1970

Search algorithms for the simple plant location problem

John Bruce Prater

Follow this and additional works at: http://scholarsmine.mst.edu/doctoral_dissertations

Part of the Mathematics Commons

Department: Mathematics and Statistics

Recommended Citation

SEARCH ALGORITHMS FOR THE SIMPLE PLANT LOCATION PROBLEM

By

JOHN BRUCE PRATER, 1932

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MATHEMATICS

1970

Advisor

Ralph E. Lee

Co-Advisor

C. J. Ho
ABSTRACT

Two algorithms are developed, one exact, one approximate, for finding solutions to the simple plant location problem. Theorems are proved which give sufficient conditions for the inclusion of a plant in the optimal solution. The exact algorithm which is developed is similar to the Branch and Bound method. The approximate technique consists of a directed search through the solution tree for the problem, followed by terminal iterations. The terminal iterations are justified by empirical results obtained from a preliminary version of the technique and a theorem which is proved. Statistics from the results of applying the algorithm to a large number of problems are given. Listings of computer programs which are implementations of the algorithms are provided together with sample output from those programs.
ACKNOWLEDGEMENT

The author wishes to express his sincere appreciation to Dr. Billy E. Gillett, Chairman of the Department of Computer Science, for his help in the selection of this thesis subject and for his assistance, advice, and guidance in the work which led to the preparation of this dissertation.

The author would also like to thank Professor Ralph E. Lee and the staff of the University of Missouri-Rolla Computer Center for the use of their facilities and for their ever present willingness to help.

The author would like to extend a special expression of thanks to the National Science Foundation for their support of his efforts with a Science Faculty Fellowship.

To my wife and daughters goes the final and perhaps deepest expression of gratitude for their patience and understanding during the years of study.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS.</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. REVIEW OF LITERATURE.</td>
<td>3</td>
</tr>
<tr>
<td>III. FORMULATION OF THE PROBLEM.</td>
<td>19</td>
</tr>
<tr>
<td>IV. EXISTENCE OF A SOLUTION TO THE SIMPLE PLANT LOCATION PROBLEM</td>
<td>23</td>
</tr>
<tr>
<td>V. GRAPHIC REPRESENTATION OF ALL SOLUTIONS TO THE SIMPLE PLANT LOCATION PROBLEM</td>
<td>24</td>
</tr>
<tr>
<td>VI. PROOFS OF THEOREMS.</td>
<td>27</td>
</tr>
<tr>
<td>VII. DISCUSSION AND DEVELOPMENT OF ALGORITHMS.</td>
<td>35</td>
</tr>
<tr>
<td>A. An Exact Algorithm, PLANT</td>
<td>37</td>
</tr>
<tr>
<td>B. An Algorithm for Approximating a Solution, DETER</td>
<td>43</td>
</tr>
<tr>
<td>VIII. CONCLUSIONS AND DIRECTIONS FOR FURTHER WORK</td>
<td>54</td>
</tr>
<tr>
<td>A. Computational Results</td>
<td>54</td>
</tr>
<tr>
<td>B. Conclusions</td>
<td>58</td>
</tr>
<tr>
<td>C. Directions for Further Work</td>
<td>62</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>65</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td></td>
</tr>
<tr>
<td>General Warehouse Functions</td>
<td>67</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td></td>
</tr>
<tr>
<td>Program Listings and Sample Output.</td>
<td>70</td>
</tr>
<tr>
<td>APPENDIX C</td>
<td></td>
</tr>
<tr>
<td>Description of Computer Equipment at the University of Missouri-Rolla</td>
<td>84</td>
</tr>
<tr>
<td>VITA</td>
<td>85</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Solution Tree for Zero-one Problem with Three Variables</td>
<td>12</td>
</tr>
<tr>
<td>2. Cost Function Associated with Operating Plant i</td>
<td>20</td>
</tr>
<tr>
<td>3. Solution Tree for Four Potential Warehouse Sites</td>
<td>25</td>
</tr>
<tr>
<td>4. F Vector and A Matrix for the Simple Plant Location Problem</td>
<td>27</td>
</tr>
<tr>
<td>5. Graph of Values of the Objective Function in One Path Through the Solution Tree</td>
<td>29</td>
</tr>
<tr>
<td>6. Solution Tree for the Example</td>
<td>30</td>
</tr>
<tr>
<td>7. Nodes of the Solution Tree for the Example</td>
<td>32</td>
</tr>
<tr>
<td>8. Solution Tree for DETER</td>
<td>49</td>
</tr>
<tr>
<td>9. Linear Warehousing Cost Function</td>
<td>67</td>
</tr>
<tr>
<td>10. Piecewise Linear Warehousing Cost Function</td>
<td>68</td>
</tr>
<tr>
<td>11. Concave Warehousing Cost Function</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Summary of Problems Solved by Plant; 11 Warehouse Sites, 17 Customers.</td>
<td>41</td>
</tr>
<tr>
<td>II. Summary of Problems in Which DETER Did Not Give Optimal Solution</td>
<td>55</td>
</tr>
<tr>
<td>III. Comparison of Maximum Numbers of Nodes</td>
<td>56</td>
</tr>
<tr>
<td>IV. Representative Times for Solving the Simple Plant Location by Algorithm DETER</td>
<td>57</td>
</tr>
<tr>
<td>V. Summary of the Performance of the Algorithms</td>
<td>58</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

The problem of locating warehouse facilities in order to supply, at minimum cost to the manufacturer, a set of customers with known demands is one which frequently arises in industrial firms. Aside from the usefulness of its solution to many business firms, it is also a very interesting and difficult problem in a mathematical sense. The problem as it pertains to the firm may be stated as follows:

Given:

1. The location of each customer.
2. The demands of each customer.
3. The unit cost of shipment from each potential warehouse location to every customer.
4. A cost function associated with operating each warehouse over a fixed period of time.

Find:

1. The number of warehouses to be operated.
2. The location of each warehouse which is to be operated.
3. The capacity of each warehouse.
4. A warehouse to customer assignment.
5. The minimum cost of supplying the customers.

The mathematical problem is one of minimizing an objective function on a discrete domain, every point of the domain having as each of its coordinates an element from the set \( \{0,1\} \). The classical methods for finding
extreme points of a function on a continuous domain are thus not applicable to the problem being considered.

Since existence of a solution is guaranteed (see Section IV) and total enumeration will yield the optimal solution, the object of this dissertation is to develop an algorithm which yields a near optimal solution in an "acceptable" amount of time when programmed for a digital computer.

The development of the algorithm proceeds as follows:

1. Theorems are proved which give sufficient conditions for the inclusion of a warehouse in the optimal solution.

2. An algorithm is developed which uses the results of 1.

3. This algorithm yields empirical results which are used in altering the algorithm to an improved form.

4. The results of applying the algorithm to a large number of test cases are given.

The words 'plant location' and 'warehouse location' will be used interchangeably in this dissertation.
II. REVIEW OF LITERATURE

Since the method of solution of the plant location problem which was developed was a directed search with terminal iteration, special attention was given to those articles which employed a similar type of algorithm. A number of papers were read in which a different approach to the solution of the problem was used. Several papers were found in which the problem was similar but not identical to the plant location problem.

The approaches to solving the plant location problem may be divided into several classes which are not mutually exclusive. One such division is

1. Exact solution by classical methods.
2. Heuristic methods.
3. Branch and Bound methods.
5. Dynamic Programming.

Papers developing methods of each of these types were read.

Cooper [1,2] approaches the solution of the problem by using classical analysis and by heuristics. The classical investigation requires first the solution of the generalized Weber problem which may be stated as follows: Let the location of a set of $n$ known destinations by given by $(X_{D_j}, Y_{D_j}), j=1,\ldots,n$; the coordinates of the destinations
in two-dimensional Euclidean space. Let the coordinates of the source be \((X,Y)\). Let \(\beta_j\) \(j=1,\ldots,n\) be weights relating to amounts to be shipped. Then the problem of minimizing costs can be expressed:

Minimize the cost function

\[
\phi = \sum_{j=1}^{n} \beta_j \left[ \frac{(X_{Dj} - X)^2 + (Y_{Dj} - Y)^2}{\sqrt{(X_{Dj} - X)^2 + (Y_{Dj} - Y)^2}} \right]^{1/2}
\]

The necessary and sufficient conditions which the point \((X,Y)\) must satisfy to yield a minimum value for \(\phi\) are given as:

\[
\sum_{j=1}^{n} \beta_j \frac{(X_{Dj} - X)}{\sqrt{(X_{Dj} - X)^2 + (Y_{Dj} - Y)^2}} = 0
\]

\[
\sum_{j=1}^{n} \beta_j \frac{(Y_{Dj} - Y)}{\sqrt{(X_{Dj} - X)^2 + (Y_{Dj} - Y)^2}} = 0
\]

An iterative technique is then described for solving these equations.

When the idea is extended to the location of \(m\) sources instead of one, the General Location-Allocation problem results. If \((X_i,Y_i)\) \(i=1,\ldots,m\) are the coordinates of the \(m\) sources to be determined, the problem becomes in Cooper's formulation:

Minimize

\[
\phi = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_{ij} \psi(X_{Dj},Y_{Dj},X_i,Y_i)
\]

where \(\alpha_{ij}\) is zero or one and \(\psi(X_{Dj},Y_{Dj},X_i,Y_i)\) is the cost function for supplying the \(j^{th}\) destination from the \(i^{th}\) source. Cooper notes that in order to find the sets
\{(X_i, Y_i): \text{ } i=1, \ldots, m\} \text{ and } \{\alpha_{ij}\} \text{ that will minimize } \phi, \text{ the following conditions are necessary.}

\[ \sum_{j=1}^{n} \alpha_{ij} \frac{\partial \psi}{\partial X_i} (X_{Dj}, Y_{Dj}, X_i, Y_i) = 0 \]

\[ \sum_{j=1}^{n} \alpha_{ij} \frac{\partial \psi}{\partial Y_i} (X_{Dj}, Y_{Dj}, X_i, Y_i) = 0 \quad i = 1, 2, \ldots, m. \]

Solving these 2m equations will give values of \{(X_i, Y_i)\} \text{ } i=1, \ldots, m\} which give a minimum for \phi for a particular set of \alpha_{ij}. It can be noted that to determine the proper set \{\alpha_{ij}\} which will yield a minimum the system must be solved many times. The number of systems to be solved is, according to Cooper:

\[ \frac{1}{m!} \sum_{k=0}^{m} \binom{m}{k} (-1)^k (m-k)^n \]

where \( \binom{m}{k} \) denotes the binomial coefficients.

According to Cooper [2, p. 4] "For small problems this [method of solution] is feasible using a digital computer. For large-scale problems of industrial importance, the amount of computation is prohibitive."

Cooper then turns to heuristic methods of solution.

Four such methods are described briefly.

The Destination - subset method assumes that the number of sources, m, has been determined. Then the assumption is made that if all possible subsets of m of the total set of n destinations were considered, one of these
subsets would provide a close approximation to the optimal location of the sources. Computationally this is not as time consuming as calculating all possible allocations but for large \( n \) and \( m \) approximately equal to \( n/2 \), much computation will be necessary.

The Random-destination method due to Cooper is very interesting. The method consists of generating \( m \) uniform random numbers normalized to be integers between 1 and \( n \). This set of \( m \) integers determines at which destinations sources are to be located. Having determined the \( m \) sources, the allocation for each destination is chosen as the source for least cost for supplying that destination. The procedure is repeated as many times as desired, choosing the best among those solutions generated as a near optimal solution.

The Successive-approximations method begins by considering all possible locations of two plants. The best solution is chosen and a third source is considered for addition at each of the possible \( m-2 \) locations which were not used initially. The entire process is repeated until the number of sources equals \( m \). Again in this method the assumption is made that the number of plants, \( m \), in the optimal solution is known.

Alternate-location-and-allocation is an iterative method in which the set of \( n \) destinations is divided into \( m \) subsets. A subproblem associated with each of these
subproblems is solved as a generalized Weber problem. Each
destination is then examined to see if it can be more
economically supplied from one of the other m-1 sources.
The problems are then solved again after the source of
supply has been changed. The process is continued until
no further improvement is possible.

Timing information for problems of various sizes is
given in Cooper and a statistical analysis of the results
produced is also included. Surprisingly, Cooper concludes
that the random destination method is probably the best to
use in practice. This conclusion is further substantiated
by his statement that a problem which required three and
one-half hours of computer time when solved by the desti­
nation-subset method required only eight minutes of time
when solved by the random destination method. In the latter
case the approximate solution was only 1% higher than the
one obtained from the three and one-half hour calculation.

Kuehn and Hamburger[3] propose and develop a heuristic
method for determining a near optimal solution to the plant
location problem. Their method of solution consists of
two major parts: (1) the main program which locates ware­
houses one at a time until no warehouses can be added
without increasing total costs, and (2) their bump and shift
routine, which is entered after the main program is complete.
This routine attempts to modify the solutions obtained in
the main program. The three principal heuristics used in
the main program are:
1. Warehouse locations will be at or near concentrations of demand.

2. Near optimum warehousing systems can be developed by locating warehouses one at a time, adding at each stage that warehouse which decreases costs the most.

3. Only a small subset of all possible warehouse locations need be evaluated in detail at each stage to determine the next warehouse site to be added.

The bump and shift routine then attempts to modify the solution in two ways:

1. When a warehouse is added the most economical customer to warehouse assignments are determined and those warehouses which have lost customers may be eliminated (bumped).

2. The set of customers served by a warehouse determines a territory. After the partitioning of the customer set into territories by the main program, consideration is given to shifting each warehouse to every other city in its territory for the purpose of decreasing the cost.

The program was used to find approximate solutions to 12 sample problems each involving 24 warehouses and 50 customers. In four of the problems improvements were discovered for the approximate solutions found by the program.
Optimal solutions to the problems were not obtained so others of the eight remaining approximate solutions may not be optimal. The authors conclude that the heuristics for the design of the bump and shift routine may be changed in such a manner that the optimal solution will be found more frequently. However, evidence seems to indicate that a rather small improvement may be expected from any bump and shift routine and a limit might be set on the gain realized by this type of routine, the iteration terminating after this limit is reached.

Computation time for solving the 12 problems was 72 minutes on an IBM 650.

Another heuristic approach to the problem is given by Feldman, Lehrer and Ray[4]. This paper extends the Kuehn and Hamburger results in several directions. First, the method has been extended to handle concave warehouse costs instead of the fixed warehouse costs of the Kuehn and Hamburger method. Second, Kuehn and Hamburger use an adding heuristic assuming that the best N warehouse locations will contain the best N-1. Feldman et al. use both an add and a drop heuristic choosing the best answer produced by either of the two methods. The method initially proposed by Feldman et al. for handling convex curves did not perform adequately and the technique finally settled upon was the replacement of the convex curve by a series of
line segments (see Appendix A). The method was compared with the Kuehn and Hamburger algorithm by solving the large problem which was given in that paper. Solutions at least as good as those produced by the Kuehn and Hamburger method were reportedly obtained.

The general conclusions reached by the authors were that heuristic techniques can generate near optimal solutions to large scale plant location problems. Both sequential addition and sequential elimination of warehouse sites are useful in some cases and the problem itself will dictate when each is the more useful technique. A very important observation is that optimal patterns are very sensitive to the form of the convex warehouse cost curve and one should not oversimplify the curve during the problem formulation phase.

The method of Drysdale and Sandiford[5] is also a heuristic method and is similar to those of Kuehn and Hamburger and Feldman, Lehrer and Ray which have been discussed previously. It differs from both of those methods in some respects. The dropping heuristic of Feldman et al. is augmented by a heuristic involving stepwise incrementing of each fixed cost from zero to the true value associated with opening that warehouse. The method includes some searching at each stage similar to that described in Kuehn and Hamburger. The problem considered is rather different from the simple plant location problem in that costs for
shipping from plant to warehouse and from warehouse to customer are considered as well as the costs of supplying the set of customers from the warehouses chosen. The procedure is started by obtaining a solution in which no warehouse costs are included. The existence of each warehouse is then justified by determining if more is saved in transportation and inventory cost than is required to maintain the warehouse. The first warehouse is assumed to be justified and all tested against it. If none fail, the second is assumed to be justified and all others tested against it. If, at any stage, a warehouse fails the test, it is forced out of the solution and the procedure is restarted. If each warehouse is justified, the warehouse costs are incremented by an amount equal to approximately five per cent of the total warehouse cost. The procedure is then repeated until the warehouse costs reach the final values. The authors report results of a test case involving an actual production/distribution system for RCA Victor Company, Ltd. Their heuristic solution indicated that a saving of approximately 7 percent might be realized if the changes suggested by the program were made.

The Branch and Bound method of Land and Doig[6] has been the object of study by several authors seeking a solution to the plant location problem. The method is an example of a scheme called "implicit enumeration" which is
illustrated and discussed very thoroughly in Gue et al. [7] and Golomb and Baumert [8]. The set of all possible solutions to the plant location problem can be represented by a solution tree with a single root node, (see Figure 1).

Figure 1. Solution Tree for Zero-one Problem with Three Variables.

This node corresponds to the case in which all plants are open. As one progresses one level to the right in the tree, he finds nodes which represent the possible solutions to the problem with one plant closed, one more level to the right represents possible solutions each of which has two plants closed, etc. A fundamental property of implicit enumeration is that such a technique hopefully excludes large numbers of solutions from consideration by excluding branches of this solution tree.
Branch and bound as described by Efroymson and Ray[9] requires the solution of a sequence of linear programming problems that give improving bounds on the value of the solution to the problem. The problem is first solved as a linear program. If all the solution values are integers, the solution has been found. If the value of one of the variables is fractional, then it is set to zero and the resulting problem solved, then set to one and the problem solved again. Then the minimum of the two values is a new lower bound on the value of the solution. This process results in the construction of a tree of nodes corresponding to the solutions. The process terminates when a node is found for which all the variables are integer and for which the functional value is less than the value corresponding to any other node. One difficulty with branch and bound is that it may be necessary to solve the linear programming problem many times. The success of the technique is data dependent. Storage may also be a problem, since the values of the variables and the objective function must be retained for each terminal node in the tree. This is an exact method, however. The timing information given is quite encouraging with a number of 50 plant 200 customer problems requiring an average computer time of approximately 10 minutes on an IBM 7094.
Jones and Soland[10] also use a branch and bound algorithm for solving multi-level fixed-charge problems. The warehouse cost functions in this paper are discontinuous and the problem therefore differs from the problem being considered and the reference is included for the sake of completeness.

A number of papers have introduced methods for partial enumeration of the set of solutions of the zero-one programming problems by mixed-integer programming and integer linear programming techniques. (These are references 11 through 17 in the bibliography.) The methods presented in those papers were not directly applicable to the approach to the problem which I chose. Two papers, Balinski[16] and Gue, Liggett, and Cain[10] contain descriptions of a number of those algorithms.

A "one-point-move" search algorithm for exploring branches of the binary tree of solutions is the subject of the paper by Manne[18]. The algorithm begins by starting at one feasible solution to the problem which is chosen arbitrarily. (This corresponds to choosing a vertex of the binary tree.) The algorithm then calculates the value of the objective function for those combinations of plants which differ from the one chosen originally by closing one plant and opening another. The plant is chosen
which causes the greatest decrease in the objective function and the process is repeated. The process terminates when further search does not produce a decrease in the objective function. Manne then reports results obtained when using the process twice for each plant location problem, first starting the search from the point in the tree corresponding to all plants open and then starting with all plants closed. A sample consisting of 50 problems each of which had 10 potential warehouse sites, was analyzed by Manne using the algorithm and the results were compared to the optimum solutions. (The optimum was found by finding the minimum value of the objective function at all 255 nodes of the solution tree.) The solution of these problems required 27 minutes on an IBM 7090. The fixed warehouse cost was the same for all warehouse sites in this sample. A total of 1350 problems were then analyzed using the algorithm. The problems involved 6, 8, 10 warehouse sites. Seventeen problems were found in which the error was greater than 5%, the largest error being 15.94%. A complete statistical analysis of the results are given and are encouraging. In an appendix the author calculates analytically that in a certain case (an infinite number of customers and warehouse sites) the average error will be approximately 1.9% and the maximum possible error will be 6.1%. 
The articles by Lemke and Spielberg[19] and Spielberg[20] utilize a direct search technique which is similar to those described in the article by Manne and the article by Efroymson and Ray. The method is one of searching branches of the binary tree until one reaches the terminal node of a branch or is able to determine that further search in the branch will yield no smaller value of the objective function than that which has already been found. After this has been ascertained for a branch, the algorithm "backtracks" to a branch which has not been searched. The methods of excluding branches from the search differ in that the method in [19] involves mixed-integer programming, while certain heuristics as well as mixed-integer programming is used in [20]. The methods in [19] are carefully developed and the algorithms are outlined in some detail. They will not be given here as they refer to other authors' techniques and are very lengthy. The algorithms were compared using problems ranging from four warehouse sites and 20 customers to 31 warehouse sites and 31 customers. The times required ranged from one second on an IBM System 360/40 for the first problem to 50 minutes on an IBM 7094 in the second case. (The experience of the authors is that the 7094 is 10 times as fast as the 360/40.)
In [20], Spielberg gives a brief development of the method together with additional heuristics designed to eliminate branches of the solution tree. He also considers more constrained problems and suggests changes to the algorithm for handling these additional constraints. Termination of the algorithm is suggested when a certain upper bound is reached rather than use the amount of computer time necessary to iterate to the optimal solutions.

Timing information is not too useful since IBM 360/40, IBM 360/50 and IBM 7094 computers were used in the test runs, with no indication as to which machine was used on a particular run. The problems varied from 20 warehouse sites and 35 customers, with time 52 seconds to 90 warehouse sites and 100 customers with time 5106 seconds. The authors conclude that the results obtained provide a strong case for algorithms that adapt to the special type of data in the problem being solved. Their suggestion is that multiple algorithms be used, one to find a feasible starting solution than another to improve on this until optimality is reached.

The article by Curry and Skeith[21] considers the problem of allocating a set number, \( k \), of facilities in \( M \) locations and supply a product to \( N \) customers in such a was as to minimize total cost. The mathematical problem is decomposed into the recursive equations of dynamic programming and solved using that technique. The problem
considered in Curry and Skeith's paper differs from the simple plant location problem in that the number of facilities to be located is considered to be fixed. Another problem associated with the use of this technique is the amount of computer storage necessary for storage of tables. If auxiliary storage such as magnetic tape is used, the increase in the time required for solving the problem may be increased by a prohibitive amount.

The algorithm was programmed in FORTRAN IV and applied to a problem with five demands and three facilities to be allocated among four possible facility locations. The time required to compile the program and solve four problems similar to the one described was given as one minute on an IBM System 360/75 computer.

The authors concluded that the utility of this approach to the problem is the ease with which the nonlinear objective function, a nonlinear constraint, and the discrete domains can be handled. An increase in the number of facilities and locations has only an additive effect on the solution time as compared to the exponential effect for the total enumeration method.
III. FORMULATION OF THE PROBLEM

The simple plant location problem is the name given to the problem of minimizing the cost of producing and transporting the required quantities of a single product to a fixed set of customers, which may or may not have the same demand. The manufacturing process is to take place at a subset of a given set of possible plant locations. What must be determined is the number, location, and the capacity of the plants which will supply the set of customers at a minimum cost.

Let M denote the set of m possible plant locations and N denote the set of n customers, all of whose demands must be supplied. Suppose \( d_j \), \( 1 \leq j \leq n \) is the demand of customer \( j \). Let \( A \) denote the unit transportation matrix, i.e., \( a_{ij} \in A \) is the cost of producing and shipping one unit of the product from plant \( i \) to customer \( j \). A particular plant \( i \) \( (1 \leq i \leq m) \) may or may not be operating. In case the plant is not operating the cost \( Z_i \) associated with plant \( i \) is zero. If the plant is operating, the cost \( Z_i \) consists of a fixed cost \( f_i \), which can be thought of as the cost of building and maintaining the plant, and a piecewise linear concave function of the amount produced at plant \( i \) as illustrated in Figure 2.
Figure 2. Cost Function Associated with Operating Plant $i$.
Suppose that each plant cost is a constant function, that is $Z_i = f_i$. (The more general case represented in the figure will be discussed in Appendix A and a method will be given which permits representation of that problem in the simple case to which we are restricting ourselves.) Denote the amount shipped from plant $i$ to customer $j$ by $x_{ij}$ and assume that plant $i$ produces only the necessary quantity $\sum_{j=1}^{n} x_{ij}$. We impose the additional condition that any plant can, if necessary, produce all of the product required by all customers $\sum_{j=1}^{n} d_j$. The state of operating the plant $i$ can be represented by setting a variable $y_i$ to 1. If plant
i is not operating, $y_i$ will be set to 0. The formulation will thus be

Minimize

$$Z = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i$$

subject to the constraints

$$\sum_{i=1}^{m} x_{ij} = d_j \quad j=1, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} \leq y_i \sum_{j=1}^{n} d_j \quad i=1, \ldots, m$$

$$x_{ij} \geq 0, \quad i=1, \ldots, m; \quad j=1, \ldots, n$$

$$y_i \in \{0, 1\} \quad i=1, \ldots, m$$

The first set of constraints guarantees that only the proper number will be shipped to each customer. The second set of constraints provides that only those plants which are open will produce and ship the product. For a given vector $\hat{y} = (y_1, y_2, \ldots, y_m)$ the problem can be solved very easily.

The second part of the objective function is then a constant and the total cost is minimized if for every customer $j$, the supplying plant is chosen for which the unit cost is the smallest cost for supplying that customer from those plants which are open (i.e., for which $y_i=1$.)

The problem can now be expressed in a simpler form. Replace $x_{ij}$ by variables $w_{ij}$ which represent the fraction of $d_j$ supplied by source $i$, $w_{ij} = \frac{x_{ij}}{d_j}$. In the optimal
solution of the problem for a fixed \( \hat{y} \) the \( w_{ij} \) will also be 0, 1 variables. We must then replace the matrix of unit shipping costs \( A \) by the matrix \( C \) defined by \( c_{ij} = a_{ij} \cdot d_{j} \). With these changes the problem then can be expressed in the form.

Minimize

\[
   z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} w_{ij} + \sum_{i=1}^{m} f_{i} y_{i}
\]

subject to

\[
   \sum_{i=1}^{m} w_{ij} = 1 \quad j = 1, \ldots, n
\]

\[
   \sum_{j=1}^{n} w_{ij} \leq n y_{i} \quad i = 1, \ldots, m
\]

\[
   w_{ij} \in \{0, 1\} \quad i = 1, \ldots, m; \quad j = 1, \ldots, n
\]

\[
   y_{i} \in \{0, 1\} \quad i = 1, \ldots, m.
\]
IV. EXISTENCE OF A SOLUTION TO THE SIMPLE PLANT LOCATION PROBLEM

The existence of an optimal solution to the simple plant location problem is guaranteed since the problem is one of finding the minimum element of a finite set of real numbers. Each number in the set is obtained as a result of solving a zero-one integer programming problem of the type described in the section Formulation of the Problem. The cardinality of the set \( S \), whose minimum element we seek, is equal to the total number of ways in which non-null subsets of the given set of plants, \( M \), may be chosen. Since the cardinality of the set \( M \) is \( m \), therefore the cardinality of \( S \) is \( 2^m - 1 \).
V. GRAPHIC REPRESENTATION OF ALL SOLUTIONS TO THE SIMPLE PLANT LOCATION PROBLEM

The set of all solutions of the simple plant location problem can be represented in graphical form as a tree with each node corresponding to a solution of the problem with a certain set of plants (or warehouses) open. The tree has a single root node which corresponds to the case in which all plants are open. As one moves to the right in the tree, all nodes in the next level correspond to the solutions to the problem in which all plants but one are open. The next level represents those solutions in which all but two plants are open. The solution tree extends to the right until its branches terminate at the final level in a number of nodes equal to the number of potential plant sites. Each node in the final level corresponds to a solution of the problem with exactly one plant open. Figure 3 is the solution tree for the problem in which there are four potential plant sites. Associated with each node is a subset of the set of potential plant sites. This subset, which is represented in the form of a bit string in the figure, represents those plants which are open. For example, 1010 indicates that only plants 1 and 3 are open since ones appear in the first and third positions of the bit string and zeros appear elsewhere. For each node, a minimum value of the objective function is determined. The smallest of these values is the optimal solution.
Figure 3. Solution Tree for Four Potential Warehouse Sites.
The following remarks concerning the solution tree are elementary properties from combinatorial theory and proofs are not given.

Remark 1. If there are \( m \) potential warehouse sites, then the number of nodes in the tree is given by

\[
\sum_{i=1}^{m} \binom{m}{i} = 2^m - 1,
\]

where \( \binom{m}{i} = \binom{m}{i} \) are the binomial coefficients.

Remark 2. If there are \( m \) potential warehouse sites, then the number of branches in the tree is given by

\[
\sum_{i=2}^{m} i \cdot \binom{m}{i}
\]

Note that each node lies on more than one branch of the tree.

Remark 3. Adding one potential warehouse site to a given set of \( m \) warehouse sites increases the number of nodes in the tree by \( 2^m \).

Remark 4. If there are \( m \) potential warehouse sites, then the maximum number of nodes in any level of the solution tree is given by the binary coefficient.

\[
\binom{m}{k} = \frac{m!}{k!(m-k)!}
\]

where \( k = \lfloor \frac{m}{2} \rfloor \) is the greatest integer not greater than \( m/2 \).
VI. PROOFS OF THEOREMS

Let $M$ denote the set of $m$ potential warehouse locations and $N$ denote the set of $n$ customers, all of whose needs must be supplied. Consider the vector of fixed costs and the transportation matrix as described in the formulation of the problem. Associated with each set of warehouse locations chosen from the set of $m$ potential locations is a minimum cost associated with supplying the $n$ customers from warehouses operating at these chosen locations.

The cost of supplying the customers in the case in which all of the $m$ warehouses are open is given by the expression, (which shall be denoted by $\bar{Z}$):

$$
\bar{Z} = \sum_{i=1}^{m} f_i + \sum_{j=1}^{n} \min_{1 \leq i \leq m} \{a_{ij}\}
$$
Consideration of operating with only m-1 of the warehouses open yields a (possibly) different cost for each of the m cases which result. Let \( Z_k \) denote the cost of supplying the n customers from the set of warehouses \{1,2,\ldots,k-1, k+1,\ldots,m\}. Then the minimum cost of supplying this set of warehouses is given by:

\[
Z_k = \sum_{i=1}^{m} f_i + \sum_{i \neq k}^{n} \min_{1 \leq i < m} \{a_{ij}\} \quad k = 1, \ldots, m.
\]

**Theorem 1.** If \( Z_k > Z \) for any \( k \in \{1,2,\ldots,m\} \) then warehouse \( k \) will be included in the set of warehouses which yield the minimum cost of supplying the customers.

(Alternate Statement of Theorem 1). If \( Z_k > Z \) for any \( k \in \{1,2,\ldots,m\} \) then the minimum cost of supplying the customers will not be obtained by operating the set of warehouses \{1,2,\ldots,k-1,k+1,\ldots,m\} or any subset of these warehouses.

**Proof:**

Associated with each \( Z_k \) is a number

\[
d_k = \sum_{j=1}^{n} \left( \min_{1 \leq i < m} \{a_{ij}\} - \min_{1 \leq i < m} \{a_{ij}\} \right) - f_k
\]

(This may be thought of as the change in the objective function caused by closing warehouse \( k \).) By hypothesis \( d_k > 0 \); \( k=1,2,\ldots,m \). Assume the conclusion is false, i.e., assume that warehouse \( k \) is not in the set which yields the minimum solution. But by including warehouse \( k \), the cost is
decreased by at least an amount $d_k$. Therefore the set of warehouses $\{1,2,\ldots,k-1,k+1,\ldots,m\}$ did not yield a minimum value of the objective function and the assumption is false. Therefore the $k^{th}$ warehouse must be in the set.

Figure 5 illustrates what Theorem 1 states concerning the value of the objective function as we trace along a branch of the solution tree from the root node.

Figure 5. Graph of Values of the Objective Function in One Path Through the Solution Tree.

The theorem tells us that if the objective function continues to decrease as we move along a branch of the solution tree (as at points A, B, C, D), then the minimum of the objective function may be in this branch of the tree. Once the value of the objective function increases (as at E),
then no value of the objective function in the subtree
with root node at the node at which the increase occurred
will be a minimum value of the objective function. That is,
if there is an optimal solution in this branch, then it
will occur at node D. The following example illustrates
this point.

<table>
<thead>
<tr>
<th>WAREHOUSE i</th>
<th>( F_i )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>450</td>
<td>200</td>
<td>30</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>300</td>
<td>130</td>
<td>150</td>
<td>130</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>400</td>
<td>100</td>
<td>100</td>
<td>150</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>30</td>
<td>150</td>
<td>275</td>
<td>300</td>
</tr>
</tbody>
</table>

The solution tree for this problem is given in Figure 6.

Figure 6. Solution Tree For the Example.
The fact that the function increases as we move from node R to node Ll tells us that the minimum will not be found in the subtree with root node Ll. That is, in this case, the minimum of the objective function will not occur at nodes M1, M2, M3, N1, N2, N3. A similar result is true for the subtree with root node L4.

**Theorem 2.** If \( Z_k = \bar{Z} \), then there is some optimal solution to the problem which includes warehouse k.

**Proof:**

In this case

\[
d_k = \sum_{j=1}^{n} \left( \min_{1 \leq i \leq m} \{a_{ij}\} - \min_{1 \leq i \leq m} \{a_{ij}\} \right) - f_k = 0
\]

Assume that a certain set of plants not containing plant k yield a minimum value of the objective function. Then including warehouse k in the given set would increase the value of the objective function by an amount at most \( d_k (=0) \). Therefore the set of warehouses containing the k\textsuperscript{th} one would also yield a minimum value.

**Example:**

Consider the vector of fixed costs and the matrix of transportation costs.
<table>
<thead>
<tr>
<th>WAREHOUSE i</th>
<th>$f_i$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>25</td>
<td>20</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>30</td>
<td>10</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

The nodes of the solution tree for this problem are shown below with the minimum solution for that choice of locations.

![Figure 7. Nodes of the Solution Tree for the Example.](image-url)
Notice in this example that the value for all plants open (node 1111, value = 260) is the same as the value for node 0111, plant 1 being the one dropped. Theorem 2 states that there is a solution which does include plant 1. Here there are two. Notice further that from the one solution which does not include plant 1 we can form another by simply adding plant 1 to that solution.

**Theorem 3.** If application of theorem 1 to the simple plant location problem guarantees that warehouse k "must be in every optimal solution", then this warehouse will provide an absolute minimum for the cost of supplying at least one customer.

**Proof:**

By hypothesis, for plant k

\[ d_k = \sum_{j=1}^{n} (\min_{l<i<m, i \neq k} a_{lj}) - f_k > 0. \]

Then

\[ 0 < \sum_{j=1}^{n} (\min_{l<i<m, i \neq k} a_{lj}) - \min_{l<i<m, i \neq k} a_{lj} \]

The right side of the inequality represents the change in cost of supplying the customers when warehouse k is dropped from the set of open warehouses. The last inequality implies that there exists at least one customer for which the cost of supplying this customer from the set of warehouses \( \{1, 2, \ldots, k-1, k+1, \ldots, m\} \) is greater than the cost of
supplying this customer from the set of warehouses \( \{1,2,...,m\} \). Therefore warehouse \( k \) provides the absolute minimum cost for supplying this customer.
VII. DISCUSSION AND DEVELOPMENT OF ALGORITHMS

The techniques for solving combinatorial operations research problems may be categorized as follows:

1. Techniques which will find an optimal solution to the problem.

2. Techniques which generate an approximation to the optimal solution. (In many cases the technique will yield the optimal solution.)

Techniques of the first category usually require the use of a large amount of computer storage or consume large amounts of computer time in locating the optimal solution. The techniques of the second category, those which may only yield an approximation to the optimal solution, may be more conservative of computer time and the storage required than those of the first category. A disadvantage of the techniques in the second category is that the amount by which the approximation differs from the optimal value is not known.

Algorithms of both types were developed and are described in this dissertation.

The following definitions and discussion are presented at this point to clarify the development of the algorithms.

Definition: A node of the solution tree for the simple plant location problem is determined by the warehouses which are assumed open. Associated with each node of a solution tree is a value which is the minimum value of the objective function at that node.
Definition: The minimum value of the objective function at a given node is the minimum cost for supplying the needs of all customers from the warehouses which are assumed open at that node.

Example.

Suppose we have the following vector of fixed costs (f vector) and the matrix of warehouse to customer costs (A matrix).

\[
\begin{array}{c|cccc}
  f_i & i \backslash j & 1 & 2 & 3 \\
  \hline
  10 & 1 & 4 & 2 & 3 \\
  3 & 2 & 5 & 2 & 1 \\
  5 & 3 & 6 & 4 & 7 \\
\end{array}
\]

Consider the node 1, 2 (only warehouses 1 and 2 are open.) The minimum value of the objective function at this node is given by the expression

\[
f_1 + f_2 + \sum_{j=1}^{3} \min_{\{1,2\}} a_{ij}
\]

\[
= 10 + 3 + 4 + 2 + 1 = 20
\]

Note that warehouse 3 is not permitted to supply any customer at the node 1, 2 and the fixed cost of operating plant 3 is not summed.

Definition: The optimal value of the objective function is the smallest element of the set of minimum values of the objective function at all nodes in the solution tree.
Remark: In the above example, the optimal value of the objective function occurs at the node 2. The minimum value of the objective function at this node is 11.

A. An Exact Algorithm, PLANT

An algorithm was developed which will find all optimal solutions for the simple plant location problem. This algorithm was programmed in the PL/I computer language. The program listing together with sample output appears in Appendix B.

The algorithm is an implementation of a search method in which the minimum value of the objective function is found at certain nodes of the solution tree for the problem being considered. The search begins at the root node of the solution tree (the node which corresponds to all warehouses open) and continues through the tree until all optimal solutions of the problem have been found. Total enumeration is not practical in problems with a large number of warehouse sites, therefore, the determination of some method of restricting the search was necessary. The results of theorems 1 and 2 of Chapter VI are the means whereby subtrees of the solution tree are excluded from the search. This exclusion of subtrees reduces the number of nodes which must be examined in order to determine the optimal solution for the simple plant location problem. The search continues through the m levels of the solution
tree beginning with level m and exhausting all the nodes at each level before continuing to the next lower level. The collection of all nodes which correspond to exactly k open warehouses constitute what is called level k in the solution tree. Note that level k contains the number of nodes equal to the binary coefficient \( \binom{m}{k} \) where m is the number of potential warehouse sites. For each node in a particular level of the tree, exactly one of the following actions is taken:

1. If it has been determined that a given node is in an excluded subtree, then the minimum value of the objective function is not calculated at this node.

2. The minimum value of the objective function is calculated for the node and this value is compared with the minimum value of the objective function at all predecessor nodes in the previous level. Two possibilities then exist:
   a) The value of the objective function is less than the value at all predecessor nodes. In this case the search continues through branches of the tree which contain this node.
   b) The value of the objective function is greater than or equal to the value of the objective function for at least one of the predecessor nodes. Application of theorems 1 and 2 allow us to exclude the search from subtrees with this node as their root node.
This searching process continues through all nodes of one level before considering any nodes at the next lower level. The process continues until one of two possibilities occurs.

1. The final level of the solution tree is exhausted. (The level which corresponds to all solutions with only one plant open).

2. A sufficient number of subtrees have been excluded from the search to guarantee that an optimal solution does not exist for any nodes which remain in unsearched levels of the solution tree.

Upon realization of either of these conditions, the optimal value of the objective function has been found, that value being the minimum value of the objective function found in the search to this level in the solution tree.

This method bears some resemblance to the Branch and Bound method as used by Efrymson and Ray[9]. Both are search methods which will find an exact solution to the simple plant location problem. However, in the PLANT algorithm only values of the objective function from the preceding level of the solution tree are retained. In the Branch and Bound method it may be necessary to retain values from several levels of the solution tree.

To clarify the discussion which follows, the reader should refer to Chapter III, Formulation of the Problem. In the discussion which follows, the elements of the A matrix are the elements $a_{ij}$ whose values are the cost of
manufacturing and shipping from warehouse i all the product required by customer j. The elements of the vector $F$ are the fixed costs $f_i$ associated with operating warehouse i.

The procedure PLANT was used to solve example problems of various sizes in order to obtain empirical results relative to the time required to solve problems and the number of nodes at which it was necessary to find the minimum value of the objective function. Table I, which summarizes the problems solved for 11 warehouses and 17 customers, is representative of all examples solved and is the basis of the conclusions drawn concerning the PLANT algorithm.

The following points should be noted from the table:

1. The number of nodes at which it is necessary to determine the minimum value of the objective function is data dependent. In general, the smaller the fixed costs ($F$ vector) relative to the manufacture and transportation costs ($A$ matrix), the fewer times the objective function must be evaluated (i.e., fewer nodes.)

2. Doubling the number of nodes at which it is necessary to evaluate the objective function approximately triples the amount of time necessary to determine all optimal solutions using algorithm PLANT.
TABLE I

SUMMARY OF PROBLEMS SOLVED BY PLANT;
11 WAREHOUSE SITES, 17 CUSTOMERS

<table>
<thead>
<tr>
<th>A</th>
<th>RANGE</th>
<th>TIME</th>
<th>NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1414</td>
<td>100-200</td>
<td>0:14</td>
<td>132</td>
</tr>
<tr>
<td>0-1414</td>
<td>200-300</td>
<td>1:59</td>
<td>532</td>
</tr>
<tr>
<td>0-1414</td>
<td>300-400</td>
<td>2:46</td>
<td>652</td>
</tr>
<tr>
<td>0-1414</td>
<td>400-500</td>
<td>2:51</td>
<td>668</td>
</tr>
<tr>
<td>0-1414</td>
<td>500-600</td>
<td>6:40</td>
<td>1100</td>
</tr>
<tr>
<td>0-7300</td>
<td>100-200</td>
<td>0:37</td>
<td>259</td>
</tr>
<tr>
<td>0-7300</td>
<td>200-300</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-7300</td>
<td>300-400</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-7300</td>
<td>400-500</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-7300</td>
<td>500-600</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-7300</td>
<td>600-700</td>
<td>1:42</td>
<td>514</td>
</tr>
<tr>
<td>0-7300</td>
<td>700-800</td>
<td>1:42</td>
<td>514</td>
</tr>
<tr>
<td>0-7300</td>
<td>800-900</td>
<td>1:40</td>
<td>514</td>
</tr>
<tr>
<td>0-7300</td>
<td>900-1000</td>
<td>1:42</td>
<td>514</td>
</tr>
<tr>
<td>0-7000</td>
<td>100-200</td>
<td>1:40</td>
<td>514</td>
</tr>
<tr>
<td>0-7500</td>
<td>200-300</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-5000</td>
<td>300-400</td>
<td>1:40</td>
<td>514</td>
</tr>
<tr>
<td>0-7300</td>
<td>400-500</td>
<td>0:36</td>
<td>259</td>
</tr>
<tr>
<td>0-7900</td>
<td>500-600</td>
<td>1:40</td>
<td>514</td>
</tr>
<tr>
<td>0-6050</td>
<td>600-700</td>
<td>7:00</td>
<td>1154</td>
</tr>
<tr>
<td>10-16</td>
<td>100-200</td>
<td>18:22</td>
<td>2047</td>
</tr>
</tbody>
</table>

NOTE: The maximum number of nodes in the solution tree for 11 warehouses is $2^{11} - 1 = 2047$

* These examples had the same A matrix.
** These examples had the same A matrix.
The following facts also influence the conclusions concerning this algorithm.

1. Adding one warehouse site to a given set of \( m \) warehouse sites doubles the number of nodes in the solution tree. \( (2^{m+1} \text{ as compared to } 2^m). \)

2. The amount of storage required severely limits the maximum size problem which can be solved by algorithm PLANT. On the equipment available at the University of Missouri-Rolla the largest problem which can be solved is restricted to 14 warehouses. (See Appendix C for a description of this equipment.)

It is apparent from the table and the remarks which follow it that the usefulness of the algorithm PLANT is data dependent. The storage requirements further restrict the usefulness of the algorithm. The empirical results obtained indicate that the algorithm works best when the elements of the \( A \) matrix are large in comparison to the elements of the \( F \) vector. In this case the minimum value of the objective function must be found at a smaller percentage of the total number of nodes in the solution tree. Additional conclusions concerning this algorithm will be stated in Chapter VIII.
B. An Algorithm for Approximating a Solution, DETER

The limitations on the usefulness of the PLANT algorithm dictated the development of another algorithm which could be applied to the simple plant location problem to produce an approximation to the optimal value of the objective function. It was decided that the algorithm should satisfy the following conditions:

1. The computer time necessary for solving large problems (at least 50 potential warehouse locations) must be within "reasonable" bounds.

2. Core storage availability should not be a limitation. With the current computer equipment, problems with 50 potential warehouse sites and 500 customers can be handled easily. (For a description of the computer equipment see Appendix C.)

3. The algorithm must yield the optimal solution in a large percentage of problems and the approximate answer must be "near" the optimal value of the objective functions in those problems in which the optimal value is not found.

The third requirement was the most difficult to satisfy. In order to investigate the error in the answers produced by the approximation algorithm, it was necessary to find exact answers to the experimental problems considered.
Thus the size of the problems considered was restricted to those for which an optimal solution could be found by using PLANT. The conclusions were based on the results obtained from solving these experimental problems.

The experimental problems were created in two ways. First, an attempt was made to construct problems which would cause the algorithm to fail. The problems were then solved by the two algorithms and the answers were compared. Second, a section of code was written to generate random numbers as elements of the A and F matrices. As before the results produced by the two algorithms were tabulated. These tabular results are summarized in Tables II and V and will be discussed at length in the conclusions chapter.

The approximation algorithm was developed using the three theorems which were proved in the preceding chapter together with an empirical result obtained from an early version of the algorithm. Initially, a forward search through the solution tree closing a warehouse at each step was the only technique used in the algorithm. The search began at the root node. The branch taken from this node was the one in which the maximum decrease in the objective function was obtained by "closing" one warehouse. In the first step then it was necessary to obtain the minimum value of the objective function at each of the m nodes in level m-1 of the solution tree. The search was then restricted to the subtree with the node determined
as the root node. Again, in this step the branch taken was the one in which the maximum decrease in the objective function was realized. In the second step the minimum value of the objective function was found at m-1 nodes of the solution tree. This forward search was continued through the solution tree until:

1) The end (level l) of the solution tree was reached or
2) Application of theorem 1 guaranteed that no value of the objective function less than the smallest already found existed in the subtree to which our search had been restricted.

At this point the search was terminated and the smallest value of the objective function found was the approximation to the optimal solution. This forward search method yielded the optimal solution in 94.5 percent of the experimental problems run. While this was a fairly high percentage for locating the optimal solution, the goal was to find the optimal value in 99% of the cases. It was thus necessary to improve the algorithm.

One form of theorem 1 states a sufficient condition for a warehouse to be in the solution. The first step of the forward search method just described will yield a list of these warehouses. The technique of using a backward search through the solution tree starting at a selected node was then conceived with the search beginning at a node chosen as follows:
1) If the application of theorem 1 in the forward search yielded a list of warehouses which must be in the solution, then the search began at the node representing the case in which only those warehouses are open.

2) If the application of theorem 1 did not guarantee that certain plants would be in the solution, then the backward search began at the node which yielded the minimum cost solution with only one warehouse open.

The backward search was continued to the root node of the solution tree. The reason for continuing the search to this extent may be seen from Figure 5. Searching backward through the solution tree, the value of the objective function may increase at one node then decrease at a node closer to the root node.

Combining the forward and backward search methods yielded the optimal solution in approximately 99 percent of all experimental problems which were solved. The solutions of these sample problems possessed one very striking characteristic which led to a further improvement in the algorithm. In all but two solutions of a sample of 2000 problems, the number of warehouses which were open in the optimal solution differed from the number open in the solution found by the approximate algorithm by at most one warehouse. It was decided to attempt to improve the algorithm making use of this observation.
If we let \( k \) denote the number of warehouses in the approximate solution found by the combined search methods, then it would have been possible to calculate the minimum value of the objective function for all possible combinations with \( k-1, k, \) and \( k+1 \) warehouses open. This could require an unacceptable amount of computation since the total number of nodes at which the objective function must be calculated is \( \binom{m}{k-1} + \binom{m}{k} + \binom{m}{k+1} \).

Theorem 3 states that if a warehouse satisfies the sufficient condition for inclusion as a warehouse in the optimal solution then it provides the absolute minimum cost for at least one customer. This result was used as the source of direction for improving the algorithm. The algorithm was changed to find the minimum cost warehouse for supplying each customer. The warehouse to customer assignments for the approximate solution are then determined. For each customer this determines a potential trade of warehouses, the one found by the approximate solution and the minimum cost warehouse. The algorithm was altered to consider the change in the objective function when for each customer the warehouse supplying the customer in the approximate solution is replaced by the minimum cost warehouse. If the value of the objective function decreases, this improved value is taken as the approximate solution. All possible ways of replacing pairs of the plants are then considered.
The smallest value found by the combined forward search, backward search and terminal iterations is taken as the approximate solution.

The following example illustrates the way in which algorithm DETER arrives at an approximation to the optimal solution. The F and A matrices for the example follow.

<table>
<thead>
<tr>
<th>$f_i$</th>
<th>i</th>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>W A R E</td>
<td>136</td>
<td>1</td>
<td>551*</td>
<td>841</td>
<td>767*</td>
<td>1066</td>
<td>395</td>
</tr>
<tr>
<td>104</td>
<td>2</td>
<td>897</td>
<td>879</td>
<td>1085</td>
<td>704*</td>
<td>347</td>
<td></td>
</tr>
<tr>
<td>H O U S E</td>
<td>199</td>
<td>3</td>
<td>648</td>
<td>714*</td>
<td>849</td>
<td>720</td>
<td>93*</td>
</tr>
<tr>
<td>151</td>
<td>4</td>
<td>596</td>
<td>723</td>
<td>807</td>
<td>798</td>
<td>110</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>117</td>
<td>5</td>
<td>804</td>
<td>936</td>
<td>1019</td>
<td>939</td>
<td>307</td>
</tr>
</tbody>
</table>

The value $f_i$ represents the cost of operating warehouse $i$. No manufacturing or transportation costs are included in this value. The element in the $i^{th}$ row and $j^{th}$ column of the customer-warehouse matrix is the cost of manufacturing and transporting from warehouse $i$ all of the product required by customer $j$. The solution tree for the problem appears in Figure 8. The numbers in each ellipse denote the warehouses which are open at that node. The minimum value of the objective function at each node is written adjacent to the ellipse representing that node.

Inspection of the solution tree reveals that the optimal value of the objective function is 3180 and this value is realized by operating warehouses 1 and 3.
Figure 8. Solution Tree for DETER.
The forward search method follows the path drawn through the solution tree from the root node to the node which corresponds to operating warehouse 4. The search begins by calculating the minimum value of the objective function at the root node. This value was found to be 3536. Next, the same calculation is made for each of the nodes at the next level of the solution tree. The minimum of these values is chosen as the branch to be taken. The node so determined corresponds to operations of warehouses 1, 2, 4, 5 with minimum value of the objective function 3363. The calculation is then repeated for each subset of set \{1, 2, 4, 5\} containing exactly three plants. The minimum value, 3246, was found to exist at node 1, 2, 4. In a similar fashion, the forward search continues through the nodes 2, 4 and 4 with minimum values of the objective function 3195 and 3185, respectively. This completes the forward search portion of the algorithm.

The backward search begins by checking the sufficient conditions for a plant to be in the solution. This consists of comparing the value of the objective function at level 5 with the value at each node in level 4. The value of the objective function at each node in level 4 is less than the value at the root node. Therefore no statement can be made concerning the appearance of one or more warehouses in the node which gives the optimal solution.
Consequently, the minimum value of the objective function must be found for all nodes in level 1. The minimum is found to exist for node 4. Next, the minimum value of the objective function is calculated for all nodes in level 2 which contain warehouse 4. These are nodes 1, 4; 2, 4; 3, 4; 4, 5. The minimum value, 3195, is found to exist at node 2, 4. The backward search then continues to level 3. At this level nodes 1, 2, 4; 2, 3, 4, 2, 4, 5 are involved in the search. The minimum, 3246, is found at node 1, 2, 4. The final level involved in the backward search, level 4, yields, in a similar fashion, node 1, 2, 4, 5 with the value of the objective function 3363.

The smallest value of the objective function found at any node in either search was 3185 at node 4.

The terminal iteration is the final step in the algorithm. In the A matrix, the costs of supplying the customers in the best solution found by the search are underlined. The potential absolute minimum cost for supplying each customer is indicated by an asterisk (*). The following vectors are then found.

Customer  1  2  3  4  5
Assignment  4  4  4  4  4
Minimum  1  3  1  2  3

The last two rows are compressed as follows, eliminating duplicate pairs.

\[
\begin{array}{cccc}
4 & 4 & 4 \\
1 & 3 & 2
\end{array}
\]
These vectors then are used to accomplish the following replacements.

<table>
<thead>
<tr>
<th>Replace Warehouse</th>
<th>By Warehouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1,2</td>
</tr>
<tr>
<td>4</td>
<td>1,3</td>
</tr>
<tr>
<td>4</td>
<td>2,3</td>
</tr>
</tbody>
</table>

The nodes at which the minimum value is calculated and values of the objective function at these nodes are then calculated.

<table>
<thead>
<tr>
<th>Node</th>
<th>Objective Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3765</td>
</tr>
<tr>
<td>2</td>
<td>4016</td>
</tr>
<tr>
<td>3</td>
<td>3223</td>
</tr>
<tr>
<td>1,2</td>
<td>3450</td>
</tr>
<tr>
<td>1,3</td>
<td>3180</td>
</tr>
<tr>
<td>2,3</td>
<td>3311</td>
</tr>
</tbody>
</table>

Note that node 1,3 has the value of the objective function 3180, which is less than the minimum value found by the search method. In fact this is the optimal solution, as was noted at the beginning of the example.
The DETER algorithm was also coded in the PL/1 language and appears as procedure DETER in Appendix B with sample output.

The DETER algorithm was used to solve 5856 problems of various sizes. The exact solution for each of these problems was found by the algorithm PLANT which was described in section A of this chapter. The DETER algorithm found the optimal solution in 5837 of these 5856 problems, or 99.6% of the problems. Additional statistics for this method are given in the Tables which appear in the next chapter.
VIII. CONCLUSIONS AND DIRECTIONS FOR FURTHER WORK

This chapter contains a summary of the computational results from experimental problems solved by algorithm DETER and conclusions based on these results. The experience gained while developing the algorithms has led to ideas for improving the DETER algorithm. These ideas are discussed under "Directions for Further Work.:

A. Computational Results

This section contains a summary of the computational results obtained from solving a large number of problems using algorithm DETER. The tabular summaries give information about the 19 problems in which the optimal solution was not found by the DETER algorithm. Tables were constructed which compare the maximum number of nodes at which it may be necessary to determine the minimum value for the total enumeration method, DETER, and two preliminary forms of DETER. Experimentally acquired timing information is also given.

Table II summarizes the cases in which algorithm DETER did not find the correct solution.
TABLE II
SUMMARY OF PROBLEMS IN WHICH DETER DID NOT GIVE OPTIMAL SOLUTION

<table>
<thead>
<tr>
<th>Size Warehouses</th>
<th>Customers</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>3.3</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>.4</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>.5</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>2.4</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>.2</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>.6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>4.5</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>.8</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>.7</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>.05</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>.2</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>.6</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>1.2</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>2.6</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>2.3</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>1.6</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>2.0</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>.1</td>
</tr>
</tbody>
</table>

Average error 1.4%.
Table III provides a comparison of the maximum number of nodes at which the objective function is evaluated for the various methods considered.

**TABLE III**

**COMPARISON OF MAXIMUM NUMBERS OF NODES**

<table>
<thead>
<tr>
<th>No. of Plants</th>
<th>All Combinations</th>
<th>Forward Search</th>
<th>Backward Search</th>
<th>Forward &amp; Backward with Terminal Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>$2^m - 1$</td>
<td>$\frac{m(m+1)}{2}$</td>
<td>$\frac{m(m+1)}{2}$</td>
<td>$m(m+1) + \binom{m}{1} + \binom{m}{2}$</td>
</tr>
<tr>
<td>7</td>
<td>127</td>
<td>28</td>
<td>28</td>
<td>84</td>
</tr>
<tr>
<td>11</td>
<td>2,047</td>
<td>66</td>
<td>66</td>
<td>198</td>
</tr>
<tr>
<td>12</td>
<td>4,095</td>
<td>78</td>
<td>78</td>
<td>234</td>
</tr>
<tr>
<td>24</td>
<td>16,777,115</td>
<td>300</td>
<td>300</td>
<td>900</td>
</tr>
<tr>
<td>40</td>
<td>$2^{40} \approx 10^{12}$</td>
<td>820</td>
<td>820</td>
<td>2460</td>
</tr>
</tbody>
</table>

Table IV gives typical times for finding an approximation to the solution of the plant location problem using the DETER algorithm.
TABLE IV

REPRESENTATIVE TIMES FOR SOLVING THE SIMPLE PLANT LOCATION BY ALGORITHM DETER

<table>
<thead>
<tr>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>24</td>
<td>50</td>
</tr>
<tr>
<td>40</td>
<td>80</td>
</tr>
</tbody>
</table>

Table V provides the basis for the conclusions concerning the percentage of cases in which the optimal solution was found by the algorithm DETER, as well as the preliminary forms of the algorithm.

From the results obtained from early computer solutions it was observed that the forward search method seemed to work best when the fixed charges are small relative to the elements of the A matrix. The backward search method appeared to work best in the situation in which the elements of the F matrix were large in comparison with the elements of the A matrix. A set of 40 matrices were generated for 9 by 12 problems. Of these 40 problems the forward search method found only 24 of the optimal
TABLE V

SUMMARY OF THE PERFORMANCE OF THE ALGORITHMS

<table>
<thead>
<tr>
<th>No. of Problems</th>
<th>No of Warehouses</th>
<th>Forward Search</th>
<th>Forward and Backward</th>
<th>(DETER) F &amp; B with Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>2700</td>
<td>5</td>
<td>77</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>2512(1430)*</td>
<td>7</td>
<td>195</td>
<td>33*</td>
<td>13</td>
</tr>
<tr>
<td>244</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>9</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>360</td>
<td>10</td>
<td>29</td>
<td>11</td>
<td>3</td>
</tr>
</tbody>
</table>

* Part of the 7 warehouse cases were not checked for the number of failures by an intermediate form of the algorithm. 1430 problems were checked for the number of failures using the forward and backward searches combined. Of these, there were 33 problems for which the optimal solution was not found.

solutions. When the backward search method was added to the program, the optimum solutions were found in all 40 problems. The range of values in these sample problems was:

\[ 0 \leq a_{ij} \leq 1414; \quad 1000 \leq f_i \leq 3000. \]

B. Conclusions

Two algorithms for solving the simple plant location problem have been developed in this dissertation.

The first algorithm, PLANT, is a technique of implicit enumeration, similar in some respects to the Branch and Bound Algorithm. Its performance was found to be data
dependent as evidenced by the results given in Table I, page 41. Inspection of Table I justifies the two additional conclusions:

1. There appears to be no single function which connects the number of nodes at which the minimum value of the objective function must be found with the sizes of the elements in the $A$ and $F$ matrices.

2. Inspection of those segments of Table I which represent cases in which the $A$ matrix was constant indicate that if the size of the elements of the $F$ matrix are increased, then the number of nodes tends to increase. This is forcefully illustrated by the last entry in Table I, which represents a case in which total enumeration was necessary.

As the number of nodes increases, the computer storage which may be necessary for retaining the functional values and identification of the nodes increases even more rapidly. (Adding one potential warehouse site to the problem doubles the number of nodes in the solution tree and may nearly quadruple the amount of storage required.) These requirements on computer time and storage constrained most severely the usefulness of PLANT. (These are also the types of constraints which limit the usefulness of the Branch and Bound technique.) The techniques in algorithm PLANT may be useful if an additional result can be obtained. (See Directions for Further Work.)
The limited success realized from the application of algorithm PLANT to example problems caused a re-direction of effort. It was decided to try to develop an approximation algorithm which would produce good results. The criteria for success in this effort were given as:

1. The ability to approximate an optimal solution to "large" problems in a reasonable amount of computer time with the computer equipment available.
2. Determination of an optimal solution in at least 99% of the test problems.
3. An approximate solution must be "near" the optimal solution.

The first requirement was satisfied since a problem of size 100 warehouse sites and 200 customers can be solved in 150,000 bytes of core storage. The time consumed is data dependent but a formula for an upper bound on the number of nodes is given in Table III, page 56. The second goal was met, since the optimal solution was obtained in 99.6% of the test problems solved by DETER. The third condition was also successfully realized since in the 19 problems in which DETER did not find the optimal solution, the largest relative error was 4.5% and in 10 of the 19 problems the relative error was less than one percent.

The approximation algorithm, DETER, involved a search technique and consequently was similar in some respects to other search algorithms which appear in the literature.
DETER is, however, different in at least three respects. They are:

1. The search is guided by the results of theorems which were proved.
2. The theorems may initially restrict the backward search to a very small subtree of the solution tree.
3. The terminal iteration technique was developed as a result of empirical results obtained from preliminary versions of the algorithm and a heuristic developed as a consequence of a theorem which was proved.

The results obtained compare most favorably with other results reported in the literature. The percentage of cases in which the optimal solution was found, 99.6% was higher than that reported for any other approximation technique. The average relative error was lower than any reported in the literature for an approximation algorithm. The maximum error for some algorithms was reported as high as 14%, while in other papers this value was not given.

The exact methods which were reported give encouraging times for those problems tested. However, no indication was given for the maximum amount of time and storage which might be required for solving large problems or how the character of the data might influence those two quantities.
Problems arose in comparison of results since many authors gave no descriptions of their algorithms or no statements on computational results, timing, storage requirements, etc. It is hoped that readers of this dissertation have a minimal number of such comments.

The total computational results indicate that the approach of a directed tree search with terminal iterations provides a useful and successful approach to solving the simple plant location problem.

C. Directions for Further Work

It has been generally agreed that algorithms which are "data directed" would be most valuable in solving the simple plant location problem. Such an algorithm could be developed with algorithm DETER as a starting point provided additional requirements can be achieved. One additional requirement would be proof of sufficient conditions for the exclusion of a warehouse from the set of warehouses which gives the optimal value of the objective function. (Sufficient conditions for inclusion of warehouses in this set were proved in Chapter VI.) Once such conditions are proved, application of the theorems to the problem could greatly decrease the number of potential nodes at which the optimum could occur. For example, if we have 20 potential warehouse sites and
application of the theorems reveal that a certain five warehouses will be in the optimal solution while five others will not be in the optimal solution, the number of additional nodes at which the minimum value of the objective function would have to be calculated to find the exact solution by an algorithm such as PLANT is $2^{10} = 1024$. This is small when compared to the number of nodes in the solution tree, $2^{20} = 1,048,576$. Under these conditions, a variation of an algorithm such as PLANT could be used to find the optimal solution. Should the applications of such theorems not eliminate a significant number of nodes from consideration, then DETER could be applied to the problem to obtain an approximate solution.

A natural extension of the simple plant location problem results when capacity constraints are placed on one or more warehouses. The constraint for a warehouse limits the amount of the product which can be supplied by that warehouse. This type of constraint greatly complicates the problem and renders useless the algorithms which have been created for solving the simple plant location problem.

Two variations of this problem result when the following assumptions are made:

1. The total amount of the product which is available is equal to the total of the customers' demands.
2. The total amount of the product is insufficient to supply the customers' demands. In this case a
penalty function associated with each customer for not having his demands met would be introduced. These penalty functions will, in general, increase the optimal value of the objective function. The name which has been given to this problem in the literature is the **capacitated** plant location problem. A limited number of results have been published concerning this problem and much work remains to be done on the problem.


Case 1. Linear Cost

Figure 9. Linear Warehousing Cost Function

In this case the function \( Z_i = f_i + \lambda_i t \) is composed of two parts:

\( f_i : \) the fixed cost of opening and operating the plant

\( \lambda_i t : \) the cost of manufacturing \( t \) units of the product at plant \( i \). That is \( \lambda_i \) is the unit cost of manufacturing at plant \( i \).

To solve this problem we let \( C_{ij} = (a_{ij} + \lambda_i) d_j \) in the final formulation in the statement of the problem.
Case 2. Linear Piecewise Function

Figure 10. Piecewise Linear Warehousing Cost Function.

The problem with fixed cost $f^1_i$ and linear segments can be given the following interpretation. The cost of opening and maintaining plant $i$ is given by $f^1_i$. The unit cost of manufacturing from 0 to $t_1$ units of the product is given by the slope of segment AB. The unit cost of manufacturing from $t_1 + 1$ to $t_2$ units is given by the slope of segment BC and similarly for producing from $t_2 + 1$ to $t_3$ units.

To handle this case, replace the plant by three pseudo plants with costs

$$z^1_i = f^1_i + \lambda_1 t$$
$$z^2_i = f^2_i + \lambda_2 t$$
$$z^3_i = f^3_i + \lambda_3 t$$
This causes the inclusion in the C matrix of three rows instead of one as was true in case 1. The rows in the transportation/production matrix corresponding to the three pseudo plants are given by

\[ C_{ij}^k = (a_{ij} + \lambda_k) d_j \quad k = 1, 2, 3. \]

Only one of the three pseudo plants will be chosen. If the number manufactured is between 0 and \( t_1 \), pseudo plant 1 will be chosen, etc.

Case 3. Concave Warehouse Functions

If we have a concave function which is not piecewise linear, we may approximate it by a series of line segments and proceed as in case 2.

Figure 11. Concave Warehousing Cost Function
APPENDIX B

PROGRAM LISTINGS AND SAMPLE OUTPUT

1. Algorithm PLANT
   a. Input (in PL/1 list directed format.)
      Field 1: m - the number of warehouse sites
      Field 2: n - the number of customers
      Field 3 through Field m+2: The elements of the
         F vector in order of increasing
         subscripts.
      Field m+3 through Field mn+m+2: The elements
         of the A matrix in row major order.
   b. Program Listing

```
PLANT: PROCEDURE OPTIONS(MAIN);
DECLARE TESTED(*) BIT(*) CONTROLLED;
DECLARE ALLSOL(*) BIT(*) CONTROLLED;
DECLARE (A(*,*) , COST_MIN(*), F(*), VALKEEP(*,*) , (KEEP(*,*),
EXCLUDE(*) , TESTER(*) , OUT) BIT(*)) CONTROLLED;
DECLARE IK(2) , INCLUDE BIT(100) VARYING;
ON ENDFILE(SYSIN) GO TO STOP;
NEW DATA: GET LIST(M,N);
PUT PAGE;
INCLUDED =';
PUT SKIP;
PUT LIST(M,N);
ALLOCATE A(M,N), COST_MIN(N), F(M), OUT BIT(M), TESTER(M) BIT(M);
ALLOCATE ALLSOL(M*N) BIT(M);
GET LIST((F(I) DO I=1 TO M));
GET LIST(((A(I,J) DO J=1 TO N) DO I=1 TO M));
PUT SKIP;
PUT LIST(' F-VECTOR');
PUT SKIP;
PUT LIST((F(I) DO I=1 TO M));
DO I=1 TO M;
PUT SKIP;
PUT LIST(' A-MATRIX, ROW',I);
PUT SKIP;
```
PUT LIST((A(I,J) DO J=1 TO N));
END;
PUT SKIP;
I=M/2;
TEMP=1.0;
DO K=1 TO I;
TEMP=TEMP * (M-K+1.)/(I-K+1.);
END;
I=TEMP;
ALLOCATE KEEP(0:I,2) BIT(M), EXCLUDE(I) BIT(M), VALKEEP(0:I,2);
ALLOCATE TESTED(I) BIT(M);
DO J=1 TO M;
SUBSTR(KEEP(0,1),J,1)='1'B;
END;
OUT=KEEP(0,1);
KEEP(1,2)=OUT;
CALL SOLVE;
VALKEEP(1,2)=COST;
VALKEEP(0,1)=COST;
KSOLPT=1;
ALLSOL(KSOLPT) =OUT;
DO I=1 TO M;
DO J=1 TO M;
IF I = J
THEN SUBSTR(TESTER(I),J,1)='1'B;
ELSE SUBSTR(TESTER(I),J,1)='0'B;
END; END;
IK=0;
IX=0;
DO I=1 TO M;
OUT=BOOL(TESTER(I),KEEP(1,2),'0100'B);
CALL SOLVE;
IF COST < VALKEEP(0,1)
THEN DO;
KEEP(0,1)= OUT;
KSOLPT=1;
ALLSOL(KSOLPT) =OUT;
VALKEEP(0,1)=COST;
GO TO NEXT;
END;
IF COST=VALKEEP(0,1) THEN DO;
KSOLPT=KSOLPT+1;
ALLSOL(KSOLPT) =OUT;
END;
NEXT:
IF COST<= VALKEEP(1,2)
THEN DO;
INCLUDE= INCLUDE || '0'B;
IK(1)= IK(1)+ 1;
KEEP(IK(1),1)=OUT;
VALKEEP(IK(1),1)=COST;
END;
ELSE DO;
INCLUDE = INCLUDE || '1' B;
IX=IX +1;
EXCLUDE(IX)=OUT;
END;
END;
IMC=2;
IMF=1;
/* PROBLEM INITIALIZED, START ITERATIVE PROCESS */
ITERATE:I=IMC;
IMC= IMF;
IMF=I;
IK(IMF)=0;
IV=INDEX(KEEP(1,IMC),'1'B);
IF IV=M THEN DO;
IW=C;
GO TO T;
END;
IW=INDEX(SUBSTR(KEEP(1,IMC),IV+1),'1'B);
T:IF IW = 0 THEN DO;
PUT SKIP;
PUT EDIT(' SOLUTION TO PLANT LOCATION PROBLEM',VALKEEP(0,1)) (A,F(20));
PUT SKIP;
PUT EDIT(' ALL SOLUTIONS FOLLOW') (A);
PUT SKIP;
DO I=1 TO KSOLPT;
PUT SKIP;
PUT EDIT(' SOLUTION NUMBER',I,ALLSOL(I)) (A,F(5),X(5),B);
PUT SKIP;
PUT EDIT(' CUSTOMER SUPPLIED BY PLANT') (A);
OUT = ALLSOL(I);
CALL PINUP;
END;
FREE A,COST_MIN,F,OUT,TESTER,KEEP,EXCLUDE,VALKEEP;
go TO NEW_DATA;
END;
ITESTED = 0;
Z:DO I=1 TO IK(IMC);
B:DO J=1 TO M;
OUT=BOOL(TESTER(J),KEEP(I,IMC),'0100'B);
IF OUT = KEEP(I,IMC)
THEN GO TO END_B;
IF BOOL(INCLUDE,OUT,'0001'B) = INCLUDE
THEN GO TO END_B;
ELSE
C:DO K=1 TO IX;
IF BOOL(EXCLUDE(K),OUT,'0001'B)=OUT
THEN GO TO END_B;
ELSE; END C;
IF ITESTED = 0 THEN DO;
DO JW=1 TO ITETESTED;
IF OUT = TESTED(JW) THEN GO TO END_B;
END;
END;
ITESTED=ITESTED+1;
TESTED(ITESTED)= OUT;
CALL SOLVE;
IF COST > VALKEEP(I,IMC)
THEN DO;
IX=IX+1;
EXCLUDE(IX)= OUT;
IF COST=VALKEEP(0,1) THEN DO;
KSOLPT=KSOLPT+1;
ALLSOL(KSOLPT)=OUT;
END;
GO TO END_B;
END;
ELSE DO;
DO K=IK(IMF) TO 1 BY -1;
IF KEEP(K,IMF) = OUT THEN GO TO TZ;
END;
IK(IMF)=IK(IMF)+1;
KEEP(IX,IMF)=OUT;
VALKEEP(IK(IMF),IMF)=COST;
END;
TZ: IF COST < VALKEEP(0,1)
THEN DO;
KEEP(0,1)= OUT;
KSOLPT=1;
ALLSOL(KSOLPT) =OUT;
VALKEEP(0,1)=COST;
GO TO END_B;
END;
IF COST=VALKEEP(0,1) THEN DO;
KSOLPT=KSOLPT+1;
ALLSOL(KSOLPT)=OUT;
END;
END_B: END B;
END_Z;
IF IK(IMF) = 0
THEN DO;
PUT SKIP;
PUT EDIT(' CUSTMRN SUPPLD BY PLNT') (A);
PUT SKIP;
DO I=1 TO KSOLPT;
PUT SKIP;
PUT EDIT(' SRLTN NMBR','I,ALLSOL(I)) (A,F(5),X(5),B);
PUT SKIP;
PUT EDIT(' CUSTMRN SUPPLD BY PLNT') (A);
OUT = ALLSOL(I);
CALL FINUP;
END;
FREE A,COST_MIN,F,OUT,TESTER,KEEP,EXCLUDE,VALKEEP;
GO TO NEW_DATA;
END;
GO TO ITERATE;
/* THIS PROCEDURE IS USED TO SOLVE EACH LP PROBLEM*/
SOLVE:PROCEDURE;
COST=0.0;
DO JJ=1 TO N;
  COST_MIN(JJ)=9.0E20;
END;
SEARCH:DO II=1 TO M;
  IF SUBSTR(OUT,II,1) = 'l'B THEN BEGIN;
  COST = COST + F(II);
  DO JJ=1 TO N;
    COST_MIN(JJ)=MIN(COST_MIN(JJ),A(II,JJ));
  END;
END; END SEARCH;
COST= COST + SUM(COST_MIN);
END SOLVE;
FINUP:PROCEDURE;
DO II=1 TO N;
  CM=9.0E20;
  DO JJ=1 TO M;
    IF SUBSTR(OUT,JJ,1)= 'l'B THEN DO;
    IF A(JJ,II)>CM THEN GO TO BLAH;
    CM=A(JJ,II);
    ICM=JJ;
  END;
BLAH:END;
PUT SKIP;
PUT EDIT(II,ICM)(F(5),X(16),F(3));
END;
PUT SKIP(4);
END FINUP;
STOP: END PLANT;

C. Output

The first row contains the number of warehouses and the number of customers in the problem. The next rows are labelled. They contain the F-vector and the rows of the A matrix. Next the
value of the objective function is printed. The value is 65 in the example problem. All solutions are then listed. Following the number of the solution is a bit string which indicates the status of the warehouses in this solution. A 1 in the $i^{th}$ position of the bit string indicates that warehouse $i$ must be operated, a 0 in the $i^{th}$ position indicates that the warehouse must be closed. Next is given a customer/warehouse assignment. The output could be expanded to a more elaborate form, but all of the necessary information for interpreting the solution is contained in this form.

<table>
<thead>
<tr>
<th>F-VECTOR</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00000E+00</td>
<td>1.00000E+01</td>
</tr>
<tr>
<td>A-MATRIX, ROW 1</td>
<td>2.00000E+01</td>
</tr>
<tr>
<td>2.00000E+01</td>
<td>2.00000E+01</td>
</tr>
<tr>
<td>A-MATRIX, ROW 2</td>
<td>1.00000E+01</td>
</tr>
<tr>
<td>2.50000E+01</td>
<td>1.00000E+01</td>
</tr>
<tr>
<td>A-MATRIX, ROW 3</td>
<td>1.50000E+01</td>
</tr>
<tr>
<td>3.00000E+01</td>
<td>1.50000E+01</td>
</tr>
<tr>
<td>A-MATRIX, ROW 4</td>
<td>2.00000E+01</td>
</tr>
</tbody>
</table>

SOLUTION TO PLANT LOCATION PROBLEM 65
ALL SOLUTIONS FOLLOW

SOLUTION NUMBER 1 1101
CUSTOMER SUPPLIED BY PLANT
1 4
2 4
3 2
4 1
2. Algorithm DETER

a. Input (in PL/1 list directed format).

Field 1: m - the number of warehouse sites
Field 2: n - the number of customers
Field 3 through Field m+2: The elements of the F vector in order of increasing subscripts.
Field m+3 through Field mn+m+2: The elements of the A matrix in row major order.

b. Program Listing

DETER:PROCEDURE OPTIONS(MAIN);
DCL VDING BIT(*) CONTROLLED;
DCL (ABS_MIN(*),SOLN(*)) BINARY FIXED CONTROLLED;
DCL DING^BIT(*) CONTROLLED;
DCL INCLUDE BIT(100) VARYING,
MINBIT (2) BIT(*) CONTROLLED,
(A(*,*),F(*),OUT BIT(*))CONTROLLED,
VAL(2),
ADDA BIT(*) CONTROLLED;
DCL COST_MIN(*) CONTROLLED;
on ENDFILE(SYSIN) GO TO KLOZ;
AGAIN:;
GET LIST(M,N);
ALLOCATE VDING BIT(M);
ALLOCATE ABS_MIN(N), SOLN(N);
ALLOCATE DING_BIT(M);
ALLOCATE COST_MIN(N);
ALLOCATE ADDA_BIT(M), A(M, N), F(M), OUT_BIT(M), MINBIT(2) BIT(M);
INCLUDE = 'I;
GET LIST((F(I) DO I=1 TO M));
GET LIST(((A(I, J) DO J=1 TO N) DO I=1 TO M));
DO J=1 TO M;
SUBSTR(MINBIT(1), J, 1) = '1'B;
END;
OUT = MINBIT(1);
CALL SOLVE;
VAL(1) = COST;
VAL(2) = 9.0E40;
DO J=1 TO M;
OUT = MINBIT(1);
SUBSTR(OUT, J, 1) = '0'B;
CALL SOLVE;
IF COST < VAL(2) THEN DO;
VAL(2) = COST;
MINBIT(2) = OUT;
END;
IF COST < VAL(1) THEN INCLUDE = INCLUDE || '0'B;
ELSE INCLUDE = INCLUDE || '1'B;
END;
IF VAL(1) < VAL(2) THEN DO;
IP = 1;
GO TO XX;
END;
IP = 2; IC = 1;
K4: DO I=2 TO M-1;
VAL(IC) = 9.0E40;
DO J=1 TO M;
OUT = MINBIT(IP);
SUBSTR(OUT, J, 1) = '0'B;
IF OUT = MINBIT(IP) THEN GO TO K5;
IF INCLUDE￢ = INCLUDE & OUT THEN GO TO K5;
CALL SOLVE;
IF COST < VAL(IC) THEN DO;
VAL(IC) = COST;
MINBIT(IC) = OUT;
END;
K5: END;
IF VAL(IC) < VAL(IP) THEN DO;
IZ = IP;
IP = IC;
IC = IZ;
ELSE GO TO XX;
END K4;
XX:;
TEMP=VAL(IP);
ADDA=MINBIT(IP);
IF INCLUDE= '0'B THEN DO;
MINBIT(1)=INCLUDE;
OUT=INCLUDE;
CALL SOLVE;
VAL(1)=COST;
END;
ELSE DO;
VAL(1)=9.9E40;
DO J=1 TO M;
OUT = INCLUDE;
SUBSTR(OUT,J,1)= '1'B;
CALL SOLVE;
IF COST < VAL(1) THEN DO;
VAL(1)=COST;
MINBIT(1)=OUT;
END;
END;
IC=2;
IP=1;
ITERATE:;
VAL(IC)=9.0E40;
DO J=1 TO M;
OUT=MINBIT(IP);
SUBSTR(OUT,J,1)= '1'B;
IF OUT=MINBIT(IP) THEN GO TO D1;
CALL SOLVE;
IF COST < VAL(IC) THEN DO;
VAL(IC)=COST;
MINBIT(IC)=OUT;
END;
D1:END;
IF VAL(IC)<VAL(IP) THEN DO;
IZ=IP;
IP=IC;
IC=IZ;
GO TO ITERATE;
END;
/* LOCATES ABSOLUTE MINS FOR EACH CUSTOMER*/
DO I=1 TO N;
FMIN= A(1,I);
ABS_MIN(I)=1;
DO J=2 TO M;
IF A(J,I) < FMIN
THEN DO;
FMIN=A(J,I);
ABS_MIN(I)=J;
END;
END;
END;

/* THIS FINDS MINS IN THE ASSIGNMENT */
IF VAL(IP)<= TEMP
THEN DO;
TEMP=VAL(IP);
VDING=MINBIT(IP);
END;
ELSE VDING = ADDA
DO KS=1 TO 2;
IF KS=1
THEN DING=MINBIT(IP);
ELSE DING=ADDA;
IF KS=2 & ADDA=MINBIT(IP)
THEN GO TO SKIPPER;
BLAZES: DO J=1 TO M;
IF SUBSTR(DING,J,1)='1'B THEN DO;
K=J; GO TO NEXT1; END;
END;
NEXT1: DO I=1 TO N;
FMIN = A(K,I);
SOLN(I)=K;
DO J=K+1 TO M;
IF SUBSTR(DING,J,1)='0'B THEN GO TO TEXAS;
IF A(J,I) < FMIN
THEN DO;
FMIN=A(J,I);
SOLN(I)=J;
END;
TEXAS: END;
END;
IPT=1;
DO I=1 TO N;
IF ABS_MIN(I)=SOLN(I)
THEN GO TO FFF;
IF SUBSTR(INCLUDE,SOLN(I),1)='1'B
THEN GO TO FFF;
DO J=1 TO IPT -1;
IF ABS_MIN(J)=ABS_MIN(I) &
SOLN(J)=SOLN(I) THEN GO TO FFF;
END;
ABS_MIN(IPT)=ABS_MIN(I);
SOLN(IPT)= SOLN(I);
IPT=IPT+1;
FFF:END;
/* VECTORS COMPRESSED, NOW START SWAPPING PROCESS */
/* FIRST SWAP ONE PLANT */
DO I=1 TO IPT-1;
OUT = DING;
SUBSTR(OUT,ABS_MIN(I),1)='1'B;
SUBSTR(OUT,SOLN(I),1)='0'B;
CALL SOLVE;
IF COST < TEMP THEN DO;
TEMP=COST;
VDING=OUT;
END;
END;

/* NOW SWAP TWO PLANTS */
DO I=1 TO IPT-2;
DO J=I+1 TO IPT-1;
OUT=DING;
SUBSTR(OUT,ABS_MIN(I),1)='1'B;
SUBSTR(OUT,ABS_MIN(J),1)='1'B;
SUBSTR(OUT,SOLN(I),1)='0'B;
SUBSTR(OUT,SOLN(J),1)='0'B;
CALL SOLVE;
IF COST < TEMP THEN DO;
TEMP=COST;
VDING=OUT;
END;
END;
END;

SKIPPER:;
CALL FINISH;
FREE VDING,ABS_MIN,SOLN,DING,COST_MIN,ADDA,F,OUT,MINBIT;
GO TO AGAIN;
SOLVE: PROCEDURE;
COST=0.0;
DO JJ=1 TO N;
COST_MIN(JJ)=9.0E40;
END;
SEARCH: DO II=1 TO M;
IF SUBSTR(OUT,II,1)='1'B THEN BEGIN;
COST=COST+F(II);
DO JJ=1 TO N;
COST_MIN(JJ)=MIN(COST_MIN(JJ),A(II,JJ));
END; END; END SEARCH;
COST=COST+SUM(COST_MIN);
END SOLVE;
FINISH: PROCEDURE;
PUT PAGE;
PUT EDIT (' PROBLEM HAS', M,' PLANTS',',N,' ' CUSTOMERS '))(A,F(4),A,F(4),A);
PUT SKIP(2);
DO I=1 TO M;
PUT EDIT(F(I),(A(I,J) DO J=1 TO N))(12F(10));
PUT SKIP(2);
END;
PUT EDIT(' THE FOLLOWING PLANTS MUST BE IN THE SOLN') (A);
PUT SKIP;
DO J=1 TO M;
IF SUBSTR(INCLUDE,J,1)='0'B THEN GO TO LEND;
PUT EDIT(J) (F(3));
LEND:END;
PUT SKIP(2);
PUT EDIT(' THE PLANTS IN THE APPROXIMATE SOLN ARE') (A);
PUT SKIP;
DO J=1 TO M;
IF SUBSTR(VDING,J,1)='0'B THEN GO TO GEND;
PUT EDIT(J) (F(3));
GEND:END;
PUT SKIP(2);
PUT EDIT(' THE ASSIGNMENTS FOLLOW') (A);
PUT SKIP;
PUT EDIT(' CUSTOMER SUPPLIED BY PLANT') (A);
PUT SKIP;
DO I=1 TO N;
CM=9.0E40;
DO J=1 TO M;
IF SUBSTR(VDING,J,1)='0'B THEN GO TO BEND;
IF A(J,I) >= CM THEN GO TO BEND;
CM= A(J,I);
ITTT=J;
BEND:END;
PUT EDIT(I,ITTT) (F(5),X(16),F(3));
PUT SKIP;
END;
PUT EDIT('COST OF SOLUTION = ',TEMP) (A,F(10));
END FINISH;
KLOZ:END DETER;

c. Output

The output from procedure DETER includes the number of plants and the number of customers on the first line. There are then m unlabelled groups of data,
each group consisting of n+1 numbers. The first number in group i is \( f_i \), the fixed cost associated with plant i. The next n numbers in group i are the n elements in row i of the matrix \( A \). (This gives the user the ability to check the input data for incorrect values.) Next, following the heading is a list of plants which must be in the optimal solution, if sufficient conditions for the inclusion of one or more plants are satisfied. The plants which are in the approximate solution are then listed. The plant/customer assignments are given. Finally the value of the objective function for this solution is given. (The sample output was obtained by applying the program to the example given in Chapter VII, Section B.)

<table>
<thead>
<tr>
<th>PROBLEM HAS</th>
<th>5 PLANTS, 5 CUSTOMERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>136</td>
<td>551</td>
</tr>
<tr>
<td>104</td>
<td>897</td>
</tr>
<tr>
<td>199</td>
<td>648</td>
</tr>
<tr>
<td>151</td>
<td>596</td>
</tr>
<tr>
<td>117</td>
<td>804</td>
</tr>
</tbody>
</table>

THE FOLLOWING PLANTS MUST BE IN THE SOLN

THE PLANTS IN THE APPROXIMATE SOLN ARE 1 3
THE ASSIGNMENTS FOLLOW
CUSTOMER SUPPLIED BY PLANT

\[
\begin{array}{cc}
1 & 1 \\
2 & 3 \\
3 & 1 \\
4 & 3 \\
5 & 3 \\
\end{array}
\]

COST OF SOLUTION = 3180
This description is given to make the timing information and limitations on problem size more meaningful.

Equipment:

IBM System 360 Model 50 with 256K bytes of high speed core storage.
1 selector channel
1 multiplexor channel
1 5 drive 2314 disk storage unit
1 1100 line per minute printer
1 1000 card per minute reader
2 magnetic tape units
1 200 card per minute card reader
1 500 card per minute card punch
VITA

John Bruce Prater was born on November 26, 1932 near Pleasant Hope, Missouri. He received the first six years of his education at the New Hope School. His secondary education was received in Pleasant Hope, Missouri. He has received his college education from Southwest Missouri State College, in Springfield, Missouri; the University of Missouri-Columbia, in Columbia, Missouri; and Oklahoma State University, in Stillwater, Oklahoma. He received a Bachelor of Science degree in Mathematics from Southwest Missouri State College, in Springfield, Missouri in May, 1956. He received a Master of Arts degree in Mathematics from the University of Missouri, Columbia, Missouri, in June, 1959. He has been enrolled in the Graduate School of the University of Missouri-Rolla since February 1965 and was a National Science Foundation Science Faculty Fellow for the period September 1968 to September 1969.