MANAGEMENT SCIENCE
Vol. 18, No. 12, August, 1972
*Printed in U.S.A.*

# AN EFFICIENT BRANCH AND BOUND ALGORITHM FOR THE WAREHOUSE LOCATION PROBLEM*†

BASHEER M. KHUMAWALA‡

*University of North Carolina, Chapel Hill*

This paper introduces an efficient branch and bound algorithm for a special class of mixed integer programming problems called the warehouse location problem. A set of branching decision rules is proposed for selecting warehouses to be constrained open and closed from any node of the branch and bound tree. These rules are tested for their efficiency in reducing computation times and storage requirements to reach optimal solutions. An improved method of solving the linear programming problems at the nodes which substantially reduces the computations is also introduced in this paper.

## 1. Introduction

In the literature [4], [10] the warehouse (plant) location problem in its simplest form has been formulated as a mixed integer program as follows:[1] with $m$ potential warehouses (with unlimited capacity) and $n$ customers:

$$\text{Minimize } Z = \sum_{ij} C_{ij}X_{ij} + \sum_i F_iY_i$$

subject to

$$\sum_{i\in N_j} X_{ij} = 1, \qquad j = 1, 2 \cdots n,$$

$$0 \leqq \sum_{j\in P_i} X_{ij} \leqq n_iY_i, \qquad i = 1, 2 \cdots m,$$

$$Y_i = 0 \quad \text{or} \quad 1 \text{ (integer)}, \qquad i = 1, 2 \cdots m,$$

where $C_{ij} = t_{ij}D_j$.

$t_{ij}$ = the per unit cost which includes the FOB cost at the warehouse ($i$), the warehouse handling cost and the transportation cost from the warehouse to the customer ($j$),

$D_j$ = the demand at customer $j$,

$X_{ij}$ = the portion of $D_j$ supplied from warehouse $i$,

$F_i$ = the fixed cost associated with warehouse $i$,

$N_j$ = set of warehouses which can supply[2] customer $j$,

$P_i$ = set of those customers that can be supplied by warehouse $i$,

$n_i$ = number of elements in $P_i$.

The main difficulty in this problem (which is essentially combinatorial in nature) stems from $m$, the number of potential warehouses from which an optimum subset of warehouses for use in the system is to be made. The solution methods are therefore characterized by a high degree of computer efficiency. Several heuristic methods have been developed to produce "good" solutions [3], [5], [7], [8]. The branch and bound (b & b) method was first used by Effroymson and Ray [4] to yield optimal solutions.

* Received January 1971; revised July 1971.

† Presented at the 38th National ORSA Meeting, Detroit, October 30, 1970.

‡ The author wishes to thank Professor D. Clay Whybark of Purdue University and the referees for their helpful suggestions and comments.

[1] With a few exceptions, the notations of [4] are used throughout this paper.

[2] If prohibitive routes exist, not all warehouses will be able to supply all customers.

However, experience with their algorithm shows that for practical problems, the computation times and storage requirements are relatively high [4, p. 367]. Subsequent algorithms have been proposed by Spielberg [10], who provides a wealth of computational experience. In [11], Spielberg investigates the efficiency of using a generalized search origin instead of the "natural" search origin in his algorithms, as a means of reducing computation times. Beale [2] proposes a "general strategy" for mixed integer programming problems of the warehouse location type, which involve the selection of a subset of elements from a finite set. For this general strategy to be effective, Beale points out that the structure of the problem, at hand, should allow for efficient evaluation of the objective function and a rapid update of the supplementary information when any element is either added or dropped from the subset.

The warehouse location problem has this property. It is by taking considerable advantage of this property that we have significantly improved the b & b algorithm of Effroymson and Ray [4]. Our algorithm finds optimal solutions to problems with 25 potential warehouses in less than 10 seconds on CDC 6500 (the times on IBM 360/65 were found to be almost the same). Specifically, three kinds of efficiencies have been developed and added to the b & b algorithm of [4]. Briefly, these are:

(1) At each successive stage, the b & b algorithm requires the selection of a "free"[3] warehouse, from the set of free warehouses at that stage, to be constrained open and closed. Formal rules for selecting the free warehouse have been developed and tested in this study. (These rules will be referred to as the branching decision rules.) The tests have shown the existence of a very efficient branching decision rule.

(2) At each step of the b & b algorithm, a linear program (LP), Problem I without the integer restrictions on the $y$'s, is to be solved. The method shown in this paper makes use of the information already available at that stage in such a way that optimal LP solutions are obtained very rapidly.

(3) Several improvements related to the computer programming of the b & b algorithm are proposed. The computer storage, as a result, is used very efficiently.

## 2. Application of the Branch and Bound Algorithm

Problem I is first solved as a linear program (without the integer restrictions on the $Y$'s). Let $Z_0$ be the solution to that problem. If all the $Y$'s are integer, then the problem is solved. If some $Y_k$ is fractional,[4] then: (a) the restriction $Y_k = 0$ is added to the problem and the problem is again solved. Let $Z_1$ be the solution; clearly $Z_1 \geqq Z_0$; (b) the restriction $Y_k = 1$ is added to the problem and the problem is again solved; also, $Z_2 \geqq Z_0$. Then $\bar{Z} = \min(Z_1, Z_2)$ is a new lower bound on $Z$. This procedure has resulted in the construction of a tree whose nodes are represented by the $Z$'s and the corresponding value of the fixed $Y$'s. If a node is reached where all the $Y$'s are integers the LP solution at this node forms an upper bound on $Z$. A node where all the $Y$'s are integers will be called a terminal node, as opposed to a nonterminal node, where at

[3] The terms "open", "closed" and "free" warehouse will respectively mean a potential warehouse ($i$) assigned to be used ($Y_i = 1$), assigned not to be used ($Y_i = 0$) and not yet assigned either open or closed.

[4] As stated here, the choice of fractional $Y$ from among the fractional $Y$'s is arbitrary. Effroymson and Ray [4] mention in passing that they select $Y_k$ such that warehouse $k$ can supply the largest demand. On the other hand, Atkins and Shriver [1] do not mention their rule, and state only: "the actual rules used for deciding which warehouse to open or close are not critical to understanding the procedure" [1, p. 75]. A major part of our research is in the investigation of different rules for selecting the warehouse to be constrained open and closed. These rules are discussed later in this paper.

least one $Y$ is fractional; the LP solution at a terminal node will be referred to as a terminal solution. Branching continues from nonterminal nodes, whose LP solutions are less than the current upper bound; i.e., a fractional $Y$ at such a nonterminal node is constrained 0 and 1 and the linear programs are solved at the two additional nodes. The b & b algorithm continues in this manner, updating the bounds at each stage. Of course, no branching takes place from an infeasible node, the node at which the LP solution is infeasible.[5] The algorithm stops when a nonterminal node, whose LP solution is less than the current upper bound, cannot be found. The current upper bound is then the optimal solution. The tree constructed in this manner will be called the b & b tree for the problem and its size will be measured by the number of nodes in the tree.

Because of the assumption of unlimited warehouse capacity, the optimal solutions to the LP's at any node can be very easily found. As shown in [4], define $K_0$, $K_1$ and $K_2$ as the sets of indices of warehouses that are fixed open, fixed closed and free at the node, then the optimal LP solution at the node is:

$$
\begin{aligned}
X_{ij} &= 1 \quad \text{if} \quad C_{ij} + g_i/n_i = \min_{k \in K_1 \cup K_2} [C_{kj} + g_k/n_k], \\
&= 0 \quad \text{otherwise}, \\
Y_i &= 0, && i \in K_0, \\
&= \textstyle\sum_{j \in P_i} X_{ij}/n_i, && i \in K_2, \\
&= 1, && i \in K_1,
\end{aligned}
\tag{A}
$$

where

$$
\begin{aligned}
g_k &= F_k, && k \in K_2, \\
&= 0, && k \in K_1.
\end{aligned}
$$

The use of certain simplifications at each node is also shown in [4]. These simplifications at times significantly reduce the number of possible branches which need be investigated, and thus reduce the size of the b & b tree. Our study makes considerable use of these simplifications, as such they are repeated here for convenience.

1. The first simplification determines a minimum bound for opening a warehouse. If this bound is positive the warehouse will be fixed open. Mathematically, this is:

For $i \in K_2$, $j \in P_i$, compute

$$
\begin{aligned}
\nabla_{ij} &= \min_{k \in N_j \cap (K_1 \cup K_2);\, k \neq i} [\max\,(C_{kj} - C_{ij}, 0)], \\
\Delta_i &= \textstyle\sum_{j \in P_i} \nabla_{ij} - F_i.
\end{aligned}
\tag{2.1}
$$

If $\Delta_i > 0$, then $Y_i = 1$ for all branches emanating from that node.[6]

The $\nabla_{ij}$ simply measures the minimum cost savings for customer $j$ that can be made if warehouse $i$ is opened, when considered over all nonclosed warehouses at that node. Clearly, if the sum of such minimum savings for warehouse $i$ over all customers that it can supply exceeds its fixed cost, $F_i$, it pays to always open warehouse $i$ at this node.

2. The second simplification provides a means of reducing $n_i$.

[5] This will occur when at least one customer's demand cannot be satisfied by the nonclosed warehouses (either fixed open or free) at the node.

[6] Of course if warehouse $i$ is the only nonclosed warehouse for some customer, then also $Y_i = 1$.

If for $i \in K_2$, $j \in P_i$

$$\text{Min}_{k \in K_1 \cap N_j} (C_{kj} - C_{ij}) < 0,$$

then $n_i$ is reduced by one. Of course, if the inequality holds for all $j \in P_i$, then $P_i = \varnothing$, $n_i = 0$ and $Y_i = 0$ for all branches emanating from that node.

Clearly, if an already open warehouse can supply a customer $j$ cheaper (in terms of lower variable costs) than any of the "free" warehouses at the node, then such a customer should be supplied by the open warehouse. Such a customer should therefore not be considered a potential customer of the free warehouses at the node.

3. The third simplification determines a maximum bound on the cost reduction for opening a warehouse. If this bound is negative, the warehouse will be fixed closed.

For $i \in K_2$, $j \in P_i$

$$\begin{aligned} \omega_{ij} &= \text{Min}_{k \in N_j \cap K_1} [\text{Max } (C_{kj} - C_{ij}, 0)], \\ \Omega_i &= \textstyle\sum_{j \in P_i} \omega_{ij} - F_i . \end{aligned} \tag{2.2}$$

If $\Omega_i < 0$, then $Y_i = 0$ for all branches emanating from that node. The $\omega_{ij}$ is similar to the $\nabla_{ij}$ of simplification one, except that here the comparisons are made only over all the fixed open warehouses; i.e. $\omega_{ij}$ is the minimum savings for supplying customer $j$ that can be made if warehouse $i$ was opened, when considered over all the fixed open warehouses at that node. Clearly if the sum of such savings for warehouse $i$ over all customers that it can supply fails to exceed its fixed cost $F_i$, then such a warehouse should be closed and eliminated from further consideration.

At every node, one cycles through these three simplifications until no further openings or closings of warehouses can be made.

In the following sections the efficiencies that we added to this algorithm are presented. The efficiencies are a result of exploiting the information already available at each node. They are quite simple but are very effective in reducing the size of the b & b tree and hence in reducing the computation time and computer storage requirements. A small illustrative problem is given to explain fully the entire algorithm. The computational results on problems found in literature are also given.

## 3. Efficient Method of Solving the LP Problem at Nodes

The solution to I as an LP problem given by (A) is straightforward, but it can be further simplified if we define $i_j$ as the value of $i$ that minimizes $C_{ij}$ over all $i$ in $N_j \cap (K_1 \cup K_2)$. It can then easily be seen that customer $j$ should certainly be supplied from $i_j$ if

$$\nabla_{i_j j} \geqq F_{i_j}/n_{i_j} \quad \text{if} \quad i_j \in K_2 ,$$

and

$$\nabla_{i_j j} > 0 \quad \text{if} \quad i_j \in K_1 .$$

Because the $\nabla_{ij}$'s are computed as part of the simplifications, parsimony in obtaining the optimal solutions to the LP problems is therefore achieved.

## 4. Branching Decision Rules

As seen earlier, the b & b method requires that a warehouse be selected from the set of free warehouses at the node from which further branching is to be done. The selected warehouse is constrained closed and open respectively to yield two additional nodes.

The selection of such a warehouse will be called the branching decision, and the rule used for this selection will be called the branching decision rule. The branch along which the selected warehouse is constrained closed will be called the closed branch, and along which it is constrained open will be called the open branch. Rather than arbitrarily selecting the warehouse to be constrained at each node, a judicious selection would hopefully reduce the size of the b & b tree and thus lead to considerable savings from a computational standpoint.

Eight branching decision rules[7] were developed and tested on problems found in the literature. The development of these rules is principally based on the information gained and stored when cycling through the simplifications. Extensive use is made of this information, while care is taken to keep any additional computations and/or storage requirements to a minimum. These rules and the rationale for their formulation are discussed next.

*Delta Rules*

In simplification one, $\Delta_i$'s given by (2.1) are computed for each free warehouse at every node. As noted, if $\Delta_i \geqq 0$, then the warehouse $i$ is fixed open for all branches emanating from the node. However, from the warehouses whose $\Delta$ are negative, those having large $\Delta$ values are likely to be open in the terminal solution reached from this node. On the other hand, those warehouses having small $\Delta$ values are likely to be closed in the terminal solution reached from this node. We form two branching decision rules based on $\Delta$'s:

(1) *Largest Delta Rule.* Select the free warehouse which has the largest $\Delta$ from the set of free warehouses at the node having negative $\Delta$.

(2) *Smallest Delta Rule.* Select the free warehouse which has the smallest $\Delta$ from the set of free warehouses at the node having negative $\Delta$.

*Omega Rules*

In simplification three, $\Omega_i$'s given by (2.2) are computed for each free warehouse at every node. As noted, if $\Omega_i \leqq 0$, then the warehouse $i$ is fixed closed for all branches emanating from that node. However, from the warehouses whose $\Omega$ are positive, those having large $\Omega$ values are likely to be open in the terminal solution reached from this node and vice versa. This, therefore, suggests two more branching decision rules:

(1) *Largest Omega Rule.* Select the free warehouse which has the largest $\Omega$ from the set of free warehouses at the node having positive $\Omega$.

(2) *Smallest Omega Rule.* Select the free warehouse which has the smallest $\Omega$ from the set of free warehouses at the node having positive $\Omega$.

*Y Rules*

The optimal LP solution at a node gives fractional values of $Y$ for free warehouses. A free warehouse whose $Y$ is close to one would more likely be open in the terminal solution reached from the node, than a warehouse whose $Y$ is less. Conversely the warehouse whose $Y$ is close to zero is likely to be closed in the terminal solution reached from the node. This leads to two branching decision rules based on the $Y$'s:

(1) *Largest Y Rule.* Select the free warehouse with the largest $Y$ from the set of free warehouses at the node having fractional $Y$.

[7] It is possible to develop several other branching decision rules; see [6] for a discussion of other rules.

(2) *Smallest Y Rule.* Select the free warehouse with the smallest $Y$ from the set of free warehouses at the node having fractional $Y$.

*Demand Rules*

The rationale here is that if a warehouse capable of supplying very large demand (the sum of the demands of customers which the warehouse can supply) is closed, this would possibly result in an infeasible node along the closed branch. If the closed branch does in fact generate an infeasible node, no further branching is necessary from such a node. Such a rule would therefore hopefully reduce the size of the b & b tree. The two branching decision rules based on demand considerations are:

(1) *Largest Demand Rule.*[8] Select the free warehouse which can supply the largest total demand from the free warehouses at the node.

(2) *Smallest Demand Rule.*[9] Select the free warehouse which can supply the smallest total demand from the free warehouses at the node.

## 5. The Computer Program for the Branch and Bound Algorithm

The b & b algorithm with the improved method of solving the LP problems at each node was programmed in Fortran IV. This program[10] contains the option of using any of the eight branching decision rules. The flow chart of Figure 1 describes the iterative procedure of the computer program. An important part of this procedure is "cycling through the simplifications" at each node. This is described by means of a separate flow chart given in Figure 2. In order to fully illustrate this procedure, a small illustrative warehouse location problem is given in the next section.

A major limitation of the b & b algorithm is the amount of computer storage required to store all the eligible nonterminal nodes and associated information.[11] However, it was found that these storage requirements can be reduced by judiciously deleting nodes no longer necessary for the algorithm. The storage used for these deleted nodes is effectively used over and over again for the new nodes that are generated as the algorithm proceeds.

As an example, suppose that at the eighth node, the LP solution is found infeasible; then all the information for node eight is no longer necessary. This fact is noted and when a new node is generated at some other part of the tree, that node is assigned the number eight. Nodes (terminal or nonterminal) for which solutions to the LP problems exceed the current upper bound are deleted from memory in a similar manner. This procedure eliminates the necessity of adding storage for every new node.

Other modifications were also made to increase the efficiency with which computer storage is used. For example, at node 50, a certain $Y_k$ is selected to be constrained 0 and 1 respectively. This creates two new nodes from node 50. After the branching decision is made, node number 50 and all its pertinent information will no longer be needed for

[8] Effroymson and Ray's code uses this rule [4, p. 364].

[9] Although there is no strong rationale for this rule, it is included here to maintain the consistency of the demand rules with the other rules.

[10] The computer program can be found in [6].

[11] The computer storage requirements can be reduced by using a different node selection rule, i.e. instead of branching next from the node with the least lower bound, one could branch from the node with the least number of free warehouses. The latter rule would minimize the total number of eligible nonterminal nodes that need to be stored at any time; of course, this would possibly enlarge the size of the b & b tree and thus increase total computation time. In this paper, we limit ourselves to the results obtained by using the least lower bound node from which to branch next.

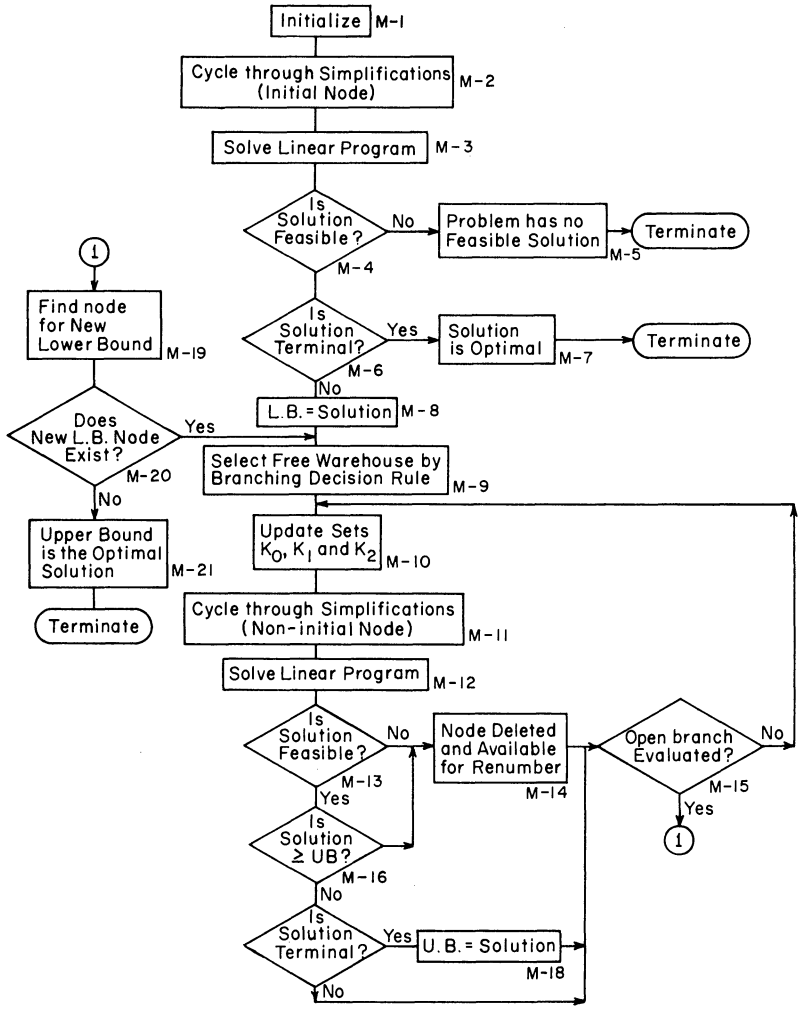


FIGURE 1. Branch and bound procedure-flow diagram.

the algorithm. Hence, instead of numbering the two nodes 51 and 52, they are numbered 50 and 51.

The result is that the assignment of a new number for any new node generated by the algorithm is made only when it is absolutely necessary. Anytime a new number is required, this is defined as a "distinct" node. In a particular warehouse location problem, this means the minimum number that was absolutely necessary to represent all nodes in the b & b tree for the problem.

## 6. Illustration

Consider the (5 × 8) example problem given in Table 1. For convenience, the demands of all the customers are assumed identical. This example is used for illustrative purposes only and is not intended to demonstrate the efficiency or inefficiency of the branching decision rules. These rules are tested on larger problems, which is discussed later. The important steps for solving this example problem are given along with the corresponding steps of the two flow-charts. The individual steps in Figures 1 and 2 are respectively marked with prefixes M and S.

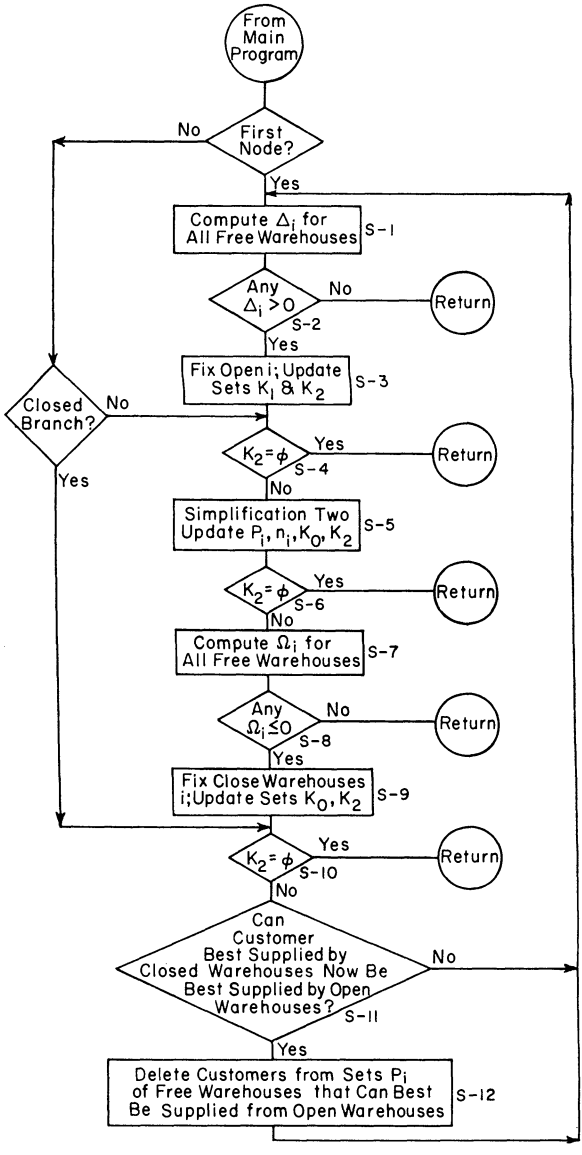


FIGURE 2. Simplification cycle-flow diagram.

| KEY | |
|---|---|
| □ | Terminal node |
| ○ | Node number |
| ( ) | Solution of linear program |
| $y_i$ | Selected warehouse $i$ constrained open |
| $\bar{y}_i$ | Selected warehouse $i$ constrained closed |
| $\| r+, S- \|$ | Due to simplifications, warehouse $r$ is fixed open and warehouse S is fixed closed. |

*Initialization* (M-1). At the first node, $K_0 = K_1 = \varnothing$, the empty set, and $K_2 = \{1, 2, 3, 4, 5\}$; the sets $P_i$ ($i = 1, 2, \cdots 5$) and $N_j$ ($j = 1, 2, \cdots 8$) are also initialized, e.g. $P_3 = \{1, 2, 3, 4, 5, 8\}$; $n_3 = 6$; $N_7 = \{1, 2, 4, 5\}$. The initial lower bound (LB) = 0 and upper bound (UB) = $+\infty$.

TABLE 1
*Illustrative Problem*

| Fixed Warehouse Costs $F_i$, $ | Warehouse | Customers ($Cij$) $ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 100 | 1 | 120 | 180 | 100 | L | 60 | L | 180 | L |
| 70 | 2 | 210 | L | 150 | 240 | 55 | 210 | 110 | 165 |
| 60 | 3 | 180 | 190 | 110 | 195 | 50 | L | L | 195 |
| 110 | 4 | 210 | 190 | 150 | 180 | 65 | 120 | 160 | 120 |
| 80 | 5 | 170 | 150 | 110 | 150 | 70 | 195 | 200 | L |
| Customer Demands (units) | | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |

**L** indicates a prohibitive route.

*Simplification Cycle* (M-2). The cycle is entered from step S-1 and the $\nabla_{ij}$ and $\Delta_i$ are computed, e.g. for warehouse 1, the nonzero $\nabla_{ij}$ are: $\nabla_{11} = 50$, $\nabla_{13} = 10$ and hence $\Delta_1 = -40$. Similarly, $\Delta_2 = -20$, $\Delta_3 = -55$, $\Delta_4 = 10$ and $\Delta_5 = -20$. Because $\Delta_4 > 0$, warehouse 4 is fixed open and $K_1 = \{4\}$ and $K_2 = \{1, 2, 3, 5\}$ according to steps S-2 and S-3. Also, customers 6 and 8, which are best supplied from open warehouse 4, are now eliminated from consideration of the free warehouses (step S-5). Thus, $P_2$, $P_3$ and $P$ are changed and $n_2 = n_3 = 5$; $n_1 = 6$. $P_1$ and $n_5$ remain unaltered.

The procedure continues with simplification three (step S-7) and computes $\omega_{ij}$ and $\Omega_i$, e.g. for warehouse 1, the nonzero $\omega_{ij}$ are: $\omega_{11} = 90$, $\omega_{12} = 10$, $\omega_{13} = 50$, $\omega_{15} = 5$ and hence $\Omega_1 = 55$. Similarly, $\Omega_2 = -10$, $\Omega_3 = 25$ and $\Omega_5 = 70$.

Because $\Omega_2 < 0$, warehouse 2 is fixed closed and $K_0 = \{2\}$ and $K_2 = \{1, 3, 5\}$ according to steps S-8 and S-9. $K_1$ remains unaltered.

At step S-11, we find that the customer 7, which was best supplied from warehouse 2, now can be best supplied by open warehouse 4. Hence, $P_1$ and $P_5$ are changed and $n_1 = 4$, $n_5 = 5$ (step S-12). $P_3$ and $n_3$ remain unaltered.[12]

The simplification cycle is continued by going back to simplification one again (step S-1) and we get:

$$\nabla_{11} = 50, \quad \nabla_{13} = 10 \quad \text{and hence} \quad \Delta_1 = -40,$$
$$\nabla_{35} = 5, \qquad\qquad\qquad\quad \Delta_3 = -55,$$
$$\nabla_{52} = 30, \quad \nabla_{54} = 30, \qquad\quad \Delta_5 = -20.$$

As all $\Delta_i < 0$, the procedure returns to the main program at step M-3.

*Linear Program Solution* (M-3). Customers 6, 7 and 8 are best supplied from open warehouse 4 as $\nabla_{46}$, $\nabla_{47}$ and $\nabla_{48}$ are positive. Hence, $X_{46} = X_{47} = X_{48} = 1$. For customers 1, 2 and 4 we find $\nabla_{11} > F_{1/n_1}$; $\nabla_{52} > F_{5/n_5}$ and $\nabla_{54} > F_{5/n_5}$, hence $X_{11} = X_{52} = X_{54} = 1$. For the remaining customers 3 and 5, we have $X_{33} = X_{35} = 1$ using (A). Thus, the LP solution at node 1 is:

$$X_{11} = X_{52} = X_{33} = X_{54} = X_{35} = X_{46} = X_{47} = X_{48} = 1 \qquad \text{(all other } X_{ij} = 0\text{)}$$

and $Y_2 = 0$, because $K_0 = \{2\}$; $Y_4 = 1$, because $K_1 = \{4\}$ and $Y_1 = \frac{1}{4}$; $Y_3 = \frac{2}{5}$; $Y_5 = \frac{2}{5}$, because $K_2 = \{1, 3, 5\}$ and $Z = 1171$.

This solution is feasible and nonterminal; hence the procedure continues. The lower bound (LB) is now 1171 (step M-8).

[12] In order to fully utilize the improved LP solution method, $\nabla_{47}$ is also made positive.

TABLE 2

*Information on Free Warehouses*

| Free Warehouse | $n_i$ | $Td_i$[13] | $Y_i$ | $\Delta_i$ | $\Omega_i$ |
|---|---|---|---|---|---|
| 1 | 4 | 40 | 0.25 | −40 | 55 |
| 3 | 5 | 50 | 0.4 | −55 | 25 |
| 5 | 5 | 50 | 0.4 | −20 | 70 |

TABLE 3

*Selection of Free Warehouses by Branching Decision Rules*

| Individual Branching Decision Rule | Free Warehouse Selected |
|---|---|
| Largest Delta | 5 |
| Largest Demand | 3 or 5 |
| Largest Omega | 5 |
| Largest $Y$ | 3 or 5 |
| Smallest Delta | 3 |
| Smallest Demand | 1 |
| Smallest Omega | 3 |
| Smallest $Y$ | 1 |

*Selection of the Free Warehouse* (M-9). The information available on all the free warehouses at the current node is given in Table 2.

Table 3 shows the free warehouse which would be selected from the node when different branching decision rules are used.

The selected free warehouse is first constrained closed and the node resulting from the closed branch is evaluated in the manner explained in the flow chart and illustrated for the first node. The selected warehouse is then constrained open and the same procedure is repeated.

Figure 3 shows the resulting b & b tree using the largest omega branching decision rule The free warehouse 5 was selected as shown in Table 3. At the node resulting from the closed branch (node two in Figure 3), no free warehouses could be fixed opened or fixed closed during the simplification cycle. The LP solution (nonterminal) found was 1209. On the other hand, at the node resulting from the open branch (node three in Figure 3), free warehouses one and three were fixed closed during the simplification cycle. Thus, node three is terminal and its LP solution (which is a terminal solution) is 1235. The upper bound is therefore revised (step M-18) and is now 1235.

The only nonterminal node existing at this stage is node two and its solution does not exceed the current upper bound. Thus, the solution at node two is the new lower bound; LB = 1209 and the new lower bound node is node two (step M-19). Further branching continues from this node, and the results are illustrated in Figure 3.

Note that the node labeled $A$ is also numbered two in the tree. This is because the information at the original node two is no longer needed. Its storage is therefore very effectively reused for the new node $A$. Both the new nodes (4 and 2) are terminal and their solutions exceed the current upper bound. At this stage, there is no nonterminal node and hence the optimal solution is 1235 and it occurs at node three.

[13] $TD_i$ refers to the total demand, which warehouse $i$ can supply at this node.

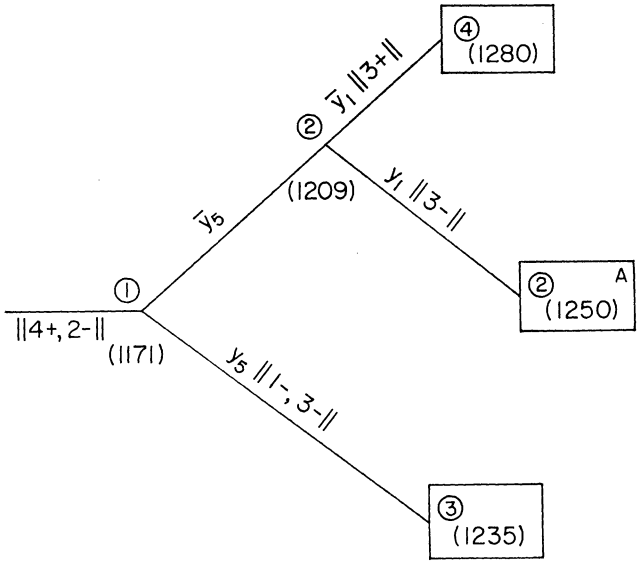


FIGURE 3. Branch and bound tree for illustrative problem.

## 7. Computational Results

In order to determine the relative effectiveness of the eight branching decision rules, 16 test problems of size 25 × 50, which have appeared in the literature [7], [9], were solved by each branching decision rule.[14] Because of excessive storage requirements in some cases, the program had to be terminated without completing the entire branch and bound tree. In these cases, the current upper bound (if any) at termination was recorded, which of course is not a guaranteed optimum solution. As a rule, the program was terminated anytime the number of distinct nodes reached 99. Because of the use of distinct nodes as opposed to the nodes in the usual sense, a separate count called iterations was made to account for the latter. Thus, the size of the b & b tree was measured by the total number of iterations. On the other hand, the maximum numbered distinct node represents the minimum number of nodes for which storage was required at any one time. The iteration at which the optimum solution was found was also recorded. This gives an indication of the proportion of the tree investigated for guaranteeing optimality.

The following efficiency criteria were established for comparing the branching decision rules:

(1) the total number of iterations which reflects the size of the branch and bound tree generated.

(2) the iterations number at which the optimum solution occurs; this reflects the speed with which the optimum solution is reached.

(3) the number of distinct nodes used which indicates the storage requirements for solving the problem.

(4) the computer time taken to solve the problem.

(5) the "quality" of the solution reached, as measured by its deviation from the optimum, in cases of incomplete runs.

[14] See Kuehn and Hamburger [7] for full details of these problems. Sã [9] used all the twelve problems of [7] and an additional set of four with a very slight modification (problem sets 1, 2, 3 and 7 [9, p. 1013]). We have used these same sets of problems and they are given in that order in Table 5.

TABLE 4

*Comparison of the Branching Decision Rules*

| bdr | ng opt | $n\bar{g}$ opt | n $\overline{opt}$ | nns | tot-it | | it opt | | ndn | | t | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | mean | max | mean | max | mean | max | mean | max |
| Largest Delta | 16 | — | — | — | 150 | 535 | 78 | 328 | 27 | 95 | 10.8 | 48.6 |
| Largest Demand | 12 | — | — | 4 | 98 | 457 | 66 | 308 | 23 | 73 | 7.3 | 30.8 |
| Largest Omega | 16 | — | — | — | 45 | 197 | 19 | 62 | 11 | 30 | 3.8 | 17.4 |
| Largest $Y$ | 16 | — | — | — | 95 | 369 | 62 | 266 | 24 | 86 | 7.2 | 27.8 |
| Smallest Delta | 15 | 1 | — | — | 107 | 383 | 59 | 238 | 29 | 99 | 9.3 | 32.3 |
| Smallest Demand | 12 | — | — | 4 | 98 | 457 | 66 | 308 | 23 | 73 | 7.3 | 30.8 |
| Smallest Omega | 8 | 3 | — | 5 | 104 | 256 | 63 | 188 | 38 | 99 | 8.1 | 24.3 |
| Smallest $Y$ | 14 | — | — | 2 | 104 | 413 | 63 | 280 | 23 | 87 | 9.4 | 37.4 |

*Note.* The computer runs which did not yield any solutions were not included in determining the means and the maximum values.

The results, which are given in detail in [6], are summarized here in Table 4. The following symbols are used in this table:

bdr.....the branching decision rule used,

ng opt.....the number of problems for which the optimum solution was found and proven optimum,

$n\bar{g}$ opt.....the number of problems for which the optimum solution was found but not proved optimum,

n $\overline{opt}$.....the number of problems for which a nonoptimum solution was found,

nns.....the number of problems for which no solution was found,

tot-it.....total number of iterations,

it opt.....iterations number at which optimum solution occurs,

ndn.....number of distinct nodes used,

t.....CPU time in seconds on CDC 6500,

nop.....number of warehouses open in the optimum solution (Table 5).

The most interesting result that can be seen from Table 4 is that, regardless of the branching decision rule employed, the computer times are very low. Of course, as was expected, some branching decision rules are found more efficient than others. The following observations are made on their performance:

(1) The two rules based on demand information give exactly identical results and are relatively poor compared to other rules. Recall that the rationale used in developing the largest demand rule was that when a warehouse with substantially large demand is fixed closed, the resulting node might be infeasible. As branches cannot emanate from an infeasible node, this phenomenon would therefore help reduce the size of the b & b tree. An explanation for the relatively poor performance of these rules in the test problems is that there are no prohibitive routes in the Kuehn and Hamburger data, and so infeasible nodes do not occur. These rules therefore may exhibit better performance in cases where the $C_{ij}$ matrices are sparse.

The explanation for the identical results from the application of the two demand rules is simple. Because the $C_{ij}$ matrix is complete in these problems, the number of customers that a free warehouse can supply at a node will be the same for all the free warehouses; and so will be the total demand that the free warehouses can supply. Thus, the smallest or the largest demand rules do not offer any differentiation between the free warehouses at the node in this case.

(2) The omega rules are based on the information of the maximum net gain of open-

ing the free warehouse over all the warehouses that are already open. The tests reveal that the performance of these rules is diametrically opposite. The smallest omega rule shows the poorest performance, whereas the largest omega rule shows by far the best performance of all the rules tested. In all but one problem, the largest omega rule was found superior (and significantly so in many cases) to the other rules when measured according to the efficiency criteria. It ranked a close second in the exceptional case. Not only are the computer times lower when the largest omega rule is applied, but it also requires considerably less computer storage and reaches the optimal solution at relatively early stages of its b & b tree.

(3) The distinction between the two $Y$ rules is not as pronounced. They are both inferior to the largest omega rule and, in general, the largest $Y$ rule seems preferable to the smallest $Y$ rule. The smallest $Y$ rule appears inferior also to the delta rules.

Recall that $Y_i$, obtained from the LP solution at a node, is given by $Y_i = \sum_{j \in P_i} X_{ij}/n_i$. It was demonstrated earlier in the discussion of the demand rules that the $n_i$ will be the same for all the free warehouses at the node for the test problems because the $C_{ij}$ matrix is complete. Thus, the performance of the $Y$ rules can also be expected to improve when they are applied to problems having sparse $C_{ij}$ matrices.

(4) It is difficult to distinguish between the two delta rules. In eight out of the sixteen test problems, the largest delta rule performed better than the smallest delta rule and the reverse was true in the remaining eight problems. It should be noted, however, that in the latter case, some of the differences in their performance were significant. This was not so in the former eight problems. It is interesting to note that the smallest delta rule outperformed the largest omega rule in the exceptional case.

Summarizing the comparisons of the branching decision rules, the objective of developing and testing these rules was to determine that rule which would yield optimal solutions to problems in minimum computer time and storage. The results indicate the largest omega rule to be the most efficient. It was pointed our earlier that the efficiency of the demand and $Y$ rules will improve in problems having sparse $C_{ij}$ matrices. However, it can be seen that the advantages gained in better discrimination between the free warehouses at a node when the $C_{ij}$ matrix is sparse are also gained by the $\Delta_i$ and $\Omega_i$. Hence, the improvements in efficiencies due to the sparse $C_{ij}$ matrix would be equally (or more) applicable also to the omega and the delta rules. The largest omega rule would therefore be expected to be the most efficient rule also in cases when the $C_{ij}$ matrices are sparse.

The branching decision rules were designed to make efficient use of the information available on the free warehouses at the nodes and care was taken not to incur any additional computations[15] or storage. Let us now look at how much information is used by the individual rules in selecting the free warehouse. The rules which would be based simply on $n_i$, the number of customers the free warehouse can supply, would use the least information on the free warehouses. Next in order of information usage are the demand rules which, in addition to the number of customers, also take into account the demands of these customers. The rules based on $Y$, delta and omega values, on the other hand, use even more information about the free warehouses than the demand rules. However, because of the varied information used by these three sets of rules, it is difficult to estimate which set of rules uses the most information. The relevant question between these three sets of rules is what information should be used in order to select

[15] The only additional computation needed in applying all the eight rules is determining the largest or the smallest values of the factors used as the basis for the rules.

TABLE 5

*Detailed Results of the Test Problems Using the Largest Omega Rule*

| | Number | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| nop | 15 | 11 | 8 | 4 | 14 | 10 | 9 | 8 | 13 | 8 | 8 | 5 | 15 | 10 | 8 | 6 |
| tot-it | 15 | 35 | 29 | 19 | 7 | 47 | 67 | 197 | 15 | 15 | 17 | 9 | 13 | 39 | 95 | 105 |
| it-opt | 14 | 4 | 12 | 4 | 4 | 30 | 46 | 62 | 14 | 2 | 6 | 2 | 8 | 24 | 40 | 36 |
| ndn | 6 | 9 | 7 | 5 | 4 | 14 | 18 | 30 | 6 | 6 | 5 | 4 | 6 | 10 | 20 | 22 |
| t | 1.61 | 3.43 | 2.61 | 1.56 | 0.88 | 4.37 | 5.86 | 17.38 | 1.50 | 1.36 | 1.66 | 0.85 | 1.55 | 3.87 | 6.39 | 6.80 |

the best free warehouse. The results demonstrate that the omega values of the free warehouses provide the most useful information for making this selection.

Our attention was drawn by a referee to the fact that the problems for which there are about eight warehouses open in the optimum solution seem to be most difficult; whereas, those with appreciably less than or greater than eight are relatively easier. In Table 5, we give the results of all 16 test problems when the largest omega branching decision rule was used. The results do indicate the existence of this particular phenomenon.

## 8. Conclusion

The efficiencies in the branch and bound algorithm introduced in this paper are derived from the structure of the uncapacitated warehouse location problem. These efficiencies have led to very promising results from both practical and theoretical considerations. The algorithm can now obtain global optimal solutions to fairly large size realistic warehouse locations problems in reasonable computing times and storage. The results indicate great promise for the algorithm and commend attempts to formulate other problems (the fixed charge problems, the knapsack problems and other zero-one type of problems) using this structure. As this is done, the applications and generality of the algorithm can be increased greatly.

### References


1. Atkins, R. J. and Shriver, R. H., "New Approach to Facilities Location," *Harvard Business Review* (May–June 1968).
2. Beale, E. M. L., "Selecting an Optimum Subset," Chapter 22 in *Integer and Nonlinear Programming*, Edited by J. Abadie, North-Holland, Amsterdam, 1970.
3. Drysdale, J. K. and Sandiford, P. J., "Heuristic Warehouse Location—A Case History Using a New Method," *Canadian Operational Research Journal*, Vol. 7 (1969).
4. Effroymson, M. A. and Ray, T. L., "A Branch-Bound Algorithm for Plant Location," *Operations Research*, Vol. 14 (May–June 1966).
5. Feldman, E., Lehrer, F. A. and Ray, T. L., "Warehouse Locations Under Continuous Economies of Scale," *Management Science*, Vol. 12 (May 1966).
6. Khumawala, B. M., "An Efficient Branch and Bound Algorithm for Warehouse Location," unpublished Ph.D. dissertation, Krannert Graduate School of Industrial Administration, Purdue University, June 1970.
7. Kuehn, A. A. and Hamburger, M. J., "A Heuristic Program for Locating Warehouses," *Management Science*, Vol. 9 (July 1963).
8. Manne, A. S., "Plant Location Under Economies of Scale—Decentralization and Computations," *Management Science*, Vol. 11 (November 1964).
9. Sã, G., "Branch-and-Bound and Approximate Solutions to the Capacitated Plant-Location Problem," *Operations Research*, Vol. 17 (November–December 1969).
10. Spielberg, K., "An Algorithm for the Simple Plant Location Problem with Some Side Conditions," *Operations Research*, Vol. 17 (January–February 1969).
11. ——, "Plant Location with Generalized Search Origin," *Management Science*, Vol. 16 (November 1969).