of the problems are derived from the Kuehn and Hamburger data which was originally presented in [14]. Problem sets I through VII are taken from Akinc and Khumawala [1], and VIII and IX are taken from Ellwein and Gray [8].

These are single stage problems of the type PI. Problem sets X through XV are multistage problems which are solved here for the first time.

Since it is necessary to read several articles to determine how the problems I through VII were originally created, we will describe them here. The Kuehn and Hamburger data represents a multistage system with 3 factories, 24 potential warehouse sites, and 50 demand centers located in the continental United States. Problems I through IV were formed by considering only the first 15 potential warehouse sites (i.e., Atlanta, Boston, through New Orleans as in [14]). To obtain the shipping costs (c_{ij}) from these sites to the 50 demand centers multiply each distance by $.025/mile unit cost, which is the bulk shipping rate. The sixteenth warehouse is actually the factory at Indianapolis. We used the distance from Indianapolis to the 50 demand centers multiplied by $.0125/mile unit as its shipping cost. The factory at Indianapolis has the same capacity as the other warehouses and it has a zero fixed charge. The problems V through VII are set up in a similar fashion except that all 24 potential warehouse sites are used. Problems X through XV are multistage problems which are also derived from the Kuehn and Hamburger data. All three factory sites were used. The factories have capacities of 30,000 or 35,000 units. To derive the shipping costs, we multiplied the distances from each factory to each warehouse by $.0125/mile unit cost, and from each factory to each demand center by .0375/mile unit cost. To get the shipping costs from the warehouses to the demand centers, we multiplied the distance by $.0250/mile.

Table 1 contains a description of all of the test problems. Table 2 contains the computational results for the single stage problems and Table 3 for the multistage problems. The percent error formula used to evaluate the greedy solution was,

$$\frac{Z^G - Z^*}{Z^*} \times 100.$$

where Z^G is the greedy value and Z^* is the optimal value. Also included in Table 2 are the execution times for the same problems solved in [1] and [17]. For test purposes the problem sets I through XV are very interesting because they contain problems with a wide range of difficulty.

The results for the greedy solution were startling. The largest percent error incurred was 3.7 (see Table 2, IX), however in general the percent error was less than .5 percent and the greedy solution was an optimal solution in 32 out of 51 problems.

Another interesting statistic is the number of vertices required to find the optimal solution as compared to the total number of vertices searched. An optimal solution is usually located very early in the search process and most of the effort is typically spent verifying its optimality. This experience is similar to that found in solving other kinds of integer programming problems.

The set I , which represents the warehouses fixed open in the initialization phase of the cost operator algorithm, often accounts for as much as 85% of the total number of open warehouses in an optimal solution. The uncapacitated problems (sets IV, VII, XII, XV) were very easy to solve as expected. As far as problem difficulty is concerned, those problems for which an optimal number of warehouses is approximately one half of the total number of potential warehouse sites are usually the most difficult. This is probably due to the fact that the number of possible warehouse combinations, $\binom{m}{k}$, when k is even or $\binom{m}{k}$ when k is odd, is the maximum of the binomial coefficients $\binom{m}{j}$,

The CPU times quoted in Tables 2 were obtained by three different sets of authors and are difficult to compare due to the differences in computers and in programming efficiency. As far as machine speeds is concerned, the IBM 370/168 is the fastest, followed by the IBM 370/165 and then by the DEC-20. However, actual speed ratio factors for pairs of the computing machines are virtually impossible to find. It seems fair to state that the performance of the three codes shown in the table are not significantly different; we may say they represent the state of the art of computational results on these problems.

Table 3 exhibits computational results on multistage problems having 3 factories, 24 warehouses, and 50 demand centers. As can be noted, the computation times are relatively small and exhibit a relatively small variance for integer programming problems. We again found the performance of the greedy solution to be even better than for the single stage problems. This is perhaps due to the fact that the factories have a large total capacity, and they can ship to customers directly if many of the warehouses are closed down.

8. Conclusions.

We have presented a cost operator algorithm for solving multistage location-allocation problems which does not employ problem relaxations as do the other currently best approaches [1, 17]. Computational results indicate that this method is competitive with the others. The greedy solutions obtained by the method are usually extremely close to or are optimal. Also, because the method computes many near optimal solutions as it solves the problem, these near optimal solutions can be saved and printed out for use by a manager if he desires. Computational results on the solution of multistage location problems are presented here, but we have been unable to find other published results on such problems for comparison. The performance of our method on these problems, is encouraging.

We wish to thank Professor limit Akinc for supplying the Kuehn-Hamburger data used in this paper.

Table 1

Test Problems

Problem Set	# Problems	(q*txr)	Sd_j	A_i	S_i	f_i
I	4	0X16X50	58268	-	5000	7,500/12,500/ 17,500/25,000/
II	1	0X16X50	58268	-	10000	17,500
III	4	0X16X50	58268	-	15000	7,500/12,500/ 17,500/25,000/
IV	4	0X16X50	58268	-	58628	7,500/12,500/ 17,500/25,000/
V	4	0X25X50	58268	-	5000	7,500/12,500/ 17,500/25,000/
VI	4	0X25X50	58268	-	15000	7,500/12,500/ 17,500/25,000/
VII	4	0X25X50	58268	-	58628	7,500/12,500/ 17,500/25,000/
VIII*	1	0X15X45	37440	-	1000' 5000/	15,0001 40,000/
IX*	1	0X15X45	37440	-	1500' 7500/	22,5001 60,000/
X	4	3X24X50	58268	35000	5000	7,500/12,500/ 17,500/25,000/
XI	4	3X24X50	58268	35000	15000	7,500/12,500/ 17,500/25,000/
XII	4	3X24X50	58268	35000	58628	7,500/12,500/ 17,500/25,000/
XIII	4	3X24X50	58268	30000	5000	7,500/12,500/ 17,500/25,000/
XIV	4	3X24X50	58268	30000	15000	7,500/12,500/ 17,500/25,000/
XV	4	3X24X50	58268	30000	58268	7,500/12,500/ 17,500/25,000/

* Numbers in the brackets r correspond to ranges.

Table 2

Computational Results

Problem Set	% error Greedy	# Vertices to find Optimal	Total # Vertices	$\lvert T^* \rvert$	Total # Warehouse	Total # Pivots	N & T (DEC-20)	A & K (IBM 370/165)	Nauss (IBM 370/168)
I-1	0	25	25	11	13	163	1.61	10.21	7.6
I-2	0	21	21	11	12	152	1.43	9.15	1.5
I-3	0	21	21	11	12	156	1.46	9.26	6.8
I-4	0	21	21	11	12	156	1.45	9.58	10.8
II	.19	58	147	5	8	1712	6.16	19.68	6.6
III-1	0	28	32	9	11	158	1.02	.23	1.5
III-2	0	30	37	7	9	217	1.19	.43	1.7
III-3	0	51	102	4	7	858	3.37	38.65	5.3
III-4	0	41	158	3	5	2582	8.36	34.39	2.9
IV-1	0	28	31	9	11	115	.96	47.5	-
IV-2	0	27	31	7	9	149	1.03	.44	-
IV-3	0	38	41	3	5	218	1.2	.30	-
IV-4	0	30	30	3	4	133	.85	.15	-
V-1	.47	254	1303	11	17	11700	56.7	.23	18.4
V-2	.72	128	6278	9	14	96462	422.8	120*	18.1
V-3	-	-	-	-	-	-	500*	120*	9.1
V-4	-	-	-	-	-	-	500*	120*	73.4
VI-1	.10	67	360	10	15	1417	10.35	.75	2.2
VI-2	0	78	908	6	11	6952	31.8	7.75	9.2
VI-3	.19	982	2172	5	8	19420	79.9	120*	18.1
VI-4	.15	1299	16367	2	7	146900	495.2	120*	27.2
VII-1	0	62	238	10	14	978	5.5	.68	-
VII-2	0	69	408	6	11	2408	9.87	1.65	-
VII-3	.20	104	264	4	8	1922	6.72	1.34	-
VII-4	.06	75	161	2	4	1088	4.05	.46	-
VIII	0	60	804	5	11	6247	29.6	12.55	1.8
IX	3.7	92	1304	2	7	9367	41.6	4.17	8.1

* Computation terminated

\- Not available

Table 3

Computational Results

Problem Set	% error Greedy	# vertices to Optimal	Total # Vertices	ill	Optimal # Warehouses	Total # Pivots	CPU Time (Dec-20)
X-l	0	56	143	11	15	2414	11.88
X-2	.30	514	858	8	13	12287	61.39
X-3	.24	277	680	6	11	7411	39.15
X-4	1.6	492	3627	1	7	39084	197.24
XI-1	0	54	99	9	13	1708	8.51
XI-2	0	71	521	5	8	7895	36.09
XI-3	0	62	184	5	8	2830	13.69
XI-4	0	90	837	2	6	14523	66.88
XII-1	.02	66	76	9	12	2874	17.69
XII-2	0	69	352	5	8	6885	34.43
XII-3	0	75	452	4	8	8431	41.26
XII-4	0	81	499	3	6	10383	50.1
XIII-1	0	56	145	11	14	2412	12.16
XIII-2	.30	514	858	8	13	12144	62.97
XIII-3	.24	277	680	6	11	7840	45.44
XIII-4	1.7	492	3627	1	7	41054	246.83
XIV-1	0	54	100	9	13	2060	12.85
XIV-2	0	71	537	5	8	8480	44.7
XIV-3	0	62	184	5	8	3152	19.9
XIV-4	0	90	838	2	6	14463	75.7
XV-1	.01	73	84	9	12	3627	23.8
XV-2	0	69	363	5	8	7813	44.3
XV-3	0	75	454	4	8	9187	49.8
XV-4	0	81	499	2	6	11234	60.53

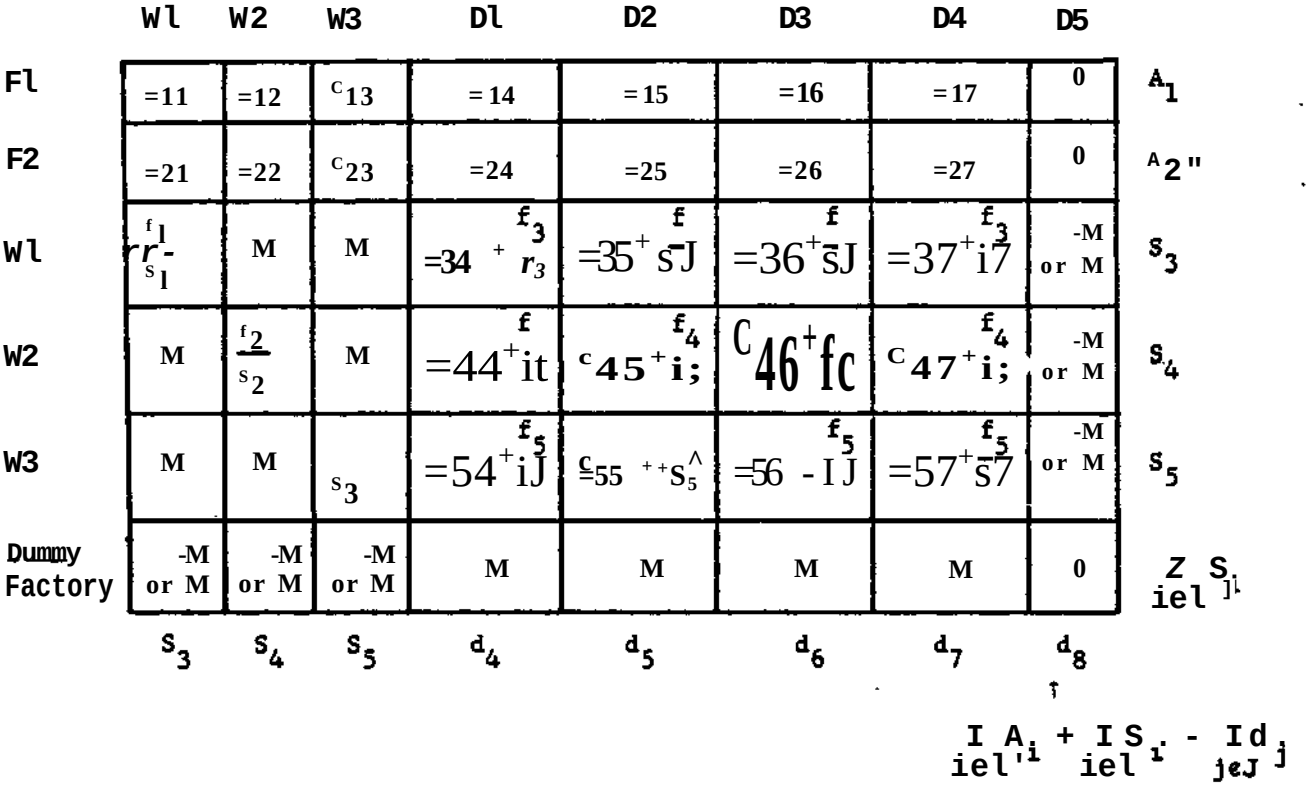
W1
W2
W3
D1
D2
D3
D4
D5
F1
F2
W1
W2
W3
Dummy Factory
=11
=12
C13
=14
=15
=16
=17
0
A1
=21
=22
C23
=24
=25
=26
=27
0
A2"
-M or M
S3
-M or M
S4
-M or M
S5
M
0
Z Sj
iel
S3
S4
S5
d4
d5
d6
d7
d8
I Ai + I Si - Idj
iel iel jeJ

Figure 1

	W1	W2	W3	D1	D2	D3	D4	D5	
F1	=11	=12	c_{13}	=14	=15	=16	=17	0	A_1
F2	=21	=22	c_{23}	=24	=25	=26	=27	0	A_2
W1	$\frac{f_3}{s_3}$	M	M	=34 + $\bar{s}$J	=35+i^	=36 + $\bar{s}$^	=37+ $\bar{s}$^	-M S_3	S_3
W2	M	$\frac{f_4}{s_4}$	M	=44+$\bar{s}$^	=45+$\bar{s}$^	=46+$\bar{s}_2$	=47+ $\bar{s}$f	M	S_4
W3	M	M	$\frac{f_5}{s_5}$	=54+i;	=55+s;	=56+r_3	=57+ i j	M	S_5
Dummy Factory	&	M	M	M	M	M	M	0	iel 3i
	S_3	S_4	S_5	d_4	d_5	d_6	d_7	d_8	

Figure 2

	W1	W2	W3	D1	D2	D3	D4	D5	
F1	=11	=12	=13	=14	=15	=16	=17	0	A_1
F2	c_{21}	=22	=23	=24	=25	=26	=27	0	A_2
W1	$\frac{f_3}{s_3}$	M	M	=34+ i;	=35+?;	C + i =36 + S_3	=37 + $\bar{s}$^	M	S_3
W2	M	$\frac{f_4}{s_4}$	M	c_{44} + $r_{\bar{S}_4}$	=45 + $\bar{S}$£	46 S_A	=47 + $\bar{z}$£	M	S_4
W3	M	M	$\frac{f_5}{s_5}$	c_{54}+$\bar{S}$j	=55 + $\bar{s}$;	c_{56}+i7	=57+i7	M	S_5
Dummy Factory	M	M	M	M	M	M	M	0	IS i iel
	S_3	S_4	S_5	d_4	d_5	d_6	d_7	d_8	

Figure 3

	W1	W2	W3	W4	W5	D1	D2	D3	D4	D5	
F1	8	5	7	5	9	37	28	33	26	0	61
F2	6	6	6	5	9	22	27	36	28	0	79
W1	6	M	M	M	M	17 + 6	16 + 6	10 + 6	16 + 6	X	25
W2	M	8	M	M	M	18 + 7	14 + 7	17 + 7	14 + 7	X	31
W3	M	M	5	M	M	14 + 5	13 + 5	19 + 5	11 + 5	X	40
W4	M	M	M	11	M	13 + 11	12 + 11	11 + 11	13 + 11	X	24
W5	M	M	M	M	5	14 + 5	13 + 5	14 + 5	15 + 5	X	28
Dummy Factory	X	X	X	X	X	X	X	X	X	0	148
	25	31	40	24	28	16	22	21	18	211	

Figure 4

	W1	W2	W3	W4	W5	D1	D2	D3	D4	D5	
F1	8	5	7	5	9	37	28	33	26	(0)[61]	61
F2	(6)[21]	6	G°	5	9	(22)[16]	27	36	28	(0)[2]	79
W1	(6)[4]	M	M	M	M	23	22	(16)[21]	22	M 31	25
W2	M	8	M	M	M	26	22	25	22	0	31
W3	M	M	5	M	M	(19)[0]	(18)[22]	24	(16)[18]	M	40
W4	M	M	M	11	M	24	23	22	24	(-M)[24]	24
W5	M	M	M	M	5	19	18	19	20	(-M)[28]	28
Dutnny Factory	M	(-M)[31]	M	(-M)[24]	(-M)[28]	M	M	M	M	(0)[65]	148
	25	31	40	24	28	16	22	21	18	211	

Figure 5

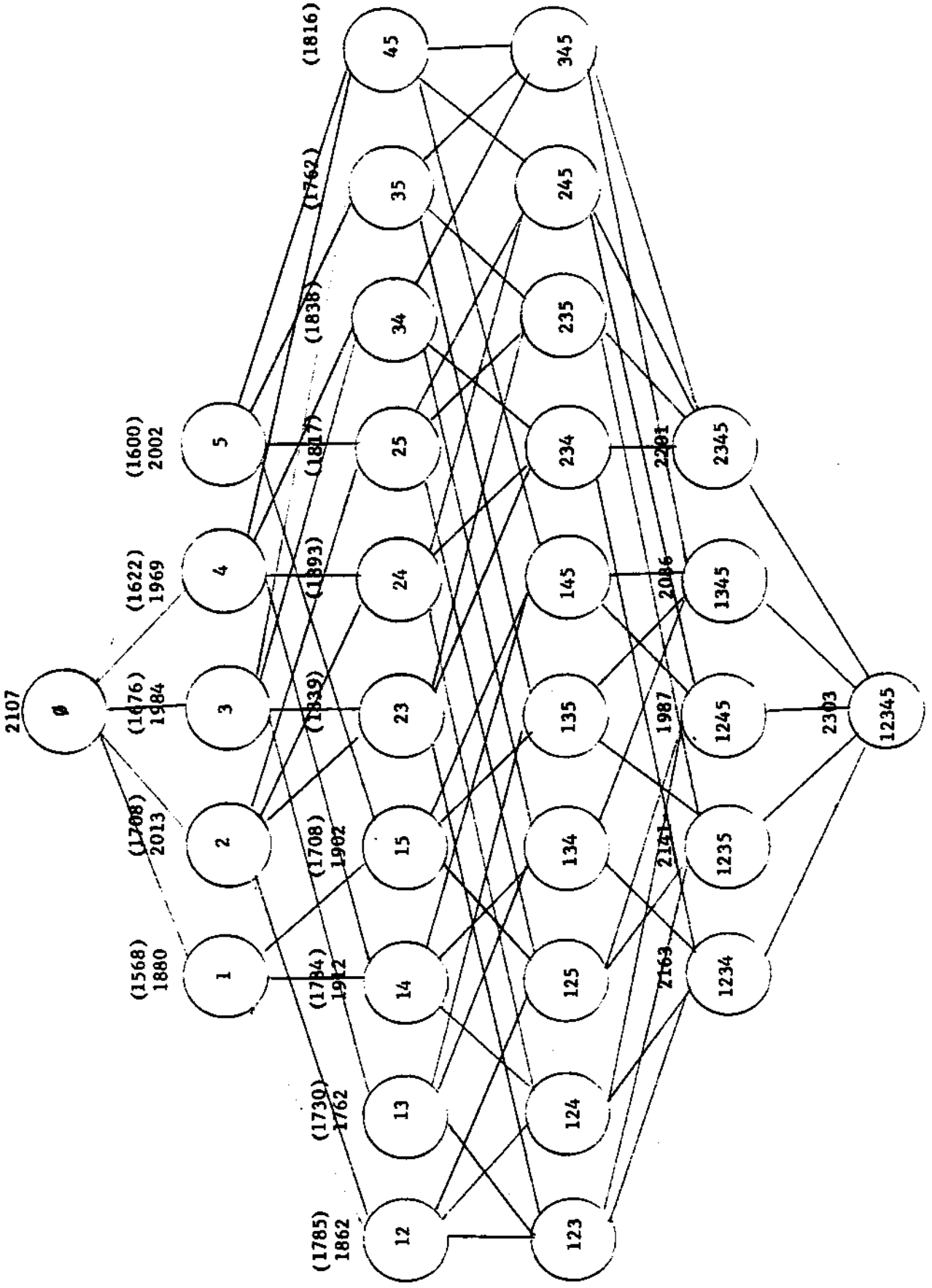
2107
∅
(1476) 1984
3
(1622) 1969
4
(1600) 2002
5
(1568) 1880
1
(1708) 2013
2
(1730) 1762
13
(1785) 1862
12
(1784) 1912
14
(1708) 1902
15
(1839)
23
(1893)
24
(1817)
25
(1838)
34
(1762)
35
(1816)
45
1987
1245
2036
1345
2201
2345
2141
1235
2163
1234
2303
12345
135
145
234
235
245
345
125
134
124
123

Figure 6

References

1. Akinc,U., and B. M. Khumawala, "A Branch and Bound Algorithm for Capacitated Warehouse Location," Management Science, Vol. 23, 6, pp. 585-594, 1977.

2. Babayev, D.A., "Comments on the note of Frieze," Math Programing 7, pp- 249-252, 1974.

3. Bilde, 0., and J. Krarup, "Sharp Lover Bounds and Efficient Algorithms for the Simple Plant Location Problem," Ann. Discrete Math, pp. 79-88, 1977.

4. Cornuejols, G., "Analysis of Algorithms for a Class of Location Problems," Technical Report No. 382, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York, July 1978.

5. Corneujols, G-, M. L. Fisher and G. L. Nemhauser, "Location of Bank Accounts to Optimize Float: An Analytic Study and Approximate Algorithms," Management Science, 23, pp. 789-810, 1977.

6. Davis, P. S., and Ray, T. L., "A Branch and Bound Algorithm for the Capacitated Facilities Location Problem," NRLQ, 16, 3, 331-343, 1969.

7* Efroymson, M. A., and Ray, T. L., "A Branch and Bound Algorithm for Plant Location," Operations Research, 14, 3, pp. 361-368, 1966.

8. Ellvein, L. B. and P. Gray, "Solving Fixed Charge Location-Allocation Problems with Capacity and Configuration Constraints," AIIE Transactions, 3: 290-299, 1971.

9. Erlenkotter, Donald, "A Dual Based Procedure for Uncapacitated Facility Location," Operations Research, Vol. 26, No. 6, pp. 992-1009, 1978.

10. Frieze, A. M., "A Cost Function Property for Plant Location Problems," Math Programming 1, pp. 127-136, 1971.

11. Geoffrion, A., and R. Mcbride, "Lagrangean Relaxation Applied to Capacitated Facility Location Problems," AIIE Transactions, Vol. 10, No. 1, pp. 40-47, 1977.

12. Geoffrion, A. M. and G. W. Graves, "Multicommodity Distribution System Design by Benders Decomposition." Math Prog. Study 2, pp. 82-114, 1974.

13. Guignard, M., and K. Spielberg, "A Dual Method for the Mixed Plant Location Problem," Tech. Report No. 20, The Wharton School, Univ. of Pennsylvania, Philadelphia, PA 19104, January 1977.

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

15. Melnikov, D. I., "The Method of Successive Calculations and the Plant Location Problem with Nonfixed Demands," Work on Discrete Mathematics, (Moscow, "Science" publishers, Moscow, 1973).

16. Narula, S. C., U. I. Ogbu, and H. M. Sammuelson, "An Algorithm for the p-Median Problem," Operations Research, 25, pp. 709-713, 1977.

17. Nauss, Robert M., "An Improved Algorithm for the Capacitated Facility Location Problem," J. Opl. Res. Soc., Vol. 29, 12, pp. 1195-1201, (1978).

18. Nemhauser, G. L., L. A. Wolsey and M. L. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions - I," Math Programming, Vol. 14, No. 3, pp. 265-294, 1978.

19. Sa, 6., "Branch and Bound and Approximate Solutions to the Capacitated Plant Location Problem," Operations Research, 17, 6, pp. 1005-1016, 1969.

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

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

22. Srinivasan, V. and G. L. Thompson, "An Operator Theory of Parametric Programming for the Transportation Problem" - I and II, NRLQ, 19, pp. 205-252, 1972.