MIXED INTEGER PROGRAMMING ALGORITHMS FOR SITE SELECTION AND OTHER FIXED CHARGE PROBLEMS HAVING CAPACITY CONSTRAINTS

BY

PAUL GRAY

TECHNICAL REPORT NO. 101

November 30, 1967

SUPPORTED BY THE ARMY, NAVY, AIR FORCE AND NASA UNDER
CONTRACT Nonr-225(53)(NR-042-002)
WITH THE OFFICE OF NAVAL RESEARCH

DEPARTMENT OF OPERATIONS RESEARCH

AND

DEPARTMENT OF STATISTICS

STANFORD UNIVERSITY

STANFORD, CALIFORNIA

MIXED INTEGER PROGRAMMING ALGORITHMS FOR SITE SELECTION AND OTHER FIXED CHARGE PROBLEMS HAVING CAPACITY CONSTRAINTS*

by

Paul Gray

TECHNICAL REPORT NO. 101

November 30, 1967

Supported by the Army, Navy, Air Force and NASA
under Contract Nonr-225(53)(NR-042-002)

Gerald J. Lieberman, Project Director

*Work on this project was supported in part by Naval Research Contract Nonr-225(89), National Science Foundation Grant GP-7074, and U.S. Army Contract DA-49-092-ARO-10

DEPARTMENT OF OPERATIONS RESEARCH
AND
DEPARTMENT OF STATISTICS
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

ABSTRACT

This dissertation presents algorithms for solving site-selection and similar fixed-charge problems with upper bound constraints. These problems typically arise in the following way: a number of sites are known to be available at which facilities can be established for performing a given service. The set of facilities must provide service at a specified level of effectiveness. Fixed costs are associated with each site if a facility is opened there, and variable costs are associated with operations. Upper-bound constraints on the allowable size of a facility at each site must be taken into account. The problem to be solved by the operations researcher is to select a subset of the available sites which meets the constraints and minimizes the total cost.

The basic approach to obtaining algorithms for the exact solution of the fixed charge problem is to formulate them as mixed integer programs and to solve these programs by decomposing them into a master problem (which is an integer program) and a series of subproblems which are linear programs. As in all decomposition schemes, heavy use is made of the fact that large portions of the constraint matrix contain only zero elements. In addition, a theorem of Dantzig and Hirsch which states that an optimal solution of the fixed-charge problem must occur at an extreme point, plays a central role.

To achieve computationally feasible algorithms, the number of vertices that have to be examined must be reduced to manageable proportions while still guaranteeing that the optimal solution will be found. This was accomplished by adapting the bound-and-scan integer programming algorithm by F. S. Hillier to the fixed charge problem. The decomposition idea was applied to the following four classes of problems and detailed algorithms are presented for each:

- A fixed charge problem with linear variable costs in which a fixed charge is associated with each continuous variable that appears at non-zero level.
- The same problem with separable concave or convex costs rather than linear costs.
- A warehouse location problem in which variable costs and constraints are of the transportation type. A fixed charge is associated with each warehouse opened, irrespective of the number of customers served.
- The fixed charge transportation problem where a fixed charge is associated with each route rather than each warehouse.

The first, second, and fourth algorithms were coded in ALGOL for the Burroughs B-5500 computer system and sample problem calculations run. Results of these calculations are presented.

A brief discussion of duality considerations for the fixed charge problem is included.

CONTENTS

CONTENTS

ILLUSTRATIONS

TABLES

I INTRODUCTION

A. Statement of the Problem

The purpose of this dissertation is to present algorithms for solving the site-selection and similar fixed-charge problems. These problems typically arise in the following way: a number, possibly a large number, of sites are known to be available at which facilities can be established for performing a given service. The set of facilities must provide service at a specified level of effectiveness. Fixed costs are associated with each site if a facility is opened there, and variable costs are associated with operations. Upper bound constraints on the allowable size of a facility at each site must be taken into account. The problem to be solved by the operations researcher is to select a subset of the available sites which meets the constraints and which minimizes the total cost.

This problem arises in many contexts and can be formulated as a mixed-integer programming problem, that is, a programming problem in which some of the variables take on values 1 and 0 (corresponding to a site being opened or not) and the rest are continuous variables. If there are no upper bounds on the size of the facilities, then the branch and bound algorithm presented by Efroymson and Ray [1] is applicable. However, imposing upper bounds on facility size invalidates the Efroymson-Ray method. It is this upper bounded problem to which this dissertation is primarily addressed.

More generally, the fixed-charge problem considered in this dissertation may be stated as:

$$\text{Minimize} \quad \sum_{i=1}^{m} c_i(x_i) + \sum_{j=1}^{n} f_j y_j \quad .$$

subject to

$$\mathbf{A}\mathbf{x} \geq \mathbf{b}$$

$$\mathbf{x} \leq \mathbf{m}$$

$$y_j = \begin{cases} 1 \text{ if } \sum_{i \in S_j} x_i > 0 \\ 0 \text{ otherwise} \end{cases} \qquad \text{for } j = 1, 2, \ldots, p$$

$$\mathbf{x} \geq 0$$

where $\mathbf{A}$ is an $m \times n$ matrix,[1] $\mathbf{x}$ and $\mathbf{m}$ are n-vectors, $\mathbf{b}$ is an m-vector, the $c(\cdot)$ are functions of a single variable, the f_j are positive constants, $p \leq m$, and the S_j are subsets of $\{1, 2, \ldots, m\}$.

This general structure encompasses each of the special cases for which algorithms are developed in this dissertation. The $\mathbf{A}$ matrix is considered either as a general matrix or as a matrix having special structure, such as all elements non-negative or a transportation-problem matrix. Both the case in which a fixed charge is associated with each component of $\mathbf{x}$ ($p = m$ and the sets S_j each contain one element) and the case where a fixed charge is associated with disjoint groups of components of $\mathbf{x}$ are investigated. Finally, both linear and separable nonlinear variable cost structures are considered.

B. Past Work

The fixed-charge problem has appeared in the Operations Research literature at least since 1954 when Hirsch and Dantzig [2] formulated the problem and showed that if the objective function is linear, an optimal solution would occur at an extreme point of the constraint set, $\mathbf{A}\mathbf{x} \geq \mathbf{b}$. This important result is used in each of the algorithms presented here.

In more recent years, a number of algorithms have been developed for solving certain fixed charge problems, especially the warehouse location problem (see Section I-C and Appendix A). Prominent among the approximate algorithms for this latter problem are those presented by Kuehn and Hamburger [3], Manne [4], and Feldman, Lehrer, and Ray [5]. In addition, Efroymson and Ray [1] have developed an exact algorithm for warehouse

[1] In this dissertation vectors and matrices are denoted by lower-case and upper-case boldface type, respectively.

location applicable to the case of no bounds on allowable warehouse size. An approximate algorithm has been presented by Balinski [6] for the fixed charge transportation problem. Another noteworthy contribution is an exact algorithm for the fixed charge problem with linear costs recently developed by Murty [7]. Murty's approach is to solve the problem as a linear program (*i.e.*, without fixed charges), to bound the total cost, and then to search systematically among the extreme points adjacent to the linear program optimum for the minimum total cost.

In principle it is also possible to obtain an optimal solution to a linear version of the fixed-charge problem by applying a general, mixed-integer, linear programming algorithm[1] such as Benders' partitioning algorithm [8] discussed in Appendix B. However, the state of the art in mixed-integer programming is such that existing algorithms are computationally feasible only for small fixed charge problems.

In summary, in addition to a number of approximate algorithms, three exact algorithms (Efroymson-Ray, Murty, Benders) were found in literature for solving fixed charge problems. As indicated in Part A, one of these (Efroymson Ray) is limited to cases without capacity constraints. The Murty algorithm has been proposed but, as far as is known, has not been tested. This algorithm appears most suited to cases in which fixed costs are small relative to variable costs.

Before discussing the decomposition approach used in this dissertation, a few practical examples of fixed-charge problems are presented.

C. Examples

The literature survey presented in Appendix A shows that the fixed-charge problem occurs in many operational situations. A few of these are listed here to indicate the range of fixed charge problems encountered in practice. Further discussions of some of these problems may be found in Appendix A.

1. The Warehouse (or Plant) Location Problem

Select a subset of warehouse or plant locations from among a list of available locations. Variable costs include the cost of the goods

[1] For a survey of these algorithms see Balinski [20] and Beale [21].

or equipment stored or the transportation cost of shipping goods to customers. A single fixed cost is associated with each warehouse opened. Constraints include total quantity of goods required from the warehouses and maximum size of the warehouses. The objective is to minimize total cost (fixed plus variable costs) while satisfying the given constraints. To fix ideas, the terminology in this dissertation will be in terms of warehouse locations, and reference will be made to sites and maximum warehouse size.

2. The Lock Box Problem

Select a set of Post Office lock box locations from among a list of potential locations to which customers send checks for bank clearance. Such a system reduces "float time" on accounts receivable at the cost of fixed charges for the lock boxes and both fixed and variable bank charges.

3. Fire (or Police) Station Problem

Given a set of potential fire-station locations, the area that can be covered from each location, and the distribution of value (or population) protected by each potential location, select a set of stations that provides an equal level of protection throughout a city. Available lot size limitations result in upper bounds on the size of each potential station. A fixed charge (for real estate, construction, etc.) is incurred for each station opened.

4. Detection-Station Placement Problem

Find the optimum location of n detection stations, say for detecting nuclear detonations. Each station has associated with it a distribution $p(r)$ which defines the probability of detection as a function of the distance r. A fixed charge is associated with each station opened.

5. The Fixed-Charge Transportation Problem

This problem differs from the standard Hitchcock transportation problem in that a fixed charge (*e.g.*, for route establishment or franchise) is associated with each route over which goods are shipped.

6. Billboard Location Problem

Given a fixed budget and a set of potential billboard locations, select a subset of billboards and rental times which maximizes the effectiveness (*e.g.*, in units of people-exposures) subject to the budget constraint.

7. Interceptor-Missile Site Problem

Select a set of interceptor missile sites and stockpiles from a list of available site locations that provide a balanced defense against a specified threat. Upper bounds on stockpiles that can be emplaced result from acreage limitations of available real estate. Fixed charges result from establishment costs and radar requirements; variable costs result from the size of the missile force emplaced.

The algorithms presented in this dissertation are not applicable to all of these examples. Specifically, problems in which site locations can be anywhere on a plane (such as the detection station problem discussed by Smallwood [9]) are not considered. The billboard problem is more closely related to the dual of the problems considered (see Section V), since its objective is to maximize effectiveness at fixed budget rather than minimizing cost at fixed effectiveness.

D. Decomposition Approach

The basic approach to obtaining algorithms for the exact solution of the fixed-charge problem is to formulate them as mixed-integer programs and to solve these mixed-integer programs by decomposing them. Decomposition ideas are not new; Dantzig [10, p. 449] states that decomposition was first proposed in 1958 in two papers, one by L. S. Ford and D. R. Fulkerson and the other by W. S. Jewell, as a method of solving multi-stage commodity-flow problems. These are large linear programming problems. In 1962, Benders [8] presented an algorithm for solving the mixed-integer programming problem by decomposition, in which he reduced the mixed problem to the solution of a series of integer programs.

The present work differs from Benders' in that the approach is to decompose the problem into a master problem and a series of linear programs rather than a series of integer programs. As in all decomposition schemes, heavy use is made of the fact that large portions of the

constraint matrix contain only zero elements. In the fixed-charge problem, the constraint matrix typically contains a large number of constraints which define the requirements on the continuous variables. In addition, there are integer constraints related to the fixed charges and, finally, there are constraints which relate the fixed and variable costs. The last reflect the conditions under which the fixed charges are incurred. They can be thought of as providing a weak coupling between the discrete and continuous parts of the problem.

It is the weak coupling that leads to decomposition. Specifying the values of the integer variables so that the integer constraints are satisfied produces a sub-problem which involves only continuous variables and hence can be solved by the now classical simplex method. Once the variable cost has been minimized by the simplex method and the fixed cost specified through the integer variables, the total cost is known. A systematic search through the allowable integer values, then, is all that is required to find an optimal solution.

The key question to be resolved for computability in any search scheme is how to reduce the size of the search to manageable proportions, while guaranteeing that the optimal solution will be found. Branch-and-bound methods provide the necessary technique for examining the integer parts of the problem. Of the various such algorithms currently available, the bound-and-scan algorithm by F. S. Hillier [11] was selected for use in many of the numerical calculations. This algorithm is particularly efficient if a good sub-optimal solution is available. Some attention is therefore paid to finding initial feasible solutions.

E. Organization of the Dissertation

Section II presents the formulation of the decomposition idea and shows how it can be applied to four classes of problems:

- A fixed-charge problem with linear variable costs in which a fixed charge is associated with each continuous variable that appears at non-zero level.
- A warehouse location problem in which variable costs and constraints are of the transportation type. A fixed charge is associated with each warehouse opened, irrespective of the number of customers served.

- The fixed-charge transportation problem in which a fixed charge is associated with each route rather than each warehouse.
- The first problem with separable concave or convex costs rather than linear costs.

In Section III, detailed algorithms for solving these problems are presented. The algorithms for the first and fourth problems incorporate the Hillier algorithm to generate combinations of open and closed sites. For the second problem, only a portion of the algorithm is used. The fixed charge transportation problem is treated by a special bound-and-search technique.

Results of sample calculations using these algorithms are presented in Section IV. The algorithms were coded in ALGOL for the Burroughs B-5500 computer system.

Section V presents a brief discussion of duality considerations.

Conclusions and directions for further work are presented in Section VI. Appendices present a literature survey (Appendix A), a discussion of the Benders and Murty algorithms (Appendices B and C) and a detailed proof (Appendix D) of an assertion in the text.

II FORMULATION

A. General Decomposition

Consider the mixed-integer linear programming problem in which all variables have finite upper bounds:[1]

$$\left.\begin{aligned} &\text{Minimize} \quad c_1 x + c_2 y \quad , \\ &\text{subject to} \\ &\qquad A_1 x + A_2 y \geq b \\ &\qquad\qquad x \leq m_1 \\ &\qquad\qquad y \leq m_2 \\ &\qquad x, y \geq 0 \\ &\qquad\qquad y \ \text{integer}. \end{aligned}\right\} \text{Problem I}$$

If at least one vector (x,y) exists which satisfies the constraints, an optimal solution to the problem must exist since all components of x and y are bounded above and below. Furthermore, since each component y_i of y is bounded between 0 and m_i, and y is defined only at integer lattice points, there are only a finite number of feasible values of y.

Let y_1 be a particular vector that satisfies the constraints, $0 \leq y \leq m_2$ and y integer. For this *fixed* value of y, the original problem reduces to:

$$\left.\begin{aligned} &\text{Minimize} \quad c_1 x \quad , \\ &\text{subject to} \quad A_1 x \leq b - A_2 y_1 \\ &\qquad\qquad x \leq m_1 \\ &\qquad\qquad x \geq 0 \quad . \end{aligned}\right\} \text{Problem II}$$

[1] c_1, m_1 and x are n_1 - vectors; c_2, m_2, y are n_2 - vectors; A_1 is an $(m \times n_1)$ matrix; and A_2 is an $(m \times n_2)$ matrix.

and Problem II is a linear program that can be solved by the simplex method. If Problem II does not have a feasible solution for this particular y_1 then Problem I also does not have a feasible solution for y_1. If, however, an optimal solution x_1 is found for Problem II, then the value of the objective function for Problem I becomes $c_1x_1 + c_2y_1$.

The definitions of Problems I and II immediately suggest an algorithm for solving the mixed-integer programming problem:

(1) Find a feasible solution to Problem I.[1]

(2) Enumerate the finite set of vectors y_1 that satisfy the constraints $0 \leq y \leq m_2$, y integer.

(3) For each y_1 found in Step 2, solve Problem II. If Problem II is infeasible for this value y_1, so is Problem I. If an optimal solution x_1 to Problem II is found, evaluate $c_1x_1 + c_2y_1$.

(4) The optimal solution is

$$\min_{\text{all } y_1} \{c_1x_1 + c_2y_1\} \quad .$$

Although this decomposition procedure is guaranteed to produce an optimal solution to Problem I, it does not, on the face of it, appear particularly attractive computationally for problems of even moderate size. For example, a problem containing ten integer variables each with range 0 - 1 would require attempting to solve 1024 linear programs; for twenty such variables the number of linear programs to be solved jumps to 10^6.

However, the decomposition procedure can be made workable for a wide range of mixed-integer programming problems, particularly fixed-charge problems, by making use of the detailed structure of the constraints to reduce drastically the number of vectors y_1 (and hence the number of linear programs) examined.

B. A Linear Fixed-Charge Problem

The linear fixed-charge problem is a special case of Problem I in which each component of y can take on only the values 0 or 1 and the

[1] This step could be dispensed with. If the exhaustive search of Step 2 does not turn up a feasible solution to Problem II, no feasible solution exists.

value 1 is assumed only if some combination of components of x is non-zero. If a fixed charge, f_i, is incurred for each activity engaged in, the problem can be stated in the form[1]:

$$\begin{aligned} &\text{Minimize} && \mathbf{cx} + \mathbf{fy} \quad , \\ &\text{subject to} && \mathbf{Ax} \geq \mathbf{b} \\ & && \mathbf{x} \leq \mathbf{m} \\ & && y_i = \begin{cases} 0 & \text{if } x_i = 0 \\ 1 & \text{if } x_i > 0 \end{cases} \\ & && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

so that the activities must satisfy certain constraints and cannot exceed upper bounds. A typical problem of this sort is the warehouse location problem with stockpile costs. The vector $\mathbf{x}$ corresponds to the supplies to be located at potential warehouses; $\mathbf{b}$ to the demands that must be met, and $\mathbf{m}$ to the maximal capacity of the warehouses. The objective is to select a set of warehouses that satisfy the demands, stay within the size constraints on individual sites, and minimize total cost. The total cost has two components: the variable-charge component, resulting from the allocation of stockpile to the warehouses; and the fixed-charge component resulting from the costs associated with opening up each warehouse selected.

For present purposes, this fixed-charge problem can be written in the form:[2]

$$\left.\begin{aligned} &\text{Minimize} && \mathbf{fy} + \mathbf{cx} \quad , \\ &\text{subject to} && \mathbf{Ax} \geq \mathbf{b} \\ & && \mathbf{My} - \mathbf{x} \geq \mathbf{0} \\ & && \mathbf{y} \leq \mathbf{1} \\ & && \mathbf{x}, \mathbf{y} \geq \mathbf{0} \\ & && \mathbf{y} \quad \text{integer} \quad . \end{aligned}\right\} \text{Problem III}$$

[1] This formulation assumes $n_1 = n_2$. In general, n_1 can be greater than n_2, with y_i taking on a value 1 if any one of several components of x is non-zero. The fixed-charge problem with transportation costs discussed in Part D is an example.

[2] In Problem III, M denotes a diagonal matrix with value m_j in the jth position on the diagonal corresponding to the upper bound on x_j.

This statement of the problem presupposes that all components of $\mathbf{f}$ and $\mathbf{c}$ are strictly positive (no free goods). In this case, the constraints $\mathbf{My} - \mathbf{x} \geq \mathbf{0}$ and $\mathbf{y} \leq \mathbf{1}$ assure that, in the optimal solution, $\mathbf{x}$ will be bounded by $\mathbf{m}$ and that y_i will be 0 if x_i is 0. This results from the fact that if x_i is 0 in the optimal solution, then $y_i = 0$ satisfies the constraints and minimizes the objective function. On the other hand, if x_i is positive, it cannot exceed m_i because the maximum feasible value of y_i is 1.

If the integrality requirement on $\mathbf{y}$ were dropped, Problem III would become a conventional linear program and could be solved by the simplex method. The linear program solution[1] will typically call for fractional values of some or all components of $\mathbf{y}$. Although this is not a feasible solution to Problem III it can be made into one by simply rounding each fractional y_i up to 1. Thus, a feasible solution to this mixed-integer problem can be found readily. Furthermore, the value of the objective function for this initial solution, (call it L_0), also provides a bound on the total fixed charge $\mathbf{fy}$.

This bound is found by solving a second linear program. Consider the case in which all fixed charges are incurred (*e.g.*, all warehouses are opened). For this case, Problem III reduces to the linear program:

$$\left.\begin{array}{lll} \text{Minimize} & \mathbf{cx} & , \\ \text{subject to} & \mathbf{Ax} \geq \mathbf{b} & \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{m} & . \end{array}\right\} \text{Problem IV}$$

If Problem III has a solution when considered as a linear program, then so will Problem IV, since it has the same constraint set and objective function in $\mathbf{x}$ and all the y_i components are fixed at their upper bound of 1.

The variable cost $\mathbf{cx}_0$ found in this way is the minimum variable cost, since it corresponds to the case in which all activities are available. If we now subtract $\mathbf{cx}_0$ from L_0 we obtain at once an upper bound, FMAX, on the fixed cost $\mathbf{fy}$ in the optimal solution of Problem III, where

$$\text{FMAX} = L_0 - \mathbf{cx}_0 \quad .$$

[2] If no feasible solution to the linear program exists, there is no feasible solution to Problem III.

Thus, we can adjoin an additional constraint to Problem III, namely, $\mathbf{fy} \leq$ FMAX. This constraint enables us to reject at once any value of the vector $\mathbf{y}$ for which $\mathbf{fy} >$ FMAX. Similarly, a lower bound on the value of $\mathbf{fy}$, in the optimal solution, call it FMIN, can be established. In terms of the warehouse location problem, if it is possible to supply all demands from any site then FMIN is equal to the fixed cost of the cheapest warehouse and FMIN provides no new information. However, if this situation does not obtain, the constraint $\mathbf{fy} \geq$ FMIN can be adjoined to Problem III. A loose FMIN bound is found by solving Problem III as a linear program with the variable-costs vector set to zero. Methods for finding FMIN are discussed in more detail under computational considerations (Section III-D).

Although no method has been found for improving FMIN as calculations proceed, it is possible to improve FMAX. Each time a value of the objective function, L_i, of Problem III is found which is better than any found thus far, FMAX can be reduced to $L_i - \mathbf{cx}_0$, and the constraint $\mathbf{fy} \leq$ FMAX tightened.

Summarizing in terms of the algorithm presented in Part A, for the fixed-charge problem the procedure is

(1) Find a feasible solution to Problem III. This can be done by solving Problem III as a linear program and rounding all fractional values of y_i up to 1.

(2) Find values for FMAX and FMIN and adjoin the constraints $\mathbf{fy} \geq$ FMIN, $\mathbf{fy} \leq$ FMAX.

(3) Enumerate the finite set of vectors $\mathbf{y}_1$ that satisfy the constraints $0 \leq \mathbf{y} \leq 1$, $\mathbf{y}$ integer, $\mathbf{fy} \leq$ FMAX, $\mathbf{fy} \geq$ FMIN.

(4) For each $\mathbf{y}_1$ found in Step 3, solve the problem:

$$\begin{aligned} &\text{Minimize} && \mathbf{cx} && , \\ &\text{subject to} && \mathbf{Ax} && \geq \mathbf{b} \\ & && \mathbf{x} && \leq \mathbf{my}_1 \\ & && \mathbf{x} && \geq \mathbf{0} \quad . \end{aligned}$$

If no feasible solution exists, find a new $\mathbf{y}_1$. If an optimal solution $\mathbf{x}_1$ is found, evaluate $\mathbf{c}_1\mathbf{x}_1 + \mathbf{f}_1\mathbf{y}_1$. Determine if FMAX can be reduced and, if it can, tighten the FMAX bound.

(5) The optimal solution is

$$\underset{\text{all } y_1}{\text{minimum}} \{cx_1 + fy_1\} \quad .$$

These steps are the essence of the computational method presented in Section III-D. Further refinements are introduced there to speed the computations, particularly in reducing the number of vectors y_1 for which linear programs must be solved.

C. Non-Linear Variable Costs

The discussions thus far have been restricted to linear variable costs. Concave and convex costs which can be approximated by piecewise linear costs can also be treated. The only restriction imposed is that the objective function be separable.

Consider a single variable as shown in Figure 1. Two cases are shown, corresponding to concave and convex cost function $c(x)$. The cost functions both have breakpoints at a and b, and the variable x is bounded at m. The fixed cost f is incurred only if $x > 0$. The two cases represented in Figure 1 will now be discussed in turn.

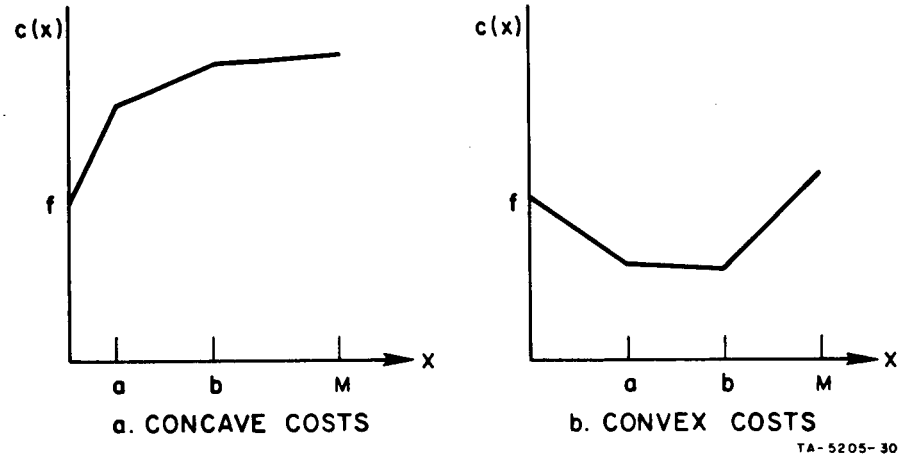


FIG. 1 PIECEWISE LINEAR APPROXIMATION TO COST FUNCTION OF ONE VARIABLE

1. Concave Costs

The concave cost case is of particular importance because it is often encountered in warehouse location problems. The concave cost function in Figure 1 can be replaced by three separate linear cost functions as shown in Figure 2. Note that the lower envelope of the three cost functions is the original concave cost. Replacing the variable x with three variables (x_1, x_2, and x_3) allows the concave cost function to be replaced by three linear segments, each having a different associated fixed cost (f_1, f_2, and f_3 in Figure 2). Thus, the problem has been expanded to six variables, three of them fixed-charge variables, to remove the non-linearity. Since the objective function is defined only on line segments, it is necessary to add upper and lower bound constraints on x_1, x_2, and x_3 to assure that these variables stay within their regions of definition. Furthermore, it is necessary to impose the constraint $\sum_{i=1}^{3} y_i \leq 1$ so that only one of the three segments is selected in any solution.

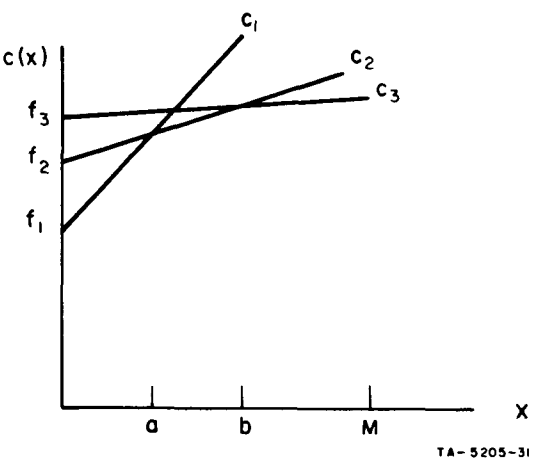


FIG. 2 REPLACING A CONCAVE COST FUNCTION BY SEVERAL LINEAR COST FUNCTIONS

The foregoing discussion for a concave cost function in one variable generalizes at once to separable concave objective functions. Suppose that the objective function is

$$\sum_{j=1}^{n} [\phi_j(u_j) + f_j v_j]$$

where $\phi_j(u_j)$ is a concave function in one variable and v_j is a 0 - 1 variable. The terms of the objective function can be represented as

$$\phi_j(u_j) = \sum_{i=1}^{n_j} c_i^j x_i^j \quad , \quad m_{i-1}^j \leq x_i^j \leq m_i^j$$

and

$$f_j v_j = \sum_{i=1}^{n_j} f_i^j y_i^j \quad , \quad y_i^j = 0, 1$$

where

n_j = Number of segments in the jth concave function,

m_i^j = Upper end of segment i in the jth concave function,

m_0^j = 0 .

Substituting into Problem III yields

$$\left.\begin{aligned}
&\text{Minimize} \quad \sum_{j=1}^{n} \sum_{i=1}^{n_j} [c_i^j x_i^j + f_i^j y_i^j] \quad , \\
&\text{subject to} \quad \mathbf{A'X'} \geq \mathbf{B} \\
&m_i^j y_i^j - x_i^j \geq 0 \qquad j = 1, \ldots n; \quad i = 1, \ldots n_j \\
&-m_{i-1}^j y_i^j + x_i^j \geq 0 \qquad j = 1, \ldots n; \quad i = 2, \ldots n_j \\
&\sum_{i=1}^{n_j} y_i^j \leq 1 \qquad j = 1, \ldots n \\
&x_i^j, \; y_i^j \geq 0 \\
&y_i^j \text{ integer} \quad ,
\end{aligned}\right\} \text{Problem V}$$

where $\mathbf{X'}$ is the column vector $(x_1^1, x_2^1, \ldots x_{n_j}^1, x_1^2, \ldots x_{n_j}^n)$, and $\mathbf{A'}$ is the matrix whose columns are $(A_1, \ldots A_1, A_2 \ldots A_2, \ldots A_n, \ldots A_n)$, each column A_j being repeated n_j times, and $\mathbf{b}$ is the same as in Problem III.

Examination of Problem V shows its structure to be that of Problem III and hence the algorithm for solving that problem can be applied here. Computationally, Problem V is a much larger problem than Problem III because each variable has been replaced by several

variables to remove the non-linearity in the objective function and because constraints have been added. Computational aspects are discussed further in Section III.

2. Convex Costs.

In the absence of fixed charges, problems involving separable convex costs and linear constraints can be treated by the techniques of separable convex programming [10, Chap. 24]. The bounded-variable method of separable convex programming calls for representing each function of one variable in the convex objective function by a piecewise linear function. For the ith variable, the line segments are expressed in terms of the variables Δ_{ij} and have slopes S_{ij}. Then x_i is replaced by $\sum_{j=1}^{k} \Delta_{ij}$, and $c(x_i)$ is replaced by $\sum_{j=1}^{k} S_{ij}\Delta_{ij}$ if there are k segments. If these changes are made in the fixed-charge problem, then the fixed-charge constraints can be represented as

$$M_i y_i - \sum_{j=1}^{k} \Delta_{ij} \leq 0 \qquad i = 1, 2, \ldots, n$$

$$y_i \leq 1$$

$$y_i = 0 \text{ or } 1 \quad .$$

Therefore, the use of a single fixed-charge variable for each convex cost function is sufficient to treat the problem.

D. A Fixed-Charge Problem with Transportation Costs

For the warehouse location problem in particular, the variable costs considered are typically transportation costs from the warehouses to the customers, with the fixed charges being incurred for each warehouse opened. Thus Problem III becomes

$$\left.\begin{aligned} &\text{Minimize} \quad \sum_i \sum_j c_{ij} x_{ij} + f_i y_i \quad , \\ &\text{subject to} \quad \sum_i x_{ij} \geq D_j \qquad j = 1, 2, \ldots n \\ &\qquad\qquad M_i y_i - \sum_j x_{ij} \geq 0 \qquad i = 1, 2, \ldots m \\ &\qquad\qquad x_i \leq 1 \\ &\qquad\qquad x_i, y_i \geq 0 \quad . \end{aligned}\right\} \text{Problem VI}$$

Here D_j represents the demands of the jth customer, x_{ij} the amount shipped from warehouse i to customer j, M_i the ith warehouse capacity, and y_i the fixed charge 0 - 1 variable associated with warehouse i.

As in Problem III, an initial feasible solution can be found by solving Problem VI as a linear program. A more efficient method is to recognize that if y is treated as a continuous variable, then at optimality the constraints $M_i y_i - \sum_j x_{ij} \geq 0$ will be satisfied exactly. Since all coefficients in the objective function are positive, minimization will be achieved by making the y_i as small as possible. Thus $y_i = (1/M_i)\sum_j x_{ij}$. Substituting for y_i in the other constraints and in the objective function leads to

$$\text{Minimize} \quad \sum_i \sum_j \{c_{ij} + (f_i/M_i)\} x_{ij} \quad ,$$

$$\text{subject to} \quad \sum_i x_{ij} \geq D_j \qquad j = 1, 2, \ldots n$$

$$-\sum_j x_{ij} \geq -M_i \qquad i = 1, 2, \ldots m$$

$$x_{ij} \geq 0 \quad ,$$

which is a pure transportation problem if $[c_{ij} + (f_i/M_i)] \geq 0$, for all i, j. The shipping costs are weighted by an additional factor (f_i/M_i) which can be thought of as the overhead cost borne per unit shipped if a site were stockpiled to capacity. As a weighting factor, (f_i/M_i) tends to make large warehouses with small fixed costs attractive and hence acts in the right direction. As previously, if y_i is positive at optimality, then setting it to 1 yields an initial feasible solution.

FMAX and FMIN values can also be found in this case by solving transportation problems; the former by setting all y_i to 1 and the latter by setting all c_{ij} to 0 in Problem VI. Finally, given a vector $\mathbf{y}_1$ with components y_{i1}, Problem VI reduces to

$$\text{Minimize} \quad \sum_i \sum_j c_{ij} x_{ij} \quad ,$$

$$\text{subject to} \quad \sum_i x_{ij} \geq D_j \qquad j = 1, 2, \ldots n$$

$$-\sum_j x_{ij} \geq -M_i y_{i1} \qquad i = 1, 2, \ldots m$$

$$x_{ij} \geq 0 \quad ,$$

which is also a transportation problem. Since y_{i1} has values of either 0 or 1, the second set of constraints expresses the fact that a warehouse is either open (with capacity M_i) or closed (zero capacity).

The arguments in the foregoing two paragraphs indicate that if the variable costs are of the transportation type, it is possible to replace the simplex method by efficient transportation-problem algorithms in solving the subproblem.

E. The Fixed-Charge Transportation Problem

This problem, described by Balinski [6], involves a fixed charge associated with each transportation link that is used rather than with each source opened. Formally, the problem can be stated as

$$\left.\begin{aligned}
&\text{Minimize} \quad \sum_i \sum_j c_{ij}x_{ij} + f_{ij}y_{ij} \quad , \\
&\text{subject to} \quad \sum_i x_{ij} \geq D_j \qquad j = 1,2, \ldots n \\
&\qquad\qquad -\sum_j x_{ij} \geq -S_i \qquad i = 1,2, \ldots m \\
&m_{ij}y_{ij} - x_{ij} \geq 0 \qquad \text{for all } i, j, \\
&\qquad\qquad x_{ij} \geq 0 \\
&\qquad\qquad y_{ij} = (0,1) \quad ,
\end{aligned}\right\} \text{Problem VII}$$

where D_j is the jth demand, S_i the ith supply, and $m_{ij} = \min(D_j, S_i)$. In his paper Balinski notes, as we do, that if the fixed-charge variables y_{ij} are treated as continuous, then in the optimal linear programming solution

$$m_{ij}y_{ij} = x_{ij}$$

and, hence, the linear problem can be reduced to a problem in x_{ij}. Balinski uses the formulation as an approximation to the integer solution, where any link that carries some goods is considered open and the full fixed charge is assessed. He presents computations to show that this approximation has the desirable property that, as the fixed costs of individual links increase, the number of links chosen is reduced.

Problem VII has the same structure as Problem I and hence an exact solution can be obtained in the same way. As in Problem VI, the subproblem is a transportation network and can be solved by application of a transportation algorithm. Initial feasible solutions, and values of FMAX and FMIN, can be obtained in the same way as described previously.

Transportation problems are characterized by a large number of variables. In the warehouse location problem, the calculations are simplified because only one integer variable is associated with each source. In the fixed-charge transportation problem, however, an integer variable is associated with each feasible link. For example, the test problems discussed in Section IV-C involve only five sources and seven destinations, with each source able to send goods to all destinations. Hence the problem involves 35 integer and 35 continuous variables (and 47 constraints). The FMAX and FMIN tests, although they reduce considerably the allowable number of link combinations, still leave an extremely large number of vectors $\mathbf{y}_1$ (and hence transportation problems) to be considered. However, there are a number of additional regularities in the structure of the problem that can be used to reduce this number. The simplest of these is the fact that at least one source must supply each destination and, hence, at least one link must be incident to each destination. If it should happen that no single source can supply the entire demand of a particular destination, then a minimum number of links incident to this destination can be determined and this fact can be used computationally. By symmetry between source and destination, similar constraints apply to the destination. Balinski proves that there exists an optimal integer solution which is also a basic solution. Since the maximum number of open links in any basic solution is $m + n - 1$ (m = number of sources, n = number of destinations), all combinations with more than this number of open links can be discarded.

In summary, although the fixed-charge transportation problem involves a large number of 0 - 1 variables, the number of $\mathbf{y}_1$ vectors that need to be considered can be reduced by taking into account the structure of the problem.

III COMPUTATIONAL METHODS

A. General Approach

The decomposition algorithm presented in Section II involves two major procedures:

(1) Finding an eligible vector $\mathbf{y}_1$, and

(2) Solving a linear program in $\mathbf{x}$ given $\mathbf{y}_1$.

In this section, methods for finding the vectors $\mathbf{y}_1$ in an efficient manner and for solving the resulting linear programs in $\mathbf{x}$ are presented.

An eligible vector $\mathbf{y}_1$ may be thought of as a *0-1 string*, that is, a sequence of 0's and 1's corresponding to the values[1] of the components y_i. If n is the number of integer variables, there are of course, 2^n such 0-1 strings and it rapidly becomes infeasible to examine each of them. At this point, we take recourse to the structure of the problem. As was pointed out in Section II, for fixed-charge problems with positive cost coefficients, simplex solutions of linear programs can be used to find both the continuous optimum and an initial feasible solution in integers. Furthermore, it is possible to find an upper and a lower bound (FMAX, FMIN) on the fixed cost $\mathbf{c}_1\mathbf{y}$. These bounds serve to reduce the number of 0-1 strings that need to be examined. For example, for the typical nine-integer variable problem discussed in Section IV-A, imposition of these bounds reduced the number of allowable 0-1 strings from 512 to 85. Thus, for small problems (say, ten to twelve integer variables) it may be possible to enumerate *all* 0-1 strings between the FMAX and FMIN limits, particularly if the FMAX bound is tightened as better solutions are found.

Mere enumeration, however, ignores a large part of the information available from the structure of the problem. Integer-programming algorithms which successively bound the values of integer variables given the values of the previous variables are particularly attractive for making use of the problem structure. These algorithms, generically

[1] Physically, 0 represents a closed warehouse and 1 an open warehouse.

known as branch-and-bound algorithms,[1] were developed for all-integer programming problems. One such algorithm, the bound-and-scan algorithm, developed by F. S. Hillier [11], was adapted for the present work.

The basic idea, then, is to use the Hillier algorithm to generate 0-1 strings which, it turns out, satisfy the constraints that are binding (constraints satisfied with equality) if the problem is solved as a linear program and which, furthermore, yield values of the fixed cost that fall within the FMAX and FMIN bounds. Since upper bounds are known for the components of the $\mathbf{x}$ vector, a simple algebraic test for feasibility of the sub-problem (Problem II of Section II) can be performed if all elements of the $\mathbf{A}$ matrix are non-negative. This test, which is necessary (but not sufficient in all cases) involves setting those components of $\mathbf{x}$ that correspond to $y_i = 0$ to zero and setting all the other components of $\mathbf{x}$ to their upper bounds. If any of the constraints are not satisfied, the 0-1 string does not lead to a feasible solution. If the algebraic test is passed, a linear program is solved which involves only the components of $\mathbf{x}$ that correspond to $y_i = 1$. This linear program leads either to an infeasible solution or to an optimal solution of the subproblem in $\mathbf{x}$. This optimal solution, if it exists, is then substituted back into the objective function of the original problem together with y_1 to determine whether an improved solution has been found and whether tighter bounds can be imposed on FMAX.

Because the Hillier bound-and-scan algorithm was an essential part of the computations performed, Part B sketches this algorithm and Part C describes the modifications made to adapt it for use in fixed-charge problems. A general description and flow diagram of the fixed charge algorithm is presented in Part D. The remaining parts (Parts E-G) present variations on the basic computational method to deal with special problems.

B. The Hillier Bound-and-Scan Algorithm

This part describes the Hillier bound-and-scan algorithm for the pure integer linear programming problem as presented by Hillier [11].

[1] See, e.g., [12] and [13] for a survey of these algorithms.

This algorithm was originally developed for problems with general integer variables in which the range of the variables could be quite large. Hillier assumed that the objective function is to be maximized.

Hillier began by finding an approximate feasible integer solution, $X = X_0$, and adding the constraint that the objective function $C X$ must equal or exceed the value $C X_0$ for this suboptimal solution. He then made the following key observation: the set of solutions (integer or non-integer) that satisfy both this objective function constraint and the constraints that are binding on the optimal non-integer (*i.e.*, linear program or LP[1]) solution must contain the optimal integer solution. Furthermore, the extreme points of this set form an *n*-simplex. This is illustrated in Figure 3.

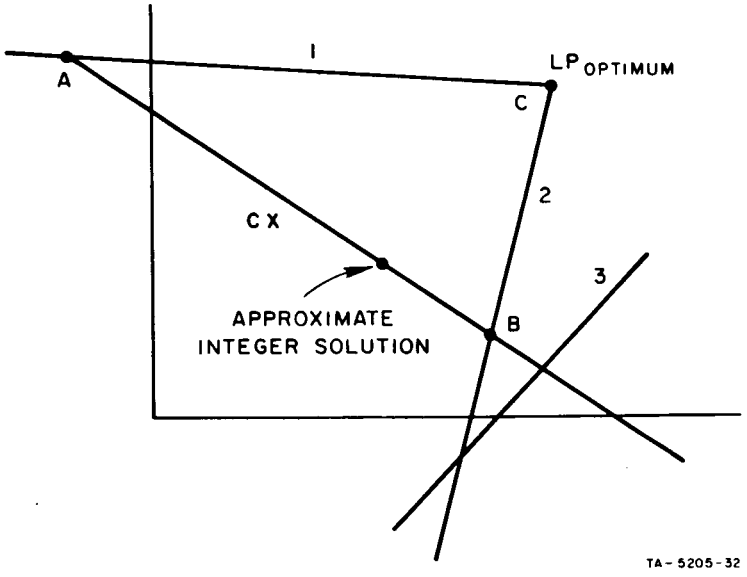


FIG. 3 EXAMPLE OF N-SIMPLEX FORMED BY BINDING CONSTRAINTS

Here line segments 1, 2, and 3 represent constraints, of which 1 and 2 are binding but 3 is not if the problem is solved as a linear program. The objective function $C X$ is drawn through an approximate integer solution. The triangle defined by *ABC* is a simplex in 2-space. Since any improvement in the integer solution permits moving the line $C X$ closer to the point *C* (the LP optimum), any such improved solution lies within the triangle.

[1] For convenience, the abbreviation LP will be used for *linear program*.

The extreme points of the simplex plus the positivity constraints provide two pieces of information:

(1) Any point inside the n-simplex can be written as a convex combination of the $n + 1$ extreme points; *i.e.*,

$$X = \sum_{i=1}^{n} \rho_i x_i \quad ,$$

where

$$\sum_{i=1}^{n} \rho_i = 1 \quad , \quad \rho_i \geq 0 \quad \text{for} \quad i = 1, 2, \ldots n \quad .$$

Upper bounds (*i.e.*, bounds less than one) can then be determined on some of the values of ρ_i by using other information about the original problem.

(2) The coordinates of the extreme points (plus the coordinate axes) determine the minimum and maximum values of each of the variables x_i and hence their range. An example is shown in Figure 4.

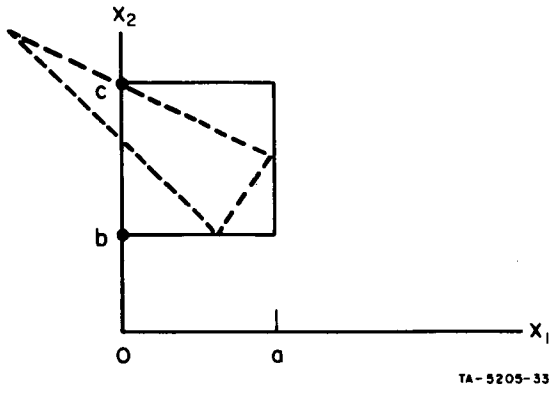
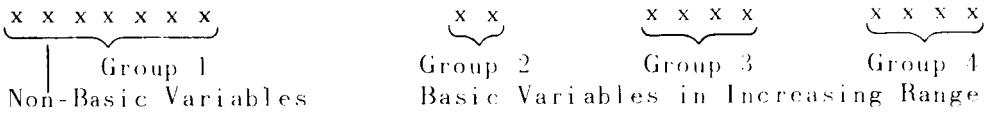


FIG. 4 BOUNDS ON THE ALLOWABLE RANGE OF VARIABLES

Here the solid rectangle drawn through the extreme points of the simplex of Figure 3 and the first quadrant x_2-axis indicates that $0 \leq x_1 \leq a$ and $b \leq x_2 \leq c$.

The next step in the algorithm is to separate the variables into four groups:



Group 1 consists of the variables that are non-basic in the *LP* solution; Group 2 contains the two basic variables with the smallest range of values within the *n*-simplex; and the remaining variables are split into Groups 3 and 4, with the variables ordered in increasing range. For convenience of presentation, the components of **x** will be assumed to be in the order in which they appear after sorting into groups.

It can be shown that the set of non-basic variables must satisfy the relation,

$$\sum_{j \in \{j \mid x_j \text{ non-basic}\}} x_j/(\max x_j) \leq 1 \quad , \tag{1}$$

where $\max x_j$ is the largest value of x_j within the *n*-simplex. Computational experience has shown that for non-basic variables, $\max x_j$ is generally = 1 or 2. These small ranges on x_j, combined with constraint (1), result in making the number of non-basic combinations that need to be examined approximately linear with the number of non-basic variables.

The allowable combinations of values of the non-basic variables are examined successively.[1] Each combination forms a hyperplane. The intersection of this hyperplane with the *n*-simplex determines tight conditional bounds on all the remaining variables. This is illustrated for a very simple case in two dimensions in Figure 5, where x_2 is basic and x_1 is non-basic. Fixing x_1 at 1 reduces the range of x_2 to $d \leq x_2 \leq e$. If the bounds on the basic variables are such that the range of one or more of them does not contain an integer value, this partial solution need not be examined further. That is, the partial solution is *ineligible*.

Given an eligible partial solution, the algorithm turns to the Group 2 variables. Hillier projects the intersection of the partial solution hyperplane and the *n*-simplex onto the plane determined by the

[1] Any such specification of values to only certain of the variables will be referred to as a "partial solution."

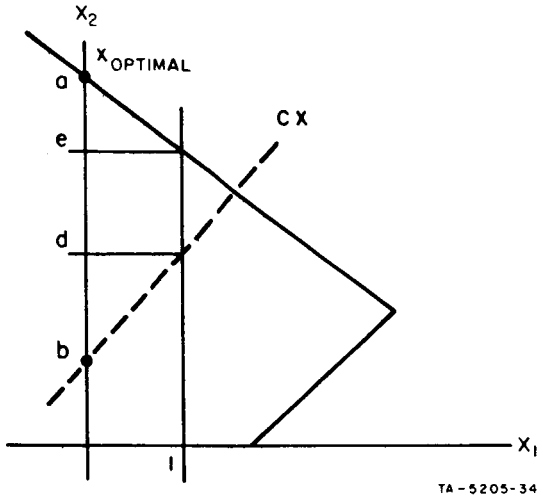


FIG. 5 EXAMPLE OF CONDITIONAL BOUNDS ON VARIABLES

two Group 2 variables. Any feasible integer values of these two variables must lie inside the convex hull of the projected extreme points (Fig. 6). A simple trigonometric procedure can be used to find all values of the Group 2 variables inside the convex hull. If there are none, a new partial solution is found for Group 1. Otherwise, the algorithm proceeds on to the Group 3 variables with the Group 1 and 2 variables fixed temporarily.

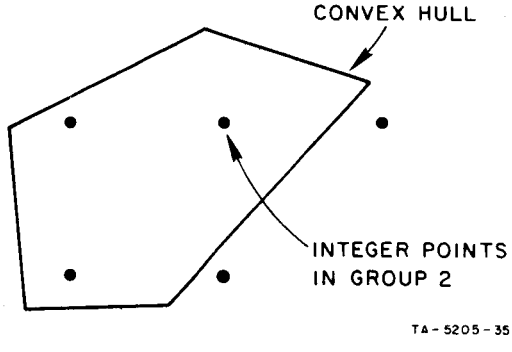


FIG. 6 CONVEX HULL OF PROJECTED EXTREME POINTS

To make the Group 3 and 4 procedures work, an initialization procedure is performed at the time the variables are grouped. Specifically, for each Group 3 variable use the simplex method to find

$$x_j^* = \max \{ x_j \mid x_1^0, x_2^0, \ldots x_{j-1}^0 \}$$

and

$$x_j^{**} = \min \{ x_j \mid x_1^0, x_2^0, \ldots x_{j-1}^0 \}$$

subject to the original problem constraints and the constraint $\mathbf{c}\,\mathbf{x} \geq \mathbf{c}\,\mathbf{x}_0$. That is, fix the first $(j-1)$ variables at their values in the initial feasible solution, $\mathbf{x}_0$, and then solve the *LP* problems of maximizing x_j and minimizing x_j. For convenience and for later use, let $\mathbf{b}_0$ be the vector obtained by subtracting these fixed values from $\mathbf{b}$ (the right-hand side of the constraint set).

The simplex method is used to obtain these maxima and minima and the optimal values of corresponding dual variables $(y_{j1}^*, \ldots y_{jm}^*)$ and $(y_{j1}^{**}, \ldots, y_{jm}^{**})$. Given the values of the Group 1 and 2 variables and the previous Group 3 variables, bounds on each Group 3 variable can be determined by using the fact that

$$\max \mathbf{c}\,\mathbf{x}$$

is a concave function of $\mathbf{b}$ (see, *e.g.*, Karlin [14, p. 239]). This fact implies the following bounds as the maximum and minimum values of Group 3 variables, given the previous variables:

$$\max x_j \leq x_j^* + \sum_{i=1}^{m} y_i^* \Delta b_i \quad ,$$

$$\min x_j \geq x_j^{**} - \sum_{i=1}^{m} y_i \Delta b_i \quad ,$$

where Δb_i is the difference between the right-hand side of the constraint set for the initial integer solution $(\mathbf{b}_0)$ and the right-hand side for the current partial solution being examined. The linear programming bounds established in this way for the Group 3 variables are compared with the extreme point bounds, and whichever is tighter is

used. Note that the bounds for each Group 3 variable depend on the values assigned to each of the previous Group 3 variables. The Group 3 variables are systematically examined within their bounds until no more eligible partial solutions can be found for them, at which point the process returns to the Group 2 procedure.

The initialization procedure for the Group 4 variables is similar to that for the Group 3 variables, except that rather than fixing *all* previous variables, only the variables up through the end of Group 3 are fixed. The extreme point bounds and the *LP* bounds are compared and whichever is tighter is used. Since the Group 4 variables have a wider initial range than either Group 2 or Group 3, a scanning procedure is used to speed the search for eligible trial completions (*i.e.*, completions that satisfy all constraints of the original problem plus the constraint that $CX \geq X_0$, the current integer solution). If an eligible completion is found which increases the value of the objective function, then the corresponding complete solution is recorded and the extreme point bounds are tightened. (Geometrically, CX is moved closer to the *LP* optimum in Fig. 3.) The last such solution to be recorded before exhausting eligible partial solutions must be an optimal integer solution.

A simple block diagram of the Hillier algorithm is shown in Figure 7.

C. Modifications of the Hillier Algorithm for the Fixed-Charge Problem

The fixed-charge problem is a mixed-integer programming problem whose integer variables are restricted to either 0 or 1. Thus, changes had to be made in the Hillier algorithm to adapt it to the fixed-charge problem.

The basic change was to use only the Groups 1, 2 and 3 portions of the algorithm to generate an eligible 0-1 string. To do this, variables were arranged so that:

(1) Group 1 consists of non-basic integer variables.

(2) Group 2 consists of 2 basic integer variables plus any variables with 0 range.

(3) Group 3 consists of the rest of the basic integer variables.

All non-integer variables (basic and non-basic) can be considered as Group 4 variables to be handled in a special way. In using the algorithm,

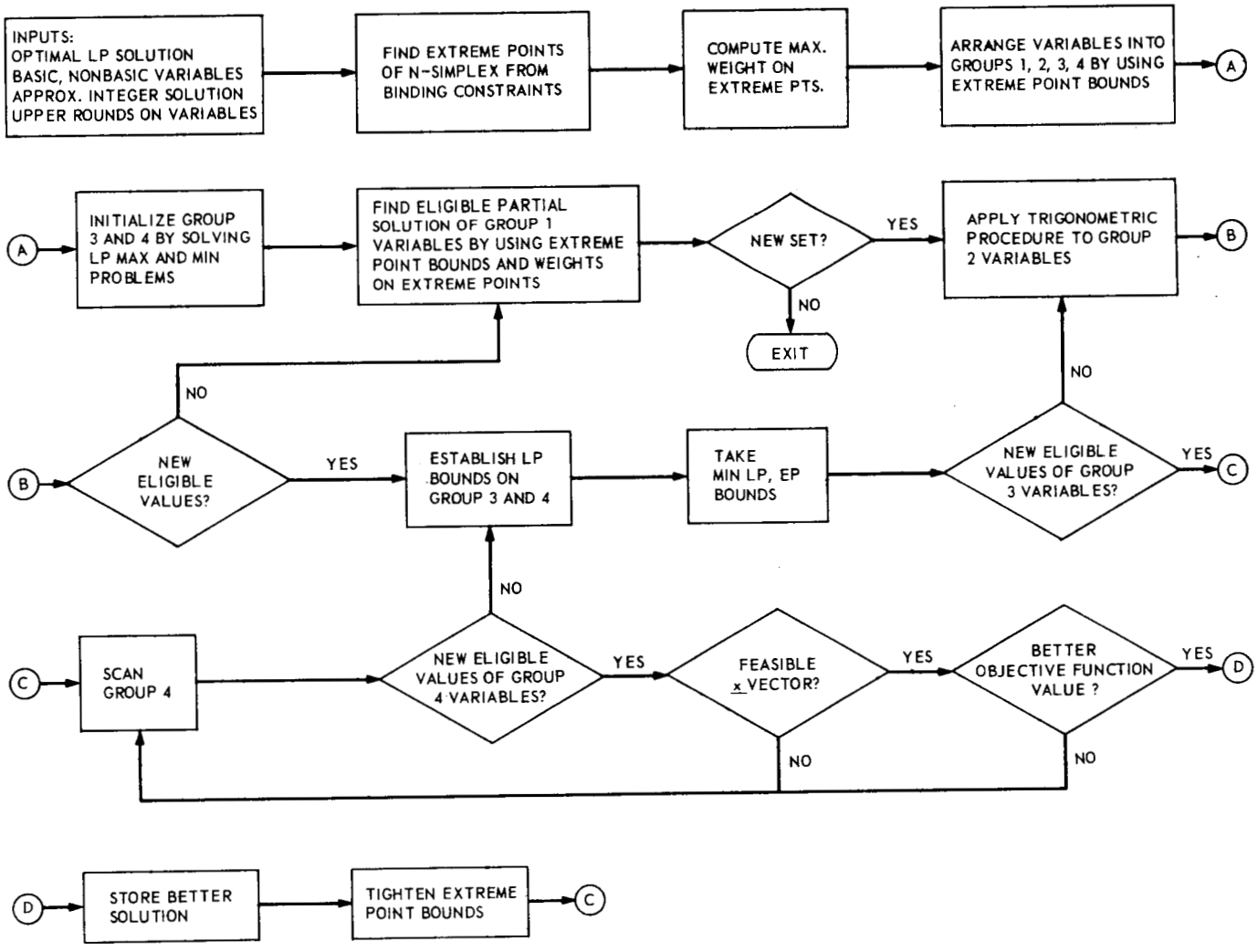


FIG. 7 FLOW DIAGRAM FOR HILLIER ALGORITHM

the allowable ranges of both the integer and the non-integer variables were considered in applying the extreme point and the linear programming bounds. In this way, the effect of constraints involving the non-integer variables as well as the integer variable constraints could be taken into account to reduce the number of 0-1 strings to be examined.

A second modification of the algorithm was to include upper bounds explicitly in determining the maximum allowable weights on the extreme points. A procedure was added which is analogous to that already in the algorithm for including the non-negativity requirement on each of the variables.

A number of minor computational changes were made, such as adding a more efficient linear program solver for initialization in Group 3, to increase computational efficiency of the algorithm.

The important point about the use of the Hillier algorithm is that it provides a convenient way of reducing the number of eligible 0-1 strings to examine. The algorithm was chosen for several reasons, including the desire to test this algorithm for a different class of problems and its availability in a compatible computer code. As will be discussed in more detail in Section IV, the algorithm as modified proved extremely suitable for the present purpose. Whether other implicit enumeration integer-programming algorithms might be superior for generating 0-1 strings remains an open question.

D. Description of the Fundamental Fixed-Charge Algorithm

This part presents a description of the fundamental fixed-charge algorithm. For expositional purposes, the desctiption is in terms of the steps required to solve Problem III, which is repeated here for convenience:

$$\left.\begin{array}{ll} \text{Minimize} & \mathbf{f}\,\mathbf{y} + \mathbf{c}\,\mathbf{x} \\ \text{subject to} & \mathbf{A}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{M}\,\mathbf{y} - \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \leq \mathbf{1} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \\ & \mathbf{y} \text{ integer} \end{array}\right\} \text{Problem III}$$

In this problem, a fixed-charge variable is associated with each continuous variable. Only minor modifications are needed if more general relations exist between the fixed charge and continuous variables. Examples of these modifications are presented in Parts E and F.

Figure 8 is a summary flow diagram of the algorithm. Each step of the algorithm is numbered on the flow diagram and this numbering system will be followed in the description. Both a mathematical statement of the step and, where applicable, a discussion of the computer technique, are presented.

Step 1: Solve the Problem as a Linear Program

If the integer constraint is removed, Problem III reduces to a linear program. The dimensionality of this program is shown as follows:

	m_1	m_1	
n	0	A x	$\geq$ B
m_1	M y	x	$\geq$ 0
m_1	y	0	$\leq$ 1 .

That is, the problem contains $(n + 2m_1)$ constraints and $2m_1$ variables. The dimensionality of the program can be reduced to $(n + m_1) \times m_1$ by recognizing the relation between x and y at optimality. Since all coefficients of the objective function are assumed to be positive (Section II-B), the objective function will be minimized if each component of y is as small as possible. Setting

$$y_i = x_i / m_i \qquad (2)$$

achieves this result. If x_i is 0 at optimality then y_i is 0. If $x_i > 0$, then x_i/m_i is the smallest value y_i can have and still satisfy the constraint $m_i y_i - x_i \geq 0$.

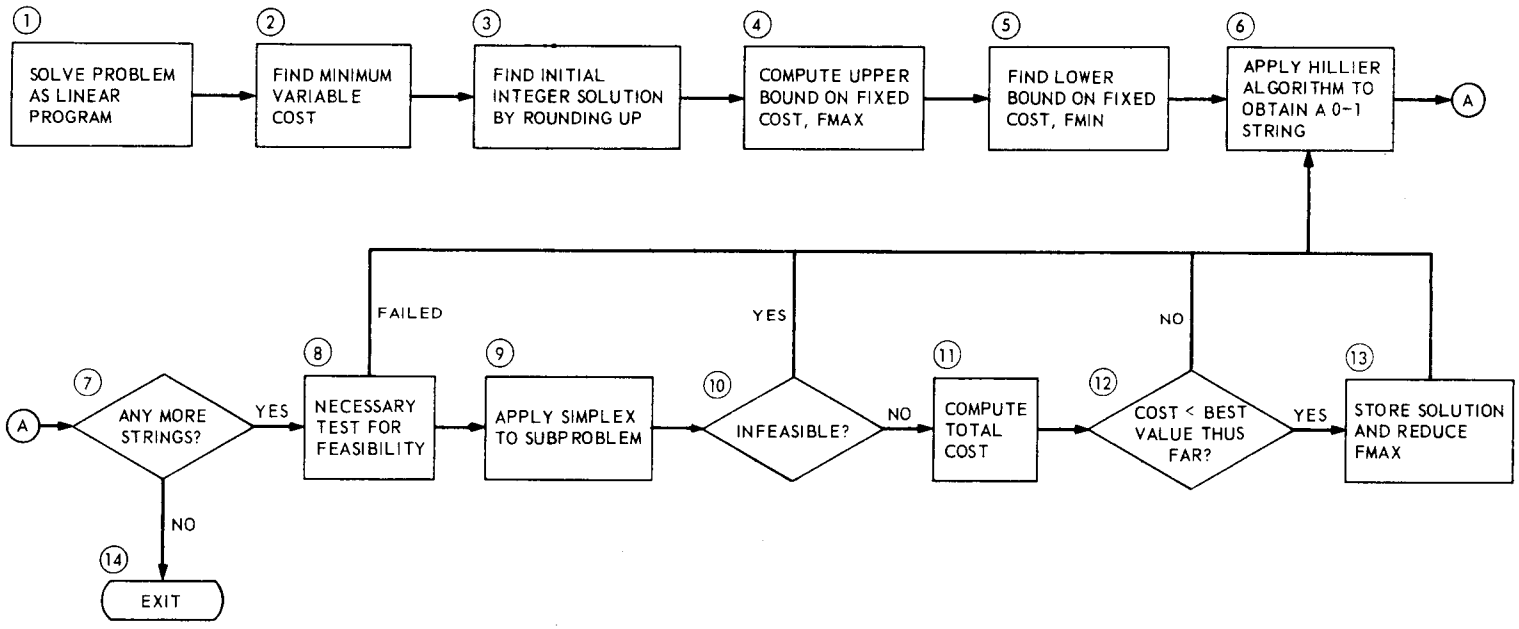


FIG. 8 FLOW DIAGRAM FOR FUNDAMENTAL ALGORITHM

Substituting Eq. (2) into Problem III yields the following:

$$\left.\begin{array}{ll} \text{Minimize} & \sum_i \left(c_i + \dfrac{f_i}{m_i}\right) x_i \quad , \\ \text{subject to} & \mathbf{A}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \leq \mathbf{m} \\ & \mathbf{x} \geq \mathbf{0} \quad . \end{array}\right\} \text{Problem VIII}$$

The upper bound constraint on $\mathbf{x}$ is still needed and is stated explicitly.

Problem VIII was solved using an *LP* package provided by Burroughs for the B-5500 computer. This program is limited to relatively small problems (limit: $m \times n \leq 5000$).[1] The program output provides not only the value of the objective function but also the value of the individual variables and a list of the basic and non-basic variables. These quantities are required in applying the Hillier algorithm. Once Problem VIII was solved, the list of basic variables was increased to include those y_i which are strictly positive according to Eq. (2). All other y_i were treated as non-basic.

Step 2: Find the Minimum Variable Cost

This step involves solving Problem IV, repeated here:

$$\left.\begin{array}{ll} \text{Minimize} & \mathbf{c}\,\mathbf{x} \quad , \\ \text{Subject to} & \mathbf{a}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{m} \geq \mathbf{x} \geq \mathbf{0} \, . \end{array}\right\} \text{Problem IV}$$

Problem IV has the same constraints as Problem VIII, but since it has a different objective function, the Burroughs *LP* code was applied again. The only datum needed was the optimal value of the objective function, which is equal to the minimum variable cost that can be achieved. The value of the objective function of Problem IV was called $\mathbf{c}\,\mathbf{x}_0$ in Section II.

[1] The choice of this relatively inefficient code (rather than say a specialized algorithm for upper bounded variables—see, *e.g.*, Dantzig [10, Chap. 18]) was dictated by availability of the program.

Step 3: Find Initial Integer Solution by Rounding Up

The linear program solution found in Step 1 can be converted into a feasible solution for Problem III by rounding all non-zero y_i up to 1. This rounded solution, called L_0, is guaranteed to be feasible because:

(1) The constraint, $\mathbf{A}\mathbf{x} \geq \mathbf{b}$, is satisfied by the *LP* solution found in Step 1.

(2) The constraint, $\mathbf{M}\mathbf{y} - \mathbf{x} \geq \mathbf{0}$, is satisfied since, for each index i, $x_i \leq m_i$ by Step 1 and y_i is set equal to 0 only if $x_i = 0$.

(3) The constraint $\mathbf{y} \leq 1$ is satisfied since the y_i are only assigned values of 0 or 1.

This feasible solution appears, *a priori*, to be a desirable approximation from which to begin search for the optimal integer solution. As shown in Step 1, the upper bounds and the fixed costs weight the variable costs in the objective function by

$$c_i + f_i/m_i \quad .$$

The ratio f_i/m_i can be thought of as the overhead burden borne by each unit if a particular warehouse were filled to capacity. Thus, for equal variable costs c_i, the linear program will favor those warehouses which have low f_i/m_i ratios. For equal fixed and variable costs, the linear program will favor the warehouse with the largest capacity.

An improvement can be achieved in the initial solution if an upper bound on the total stockpile is known; that is, $m_t \geq \sum_{i=1}^{n} x_i$. In this case, the capacities of individual warehouses which are above m_t can be reset to m_t for purposes of calculation. The improvement comes about because m_i appears in the denominator and, hence, large-capacity warehouses tend to be favored by the linear program.

If a better feasible integer solution (*i.e.*, one with smaller value of the objective function) is known, it can and should be used. The initial integer solution was called L_0 in Section II.

Step 4: Compute Upper Bound on Fixed Cost, FMAX

The minimum variable cost found in Step 2 and the initial integer solution found in Step 3 provide an upper bound on the maximum value of the fixed cost; that is,

$$\text{FMAX} = L_0 - \mathbf{c}\,\mathbf{x}_0 \quad .$$

Step 5: Find Lower Bound on Fixed Cost, FMIN

A lower bound on the fixed cost can be found by solving Problem III as a linear program without the integer constraint and with the variable costs set equal to 0. That is,

$$\left.\begin{array}{ll} \text{Minimize} & \mathbf{f}\,\mathbf{y} \quad , \\ \text{subject to} & \mathbf{A}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{M}\mathbf{y} - \mathbf{x} \geq \mathbf{0} \\ & \mathbf{y} \leq \mathbf{1} \quad , \end{array}\right\} \text{Problem IX}$$

where 1 is an n-vector all of whose elements are unity. If every warehouse can supply every customer and if each has sufficient capacity to supply the entire demand, then Problem IX can be solved by inspection, the value of FMIN being equal to the smallest fixed charge. Otherwise, the optimal value of the objective function of Problem IX will lead to a conservative lower bound on the fixed cost, provided that the individual m_i are adjusted so that none is larger than the total demand (see Step 3). The bound is conservative because it is based on the smallest fixed charge that could be achieved if the y_i were continuous. (Unfortunately it is not valid to round the strictly positive y_i up to 1 in order to set FMIN in this way).

The solution to Problem IX was used to obtain FMIN in the test problems in Section IV. If a better lower bound on the fixed cost is known, it can, of course, be used.

Step 6: Apply Hillier Algorithm to Obtain a 0-1 String

As discussed in Parts B and C of this Section, the Hillier integer programming algorithm was used to obtain 0-1 strings that define the set of open and closed warehouses. A computer code for this algorithm is

available for the Burroughs B-5500, the computer used for the test problems in Section IV. Call the vector defined by the 0-1 string, $\bar{y}$.

Step 7: Any More Strings?

The Hillier algorithm contains a test for determining if there are any additional 0-1 strings eligible for consideration. If there are no more, the program exits (Step 14).

Step 8: Necessary Test for Feasibility

Two tests are used to check whether or not necessary conditions for feasibility are satisfied by the 0-1 strings generated by the Hillier algorithm. The first test checks if

$$\text{FMIN} \leq \sum_i f_i y_i \leq \text{FMAX} \quad .$$

If this test is failed, the program returns to Step 6 and searches for the next 0-1 string of y_i values. If it is passed, a necessary test for feasibility of the sub-problem (Problem IV) is performed.

Suppose that all elements a_{ij} of the matrix $\mathbf{A}$ in the constraint set $\mathbf{A}\,\mathbf{x} \geq \mathbf{b}$ are positive. Therefore, for feasibility,

$$\mathbf{A}(\mathbf{M}\,\bar{\mathbf{y}}) \geq \mathbf{b}$$

must be satisfied. That is, set

$$x_i = \begin{cases} 0 & , \quad \text{if} \quad y_i = 0 \\ m_i & , \quad \text{if} \quad y_i = 1 \end{cases} \quad , \quad \text{for} \quad i = 1, 2, \ldots m_i \quad ;$$

then check to see whether

$$\sum_i a_{ij} m_i \bar{y}_i \geq b_j \quad , \qquad j = 1, 2, \ldots, n \quad . \tag{3}$$

If any constraint in the set defined by Eq. (3) is not satisfied, the 0-1 string leads to an infeasible solution. The reason is that, since all the x_i take on their maximum allowable value given the 0-1 string,

and since all the a_{ij} are positive, there is no feasible way of increasing any x_i so that Eq. (3) is satisfied.

The test defined by Eq. (3) can be extended to the case in which particular x_i have only negative coefficients a_{ij} by setting $x_i = 0$ irrespective of the value of y_i for these variables.

The test as outlined is not applicable if any of the x_i have both positive and negative coefficients in the matrix **A**. In this case, the test using Eq. (3) is skipped and infeasibility, if it occurs, will be found during the linear program (Step 9). The algebraic test defined by Eq. (3) has the advantage of requiring no matrix manipulation, an advantage that will be lost if Step 9 must be used to find infeasibility.

Step 9: Apply Simplex Method to Subproblem

Once the necessary tests for feasibility have been passed, the next step is to solve the following linear program (Problem IV):

$$\left.\begin{array}{ll} \text{Minimize} & \mathbf{c}\,\mathbf{x} \\ \text{subject to} & \mathbf{A}\,\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \leq \mathbf{M}\,\mathbf{y} \\ & \mathbf{x} \geq \mathbf{0} \end{array}\right\} \text{Problem IV}$$

This upper-bounded variable problem has a special structure that can be exploited. Since there are more equations $(n + m_1)$ than unknowns (m_1), it is natural to consider solving the dual of this problem:

$$\left.\begin{array}{ll} \text{Maximize} & \mathbf{b}\,\mathbf{u} + \mathbf{M}\bar{\mathbf{y}}\,\mathbf{v} \\ \text{Subject to} & \mathbf{A}^{\mathrm{T}}\mathbf{u} + \mathbf{I}\,\mathbf{v} \leq \mathbf{c} \\ & \mathbf{u}, \mathbf{v} \geq \mathbf{0} \end{array}\right\} \text{Problem X}$$

Since the upper bound constraints, $\mathbf{x} \leq \mathbf{M}\,\mathbf{y}$, all have positive coefficients, the dual problem contains within it an initial feasible solution: $\mathbf{u} = 0$, $\mathbf{v} = \mathbf{c}$. Such an initial feasible solution is invaluable since it eliminates the need for a Phase I calculation and allows proceeding directly to Phase II. In terms of the primal, it is equivalent

to starting with all variables at their upper bounds and then successively reducing X variables until C X is minimized. The dual formulation also has the advantage of reducing the size of the basis matrix from $(n + m_1) \times (n + m_1)$ to $(m_1) \times (m_1)$.

In performing the computations, it was found advantageous to write a special-purpose code that solves the dual problem starting with the V = C solution. The code is organized so that it first eliminates all x_i for which $y_i = 0$. Thus, the number of variables is equal to the number of y_i that equal 1. This further reduces the number of equations and the size of the basis in the dual problem. For example, in a 30-variable problem with 28 constraints, the original problem (Problem III) has 60 variables (30 x_i, 30 y_i) and 88 constraints. A typical dual sub-problem with five warehouses open has 33 variables but only five constraints; the basis matrix is only 5×5. With an initial feasible solution available, such problems are solved in a few iterations and in at most a second on the B-5500 computer.

Step 10: Infeasible?

If the linear program leads to the conclusion that the problem is infeasible,[1] further consideration of the 0-1 string is dropped and the algorithm returns to Step 6 to seek a new set of open and closed sites.

Step 11: Compute Total Cost

The value of the objective function, $\sum_i (c_i x_i + f_i y_i)$, is determined from the results of Step 6 and Step 9 which provide the fixed cost and the minimal variable cost, respectively. The one exception is that if, even though a warehouse (say k) is specified as open ($y_k = 1$), no stockpile is assigned there ($x_k = 0$ at optimality of the sub-problem). In this case, the total cost can be reduced by f_k since the fixed change is not incurred. The computer code was written to take this possibility into account.

Step 12: Cost < Best Value Thus Far?

The total cost just computed in Step 11 is compared with the lowest value of the objective function found previously. If the total cost is not improved, the program returns to Step 6 to find a new 0-1 string.

[1] Actually, since the dual of the sub-problem is solved, the computer determines unboundedness.

Step 13: Store Solution and Reduce FMAX

On the other hand, if the current solution is better than any found thus far, the new solution is stored and the value of the objective function recorded as L_t. Furthermore, the value of FMAX is reduced to $L_t - \mathbf{c}\,\mathbf{x}_0$, thereby reducing the number of 0-1 strings to be considered. The program returns to Step 6 with this tighter bound established.

Step 14: Exit

If no more 0-1 strings can be found, there are no more eligible solutions to be considered. Since the Hillier algorithm guarantees that no 0-1 string will be examined more than once, and since there are a finite number of these strings, the process must terminate in a finite number of steps. Furthermore, the Hillier algorithm guarantees that the 0-1 strings will be examined in such a way that the string corresponding to the optimal solution will not be missed.

The lowest value of the objective function found during the computations and the corresponding values of the x_t and y_t variables are the optimal solution to the problem.

E. Non-Linear Variable Costs

It was pointed out in Section II-C that the fundamental algorithm could be extended directly to fixed-charge problems with separable concave or convex variable costs. In the case of convex costs, the bounded variable method for separable convex programming [10, p. 484] is directly applicable and the fundamental fixed-charge algorithm can be applied directly once the problem has been set up as indicated in Section II-C. If the objective function contains separable concave variable costs, the fundamental algorithm can be modified to handle them. In the remainder of this part we present the details of the computational algorithm for concave costs.

Formally, the concave case was stated as Problem V as follows:

Problem V

$$\text{Minimize} \quad \sum_{j=1}^{n} \sum_{i=1}^{n_j} [c_i^j x_i^j + f_i^j y_i^j] \quad ,$$

subject to

$$\mathbf{A}'\mathbf{x}' \geq \mathbf{b}$$

$$m_i^j y_i^j - x_i^j \geq 0 \qquad j = 1, \ldots n \; ; \; i = 1, \ldots n_j$$

$$- m_{i-1}^j y_i^j + x_i^j \geq 0 \qquad j = 1, \ldots n \; ; \; i = 2, \ldots n_j$$

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

$$x_i^j, y_i^j \geq 0$$

$$y_i^j \quad \text{integer} \quad ,$$

where

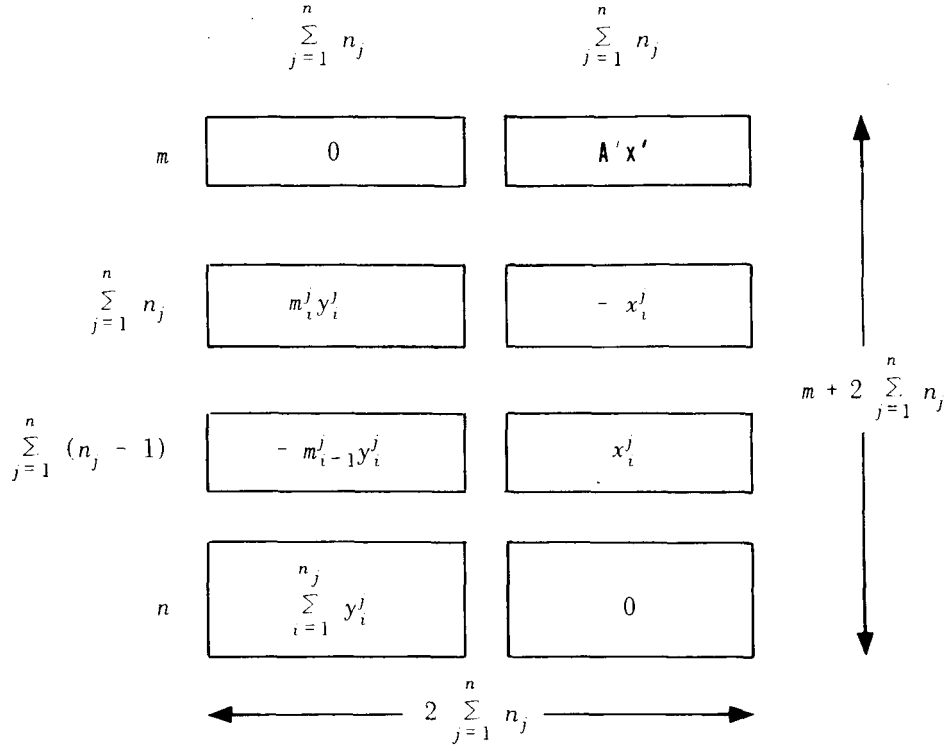
n_j = Number of segments in the jth concave function,

m_i^j = Upper bound of segment i in the jth concave function,

m_0^j = 0 .

Here, $\mathbf{x}'$ is the vector $(x_1^1, x_2^1, \ldots x_{n_j}^1, x_1^2, \ldots, x_{n_j}^n)$ and $\mathbf{A}'$ is the matrix whose columns are $(A_1, \ldots A_1, A_2, \ldots A_2, \ldots A_n, \ldots A_n)$, A_j being repeated n_j times, and $\mathbf{b}$ is the right-hand side of the original constraints.

Problem V is much larger than Problem III if each of the variables has several segments. This can be seen by examining the dimensionality of the constraint matrix:



For example a problem in nine variables with twelve constraints in $A x \geq b$ requires a constraint matrix of dimension (21×18) if the variable costs are linear. However, if each concave cost function must be represented by, say, three linear segments, the matrix grows to (66×54), which is an increase of a factor of 9.4 in the number of elements.

The growth in the number of rows and the number of elements in the non-linear case is offset considerably by the regularity in the structure of the problem, particularly the requirement that only one segment of each concave cost function can appear in the solution at a time. This is assured by the constraints

$$\sum_{i=1}^{n_j} y_i^j \leq 1 \quad \text{and} \quad y_i^j \text{ integer}$$

Computationally, Problem V requires several modifications in the algorithm presented for Problem III in the previous section. The flow diagram of Fig. 8 remains unchanged; however, several of the steps require additional care. The changes will now be described; steps for which no changes are listed remain as in the previous part.

Step 1: Solve as Linear Program

It is no longer possible to eliminate the y_i variables from the linear program by using the fact that, at optimality, $y_i^j = x_i^j / m_i^j$. Although the relation is true, the y_i must be included explicitly to guard against having two segments of the same cost function appear in the optimal solution. Problem V as formulated does not guarantee that this problem will be avoided. In the sample problem run on the computer (Section IV-B), a visual check of the linear program output showed that it was not encountered. If a large number of problems are being solved on a routine rather than an experimental basis, it would be necessary to modify the computer code so that only one segment at a time could appear in the optimal solution.

A saving can be made, however, in that the constraints,

$$m_{i-1}^j y_i^j + x_i^j \geq 0 \quad , \quad j = 1, \ldots n \quad ; \quad i = 2, \ldots n_j \quad ,$$

need not be stated explicitly, inasmuch as they are always non-binding if only one segment is selected. To see this, consider a particular concave function with two segments. The upper and lower bound constraints are

$$m_1^1 y_1^1 - x_1^1 \geq 0 \tag{4}$$

$$m_1^2 y_1^2 - x_1^2 \geq 0 \tag{5}$$

$$m_1^1 y_1^2 - x_1^2 \leq 0 \tag{6}$$

$$y_1 + y_2 \leq 1 \quad , \tag{7}$$

and the applicable terms of the objective function are

$$f_1^1 y_1^1 + f_1^2 y_1^2 + c_1^1 x_1^1 + c_1^2 x_1^2 \quad .$$

Suppose that x_1^2 is non-zero. Then by Eqs. (4) and (6),

$$\frac{x_1^2}{m_1^2} \leq y_1^2 \leq \frac{x_1^2}{m_1^1} \quad .$$

To minimize the objective function, $y_1^2 = x_1^2/m_1^2$, so that the constraint Eq. (5) is binding and Eq. (6) is non-binding.

Step 8: Necessary Test for Feasibility

An additional test for feasibility is performed. This test examines the 0-1 string produced by the Hillier algorithm to verify that at most one segment of each cost function is specified by the string. To make this check, the constraints

$$\sum_{i=1}^{n_j} y_i^j \leq 1 \quad \text{for} \quad j = 1 \ldots, n \quad ,$$

are examined. If any of these constraints are violated, the program returns to Step 6 and finds a new 0-1 string.

Step 9: Apply Simplex Method to Sub-problem

The sub-problem is reduced in size by letting the lower bounds be zero for all x_i^j. Thus, the subproblem is written as:

$$\text{Minimize} \quad \sum_{j=1}^{n} \sum_{i=1}^{n_j} c_i^j x_i^j \quad ,$$

$$\text{subject to} \quad \mathbf{A'x'} \geq \mathbf{b}$$

$$x_i^j \leq m_i^j y_i^j \qquad i = 1, \ldots n_j \; ; \; j = 1, \ldots n$$

$$x_i^j \geq 0 \quad ,$$

where y_i^j is 0 or 1 as defined by the 0-1 string. This simplification is obtained by making use of the fact that the variable cost is concave and non-decreasing. For example, in the simple two-segment function shown in Fig. 9, suppose that Segment 2 is being considered (that is, Segment 2 is "open" and Segment 1 is "closed"). Then if the optimal

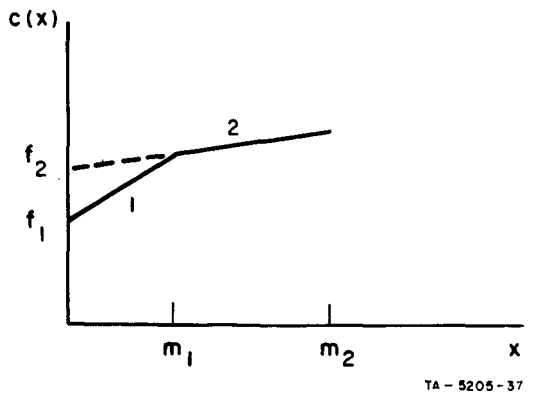


FIG. 9 TWO-SEGMENT CONCAVE FUNCTION

solution of the sub-problem calls for a value of $x \leq m_1$, it is a relatively simple matter to substitute the corresponding value of the function on Segment 1 in evaluating the objective function of the main program. Note that Segment 2 is always above Segment 1 for $x \leq m_1$ (because the objective function is assumed to be concave, non-decreasing) and hence an improvement is guaranteed in this region.

The advantage of neglecting the lower bound is twofold: (1) the number of variables in the dual problem to be solved is reduced because there are fewer constraints; and (2) an optimal rather than an infeasible solution is often reached. The second advantage permits finding lower total cost solutions at an earlier stage of the algorithm, which in turn permits reducing FMAX and, hence, the number of extreme points to be examined.

Step 11: Compute Total Cost

In this step, the actual line segment called for by the optimal solution of the sub-problem is used.

F. Fixed-Charge Problem with Transportation Cost

In a distribution network, the cost structure often involves a fixed charge for opening a facility such as a warehouse, plus transportation costs to each of the customers served from that warehouse. This problem differs from Problem III in that a single fixed charge can be

associated with more than one variable, since a facility can serve all customers. The fixed-charge problem with transportation costs was stated as Problem VI:

$$\left.\begin{aligned} \text{Minimize} \quad & \sum_i \sum_j c_{ij} x_{ij} + \sum_i f_i y_i \\ \text{Subject to} \quad & \sum_i x_{ij} \geq D_j \qquad j = 1, 2, \ldots n \\ & M_i y_i - \sum_j x_{ij} \geq 0 \qquad i = 1, 2, \ldots m \\ & y_i \leq 1 \qquad i = 1, 2, \ldots n \\ & x_i, y_i \geq 0 \end{aligned}\right\} \text{Problem VI}$$

Unfortunately, even modest transportation problems involve large numbers of variables. For example, the sample problems treated by Kuehn and Hamburger [3] involved 24 potential sources and 50 customers, a total of 1200 continuous variables in addition to the 24 integer variables. This creates difficulty in applying the Hillier algorithm to obtain 0-1 strings. These difficulties arise from two sources:

(1) Finding the extreme points of the n-simplex involves solving a linear system of equations; this requires excessive computation time when the number of variables is as large as in the Kuehn-Hamburger problems (1224).

(2) To obtain bounds on the Group 3 variables, it is necessary to solve a linear program consisting of Problem VI with the initial feasible solution as an added constraint. Although it was shown in Section II-D that Problem VI can be reduced to a transportation problem if the y_i are treated as continuous rather than discrete variables, adding the initial solution removes this advantage. Thus, the simplex method has to be used to establish bounds, a not-inconsiderable computational task with 1200 variables.

A further practical difficulty can be encountered. If, as is the case in the Kuehn-Hamburger example, the source locations are also destinations with zero shipping cost (customers located locally) and all fixed costs are equal, the optimal solution with no fixed costs and the initial linear program solution with fixed costs both call for all

sites open. Thus, all y_i are basic and Group 1 in the Hillier algorithm is an empty set. Since the Hillier algorithm works best when Group 1 contains most of the variables, a major advantage of the algorithm is lost.

It is quite clear that theoretically the decomposition idea of specifying 0-1 strings followed by solving transportation sub-problems will lead to an optimal solution. The nub of the problem is to reduce the concept to a form that can be handled by existing machines.

One approach to a computational solution is the following:

(1) Solve the problem as a linear program.

(2) Obtain an initial integer solution by approximate techniques. Several such techniques are in the literature such as the algorithm by Kuehn and Hamburger (see Appendix A for a discussion of the various algorithms).

(3) Establish FMAX and FMIN values.

(4) Use a stripped-down version of the Hillier algorithm to generate 0-1 strings (see below for a discussion). Alternatively, use other branch-and-bound methods for this purpose.

(5) For each 0-1 string, solve the transportation sub-problem to minimize variable cost.

(6) Determine total cost. If this cost is lower than any found previously, store the solution and tighten the FMAX bound.

This series of steps differs from the algorithm presented in Part D in several key respects: It calls for a completely separate calculation to establish an initial feasible solution by some approximate algorithm. FMAX can be established as before. FMIN, the minimum fixed cost, is determined by minimizing the fixed cost subject to the feasibility condition; namely,

$$\text{Minimize} \quad \sum_{i=1}^{n} f_i y_i \quad ,$$

$$\text{Subject to} \quad \sum_{i=1}^{n} M_i y_i \geq \sum_{j=1}^{n} D_j$$

$$y_i = 0 \text{ or } 1 \qquad i = 1, 2, \ldots n \quad .$$

This is a small but simple integer program. Since it has only one constraint it can be solved easily by Everett's method of Lagrange multipliers [15], if the answer is not available by inspection.

The stripped-down version of the Hillier algorithm referred to in Step 4 is the following: Since all variables may be basic in solving the linear program, discard Group 1. In fact, treat all integer variables as Group 3. The Group 3 procedure requires solving a series of linear programs which minimize and maximize the values of each of the variables, given the values of the previous variables. These minimum and maximum values, together with the dual variables, are used in determining bounds on the variables while generating 0-1 strings. In the Hillier algorithm, the constraint set used to establish these bounds includes the condition

$$\mathbf{c}\,\mathbf{x} \leq L_0 \quad ;$$

that is, the value of the objective function must be at most equal to the value of the initial solution. However, adding this constraint eliminates the advantage of solving only a transportation problem. Therefore, it is proposed not to add this constraint. In this case, as is shown in Appendix D, the bounds and the dual variables can be found by solving an equivalent transportation problem.

G. Algorithm for the Fixed-Charge Transportation Problem

1. Problem

The fixed-charge transportation problem described by Balinski [6] is a generalization of a Hitchcock transportation problem whereby a fixed charge, b_{ij}, also is incurred for each link opened. This problem can be stated in the following terms:

Problem VII

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}x_{ij} + f_{ij}y_{ij} \quad ,$$

$$\text{subject to} \quad \sum_{i} x_{ij} \geq D_j \qquad j = 1, \ldots n \qquad (1)$$

$$\sum_{j} x_{ij} \leq S_i \qquad i = 1, \ldots m \qquad (2)$$

$$m_{ij}y_{ij} - x_{ij} \geq 0 \qquad \text{all } i, j \qquad (3)$$

$$x_{ij} \geq 0 \quad , \quad y_{ij} = (0,1) \quad .$$

Constraint sets (1) and (2) express the usual supply and demand requirements. The quantity m_{ij} in constraint set (3) is given by

$$m_{ij} = \min (S_i, D_j)$$

and represents the maximum amount of goods that can be shipped[1] from source i to destination j.

As Balinski showed, an approximate solution to this problem can be obtained by disregarding the indivisibilities of the y_{ij} and recognizing that, at linear programming optimality, constraint set (3) will be satisfied with equality. That is, at LP optimality,

$$m_{ij} y_{ij} - x_{ij} = 0 \quad .$$

Solving for y_{ij} and substituting in the objective function yields

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1} \left[c_{ij} + \left(\frac{f_{ij}}{m_{ij}} \right) \right] x_{ij} \qquad (8)$$

subject to constraint sets (1) and (2), and $x_{ij} \geq 0$.

This problem is an ordinary transportation problem with the cost coefficients adjusted to reflect the fixed charges. Accepting the fixed charge (*i.e.*, rounding y_i up to 1) for any route for which $x_{ij} > 0$ yields Balinski's approximation.

This approximate solution also yields a bound on the maximum fixed charge, FMAX, that can be incurred in the optimum solution. FMAX is calculated in the same way as in the problems considered previously. That is,

$$\text{FMAX} = L_0 - \mathbf{c}\mathbf{x}_0$$

where L_0 is value of the objective function for the approximate integer solution and $\mathbf{c}\mathbf{x}_0$ is the minimum transportation cost if there were no fixed charges.

---

[1] m_{ij} expresses the fact that if the supply at source i is less than the demand at destination j, at most S_i can be shipped over route ij, whereas if the supply exceeds the demand, at most D_j will be shipped over ij.

A crude lower bound, FMIN, on the fixed costs can be found by considering the supply constraints one at a time. Enough routes must be opened from each source to, at least, permit disposing of the supplies there. Thus, consider the sources one at a time and find the cheapest set of routes out of that source which can absorb the supply. Formally, solve the problem

$$\text{Minimize} \quad \sum f_{ij} y_{ij} \,,$$

$$\text{subject to} \quad \sum_j D_j y_{ij} \geq S_i \tag{9}$$

for each source i ($i = 1, 2, \ldots, m$) and then sum these minima. For small problems (*e.g.*, eight destinations) these problems can be solved by exhaustive enumeration (see below).

The algorithm for finding eligible 0-1 strings is structured in such a way that this problem need not be solved explicitly, since it is guaranteed that all 0-1 strings generated will satisfy the source constraints of Eq. (9) and hence will equal or exceed this crude bound, FMIN.

Preliminary tests of the algorithm indicated that the range defined by FMAX and FMIN was often large and that it included a large number of feasible extreme points for which transportation problems had to be solved. To obtain tighter bounds on FMIN, the basis idea for branch-and-bound techniques was added. Suppose that the allowable combination of open routes from Source 1 is specified but open routes from other sources are unspecified. Solving the transportation problem under these conditions specifies the minimum variable cost given these open routes. Thus,

$$\text{FMAX} = L_0 - \text{CX}_1$$

where L_0 is the value of the objective function for the approximate integer solution and CX_1 is the minimum transportation cost if the allowable open routes from Source 1 are specified. Note that

$$\text{CX}_1 = \text{CX}_0$$

and, hence, that the new value of FMAX must be smaller than that obtained when the routes from Source 1 are specified. Therefore, adding

this bounding calculation can (and for most route combinations does) reduce the number of extreme points that need to be examined.

This bounding technique can be applied iteratively; that is, it can be applied with the routes from Source 1 specified, or from Sources 1 and 2, or Sources 1, 2, and 3, and so on. It can also be used selectively; for example, one could specify the open routes from Sources 1 and 5. The question of how much bounding to do is explored in Section IV-C.

2. Generating 0-1 Strings

The key part of the problem is to generate a small number of 0-1 strings. Since there are mn routes, there are 2^{mn} potential 0-1 strings. This number can be reduced at once to $(2^n - 1)^m$ by recognizing that each source must ship to at least one destination.

It is advantageous in what follows to assume that $m > n$. Inasmuch as the role of source and destination can be interchanged in the transportation problem without affecting results, this condition can always be satisfied. We will now present a method for generating 0-1 strings designed to reduce the number of combinations considered to a level manageable on the computer.

The basic idea for enumerating combinations of open routes is to treat these combinations as m-digit numbers in a base 2^n number system. For example, for four sources and three destinations, the number 6312 would denote the combinations of open routes shown in the box.

SOURCE	DESTINATION		
	3	2	1
1	1	1	0
2	0	1	1
3	0	0	1
4	0	1	0

(1 = open, 0 = closed)

Thus, a systematic way of enumerating the combinations of open routes is to count in a base 2^n system starting from 11111 ... 1 and continuing to $2^n 2^n \ldots 2^n$. There are, of course, many regularities in the structure of transportation problems that allow us to throw out many of these numbers. We will make use of the following:

(1) Since the optimal solution in the fixed-charge case must lie at a vertex (for proof, see Balinski [6, Theorem 4]), we need consider only basic solutions. The maximum number of non-zero elements in a basic solution is $m + n - 1$ (*e.g.*, Dantzig [10, p. 301]). Hence, any m-digit number that calls for more than $m + n - 1$ open routes can be discarded.

(2) The total cost cannot exceed FMAX.

(3) Each source must dispose of its entire supply, *i.e.*, the constraint Eq. (9) must be satisfied.

The procedure consists of three parts: initialization, bounding, and iteration. The initialization which is performed once, and arranges the data in a favorable way for the iterations, consists of five steps:

(1) Label the sources in decreasing order of the supplies available.

(2) For each source, compute the total demand for each combination of open routes [a total of $m(2^n - 1)$ values]. If the supply exceeds the demands from a combination of destinations, Eq. (9) is not satisfied and, hence, the combination is infeasible. Assign an effectively infinite fixed charge, M, to such combinations.

(3) For the remaining combinations, compute the total fixed charge associated with opening the combination.

(4) Sequence the combinations in order of increasing fixed cost and store two tables:

(i) The fixed costs in increasing order;

(ii) The combinations that correspond to the ordered fixed costs.

(5) For each row in the fixed cost table prepared in Step 4, find the largest allowable fixed charge in that row. Set all larger fixed charges to M. The largest allowable fixed charge is determined by subtracting the sum of the smallest fixed charges in the other rows from FMAX.

The data are now arranged so that the sum of the fixed costs in the first column corresponds to FMIN defined previously and so that infeasible source-destination combinations are not considered. Each element in the table defined in Step 4(i) above is treated as a digit in a base 2^n number system. The algorithm involves enumerating the combinations systematically, calculating the fixed cost associated with the m-digit numbers.

The bounding procedure involves two steps. The routes for which bounding is to be performed are specified as input and the procedure is carried out whenever the combination of open routes is changed. Assume that the bounding is being done for the kth digit in the m-digit number. Then

(6) Solve the transportation problem with the routes specified for the first k digits open and with *all* routes for digits $k+1$ through m open.

(7) Determine the value of FMAX to be used in the subsequent iterations.

The iterations proceed through the following four steps:

(8) Assume that the source in the last row can ship to any destination (all routes open), and perform the necessary test for column feasibility:

$$\sum_{i=1}^{m} S_i y_{ij} \geq D_j \quad .$$

If this test is failed, increase the next-to-the-last digit by one and repeat. If it is passed, proceed to increment in the last row until the FMAX test is failed. Note that by arranging the supplies in decreasing order, the test is iterative. That is, the test can be applied to the last 2, 3, ... sources together by assuming they can ship to all destinations and applying the column feasibility test.

(9) Check the site combinations called for by each m-digit number to make sure that no more than $n+m-1$ routes are open. Discard combinations that exceed this bound.

(10) Check each m-digit number for feasibility. First, check that at least one route is open to each destination and, second, use the weak and strong feasibility tests (see subsection 4).

(11) If an m-digit number passes the FMAX, $n+m-1$, and destination feasibility tests, solve the transportation problem with only the routes defined by the m-digit number open. Return to Step 7 to obtain another m-digit number, or terminate if there are no more numbers to be examined.

Before describing how the data from the transportation problem are used and how optimality is established, we present an example of Steps 1-5.

3. Example

Consider a problem with four sources and three destinations. For this problem, $m+n-1 = 6$ and the maximum number of allowable 0-1 strings

is $(2^3 - 1)^4 = 2401$. Assume that the supplies, demands, and fixed charges are as shown in Table I. Examining the supplies and demands in Table I shows that the following combinations are infeasible:

From Source 1 to Destination 2, or to 3 or to 2 and 3;
From Source 2 only to Destination 2 or to 3;
From Source 3 only to Destination 3;
From Source 4 only to Destination 3.

Computing the total fixed costs and assigning a value of 1000 to the infeasible combinations yields Table II. Note that the number of combinations that must be considered further has been reduced to 4(5)(6)(6) = 720 from the original 2401.

Table I

FIXED-CHARGE TABLE

SOURCE	DEMAND		
	50	15	5
25	14	14	11
20	15	8	7
15	12	12	9
10	13	11	5

Table II

TOTAL FIXED COSTS

SOURCE	DESTINATION COMBINATION						
	1	2	3	4	5	6	7
1	14	1000	28	1000	25	1000	39
2	15	1000	23	1000	22	15	30
3	12	12	24	1000	21	21	33
4	13	11	24	1000	18	16	29

Sequencing the total costs in increased order for each source and setting up a separate table showing the order of destination combinations yields Tables III and IV.

Table III

SEQUENCED COST TABLE

SOURCE	FIXED COST						
1	14	25	28	39	1000	1000	1000
2	15	15	22	23	30	1000	1000
3	12	12	21	21	24	33	1000
4	11	13	16	18	24	29	1000

Table IV

COMBINATION ORDER

SOURCE	DESTINATION COMBINATION						
1	1	5	3	7	2	4	6
2	1	6	5	3	7	2	4
3	1	2	5	6	3	7	4
4	2	1	6	5	3	7	4

One feasible solution involves opening routes (1,1) (2,1) (3,2) (4,1) and (4,3) with a fixed cost of 59. For the cost structure used in the test problem, this solution results in an FMAX value of 65. Using this value, additional entries can be set to 1000 in Table III. The reason is that the total fixed cost when these combinations are used even with the cheapest entry (first column) for all other sources

exceeds 65. A revision of Table III to take this additional information into account is shown in Table V. The total number of combinations has now been reduced to 200 from the original 2401 and the previous 720. Of the combinations that remain, 189 fail the tests in Steps 8, 9, and 10, as follows:

	No. of Combinations Failing Tests
First column feasibility check	30
FMAX test	136
Number of open routes exceeds $n + m - 1$	2
No route open to some destination	8
Tests for column feasibility	13

The eleven combinations that pass all tests require solution of a transportation problem. The power of the FMAX test is illustrated here, since at most 64 of the combinations pass it.

Table V

SEQUENCED COST TABLE (Revised)

SOURCE	FIXED COST						
1	14	25	1000	1000	1000	1000	1000
2	15	15	22	23	1000	1000	1000
3	12	12	21	21	24	1000	1000
4	11	13	16	18	24	1000	1000

4. Feasibility Tests

To reduce the number of transportation problems that have to be solved, a series of feasibility tests are performed (Step 10 above). The simplest of these is to check the open routes to make sure that at least one route is open to each destination. If this is the case, the weak feasibility test can be applied. In this test, each source is assumed capable of shipping its entire supply to all customers to which routes are open. An example is shown in Part a. of Table VI where the numbers indicate the open routes and the maximum shipment. The test consists of summing the available supplies at each destination and checking whether this sum is greater than or equal to the total demand. If it is not, the test is failed and that extreme point need not be considered further.

Table VI

EXAMPLE OF STRONG FEASIBILITY TEST

SUPPLIES	DEMANDS			
	45	35	20	15
35	35			
30	30			
25		25	20	
15		15		15
5				5
5			5	

a.

Numbers in matrix indicate open routes and maximum supplies that can be shipped over them. Weak feasibility test is passed.

SUPPLIES	DEMANDS			
	0	35	15	10
0	0			
20	0			
25		25	15	
15		15		10
0				0
0			0	

b.

Remove stock from rows that serve only one customer.

SUPPLIES	DEMANDS			
	0	35	0	0
0	0			
20	0			
10		10	0	
5		5		0
0				0
0			0	

c.

Remove stock from columns which serve only one customer.

Conclusion: infeasible.

If the weak feasibility test is passed (as it is in Part a. of Table VI) the strong feasibility test is applied. This test, illustrated in Parts b. and c. of Table VI, consists of the following steps:

(1) For each destination supplied by a single source, reduce the supply at that source by the demand and eliminate that demand from further consideration.

(2) For each source supplying only one demand, decrease the demand by the source supply or to zero, whichever is larger.

(3) Work back and forth between Steps (1) and (2) until no further improvements are possible. At each step, perform the weak feasibility test.

5. Transportation Problem

The 0-1 strings generated define a set of open routes. There is no guarantee of feasibility. Furthermore, even if feasible, there is no guarantee that applying a transportation problem algorithm starting from this set of open routes will produce this set of routes as the minimum-cost solution. Since the transportation structure is desirable computationally, it can be retained if the unit shipping cost is set large, say M, over each closed route. Applying a transportation problem algorithm will then lead to one of two results:

(1) an optimal value of the objective function larger than M, which implies that the originally selected set of open routes does not lead to a feasible solution; or

(2) an optimal value less than M.

If any of the open routes are not used, then their fixed cost can be subtracted from the total fixed cost.

If the solution obtained has a total cost (fixed plus variable) which is better than any found previously, then the FMAX bound can be tightened.

6. Summary of Algorithm

(1) Solve the transportation problem with no fixed costs.

(2) Solve the transportation problems with unit costs of $(c_{ij} + b_{ij}/m_{ij})$.

(3) Compute FMAX given a partial solution.

(4) Generate a 0-1 string that satisfies FMAX, FMIN, row feasibility, and column feasibility tests, and has at most $m+n-1$ open routes.

(5) Solve the transportation problem with the unit costs for closed routes set equal to M. If the total cost is better than any found thus far, store the result and go to Step 3. Otherwise, go to Step 4.

The algorithm terminates when no new 0-1 string can be found. The optimal solution is the lowest total cost solution found during the computations.

IV COMPUTATIONAL EXPERIENCE

A. Fundamental Fixed-Charge Algorithm

Four warehouse location problems were investigated using the fundamental fixed-charge algorithm (Section III-D). These problems had the characteristics listed in Table VII. The entire constraint set and the objective function for Problem 1 is shown in Fig. 10. Figure 11 shows the **A** matrix and the **b**, **c**, and **f** vectors for Problems 2, 3, and 4. Note that for convenience, the cost of each item of stockpile was taken as 1 and the fixed cost of each site was expressed as a multiple of the unit stockpile cost. The fixed costs ranged between 45 and 60

Table VII

CHARACTERISTICS OF TEST PROBLEMS FOR FUNDAMENTAL FIXED-CHARGE ALGORITHM

PROBLEM NUMBER	NUMBER OF POTENTIAL SITES	MINIMUM TOTAL STOCKPILE	NUMBER OF EQUATIONS IN **A** MATRIX
1	9	189	10
2	15	325	18
3	19	212	20
4	30	252	28

Calculations were performed for the nine-warehouse problem for upper bounds of 190, 100 and 69 on the stockpile at any one site. Two types of initial feasible solutions were assumed: one was obtained by rounding[1] the linear programming solution of the problem, the other by assuming all sites open and full (*i.e.*, containing stockpile equal to the upper bound).

Because the work reported here was experimental in nature, it proved convenient to break the computations into three parts and to use the results of one part as input to the next. This division into parts also allowed experiments to be performed, such as the use of different initial solutions, and permitted modification of the codes without requiring continual recomputation.

The first part consisted of solving three linear programs: one including the fixed and variable costs, one with the fixed costs set to zero, and one with the variable cost set to zero. This part provided

[1] That is, if $y_i > 0$, set $y_i = 1$.

CONSTRAINT SET																			B VECTOR	
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0		29	
0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0		75	
0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0		24	
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0		29	
0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0		71	
0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0		89	
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1		4	
0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1		23	
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1		68	
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1		189	
189	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0		0	
0	180	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0		0	
0	0	189	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0		0	
0	0	0	181	0	0	0	0	0	0	0	0	-1	0	0	0	0	0		0	
0	0	0	0	182	0	0	0	0	0	0	0	0	-1	0	0	0	0	≥	0	
0	0	0	0	0	189	0	0	0	0	0	0	0	0	-1	0	0	0		0	
0	0	0	0	0	0	183	0	0	0	0	0	0	0	0	-1	0	0		0	
0	0	0	0	0	0	0	189	0	0	0	0	0	0	0	0	-1	0		0	
0	0	0	0	0	0	0	0	184	0	0	0	0	0	0	0	0	-1		0	
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0		-1	
0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0		-1	
← y COMPONENTS →									← x COMPONENTS →											

OBJECTIVE FUNCTION

F VECTOR: 53 54 52 55 56 51 57 50 58

C VECTOR: 1 1 1 1 1 1 1 1 1

TB-5205-38

FIG. 10 CONSTRAINT SET AND OBJECTIVE FUNCTION FOR THE 9-SITE PROBLEM

A MATRIX																B VECTOR
0	0	1	1	1	1	0	0	0	0	0	0	0	0	0		29
1	0	1	1	1	1	0	0	0	0	0	0	0	0	0		75
1	0	1	0	0	0	1	0	0	0	0	0	0	0	0		24
0	1	1	0	0	0	1	0	0	0	0	0	0	0	0		29
1	0	1	1	0	0	1	0	0	0	0	0	0	0	0		71
1	1	1	1	0	1	1	0	0	0	0	0	0	0	0		89
0	0	0	1	1	0	0	1	1	0	0	0	0	0	0		4
0	0	1	1	1	0	0	1	1	0	0	0	0	0	0		23
0	0	1	1	1	1	0	1	1	0	0	0	0	0	0		68
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	≥	31
0	0	0	0	0	0	1	0	1	1	1	1	0	0	0		77
0	0	0	0	0	0	1	0	1	0	0	0	1	0	0		33
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0		76
0	0	0	0	0	0	1	0	1	1	0	0	1	0	0		94
0	0	0	0	0	0	0	0	0	1	1	0	0	1	1		7
0	0	0	0	0	0	0	0	1	1	1	0	0	1	1		41
0	0	0	0	0	0	0	0	1	1	1	1	0	1	1		57
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		325

OBJECTIVE FUNCTION

F VECTOR: 51 53 55 55 57 59 53 54 52 55 56 51 57 50 58

C VECTOR: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

TB-5205-39

FIG. 11 **A, b,** AND **c** FOR PROBLEMS 2, 3, AND 4

A MATRIX

																			B VECTOR
0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	24
0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	97
0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	99
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	20
0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	119
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	29
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	30
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	24
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	98
0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	68
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	91
0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	1	0	0	120
1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	21
1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	24
0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	23
0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	106
0	0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	50
0	0	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	0	0	50
0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	46
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	212

$\leq$

OBJECTIVE FUNCTION

F VECTOR: 59 58 56 51 59 55 53 52 51 55 57 56 53 55 51 51 51 50 51

C VECTOR: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

TA-5205-40

FIG. 11 Continued

A MATRIX																														B VECTOR
0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	3
0	0	0	0	1	0	0	1	0	1	0	1	0	0	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	3
0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1	0	0	0	14
0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	4
0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	0	1	1	1	0	0	0	18
0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1	1	1	0	1	0	0	1	0	0	0	0	1	0	26
0	0	0	0	1	0	0	1	1	0	0	1	1	0	1	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	4
0	0	0	0	1	0	0	1	1	0	0	1	1	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	1	62
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	1	0	1	0	0	1	0	149
0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	1	0	1	0	0	1	0	1	0	0	1	0	80
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0	1	0	0	1	0	110
0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	0	1	0	1	0	0	1	0	182
0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	1	0	8
0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	0	1	1	0	1	0	88
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1	0	49
1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	5
0	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	5
1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	1	0	23
0	0	0	0	0	0	0	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	0	1	0	0	0	1	1	0	89
0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	0	23
0	0	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	0	0	1	0	1	0	1	1	0	106
0	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	1	3
0	0	1	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0	1	18
0	0	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	12
0	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1	51
0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	8
0	0	0	0	0	0	1	0	0	1	1	0	0	1	1	0	1	1	1	0	1	0	0	1	0	1	0	0	1	1	155
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	252

≥

OBJECTIVE FUNCTION

F VECTOR: 45 46 47 48 49 50 51 52 53 54 55 44 43 42 41 40 59 58 57 56 55 56 47 48 59 50 60 41 42 43

C VECTOR: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

TB-5205-41

FIG. 11 Concluded

the optimal value of the problem treated as a linear program, a list of basic and non-basic variables required for the Hillier algorithm, a rounded initial solution if desired, values of FMAX and FMIN, and bounds on the fixed costs.

The second part consisted of initializing the Hillier Algorithm by computing the values of the dual variables required for Group 3. For each basic integer variable it is necessary to solve two linear programs, one maximizing and the other minimizing the value of the variable given the values of the preceding Group 1, Group 2 and Group 3 variables. These linear programs are solved only once and their results could be put conveniently on punched cards and used as inputs for the algorithm proper. As mentioned several times previously, the linear program package for the Burroughs B-5000 is general and hence not very efficient. As a result, solving these programs is a relatively lengthy process and it proved economical to do it "off-line" rather than on-line. As it was, the program for solving these linear programs was improved somewhat from the program originally used by Hillier. The main penalty paid for separating this part of the program was that certain set-up operations (such as finding extreme points of the n-simplex, and Group 1 and Group 2 initializations) were repeated. Because the amount of duplication involved in set-up was relatively small, no attempt was made to improve efficiency by saving some of the intermediate results that were calculated in both parts.

The final part consisted of the algorithm proper. Two methods were used in the algorithm to generate 0-1 strings. The first of these was the Hillier algorithm as described in Section III-B. The second was an enumeration subroutine. This subroutine was introduced when it became clear that if the problem being considered had very few basic variables, a disproportionate amount of time was being spent to run Part 2 to compute dual variables for the Hillier algorithm compared to the total time required to solve the problem. It was thought possible that, instead of using the Group 3 procedure, it might be more economical to enumerate the completions between FMIN and FMAX given the values of the Group 1 and Group 2 variables. Although this approach would result in generating more 0-1 strings than with the Hillier algorithm, the extra time taken in examining them was expected to be small in comparison to the time spent on Part 2.

A second variation examined consisted of setting up the nine-warehouse problem without stating the $y \leq 1$ constraint explicitly in Parts 1 and 2. This approach has the advantage that it eliminates one constraint per site, thereby reducing the size of the problem that has to be solved. The upper bound of 1 still is used as an input to the portion of the Hillier algorithm which generates 0-1 strings. The disadvantage is that eliminating this set of constraints can (and did) lead to linear program solutions having some y_i variables greater than one. The experience obtained indicates that if the upper-bound constraints on warehouse capacity are loose compared to requirements, no difficulty is encountered. However, if the warehouse capacities are small, the linear program will tend to use small warehouses with attractive fixed charges beyond $y_i = 1$. Thus, to use this approach it is necessary to check the results of the initial linear program solutions to make sure that the $y \leq 1$ constraint is satisfied. If it is not, the larger problem must be solved.

Table VIII lists the cases examined and the solutions obtained. The number of sites and the total cost associated with the LP rounded solution are also shown, as is the percent improvement provided by the optimal solution.[1]

The timings achieved are listed in Table IX. The times are elapsed processor times on the Burroughs B-5500, a medium-speed multiprocessing machine. The multiprocessing feature causes the time required for a program to vary somewhat from run to run. In the nine-site problem various initial solutions were tried: the LP rounded solution referred to earlier; a feasible three-site solution which was non-optimal; and a solution in which all nine sites were initially open. The last case had two variations: all sites filled to capacity initially, and all sites with an initial stockpile of 95. The table also shows the number of basic variables for each problem. The major time required was in computing the dual variables for initializing Group 3. The larger the problem and the larger the number of basic variables, the longer the time that was required for this part.

Table IX also lists the cases in which the enumeration subroutine was used. In these cases, the total computation time after setup was essentially the same as when the Hillier Group 3 procedure was used. Thus, for these small number of variables, the enumeration routine is advantageous.

[1] Computed conservatively by taking 100 x (Initial-Optimal)/Optimal solution.

Table VIII

SUMMARY OF PROBLEMS SOLVED USING FUNDAMENTAL ALGORITHM

NUMBER OF POTENTIAL WAREHOUSES	TOTAL DEMAND	UPPER BOUND ON WAREHOUSE SITE	STOCKPILE ALLOCATION		TOTAL FIXED COST	TOTAL COST	APPROXIMATE SOLUTION [1]		IMPROVEMENT
			Site Number	Amount			Number of Open Sites	Cost	
9	189	190	3 8	185 4	102	291	3	349	16.6%
		109	3 8	109 80	102	291	3	342	14.9%
		69	1 3 8	69 69 51	155	344	4	395	13.6%
15	325	109	1 3 9 14	109 29 94 93	218	533	4	533	0%
		59							
19	212	150	4 8 13 17 18	53 53 32 46 30	257	471	7	575	18.1%
30	252	45-252	4	52	173	425	5	467	9.1%

[1] Based on LP rounded approximate solution.

Table IX

SUMMARY OF TIMINGS ON B-5500 COMPUTER FOR THE FUNDAMENTAL ALGORITHM

PROBLEM	UPPER BOUNDS	INITIAL SOLUTION	NUMBER OF BASIC y_i VARIABLES	TIME (seconds)				
				LP Solution	Extreme Point Bounding	Dual Variable Computation	Computation After Setup	Total
9-site	190	LP rounded	6	33	33	143	8	206
	109	LP rounded	6	32	24	120	7	183
	69	LP rounded	6	31	23	149	4	200
	190	3 site, non-optimal	6	31	19	162	11	223
	109[1]	3 site, non-optimal	3	4	17	16	13	40
	109[1]	all 95	3	4	21	16	9	50
	109[1]	all full	3	4	21	16	9	50
	109[1]	all full	3	4	20	enumeration[2]	8	32
15-site	109	LP rounded	4	37	52	201	80	370
19-site	150[1]	LP rounded	7	89	106	361	44	590
30-site	≤ 252[1]	LP rounded	5	106	290	698	32	1317
	≤ 252[1]	LP rounded	5	106	281	enumeration[2]	24	411

[1] $y_i \leq 1$ constraints not explicitly stated.

[2] Enumeration subroutine rather than dual variables used.

It would be anticipated that a cross-over would occur and that the Hillier algorithm would be preferred if the number of basic variables increased and hence the number of combinations that had to be examined increased. In the cases considered here there were at most 2^3 combinations in Group 3 to be examined for each partial completion specified by Groups 1 and 2.

Considerable time was saved if the $y_i \leq 1$ constraints did not have to be stated explicitly when solving the initial linear program. These cases are noted in Table IX. The linear program in Part 1 was run without these constraints and the answer obtained had $y_i < 1$ for all i. This answer is fortuitous; it is not guaranteed. Cases with y_i values larger than 1 should be expected (and have been observed)[1] for all problems with small values for the upper bounds M_i on the x_i, since the constraint $M_i y_i - x_i \geq 0$ is not sufficient by itself to guarantee that $x_i \leq M_i$. The time savings result both from a reduction in the number of constraints, which makes the problem smaller throughout the algorithm, and from a reduction in the number of y_i variables which are basic in the LP solution, which reduces the initial dual variable computation. If the $y_i \leq 1$ constraints are stated explicitly, some of the y_i became basic at 0 level when solving the initial linear program. This did not occur when these constraints were removed. In applying the Hillier algorithm, y_i could take on only values of 0 or 1. Thus, the $y_i \leq 1$ constraint is implicit in the algorithm and need not be stated.

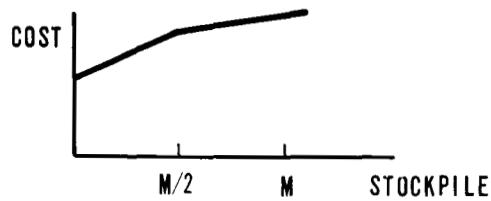
Examination of the output data indicates that a major portion of the work in reducing the number of combinations that had to be examined was done by the Group 1 procedures, which examine the non-basic variables in the LP solution. It was also found that starting with a seemingly bad solution (all sites open and full which results in the maximum possible cost for a feasible solution) did not affect computation time. This was due in part to the test for zero stockpile incorporated in the program. As described in Section III-D, the Hillier algorithm specifies a set of open sites which are used to define the subproblem. Being a linear program, the subproblem provides a basic solution for stockpile allocations. This solution can involve the allocation of zero stockpiles to one or more open sites. Such zero stockpiles were encountered, usually in the first several subproblems solved. In these cases the sites with

[1] The nine-site problem with upper bounds of 69 (see Table IX) showed this effect.

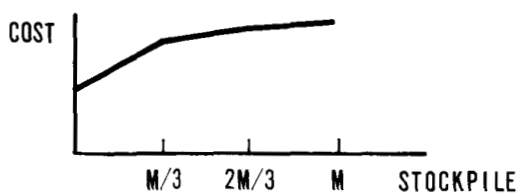
zero stockpile need not be opened and hence there is no fixed charge for them. By noting when these cases occurred, both the FMAX and the best current solution (L_t) bounds could be tightened to the same level as when starting with near-optimal solutions.

B. Non-Linear Variable Cost

The nine-warehouse problem used to explore the general algorithm was expanded to provide a test case when the objective function is non-linear. Eight of the sites were assumed to have two-segment cost functions of the form shown in the following diagram, where M is the upper bound on stockpile.



One site (Site 8) was assumed to have a three-segment cost function of the form shown below.



The constraint matrix and the objective function for this problem are shown in Figure 12. It can be seen that there are now 38 variables and 38 constraints instead of 18 variables and 28 constraints.

The computing times were

LP solution	144 seconds
Extreme point bounding	79
Dual variable computation	604
Computation after set-up	30
Total	857

The large time required for dual variable computation resulted because ten of the components of the y vector were basic variables. No particular difficulty was encountered in coding or in solving this example.

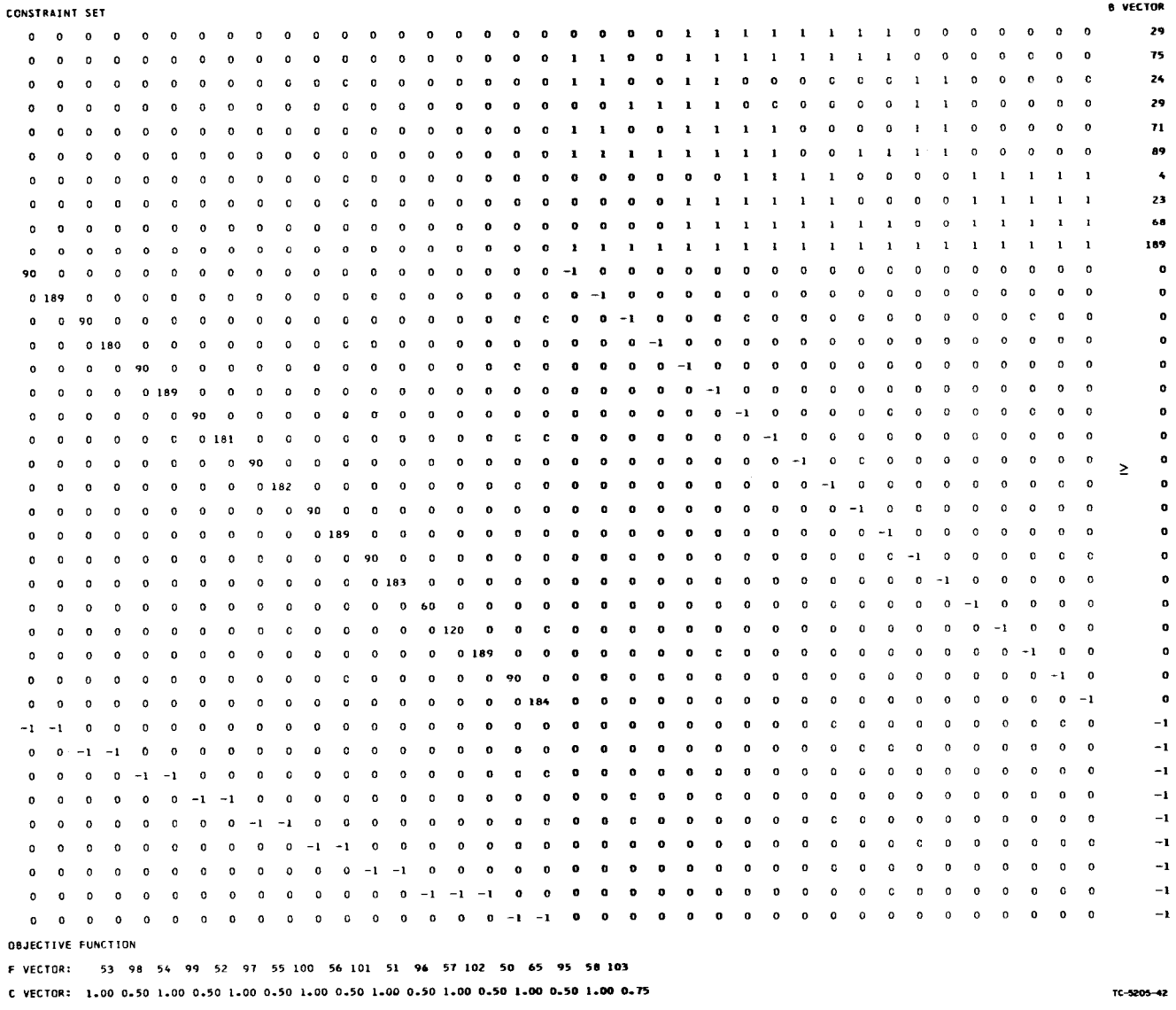


FIG. 12 NON-LINEAR OBJECTIVE FUNCTION PROBLEM

C. The Fixed-Charge Transportation Problem

The algorithm for the fixed-charge transportation problem described in Section III-G was coded and run for problems having transportation matrices of dimension 3 × 4, 4 × 6, 5 × 7, and 6 × 8. A total of nine problems were run.[1] The demands, supplies, unit transportation costs, and fixed charges for these problems are shown in Figure 13. The variable costs for all except Problems 3 and 8 were generated by using subsets of the data presented by Balinski [6] for his test problem. The variable costs for Problems 3 and 8 are based on Murty's sample problem [7]. The fixed charges for most of these sample problems were also based on Balinski's and Murty's data. Balinski's fixed charges were obtained by using random numbers between 10 and 20. In Problem 6, the fixed charges were obtained by using random numbers between 0 and 99 and in Problem 7 the fixed charges were increased to the order of 100 and 200 to obtain a problem in which the fixed costs are large compared to the variable costs.

Since the algorithm requires the solution of a series of transportation problems, a relatively efficient transportation code was used as a subroutine.[2] In the 5 × 7 problems, for example, an average of 0.6 second was required to solve each transportation problem.

To specify open and closed routes, the cost matrix was adjusted so that closed routes had unit shipping costs equal to the total cost of the initial approximate solution. In this way, capacity constraints did not have to be introduced into the transportation problem algorithm and an optimal solution was always obtained. If the set of routes specified as open were actually infeasible, the infeasibility would be reflected in the optimal transportation problem solution containing a closed route and having a large associated cost.

Table X is a summary of the solutions obtained. Three values of total cost are shown:

(1) The approximate solution obtained by solving the problem with no fixed costs and accepting the fixed charges for those routes opened (labeled XPORT);

---

[1] Two variations on Problem 1 and a variation of Problem 9 were also run. The first two differed only in that 20 and 50 were added to each fixed charge. These variations are labeled Problems 1a and 1b. The variation of Problem 9 (labeled 9a) involved adding 250 to each fixed charge.

[2] This code is PROCEDURE XPORT programmed in ALGOL by M. Chambreau in 1965 for Stanford Research Institute's B-5500 computer. It is based on a paper by Glicksman *et al.*, in Naval Research Logistics Quarterly in 1960 [16].

SUPPLIES ARE 50 15 5

DEMANDS ARE 25 20 15 10

B[I,J]

14	15	12	13
14	8	12	11
11	7	9	5

C[I,J]

7.60	0.71	2.83	5.94
5.94	0.64	1.70	5.64
5.94	0.69	0.79	2.02

1

SUPPLIES ARE 45 35 30 25

DEMANDS ARE 35 30 25 15 10 10 5 5

B[I,J]

11	16	18	17	10	20	17	13
14	17	17	13	15	13	16	11
12	13	20	17	13	15	16	13
16	19	16	11	15	12	18	12

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83	2.02	5.64
1.01	0.75	0.88	0.59	1.50	2.63	2.26	5.64
1.05	1.06	1.08	0.64	1.22	2.37	1.66	5.64
1.94	1.50	1.56	1.22	1.98	1.98	1.36	6.99

4

SUPPLIES ARE 40 35 25 20 5

DEMANDS ARE 35 25 25 15 10 10 5

B[I,J]

91	216	218	217	210	220	217
94	217	217	213	215	213	216
92	213	220	217	213	215	216
96	219	216	211	215	212	218
99	218	215	216	212	214	220

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83	2.02
1.01	0.75	0.88	0.59	1.50	2.63	2.26
1.05	1.06	1.08	0.64	1.22	2.37	1.66
9.01	5.01	5.01	2.01	9.01	9.01	3.06
1.61	1.40	1.61	1.33	1.68	2.83	1.54

7

SUPPLIES ARE 45 35 20 15

DEMANDS ARE 35 30 25 15 5 5

B[I,J]

11	16	18	17	10	20
14	17	17	13	15	13
12	13	20	17	13	15
16	19	16	11	15	12

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83
1.01	0.75	0.88	0.59	1.50	2.63
1.05	1.06	1.08	0.64	1.22	2.37
1.94	1.50	1.56	1.22	1.98	1.98

2

SUPPLIES ARE 40 35 25 20 5

DEMANDS ARE 35 25 25 15 10 10 5

B[I,J]

11	16	18	17	10	20	17
14	17	17	13	15	13	16
12	13	20	17	13	15	16
16	19	16	11	15	12	18
19	18	15	16	12	14	20

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83	2.02
1.01	0.75	0.88	0.59	1.50	2.63	2.26
1.05	1.06	1.08	0.64	1.22	2.37	1.66
1.94	1.50	1.56	1.22	1.98	1.98	1.36
1.61	1.40	1.61	1.33	1.68	2.83	1.54

5

SUPPLIES ARE 23 26 38 56 75

DEMANDS ARE 55 54 35 35 22 9 8

B[I,J]

4	3	19	0	6	8	7
10	1	2	5	35	4	26
5	16	4	24	9	11	2
12	5	18	10	6	9	43
8	31	5	6	12	36	19

C[I,J]

19	6	20	12	16	13	24
11	8	5	15	17	40	13
5	29	25	8	19	109	26
38	17	26	14	23	27	114
6	20	17	2	92	29	42

8

SUPPLIES ARE 23 38 56 66

DEMANDS ARE 55 54 35 22 9 8

B[I,J]

4	3	0	6	8	7
5	16	24	9	11	2
12	5	10	6	9	43
8	31	6	12	36	19

C[I,J]

19	6	12	16	13	24
5	29	8	19	109	26
38	17	14	23	27	114
6	20	2	92	29	42

3

SUPPLIES ARE 40 35 25 20 5

DEMANDS ARE 35 25 25 15 10 10 5

B[I,J]

10	22	24	42	37	77	99
15	46	48	93	39	6	72
1	25	22	6	81	11	56
2	85	97	61	16	42	69
81	30	76	7	6	27	98

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83	2.02
1.01	0.75	0.88	0.59	1.50	2.63	2.26
1.05	1.06	1.08	0.64	1.22	2.37	1.66
1.94	1.50	1.56	1.22	1.98	1.98	1.36
1.61	1.40	1.61	1.33	1.68	2.83	1.54

6

SUPPLIES ARE 45 40 30 20 10 5

DEMANDS ARE 35 30 25 20 15 10 10 5

B[I,J]

11	16	18	17	10	20	17	13
14	17	17	13	15	13	16	11
12	13	20	17	13	15	16	13
16	19	16	11	15	12	18	12
19	18	15	16	12	14	20	19
13	20	20	17	15	12	14	11

C[I,J]

0.69	0.64	0.71	0.79	1.70	2.83	2.02	5.64
1.01	0.75	0.88	0.59	1.50	2.63	2.26	5.64
1.05	1.06	1.08	0.64	1.22	2.37	1.66	5.64
1.94	1.50	1.56	1.22	1.98	1.98	1.36	6.99
1.61	1.40	1.61	1.33	1.68	2.83	1.54	4.26
5.29	5.94	6.08	5.29	5.96	6.77	5.08	0.31

9

TC-5205-43

FIG. 13 FIXED-CHARGE TRANSPORTATION PROBLEMS

(2) The approximate solution obtained by using $(c_{ij} + f_{ij}/M_{ij})$ as transportation cost (labeled Balinski);

(3) The exact solution obtained from the algorithm. Three values are shown: total cost, transportation cost and fixed cost.

In addition, the initial and final values of FMAX and minimum transportation cost are shown.

Table X

FIXED-CHARGE TRANSPORTATION PROBLEM SOLUTIONS

PROBLEM NO.*	DIMENSIONS	SOLUTIONS			FMAX		MINIMUM VARIABLE COST
		XPORT	Balinski	Exact	Initial	Final	
1	3 x 4	335	335	329 (270/60)#	72	67	263
2	4 x 6	220	229	202 (103/99)	121	103	99
3	4 x 6	1999	1999	1999 (1947/52)	52	52	1947
4	4 x 8	302	291	273 (165/108)	141	123	150
5	5 x 7	282	267	245 (128/117)	148	126	119
6	5 x 7	549	369	317 (173/144)	250	198	119
7	5 x 7	2060	2189	1638 (166/1472)	1910	1488	150
8	5 x 7	2297	2289	2289 (2230/59)	75	75	2214
9	6 x 8	353	349	314 (120/194)	171	136	178
1	3 x 4†	455	429	429 (270/160)	167	167	263
1b	3 x 4§	635	570	570 (270/310)	317	270	263
9	6 x 8△	3353	3357	2357 (2119/230)	3175	2179	178

* Numbers correspond to numbers in Figure 13.
† Identical to Problem 1 with 20 added to each fixed charge.
§ Identical to Problem 1 with 50 added to each fixed charge.
△ Identical to Problem 9 with 250 added to each fixed charge.
(270/60) denotes a variable cost of 270 and a fixed cost of 60.

The results shown in Table X have several interesting features. In three cases (Problems 2, 7, and 9a) accepting the fixed charges associated with the transportation solution (XPORT approximation) was better (lower total cost) then using the Balinski approximation method. On the other hand, the Balinski method yields an optimal solution for problems 1a, 1b, 3 and 8. Although this result reinforces Balinski's claim that his approximation method improves as the

fixed charges become a larger fraction of the total cost, Problem 9a does not support this conclusion.

For the problems listed the methods of approximation give answers which range from exact to as much as 15 percent high in total cost and 20 percent high in fixed cost. There appears to be no way of judging *a priori* the extent to which the approximate solution differs from the exact solution.

A word is in order about Murty's solution to his test problem. His sample calculation contains a numerical error in that he computes the total fixed charge in his optimal answer at 46 rather than the correct value of 59. This error leads him to consider all vertices with variable cost 2260 when he should use 2247 based on his ultra-conservative lower bound of 16 on the fixed charge. However, merely insisting on row-feasibility leads to a fixed-charge bound of 30, which is sufficient to ensure that his solution will be optimal for this problem.

Before presenting the computer time results for these problems it is instructive to analyze the number of combinations being dealt with. The number of combinations of open and closed routes possible is 2^{mn} since each route can be either open or closed. However, since each destination must receive goods, we must have at least n routes open (assuming $n \geq m$). Furthermore, as pointed out several times previously, we need consider only basic solutions and the maximum number of open routes in a basic solution is $n + m - 1$. Hence, an upper bound on the number of combinations to be considered is

$$\sum_{i=n}^{n+m-1} \binom{n}{i} \binom{m}{i} .$$

Typical values for this sum are shown in Table XI. Table XI also shows the number of combinations that passed all tests and required solution of transportation problems for Test Problems 1, 2, and 5.

It is seen from Table XI that for a 5×7 problem, the algorithm presented reduced the number of vertices to be examined by solving a transportation problem from the order of 10^8 to the order of 10^2, which is a manageable number. However, the number of vertices goes up very rapidly with problem size. For an 8×12 problem we are dealing with

Table XI

UPPER BOUND ON NUMBER OF BASIC SOLUTIONS
OF THE FIXED-CHARGE TRANSPORTATION PROBLEM

	MATRIX SIZE				
	3 x 4	4 x 6	5 x 7	7 x 9	8 x 12
mn	12	24	35	63	96
n	4	6	7	9	12
$n + m - 1$	6	9	11	15	19
2^{mn}	4096	$1.67(10^7)$	$3.43(10^{10})$	$9.2(10^{18})$	$8(10^{28})$
$\sum_{i=n}^{n+m-1} \binom{n\ m}{i}$	2211	$2.45(10^5)$	$6.2(10^8)$	$1.7(10^{14})$	$8(10^{19})$
No. of transportation problems solved (typical)	6	28	234		

the order of 10^{19} vertices. Although it can be expected that many of of these would be eliminated by the algorithm, the number of transportation problems to be solved still goes up quite rapidly. Furthermore, the search time to find the vertices to be examined also rises rapidly. A test run using Balinski's 8 × 12 test problem, for example, required 2 minutes to find the first vertex that passed all tests prior to solving a transportation problem.

The increase in computing times with problem size may be seen from the results listed in Table XII. The times are broken into four parts (SETUP, GENERATE, XPORT, OUTPUT) which correspond to (1) the time required to obtain the initial approximate solutions and FMAX and to sort the fixed charges, (2) the time to find a 0-1 string which passes all tests, (3) the time to solve the required transportation problems for both 0-1 strings and branch-and-bound, and (4) the time required for setting up output and printing it out.

The amount of time required is governed principally by the time spent in GENERATE, that is, the time required to find extreme points which pass all tests. The time in XPORT of course increases with the number of transportation problems to be solved. However, the increase does not appear to be as rapid as the increase in GENERATE.

Murty's 5 × 7 problem (Problem 8) was one in which the fixed cost is very much smaller than the variable cost. This problem required

Table XII

COMPUTER TIMES FOR FIXED-CHARGE TRANSPORTATION PROBLEM

PROBLEM[1]	SIZE	SETUP	GENERATE	XPORT	OUTPUT	TOTAL	NUMBER OF TRANSPORTATION PROBLEMS	
							For 0-1 Strings	For Branch-and-Bound
1	3 x 4	6.7	0.3	0.4	0.3	7.7	2	4
2	4 x 6	9.8	16.1	4.8	1.9	32.6	23	22
3	4 x 6	11.0	10.1	3.3	2.0	26.3	12	16
4	4 x 8	12.4	97.2	41.8	20.0	171.4	401	10
5	5 x 7	12.9	211.8	27.4	11.7	263.8	191	44
6	5 x 7	12.2	103.5	22.8	8.4	146.9	97	70
7	5 x 7	12.4	74.0	8.8	2.8	97.0	42	13
8	5 x 7	12.6	2937.1	209.5	103.6	3262.8	1437	97
9	6 x 8	17.4	1404.3	77.6	28.2	1510.1	434	61
1a	3 x 4	6.6	0.3	0.4	0.3	7.6	2	4
1b	3 x 4	6.8	0.4	0.4	0.2	7.8	2	4
9a	6 x 8	17.2	69.3	1.5	0.6	71.4	5	4

[1] Numbering is in accordance with Figure 13.

almost 55 minutes to complete. The Balinski approximation method yields the optimal solution for this and was found prior to the iterations. Thus, the iterations were verifying optimality. To do so, all feasible vertices within the range of fixed charges between 31 and 75 had to be examined even though the optimal answer had a fixed charge of 59.

A separate test run was made for Murty's problem with FMAX arbitrarily set to 50 initially. No improvement in the value of the objective function was, of course, found. A total time of 402 seconds was required for this run with 173 transportation problems solved. This faster approach bounded the optimal solution as being between 2264 and 2289 and so guaranteed that the approximate answer obtained from Balinski's procedure is correct within 1.2 percent.

The approach of speculating on a value of FMAX to speed computations and to obtain either a solution or a bound on the solution was also used for the 6 × 8 problem. In this case, a test run indicated that nearly 10 minutes were required to explore all combinations involving the first combination for Sources 1 and 2. Therefore, a

relatively low value of FMAX was inserted as input and the optimal solution found in 25 minutes. As a further check on this phenomenon, the fixed costs were made dominant relative to the variable costs for the 6 × 8 problem by increasing each fixed cost by 250. Again, a low value of FMAX was assumed and a solution was reached in little over a minute.

A word is also in order about the use of the branch-and-bound feature. Experiments performed principally with the 4 × 6 problems indicated it to be most advantageous to use this feature only on the first two sites. That is, whenever a new combination in Site 1 or 2 was considered, the minimum variable cost given that combination of open routes was computed and FMAX determined. Using one site only did not seem to reduce running time appreciably. On the other hand, using the first two sites did introduce considerable time saving (a factor of two in some cases).

There is no point to increasing beyond two the number of sites for which branch-and-bound is performed because of the rapidly diminishing returns. If the average number of combinations to be considered for each site is n, then if branch-and-bound calculations are performed for Sites 1 and 2, a total of n^2 transportation problems have to be solved; with k sites a total of n^k transportation problems have to be examined. This exponential relation makes it inadvisable to carry branch-and-bound beyond two levels.

Examination of the test runs indicated that large numbers of branch-and-bound transportation problems were being solved near the end of the search with few new vertices found. These cases, involving higher fixed costs at Sites 1 and 2, also involved site combinations that contained previously examined combinations as subsets. It was therefore found desirable to make the program adaptive and to use search rather than branch-and-bound near the end of the process. That is, the following heuristic procedure was adopted for deciding when to use branch-and-bound:

(1) Compute the fixed cost associated with Site 1 plus Site 2 plus the lowest cost (first column in Table V) for all other sites.

(2) If this cost (which is the minimum cost given the combination of routes open from Sites 1 and 2) is more than a fixed percentage of FMAX, do not solve the transportation problem but do search for a better solution with the current value of FMAX.

In the results reported here, branch-and-bound calculations were discontinued when the cost computed in Step 1 was greater than 90 percent of FMAX. This value, chosen arbitrarily, proved to be a good one in that when tested on the 4 × 6 problems it reduced the number of transportation problems and the total time.

Based on the results obtained thus far, it is clear that the algorithm provides a computationally feasible way of obtaining exact solutions to fixed-charge transportation problems of small size (say, 6 × 8 or less). The algorithm appears best suited for those cases in which the fixed charge is dominant and in which there is a wide range of values of the fixed charges. Further work is required to reduce computing times to a point where they are suitable for larger problems. Some approaches to achieving these improvements are described in Section VI.

V DUALITY

A. Previous Work

Until Balas [17] presented his theory for duality in discrete programming in mid-1967, duality theory for mixed-integer programming had not been well developed. Only two previous papers on this subject were found in the literature, one by Gomory and Baumol in 1960 [18] and the other by Alcaly and Klevorick in 1966 [19].

Both of the previous papers are concerned with interpreting dual prices implied by the linear program solved at optimality in the Gomory cut method [*e.g.*, Ref. 9, Ch. 26]. The interpretation of the dual prices is based on the dual variable in the final augmented linear program which is solved, *i.e.*, the one from which the integer solution is obtained. The results presented are deemed unsatisfactory by the authors of both papers because they are not unique. An alternative view of the cause of the difficulty is mathematical. The usual linear-programming interpretation of dual variables revolves around arguments about the nature of the solution at the extreme point of the constraint set. Since the linear problem is continuous, it is possible to talk about partial derivatives and hence rates of change of individual variables. Continuity does not exist for the integer variables, making it unreasonable to expect that derivatives obtained by adding artificial constraints will have meaning.

Balas, by addressing himself to finding a generalized dual of integer programs directly, was able to develop a duality theory which has the following desirable properties:

(1) The dual of the dual is the primal.

(2) Complementary slackness conditions hold.

(3) If an optimal solution to the primal exists, then both a solution of the dual and a global saddle point exist.

In this Section, Balas' results are applied to the fixed-charge problem to find the dual and explore its significance.

B. Definition of the Dual of the Fixed-Charge Problem

The primal of the upper-bounded fixed-charge problem (Problem III) may be written as:[1]

$$\begin{aligned} \text{Maximize} \quad & -fy - cx \quad , \\ \text{subject to} \quad & -Ax < -b \\ & -My + Ix \leq 0 \\ & y = 0,1 \\ & x \geq 0 \quad . \end{aligned}$$

There are n primal variables, n_1 of which are integers and $n - n_1$ $(= n_1)$ are continuous. The number of constraints is m.

According to Balas' theory, the dual is then written as:

$$\begin{aligned} & \underset{y}{\text{Max}} \ \underset{u}{\text{Min}} \quad u^2 \begin{pmatrix} -b \\ 0 \end{pmatrix} - v^1 y \quad , \\ & \text{subject to} \quad u^2 \begin{pmatrix} 0 & -A \\ -M & I \end{pmatrix} - v = \begin{pmatrix} -f \\ -c \end{pmatrix}^T \\ & \qquad\qquad u, y \geq 0 \\ & \qquad\qquad y = 0,1 \end{aligned}$$

v_j unconstrained if $j \in N_1$

$v_j \geq 0$ if $j \in N - N_1$.

[1] An alternative formulation would involve adding the constraint $y \leq 1$ and specifying y_i integer. The same conclusions are reached in either case.

Here $\mathbf{u}$ and $\mathbf{v}$ are the dual vectors having m and n components, respectively. The $\mathbf{v}$ is decomposed into

$$\begin{pmatrix} \mathbf{v}^1 \\ \mathbf{v}^2 \end{pmatrix}$$

where $\mathbf{v}^1$ has N_1 components corresponding to the $\mathbf{y}$ components. For mixed-integer programs $\mathbf{u}$ is equal to $\mathbf{u}^2$ since it turns out that $\mathbf{u}^1$ has no components. Note that the same $\mathbf{y}$ vector appears in both the primal and the dual.

For simplicity in what follows, $\mathbf{u}^2$ will be decomposed into $\mathbf{u}_1$ and $\mathbf{u}_2$ vectors corresponding to the two sets of equations in the primal problem. Thus, we write the dual as

$$\max_{\mathbf{y}} \min_{\mathbf{u}_1, \mathbf{u}_2} (\mathbf{u}_1 \mathbf{u}_2) \begin{pmatrix} -\mathbf{b} \\ \mathbf{0} \end{pmatrix} - \mathbf{v}^1 \mathbf{y} \quad ,$$

$$\text{subject to} \quad (\mathbf{u}_1 \mathbf{u}_2) \begin{pmatrix} \mathbf{0} & -\mathbf{A} \\ -\mathbf{M} & \mathbf{I} \end{pmatrix} - \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix}^T = \begin{pmatrix} -\mathbf{f} \\ -\mathbf{c} \end{pmatrix}^T$$

$$\mathbf{u}_1, \ \mathbf{u}_2, \ \mathbf{y}, \ \mathbf{v}^2 \geq \mathbf{0}$$

$$\mathbf{y} = 0, 1 \quad .$$

Multiplying out results in:

$$\max_{\mathbf{y}} \min_{\mathbf{u}_1, \mathbf{u}_2} \{ - \mathbf{u}_1 \mathbf{b} - \mathbf{v}^1 \mathbf{y} \} \quad ,$$

$$\text{subject to} \quad -\mathbf{u}_2 \mathbf{M} - \mathbf{v}^1 = -\mathbf{f} \qquad (1)$$

$$-\mathbf{u}_1 \mathbf{A} + \mathbf{u}_2 \mathbf{I} - \mathbf{v}^2 = -\mathbf{c} \qquad (2)$$

$$\mathbf{u}_1, \ \mathbf{u}_2, \ \mathbf{y}, \ \mathbf{v}^2 \geq 0$$

$$y_i = 0, 1 \quad .$$

Thus the dual is a max-min problem that involves maximizing over the same fixed-charge vector y that appears in the primal. It will be convenient in what follows to substitute for v^1 from (1) to obtain:

$$\begin{aligned} &\underset{y}{\text{Max}} \quad \underset{u_1, u_2}{\text{Min}} \quad (-u_1 b + u_2 M y - f y) \quad , \\ &\text{subject to} \quad -u_1 A + u_2 I - v^2 = -c \qquad (3) \\ &\qquad\qquad u_1,\ u_2,\ y,\ v^2 \geq 0 \quad . \end{aligned}$$

Note that a typical equation in (3) has the form

$$-u_1 A_i + u_{2i} - v_i^2 = -c_i \quad ;$$

that is, the ith equation contains only the ith components of A, u_2, and c. Here A_i is the vector defined by the ith column of the A matrix.

C. Interpretation of the Dual

By Balas' saddle-point theorem, the value of the primal and dual objective function will be equal at optimality. Thus, if the optimal solution of the primal is known, the value of the optimal solution of the dual is also known. Furthermore, since y appears in both the primal and the dual and since it must have the same value at optimality, if the optimal y is known from the solution of the primal problem, the values of u_1 and u_2 at optimality in the dual problem can be determined by solving a linear program.

Using superscript bars to denote value at optimality, we have

$$-\bar{f}\,\bar{y} - \bar{c}\,\bar{x} = -\bar{u}_1 b + \bar{u}_2 M \bar{y} - \bar{f}\,\bar{y} \quad .$$

Since $\bar{f}\,\bar{y}$ appears on both sides, we have, furthermore, that the variable cost $-\bar{c}\,\bar{x}$ is equal to $-\bar{u}_1 \bar{b} + \bar{u}_2 M \bar{y}$. This relation is also obtained from substituting into one of Balas' three complementary slackness conditions.

Insight into the meaning and value of u_1 and u_2 can be obtained by applying Balas' other two complementary slackness conditions:

$$\bar{u}^2 \bar{z}^2 = 0 \qquad (4)$$

$$\bar{v}^2 \bar{x} = 0 \quad , \qquad (5)$$

where z^2 is the slack vector in the primal and is given by

$$z^2 = \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & -A \\ M & I \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} \quad \cdot \quad \underline{\text{Dual Vector}} \quad \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Substituting $\bar{u}^2 = \begin{pmatrix} \bar{u}_1 \\ \bar{u}^2 \end{pmatrix}$ in Eq. (4) and multiplying yields

$$\bar{u}_1(b - A\bar{x}) = 0 \tag{6}$$

$$\bar{u}_2 M y - I\bar{x} = 0 \tag{7}$$

If the optimal solution of the primal is known, complementary slackness conditions (5) and (7) can be used to determine the values of the individual components of $\bar{u}_2$ and $\bar{v}^2$, that is, $\bar{u}_{2i}$ and $\bar{v}_i^2$. The values depend on the value of x_i at optimality. Applying Eqs. (5) and (7) to the objective function and to Condition (3) at optimality results in the following tabulation:

$\bar{x}_i$	$\bar{y}_i$	$\bar{u}_{2i}$	$\bar{v}_i^2$	CONDITION (3) AT OPTIMALITY	CONTRIBUTION OF $\bar{u}_2 M y$ TO OBJECTIVE FUNCTION
0	0	≥ 0	≥ 0	$-\bar{u}_1 A_i + u_{2i} - v_i^2 = -c_i$	0
$0 < x_i < m_i$	1	0	0	$-\bar{u}_1 A_i = -c_i$	0
m_i	1	≥ 0	0	$-\bar{u}_1 A_i + \bar{u}_{2i} = -c_i$	$\bar{u}_{2i} m_i$

The important result is that a component of $\bar{u}_2$ contributes to the value of the objective function only if the corresponding site is filled to capacity ($\bar{x}_i = \bar{m}_i$) and that in this case a positive term appears. Since the max-min problem calls for minimization over u_2, $\bar{u}_{2i}\bar{m}_i$ can be thought of as a penalty cost which results from having Site i filled to capacity. That is, if $\bar{u}_{2i} m_i > 0$, then it would be economically advantageous to put additional supplies at Site i were it not for the capacity constraint. If several sites are used to capacity, the values of $\bar{u}_{2i} m_i$ can be used to determine at which site it is most advantageous to increase capacity. By comparing the values of $\bar{u}_{1i}$ with $\bar{u}_{2i}$ it is possible to determine whether a greater marginal return can be obtained from increasing the stockpile at a site not used to capacity or from

expanding capacity. If a site is open but only partially filled ($0 < x_i < m$) then the shadow cost for additional capacity ($\bar{u}_{2i}$) is zero since excess capacity is available. If a site is closed ($x_i = 0$) the value of $\bar{u}_{2i}$ is a measure of the marginal return which could be obtained if the fixed charge did not have to be incurred.

Another economic interpretation in terms of subsidies and penalties[1] can be obtained by applying Balas' Theorem 6[17, p. 24]. He shows that if $(\bar{\mathbf{y}}, \bar{\mathbf{x}})$ is an optimal solution of the primal, then it is also an optimal solution of the following linear program:

$$\begin{aligned} \text{Maximize} \quad & (-\mathbf{f} + \mathbf{s})\mathbf{y} - \mathbf{c}\,\mathbf{x} \quad , \\ \text{subject to} \quad & -\mathbf{A}\,\mathbf{x} \leq -\mathbf{b} \\ & -\mathbf{M}\,\mathbf{y} + \mathbf{I}\,\mathbf{x} \leq 0 \\ & \mathbf{y} \leq 1 \\ & \mathbf{x}, \mathbf{y} \geq 1 \quad , \end{aligned}$$

where

$$s_i = \begin{cases} \bar{v}_i^1 & \text{if } \bar{y}_i = 1 \\ \min\,[0,\ \bar{v}_i^1] & \text{if } \bar{y}_i = 0 \end{cases}$$

and, from Eq. (3)

$$\bar{v}_i = f_i - \bar{u}_i^2 m_i \quad .$$

Balas' results show that

(a) If $\bar{y}_i = 0$ and $\bar{x}_i = 0$ then $s_i \leq 0$;

(b) If $\bar{y}_i = 1$ and $0 < x_i < m_i$, then $s_i = f_i$;

(c) If $\bar{y}_i = 1$ and $x_i = m_i$, then $0 \leq s_i \leq f_i$.

[1] E. Balas, private communication (October 1967).

In other words:

(a) There may be a *penalty* ($s_j \leq 0$) associated with not opening some sites.

(b) Open sites not used to capacity must bear a *subsidy* equal to the fixed charge.

(c) Even if an open site is operated at full capacity it may have to bear a *subsidy*, but this subsidy will not exceed the fixed charge.

These penalties and subsidies are the direct result of the integrality requirement on y. Since $(f + s) \leq 0$ and $y_i \geq 0$, the term $(-f + s)y$ in the objective function indicates the additional cost resulting from the inability of the y_i to take on fractional values.

Condition (6) is the usual complementary slackness condition of linear programming. Thus, the components of u_1 can be interpreted as conventional shadow prices.

D. Conclusions

In this section we have briefly examined duality theory for the linear fixed-charge problem. Balas' theory has been applied to formulate the dual and to explore the implications of complementary slackness conditions. The dual problem is a mixed-integer program like the primal, but it does not have the primal's simple structure. Thus, no advantage seems to be gained by attempting to solve the dual rather than the primal. Conversely, if the dual of the fixed-charge problem is encountered, it is advantageous to formulate the primal and solve it by the methods of this dissertation.

VI CONCLUSIONS AND DIRECTIONS FOR FURTHER WORK

The following conclusions are based on the algorithms presented and the computational tests which have been performed. The experience gained has inevitably led to ideas for improving the algorithms, both in terms of approach and computational strategy. These improvements are discussed under "Directions for Further Work," (Part B).

A. Conclusions

This dissertation has presented and explored the concept of decomposing fixed-charge problems into a pure-integer master program and a series of linear subprograms. The objectives sought and achieved were algorithms for obtaining exact solutions to fixed-charge problems in which the continuous variables have upper bounds. The algorithms developed are primal algorithms in that they provide continually improving feasible solutions in their search for the optimum.

Four computational algorithms were presented:

(1) The fundamental fixed-charge algorithm suitable for linear constraints and non-negative linear costs for the continuous variables. A fixed charge may be associated with each activity engaged in at non-zero level.

(2) A modification of the fundamental algorithm for non-linear variable cost functions.

(3) An algorithm for solving the fixed-charge problem in which the costs are transportation costs and fixed charges are associated with each source opened.

(4) An algorithm for solving the fixed-charge transportation problem in which a fixed charge is associated with each route opened.

Computational experience is reported for Algorithms (1), (2), and (4).

Limited computational experience has been obtained for the fundamental fixed-charge algorithm for problems having up to 30 variables. The test problems were motivated by the site-selection problem in which a number of sites are available at which activities can be located and

it is desired to select the lowest-cost subset that meets minimum performance requirements. Although satisfactory results were obtained for a 30-site problem, computation times for this 30-site problem were sufficiently large to indicate that larger problems (*e.g.*, 50 or more sites) would be expensive to run with the algorithm as developed.

The Hillier algorithm as modified proved to be quite suitable for the decomposition approach. By bounding the region to be explored and making extensive use of the structure of the problem, this algorithm results in requiring very few linear programs to be solved as subproblems. However, the Hillier algorithm proves to be one of the major limitations on problem size that can be handled at present because it requires two time-consuming computations in setting up: (1) solution of a set of simultaneous linear equations for establishing weights on the extreme points[1] and (2) solutions of a large number of linear programs for establishing the bounds in Group 3. If the number of variables in Group 3 is small, then enumerating the extreme points between FMAX and FMIN provides a way of overcoming the second limitation.

Examination of the detailed printout showed that, for the three algorithms tested computationally, improved solutions and the optimal solution are obtained early in the iterations and that the major portion of the time is spent in search verifying optimality. For the nine-site test problems, the quality of the initial feasible solution did not appear to have an appreciable effect on computing time for the fundamental algorithm. However, preliminary experiments with the fixed-charge algorithm with transportation costs (Algorithm 3) indicated that good starting solutions would be of considerable help there.

The algorithm for the fixed-charge transportation problem (Algorithm 4) provided satisfactory results for problems of size up to 6×8. Here the quality of the initial solution and the bounds seem particularly important since the algorithm searches exhaustively between FMIN and FMAX bounds. The algorithm seems to be particularly suitable when the fixed costs are large compared to the variable costs, whereas the Murty algorithm seems to be more suitable for large variable and small fixed costs.

[1] The computer code for the Hillier algorithm solves a set of simultaneous equations for *each* extreme point. As pointed out by Hillier in [11] this is not necessary. Also, the Hillier code uses an existing but very inefficient LP program for obtaining duals. Major improvements in the running time for the Hillier algorithm are certainly possible and appear easy to obtain. Thus, it is to be anticipated that efficient coding will increase the number of variables that can be dealt with economically.

The FMAX and FMIN bounds are powerful tools for reducing the size of the lattice to be searched. However, they still require exploration of a large number of extreme points. For example, in a typical case, the value of FMAX associated with the optimal solution may be 75 whereas the optimal solution has a total fixed cost of 60. Nonetheless, even if the optimal solution is found on the first iteration, a large number of feasible extreme points with fixed charges between 60 and 75 must still be checked to make sure that there is no solution whose fixed cost is greater than 60 but whose total cost is lower overall.

The total computational experience indicates that the approach of decomposing of mixed-integer programs into a pure-integer master program and linear subprograms provides a fruitful approach to solving site-selection and similar fixed-charge problems.

B. Directions for Further Work

There are various improvements in the algorithms that appear to merit further investigation. These are discussed here, together with some alternative approaches, and various calculations are suggested that will permit more complete assessment of the merits of the algorithms.

The fundamental fixed-charge algorithm makes heavy use of the Hillier integer-programming algorithm to generate 0-1 strings. Computational experience indicates that, in its present form, the Hillier algorithm is suitable for calculations involving of the order of 30 continuous and 30 integer variables. The Hillier algorithm is, of course, not basic to the decomposition approach; its use was dictated by interest in the algorithm and its ready availability. Other ways of generating 0-1 strings, such as branch-and-bound techniques, should be investigated. Improvements in the Hillier algorithm as they become available will also increase the size of problem that can be handled economically.

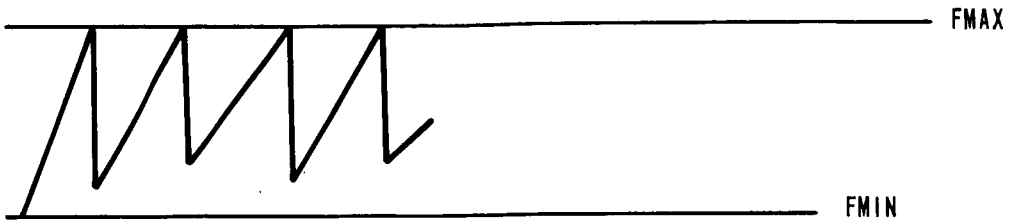
As was pointed out in the conclusions, enumerating extreme points within the FMAX and FMIN limits for the Group 3 variables provides considerable time savings in the Hillier algorithm if the number of basic variables is small. The time is saved by eliminating the need for solving two linear programs for each Group 3 variable. In return, more 0-1 strings have to be evaluated because the dual variable approach in Group 3 takes into account the constraints on the entire problem

(including those generated by the continuous variables), whereas the enumeration procedure looks at only the constraints on the integer portion of the problem. Thus, the enumeration procedure disregards much of the information available for generating eligible 0-1 strings. A tradeoff point may exist where the number of 0-1 strings generated by the counting procedure becomes large enough (because of the number of Group 3 variables) to make it more efficient to use Hillier's Group 3 procedure. Explorations of this question for the fundamental algorithm should be undertaken.

A separate approach is to improve further the methods of calculating the dual variables required for initialization of Group 3 in the Hillier algorithm. At present, one Phase I and two Phase II simplex calculations are performed for each Group 3 variable. Successive problems being solved differ only slightly in that an additional variable is held fixed in the constraint set. This suggests that rather than starting from Phase I each time, the dual simplex method can be used once the first Group 3 variable calculations have been completed. Another aspect of the Hillier algorithm which deserves further work is improving the efficiency of the computations required to determine the extreme points of the n-simplex and their weights.

For the fixed-charge problem with transportation costs (Algorithm 3), only an initial computational scheme has been developed. Many details still warrant further exploration. For example, it may be useful to impose a series of simple tests before solving the transportation subproblem. One such test would determine the minimum transportation cost associated with a 0-1 string in the absence of supply limitations. Specifically, the cost of shipping the entire demand over the cheapest route from an open site would be determined. Such a solution, though generally not feasible, does provide a lower bound on the variable cost and can be used to determine whether it is worthwhile to pursue this 0-1 string further.

An aspect of Algorithm 4 that deserves further exploration is the strategy for considering extreme points. As presented in Section III-G, the algorithm searches among all the extreme points between FMAX and FMIN. In so doing, it follows a zigzag path between these bounds. An alternative search pattern is to pick an arbitrary FMAX lower than the initial one. Call it FMAX1. If a better solution is found using FMAX1,



it is retained and the algorithm proceeds normally. If no improvement is found, the algorithm is restarted and only fixed-charge values between FMAX1 and FMAX are examined. Such a restart capability would be facilitated by storing the locations in the fixed-charge table at which the FMAX1 bound was reached. The advantage of this scheme is that if a better solution is found, a lower value of FMAX is established and fewer extreme points need to be examined. Nothing is lost if an improvement is not found, since all feasible extreme points with fixed charges below FMAX1 have to be examined anyway. Furthermore, a better bound has been obtained on the optimal solution.

From a computational point of view, setting FMAX1 too low results in a low probability of finding a better solution, whereas setting it too high results in very little computation saving. A reasonable value for FMAX1 may be the average of FMAX and FMIN.

A complementary approach to the use of a boundary for reducing FMAX in the fixed charge transportation problem would be to combine the present algorithm with the Murty algorithm. Both algorithms use a decomposition approach; the Murty algorithm searches systematically among the extreme points of the transportation subproblems and iteratively decreases the maximum allowable variable cost, whereas Algorithm 4 searches systematically among the extreme points defined by the fixed charges and iteratively decreases the maximum allowable fixed cost.

It will be recalled that

$$\text{FMAX} = L_0 - \mathbf{c}\,\mathbf{x}_0$$

where L_0 is the initial approximate solution and $\mathbf{c}\,\mathbf{x}_0$ is the minimum variable cost. There are two ways of reducing FMAX: finding a better (lower-cost) approximate solution or determining a higher bound on the variable cost. The latter is exactly what the Murty algorithm does.

Thus, a combined algorithm would obtain an approximate solution as at present and determine FMAX. It would then use the Murty algorithm to increase cx_0, thereby reducing FMAX. At some point, the Murty algorithm would be discontinued (either because of excessive branching or excessive number of iterations) if an optimal solution has not been found, and Algorithm 4 would be initiated with the best value of FMAX found thus far. Such a combined algorithm appears attractive as a means for increasing the size of the fixed-charge transportation problem that can be solved economically.

APPENDIX A

LITERATURE SURVEY

APPENDIX A

LITERATURE SURVEY

The operations research literature on location problems and fixed-charge problems was surveyed as an initial step in preparing this dissertation. All back issues of the *International Abstracts in Operations Research* were checked out and, where possible, the relevant papers obtained. Various articles appearing as references in the papers consulted were also examined. In addition, several articles that have appeared within the last year have been reviewed.

The literature on location problems can be divided into two categories: (1) papers concerned with the optimal placement of facilities in two dimensions where there are no fixed costs; and (2) papers concerned with the fixed-charge problem discussed in this dissertation. In this appendix, eight papers in the second category are summarized. One of the more recent papers in the first category is that of Hillier and Connors [22], which also includes a bibliography of previous work.

Each of the eight papers discussed in this appendix presents an algorithm for solving fixed-charge problems. The first four of these (Kuehn-Hamburger, Manne, Feldman *et al.*, and Jandy) contain approximate algorithms for warehouse location. The fifth paper (Efroymson-Ray) presents an exact algorithm for warehouse location for the case of unlimited warehouse size. The final three papers (Levy, Cooper and Drebes, and Dwyer) present approximate algorithms for special cases: the lock box problem, the linear fixed-charge problem, and the fixed-charge transportation problem. Two methods for obtaining optimal solutions to fixed-charge problems with upper bounds (the Benders partitioning algorithm and the Murty algorithm) are described in Appendices B and C.

"A Heuristic Program for Locating Warehouses"
A. A. Kuehn and J. J. Hamburger
Management Science, Vol. 9, No. 4 (July 1963), pp. 643-667.

Kuehn and Hamburger recognize that the basic problem of locating warehouses involves a trade-off (my phrase, not theirs) between the marginal cost of warehouse operation and the increased profits resulting from lower transportation costs and more rapid delivery. The paper is quite thorough, presenting the algorithm, details of a sample problem, a programming formulation (but not a solution) and a description of previous methods including those of Baumol and Wolfe[1], Balinski and Mills[2], and Shycon and Maffei[3]. The last is a simulation approach.

Basic Idea

The program is divided into two parts, a main program which locates warehouses one at a time until no additional warehouses can be added without increasing cost, and a bump-and-shift routine which tries to improve the initial solution by removing and by interchanging warehouses. Kuehn and Hamburger make use of the following three heuristics in setting up their algorithm:

(1) Good locations for regional warehouses will be at or near concentrations of demand. Therefore, only the largest demand points need be considered as potential warehouse locations.

(2) A sub-optimal warehousing system can be developed by adding one warehouse at a time. Thus, at each stage of the algorithm, choose the warehouse that offers the biggest improvement and terminate when no improvements can be made.

(3) At each stage only a subset of the potential locations need be considered for detailed evaluation. Choose as members of this subset those locations which offer the greatest local improvement.[4]

---

[1] W. J. Baumol and P. Wolfe, "Warehouse Location," *Operations Research*, Vol. 6, No. 2, (March-April 1958), pp. 232-263.

[2] M. L. Balinski and H. Mills, "A Warehouse Problem," *Mathematica*, Princeton, N.J., (April 1960).

[3] H. N. Shycon and R. B. Maffei, "Simulation—Tool for Better Distribution," *Harvard Business Review* (Nov.-Dec. 1960), pp. 65-75.

[4] Local improvement refers to the decrease in transportation cost if a city is served by its own warehouse rather than from another city.

The bump-and shift-routine is designed to check on the solution derived in two ways: (a) any warehouse which is no longer economic because some of its customers have been shifted to newly located warehouses is eliminated (bumped) and its customers reassigned, and (b) attempts are made to interchange each remaining site with others in the territory it serves.

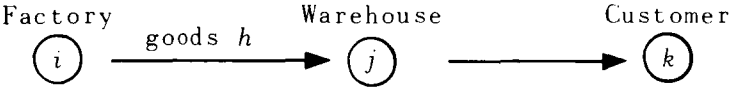
Sample Computation

The results of twelve sample problems based on all the combinations of three factory locations and four levels of fixed warehouse costs are presented. Fifty destinations, 24 of which are potential warehouse sites are considered. At any stage, only five of the potential warehouses, selected on the basis of maximum local demand, are considered for addition to the warehouse network. This set of data is quite useful as a means of comparing alternative methods, and has been used by Feldman *et al.*, as a convenient standard by which to judge their algorithm.

Kuehn and Hamburger claim that the time required to obtain a solution increases as problem size increases at a much slower rate than it does with the simplex method.

The authors mention briefly the possibility of starting with all warehouses operative and removing them one at a time. They feel that this approach would be computationally inefficient unless more than half the potential warehouses are retained. Thus, a typical application would be the selection of warehouses to be closed.

Mathematical Formulation



Define the following quantities:

X Amount of goods

A Unit transportation cost from factory to warehouse

B Unit transportation cost from warehouse to customer

$D(T)$ Implicit cost of a delay of T units in delivery

F Planned fixed cost of operating a warehouse per time period

S Semi-variable cost of operating a warehouse

Q Demand

W Capacity of a warehouse

Y Capacity of a factory

Z A 0-1 variable that is one only if a warehouse is open

i,j,k,h Subscripts that denote factory, warehouse, customer, and type of goods, respectively.

The objective is to minimize

$$\sum_{h,i,j,k}(A_{hij} + B_{hjk})X_{hijk} + \sum_{j} F_j Z_j + \sum_{h,j} S_{hj}\left(\sum_{ik} X_{ijk}\right) + \sum_{h,k} D_{hk}\,(T_{hk})$$

subject to the following constraints:

(1) All demands must be supplied,

$$\sum_{i,j} X_{hijk} = Q_{hk} \quad , \quad \text{for all } h,k$$

(2) Factory capacity limits,

$$\sum_{j,k} X_{hijk} \leq Y_{hi} \quad , \quad \text{for all } h,i$$

(3) Warehouse capacity,

$$I_j\left(\sum_{h,i,k} X_{hijk}\right) \leq W_j \quad , \quad \text{for all } j \quad ,$$

where I is an arbitrary function describing the inventory level.

Note that this formulation is quite general and takes into account upper bound limits on warehouses and factories as well as costs of shipping delays. The algorithm as described in the paper does not take the limits or opportunity costs into account.

Baumol-Wolfe Solution

Kuehn and Hamburger review the Baumol-Wolfe formulation which is based on strictly concave cost functions with a fixed initial cost. They conclude that the Baumol-Wolfe algorithm, based on the marginal warehouse cost, does not take fixed costs into account explicitly in generating solutions and that the degree of concavity has strong influence on the solution obtained. They then show that an improved (lower-cost) solution of the Baumol-Wolfe sample problem can be obtained either by using their algorithm or by perturbing the Baumol-Wolfe algorithm.

Balinski-Mills Solution

Balinski and Mills present a solution for the single-factory, single-product case with delivery and storage costs treated as piecewise linear. They linearize the problem by assuming that the warehousing cost is approximated by the average unit cost of operating the warehouse at some high level (such as capacity). This leads to an approximate solution which they show to be a lower bound on the problem. Putting the solution back into the original integer formulation leads to an upper bound. Unfortunately, these bounds are quite loose.

Shycon-Maffei Simulation

Shycon and Maffei are in the simulation business and they offer to simulate any proposed warehousing scheme. According to Kuehn and Hamburger, the simulation method is not really suited to parameter variation, although Shycon and Maffei claim that it can be done. In a two-page, heated rebuttal, printed after the Kuehn-Hamburger article, Shycon and Maffei argue that the real world is not well modeled by Kuehn and Hamburger because of the simplistic treatment of transportation costs and order patterns. The differences in customer order patterns reflect differences in timing of orders, size of orders, product mixes, and so on. Transportation costs are affected by the rate structures and by such phenomena as size of order, geographic area, minimum charges per shipment, and cartage.

"Plant Location Under Economies of Scale—Decentralization and Computation"
Alan S. Manne
Management Science, Vol. 11, No. 9 (November 1964), pp. 213-235.

Manne evaluates the steepest-ascent, one-point-move algorithm (which he abbreviates SAOPMA) proposed by Reiter and Sherman[1] for discrete optimizing problems. He limits his discussion to the case of a single product, a single degree of vertical integration, a single time period, known demands, and no capacity limitations. Manne concludes that the technique works well in this highly restricted case, but he warns that its performance under more realistic assumptions is not vouched for.

[1] See, for example, S. Reiter and G. R. Sherman, "Allocating Indivisible Resources Affording External Economies or Diseconomies," *International Economic Review*, (January 1962).

Basic Assumptions

Manne makes the following basic assumptions:

(1) There are I possible sources of supply (plants).

(2) There are J distinct market points to be served.

(3) The requirements at the jth market are R_j.

(4) Costs are the sum of manufacturing costs plus shipping costs. Shipping costs are linear in quantity shipped but economies of scale are possible in manufacturing.

Formulation as a Programming Problem

$$\text{Minimize} \quad \sum_i a_i y_i + \sum_{i,j} b_{ij} x_{ij} \quad ,$$

$$\text{subject to} \sum_i X_{ij} = R_j \qquad j = 1, 2, \ldots J \quad ,$$

$$y_i = 0 \text{ implies } x_{ij} = 0 \text{ for all } j \quad ,$$

$$y_i = 1 \text{ implies } x_{ij} > 0 \text{ for some } j \quad ,$$

$$x_{ij} \geq 0 \; ; \; y_i = 0 \text{ or } 1 \quad ,$$

where a_i is the fixed charge for plant i if this plant is built and b_{ij} is the unit manufacturing and shipping cost for the product if built at plant i and shipped to market j.

Solution Approach

The formulation of the problem is that of a standard transportation problem once the y_i are known. SAOPMA makes use of this by starting at an arbitrary lattice point in the unit hypercube defined by the y_i and then searching for the next *adjacent* feasible lattice point which results in the greatest improvement. Thus, for example, if the vector (0 1 0 0) represents the initial solution of Plant 2 open and three other plants closed, SAOPMA searches among the vectors (1 1 0 0), (0 1 1 0), and (0 1 0 1) to find which, if any, combination of open plants leads to improvement. If none do, the program stops. If one or more do, the combination with the biggest improvement is picked. The search is repeated, starting each time with the current vector, until no more improvements can be achieved.

Computer Results

Manne presents a series of results for 6-, 8-, and 10-point-symmetrical cases which are sufficiently small so that complete enumeration is possible for comparison with SAOPMA results. He chose potential plant locations by picking numbers at random on a unit square. Destinations were assumed to be identical to potential locations. A further randomization was introduced by multiplying the distance by a random number between 0.9 and 1.1 to account for variable transportation rates between centers and using this product as the transportation cost. Three levels of fixed charge were tested. Market requirements, fixed charges, and unit manufacturing costs were assumed to be constant throughout. Fifty maps were constructed for each of the three symmetrical cases. The results showed that, as was to be expected, the larger the fixed charges the fewer the plants. Analyses of the results also showed that the SAOPMA led to tolerable errors in almost all cases. In some 1350 cases tested, only four errors of more than 10 percent were encountered.

Manne also gives some asymmetrical results where market requirements, or fixed charges, or unit manufacturing cost varies from plant to plant and finds that for his 8-point examples he still gets reasonable results.

The computer tests are encouraging, and clever from a design viewpoint, but are far removed from real data.

"Warehouse Location Under Continuous Economies of Scale"
E. Feldman, F. A. Lehrer, and T. L. Ray
Management Science, Vol. 12, No. 9 (May 1966), pp. 670-684.

This is one of a pair of papers from Esso Research and Engineering; the other, by Efroymson and Ray, is described later. Whereas Efroymson and Ray take an integer programming approach (which was originally designed to test the results of this paper), Feldman *et al.* take a heuristic approach. Rather than characterizing economies of scale by a single number for "opening" a warehouse (*i.e.*, a fixed cost), they allow the economies of scale to affect warehousing costs over the entire range of warehouse sizes.

Mathematical Formulation

$$\text{Minimize} \quad \sum_{i,j} b_{ij} x_{ij} + \sum_i f_i(T_i) \quad ,$$

$$\text{subject to} \quad \sum_i x_{ij} = D_j \quad , \quad x_{ij} \geq 0 \quad \text{for } i = 1, 2, \ldots, n \quad ,$$

$$j = 1, 2, \ldots m \quad ,$$

where

x_{ij} = flow from warehouse i to demand center j,

$T_i = \sum_j x_{ij}$ = throughput of warehouse i,

b_{ij} = unit cost of flow from warehouse i to center j,

D_j = demand at center j,

$f_i(\cdot)$ = warehousing cost function for warehouse i, assumed continuous and concave over the range of interest.

The concavity property ensures that in the optimal solution, no demand center will receive flows from more than one warehouse.

Heuristic Approach

The basic concern of this paper is to extend the Kuehn-Hamburger results to the case in which the warehousing cost function is concave rather than of the form $a_i + b_i T_i$ (fixed cost plus linear operating cost). Feldman *et al.* point out that the basic difference between the linear and concave cases is in the assignment of customers to warehouses that have been opened. They also claim as an advantage the ability to deal with different concave warehousing cost functions for each potential warehouse.

Another important step forward from Kuehn-Hamburger is the use of both an add and a drop approach. Instead of starting with the best single location and merely building up one at a time until a local optimum set is found, Feldman *et al.* also start with all warehouses as an initial solution and drop one at a time until a local optimum is reached. The drop routine provides alternative solutions that often prove to be better than those obtained with the add routine. One reason for this is feasibility. In an add routine the program first tries to reach a feasible solution (*i.e.*, all demands satisfied), and this initial rapid drive to feasibility is not necessarily the route to an optimal solution.

To handle the non-linear warehousing costs, the authors start by associating a "local customer set" with each warehouse. This gives them an initial point on the cost curve by defining an initial throughput volume for each warehouse and they then use the incremental costs from this point in their iterations. In the case of the drop routine, they use the actual throughputs in the current solution.

Comparison with Kuehn-Hamburger

The first numerical tests were on the problems solved in the Kuehn-Hamburger paper. Feldman *et al.* claim that their algorithm yields solutions that have costs equal to or lower than those of Kuehn-Hamburger, although no dramatic savings were realized.

Other Numerical Tests

A much larger problem involving four factories, 49 warehouses, and 200 customer locations[1] also was run. Demand at each location was assumed to be proportional to population and transportation costs were assumed to be linear with distance. Each potential site was allowed to serve only the fifteen closest customers (a crude representation of the penalty for undue delay in shipment). The concave functions were approximated by line segments. The tests involved only cost functions consisting of two line segments or of a single straight line.

The drop routine was used as a way of obtaining an initial feasible solution for the add routine. Solutions were obtained for both types of cost functions by both the drop and add routines. The answers from the two routines differed considerably in the number of warehouses selected when the two-line-segment cost function was used. The drop solution yielded an answer involving 32 warehouses (and was 3 percent cheaper) whereas the add solution resulted in only 17 warehouses. An optimal solution found by integer programming resulted in an additional 0.5 percent saving and 34 warehouses. Feldman *et al.* conclude from their data that routine use of the programs would *not* lead to satisfactory results, so that the user must have insight for selecting starting solutions.

[1] Locations refer to cities in the U.S.

In the linear cost case, both the add and drop solutions led to nine warehouses with only one difference in location. The two answers differed by 0.3 percent, and the add solution was shown to be optimal by the integer programming routine. It is interesting to note that although the straight-line cost curves were chosen so as to approximate the two-segment cost curves, they resulted in a far different set of warehouses being selected and in 10 percent higher cost.

"Approximate Algorithm for the Fixed Charge Capacitated Site Location Problem"
Géza Jándy
Technical Report 67-3, Operations Research House, Stanford University, Stanford California (April 1967).

Jándy presents an approximate algorithm for the linear fixed-charge problem with transportation costs and with upper bounds (Problem VI, Section II-D) based on imputed unit-investment costs. He starts out by assigning an initial value of f_i/M_i to each investment cost. This initial value is just the increase in unit variable cost if the fixed-charge problem is solved as a linear program (see Section III-F)

Based on the optimal *LP* solution, Jándy modifies the imputed investment costs for plants not used to capacity and solves the problem again as a linear program. This process is continued until no more investment cost modifications can be made. At this point he checks for pairwise interchanges (bringing in an unused site to replace one in the current solution) that would lead to improvement in the objective function. Iterations are terminated when no more cost changes or variable interchanges can be made.

Four small sample problems solved by hand are presented.

"A Branch-Bound Algorithm for Plant Location"
M. A. Efroymson and T. L. Ray
Operations Research, Vol. 14, No. 3 (May-June 1966), pp. 361-368.

This is a companion paper to Feldman *et al.* discussed earlier. The original intent was to use mixed-integer programming techniques to verify the examples of Feldman *et al.* Actually, Efroymson and Ray came up with somewhat more.

The basic problem considered is the same as in Feldman, *et al.* namely, site selection from among a finite number of alternatives with known transportation and fixed costs and no restrictions on warehouse size. The basic integer programming technique is branch-and-bound (see, for example, [12] and [13]). Efroymson and Ray point out that branch-and-bound can be troublesome computationally because of the large number of linear programs that must be solved. To overcome this, they formulate the problem carefully and then introduce simplifications that make use of the structure of the plant location problem.

Formulation

Define the following quantities:

N_j = the set of indices of those plants that can supply customer j.

P_i = the set of indices of those customers that can be supplied from plant i

n_i = number of elements in P_i

x_{ij} = the fraction of demand D_j supplied from i

t_{ij} = unit transportation cost from plant i to customer j

$c_{ij} = t_{ij} x_{ij} D_j$ = total transportation cost from i to j

$f_i \geq 0$ = fixed cost associated with plant i .

The problem then becomes:

$$\text{Minimize } z = \sum c_{ij} x_{ij} + \sum f_i y_i \quad ,$$

$$\text{subject to } \sum_{i \in N_j} x_{ij} = 1 \quad ; \quad j = 1, \ldots n$$

$$0 \leq \sum_{j \in P_i} x_{ij} \leq n_i y_i \quad ; \quad i = 1, \ldots m$$

$$y_i = 0 \text{ or } 1 \quad ; \quad i = 1, \ldots m \quad .$$

The three constraints express the conditions that

(1) The demands of all customers are fully satisfied.

(2) A plant supplies 0 customers if it is closed ($y_i = 0$) or at most n_i if it is open.

(3) A plant (y) is either open ($y_i = 1$) or closed ($y_i = 0$).

Efroymson and Ray then assert that if, at any stage of branch-and bound,

K_0 = set of indices of y_i fixed at 0,

K_1 = set of indices of y_i fixed at 1,

K_2 = set of indices of the remaining y_i,

then the optimal solution of the linear programming problem is

$$x_{ij} = \begin{cases} 1 & \text{if } c_{ij} + g_i/n_i = \min_{k \in K_1 \cup K_2} [c_{kj} + g_k/n_k] \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \sum_{j \in P_i} x_{ij}/n_i \qquad (i \in K_2)$$

$$g_k = \begin{cases} f_k & (k \in K_2) \\ 0 & (k \in K_1) \end{cases}$$

This formulation has the advantage that it is quite simple to evaluate nodes on a computer, but has the disadvantage that, since the number of customers served by a plant is small compared to the potential number of customers that it can serve, the amount of fixed cost absorbed by a plant at any iteration will be small, hence the program will run long. Efroymson and Ray then turn to the structure of the problem to make three simplifications:

(1) Find the smallest net saving that could be made if a new plant is opened. If this smallest saving is positive it always pays to open this plant, so open it.

(2) If it is cheapest to ship to a given customer from a plant that is already open, ship to him from there.

(3) Find the largest net saving that could be made from opening a new plant. If this maximum is negative, leave this plant shut.

These three simplifications are discussed in quantitative terms in the annex following this discussion. The important points are that these are simple physical principles and that they cyclic. That is, the steps are followed over and over until no improvement is obtained. Then another iteration of branch-and bound is performed and these checks are run again.

Efroymson and Ray then talk about three extensions to more complicated plant cost structures. These are shown by the following plots:

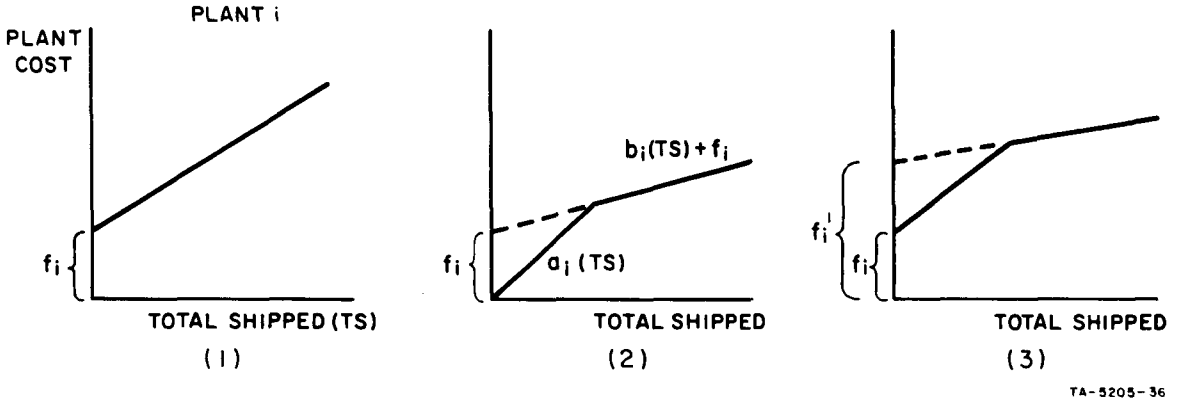


FIG. A-1 PLANT COST STRUCTURES

The first case is a simple modification of the basic case, where c_{ij} is replaced by $c_{ij} = (t_{ij} + \cdot_i)D_j$ where $\cdot_i$ is the unit plant cost for plant i.

The second case involves no fixed costs but does have concave plant costs. For simplicity, consider two linear segments. To solve the problem, consider two plants one small with linear cost a_i, and the other large with linear cost b_i and fixed cost f_i. The problem then reduces to one with restricted entry: namely, if the small plant is in the solution, the large one is not, and conversely: however, the plant can be closed (neither segment in the solution). The third case is a combination of the first two.

Efroymson and Ray conclude by discussing computational aspects. They indicate that they solved a 50-plant, 200-customer problem in 10 minutes on an IBM 7094 computer for cost structures (a) and (b) in Figure A-1. They also point out that if a "good" feasible solution is

known, the objective function value can be used as an initial bound.. Furthermore, they recommend terminating if an all-integer node is found that is within a preset ϵ of all smaller non-integer nodes.

Comments on Efroymson and Ray

The Efroymson-Ray algorithm is applicable only to the case in which there are no upper bounds on warehouse size. The central element in the algorithm is the fact that, in the absence of upper bounds, $x_{ij} = 0$ or 1; that is, a plant supplies either all or none of the requirements of a particular customer. This *all-or-none* relation leads to the three simplifications described and makes it possible to solve the linear programs by applying a test rather than the simplex method. No way has been found of adapting the algorithm to the upper-bounded case.[1]

Annex: Mathematical Formulation of Simplifications by Efroymson and Ray

1. Finding the minimum net gain from opening a new warehouse.

Let

$$\Delta_{ij} = \min_{k \in K_1 \cup K_2} [\max(c_{kj} - c_{ij}, 0)] \quad \text{for}$$

$$i \in K_2,\ i \neq k,\ j \in P_i,\ k \in N_j$$

where

Δ_{ij} is the smallest gain that can be obtained in shipping from i to j.

The conditions on i and j imply that the paths ij and kj must be feasible and that only those plants are considered which are open or could be opened. Δ_{ij} is greater than 0 only if plant i is the cheapest source for customer j.

A plant should be opened if

$$\sum_{j \in P_i} \Delta_{ij} - f_i > 0 \quad .$$

This condition states that if the sum of the smallest net gains from opening plant i exceeds the fixed cost of the plant, f_i, it should be opened since a profit is guaranteed.

[1] Private conversation with M. Efroymson, January 1967.

2. Reducing the Value of n_i

The number of customers that can be supplied from plant i is n_i. Since, in solving the successive linear programs in the branch-and-bound technique, the optimal value of y_i is divided by n_i, the process converges slowly. One way of speeding it up is to reduce this divisor.

Certainly, if we can find a plant that is already open (k) which provides cheaper shipping costs than a plant we are considering opening (i) we will want to ship from the open plant. Mathematically if

$$\min_{k \in K_1} (c_{kj} - c_{ij}) < 0 \qquad i \in K_2,\ j \in P_i$$

holds, eliminate the c_{ij} term from further consideration and reduce n_i by 1. Note that if this inequality holds for all customers that plant i can serve, it eliminates that plant from further consideration and sets $y_i = 0$.

3. Finding the Maximum Gain from Opening a Plant

Define Δ_{ij} as in Simplification 1, but consider only comparisons of a new plant with an existing (open) plant (that is, $k \in k_1 \cup N_j$). If

$$\sum_{j \in P_i} \Delta_{ij} - f_i < 0 \quad ,$$

then it is cheaper to use the existing warehouses than to open the new one because, even with the largest savings, the fixed cost cannot be recovered. If the condition is satisfied, then y_i is set to 0 and this plant need not be considered further.

"An Application of Heuristic Problem Solving to Accounts Receivable Management"
Ferdinand Levy
Management Science, Vol. 12, No. 6 (February 1966), pp. 245-254.

Levy is concerned with the lock box problem described in Section I-C.

Basic Assumptions

Levy makes the following basic assumptions:

(1) There are J possible lock box locations.

(2) The distribution of checks, both in terms of source, i, and dollar amount, A_i, is known. Levy suggests stratified sampling (based on geographical distribution and amount) to establish the values of the A_i.

(3) The following quantities are also known:

the mail time from the ith source to the jth lockbox, M_{ij};

the number of days from lock box to bank clearance, if combination ij is used, B_{ij};

the clearance cost of a check, c_j, by the bank in city j;

the fixed charge for a lock box, F_j; and

the interest rate, r.

Algorithm

Step 1. Compute the cost associated with each check if it is sent to the jth city:

$$Y_{ij} = r(A_i)(M_{ij} + B_{ij}) + C_j \quad .$$

Step 2. For each potential city, j, compute the total cost if *all* checks are sent there:

$$L_j = Y_{ij} + F_j \quad . \qquad \text{(A-1)}$$

Step 3. Open a lock box permanently in that city for which L_j is a minimum.

Step 4. Examine Y_{ij} for each check. If it is cheapest to send the check to the city just selected, assign that check there permanently and remove it from further consideration.

Step 5. Compute new values of L_j for the remaining checks and potential locations. Tentatively open the box with the lowest value of L_j.

Step 6. Now tentatively allocate all the remaining checks among the open boxes. If the total cost with the box found in Step 5 is smaller than the total cost without this additional box, open the new box permanently and go to Step 4. Otherwise, do not select this box and terminate.

Computational Experience

Levy cites results for an industrial manufacturer whose sales total $\$10^9$ who was able to reduce his accounts receivable float by 64 percent and to achieve a saving of $180,000 in interest costs.

Comments on Levy's Paper

Levy's algorithm is of the class of constructive hill climbing algorithms that successively select solutions by steepest ascent. There is no guarantee against stopping at a relative maximum. Levy in fact is satisfied with a single pass and makes no attempt to see whether or not interchanges or shifts could result in an improvement, nor does he try to find either an upper or a lower bound. As a detail, he does not include a check of the one-lock-box solution against the no-lock-box solution. An obvious crude bound on the minimum cost is available in the data Levy generates: for each check determine the minimum cost (over all lock boxes) and assign that cost to the check. The minimum total cost then is the sum of these minima plus the cost of the cheapest lock box (in symbols, find $[(\sum_i \min_j Y_{ij}) + \min_j F_j]$).

"An Approximate Solution Method for the Fixed Charge Problem"
L. Cooper and C. Drebes
Naval Research Logistics Quarterly, Vol. 14, No. 1 (March 1967) pp. 101-14.

This recent paper presents two heuristic methods for solving the linear fixed-charge problem with a fixed charge associated with each variable. The fundamental idea is to start with the simplex solution and then bring adjacent extreme points into the basis one at a time, where the extreme points are selected on the basis of their fixed costs. Computational results are presented for problems having constraint matrices, **A**, up to 15×30. Average computing times reported on the IBM 7072 were 20 seconds for 5×10 problems, 1.6 minutes for 7×15, and 15 minutes for the 15×30 problems. Comparison with the

exact solution obtained by complete enumeration of extreme points showed the algorithms to give answers within 5 percent or less for the two smaller problems and within 10 percent for the largest problems.

"Use of Completely Reduced Matrices in Solving Transportation Problems with Fixed Charges"
P. S. Dwyer
Naval Research Logistics Quarterly, Vol. 13, No. 3 (September 1966) pp. 289-315.

Dwyer applies his method of completely reduced matrices [23] to the fixed-charge transportation problem. His main concern is with the case of equal fixed charges where the optimal solution may be a degenerate solution ($< m + n - 1$ routes open). In this case, if fixed charges are large compared to variable costs, the most degenerate solution is optimal. Dwyer presents exact and approximate methods for finding this most degenerate solution. He also discusses approximate solutions for unequal fixed charges based on his reduced matrix approach.

APPENDIX B

THE BENDERS PARTITIONING ALGORITHM

APPENDIX B

THE BENDERS PARTITIONING ALGORITHM

The algorithm presented in this thesis makes heavy use of partitioning the constraint set into integer and non-integer variables. In this respect it is similar to the approach used by Benders [8]. This Appendix summarizes the Benders approach and compares it with the present work. The discussion of the algorithm (and the notation used) follows that given by Balinski in his summary article on integer programming [20].

Consider the mixed integer programming problem:

$$\begin{aligned} &\text{Minimize} && y_0 = C_1X + C_2Y \quad , \\ &\text{Subject to} && A_1X + A_2Y \geq B \quad , \\ & && X, Y \geq 0 \quad , \\ & && Y \text{ integer} \quad . \end{aligned} \tag{B-1}$$

Let R denote the permissible values of Y. Then Eq. (B-1) may be written as

$$\min_{Y \in R} \{C_2Y + \min_{X} [C_1X | A_1X \geq B - A_2Y \quad , \quad X \geq 0]\} \quad . \tag{B-2}$$

Given the vector Y, the minimization over X is a linear program. This linear program can be replaced by its dual,

$$\max_{U} \{U(B - A_2Y) | UA_1 \leq C_1 \quad , \quad U \geq 0\}$$

to obtain

$$\min_{Y \in R} \{C_2Y + \max_{U} [U(B - A_2Y) | UA_1 \leq C_1 \ , \ U \geq 0]\} \quad . \tag{B-3}$$

Consider the convex polyhedral set, $S = \{U | UA_1 \leq C_1 , U \geq 0\}$, which is independent of Y. If the set is empty, no solution to the original problem exists (Farkas' Theorem). Otherwise, the maximum value of $U(B - A_2Y)$ attains its optimum at an extreme point of S or grows without bound along an extreme ray of S. But both the extreme points (vertices) and extreme rays of S are finite in number and, hence, can be enumerated. Let

$$L = \{U^l | U^l = \text{extreme point of } S\} ,$$

$$K = \{U^k | UA_1 \geq 0 , U \geq 0\} = \text{set of extreme rays of } S .$$

If for some Y there exists a U^k, $k \in K$, such that $U^k(B - A_2Y) > 0$, then the maximization in Eq. (B-3) leads to a value of $+\infty$, which implies that there is no solution to the minimization over X. Hence, a necessary and sufficient condition on Y to admit a feasible X is that

$$U^k(B - A_2Y) \leq 0 \quad \text{for all } k \in K . \tag{B-4}$$

We can use Eq. (B-4) to rewrite Eq. (B-3) as:

$$\min_{Y \in R} \{y_0 | y_0 \geq C_2Y + \max_{l \in L} [U^l(B - A_2Y) , \text{ and } U^k(B - A_2Y) \leq 0 \quad \text{all } k \in K]\} . \tag{B-5}$$

Thus, the partitioning transforms the mixed-integer program into an all-integer program in variables Y containing potentially vast numbers of linear constraints on y_0 and Y. As in decomposition for linear programming, the hope is that only a small subset of these constraints need to be enumerated.

Computationally, one iteration of the Benders algorithm proceeds as follows:

Step 1. Given a finite subset of U^j, $J \in Q$, where $Q \subseteq K \cup L$, solve the all-integer program defined by Eq. (B-5) over the set of constraints associated with U^j, for $j \in Q$.

Step 2. If no feasible solution exists, then Eq. (B-1) has no feasible solution either. Terminate.

Step 3. Let y_0^*, Y be the optimal solution obtained (or y_0^* small, Y^* feasible if y_0 is unbounded below) for Eq. (B-5). Determine whether or not this solution is optimal for Eq. (B-1) by solving the linear program:

$$\min_X \{C_1X \mid A_1X \geq B - A_2Y^*, X \geq 0\}$$

$$= \max_U \{U(B - A_2Y^*) \mid UA_1 \leq C_1, U \geq 0\} .$$

Let X^*, U^* be the optimal solutions of the primal and dual linear programs, respectively, and let $f(Y^*)$ be the value of the objective function.

Step 4. If $f(Y^*) = +\infty$, then the current Y^* does not admit a feasible X and a new U^k, $k = K$, has been found from the dual. If $f(Y^*) = -\infty$, then no feasible solution exists. Terminate. If $f(Y^*)$ is finite, then either the solution is optimal or an additional element of Q has been found. In either case, (X^*, Y^*) is a feasible solution of Eq. (B-1). If in addition,

$$y_0^* \geq C_2Y^* + f(Y^*) = C_2Y^* + C_1Y^* \qquad \text{(B-6)}$$

then (X^*, Y^*) is optimal. Terminate.

If Eq. (B-6) is not satisfied, then, one or more of the linear constraints of Eq. (B-5) is not satisfied by y_0^*, Y^*. The most "violated" one corresponds to U^*. Hence, U^* is adjoined to the set Q and the process repeats, starting from Step 1.

The algorithm has the following properties:

- It must terminate in a finite number of steps.
- At each set an upper and a lower bound on y_0 is obtained. (However, the upper bound may be infinite if $f(Y^*)$ is infinite.)
- If the set S is bounded, then $f(Y^*)$ is always finite and each iteration yields a feasible pair (X^*, Y^*).
- The partitioning preserves the structure of the matrix A_1. Thus, if a special linear programming technique (such as the transportation problem, upper bounded variables, or network flow) is applicable to A_1 it can be used.

Two main differences between the Benders algorithm and the decomposition algorithm presented here are:

(1) The Benders algorithm requires the solution of a series of integer programs whereas the present algorithm involves only one integer program.

(2) The Benders algorithm, being a general algorithm, does not make use of the specific structure of the problem, whereas the present algorithm makes explicit use of problem structure, particularly the relations of the integer and continuous variables.

Some computational experience with the Benders algorithm has been obtained for fixed charge problems by Bruce R. Buzby of Union Carbide on what he calls "non-linear distribution" problems.[1] His problems have been concerned primarily with adding facilities to existing networks, and with non-linear objective functions. He concludes, based on 135 randomly generated sample problems, that at present the Benders algorithm is efficient for situations in which the integer variables result in at most 10^6 combinations having to be searched.

[1] Private communication, March 1967.

APPENDIX C

MURTY'S SOLUTION OF THE FIXED-CHARGE PROBLEM BY RANKING THE EXTREME POINTS

APPENDIX C

MURTY'S SOLUTION OF THE FIXED-CHARGE PROBLEM BY RANKING THE EXTREME POINTS

This appendix summarizes Murty's method [7] for solving the fixed-charge problem by ranking the extreme points. Murty defines the subproblem as we do, that is,

$$\left.\begin{array}{lll} \text{Minimize} & \mathbf{c}\,\mathbf{x} & , \\ \text{subject to} & \mathbf{A}\mathbf{x} \geq \mathbf{b} & \\ & \mathbf{x} \geq \mathbf{0} & \end{array}\right\} . \qquad \text{(C-1)}$$

He uses the result of Hirsch and Dantzig [2], that the minimum of the fixed-charge problem will occur at an extreme point of the constraint set (C-1). His only assumption is that the solution of the subproblem is finite. In this case it is possible to rank all the extreme points of (C-1) in increasing order of $\mathbf{c}\,\mathbf{x}$.

Suppose that such a ranking has been achieved. Then it is possible to bound the maximum value of the variable costs.

Z_k = variable cost of the kth ranked extreme point;

D_k = fixed cost of the kth ranked extreme point;

D_0 = a lower bound on the fixed charge component (that is, $D_0 \leq D_k$ for all k);

$$\delta_r = \min_{k=1,\ldots,r} \{Z_k - Z_1 + D_k - D_0\};$$

$$\Delta_r = Z_r - Z_1 \quad .$$

The condition for optimality is

$$\delta_r \geq \Delta_r \quad .$$

This condition states that

$$Z_r - Z_1 \geq Z_k + D_k - Z_1 - D_0 \quad .$$

The first two quantities on the right are the best solution found thus far, and the second two are a lower bound on the total cost. Rearranging terms and cancelling Z_1 yields:

$$Z_r + D_0 \geq Z_k + D_k \quad , \quad k = 1,2, 1.., r = 1 \quad .$$

Since D_0 is a lower bound on the fixed cost, and since all subsequent ranked extreme points Z_{r+1}, ..., Z_n are larger, the current best solution must be optimal.

Murty presents an algorithm for finding the adjacent extreme points which involves only one-step pivot operations. The algorithm becomes somewhat complicated in degenerate cases where several basic feasible solutions represent the same vertex In this case, all these basic solutions must be explored by branching from the degenerate vertex.

Murty points out that his algorithm works well when the extreme points of the subproblem are non-degenerate and "the range of values of Z for feasible x is large compared to the fixed charges." The latter statement really implies that the values of Z are large compared to the fixed charges and that there are few extreme points which are close (in value of the objective function) to the optimal extreme point. The efficiency of the algorithm also improves with the nearness of D_0 to the greatest lower bound of D_k, the fixed charges associated with the vertices.

APPENDIX D

DETERMINING DUAL VARIABLES IN THE FIXED CHARGE PROBLEM WITH TRANSPORTATION COSTS

APPENDIX D

DETERMINING DUAL VARIABLES IN THE FIXED CHARGE PROBLEM WITH TRANSPORTATION COSTS

To use the Hillier algorithm for fixed-charge problems with transportation costs (Section III-F), it was proposed to declare all variables to be Group 3 and to use the Group 3 procedures. The key to this approach is to be able to solve the series of maximization and minimization problems required for initialization in Group 3 efficiently. To this end, the initial solution is not added to the constraint set, and the problems are claimed to reduce to a series of transportation problems. The purpose of this Appendix is to exhibit these transportation problems and to prove the assertion true.

1. Minimization

The series of minimization problems to be solved for $k = 1, 2, \ldots, m$ are

Minimize y_K

				Dual Variable
subject to	$\sum_{i=1}^{m} x_{ij} \geq D_j$;	$j = 1, \ldots n$	Problem D-1	π_j
	$-\sum_{j=1}^{n} x_{ij} \geq -M_i \bar{y}_i$;	$i = 1, 2, \ldots k-1$		ρ_i
	$M_k y_k - \sum_{j=1}^{n} x_{ij} \geq 0$;			ρ_k
	$M_i y_i - \sum_{j=1}^{n} x_{ij} \geq 0$;	$i = k+1, \ldots m$		ρ_i
	$0 \geq -1 + \bar{y}_i$;	$i = 1, \ldots, k-1$		σ_i
	$-y_i \geq -1$;	$i = k, \ldots, m$		σ_i
	$x_{ij}, y_i \geq 0$;			

where $\bar{y}_i$ is the given value of the ith 0 - 1 variable in the initial feasible solution and the other symbols are as defined previously throughout this dissertation.

The dual variables corresponding to each of the contraints are listed beside the constraints. In terms of this notation, the dual problem is

$$\left.\begin{array}{lll}
\text{Maximize} & \sum_{j=1}^{n} D_j \pi_j - \sum_{i=1}^{k-1} M_i \bar{y}_i \rho_i + \sum_{i=1}^{k-1} (\bar{y}_i - 1)\sigma_i - \sum_{i=k}^{m} \sigma_i & \\
\text{subject to} & \pi_j - \rho_i \leq 0 \quad ; \quad i = 1, \ldots m;\ j = 1, \ldots n & \\
& M_k \rho_k - \sigma_k \leq 1 \quad ; & \\
& M_i \rho_i - \sigma_i \leq 0 \quad ; \quad i = k+1, \ldots n & \\
& \pi_j, \rho_i, \sigma_i \geq 0 \quad ; \quad \text{for all } i, j &
\end{array}\right\} \text{Problem D-II}$$

Examination of Problem D-I shows that, at optimality,

$$y_k = \sum_{j=1}^{n} x_{kj} / M_k \quad ,$$

since the objective is to minimize y_k so that the constraint,

$$M_k y_k - \sum_{j=1}^{n} x_{kj} \geq 0 \quad ,$$

will be satisfied with equality. If we make this substitution for y_k and substitute 1 for all y_i for which $i \geq k$, we have the following problem:

$$
\left.
\begin{aligned}
&\text{Minimize} \quad \frac{1}{M_k} \sum_{j=1}^{n} x_{kj} \\
&\text{subject to} \quad \sum_{i=1}^{m} x_{ij} \geq D_j \qquad j = 1, 2, \ldots n \\
&\qquad - \sum_{j=1}^{m} x_{ij} \geq - M_i \bar{y}_i \qquad i = 1, \ldots k-1 \\
&\qquad - \sum_{j} x_{kj} \geq - M_i \qquad i = k, k+1, \ldots M \\
&\qquad x_{ij} \geq 0 \qquad \text{for all } i, j
\end{aligned}
\right\} \text{Problem D-III}
$$

Here $\bar{y}_i = 0$ or 1 and are given for each $i < k$.

The dual of problem D-III is:

$$
\left.
\begin{aligned}
&\text{Maximize} \quad \sum_{j=1}^{n} D_j \pi'_j - \sum_{i=1}^{k-1} M_i \bar{y}_i \rho'_i - \sum_{i=1}^{n} M_i \rho'_i \\
&\text{subject to } \pi'_j - \rho'_i \leq 0 \qquad j = 1, \ldots n;\ i = 1, \ldots k-1, k+1, \ldots n \\
&\qquad \pi'_j - \rho'_k \leq \frac{1}{M_k} \qquad j = 1, \ldots n \\
&\qquad \pi'_i, \rho'_j \geq 0 \qquad \text{all } i, j
\end{aligned}
\right\} \text{Problem D-IV}
$$

where the primes indicate that the dual variables in Problem D-IV may have different values than those in Problem D-II. In fact, according to the following Theorem, the two problems are related:

Theorem. Problems D-I and D-III are equivalent problems having the same value of the objective function. The dual variables for these problems satisfy the following relations:

$$
\begin{aligned}
\pi_j &= \pi'_j \qquad ; \quad j = 1, \ldots n \\
\rho_i &= \rho'_i \qquad ; \quad i \neq k \\
\rho_k &= \rho'_k + \frac{1}{M_k} \qquad ;
\end{aligned}
$$

$$\sigma_i = M_i \rho_i' \quad ; \quad i \geq k$$

$$\sigma_i = 0 \quad ; \quad i < k \quad .$$

Proof. It was pointed out above that for Problem D-I,

$$y_k = \frac{1}{M_k} \sum_{j=1}^{n} x_{kj}$$

at optimality. The right-hand side is also the form of the objective function for Problem D-III. It remains to be shown that the objective functions are equal. Examination of Problems D-I and D-III shows that the constraints are identical for all $i < k$. For $i \geq k$, since $x_{ij} \geq 0$ and since y can range from 0 to 1, the constraints

$$M_i y_i - \sum_{j=1}^{n} x_{ij} \geq 0 \quad ,$$

in Problem D-I can be written as

$$0 \leq \sum_{j=1}^{n} x_{ij} \leq M_i y_i \leq M_i \quad .$$

But this is precisely the constraint in Problem D-III. Hence, since the constraints in the two problems are equivalent, the problems must have equal minima.

By the duality theory of linear programming, the objective functions of Problem D-I and its dual, Problem D-II, are equal at optimality. Similarly, the objective functions of Problems D-II and D-IV are equal at optimality. Since the objective functions of Problems D-I and D-III are also equal at optimality, we conclude that the objective functions of Problems D-II and D-IV must be equal at optimality. That is,

$$y_k = \sum_{j=1}^{n} D_j \pi_j - \sum_{i=1}^{k-1} M_i \bar{y}_i \rho_i + \sum_{i=1}^{k-1} (\bar{y}_i - 1)\sigma_i - \sum_{i=k}^{M} \sigma_i \quad ,$$

$$\frac{1}{M_k} \sum_{j=1}^{n} x_{kj} = \sum_{j=1}^{n} D_j \pi_j' - \sum_{i=1}^{k-1} M_i \bar{y}_i + \sum_{i=1}^{n} M_i \rho_i' \quad .$$

Suppose that we have solved Problem D-IV. If we set

$$\pi_j = \pi'_j \quad ;$$

$$\rho_i = \rho'_i \quad \text{for} \quad i \neq k$$

$$\rho_k = \rho'_k + \frac{1}{M_k} \quad ;$$

$$\sigma_i = M_i \rho'_i \quad \text{for} \quad i \geq k \quad ;$$

and

$$\sigma_i = 0 \quad \text{for} \quad i < k \quad ,$$

then the objective functions of Problems D-IV and D-II are equal. We must still show that the constraints of Problem D-II are satisfied.

Since

$$\pi'_j - \rho'_i \leq 0 \quad \text{for} \quad i \neq k, \quad \text{all } j \quad ,$$

substituting π_j and ρ_i yields

$$\pi_j - \rho_i \leq 0 \quad \text{for} \quad i \neq k, \quad \text{all } j \quad .$$

Since

$$\pi'_j - \rho'_k \leq \frac{1}{M_k} \quad \text{for all } j \quad ,$$

substituting π_j and $\rho_k = (1/M_k)$ for π'_j and ρ'_k yields

$$\pi_j - (\rho_k - \frac{1}{M_k}) \leq \frac{1}{M_k}$$

or

$$\pi_j - \rho_k \leq 0 \quad .$$

Substituting

$$\rho_k = \rho'_k + \frac{1}{M_k} \quad \text{and} \quad \sigma_k = M_k \rho'_k$$

implies

$$M_k \rho_k - \sigma_k = M_k \rho'_k + M_k \rho'_k = 1.$$

Hence

$$M_k \rho_k - \sigma_k \leq 1$$

is satisfied. Finally, substituting $\sigma_i = M_i \rho_i' = M_i \rho_i$ for $i > k$ shows that $M_i \rho_i = \sigma_i \leq 0$ is satisfied, which completes the proof.

2. Maximization

For maximization an entirely analogous argument holds. For convenience, the maximization of y_k will be written in its equivalent form as the minimization of $(-y_k)$. The primal problem is

Minimize $(-y_k)$

subject to

$$\left.\begin{aligned}
&\sum_{i=1}^{m} x_{ij} \geq D_j \quad , \quad j = 1, \ldots n \ ; \\
&-\sum_{j=1}^{n} x_{ij} \geq -M_i \bar{y}_i \quad , \quad i = 1, \ldots k-1 \ ; \\
&M_k y_k - \sum_{j=1}^{n} x_{kj} \geq 0 \quad , \\
&M_i y_i - \sum_{j=1}^{n} x_{ij} \geq 0 \quad , \quad i = k+1, \ldots n \ ; \\
&-y_i \geq -1 \quad , \quad i = k+1, \ldots n \ ; \\
&x_{ij}, y_i \geq 0 \quad .
\end{aligned}\right\} \text{Problem D-V}$$

The dual of Problem D-V is

$$\left.\begin{aligned}
&\text{Maximize} \ \sum_{j=1}^{n} D_j \pi_j - \sum_{i=1}^{k-1} M_i \bar{y}_i \rho_i - \sum_{i=k+1}^{n} \sigma_i \quad , \\
&\text{subject to} \ \pi_j - \rho_i \leq 0 \quad , \quad i = 1, \ldots m, \quad j = 1, \ldots n \ ; \\
&M_k \rho_k - \sigma_k \leq -1 \ , \\
&M_i \rho_i - \sigma_i \leq 0 \quad , \quad i = k+1, \ldots m \ ; \\
&\pi_j, \sigma_i \geq 0 \quad .
\end{aligned}\right\} \text{Problem D-VI}$$

As in minimization, at optimality,

$$M_k y_k = \sum_{j=1}^{n} x_{kj} \quad ,$$

and the following equivalent problem and its dual can be written:

Problem D-VII

$$\text{Minimize} \quad -\frac{1}{M_k} \sum_{j=1}^{n} x_{kj} \quad ,$$

$$\text{subject to} \quad \sum_{i=1}^{m} x_{ij} \geq D_j \quad , \quad j = 1, \ldots n \quad ;$$

$$-\sum_{j=1}^{n} x_{ij} \geq -M_i \bar{y}_i \quad , \quad i = 1, \ldots k-1 \quad ;$$

$$-\sum_{j=1}^{n} x_{ij} \geq -M_i \quad , \quad i = k, k+1, \ldots n$$

$$x_{ij} \geq 0 \quad .$$

Problem D-VIII

$$\text{Maximize} \quad \sum_{j=1}^{n} D_j \pi'_j - \sum_{i=1}^{k-1} M_i \bar{y}_i \rho'_i - \sum_{i=k}^{n} M_i \rho'_i$$

$$\text{subject to}$$

$$\pi'_j - \rho'_i \geq 0 \quad , \quad i = 1, \ldots k-1, k+1, \ldots n \quad ;$$

$$j = 1, \ldots n \quad ;$$

$$\pi'_j - \rho'_k \geq -\frac{1}{M_k} \quad , \quad j = 1, \ldots n \quad ;$$

$$\pi'_j, \rho'_i \geq 0 \quad .$$

The following Theorem shows the relation between Problems D-V and D-VII.

Theorem. Problems D-V and D-VII are equivalent problems having the same value of the objective function. The dual variables for these problems satisfy the following relations:

$$\pi_j = \pi'_j \quad , \quad j = 1, \ldots n \quad ;$$

$$\rho_i = \rho'_i \quad , \quad i \neq k \quad ;$$

$$\rho_k = \rho'_k - \frac{1}{M_k} \quad ,$$

$$\sigma_i = M_i \rho'_i \quad , \quad i \geq k \quad ;$$

$$\sigma_i = 0 \quad , \quad i < k \quad .$$

Proof. The proof of this theorem is identical to that for the previous theorem, except for showing the relations between primed and unprimed dual variables.

Assume that Problem D-VIII has been solved. We first show that

$$\rho'_k - \frac{1}{M_k} \geq 0 \quad .$$

Assume that $\rho'_k - (1/M_k) < 0$. Then, since $\pi'_j \leq \rho'_k - (1/M_k)$ is satisfied, the assumption implies that $\pi'_j < 0$. This is a contradiction, because $\pi'_j \geq 0$. Hence $\rho_k \geq 0$.

Since $\pi'_j - \rho'_i \leq 0$ for $i \neq k$ and all j, substituting π_j and ρ_i yields $\pi_j - \rho_i \leq 0$ for $i \neq k$ and all j. For $i = k$, $\pi'_j - \rho'_k \leq -(1/M_k)$. Substituting for π'_j and ρ'_k yields $\pi_j - [\rho_k + (1/M_k)] \leq -(1/M_k)$ or $\pi_j - \rho_k \leq 0$.

Substituting for ρ_k and σ_k yields

$$M_k \rho_k - \sigma_k = M_k \left(\rho'_k - \frac{1}{M_k} \right) - M_k \rho'_k = -1$$

and substituting for ρ_i and σ_i ($i > k$) yields $M_i \rho_i - M_i \rho_i = 0$. Hence, all constraints involving σ_i are satisfied, which completes the proof.

REFERENCES

1. M. A. Efroymson and T. L. Ray, "A Branch-Bound Algorithm for Plant Location," *Operations Research*, Vol. 14, No. 3, pp. 361-368 (May-June 1966).

2. W. M. Hirsch and G. B. Dantzig, "The Fixed Charge Problem" Rand Corporation Memo P-648, The Rand Corporation, Santa Monica, California (1 December 1954).

3. A. A. Kuehn and M. J. Hamburger, "A Heuristic Program for Locating Warehouses," *Management Science*, Vol. 9, No. 4, pp. 643-666 (July 1963).

4. A. S. Manne, "Plant Location Under Economies of Scale-Decentralization and Computation," *Management Science*, Vol. 11, No. 2, pp. 213-235 (November 1964).

5. E. Feldman, F. A. Lehrer, and T. L. Ray, "Warehouse Location under Continuous Economies of Scale," *Management Science*, Vol. 12, No. 9, pp. 670-684 (May 1966).

6. M. L. Balinski, "Fixed Cost Transportation Problems," *Naval Research Logistics Quarterly*, Vol. 8, No. 1, pp. 41-54 (March 1961).

7. K. G. Murty, "Solving the Fixed Charge Problem by Ranking the Extreme Points" Operations Research Center Report 66-7, University of California-Berkeley Operations Research Center (March 1966).

8. J. F. Benders, "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematik*, Vol. 4, No. 3, pp. 238-252 (October 1962).

9. R. D. Smallwood, "Minimas Detection Station Placement," *Operations Research*, Vol. 13, No. 4, pp. 632-646 (July-August 1965).

10. G. B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, New Jersey, 1963).

11. F. S. Hillier, "An Optimal Bound-and-Scan Algorithm for Integer Linear Programming," Department of Industrial Engineering, Technical Report No. 3, Stanford University, Stanford, California (August 19, 1966).

12. N. Agin, "Optimal Seeking with Branch and Bound," *Management Science*, Vol. 13, No. 4, pp. B-176-B-185 (December 1966).

13. E. L. Lawler and D. E. Wood, "Branch-and-Bound Methods: A Survey," *Operations Research*, Vol. 14, No. 4 pp. 699-719 (July-August 1966).

14. S. Karlin, *Mathematical Methods and Theory in Games, Programming, and Economics* (Addison-Wesley Publishing Company, Cambridge, Massachusetts, 1959).

15. H. Everett, III, "Generalized LaGrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research*, Vol. 11, No. 3, pp. 399-417 (May-June 1963).

16. S. Glicksman, L. Johnson, and L. Eselson, "Coding the Transportation Problem," *Naval Research Logistics Quarterly*, Vol. 7, No. 2, pp. 169-183 (June 1960).

17. E. Balas, "Duality in Discrete Programming," Operations Research House, Technical Report 67-5, Stanford University, Stanford, California (August 1967).

18. R. E. Gomory and W. J. Baumol, "Integer Programming and Pricing," *Econometrica*, Vol. 28, No. 3, pp. 521-550 (July 1960).

19. R. E. Alcaly and A. K. Klevorick, "A Note on the Dual Prices of Integer Programs," *Econometrica*, Vol. 34, No. 1 (January 1966).

20. M. L. Balinski, "Integer Programming: Methods, Uses, Computation," *Management Science*, Vol. 12, No. 3, pp. 253-314 (November 1965).

21. E. M. L. Beale, "Survey of Integer Programming," *Operational Research Quarterly*, Vol. 16, No. 3 (June 1965) pp. 219-228.

22. F. S. Hillier and M. M. Connors, "Quadratic Assignment Problem Algorithms and the Location of Indivisible Facilities," *Management Science*, Vol. 13, No. 1 (September 1966) pp. 42-57.

23. P. S. Dwyer, "The Direct Solution of the Transportation Problem with Reduced Matrices," *Management Science*, Vol. 13, No. 1, pp. 77-98 (September 1966).

24. H. W. Kahn and W. J. Baumol, "An Approximative Algorithm for the Fixed-Charges Transportation Problem," *Naval Research Logistics Quarterly*, Vol. 9, No. 1, pp. 1-16 (March 1962).

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (*Corporate author*)
Stanford University
Department of Operations Research and Department of Statistics
Stanford, California

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
MIXED INTEGER PROGRAMMING ALGORITHMS FOR SITE SELECTION AND OTHER FIXED CHARGE PROBLEMS HAVING CAPACITY CONSTRAINTS

4. DESCRIPTIVE NOTES (*Type of report and inclusive dates*)
Technical Report

5. AUTHOR(S) (*First name, middle initial, last name*)
Paul Gray

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 30, 1967	134	24

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Nonr-225(53)	Technical Report No. 101
b. PROJECT NO. NR-042-002	
c.	9b. OTHER REPORT NO(S) (*Any other numbers that may be assigned*) Nonr-225(89) Technical Report No. 6
d.	DA-49-092-ARO-10 Technical Report 5025-36
	NSF Grant GP 7074 Technical Report 1

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
Logistics and Mathematical Statistics Branch
Office of Naval Research
Washington, D.C. 20360

13. ABSTRACT

Algorithms for solving site-selection and similar fixed charge problems with upper bound constraints are presented. The basic approach is to formulate the problem as a mixed integer program and to solve these programs by decomposing them into a master integer program and a series of subproblems which are linear programs. To reduce the number of vertices to be examined to manageable proportions, the bound-and-scan algorithm by F. S. Hillier was adapted to the fixed charge problem. Algorithms are presented for four classes of problems:

1. Fixed charge problem with linear variable costs and a fixed charge for each variable at non-zero level.
2. Problem 1 with separable concave or convex variable costs.
3. A warehouse location problem in which variable costs and constraints are of the transportation type. A fixed charge is associated with each warehouse opened.
4. The fixed charge transportation problem in which a fixed charge is associated with each route rather than with each warehouse.

Computational results for Problems 1, 2, and 4 are presented.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Fixed Charge Problem						
Location Problems						
Mixed Integer Programming						