1967

On the solution of the general single-stage location-allocation problem by a branch-and-bound algorithim

Robert E. Fleisher
Lehigh University

ON THE SOLUTION OF THE GENERAL SINGLE-STAGE LOCATION-ALLOCATION PROBLEM BY A BRANCH-AND-BOUND ALGORITHM

by

ROBERT ERWIN FLEISHER

A Thesis

Presented to the Graduate Faculty

of Lehigh University

in Candidacy for the Degree of

Master of Science

Lehigh University

1967

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 3, 1967
Date

Gary E Whitehouse
Professor in Charge

Head of the Department of
Industrial Engineering

ACKNOWLEDGEMENTS

The author wishes to thank his thesis advisor, Professor G. E. Whitehouse, for his guidance and interest in the work presented here. Thanks are also due Professors A. F. Gould and J. M. Carroll for serving as members of the advisory committee.

I wish to also record my appreciation of the time spent by Mr. J. H. Lamb of the Western Electric Co., Inc., Engineering Research Center during numerous discussions with the author and his resulting helpful suggestions.

Last but by no means least, many thanks to my wife, Anna Jane, and children, Katherine and Robert, Jr. Their encouragement and consideration immeasurably contributed to the completion of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The location-allocation problem is treated from the standpoint of the computational efficiency of a mixed-integer programming formulation solved by a branch-and-bound algorithm. Previous formulations of the problem are analyzed and a new formulation is developed. The new formulation is shown to be computationally superior to prior models from analytic and experimental results obtained with the branch-and-bound algorithm used. A mathematical expression for the expected number of nodes that must be evaluated to obtain a solution for problems of varying size is developed. The results obtained concur with the contention made by some that the efficiency of the branch-and-bound method is dependent upon both the problem at hand and the algorithm used for solution. The structure of these elements must be considered simultaneously for computational efficiency.

I INTRODUCTION

The attempts to apply quantitative methods to the solution of problems formerly relegated to qualitative analysis and solution by insight or "guess" has forced the problem solver to utilize the concepts and techniques developed for use in diverse fields as well as requiring him to develop new tools unique to his situation. The concept of mathematical models, used extensively in the physical sciences, has certainly been applied to great advantage in the "management" sciences. It frequently occurs, however, that a given mathematical model cannot readily be solved for the optimal or near optimal parameters with existing techniques. Consequently, new tools must be developed to overcome this situation.

Frequently, a mathematical model of a physical situation takes the form of a combinatorial problem; i.e., its solution requires the examination of numerous combinations of the variables over their allowed ranges. These problems always have associated with them some function to be optimized and often restrictions are imposed on the combinations that can be used to optimize the function of interest. In a loose sense, the myriad of problems efficiently solved by linear programming algorithms can be considered combinatorial problems. In the case of these problems amenable to solution by linear programming, the tools for solution have been developed to a high degree of efficiency. There are, however, numerous combinatorial problems which cannot be solved by linear programming because one

or more of the variables in the model are not continuous over their range. The techniques of integer programming, mixed-integer programming, dynamic programming, or the branch-and-bound method must be used if one is not willing to enumerate all possible combinations. Unfortunately, the present body of knowledge on integer and mixed-integer programming is not as developed as in the case of linear programming. Consequently, the efficiency of the existing algorithms precludes their use in many problems. Dynamic programming will not apply to problems unless they fit the mold of the dynamic programming theory. The branch-and-bound method's efficiency is a function of the particular problem and may degenerate to a complete enumeration in some cases. Clearly, one is faced with a dilemma if a problem's solution requires the use of one of these techniques. This thesis is addressed to the solution of such a problem. It has been named the location-allocation problem and can be classified as an optimal facility planning problem.

The general problem of optimal facility planning has been stated by Elmaghraby (15) in the following way: given the location of each consuming center, together with its demand and the cost of transportation from any place to each consuming center, find the number, location, and capacity of each source that must be provided in order to minimize the system's total cost of operation. The location-allocation problem is simultaneously concerned with the allocation of consumers to sources. The problem is restricted here to a single-stage problem; i.e., one transportation stage. Ideally,

it would be desirable to optimally locate sources considering two or more transportation stages; but as will be shown, the single-stage problem poses enough difficulties to induce one to set aside consideration of the multi-stage problem until the single-stage problem can be solved efficiently.

The terms "location," "consumer," and "capacity" should be considered generic rather than literal. For example, in the problem of optimally locating a switching center in a communication network, the term "capacity" will certainly have a different connotation than in the problem of locating warehouses.

In the purely mathematical literature this problem, in one form or another, is very old. Cooper (8) reports finding references dating back to 1647 when Cavalieri considered the problem of finding the point the sum of whose distances from three given points is a minimum. It has only been in the last few years, however, that significant literature has appeared proposing methodologies for the solution of various types of location-allocation problems of industrial interest.

The most recent approaches (12)(23) formulate versions of the problem as mixed-integer programming problems. In (12), the solution of the mixed-integer program is obtained by a branch-and-bound algorithm. As discussed in (2) and (22), the efficiency of the branch-and-bound method varies greatly with the structure of the particular algorithm used. Using the branch-and-bound method suggested in (22), the formulation in (12) proves to be unsatisfactory from a computational standpoint.

This thesis will analyze the reasons for the computational inefficiency in (12) and develop a more efficient mixed-integer programming formulation from the standpoint of computational efficiency. The improvement will be demonstrated and an attempt will be made to indicate computing requirements of problems with varying size.

The thesis is organized in the following manner. Chapter II contains the statement of the problem as it will be considered here. This is necessary because the great interest in this topic has generated numerous ramifications of the problem statement and approaches to a solution. Chapter III will review some of the literature that has appeared on the subject. The branch-and-bound method is discussed in Chapter IV and Chapter V contains the mathematical development of the mixed-integer program and the algorithm for its solution. The computational results and comparisons are presented in Chapter VI.

II THE PROBLEM

The general location-allocation problem can be stated as follows: given the location of a number of consuming points, together with their demands and the transportation costs from any place to each consuming point, find the number, location, capacity, and consumer allocations of each source that must be provided in order to minimize the sum of transportation costs and cost of operating the sources. This statement deserves investigation into the precise meanings of its terms in order to arrive at a formal statement of the problem. Such an inquiry will be the theme of this chapter.

A Mathematical Formulation

Disregarding momentarily the requirement that each source's capacity must be determined, the remaining mathematical problem has been stated (8) as follows: suppose that, if m is the number of sources, the costs associated with the operation of these sources, Z_1, is given by some function of m, i.e.,

$$Z_1 = g_1(m) \qquad \text{eq. 1}$$

Further, let the cost of supplying the given set of n demand points be Z_2, which is again a function of m, given by:

$$Z_2 = g_2(m) \qquad \text{eq. 2}$$

where Z_2 is the minimum cost solution, i.e., the transportation costs to supply the n demand points from m optimally located sources.

The total cost is then given by:

$$Z = Z_1 + Z_2 = g_1(m) + g_2(m) \qquad \text{eq. 3}$$

Since m, the number of sources, is clearly discrete, the minimum cost solution is obtained when:

$$\Delta[g_1(m^*-1) + g_2(m^*-1)]<0<\Delta[g_1(m^*) + g_2(m^*)]$$

and

$$\Delta^2[g_1(m) + g_2(m)] \geq 0 \qquad \text{for all m,}$$

where m^* is the number of locations minimizing Z.

Assuming that the operation of m sources is independent of their locations, equation 1 can be obtained by fitting cost data with some emperical equation. A more complex problem is the determination of Z_2 for some fixed m. If it were possible to determine Z_2 for various values of m, the determination of Z_2 for all m is again a problem of fitting data. A critical assumption here is the independence of Z_1 and the locations of the m sources. With this assumption, the problem can be solved as two distinct problems, i.e. obtaining a function for Z_1 and Z_2 and then computing the minimum of Z. Unfortunately, the determination of Z_2 is of no small consequence.

Cooper (8) considers the determination of Z_2 for some fixed m. Let the location of the n demand points be given by (X_j, Y_j); j = 1, 2, ..., n, their Cartesian coordinates. Similarly, let the coordinates of the m sources to be determined be given by (X_i, Y_i); i = 1, 2, ..., m. Assume that any demand point, henceforth referred to as a consumer, can be supplied by any source. Then

$$Z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}\Phi(X_i, Y_i, X_j, Y_j) \qquad \text{eq. 4}$$

where

$$a_{ij} = \begin{cases} 1 \text{ if source i serves consumer j} \\ 0 \text{ otherwise} \end{cases}$$

$\Phi(X_i, Y_i, X_j, Y_j)$ = transportation costs associated with serving the j th consumer from the i th source.

In order to find the set of (X_i, Y_i) minimizing Z_2, differentiate equation 4 with respect to X_i and Y_i and solve the equations resulting from setting these derivatives equal to zero. Then

$$\frac{\partial Z_2}{\partial X_i} = \sum_{j=1}^{n} a_{ij} \left[\frac{\partial \Phi(X_i, Y_i, X_j, Y_j)}{\partial X_i} \right] = 0 \qquad \text{eq. 5}$$

and

$$\frac{\partial Z_2}{\partial Y_i} = \sum_{j=1}^{n} a_{ij} \left[\frac{\partial \Phi(X_i, Y_i, X_j, Y_j)}{\partial Y_i} \right] = 0$$

This then results in a set of 2m equations which when solved as pairs yield the m sets of optimal (X_i, Y_i) which minimize Z_2 for some set of a_{ij}. There are, however, for m sources and n consumers, S(n,m) possible allocations of n consumers to m sources, where S(n,m) is the Stirling number of the second kind and is given by

$$S(n,m) = \frac{1}{m!} \sum_{k=0}^{m} \binom{m}{k} (-1)^k (m-k)^n$$

This assumes that a consumer is supplied by only one source. For very large n, these Stirling numbers can be formidably large. For example, $S(25, 3) \simeq 14.1\ (10^{10})$, a truly magnificent number (1). Equations 5 would then require solution S(n,m) times and the minimum of these solutions will be Z_2 for some m. As Cooper points out, for large scale problems of industrial importance the amount of computation is prohibitive. Cooper develops an iterative procedure for the solution of equations 5 in which $\Phi(X_i, Y_i, X_j, Y_j)$ is a linear function of the Euclidean distance between sources and consumers. Although the procedure is of interest, it will not be discussed further here. Of more import toward obtaining a formal statement of the problem to be considered in this thesis is a review of this mathematical formulation and the significance of the results.

An Analysis of the Formulation

Consider the cost function, equation 1, for the operation of m sources. It is assumed by Cooper, although not explicitly stated, that equation 1 is a function of only the value of m. This does not appear to be a realistic assumption from a practical standpoint. Factors such as real estate availability, labor costs, tax rates, etc. will vary the fixed costs associated with the establishment and operation of a source as a function of where it is sited. Clearly, Z_1 is then no longer a function of m alone, but also of the particular m points chosen. Then for each m, there exists a set Z_1 of fixed costs for all possible distinct combinations of m

points selected from all possible source locations. Then, if S is the set of M (X_i, Y_i) that are possible locations for sources and Z_1 is the minimum fixed cost for some m, the following is true:

$$Z_1 = \min Z_1' = g_1(m, f_i),$$

where

$$Z_1' = \left\{ g_1(m, f_i) \mid (X_i, Y_i) \epsilon S \; \forall i \right\};$$

f_i = the fixed costs of a source at (X_i, Y_i); and

$g_1(m, f_i)$ = the fixed costs associated with the location of m sources at one of the $\binom{M}{m}$ possible location sets.

Returning now to the transportation costs, the minimum cost solution for some fixed m will again be

$$Z_2 = g_2(m)$$

since the transportation costs are clearly not a function of the f_i associated with the m points determined by equations 5. The total cost is then

$$Z = Z_1 + Z_2 = g_1(m, f_i) + g_2(m) \qquad \text{eq. 8}$$

Recall that in equation 3, the former total cost expression, a minimum can be obtained since Z was only a function of m. Equation 8, however, is a function of m and the f_i's associated with the m points formerly computed from equation 2. Obviously, it is no longer possible to obtain a minimum cost solution by treating the cost components of Z independently as was done previously. It follows then that a solution of the problem requires the optimization of Z where the location of m sources affect not only the transporta-

tion costs but also the fixed costs.

Consider now the significance of the solution obtained by Cooper; that is, the location of the m sources expressed as Cartesian coordinates. This allows the solution to fall in an infinity of points--a generalization of the classic Weber problem posed by Alfred Weber (29) in 1909. It would be more realistic, in problems of industrial importance, to consider some finite set of points as feasible solution points rather than the continuum. The solution locating a plant in a large lake or sparsely populated area would be of little or no value in a practical problem.

Assume then that the possible locations for sources are limited to some finite set of points. It is then of no significance to express the location of the consumers in terms of Cartesian coordinates, but rather the distances between the consumers and the possible source locations are of interest. The transportation costs will certainly be a function of distance and not necessarily straight line distances, as is often assumed, but physical distances over which the carrier must travel. In view of the previous discussion, the problem to which this thesis is devoted can now be stated.

Statement of the Problem

Given: (a) a set $C = \{C_j | C_j \in C \text{ for } j = 1, 2, \ldots, n\}$ of n consumer points, C_j,

(b) a set $S = \{S_i | S_i \in S \text{ for } i = 1, 2, \ldots, m\}$ of m possible points, S_i, at which a source can be located,

(c) the demand; d_j $j = 1, 2, \ldots, n$, for some commodity

at C_j,

(d) the distance, x_{ij}, from any S_i to any C_j

(e) the transportation cost, t_{ij}, of providing one unit of d_j from S_i per unit distance and

(f) the fixed cost, f_i, of operating a source at S_i.

Find: (a) the number of sources required,

(b) the location in S of each source,

(c) the capacity, K_i, required at each source and

(d) the C_j allocated to each source.

Such that: (a) each d_j is satisfied and

(b) the total cost of operation is minimized.

This problem can be considered a single-stage location problem as opposed to a multi-stage location problem. As Kuehn and Hamburger (19) treated a warehouse location problem, the problem as stated here is in fact a subset of a bigger problem. Their warehouse locations are determined as a function of the plants' locations supplying the warehouses as well as the locations of the consumers drawing on the warehouses. This is in fact treating two distribution stages, i.e., from plant to warehouse and from warehouse to consumer. Only the single-stage problem will be considered here.

As alluded to earlier, even the single-stage problem can take on a varied structure. Of particular interest is the general problem in which all consumer locations are considered as possible source locations. It follows that in this case, S equals C; i.e., $S_i = \delta_{ij}C_j$ for i = 1, 2, ..., m and j = 1, 2, .. , n and where δ_{ij}

is the Kronecker delta defined by:

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases}$$

Clearly, it follows that m = n for this case.

Hakimi (17) has conceptualized a similar problem in a communication system context as a weighted graph. Maranzana (25) has also drawn this analogy in an approach to the location problem minimizing transport costs with some fixed number of warehouses or "supply points." Figure 1 illustrates a weighted graph of the general location-allocation problem in which S = C.

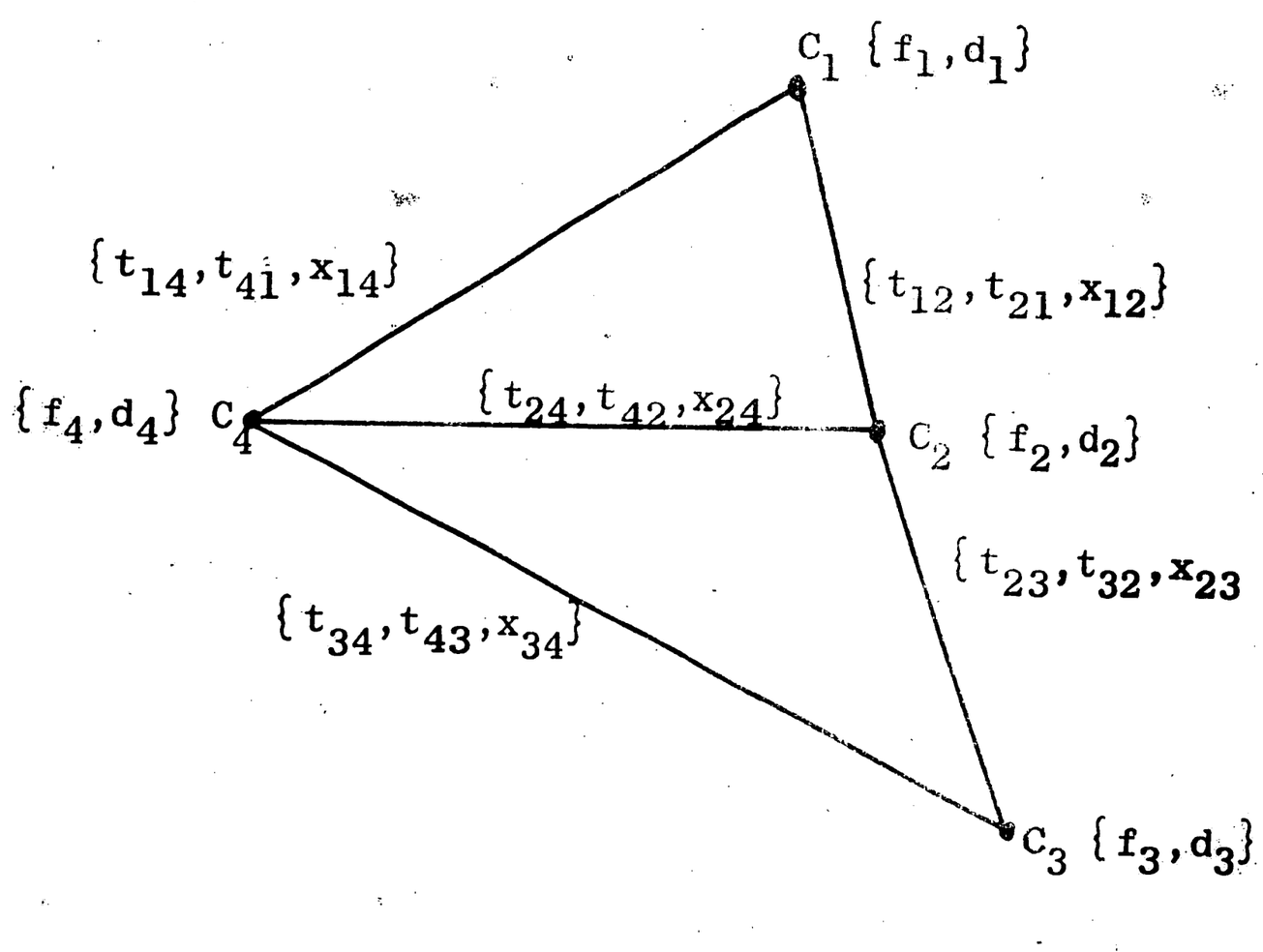


FIGURE 1 A Weighted Graph Representation of the Location-Allocation Problem

Associated with each vertex, analogous to the C_i, is a set of weights corresponding to the demand, d_i, at i and the fixed cost, f_i, of establishing a source at i. Associated with each edge, analogous to the transportation links, is a set of weights corresponding to the per unit transportation costs, t_{ij} and t_{ji}, and the distance, x_{ij}, from i to j. Some observations that should be made about Figure 1 are: the per unit transportation costs need not necessarily be equal on the same edge; an edge's length may or may not represent its relative actual distance but rather x_{ij} is the measure of distance; and finally, there need not be an edge between all pairs of vertices, i.e., the graph need not be "complete" (7) but it must be connected.

Given this weighted graph, the problem remaining is one of determining at which vertices sources should be established and the consumers allocated to each source. Before reviewing the prior approaches to obtaining a solution to this problem in hopes of benefiting from these attempts, mention will be made of some variations that will be considered. Firstly, the set S can reasonably be a subset of C in a practical problem. Secondly, the set C may be a subset of S. This will increase the problem's dimension but it may be a valid situation which will be considered.

III A REVIEW OF THE LITERATURE

To exhaustively review all the literature pertaining to the numerous ramifications of the location-allocation problem would require a volume in itself. Consequently, this chapter will be primarily concerned with the literature related to the problem stated in the previous chapter. One might ask, "What does this exclude from review?" To answer this question, a brief discussion of some related works follows.

Aside from the purely mathematical literature, the location problem has been treated from many fronts. Haley (18) has treated the multi-stage location problem through the use of a mechanical analog. The analog consists of a set of k + n pulleys; k plants and n consumers, located on a vertical plane corresponding to their relative locations. Strings are passed over the pulleys. One end of each string supports a weight proportional to the demand or output; i.e., demand for consumer and output for plant, times the transportation costs per unit per distance unit. The ends of the strings are tied together and released coming to rest with the knot at the minimum cost location for a single distribution center. The assumption here is that transportation costs are linear with distances and that the friction in the system can be ignored. Burstall, et. al. (6), independently of Haley and almost simultaneously, used the mechanical analog to determine the near optimum locations of a pair of factories. They extended the analog solution with subsequent comparison by digital computer of several alternatives obtained from the analog.

Eiseman (14) has presented an identical mechanical analog to the single location problem. Eilon and Deziel (13) present the solution to a single warehouse location problem as well as a two warehouse location problem utilizing an electronic analog computer to obtain the solution.

Closely related to the approaches utilizing a mechanical analog to obtain the single optimum location for a warehouse is the concept of determining the "center" of the demand system by mathematical means. Smykay and Fredericks (28) compute a "ton-mile center" for a system of consumers by computing the "pull" in two orthogonal directions from a reference point. The "ton-mile center" will yield the minimum transportation costs assuming straight line transportation links and linear costs with quantity. Hakimi (17) presents a computational technique to determine an m-median of a graph which minimizes the total interconnecting wire lengths if the graph represents a communication switching system. Unfortunately, his technique consists of completely enumerating all combinations of m points selected from the n consumers. For large n, the enumeration is prohibitive from a computational standpoint. Maranzana (25) determines the "center of gravity of a set of weighted nodes" in a heuristic algorithm to locate a fixed m number of distribution points to minimize transportation costs. The "center of gravity" turns out to be equivalent to the "median" found by Hakimi.

Kuehn and Hamburger (19) have treated the multi-stage location problem through the use of a heuristic program. As stated earlier,

the program attempts to minimize transportation costs from k fixed plants to m warehouses, the location and number of which is to be determined, and the transportation costs to n consumers from the m warehouses while simultaneously minimizing the fixed warehouse costs. Baumol and Wolfe (4) treated the same problem from a mathematical programming approach. The procedure consists of repeated solution of the transportation algorithm until no further cost improvements are realized and the process is terminated.

This excursion into related problems was by necessity not all encompassing. It may nevertheless be fruitful to peruse these related problems for imbedded in their methodologies may be the tools needed to solve the problem at hand. More closely associated with the problem of concern here are the following papers.

Cooper (9), in "Heuristic Methods for Location-Allocation Problems," continues his study of the location-allocation problem presented in Chapter II by applying heuristic methods. Heuristics, as referred to by Simon (27), are rules of thumbs selected on the basis that they will aid in problem solving. Cooper attempts to determine the effect of applying some heuristics to reduce the computational effort involved in solving equations 5. Recall that the solution to equation 5 gives the locations for m sources as (X_i, Y_i), $i = 1, 2, \ldots, m$. Cooper reaffirms results obtained in (8) indicating that the solution obtained from equations 5 will be closely approximated by enumeration of all possible combinations of the n demand points taken m at a time, $\binom{n}{m}$, and selecting the minimum cost

combination as the optimal solution. This then supports the contention made earlier. The enumeration is not trivial, however. He reports that the computation of $\binom{60}{4}$ took $3\frac{1}{2}$ hours on an IBM 7072 computer. The next "heuristic" investigated is the error in sampling from the $\binom{n}{m}$ combinations and selecting the smallest of these as the optimal solution. As an empirical rule, the sampling was stopped when a value between $1\frac{1}{2}$ to 2 standard deviations below the mean and the samples taken up to that time had been obtained. It was found that the mean percent error obtained by sampling to solve 100 problems was 2.518% from the optimal solutions obtained by solving equations 5.

Cooper further investigates the error in determining the optimal location of 2 sources and adding additional sources until m sources are located. On the same 100 problems, the mean percent error was 7.086%. Finally, the n consumer locations are divided into m subsets and a source is optimally located within each of the m subsets. The results for this heuristic approach yielded a mean percent error of 2.582%. Of significance in these results are the small errors found. Cooper states that in the location-allocation problems he has experimented with, a relatively flat minimum was observed. It should be noted, however, that all the computational techniques involved only the optimal location of a fixed m sources.

Manne (23) formulates the single stage, location-allocation problem in mathematical programming terms. In the context of a plant location problem, let

p_{ij} = annual rate of manufacture at source i for shipment to consumer j,

y_i = fraction of fixed costs incurred for operation of a source at i,

t_{ij} = transport cost/unit from source i to consumer j,

c_i = manufacturing cost/unit at source i,

$b_{ij} \triangleq c_i + t_{ij}$,

f_i = fixed annual cost incurred for a source at i,

d_j = annual demand at consumer j.

Then the total system costs are given by:

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

subject to:

$$\sum_{i=1}^{m} p_{ij} = d_j \qquad j = 1, 2, \ldots, n$$

$$p_{ij} \begin{cases} = 0 \text{ if } y_i = 0 \\ \geq 0 \text{ if } y_i = 1 \end{cases} \qquad i = 1, 2, \ldots, m$$

$$y_i = 0, 1 \qquad i = 1, 2, \ldots, m$$

Although the system of constraint equations resembles the transportation problem, there are two important differences:

1. The y_i must be set equal to one to allow $p_{ij} > 0$ in the i th group. The transportation algorithm cannot be used unless $y_i = 1$, for i = 1, 2, ..., m. This clearly is not the aim of this formulation.

2. There are no capacity limitations on the sources. This again is not ameniable to solution by the transportation algorithm.

The first difference is then merely a restatement of the inability of the transportation algorithm to minimize Z. Recall, the transportation algorithm is designed to minimize transportation costs or some cost function C, where

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} p_{ij}$$

Manne utilizes an adaptation of SAOPMA (steepest-ascent one-point move algorithm), introduced by Sherman and Reiter (26), to obtain a feasible solution to the problem. SAOPMA is a heuristic rule beginning with a random selection of some m source locations and makes "one-point moves" until no further economic advantage is realized.

For example, consider the case when j = 4. Implicit in this formulation is the fact that S = C. Then the possible combinations of y_i's can be considered as lattice points of a unit hypercube in the y_i space. The lattice point (0110) would represent sources at i = 2 and i = 3. The "one-point move" requires evaluating the cost of any adjacent lattice point and shifting to the lowest cost point. The adjacent lattice points for (0110) are (1110), (0111), (0010) and (0100). If at any of these points Z<Z at (0110), the Z for the adjacent lattice points to that point are evaluated and the process is continued until no improvement is obtained at any of the adjacent

points. Clearly, the solution will often be a local minimum. Manne found the error from true optimum for a number of problems to be approximately 4.5%. Of significance in Manne's approach to the location-allocation problem are: the mathematical programming formulation; the heuristic rule applied to solve a seemingly unsolvable, from a computational standpoint, mathematical programming problem; and the small percent error obtained with SAOPMA.

Efroymson and Ray (12) formulate the single-stage, location-allocation problem as a mixed-integer programming problem similar to Manne's formulation.

Let

d_j = the demand at consumer j,

t_{ij} = transportation cost/unit from source i to consumer j,

$b_{ij} = t_{ij}\, d_j$ = the total transportation costs to supply consumer j from source i,

p_{ij} = the fraction of d_j supplied from source i,

f_i = the fixed cost of operating source i,

N_j = the set of sources that can supply consumer j,

P_i = the set of consumers that can be supplied from source i,

n_i = the number of consumers in P_i.

The problem is then stated as

$$\min Z = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{ij} p_{ij} + \sum_{i=1}^{m} f_i y_i$$

subject to:

$$\sum_{i \epsilon N_j} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

$$\sum_{j \epsilon P_i} p_{ij} \leq n_i y_i \qquad i = 1, 2, \ldots, m$$

$$p_{ij} \geq 0 \qquad i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n$$

$$y_i = 0, 1 \qquad i = 1, 2, \ldots, m$$

In the terminology used in Chapter II, this formulation will solve the general problem if and only if:

$$N_j = S \qquad j = 1, 2, \ldots, n,$$

and $$P_i = C \qquad i = 1, 2, \ldots, n;$$

the latter of which implies that:

$$n_i = n \qquad i = 1, 2, \ldots, n.$$

The solution of this mathematical program is not trivial. Efroymson and Ray utilize a "branch-and-bound" algorithm. It will suffice to note here that unlike SAOPMA, a branch-and-bound algorithm guarantees a global minimum solution to the mathematical program. Efroymson and Ray's formulation of the location-allocation problem and its subsequent solution by a branch-and-bound algorithm will be reviewed in more detail in later portions of this paper.

IV THE BRANCH-AND-BOUND METHOD

As pointed out earlier in this paper, one approach to the solution of combinatorial problems is the enumeration of all possible combinations defining the solution space. Problems of industrial importance, however, are usually of such dimensions that this approach becomes infeasible from computational aspects. A "better mousetrap" must be devised. The branch-and-bound method is such an improvement.

Like dynamic programming, the branch-and-bound method is an intelligently structured search for the optimum solution in the feasible solution space of a particular problem. The concepts of "branching" and "bounding" are utilized to repeatedly partition the feasible solution space into smaller and smaller subsets and a bound is determined for each solution in a particular partitioned subset. That subset with the most favorable bound is further partitioned until a feasible solution is obtained that is more favorable than any previously obtained subset bound. Favorable is used here to denote that the bound is either the smallest or largest bound obtained depending on whether the problem is one of minimizing or maximizing some objective function. Lawler and Wood (22) have recently published a survey of branch-and-bound applications in which diverse types of constrained optimization problems are cited. Their paper includes an extensive bibliography on the subject. To demonstrate the essential features or concepts of the branch-and-bound method, the following discussion, closely paralleling that of Lawler and Wood, is presented.

Suppose it is desired to solve a "difficult" constrained optimization problem. Consider "difficult" to mean that there is no efficient technique to solve the problem directly. Let the problem, Z, be:

$$\text{minimize} \qquad Z = g(p)$$

$$\text{subject to} \qquad g_1(p) \geq 0$$

$$g_2(p) \geq 0$$

$$\vdots \qquad \vdots$$

$$\text{and} \qquad g_m(p) \geq 0$$

$$p \in P,$$

where P denotes the feasible solution space and p denotes a vector $(p_1, p_2, \ldots, p_n)$. A solution vector p is a feasible solution if and only if $p \in P$ and satisfies the constraints. The optimal feasible solution, p_o, occurs when g(p) is minimal.

Assume now that a related "easy" problem, Z', exists. Let this problem be:

$$\text{minimize} \qquad Z' = g'(p)$$

$$\text{subject to} \qquad g_1'(p) \geq 0$$

$$g_2'(p) \geq 0$$

$$\vdots \qquad \vdots$$

$$\text{and} \qquad g_m'(p) \geq 0$$

$$p \in P.$$

Assume further that the following condition holds:

$$\min Z' = g'(p_o) \leq \min Z = g(p_o)$$

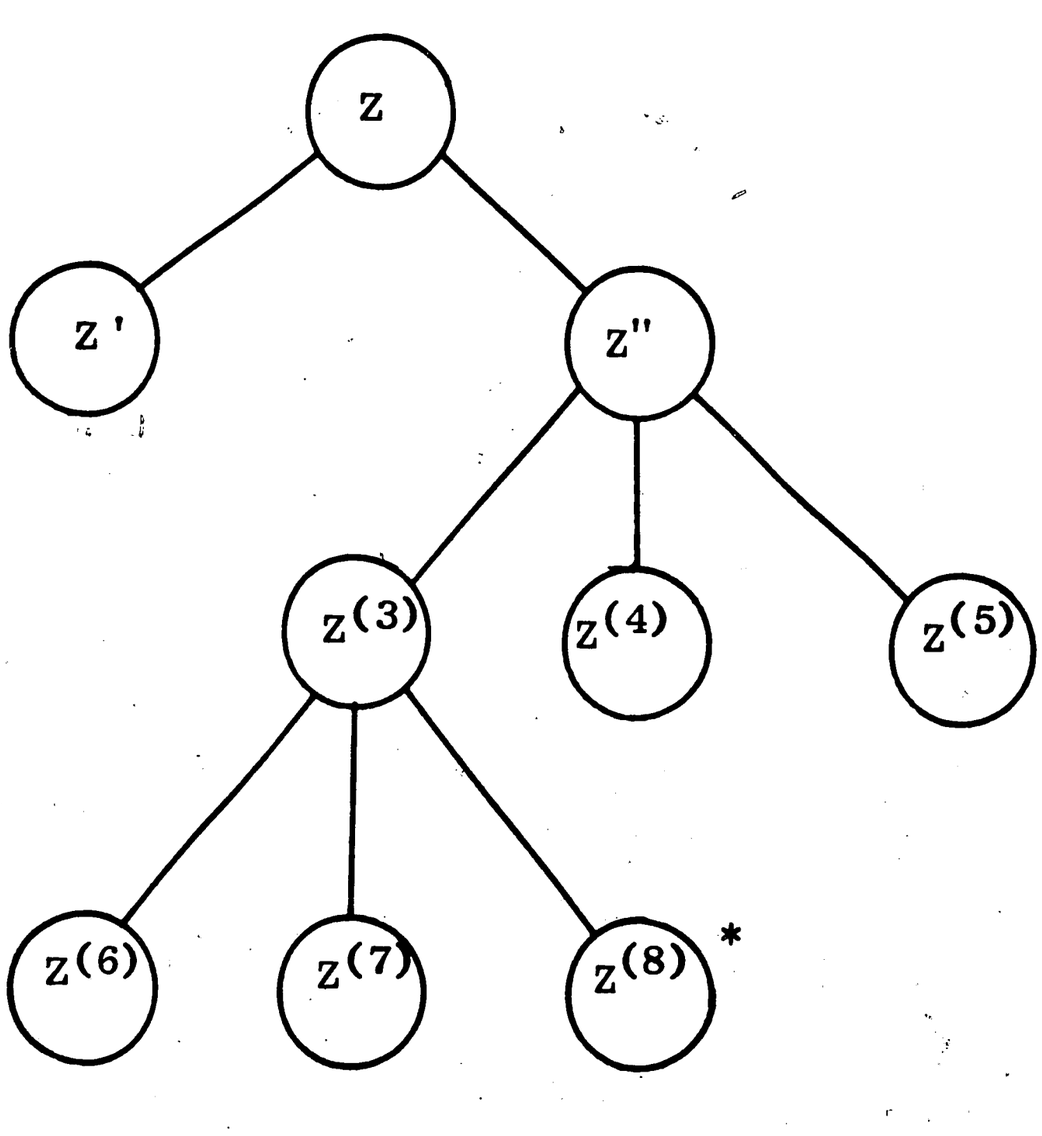
This is then a "bound" on min Z. It follows that if p_o is a feasible solution to Z' and

$$g'(p_o) = g(p_o)$$

then p_o is the optimal solution to Z. It is then possible to create a set Z', Z",..., $Z^{(n)}$ of bounding problems which for some $Z^{(i)}$ yields p_o as a feasible solution to Z and in addition

$$g^{(i)}(p_o) = g(p_o).$$

This is then the optimal solution to Z. If the structure of $Z^{(i+1)}$ is a function of the solutions of Z', Z",..., $Z^{(i)}$; i.e., the bounds previously obtained, the process of finding the optimal solution to Z can be represented as a "tree." Figure 2 is a tree representation of the branch-and-bound-method.



*Denotes Optimal Solution

Figure 2 A Branch-and-Bound Tree

Each node represents a problem. The branches emanating from a node $Z^{(i)}$ replace $Z^{(i)}$ in the bounding set. Consequently, at a particular stage in the computation, the "free" nodes which have not been branched from are the bounds of Z. At each stage branching occurs from the smallest node (Z is still a minimization problem) creating new bounds. If at any stage i, p_o is a feasible solution to Z and all bounds are greater than $g^{(i)}p_o$, the optimal solution to Z has been obtained and

$$\min Z = g(p_o) = \min Z^{(i)} = g^{(i)}(p_o)$$

For example, tracing through the tree in Figure 2, node Z is representative of the solution of Z which is to be found. Nodes Z' and Z'' are the solution of two bounding problems. Branching from Z'' implies that $Z'' < Z'$. Further, the following relationships can be concluded:

$$Z^{(3)} < \min(Z^{(4)}, Z^{(5)}, Z')$$

$$Z^{(8)} < \min(Z^{(6)}, Z^{(7)}, Z', Z^{(4)}, Z^{(5)})$$

and $$\min Z^{(8)} = g^{(8)}(p_o) = \min Z = g(p_o)$$

If, for example, the following condition had existed, i.e.,

$$Z' < \min(Z^{(3)}, Z^{(4)}, Z^{(5)})$$

branching should have occurred from Z'.

The amount of computation is clearly a function of the number of nodes in the tree. If the "easy" problem solved at each node does not bound the problem sufficiently well, the tree may grow to represent all feasible solutions of Z which is in fact a complete enumeration. The number of nodes which have not been branched from

will determine the amount of intermediate storage required. That is, it may be necessary to branch from a node determined early in the process so it must be stored. In short, the branch-and-bound method's usefulness and efficiency is a function of the particular algorithm used to construct the bounding problems at each node and the structure of the "difficult" problem as well.

Agin (2) defines a branch-and-bound algorithm as "A set of rules for

1. branching from nodes to new nodes,
2. determining lower bounds for the new nodes,
3. choosing an intermediate node from which to branch next,
4. recognizing when a node contains only infeasible or non-optimal solutions and
5. recognizing when a final node contains an optimal solution."

From the discussion of the branch-and-bound method and the definition given above, it follows that the concepts of branching and bounding can be applied to a myriad of problems. It should also be recognized that unlike the simplex algorithm in linear programming, a branch-and-bound algorithm does not follow a common procedure for all problems. It must be structured to the "difficult" problem at hand.

Land and Doig (21) consider the solution of mixed-integer programming problems by the branch-and-bound method. Their work has subsequently been extended by Beale and Small (5), Dakin (10), and Driebeck (11). The approach to the mixed-integer problem is

(22): Solve the mixed problem by the simplex algorithm. If the solution obtained has integral values for the required variables, the optimum solution has been obtained. If not, restrict some non-integral "integer" variable to the next lower and next higher integral value and apply the simplex algorithm for each case. This process is repeated until the solution is feasible and no intermediate node is more favorable. Of particular interest here is the mixed-integer programming problem in which the integer variables are Boolean variables, i.e., of the zero or one type.

Consider the solution of the following problem.

$$\text{Minimize} \qquad Z = g(p,y)$$

$$\text{subject to} \qquad g_1(p,y) \geq 0$$

$$g_2(p,y) \geq 0$$

$$\vdots \qquad \vdots$$

$$g_m(p,y) \geq 0,$$

$$\text{and} \qquad p \in P$$

$$y \in Y$$

where P = the feasible solution space for p,

Y = the feasible solution space for y,

$p = (p_1, p_2, \ldots, p_n)$

$y = (y_1, y_2, \ldots, y_n)$

$g(p,y), g_1(p,y), \ldots, g_m(p,y)$ = linear functions in p and y.

In particular, let

$p = (p_1, p_2)$,

$y = (y_1, y_2, y_3)$,

P = the positive real numbers

and Y = the integers 0 or 1.

There are then $2^3 = 8$ different solution points satisfying Y or in the SAOPMA terminology, eight lattice points of the unit hypercube in 3-space. Associated with each of these lattice points is an "easy" bounding problem, the solution of which may be infeasible with respect to P. The "easy" problem is a solution of the linear programming problem defined at that lattice point, i.e., the problem with the y_i constrained to equal one of the eight feasible solutions in Y.

One approach to the solution of Z would then be to solve the eight problems and select the minimum solution. Let the problem at a lattice point by $Z(y_1, y_2, y_3)$ where in particular Z(0,1,1) denotes the problem with the y_i's constrained as: $y_1 = 0$, $y_2 = 1$, $y_3 = 1$. The tree of all possible problems is then represented in Figure 3.

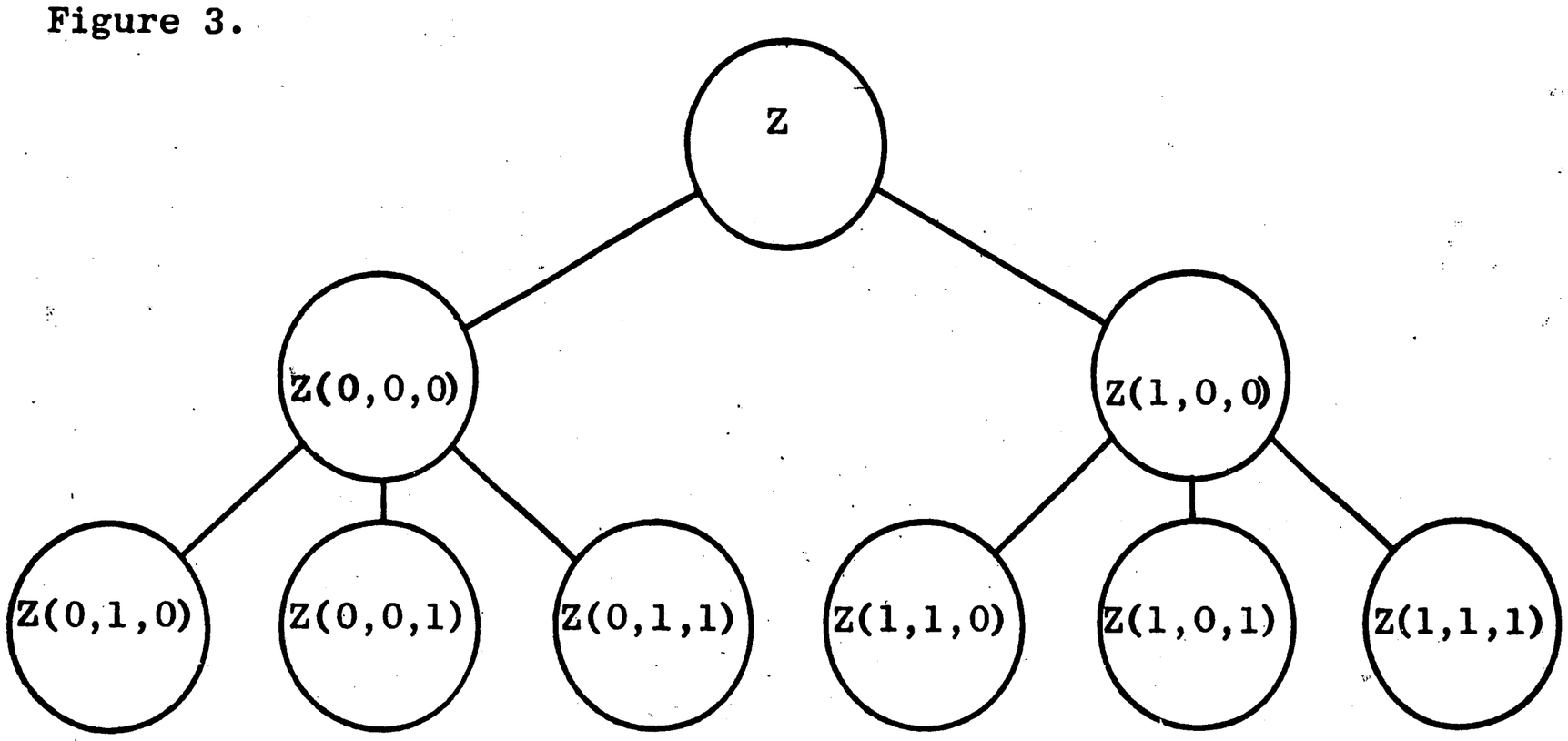


Figure 3 An Enumeration Tree

For a large number of y_i's, this is not computationally feasible. If $y = (y_1, y_2, \ldots, y_{15})$, the number of lattice point problems that exist is $2^{15} = 32,768$.

Consider a branch-and-bound approach. Let the initial problem be the continuous version of the mixed-integer problem denoted by Z(-,-,-) where the dashes indicate that none of the y_i's are constrained to be integers. It follows from the theory of linear programming (16) that the absolute minimum for Z will be the solution of Z(-,-,-). Construct two new problems based on the values of the y_i's in the solution of Z(-,-,-). Let these problems constrain some non-integer y_i to zero and one. Using the solutions to these problems as new bounds, branch from the lowest bound and create two more problems. This process will continue until all y_i's are integers and no lower bound exists which has not been branched from. This is the optimal feasible solution to Z. If at some node the solution is infeasible, the subset defined by that node is no longer considered for further partitioning.

Of importance to this approach is the structure of Z. If Z is structured to drive the y_i's to integral values at early stages of the tree growth, the tree size may be small when an optimum solution is obtained. Conversely, with a "poor" structure from the viewpoint of introducing integral values early, the tree may grow to a complete enumeration tree. As pointed out by Agin (2), the decision rules applied to create the new problems branching from a node can be a deciding factor in the size of the tree.

V A BRANCH-AND-BOUND APPROACH

This chapter is addressed to the development of a mathematical formulation of the location-allocation problem, including the implicit assumptions, and the branch-and-bound algorithm for its solution. As alluded to in prior discussions, these elements of the branch-and-bound method are not disjoint from the standpoint of computational efficiency.

The Mathematical Formulation

Recalling the problem statement in Chapter II, a weighted graph; G, Figure 1, defines the problem requiring solution. G can be represented in matrix form to facilitate further development. Consider first the general problem in which S = C.

Let X be an n x n matrix representing the distance between any two vertices on the graph. Then

$$X = [x_{ij}] \text{ where}$$

$$x_{ij} = \begin{cases} x_{ij} \text{ if } C_i \text{ and } C_j \text{ are adjacent vertices} \\ 0 \text{ if } C_i = C_j \\ \infty \text{ if } C_i \text{ and } C_j \text{ are not adjacent vertices} \end{cases}$$

Recall that $x_{ij} = x_{ji}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$.

Obtain now the shortest distance between any pair of vertices in G. This is not a trivial problem. Some approaches to the problem are presented in (15). The discussion of this problem is outside the scope of this paper and it is therefore assumed that a technique is used to obtain the shortest paths.

Let the matrix $\overline{X}$ represent the shortest paths obtained.

Then

$$X = [\overline{x}_{ij}] \text{ where}$$

$$\overline{x}_{ij} = \begin{cases} \overline{x}_{ij} \text{ if } i \neq j \\ 0 \text{ if } i = j. \end{cases}$$

$\overline{x}_{ij}$ is then the shortest path connecting C_i and C_j.

Let

$$D = (d_1, d_2, \ldots, d_n)$$

be a vector representing the d_j at each C_j. Further, let the transportation costs be represented by:

$$T = [t_{ij}] \text{ where}$$

$$t_{ij} = \begin{cases} t_{ij} \text{ if } i \neq j \\ 0 \text{ if } i = j, \end{cases}$$

and t_{ij} is the cost to transport one unit of d_j from S_i per unit distance. The total transportation costs to provide d_j from S_i can then be given by:

$$B = [b_{ij}] \text{ where}$$

$$b_{ij} = \overline{x}_{ij}\, d_j\, t_{ij}$$

The remainder of the graph consists of F, the vector of fixed costs for operation of a source S_i at i. In the former notation,

$$F = (f_1, f_2, \ldots, f_n).$$

The essential costs are then represented in matrix B and vector F.

It is noteworthy at this point that B will have zeros on the diagonal

which merely implies that the satisfaction of d_j from a source at j will not require transportation costs. This assumption does not appear unreasonable when considered with respect to the transportation costs t_{ij} when $i \neq j$. Also implicit in the foregoing development is the linearity of transportation costs with distance and quantity. The validity of this assumption will be a function of the particular application. It will here be considered to apply in all cases.

Continuing with the development, let p_{ij} be the fraction of d_j supplied from S_i and let y_i represent the fraction of a source at i. Minimizing the operating costs of the system will again require the minimization of fixed costs and transportation costs.

The fixed costs are:

$$Z_1 = \sum_{i=1}^{n} f_i y_i$$

The transportation costs can be expressed as:

$$Z_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} p_{ij}$$

and the total operating costs are then:

$$Z = Z_1 + Z_2 = \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} p_{ij}$$

Z is identical to the total cost expression obtained by Efroymson and Ray.

To insure that each d_j is satisfied, a constraint must be imposed

on the minimization process. Since p_{ij} is the fraction of d_j supplied from S_i, the sum of the p_{ij} for each j must equal one, i.e.,

$$\sum_{i=1}^{n} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

Further, consider $y_i \leq 1$; i.e , a maximum fraction equalling a complete source. It is required to constrain the number of consumers that can be served by a source. This constraint is expressed as:

$$\sum_{j=1}^{n} p_{ij} \leq ny_i \qquad i = 1, 2, \ldots, n$$

It is unreasonable, from a practical standpoint, to consider y_i to be continuous in the interval 0 to 1. Consequently, require that

$$y_i = 0, 1 \qquad i = 1, 2, \ldots, n$$

That is, there either does or does not exist a source at i. The above set of constraints is identical to the set imposed by Efroymson and Ray if their $P_i = S$ and $N_j = C$ for all i and j.

Consider the solution of this mixed-integer programming problem by a branch-and-bound method. Using the approach indicated in Chapter IV, the procedure begins with the minimization of Z with the p_{ij} and y_i considered to be continuous, i.e., in the former notation, Z(-,-,..,-) is solved. Subsequent nodes involve the solution of some $Z(I_0, I_1, I_2)$ where I_0 denotes the set of i's for the y_i's that have been fixed at zero, I_1 denotes the set of i's

for the y_i's that have been fixed at one and I_2 denotes the set of i's for the y_i's not in I_0 or I_1. Then in Z(-,-,... -); $I_0 = 0$, $I_1 = 0$ and $I_2 = 1, 2, \ldots, n$.

In the optimal solution at some node, note that for the $i \in I_2$ the constraint

$$\sum_{j=1}^{n} p_{ij} \leq n y_i$$

will be a strict equality. Then

$$\sum_{j=1}^{n} p_{ij} = n y_i$$

or

$$\sum_{j=1}^{n} \frac{p_{ij}}{n} = y_i$$

Substituting this result into Z, the problem at a node is

$$\min Z = \sum_{i \in I_1} f_i + \sum_{i \in I_2} f_i \sum_{j=1}^{n} \frac{p_{ij}}{n} + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} p_{ij}$$

$$= \sum_{i \in I_1} f_i + \sum_{i \in I_2} \sum_{j=1}^{n} \frac{f_i}{n} p_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} p_{ij}$$

$$= \sum_{i \in I_1} f_i + \sum_{i \in I_1} \sum_{j=1}^{n} b_{ij} p_{ij} + \sum_{i \in I_2} \sum_{j=1}^{n} \left(\frac{f_i}{n} + b_{ij}\right) p_{ij}$$

Since for the $i \in I_1$; $y_i = 1$, the first term is a constant at a node and the minimization process is concerned with the remaining terms.

It is then desired to find

$$\min Z' = \sum_{i \epsilon I_2} \sum_{j=1}^{n} \left(\frac{f_i}{n} + b_{ij}\right) p_{ij} + \sum_{i \epsilon I_1} \sum_{j=1}^{n} b_{ij}\, p_{ij} \qquad \text{eq. 9}$$

Recalling the constraint that

$$\sum_{i=1}^{n} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

the minimum for Z' will result when

$$p_{ij} = \delta_{ij} \qquad i \epsilon I_2$$

unless for some $j \epsilon I_2$ and $i \epsilon I_1 \cup I_2$,

$$b_{ij} < \frac{f_j}{n} \qquad \text{ineq. 10}$$

If, in fact

$$b_{ij} > \frac{f_j}{n}$$

then

$$b_{ij} > \frac{f_j}{n} + b_{jj} \qquad \text{since } b_{jj} = 0 \text{ for all } j$$

and the minimum solution will be:

$$p_{jj} = 1 \qquad \text{for all } j \epsilon I_2$$

In addition, for any $j \epsilon I_0$, the following must hold:

$$\sum_{i \epsilon I_1 \cup I_2} p_{ij} = 1 \qquad j \epsilon I_0$$

It follows from equation 9 that

$$p_{ij} = \begin{cases} 1 \text{ if } b_{ij} + (g_i/n) = \min\limits_{i \in I_1 \cup I_2} \left\{ b_{ij} + (g_i/n) \right\} \\ 0 \quad \text{otherwise} \end{cases}$$

where

$$g_i = \begin{cases} f_i & \text{if} \quad i \in I_2 \\ 0 & \text{if} \quad i \in I_1 \end{cases}$$

Then, if n' is the number of j's allocated to $i \in I_2$,

$$\sum_{j=1}^{n} p_{ij} = n' \qquad i \in I_2$$

or

$$y_i = \frac{n'}{n} \qquad i \in I_2$$

unless inequality 10 holds for all $i \in I_2$. What does this result mean in terms of the branch-and-bound method? Simply that for all $i \in I_2$ at a node, the solution of the linear program at that node yields the same I_2 unless $n' = n$. In most practical situations, $n' < n$. It would be advantageous to reduce I_2 at each node so that I_2 approaches the null set; or equivalently, that all y_i's have taken on integer values.

Consider the addition of another constraint set. Let this constraint require that a source be at i before a consumer at i can be served from i. That is, require that

$$p_{ii} \leq y_i \qquad i = 1, 2, \ldots, n$$

The following will then be true in the optimal solution at a node:

$$\sum_{j=1}^{n} \frac{p_{ij}}{n} = y_i \qquad i \epsilon I_2$$

and

$$p_{ii} = y_i \qquad i \epsilon I_2$$

Substituting the latter equality into the objective function yields the following problem at a node:

$$\min Z = \sum_{i \epsilon I_1} f_i + \sum_{i \epsilon I_1} \sum_{j=1}^{n} b_{ij} p_{ij} + \sum_{i \epsilon I_2} f_i p_{ii} + \sum_{i \epsilon I_2} \sum_{j=1}^{n} b_{ij} p_{ij}$$

As before, the problem reduces to:

$$\min Z' = \sum_{i \epsilon I_1} \sum_{j=1}^{n} b_{ij} p_{ij} + \sum_{i \epsilon I_2} f_i p_{ii} + \sum_{i \epsilon I_2} \sum_{j=1}^{n} b_{ij} p_{ij}$$

or

$$\min Z' = \sum_{i \epsilon I_1} \sum_{j=1}^{n} b_{ij} p_{ij} + \sum_{i \epsilon I_2} \sum_{j=1}^{n} (\delta_{ij} f_i + b_{ij}) p_{ij} \qquad \text{eq. 11}$$

with the constraint that

$$\sum_{i=1}^{n} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

still in the problem. Recall that the minimum solution to equation 9 was

$$p_{ij} = \delta_{ij} \qquad i \epsilon I_2$$

unless the inequality 10 held. This result does not however follow in equation 11. For as the p_{ii} is increased to satisfy d_i with

$b_{ii} = 0$, Z' is increased by $f_i p_{ii}$ as opposed to the increase of $\frac{f_i}{n} p_{ii}$ in equation 9. Although it is difficult to show analytically, this will generally reduce I_2 at the initial node and consequently approach the optimal feasible solution to Z more rapidly.

The mixed-integer programming problem is then:

$$\min Z = \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} p_{ij}$$

subject to:

$$\sum_{i=1}^{n} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} p_{ij} \leq n y_i \qquad i = 1, 2, \ldots, n$$

$$p_{ii} \leq y_i \qquad i = 1, 2, \ldots, n$$

$$y_i = 0, 1 \qquad i = 1, 2, \ldots, n$$

Note that if S is a proper subset of C; for any $S_i \notin C$, let $f_i = \infty$. Further, if C is a proper subset of S; for any $C_i \notin S$, let $d_i = 0$.

In the optimal feasible solution to the problem, the $i \in I_1$ will be the locations of sources and all $i \notin I_1$, i.e., all $i \in I_0$, will be supplied by some S_i where $i \in I_1$. For any $p_{ij} > 0$, i will be in I_1 and j will be in I_0. This then determines the allocation of the C_j to the S_i. It remains then to determine the required capacities for the S_i. Clearly, the necessary capacity K_i at S_i will be:

$$K_i = \sum_{j=1}^{n} p_{ij} d_j \qquad i = 1, 2, \ldots, n$$

An additional observation regarding the optimal feasible solution to Z will be made at this point.

It is noted that the optimal solution to Z will be all integer, i.e., the p_{ij}'s as well as the y_i's will be either zero or one in the optimal solution. The proof of this result follows: note that in the optimal feasible solution, $I_2 = 0$. The objective function at the node is then

$$\min Z' = \sum_{i \epsilon I_1} f_i + \sum_{i \epsilon I_1} \sum_{j=1}^{n} b_{ij} p_{ij}$$

subject to:

$$\sum_{i \epsilon I_1} p_{ij} = 1 \qquad j = 1, 2, \ldots, n$$

Let $I_1 = \{i_1, i_2\}$. Then by the constraint above

$$p_{i_1 j} + p_{i_2 j} = 1 \qquad j = 1, 2, \ldots, n$$

or

$$p_{i_1 j} = 1 - p_{i_2 j} \qquad j = 1, 2, \ldots, n$$

Recall that for a point P to be the optimal feasible solution to Z obtained by the simplex method, it must be an extreme point of the convex solution set. P is an extreme point of the convex set if and only if there do not exist other points P^1, P^2; $P^1 \neq P^2$, in the set such that $P = \lambda P^1 + (1-\lambda) P^2$, $0 < \lambda < 1$ (16). It follows then that for P to be the optimal solution to Z,

$$p_{i_1 j} = 0 \text{ or } 1 \qquad j = 1, 2, \ldots, n.$$

Clearly, this is the only solution not obtainable from a convex combination of two other points. The problem as formulated is then actually an all-integer programming problem with Boolean variables. From the standpoint of a branch-and-bound algorithm, this result is ignored because it only applies when $I_2 = 0$. It is of interest, however, that this integer programming problem allows solution by the branch-and-bound method when considering it a mixed-integer programming problem. The algorithm capitalizing on this behavior is discussed in the next section of this chapter.

The Branch-and-Bound Algorithm

Recalling the definition of a branch-and-bound algorithm stated in Chapter IV, the algorithm used to solve the mathematical program of the previous section will be described by stating the rules and the rationale behind them for the five elements of the algorithm. The terminology and notation will be that of the previous section and Chapter IV. Some repetition will occur on those points previously discussed in generalities for the sake of continuity. Appendix A contains the algorithm in flow diagram form. Consider, now the five elements in the definition.

Determination of Bounds - The bound at a node will be obtained by solving a linear program of the problem at that node. Clearly, any solution in the subset defined by a node can be no less than the bound obtained from the linear program. The problem at the node is defined by I_0, I_1, and I_2. Similarly, the solution at the node can be defined by the value of Z and some new I_0', I_1', and I_2' obtained from the values of the y_i's in the solution. The I_0', I_1' and I_2' at

a node will be used to determine the problems at the nodes branching from that node.

Branching from Nodes to New Nodes - Following the approach in (21), there will be two nodes, $Z^{(i+1)}$ and $Z^{(i+2)}$, branching from the most favorable node $Z^{(j)}$. In $Z^{(i+1)}$ constrain the maximum y_i; $y_{i(max)}$, for $i \epsilon I_2'$ at $Z^{(j)}$, equal to zero and in $Z^{(i+2)}$ constrain $y_{i(max)}$ to one. Recall that in equation 11 all $y_i = p_{ii}$ for $i \epsilon I_2'$. Since $b_{ii} = 0$ for all i, a large $p_{ii} = y_i$ indicates that the transportation costs to supply i from some other j are high and also that there exists some j for which b_{ij} is small. Consequently, let the largest y_i, $i \epsilon I_2'$, be the first integer variable integerized.

Choosing the Next Node to Branch from - Agin (2) discusses three alternatives for selecting the node from which branching continues. The first alternative, the one used here, branches from the "free" node with the smallest bound found so far. The rationale behind this alternative is that the optimal feasible solution is most likely contained in the subset defined by that node. This approach favors nodes created early in the process where few of the y_i's have been assigned integers. It is therefore important that early nodes force integers into the solution so that the bounds are not close to the absolute minimum obtained from the solution of the initial node. The second alternative proposed in (2) is to continue branching from the smallest of $Z^{(i+1)}$ and $Z^{(i+2)}$. For the location-allocation mixed-integer program, this will generally mean that branching continues from $Z^{(i+1)}$; i.e., the node with $y_{i(max)} = 0$.

This may lead to the evaluation of several nodes in a subset which does not contain the optimal solution before branching occurs to a node bounding a different subset. The third alternative consists of branching from the smallest of $Z^{(i+1)}$ and $Z^{(i+2)}$ until all nodes are either infeasible, their bounds exceed a known solution, or $I_2 = 0$ for the last node in the branch. The next node is chosen by tracing back up the tree to the first node $Z^{(i)}$ in which $Z^{(i+1)}$ is not infeasible if $Z^{(i+2)}$ was the node branched from the first time $Z^{(i)}$ was branched from. This approach suffers from the same disadvantage as the second alternative.

Recognizing When a Subset Contains Only Infeasible or Non-Optimal Solutions - The problem of infeasibility does not arise with the location-allocation formulation, for there always exists a feasible solution to the problem. Recognizing that only non-optimal solutions are possible in a given subset occurs when a subset is not further partitioned because the bound is greater than a feasible solution obtained.

Recognizing an Optimal Solution - An optimal solution is obtained when $I_2' = 0$ and no bound exists which is less than the solution at the node with $I_2' = 0$.

Given this set of rules for the branch-and-bound algorithm, it is of interest to determine the computational efficiency of the algorithm. Recognizing that certain of the rules are arrived at through the use of heuristics, the computational results must be the final evaluators. This topic is the theme of the next chapter.

VI COMPUTATIONAL RESULTS

The study of the computational aspects of the branch-and-bound algorithm was centered on the following areas:

1. Comparison of the solution times for the Efroymson and Ray formulation vs. the formulation developed here and
2. Solution times for problems with varying n

Before discussing the results obtained, the programs used for the study are briefly described.

The Computer Programs

The branch-and-bound algorithm and various support subroutines were programmed in Fortran IV and run on an IBM System 360/50 with 256,000 bytes of main memory. A listing of the programs is contained in Appendix B. The linear programming subroutine is a simplex algorithm which begins with all slack or artificial variables in the initial basis. The $y_{i(max)}$ is constrained to zero or one by making its coefficient in the L.P. matrix a large positive or negative number. The functional value and the values for the y_i for free nodes are stored in a main memory file which allows the storage of up to 150 nodes. When branching occurs from a node in the file, a large positive value is assigned to the functional at that node to preclude branching from this node in the future.

Subroutine CREATE was used to generate the problems used in the computations. The problem generation consists of sampling from uniform distributions for the elements in $\bar{X}$, D and F. The

t_{ij} were defined as:

$$t_{ij} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{if } i = j. \end{cases}$$

Since for large n, $b_{ij} > \frac{f_j}{n}$ will generally hold, the uniform distributions sampled from were chosen to guarantee the above inequality. The mean of the distribution for F was chosen to yield approximately 20 to 33% of the consumer locations as source locations in the optimal solution. In addition, five subroutines were written to determine the optimum locations by complete enumeration as suggested in (5).

The branch-and-bound program is presently restricted to a maximum problem size of n = 15 due to core storage requirements. Because of this restriction, the program is not suited for most problems of industrial importance. Significant programming will be required, however, to increase the limit on n. If in the solution at a node a y_i differed from an integer by no more than 10^{-4}, it was set identically equal to the integer. A total of 30 problems were generated and solved by the programs to arrive at the results discussed in the following sections.

Solution Time Comparison

It was originally decided to study the difference in solution times of the Efroymson and Ray formulation and the formulation developed here by performing an analysis of variance on the solution times of some number of problems of a fixed size. Consequently,

10 problems with n = 10 were generated for solution with both formulations. After attempting to solve the first three of the problems using the Efroymson and Ray formulation, the contention made in Chapter V that the algorithm will not allow efficient solution of their formulation was strongly supported and the analysis of variance was deemed unnecessary. As shown in Chapter V, I_2 does not reduce by more than $y_{i(max)}$ at each stage. This may lead to a complete enumeration. On all three problems attempted with their formulation a solution had not been obtained when the algorithm was stopped after filing the 150th free node in the file. If the algorithm degenerates to a complete enumeration, the solution will involve evaluating $2^n + 1$ nodes. If, however, I_2 is reduced early in the process the total number of possible problems will be greatly reduced. Note that to this point solution times have only been expressed in terms of the number of nodes evaluated. The reason for this is that the solution time will be a machine dependent variable whereas the number of nodes is machine independent. Recalling the additional n constraints in this paper's formulation over the Efroymson and Ray mixed-integer program, the node evaluation times differed. The Efroymson and Ray model required .213 minutes to evaluate a node as compared to .449 minutes for the proposed formulation for a problem with n = 10. Because of this increase in node evaluation time, the number of nodes for the Efroymson and Ray formulation must be at least twice the number for the other formulation to gain an advantage in computing time. In light of

this, the following is presented.

Consider the solution obtained at the initial node where $I_2 = 1, 2, \ldots, n$. The number of possible problems or nodes resulting from this node will be a function of the size of I_2 in the solution. For example, if $n = 10$, $2^n + 1 = 1025$; i.e., the algorithm may degenerate to solving 1025 linear programs to obtain the optimal feasible solution. If in the solution at the first node, only five of the y_i's are in I_2', the number of possible problems reduces to 32. Figure 4 shows a portion of the branch-and-bound tree obtained with the algorithm using the Efroymson and Ray formulation and Figure 5 shows the tree obtained with the proposed formulation for the same problem. Tables 1 and 2 present the results obtained at the nodes for Figures 4 and 5, respectively.

It can be seen from Figure 4 and Table 1, that with the branch-and-bound algorithm used here, the process appears to be degenerating into a complete enumeration. On some problems with $n = 4$, the solution with their formulation did in fact require complete enumeration where with the proposed formulation the optimum feasible solution was often obtained after evaluating only the initial node. It should be pointed out, however, that this is a comparison of the formulations solved by this branch-and-bound algorithm and not the algorithm used by Efroymson and Ray. Although their algorithm differs in determining which y_i to integerize at each stage, the size of the tree depends on reducing I_2 at each stage and as was shown earlier this does not occur by solving their linear programming formulation at a node. The results obtained for each of the problems

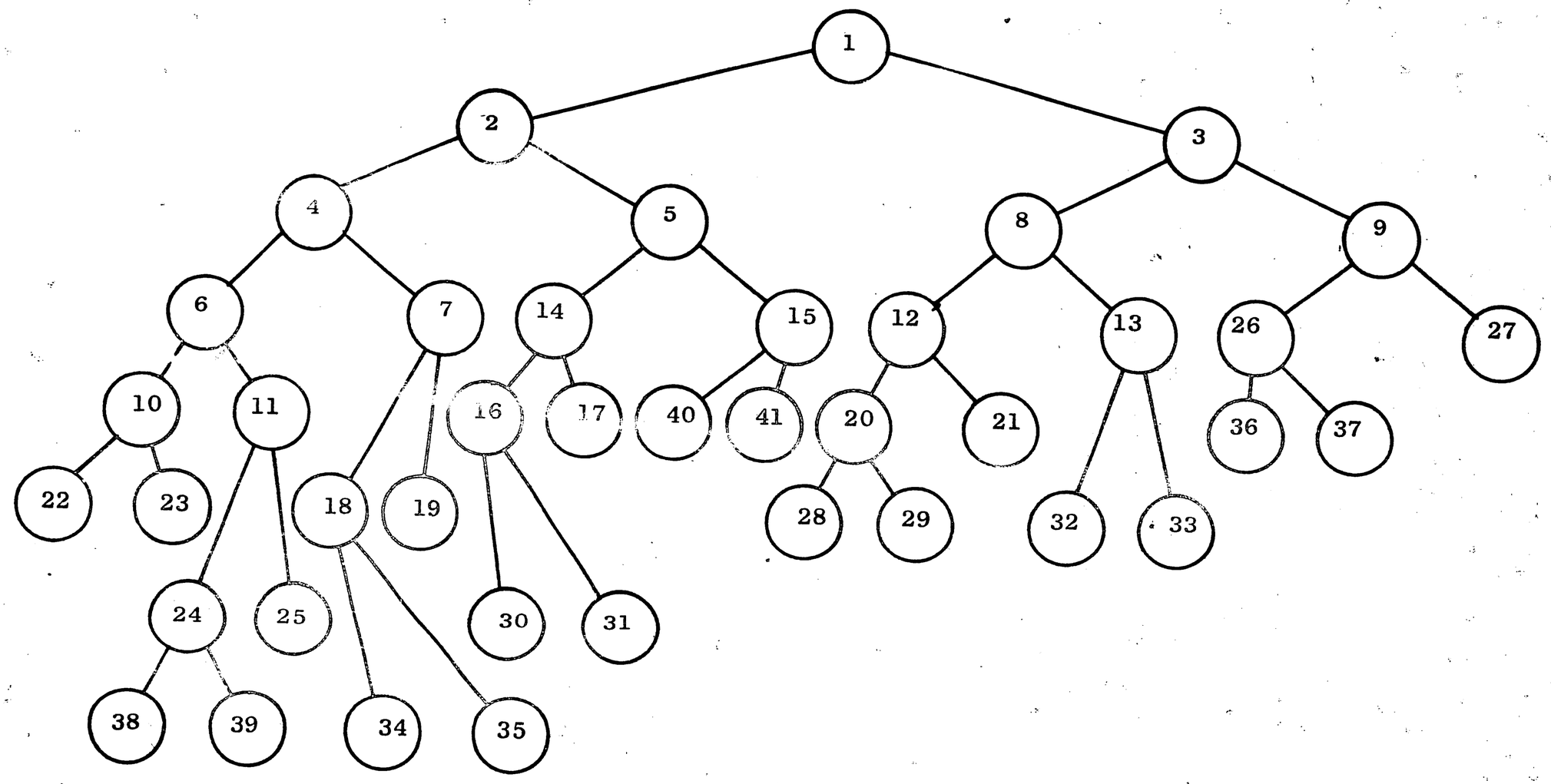


Figure 4 A Portion of the Tree Obtained with The Efroymson and Ray Formulation

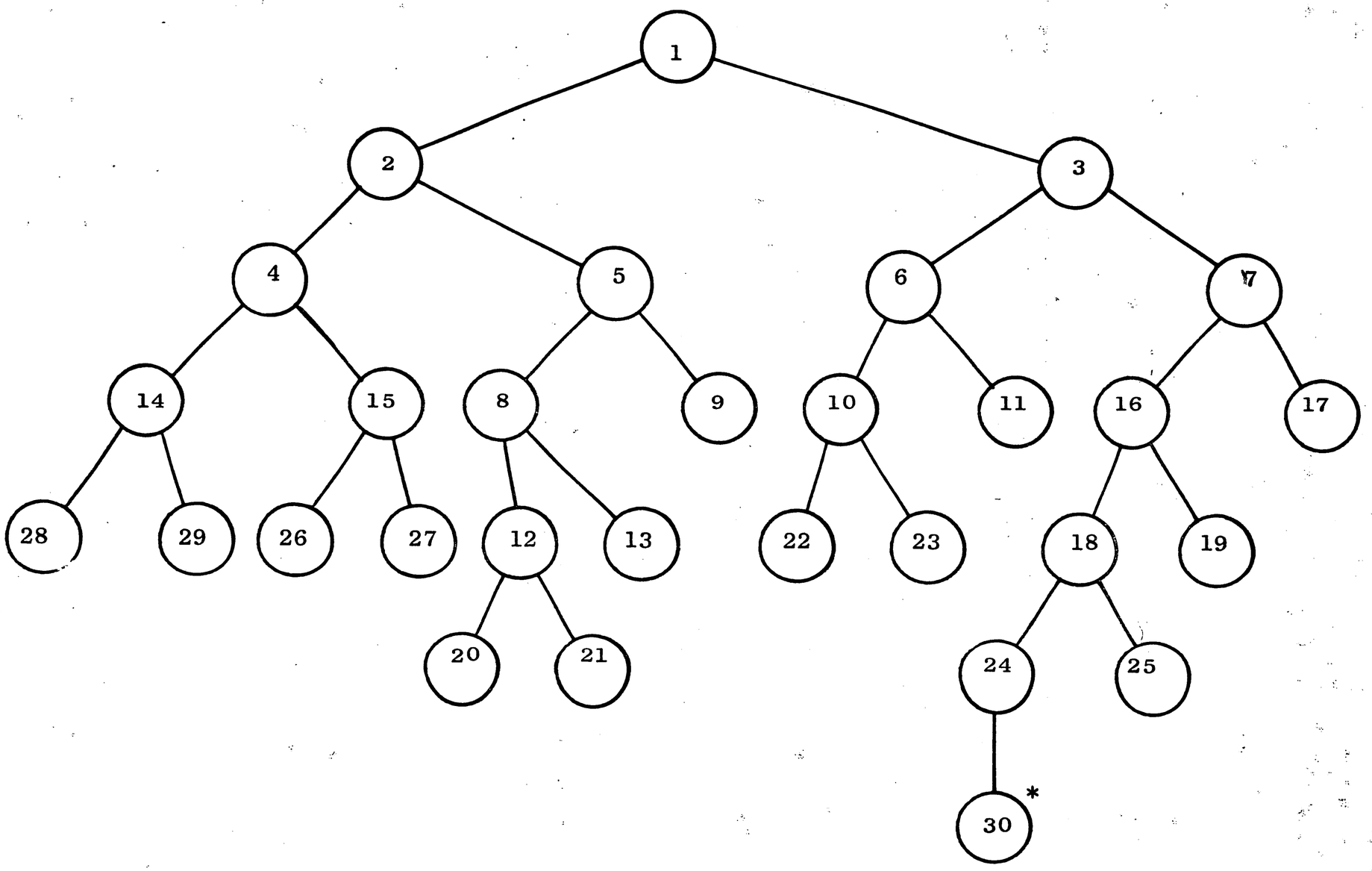


Figure 5 The Tree Obtained with the Proposed Formulation

Node	Z	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1	369.4	,10	.10	.10	.10	.10	.10	.10	.10	.10	.10
2	523.8	0.0	.10	.20	.10	.10	.10	.10	.10	.10	.10
3	722.4	1.0	.10	.10	.10	.10	.10	.10	.10	.10	.10
4	651.1	0.0	.20	0.	.10	.10	.20	.10	.10	.10	.10
5	840.8	0.0	.10	1.0	.10	.10	.10	.10	.10	.10	.10
6	763.1	0.0	0.0	0.0	.10	.10	.20	.20	.20	.10	.10
7	941.8	0.0	1.0	0.0	.10	.10	.20	.10	.10	.10	.10
8	831.2	1.0	0.0	.10	.10	.10	.10	.10	.20	.10	.10
9	1049.3	1.0	1.0	.10	.10	.10	.10	.10	.10	.10	.10
10	973.8	0.0	0.0	0.0	.10	.20	0.0	.20	.30	.10	.10
11	1044.8	0.0	0.0	0.0	.10	.20	1.0	.20	.20	.10	.10
12	950.1	1.0	0.0	.20	.10	.10	.10	.10	0.0	.20	.10
13	1111.3	1.0	0.0	.10	.10	.10	.10	.10	1.0	.10	.10
14	914.6	0.0	0.0	1.0	.10	.10	.10	.10	.10	.10	.10
15	1167.8	0.0	1.0	1.0	.10	.10	.10	.10	.10	.10	.10
16	1082.6	0.0	0.0	1.0	0.0	.10	.10	.10	.20	.10	.10
17	1238.8	0.0	0.0	1.0	1.0	.10	.10	.10	.10	.10	.10
18	1118.3	0.0	1.0	0.0	.10	.10	0.0	.10	.20	.10	.10
19	1223.6	0.0	1.0	0.0	.10	.10	1.0	.10	.10	.10	.10
20	1071.8	1.0	0.0	0.0	.10	.10	.10	.10	0.0	.20	.10
21	1267.0	1.0	0.0	1.0	.10	.10	.10	.10	0.0	.20	.10
22	1171.7	0.0	0.0	0.0	.10	.30	0.0	.30	0.0	.20	.10
23	1218.9	0.0	0.0	0.0	.10	.20	0.0	.20	1.0	.10	.10

TABLE 1. The Results at the Nodes of Figure 4

Node	Z	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
24	1145.2	0.0	0.0	0.0	.10	.20	1.0	0 0	.20	.10	.20
25	1361.1	0.0	0.0	0.0	.10	.10	1.0	1.0	.20	.10	.10
26	1130.7	1.0	1.0	0.0	.10	.10	.10	.10	.10	.10	.10
27	1405.7	1.0	1.0	1.0	.10	.10	.10	.10	.10	.10	.10
28	1181.2	1.0	0.0	0.0	.20	.10	.10	.10	0.0	0.0	.10
29	1386.5	1.0	0.0	0.0	.10	.10	.10	.10	0.0	1.0	.10
30	1214.1	0.0	0.0	1.0	0.0	.10	.10	.10	0.0	.30	.10
31	1362.8	0.0	0.0	1.0	0.0	.10	.10	.10	1.0	.10	.10
32	1192.8	1.0	0.0	0.0	.10	.10	.10	.10	1.0	.10	.10
33	1467.7	1.0	0.0	1.0	.10	.10	.10	.10	1.0	.10	.10
34	1197.8	0.0	1.0	0.0	.10	.10	0.0	.20	0.0	.10	.10
35	1398.6	0.0	1.0	0.0	.10	.10	0.0	.10	1.0	.10	.10
36	1298.8	1.0	1.0	0.0	0.0	.10	.10	.10	.20	.10	.10
37	1454.7	1.0	1.0	0.0	1.0	.10	.10	.10	.10	.10	.10
38	1303.3	0.0	0.0	0.0	.10	0.0	1.0	0.0	.20	.20	.20
39	1418.4	0.0	0.0	0.0	.10	1.0	1.0	0.0	.20	.10	.20
40	1335.8	0.0	1.0	1.0	0.0	.10	.10	.10	.20	.10	.10
41	1491.7	0.0	1.0	1.0	1.0	.10	.10	.10	.10	.10	.10

TABLE 1 (continued)

Node	Z	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1	1456.6	.200	.075	.200	0.00	0.00	0.00	.100	.325	0.00	.100
2	1530.6	.176	.111	.412	0.00	0.00	0.00	.212	0.00	0.00	.088
3	1600.0	.200	0.00	.200	0.00	0.00	0.00	.100	1.00	0.00	.100
4	1765.9	.543	.111	0.0	0.00	0.00	0.00	.264	0.00	0.00	.088
5	1657.6	.111	.111	1.00	0.00	0.00	0.00	.222	0.00	0.00	0.0
6	1729.2	0.00	0.00	.333	0.00	0.00	0.00	.175	1.00	0.00	.092
7	1766.3	1.00	0.00	.111	0.00	0.00	0.00	.100	1.00	0.0	.100
8	1752.9	.111	.222	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
9	1888.1	.111	.111	1.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
10	1813.5	0.00	0.00	0.00	0.00	0.00	0.00	.200	1.00	0.00	.200
11	1870.2	0.00	0.00	1.00	0.00	0.00	0.00	.111	1.00	0.00	0.00
12	1812.1	.722	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	1950.0	.111	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
14	1851.8	0.00	.444	0.00	0.00	0.00	0.00	.375	0.00	0.00	.181
15	1835.7	1.00	.111	0.00	0.00	0.00	0.00	.212	0.00	0.00	.088
16	1790.6	1.00	0.00	0.00	0.00	0.00	0.00	.100	1.00	0.00	.100
17	1988.2	1.00	0.00	1.00	0.00	0.00	0.00	.111	1.00	0.00	0.00
18	1835.5	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	.111
19	2031.3	1.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00
20	1944.0	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
21	1987.0	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
22	1877.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	.222
23	2007.5	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	.111
24	1852.0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
25	1989.0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00
26	1896.9	1.00	.111	0.00	0.00	0.00	0.00	0.00	0.00	0.00	.222
27	2046.7	1.00	.111	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
28	2082.4	0.00	0.00	0.00	0.00	0.00	0.00	.750	0.00	0.00	.750
29	1917.6	0.00	1.00	0.00	0.00	0.00	0.00	.325	0.00	0.00	.075
Soln	1852	1.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00

TABLE 2. Results at the Node of Figure 5

with the formulation of this thesis are given in tabular form in Table 3.

Solution Times with Varying n

As indicated earlier, the programs used in this thesis are not applicable to the solution of problems with $n > 15$. Most problems of industrial importance will however require that n is greater than 15. It would be of interest then to predict the amound of computation that would be necessary to solve problems with n large. Toward this end, a second set of 20 problems with varying n were generated and solved by the algorithm. Table 4 presents the pertinent information obtained from the solution of the problems.

To continue the analysis in machine independent terms, an expression for the expected number of nodes that must be evaluated to obtain a solution for a given n will be developed. From Table 4, obtain the fraction of all possible nodes evaluated to obtain a solution. Let the fraction be defined by:

$$Fr(n) = \frac{\text{number of nodes evaluated}}{2^n + 1}$$

Figure 6 is a plot of ln Fr(n) vs. n for the 20 problems of Table 4 and the line fitted to the data by the method of least squares. The equation for the line is:

$$\ln Fr(n) = .836 - .459n \qquad \text{eq. 12}$$

No.	Computing Time in .01 hrs.	No. of Nodes Evaluated	Fraction of 2^n+1 Evaluated	Ave. Node Comp. Time in .01 Hrs.	Total No. of Free Nodes
1	28.5	41	.04000	.695	37
2	21.2	12	.01171	.730	26
3	8.9	11	.01073	.809	9
4	12.4	15	.01463	.820	13
5	11.6	15	.01463	.777	11
6	40.9	59	.05756	.693	54
7	17.2	23	.02244	.748	21
8	27.8	39	.03805	.713	34
9	19.2	25	.02439	.768	22
10	30.4	41	.04000	.741	39

TABLE 3. The Results of 10 Problems with n = 10

No.	n	Computing Time in .01 Hrs.	No. of Nodes Evaluated	Fraction of 2^n+1 Evaluated	Ave. Node Comp. Time in .01 Hrs.	Total No. of Free Nodes
11	4	.2	3	.17647	.0666	1
12	4	.2	7	.41176	.0286	5
13	5	.4	3	.09091	.1333	1
14	5	.8	9	.27273	.0889	6
15	6	1.6	13	.20000	.1231	9
16	6	1.0	7	.10769	.1429	5
17	7	5.8	31	.24031	.1871	28
18	7	3.0	15	.11628	.2000	13
19	8	4.8	17	.06615	.2824	13
20	8	4.9	17	.06615	.2824	13
21	9	14.3	39	.07602	.3667	36
22	9	14.1	35	.06823	.4029	32
23	10	14.5	27	.02634	.5370	24
24	10	8.2	15	.01463	.5467	12
25	11	22.4	29	.01415	.7724	27
26	11	18.8	25	.01220	.7520	22
[illegible]	12	24.5	25	.00610	.9800	20
[illegible]	12	34.0	33	.00805	1.0303	27
29	14	131.8	81	.00494	1.6272	77
30	14	63.2	37	.00225	1.7081	32

TABLE 4. The Results of 20 Problems with Varying n

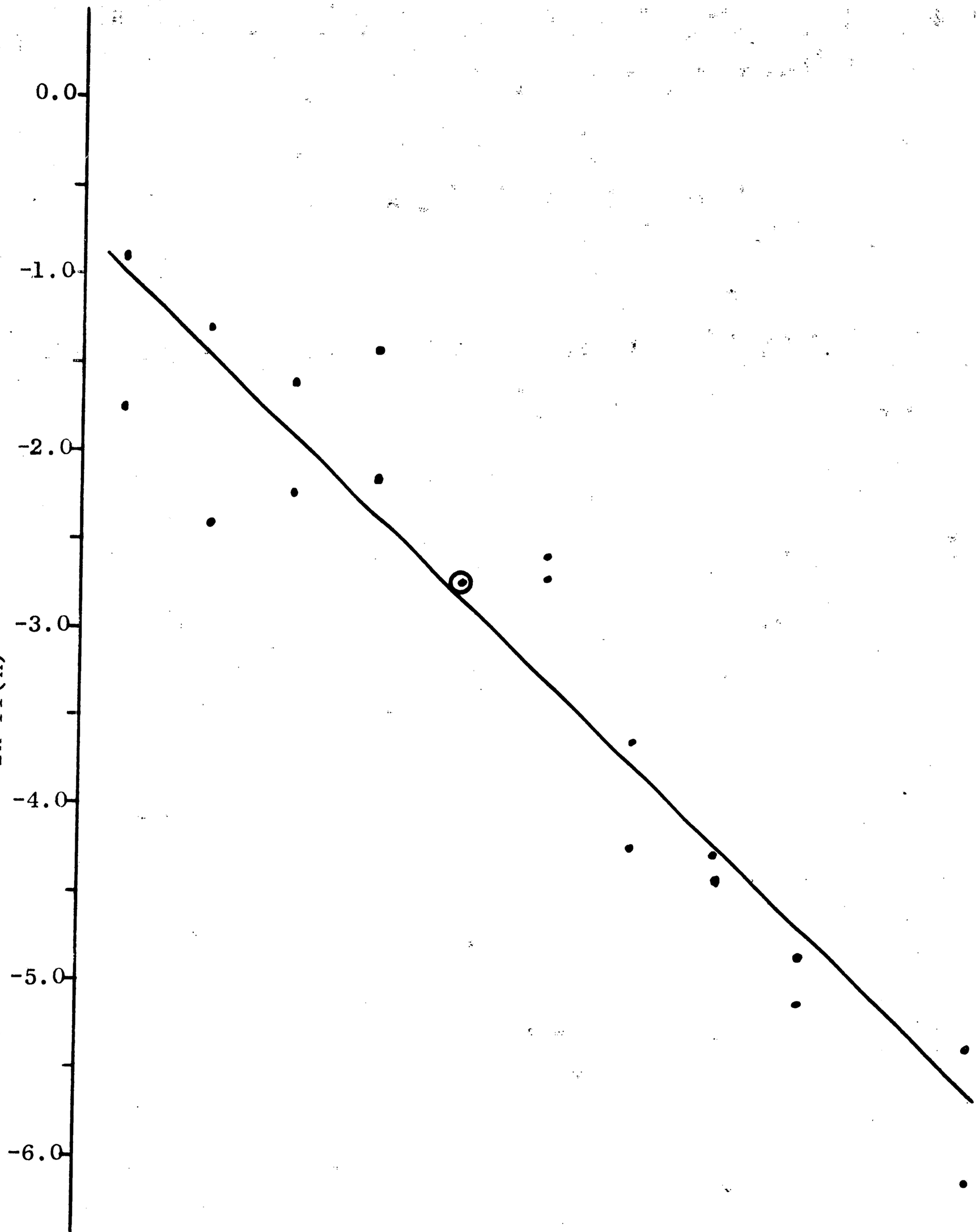


Figure 6 A Plot of ln Fr(n) vs n

or

$$Fr(n) = e^{.836}\ e^{-.459n}$$

and

$$Fr(n) = 2.307\ e^{-.459n}. \qquad \text{eq. 13}$$

From this the expected number of nodes, E(i), that must be evaluated to obtain a solution for some n can be obtained. Obtain:

$$E(i) = Fr(n)\ (2^n + 1)$$

or

$$E(i) = 2.307\left[\ e^{-.459n}\ (2^n) + e^{-.459n}\ \right]$$

or

$$E(i) = 2.307\left[e^{-.459n + .693n} + e^{-.459n}\right]$$

and finally

$$E(i) = 2.307\left[e^{.234n} + e^{-.459n}\right] \qquad \text{eq. 14}$$

Assuming now that the random variables, lnFr(n), are independently distributed with normal distributions having means given by equation 12 and a common variance σ^2, obtain the unbiased estimate of σ^2 and the standard error of the estimate. The unbiased estimate for σ^2 is:

$$S_e^2 = \frac{1}{k-2} \sum_{i=1}^{k} \left[\ln Fr(n)_i - \ln Fr(n)'_i\right]^2$$

where $\ln Fr(n)_i$ = the observed value of lnFr(n)
for the ith n,

$\ln Fr(n)'_i$ = the result obtained from equation 12 and

k = the number of observed pairs.

The standard error of the estimate is given by s_e.

The following were obtained from the data in Table 4:

$$s_e^2 = .2413$$

$$s_e = .4912$$

With the statistic s_e, the limits of prediction for lnFr(n)' can be obtained. As defined in (24), the limits of prediction for $\ln Fr(n_0)'$ are:

$$\ln Fr(n_0)' \pm t_{\frac{\alpha}{2}} s_e \left[1 + \frac{1}{k} + \frac{k(n_0 - \bar{n})^2}{k \sum_{i=1}^{k} n_i^2 - (\sum_{i=1}^{k} n_i)^2} \right]^{\frac{1}{2}}$$

where

α = 1 - the confidence interval,

$t_{\frac{\alpha}{2}}$ = the Student's t statistic with k-2 degrees of freedom

n_0 = the value of n for which lnFr(n) is to be predicted.

Let

$$d(n_o) = t_{\frac{\alpha}{2}} s_e \left[1 + \frac{1}{k} + \frac{k(n_o - \bar{n})^2}{k \sum_{i=1}^{k} n^2 - (\sum_{i=1}^{k} n_i)^2} \right]^{\frac{1}{2}}$$

Then for any n_o, the prediction limits for $\ln Fr(n_o)'$ are:

$$\ln Fr(n_o)' = .836 - .459n \pm d(n_o).$$

Equation 13 can then be rewritten as:

$$Fr(n_o)' = 2.307\ e^{-.459n.} \pm d(n_o).$$

Further it follows that the limits of i' are given by:

$$i' = Fr(n_o)' \ (2^{n_o} + 1).$$

Figure 7 is a plot of i' vs. n for $\alpha = .05$ with the results of the 20 problems plotted on the graph. As can be seen from Figure 7, the E(i) and limits of prediction for i' grow rapidly with n. Consider, however, that for $n = 30$, $2^n + 1 = 10.737\ (10^8)$. The limits of prediction for i' are 373 to 18 (10^3) computed as previously indicated. If the linearity of lnFr(n) is assumed to hold for $n = 30, 3.47\ (10^{-7}) \leq Fr(30) \leq 16.75\ (10^{-4})$. It should also be evident that as n increases, the solution time at a node will increase. In short then, the expected computing time will increase at a rate somewhat higher than expressed in equation 14 due to the increased node evaluation time as n increases.

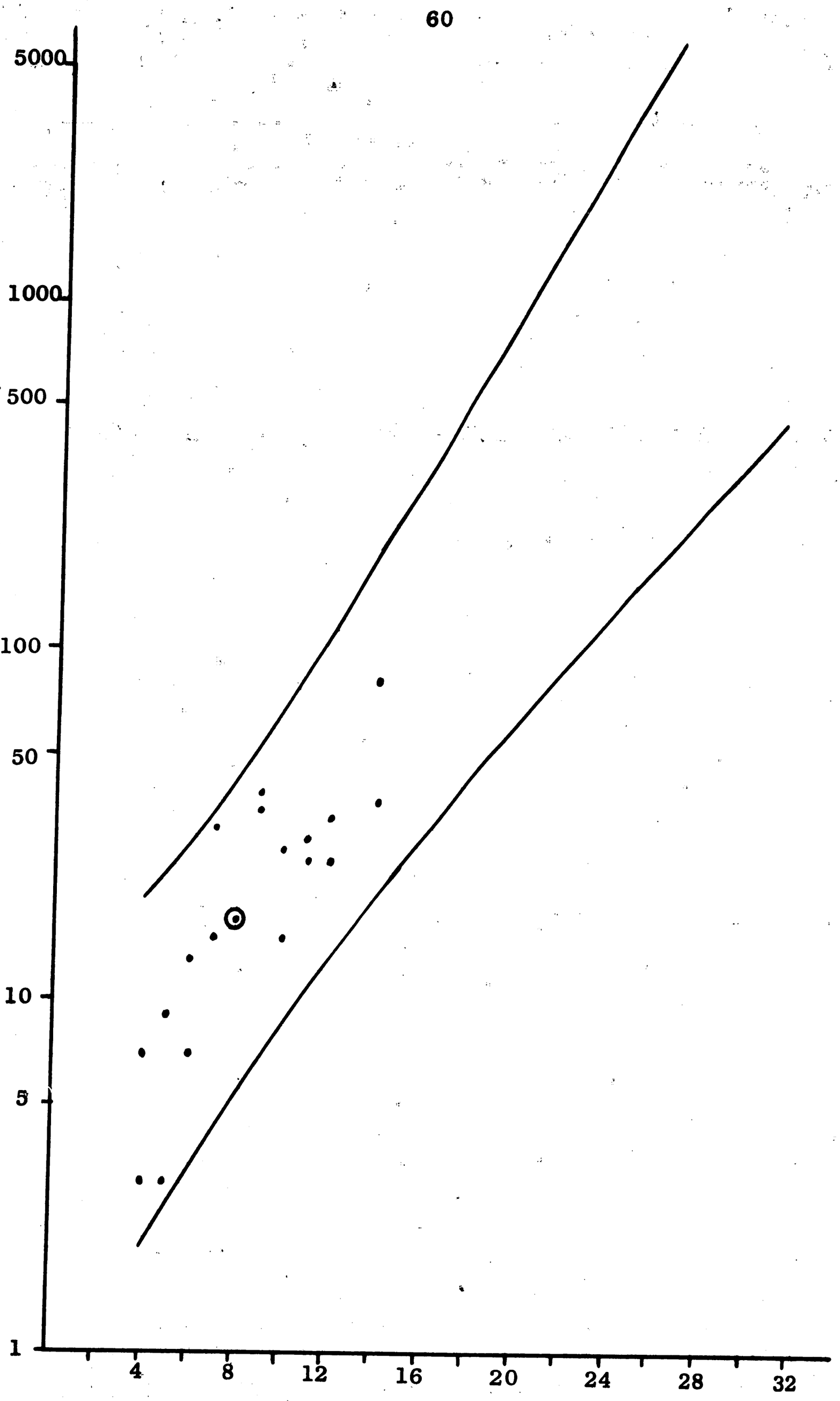


Figure 7 The Prediction Limits of i' With the Results of the 20 Problems

VII CONCLUSIONS

The analytic and experimental work of this thesis has brought to light several interesting results. Although the results were obtained in the context of the general location-allocation problem, as it was defined here, they appear to be applicable to problems unrelated to the thesis topic. The significant conclusions will be discussed from the standpoint of the thesis subject and extensions will be indicated where appropriate.

Consider first the solution of the mixed-integer programming formulation of the general location-allocation problem. The analytic and experimental results unequivocally demonstrated the computational superiority of the formulation developed in this paper over the formulation of Efroymson and Ray. However, a qualification of the preceding statement is in order. This superiority existed with the branch-and-bound algorithm of this paper and it should therefore not be construed to apply to all branch-and-bound algorithms that could be devised to solve the problem. As alluded to in the growing body of literature on applications of the branch-and-bound method, the computational efficiency is a function of both the algorithm and the bounding problem at a node. The results obtained in this thesis strongly support this contention.

The increased computational efficiency obtained with the formulation developed here was a direct result of the additional constraints imposed on the minimization process. These constraints violated none of the assumptions of the model and in fact were

redundant when considering the solution of the mixed-integer program directly. However, from the standpoint of solving the mixed-integer program by repeated solution of some linear programming analog, the results were dramatic. The additional constraints forced the linear program solution to determine integral values for some subset of the integer variables. This result appears to be significant in reducing computational effort for mixed-integer programming problems in general if solved by a similar branch-and-bound algorithm. The additional constraints must, of course, be consistent with the physical problem.

The results obtained from solving problems of various size in an attempt to predict the number of nodes that must be evaluated for a problem of a given size were received with mixed emotions. Over the range of problem sizes investigated, the expected number of nodes that required evaluation for a given size problem followed an exponential function. Fortunately, the function does not increase at the same rate as the total possible nodes for a given problem size. It is nevertheless true, if the expression for the expected number of nodes is assumed to hold, that for a problem with 30 consumer locations, the expected number of nodes requiring solution will be approximately 9193. That is, 9193 linear programs will have to be solved. Even with a problem of this relatively small dimension, the computational effort will be immense. It can only be concluded that this borders on computational infeasibility.

VIII SUGGESTIONS FOR FURTHER STUDY

A branch-and-bound algorithm is at best a set of rules, arrived at through the use of heuristics, that efficiently guide the search for an optimal solution to a particular problem. The computational efficiency of the algorithm depends upon the validity of the heuristics used to devise the necessary decision rules of the procedure. An investigation into this set of rules may prove the algorithm used in this thesis to be relatively inefficient. For example, the number of nodes may be reduced by a different decision rule used to determine the next node to branch from or computations may be reduced by an alternate procedure to determine the y_i to next integerize for new bounding problems. The investigation of these aspects of the branch-and-bound method appears to be seldom done for a particular algorithm.

As demonstrated in the thesis, the optimal feasible solution to the mixed-integer programming formulation is in fact an all-integer solution with Boolean variables. Balas (3) reports the development of an efficient algorithm for problems of this type. It would be of value to determine if improved computing time can be obtained with the algorithm of Balas.

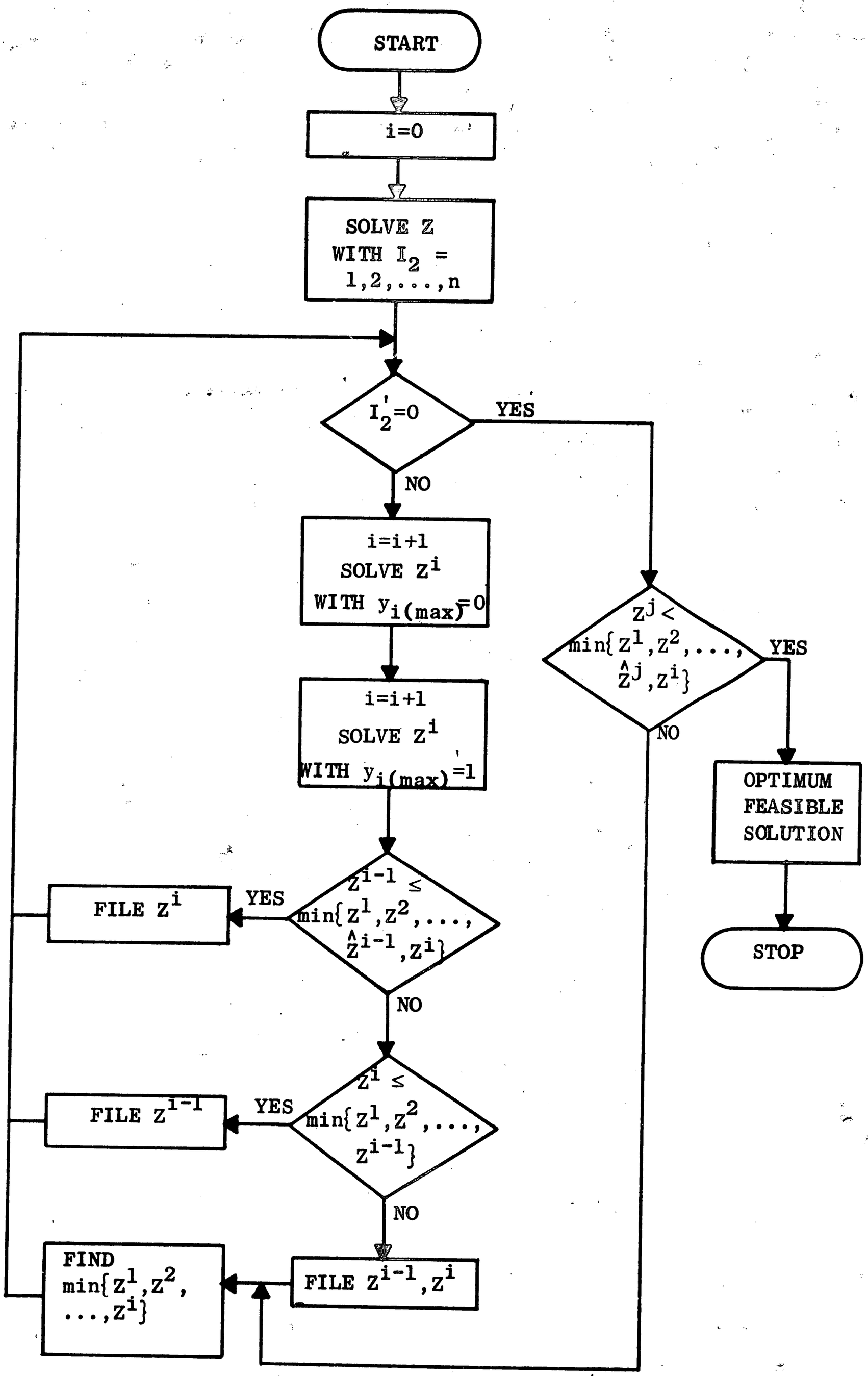
During the computational work of this thesis, it was observed that in almost all problems solved, the branch-and-bound algorithm produced several feasible solutions to the mixed-integer program before the algorithm stopped. Frequently, the optimal feasible solution was found but due to free nodes with lower bounds the algorithm continued until all free bounds were greater than the

feasible solution obtained earlier. The extra computations are clearly necessary to guarantee a global minimum. It would be of interest, however, to study the deviations from the optimal solution obtained with these intermediate feasible solutions.

As pointed out by Cooper (8)(9), location-allocation problems generally exhibit a flat minimum. Intuitively then, an intermediate feasible solution should be close to the optimal solution or in fact the optimal solution itself. What then is the risk associated with stopping the algorithm after one, two or three feasible solutions have been found and selecting the minimum of these? An investigation of this aspect of the algorithm may prove to be of great value in reducing computational effort.

APPENDIX A

Flow Diagram of the Branch-and-Bound Algorithm


START
i=0
SOLVE Z WITH I_2 = 1,2,...,n
I_2'=0
YES
NO
i=i+1
SOLVE Z^i WITH $y_{i(max)}$=0
i=i+1
SOLVE Z^i WITH $y_{i(max)}$=1
$Z^{i-1} \leq \min\{Z^1, Z^2, \ldots, \hat{Z}^{i-1}, Z^i\}$
YES
FILE Z^i
NO
$Z^i \leq \min\{Z^1, Z^2, \ldots, Z^{i-1}\}$
YES
FILE Z^{i-1}
NO
FILE Z^{i-1}, Z^i
FIND $\min\{Z^1, Z^2, \ldots, Z^i\}$
$Z^j < \min\{Z^1, Z^2, \ldots, \hat{Z}^j, Z^i\}$
YES
NO
OPTIMUM FEASIBLE SOLUTION
STOP

APPENDIX B

The Fortran IV Program Listings

```
C     LOCATION-ALLOCATION PROBLEM PROGRAM USING THE BRANCH-BOUND
C     ALGORITHM AND A SIMPLEX LINEAR PROGRAM TO OBTAIN THE BOUNDS AND
C     THE OPTIMAL INTEGER SOLUTION.  UP TO 15 INTEGER VARIABLES AND 60
C     CONSTRAINTS ARE MAXIMAL FOR THIS PROGRAM SIZE.  THE TOTAL NUMBER
C     OF VARIABLES ( INTEGER, CONTINUOUS, AND SLACK ) MUST BE LESS THAN
C     319. THE PROGRAM WILL ACCEPT A PROBLEM INPUT ON CARDS OR CREATE
C     CAN BE CALLED TO GENERATE A PROBLEM FOR EXPERIMENTAL PURPOSES

      COMMON A(62,320),L(62)
      COMMON E(15,15), LOGIC(15,15),F(15), MLOC(30,10)
      DIMENSIONSNODE(16),K0(15),K1(15),B(62,320),L1(62)
      DIMENSION WNODE(16),ZNODE(16),C(150,16)
C
C     DEFINITION OF VARIABLES
C
C     KOUNT=ITERATION OF THE BRANCH AND BOUND ALGORITHM JUST COMPLETED
C     SNODE=ARRAY OF INTEGER  VARIABLES WITH VARIABLE IN QUESTION SET
C           EQUAL TO ONE
C     WNODE=ARRAY OF INTEGER  VARIABLES WITH VARIABLE IN QUESTION SET
C           EQUAL TO ZERO
C     ITER=A BOOLEAN VARIABLE EQUAL TO ZERO IF Y(I)=0 AND EQUAL TO ONE
C          IF Y(I)=1
C     IONE=COLUMN NUMBER OF FIRST INTEGER VARIABLE
C     ILAST=COLUMN NUMBER OF LAST INTEGER VARIABLE
C     NUM=NUMBER OF INTEGER VARIABLES + ONE
C     FUNCT=ADJUSTED FUNCTIONAL VALUE FROM LP SOLUTION
C     TFUNC=TEMPORARY FUNCTIONAL VALUES ADDED TO DRIVE INTEGER VARIABLES
C           TO EITHER ZERO OR ONE
C     SMALL=THE FUNCTIONAL VALUE OF THE BEST NODE FOUND
C     NODE1=THE NUMBER OF THE BEST NODE IN FILE
C     NODE2=THE COUNT OF THE STORED NODES
C     ZNODE=THE STORING NODE
C     K0=ARRAY OF VARIABLES CONSTRAINED  TO EQUAL ZERO
C     K1=ARRAY OF VARIABLES CONSTRAINED  TO EQUAL ONE
C     IVAR= THE INTEGER VARIABLE BEING TESTED AT THIS PASS
C     IGEN=CONTROL VARIABLE FOR PROBLEM INPUT. IF IGEN NE 0, IGEN WILL
```

```
C          BE THE INITIAL VALUE FOR THE RANDOM NUMBER GENERATOR USED IN
C          SUBROUTINE CREATE TO GENERATE PROBLEMS
C     C=THE ARRAY OF NODES STORED DURING THE BRANCH AND BOUND ALGORITHM
C
    1 FORMAT(2I4)
    2 FORMAT(2I4,F20.5)
 3011 FORMAT('1',T50,'PROBLEM NUMBER  ',I5//)
 2040 FORMAT('0',T11,'KOUNT',T19,'ITER',T26,'IVAR',T35,'FUNCTIONAL',T68,
     1'VALUE OF INTEGER VARIABLES')
 2030 FORMAT('0',I12,I8,I7,E19.8,F14.8,4F15.8/' ',45X,5F15.8/' ',45X,
     15F15.8)
 2080 FORMAT('0',T3,'*** FILING NUMBER',I4,T25,'IN FILE',E16.8,F14.8,
     14F15.8/' ',45X,5F15.8/' ',45X,5F15.8)
 8010 FORMAT('          FILE  OVERFLOW                    ')
 2510 FORMAT('0',T3,'***** BRANCHING TO NODE',I4,T31,'IN FILE. SMALLEST
     1BOUND IN FILE IS NODE',I4,T75,'WHICH HAS VALUE OF',E16.8,' *****')
 3020 FORMAT('0',T30,'OPTIMAL SOLUTION ------- FUNCTIONAL EQUALS',E20.8)
 3030 FORMAT(' ',I20,E30.8)
 3040 FORMAT('0',T50,'LOCATIONS',T70,'VALUE')
 3050 FORMAT(' ',I53,F23.8)
 3060 FORMAT(' ',I53,T70,'*****')
 3100 FORMAT('0',T40,'NOTE   ***** DENOTES THAT THIS LOCATION WAS NOT IN
     1 OPTIMAL BASIS')
 3070 FORMAT('0',T52,'ALLOCATIONS')
 3080 FORMAT('0',T40,'FROM',T55,'TO',T70,'VALUE')
 3090 FORMAT(' ',I41,I14,F24.8)
 3110 FORMAT(I5)
C
C*****DATA INITIALIZATION SECTION*****
C
      READ(1,3110)IGEN
      IX=IGEN
 9996 READ(1,1)NUMBER
      IF(NUMBER.EQ.0)GOTO9998
      DO23I=1,100
      DO23J=1,16
```

```
23 C(I,J)=0.
   SMALL=10.**35
   NODE1=0
   NODE2=0
   KOUNT=0
   ITER=0
   KUTOFF=0
   PHONY=90000.0
   TFUNC=0.
   IVAR=0
   FUNCT=0.
   NODE3=0
   DO21I=1,15
   K0(I)=0
21 K1(I)=0
   DO22I=1,62
   L1(I)=0
   L(I)=0
   DO22 J=1,320
   B(I,J)=0.
22 A(I,J)=0.
C
C***** INPUT OF PROBLEM *****
C
   IF(IGEN.NE.0) GO TO 20
   READ(1,1)IONE,ILAST
   READ(1,1)II,JJ
11 READ(1,2)I,J,X
   IF(I)16,16,18
18 A(I,J)=X
   GOTO11
16 J=JJ
17 READ(1,2)I,JK,X
   IF(I)20,20,19
19 A(I,JJ)=X
   L(I)=JK
```

```
      GOTO17
   20 IF(IGEN.EQ.0) GO TO 24
      READ(1,1) NSIZE
      CALL CREATE (NUMBER,NSIZE,IX,II,JJ,IONE,ILAST)
   24 NUM=ILAST-IONE+2
      III=II+1
      DO 31 I=1,III
      L1(I)=L(I)
      DO31J=1,JJ
   31 B(I,J)=A(I,J)
      WRITE(3,3011)NUMBER
      WRITE(3,2040)
      CALL XLP (II,JJ,III)
 1125 KOUNT=KOUNT+1
C
C***** ROUNDING OF INTEGER VARIABLES TO ZERO OR ONE *****
C
      DO701NN=2,II
      IF((L(NN).LT.IONE).OR.(L(NN).GT.ILAST))GOTO701
      IF(A(NN,JJ).GT..0001) GO TO 702
      A(NN,JJ)=0.
      GOTO701
  702 IF(A(NN,JJ).LT..9999) GO TO 701
      A(NN,JJ)=1.0
  701 CONTINUE
C
C***** OBJECTIVE FUNCTION CORRECTION *****
C
      FUNCT=A(III,JJ)-TFUNC
      IF(ITER)9998,713,731
C
C***** CREATION OF FIRST NODE *****
C
  713 DO714NN=1,16
  714 WNODE(NN)=0.0
      DO721NN=2,II
```

```
      IF((L(NN).LT.IONE).OR.(L(NN).GT.ILAST))GOTO721
      WNODE(L(NN)-IONE+2)=A(NN,JJ)
  721 CONTINUE
      WNODE(1)=FUNCT
      CALL CHECK (WNODE,K1,K0,&5004,&5002)
 5004 WNODE(1)=10.**45
 5002 WRITE(3,2030)KOUNT,ITER,IVAR,WNODE
      IF(KOUNT.EQ.1)GOTO1001
      ITER=1
      GOTO1001
C
C***** CREATION OF SECOND NODE *****
C
  731 DO741I=1,16
      ZNODE(I)=0.
  741 SNODE(I)=0.0
      DO742I=2,II
      IF((L(I).LT.IONE).OR.(L(I).GT.ILAST))GOTO742
      SNODE(L(I)-IONE+2)=A(I,JJ)
  742 CONTINUE
      SNODE(1)=FUNCT
      CALL CHECK (SNODE,K1,K0,&5003,&5001)
 5003 SNODE(1)=10.**45
 5001 WRITE(3,2030)KOUNT,ITER,IVAR,SNODE
      ITER=0
      NODE2=NODE2+1
      IF(NODE2.LE.150)GOTO8002
      WRITE(3,8010)
      PAUSE'RECORD CPU TIMER READING'
      GOTO9996
C
C***** NODE STORING SECTION *****
C
 8002 IF(SNODE(1)-WNODE(1))751,761,771
  751 IF(WNODE(1)-SMALL)791,791,753
  753 IF(SNODE(1)-SMALL)792,792,773
```

```
  761 IF(WNODE(1)-SMALL)791,791,773
  771 IF(WNODE(1)-SMALL)772,782,773
  772 IF(SNODE(1)-SMALL)781,781,782
  773 DO774I=1,NUM
  774 ZNODE(I)=WNODE(I)
      WRITE(3,2080)NODE2,ZNODE
      DO777I=1,NUM
  777 C(NODE2,I)=ZNODE(I)
      DO775I=1,NUM
  775 ZNODE(I)=SNODE(I)
      NODE2=NODE2+1
      IF(NODE2.LE.150)GOTO8001
      WRITE(3,8010)
      PAUSE'RECORD CPU TIMER READING'
      GOTO9996
 8001 DO776I=1,NUM
  776 C(NODE2,I)=ZNODE(I)
      WRITE(3,2080)NODE2,ZNODE
      DO4001 I=1,NUM
 4001 WNODE(I)=C(NODE1,I)
      NODE3=NODE1
      SMALL=10.**35
      C(NODE1,1)=10.**40
      NODE1=0
      DO4003I=1,NODE2
      IF(C(I,1).GT.SMALL)GOTO4003
      SMALL=C(I,1)
      NODE1=I
 4003 CONTINUE
      WRITE(3,2510)NODE3,NODE1,SMALL
      GOTO1001
  781 SMALL=SNODE(1)
      NODE1=NODE2
  782 DO783I=1,NUM
  783 ZNODE(I)=SNODE(I)
      GOTO799
```

```
  791 SMALL=WNODE(1)
      NODE1=NODE2
  792 DO754I=1,NUM
      ZNODE(I)=WNODE(I)
  754 WNODE(I)=SNODE(I)
  799 WRITE(3,2080)NODE2,ZNODE
      DO798I=1,NUM
  798 C(NODE2,I)=ZNODE(I)
C
C***** SECTION TO DETERMINE NEXT VARIABLE TO TEST *****
C
 1001 IF(ITER.EQ.1)GOTO1100
      IVAR=0
      N=0
      J=0
      DO1021I=1,15
      K1(I)=0
 1021 K0(I)=0
      FLAG=0.
      DO1002I=2,NUM
      IF(WNODE(I))1003,1003,1004
1003  J=J+1
      K0(J)=I-1
      GOTO1002
1004  IF(WNODE(I)-1.0)1006,1005,1005
1005  N=N+1
      K1(N)=I-1
      GOTO1002
1006  IF(WNODE(I).LE.FLAG)GOTO1002
      FLAG=WNODE(I)
      IVAR=I-1
1002  CONTINUE
      IF(J+N.EQ.NUM-1) KUTOFF=1
      IF(KUTOFF.EQ.1)GOTO1014
      K0(J+1)=IVAR
      GOTO1014
```

```
 1100 DO 1104 I=1,15
      IF(K0(I).NE.IVAR)GO TO 1104
      K0(I)=0
      GO TO 1105
 1104 CONTINUE
 1105 DO 1103 I=1,15
      IF(K1(I).NE.0)GOTO1103
      K1(I)=IVAR
      GOTO1014
 1103 CONTINUE
C
C***** SECTION TO LOAD L. P. AND MODIFY OBJECTIVE FUNCTION *****
C
 1014 I=0
      DO1011M=1,III
      L(M)=0
      DO1011J=1,JJ
 1011 A(M,J)=B(M,J)
 1007 I=I+1
      IF(K0(I).EQ.0)GOTO1008
      A(1,K0(I)+IONE-1)=PHONY
      GOTO1007
 1008 I=0
      TFUNC=0.
 1009 I=I+1
      IF(K1(I).EQ.0)GOTO1012
      A(1,K1(I)+IONE-1)=-PHONY
      TFUNC=TFUNC-PHONY-B(1,K1(I)+IONE-1)
      GOTO1009
 1012 DO4011I=1,III
 4011 L(I)=L1(I)
      CALL XLP (II,JJ,III)
      IF(KUTOFF.EQ.0)GOTO1125
C
C***** REPORT GENERATOR *****
C
```

```
      FUNCT=A(III,JJ)-TFUNC
      WRITE(3,3011)NUMBER
      WRITE(3,3020)FUNCT
      DO1301I=2,II
 1301 WRITE(3,3030)L(I),A(I,JJ)
      WRITE(3,3011)NUMBER
      WRITE(3,3020)FUNCT
      WRITE(3,3040)
      NN=0
      DO 1311 I=IONE,ILAST
      K=0
      NN=NN+1
      DO 1303 J=2,II
      IF(L(J).NE.I)GO TO 1303
      IF(A(J,JJ).GT..0001) GO TO 1304
      A(J,JJ)=0.
      GO TO 1306
 1304 IF(A(J,JJ).LT..9999) GO TO 1306
      A(J,JJ)=1.0
 1306 WRITE(3,3050)NN,A(J,JJ)
      GO TO 1311
 1303 CONTINUE
      K=1
      WRITE(3,3060)NN
 1311 CONTINUE
      IF(K.EQ.0) GO TO 3101
      WRITE(3,3100)
 3101 WRITE(3,3070)
      WRITE(3,3080)
      DO 1321 I=2,II
      IF(L(I).GE.IONE)GO TO 1321
      IF(A(I,JJ).GT..0001) GO TO 1322
      A(I,JJ)=0.
      GO TO 1331
 1322 IF(A(I,JJ).LT..9999) GO TO 1331
      A(I,JJ)=1.0
```

```
1331 IF(A(I,JJ).EQ.0.) GO TO 1321
     NUMBR=L(I)
     NUMBR1=1
     IF(L(I).LE.NUM-1) GO TO 1332
     DO 1333 J=1,14
     NUMBR=NUMBR-NUM+1
     NUMBR1=NUMBR1+1
     IF(NUMBR.LE.NUM-1)GO TO 1332
1333 CONTINUE
1332 WRITE(3,3090)NUMBR1,NUMBR,A(I,JJ)
1321 CONTINUE
     IF(IGEN.EQ.0) GO TO 9996
     CALL ENUM (NUMBER,NSIZE)
     GOTO9996
9998 STOP
     END
```

```
      SUBROUTINEXLP(II,JJ,III)
C
C
C          LINEAR PROGRAM SUBROUTINE USING THE SIMPLEX ALGORITHM
C
      COMMON A(62,320),L(62)
      COMMON E(15,15), LOGIC(15,15),FCOST(15),MLOC(30,10)
      DIMENSIONW(62)
      DO701I=1,62
  701 W(I)=0.0
  140 FORMAT(1H0,9HUNBOUNDED)
      KKK=0
      DO 102 I=1,JJ
      DO99J=2,II
      KP=L(J)
   99 A(III,I )=A(III,I)+A(J ,I)*A(1,KP)
      IF(I-JJ)100,102,100
  100 A(III,I)=A(III,I)-A(1,I)
  102 CONTINUE
      K=III
   44 J=0
      W(K)=0.0
      L(K)=0
   42 J=J+1
      IF(J-JJ)41,46,46
   41 IF(A(K,J))42,42,43
   43 IF(W(K)-A(K,J))47,42,42
   47 W(K)=A(K,J)
      L(K)=J
      GO TO 42
   46 KJ=L(K)
      DO 120 I=2,II
      IF(A(I,KJ))120,120,121
  120 CONTINUE
      WRITE(3,140)
      GOTO80
  121 I=1
```

```
      JK=0
   50 I=I+1
      IF(I-II)52,52,56
   52 IF(A(I,KJ))50,50,51
   51 X=A(I,JJ)/A(I,KJ)
      IF(JK)55,53,55
   55 IF(X-XMIN)53,50,50
   53 XMIN=X
      JK=I
      GO TO 50
   56 X=A(JK,KJ)
      L(JK)=KJ
      DO 57 I=1,III
   57 W(I)=A(I,KJ)
      IJ=JK-1
      DO 59 I=1,IJ
      DO 59 J=1,JJ
      IF(A(JK,J))58,59,58
   58 IF(W(I))580,59,580
  580 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
   59 CONTINUE
      IJ=JK+1
      DO 61 I=IJ,III
      DO 61 J=1,JJ
      IF(A(JK,J))60,61,60
   60 IF(W(I))600,61,600
  600 A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
   61 CONTINUE
      DO 205 J=1,JJ
  205 A(JK,J)=A(JK,J)/X
      KKK=KKK+1
      IJ=JJ-1
      DO 65 J=1,IJ
      IF(A(K,J)-.0001)65,66,66
   65 CONTINUE
      GOTO80
```

```
66 GO TO 44
80 RETURN
   END
```

80

```
      SUBROUTINE CHECK (ANODE,K2,K3,*,*)
C
CC        SUBROUTINE TO CHECK WHETHER OR NOT A VARIABLE DETERMINED TO BE
CC        EQUAL TO EITHER ZERO OR ONE HAS TAKEN ON THAT VALUE
C
      DIMENSION ANODE(16),K2(15),K3(15)
      J=0
      I=0
    1 I=I+1
      IF(K2(I).EQ.0)GOTO3
      IF(ANODE(K2(I)+1).NE.1.0)GOTO2
      GOTO1
    2 WRITE(3,11)
   11 FORMAT (' SUBR. CHECK FOUND AN ERROR IN A ONE VARIABLE')
      J=1
    3 I=0
    4 I=I+1
      IF(K3(I).EQ.0)GOTO6
      IF(ANODE(K3(I)+1).NE.0.)GOTO5
      GOTO4
    5 WRITE(3,12)
   12 FORMAT(' SUBR. CHECK FOUND AN ERROR IN A ZERO VARIABLE')
      RETURN1
    6 IF(J.EQ.0)GO TO 7
      RETURN 1
    7 RETURN2
      END
```

```
      SUBROUTINE CREATE  ( NPROB,NSIZE,IX,II,JJ,IONE,ILAST)
C
C         PROGRAM TO GENERATE PROBLEMS IN WHICH ALL POINTS ARE REACHABLE
C     FROM ALL OTHER POINTS.  THIS PROGRAM IS A MODIFICATION OF THE
C     PROGRAM USED TO GENERATE PROBLEMS FOR THE CONSTRAINED CASE.
C     THE DISTANCES ARE SAMPLED FROM A UNIFORM DISTRIBUTION ALLOWING
C     THE INTEGERS 11 THROUGH 30.  THE DEMANDS ARE FROM A UNIFORM
C     DISTRIBUTION ALLOWING THE INTEGERS  6 THROUGH 15.  THE FIXED
C     COSTS ARE SAMPLED FROM A UNIFORM DISTRIBUTION ALLOWING THE
C     INTEGERS BETWEEN 300 AND 400.
C
      COMMON A(62,320), L(62)
      COMMON C(15,15), LOGIC(15,15),F(15), MLOC(30,10)
      DIMENSION D(15)
   30 FORMAT('1',40X,'THE DISTANCE MATRIX FOR PROBLEM NUMBER',I5////)
   50 FORMAT('0',10X,15F7.2)
   60 FORMAT('1',40X,'THE DEMAND VECTOR FOR PROBLEM NUMBER',I5////)
   70 FORMAT(////40X,'THE FIXED COST VECTOR FOR THE PROBLEM'////)
   80 FORMAT('1',40X,'THE COST MATRIX FOR PROBLEM NUMBER',I5////)
   90 FORMAT('1',37X,'THE REACHABILITY MATRIX FOR PROBLEM NUMBER',I5////
     1)
  100 FORMAT('0',10X,1517)
      FIX=1000.
      DO 69 I=1,15
      D(I)=0
      F(I)=0.
      DO 69 J=1,15
      C(I,J)=0.
   69 LOGIC(I,J)=0
      DO 501 I=1,15
      F(I)=0.
      D(I)=0.
      DO 501 J=1,15
      C(I,J)=0.
  501 LOGIC(I,J)=0
      NSTOP=NSIZE-1
```

```
C
C****************** GENERATING THE DISTANCE MATRIX ******************
C
      DO 202 I=1,NSTOP
      II=I+1
      DO 202 J=II,NSIZE
  201 CALL RANDU(IX,URAND)
      RAND=20.*URAND
      RAND=11.+RAND
      IRAND=RAND
      IF(IRAND-31)202,201,201
  202 C(I,J)=IRAND
      DO 203 I=1,NSIZE
      II=I+1
      DO 203 J=II,NSIZE
  203 C(J,I)=C(I,J)
C
C******************* GENERATING THE DEMAND VECTOR ******************
C
      DO 204 I=1,NSIZE
  211 CALL RANDU(IX,URAND)
      URAND=10.*URAND
      URAND=6.+FLOAT(IFIX(URAND))
      IF(URAND-16)204,211,211
  204 D(I)=URAND
      WRITE(3,30)NPROB
      WRITE(3,50)((C(I,J),J=1,15),I=1,15)
      WRITE(3,60)NPROB
      WRITE(3,50)D
C
C*****************  GENERATING THE FIXED COST VECTOR ******************
C
      DO 205 I=1,NSIZE
      CALL RANDU(IX,URAND)
      URAND=100.*URAND
  205 F(I)=300.+FLOAT(IFIX(URAND))
```

```
      WRITE(3,70)
      WRITE(3,50)F
      DMAX=32.
C
C****************** GENERATING THE REACHABILITY MATRIX ******************
C
      DO 209 I=1,NSIZE
      DO 209 J=1,NSIZE
      IF(C(I,J)-DMAX)208,208,209
  208 LOGIC(I,J)=1
  209 CONTINUE
C
C****************** GENERATING THE COST MATRIX **********************
C
      DO 210 J=1,NSIZE
      DO 210 I=1,NSIZE
  210 C(I,J)=C(I,J)*D(J)
      WRITE(3,80)NPROB
      WRITE(3,50)((C(I,J),J=1,15),I=1,15)
      WRITE(3,90)NPROB
      WRITE(3,100)((LOGIC(I,J),J=1,15),I=1,15)
      IONE=NSIZE*NSIZE+1
      ILAST=IONE+NSIZE-1
      II=4*NSIZE+1
      JJ=NSIZE*NSIZE+6*NSIZE+1
      IROW=1
      ICOL=0
C
C****************** GENERATING THE OBJECTIVE FUNCTION ********************
C
      DO 2 I=1,NSIZE
      DO 2 J=1,NSIZE
      ICOL=ICOL+1
    2 A(IROW,ICOL)=C(I,J)
      DO 3 I=1,NSIZE
      ICOL=ICOL+1
```

```
   3 A(IROW,ICOL)=F(I)
     NNN=2*NSIZE
     DO 4 I=1,NNN
     ICOL=ICOL+1
   4 A(IROW,ICOL)=FIX
C
C****************** GENERATING THE SUM OF X(I) EQUAL TO ONE CONSTRAINTS
C
     XX=1.0
     AXX=-1.0
     DO 5 J=1,NSIZE
     IROW=IROW+1
     DO 7 I=1,NSIZE
     K=I-1
     IF(LOGIC(I,J)-1)7,6,6
   6 ICOL=K*NSIZE+J
     A(IROW,ICOL)=XX
   7 CONTINUE
     ICOL=NSIZE*NSIZE+J+NSIZE
     A(IROW,ICOL)=AXX
     ICOL=ICOL+NSIZE
     A(IROW,ICOL)=XX
   5 CONTINUE
C
C****************** GENERATING THE SUM OF X(I) - N*Y(I) LT 0 CONSTRAINTS
C
     DO 8 I=1,NSIZE
     IROW=IROW+1
     AA=0.
     K=I-1
     DO 9 J=1,NSIZE
     IF(LOGIC(I,J)-1)9,11,11
  11 AA=AA-1
   9 CONTINUE
     DO 12 J=1,NSIZE
     IF(LOGIC(I,J)-1)12,13,13
```

```
   13 ICOL=K*NSIZE+J
      A(IROW,ICOL)=XX
   12 CONTINUE
      ICOL=NSIZE*NSIZE+I
      A(IROW,ICOL)=AA
      ICOL=ICOL+3*NSIZE
      A(IROW,ICOL)=XX
    8 CONTINUE
C
C****************** GENERATING THE Y(I) LT 1 CONSTRAINTS ******************
C
      DO 15 I=IONE,ILAST
      IROW=IROW+1
      ICOL=I+4*NSIZE
      A(IROW,I)=XX
   15 A(IROW,ICOL)=XX
C
C****************** GENERATING THE X(I) LT Y(I) CONSTRAINTS ************
C
      J=0
      DO 14 I=1,NSIZE
      IROW=IROW+1
      J=J+1
      III=I-1
      ICOL=III*NSIZE+J
      A(IROW,ICOL)=XX
      ICOL=IONE+III
      A(IROW,ICOL)=AXX
      ICOL=ICOL+5*NSIZE
   14 A(IROW,ICOL)=XX
C
C****************** GENERATING THE RIGHT HAND SIDE ******************
C
      J=ILAST+NSIZE
      DO 27 I=2,II
      J=J+1
```

```
      IF(I-NSIZE-1)18,18,19
   19 IF(I-2*NSIZE-1)21,21,23
   23 IF(I-3*NSIZE-1)18,18,21
   18 XX=1.0
      GO TO 17
   21 XX=0.0
   17 L(I)=J
      A(I,JJ)=XX
   27 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ENUM (M,NN)
C
C        SUBROUTINE TO MANAGE THE INPUT-OUTPUT AND CALLING OF THE
C        ENUMERATION SUBROUTINE LOC1, LOC2, LOC3, AND LOC4
C
      COMMON A(62,320),L(62)
      COMMON C(15,15), LOGIC(15,15),FCOST(15), LOC(30,10)
  110 FORMAT(2I4)
  120 FORMAT(11F7.2)
  130 FORMAT('1',20X,'OPTIMUM LOCATIONS TO MINIMIZE TRANSPORT COSTS FOR
     1PROBLEM NUMBER',  I4)
  140 FORMAT(1H0,22X,17HOPTIMUM LOCATIONS,8X,10HFIXED COST,4X,14HTRANSPO
     1RT COST,4X,10HTOTAL COST)
  150 FORMAT(1H ,15X,10I3,F10.2,F17.2,F15.2)
  160 FORMAT(1H0)
  170 FORMAT(15I5)
      DO2I=1,30
      DO2J=1,10
    2 LOC(I,J)=0
      WRITE(3,130)M
      WRITE(3,140)
      K=1
   10 GOTO(11,12,13,14,999),K
   11 CALL LOC1 (NN,TOCOST,TRCST,FXCST,&21)
   12 CALL LOC2 (NN,TOCOST,TRCST,FXCST,&21)
   13 CALL LOC3 (NN,TOCOST,TRCST,FXCST,&21)
   14 CALL LOC4 (NN,TOCOST,TRCST,FXCST,&21)
   21 I=1
   22 IF(LOC(I,1).EQ.0)GOTO100
      WRITE(3,150)(LOC(I,J),J=1,10),FXCST,TRCST,TOCOST
      I=I+1
      GOTO22
  100 WRITE(3,160)
      K=K+1
      GOTO10
  999 RETURN
      END
```

```
      SUBROUTINE LOC1 (N,SMALL,TRANS,FXCOST,*)
C
C        SUBROUTINE TO DETERMINE THE BEST LOCATION FOR A SINGLE
C        DISTRIBUTION CENTER BASED ON MINIMUM TRANSPORTATION AND FIXED
C        COSTS BY COMPLETE ENUMERATION
C
      COMMON E(62,320),L(62)
      COMMON A(15,15), LOGIC(15,15),FCOST(15),MLOC(30,10)
      INTEGERZ
      SMALL=99999999.
      DO4I=1,N
      TRCOST=0.
      DO2J=1,N
    2 TRCOST=TRCOST+A(I,J)
      TOCOST=TRCOST+FCOST(I)
      IF(TOCOST-SMALL)5,3,4
    5 SMALL=TOCOST
      DO 6 Z=1,30
    6 MLOC(Z,1)=0
      LL=1
      TRANS=TRCOST
      FXCOST=FCOST(I)
      GOTO7
    3 LL=LL+1
    7 MLOC(LL,1)=I
    4 CONTINUE
      RETURN1
      END
```

```
      SUBROUTINE LOC2 (N,SMALL,TRANS,FXCOST,*)
C
C        SUBROUTINE TO DETERMINE THE BEST COMBINATION OF 2 POINTS TO
C        LOCATE DISTRIBUTION  CENTERS BASED ON MINIMUM TRANSPORTATION
C        COSTS AND FIXED COSTS USING COMPLETE ENUMERATION
C
      COMMON E(62,320),LN(62)
      COMMON A(15,15), LOGIC(15,15),FCOST(15),MLOC(30,10)
      INTEGERZ
      SMALL=99999999.
      DO1I=1,N
      DO1J=1,N
      TRCOST=0.
      DO2K=1,N
      IF(A(I,K)-A(J,K))3,3,4
    3 ADD=A(I,K)
      GOTO5
    4 ADD=A(J,K)
    5 TRCOST=TRCOST+ADD
    2 CONTINUE
      TOCOST=TRCOST+FCOST(I)+FCOST(J)
      IF(TOCOST-SMALL)7,6,1
    7 SMALL=TOCOST
      TRANS=TRCOST
      FXCOST=FCOST(I)+FCOST(J)
      LL=1
      DO 11 Z=1,30
      MLOC(Z,1)=0
   11 MLOC(Z,2)=0
      GOTO8
    6 IDENT=0
      DO 12 Z=1,LL
      IF((I.EQ.MLOC(Z,1).OR.I.EQ.MLOC(Z,2)).AND.(J.EQ.MLOC(Z,1).OR.J.EQ.
     1MLOC(Z,2))) IDENT=1
   12 CONTINUE
      IF(IDENT.EQ.1)GO TO 1
```

```
      LL=LL+1
    8 MLOC(LL,1)=I
      MLOC(LL,2)=J
    1 CONTINUE
      RETURN1
      END
```

```
      SUBROUTINE LOC3 (N,SMALL,TRANS,FXCOST,*)
C
C        SUBROUTINE TO DETERMINE THE BEST COMBINATION OF 3 POINTS TO
C        LOCATE DISTRIBUTION  CENTERS BASED ON MINIMUM TRANSPORTATION
C        COSTS AND FIXED COSTS USING COMPLETE ENUMERATION
C
      COMMON E(62,320),LN(62)
      COMMON A(15,15), LOGIC(15,15),FCOST(15),MLOC(30,10)
      INTEGERZ,Y
      SMALL=99999999.
      DO1I=1,N
      DO1J=1,N
      DO1K=1,N
      TRCOST=0.
      DO2L=1,N
      IF(A(I,L).LE.A(J,L).AND.A(I,L).LE.A(K,L))ADD=A(I,L)
      IF(A(J,L).LE.A(I,L).AND.A(J,L).LE.A(K,L))ADD=A(J,L)
      IF(A(K,L).LE.A(I,L).AND.A(K,L).LE.A(J,L))ADD=A(K,L)
    2 TRCOST=TRCOST+ADD
      TOCOST=TRCOST+FCOST(I)+FCOST(J)+FCOST(K)
      IF(TOCOST-SMALL)7,6,1
    7 SMALL=TOCOST
      TRANS=TRCOST
      FXCOST=FCOST(I)+FCOST(J)+FCOST(K)
      LL=1
      DO 11 Z=1,30
      DO11Y=1,3
   11 MLOC(Z,Y)=0
      GOTO8
    6 IDENT=0
      DO 12 Z=1,LL
      IF(I.NE.MLOC(Z,1).AND.I.NE.MLOC(Z,2).AND.I.NE.MLOC(Z,3))GO TO 12
      IF(J.NE.MLOC(Z,1).AND.J.NE.MLOC(Z,2).AND.J.NE.MLOC(Z,3))GO TO 12
      IF(K.NE.MLOC(Z,1).AND.K.NE.MLOC(Z,2).AND.K.NE.MLOC(Z,3))GO TO 12
      IDENT=1
   12 CONTINUE
```

```
      IF(IDENT.EQ.1)GO TO 1
      LL=LL+1
    8 MLOC(LL,1)=I
      MLOC(LL,2)=J
      MLOC(LL,3)=K
    1 CONTINUE
      RETURN1
      END
```

```
      SUBROUTINE LOC4 (N,SMALL,TRANS,FXCOST,*)
C
C        SUBROUTINE TO DETERMINE THE BEST COMBINATION OF 4 POINTS TO
C        LOCATE DISTRIBUTION  CENTERS BASED ON MINIMUM TRANSPORTATION
C        COSTS AND FIXED COSTS USING COMPLETE ENUMERATION
C
      COMMON E(62,320),LN(62)
      COMMON A(15,15), LOGIC(15,15),FCOST(15),MLOC(30,10)
      INTEGERZ,Y
      SMALL=99999999.
      DO1I=1,N
      DO1J=1,N
      DO1K=1,N
      DO1L=1,N
      TRCOST=0.
      DO2M=1,N
      IF(A(I,M).LE.A(J,M).AND.A(I,M).LE.A(K,M).AND.A(I,M).LE.A(L,M))ADD=
     1A(I,M)
      IF(A(J,M).LE.A(I,M).AND.A(J,M).LE.A(K,M).AND.A(J,M).LE.A(L,M))ADD=
     1A(J,M)
      IF(A(K,M).LE.A(I,M).AND.A(K,M).LE.A(J,M).AND.A(K,M).LE.A(L,M))ADD=
     1A(K,M)
      IF(A(L,M).LE.A(I,M).AND.A(L,M).LE.A(J,M).AND.A(L,M).LE.A(K,M))ADD=
     1A(L,M)
    2 TRCOST=TRCOST+ADD
      TOCOST=TRCOST+FCOST(I)+FCOST(J)+FCOST(K)+FCOST(L)
      IF(TOCOST-SMALL)7,6,1
    7 SMALL=TOCOST
      LL=1
      TRANS=TRCOST
      FXCOST=FCOST(I)+FCOST(J)+FCOST(K)+FCOST(L)
      DO 11 Z=1,30
      DO11Y=1,4
   11 MLOC(Z,Y)=0
      GOTO8
    6 IDENT=0
```

```
      DO 12 Z=1,LL
       IF(I.NE.MLOC(Z,1).AND.I.NE.MLOC(Z,2).AND.I.NE.MLOC(Z,3).AND.I.NE.M
     1LOC(Z,4)) GO TO 12
       IF(J.NE.MLOC(Z,1).AND.J.NE.MLOC(Z,2).AND.J.NE.MLOC(Z,3).AND.J.NE.M
     1LOC(Z,4)) GO TO 12
       IF(K.NE.MLOC(Z,1).AND.K.NE.MLOC(Z,2).AND.K.NE.MLOC(Z,3).AND.K.NE.M
     1LOC(Z,4)) GO TO 12
       IF(L.NE.MLOC(Z,1).AND.L.NE.MLOC(Z,2).AND.L.NE.MLOC(Z,3).AND.L.NE.M
     1LOC(Z,4)) GO TO 12
       IDENT=1
   12  CONTINUE
       IF(IDENT.EQ.1)GO TO 1
       LL=LL+1
    8  MLOC(LL,1)=I
       MLOC(LL,2)=J
       MLOC(LL,3)=K
       MLOC(LL,4)=L
    1  CONTINUE
       RETURN1
       END
```

```
      SUBROUTINE RANDU (IX,YFL)
C
CC        I B M RANDOM NUMBER GENERATOR
C
      IY = IX*65539
      IF(IY)5,6,6
    5 IY =IY+2147483647+1
    6 YFL = IY
      YFL = YFL*.4656613E-9
      IX=IY
      RETURN
      END
```

BIBLIOGRAPHY

1. Abramowitz, M. and I. A. Stegun, eds., Handbook of Mathematical Functions, U. S. Government Printing Office, Washington, D. C., 1964.

2. Agin, N., "Optimum Seeking with Branch and Bound," Management Science, Vol. 13 (1966), pp. 176-185.

3. Balas, E., "An additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research, Vol. 13 (1965), pp. 517-546.

4. Baumol, W. J. and P. Wolfe, "A Warehouse-Location Problem," Operations Research, Vol. 6 (1958), pp. 252-263.

5. Beale, E. M. L. and R. E. Small, "Mixed Integer Programming by a Branch-and-Bound Technique," Proceedings of IFIP Congress 65, pp. 450-451, 1965.

6 Burstall, R. M., R. A. Leaver and J. E. Sussams, "Evaluation of Transport Costs for Alternative Factory Sites--A Case Study," Operational Research Quarterly, Vol. 13 (1962), pp. 345-354.

7. Busacker, R. G. and T. L. Saaty, Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill, New York, N. Y., 1965.

8. Cooper, L., "Location-Allocation Problems," Operations Research, Vol. 11 (1963), pp. 331-343.

9. _____, "Heuristic Methods for Location-Allocation Problems," SIAM Review, Vol. 6 (1964), pp. 37-53.

10. Dakin, R. J., "A Tree-search Algorithm for Mixed Integer Programming Problems," The Computer Journal, Vol. 8 (1965), pp. 250-255.

11. Driebeck, N. J., "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12 (1966), pp. 576-587.

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

13. Eilon, S. and D. P. Deziel, "The Use of an Analogue Computer in Some Operational Research Problems," Operational Research Quarterly, Vol. 16 (1965), pp. 341-365.

BIBLIOGRAPHY (cont'd)

14. Eiseman, K., "The Optimum Location of a Center," SIAM Review, Vol. 4 (1962), pp. 394-395.

15. Elmaghraby, S. E., The Design of Production Systems, Reinhold, New York, N. Y., 1966.

16. Hadley, G., Linear Programming, Addison-Wesley, Reading, Mass., 1962.

17. Hakimi, S. L , "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems," Operations Research, Vol. 13 (1965), pp. 462-475.

18. Haley, K. B , "The Siting of Depots," International Journal of Production Research, Vol. 2 (1962), pp. 45-45.

19. Kuehn, A. A., and M. J Hamburger, "A Heuristic Program for Locating Warehouses," Management Science, Vol. 10 (1963), pp. 643-666.

20. Kuhn, H. W., and R. E. Kuenne, "An Efficient Algorithm for the Numerical Solution of the Generalized Weber Problem in Spatial Economics," Journal of Regional Science, Vol. 4 (1962), pp. 21-33.

21. Land, A. H , and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," Econometrica, Vol. 28 (1960).

22. Lawler, E. L., and D. E Wood, "Branch-and-Bound Methods: A Survey," Operations Research, Vol. 14 (1966), pp. 699-719.

23. Manne, A. S , "Plant Location Under Economies-of-Scale--Decentralization and Computation," Management Science, Vol. 11 (1964), pp. 213-235.

24. Miller, I., and J. E. Freund, Probability and Statistics for Engineers, Prentice-Hall, Englewood Cliffs, N. J., 1965.

25. Maranzana, F. E., "On the Location of Supply Points to Minimize Transport Costs," Operational Research Quarterly, Vol. 15 (1964), pp. 261-270.

26. Reiter, S., and G. R. Sherman, "Allocating Indivisible Resources Affording External Economies or Diseconomies," International Economic Review, Vol. 3 (1962), pp. 108-135.

BIBLIOGRAPHY (cont'd)

27. Simon, H., "Modeling Human Mental Processes," Proceedings of the 1961 Western Joint Computer Conference.

28. Smykay, E. W , and W. A. Fredericks, "An Index Based Method for Evaluating Warehouse Locations," Transportation Journal, Vol. 3 (1963), pp. 30-34.

29. Weber, A , "Uber den Standorf Der Industrien," Tubingen, 1909. Translated as "Alfred Weber's Theory of the Location of Industries" by C. J. Friedrich in 1929.

VITA

Personal History

Name: Robert Erwin Fleisher

Birth Place: Stettin, Germany

Birthdate: December 13, 1939

Parents: Fred and Margaret Fleisher

Wife: Anna Jane (Hunt) Fleisher

Children: Katherine Ann and Robert Erwin, Jr.

Educational Background

Junior College of Kansas City, Mo.
Associate of Science — 1958-1960

University of Kansas
Bachelor of Science in
Electrical Engineering — 1960-1962

Lehigh University
Candidate for Master of
Science in Industrial Engineering — 1965-1967

Professional Experience

Western Electric Co., Inc.
Lee's Summit, Mo.
Manufacturing Planning Engineer — 1962-1965

Western Electric Co., Inc.
Princeton, N. J.
Research Engineer — 1965-1967