AD-A033 104 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 9/2
DECOMPOSITION IN INTEGER PROGRAMMING.(U)
SEP 76 G A KOCHMAN F44620-74-C-0079
UNCLASSIFIED SOL-76-21 AFOSR-TR-76-1213 NL

1 OF 2
AD A033104

2

B

DECOMPOSITION IN INTEGER PROGRAMMING

BY

GARY A. KOCHMAN

TECHNICAL REPORT SOL 76-21
SEPTEMBER 1976

Systems Optimization Laboratory

Department of Operations Research

Stanford University

Stanford California 94305

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is
approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR-76-1213
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): DECOMPOSITION IN INTEGER PROGRAMMING
5. TYPE OF REPORT & PERIOD COVERED: Interim
6. PERFORMING ORG. REPORT NUMBER: 76-21
7. AUTHOR(s): Gary A. Kochman
8. CONTRACT OR GRANT NUMBER(s): F44620-74-C-0079, N00014-76-C-0418
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Stanford University, Department of Operations Research, Stanford, CA 94305
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A6
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332
12. REPORT DATE: September 1976
13. NUMBER OF PAGES: 156
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): SOR-76-21
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited.
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
18. SUPPLEMENTARY NOTES
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Block angular structure
Branch-and-bound
decomposition
integer programming
linking constraints
subproblems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Linear programming models in which the constraint matrix has a block angular structure arise frequently in many applications. While much work has been devoted to exploiting this special structure when the problem variables are assumed to be continuous, little consideration has been given to models of this type in which the variables are required to take on only integer values. In this report, an algorithm for the decomposition of block angular integer programs is presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED 408 765
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract

The block angular integer program consists of several subproblems which would operate independently except that they are tied together by a set of linking constraints. Conceptually, these linking constraints are viewed as representing common resources which the subproblems must share. The problem thus becomes that of determining an optimal allocation of these resources among the subproblems.

Toward this end, a branch-and-bound search routine is developed. It is shown how the LP-optimal dual multipliers and any slacks which appear in the optimal integer solutions to the subproblems can be used to guide the search, as well as for bounding and fathoming purposes. Special structures which arise when there is only a single linking constraint are discussed in detail.

Since the problem decomposes completely once an allocation of the linking resources is specified, only the subproblems ever need be solved explicitly. Computational results obtained with the decomposition algorithm are reported.

DECOMPOSITION IN INTEGER PROGRAMMING

By

Gary A. Kochman

TECHNICAL REPORT SOL 76-21

September 1976

SYSTEMS OPTIMIZATION LABORATORY

DEPARTMENT OF OPERATIONS RESEARCH

Stanford University
Stanford, California

Research and reproduction of this report were partially supported by the Office of Naval Research Contract N00014-76-C-0418; U.S. Army Research Office - Durham Contract DAAC29-74-C-0034; and U.S. Air Force Office of Scientific Research F44620-74-C-0079.

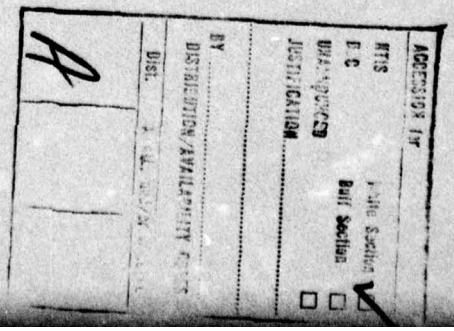

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

In this report we consider the integer programming problem (P) as defined by (1)-(5):

$$\text{(P)} \qquad \text{maximize} \quad c^{(1)}x^{(1)} + c^{(2)}x^{(2)} + \cdots + c^{(N)}x^{(N)}, \tag{1}$$

subject to

$$A^{(1)}x^{(1)} + A^{(2)}x^{(2)} + \cdots + A^{(N)}x^{(N)} \leq b \tag{2}$$

$$\left.\begin{aligned} B^{(1)}x^{(1)} &\leq \beta^{(1)} \\ B^{(2)}x^{(2)} &\leq \beta^{(2)} \\ &\ddots \\ B^{(N)}x^{(N)} &\leq \beta^{(N)} \end{aligned}\right\} \tag{3}$$

$$x^{(j)} \geq 0, \qquad \text{for } j = 1, 2, \ldots, N \tag{4}$$

$$x^{(j)} \text{ integer, for } j = 1, 2, \ldots, N, \tag{5}$$

where b is an m-vector and, for each $j = 1, 2, \ldots, N$, $c^{(j)}$ and $x^{(j)}$ are n_j-vectors, $\beta^{(j)}$ is an m_j-vector and, correspondingly, $A^{(j)}$ is an $m \times n_j$ matrix and $B^{(j)}$ is an $m_j \times n_j$ matrix. Because of its special structure, it is natural to consider problem (P) as consisting of N almost independent subproblems (1), (3), which could be solved separately except that they are tied together by the linking constraints (2). Mathematical programming problems in which the constraint matrix takes the form above have commonly been referred to as "block angular."

1.2 Integer Programming: State of the Art

The field of integer programming has progressed greatly since its inception in 1958 when Gomory first proposed a method for generating integer solutions to linear programs. In the intervening period of less than 20 years, numerous general-purpose integer programming algorithms have been developed, and several commerical computer codes have been marketed. The result is that the solution of integer programming problems which arise in the course of industrial and military decision making processes now has become a relatively routine occurrence.

The many integer programming algorithms which have been proposed seem naturally divided on the basis of their underlying theoretical foundations into three major categories: cutting plane, group theoretic, and enumerative. Cutting plane algorithms (see Gomory (1963), and Martin (1963)) proceed by first solving the linear programming (LP) relaxation of the integer programming problem. If the optimal LP solution satisfies the (dropped) integrality restrictions, this solution must be optimal and the procedure terminates. Otherwise, a new constraint ("cutting plane") is generated and added to the problem. This constraint is generated so as to render the previous LP-optimal solution infeasible, without eliminating any feasible integer solutions to the problem. The revised LP problem is reoptimized and the new solution again checked for integrality. By properly generating the additional constraints at each iteration, it is possible to ensure that the procedure will converge to the integer optimum in a finite number of steps. The major drawback inherent in cutting plane algorithms, however, is that no feasible integer solution is produced until the last step, when the optimum is obtained. As the

number of iterations required can be quite large, this aspect must be a serious consideration for users with a limited computer budget which might necessitate premature termination of the computations.

Group theoretic procedures comprise the second category of integer programming algorithms. (See Gomory (1965), and Gorry and Shapiro (1971).) The group theoretic approach begins by transforming the original problem,

$$\begin{aligned} &\text{maximize} && cx \ , \\ &\text{subject to} && \\ & && Ax = b \\ & && x \geq 0, \ \text{integer} \ , \end{aligned} \tag{6}$$

into the equivalent form,

$$\begin{aligned} &\text{maximize} && \bar{c}x_N \ , \\ &\text{subject to} && \\ & && x_B = \bar{b} - \bar{A}x_N \\ & && x_B \geq 0, \ x_N \geq 0 \\ & && x_B, \ x_N \ \text{integer} \ , \end{aligned} \tag{7}$$

where $\bar{b} = B^{-1}b$, $\bar{A} = B^{-1}N$, $\bar{c} = c_N - c_B B^{-1}N \geq 0$, and the columns of A have been partitioned into basic and nonbasic sets B, N respectively, for some dual-feasible basis B.[1/] One then works with problem (7) in place of problem (6). This approach has been called group theoretic because, dropping the nonnegativity restrictions on the variables x_B, (7) can be shown to be equivalent to the group problem,

---

[1/] Usually, an optimal basis B is chosen, though this is not necessary.

$$\begin{aligned} &\text{maximize} \quad \bar{c}x_N \,, \\ &\text{subject to} \\ &\qquad A^* x_N = b^* \pmod{D} \\ &\qquad x_N \geq 0, \ \text{integer} \,, \end{aligned} \tag{8}$$

where $D = |\det B|$, $A^* = D(\bar{A} - [\bar{A}])$, $b^* = D(\bar{b} - [\bar{b}])$, and, in general, $[y]$ denotes the integer part of y (i.e., the largest integer not greater than y). Specialized and efficient solution procedures exist for solving problems such as (8) above.

The third category of integer programming algorithms, and the one which has received the most attention, consists of the enumerative procedures. (See Balas (1965, 1967), Dakin (1965), Geoffrion (1969), Glover (1965), and Land and Doig (1960).) These methods employ a branch-and-bound type search procedure for the optimal solution to the integer programming problem, and computational experience to date has shown them to be the most consistently effective of the three types.

Consider again the integer programming problem (6), and let

$$F_0 \equiv \{x \geq 0 | Ax = b, \ x \text{ integer}\}$$

be the set of feasible solutions to (6). As with the cutting plane algorithms, the standard first step in an enumerative procedure is to solve the associated linear programming relaxation and check the optimal solution - call it $\bar{x}$ - for integrality. The procedure terminates if $\bar{x}$ satisfies the integer restrictions, since this solution must then be optimal.

Usually, of course, several integer-restricted variables will occur at non-integer levels in the optimal LP solution, and further analysis

is required. In accordance with branch-and-bound search strategy, the set of feasible solutions F_0 to (6) is partitioned into (at least) two exhaustive and mutually exclusive subsets F_1, F_2. Most commonly, the partitioning is accomplished by specifying new bounds on one of the variables which violates the integer restrictions in the optimal LP solution $\bar{x}$. That is, some such variable x_k is selected, and the sets F_1, F_2 are formed by

$$F_1 = F_0 \cap \{x | x_k \leq [\bar{x}_k]\}, \qquad F_2 = F_0 \cap \{x | x_k \geq [\bar{x}_k] + 1\} .$$

Thus, two new problems are created:

$$\text{maximize } \{cx | x \in F_1\} , \tag{9}$$

and

$$\text{maximize } \{cx | x \in F_2\} . \tag{10}$$

It is clear that since $F_1 \cup F_2 = F_0$, the optimal integer solution to one of these problems must be the overall optimum to the original problem (6). One of these problems, say (9), is treated immediately, while the other problem, say (10), is put on a list and stored, to be returned to and considered later.

Problem (9) is treated in exactly the same way as the original problem. The integer restrictions are relaxed, and (9) solved as a linear program.[1/] If the optimal LP solution satisfies the integer restrictions, it must clearly be optimal for (9). However, as the optimum for (9) may not be the overall optimum for (6), it then

---

[1/] In practice this is most efficiently done by applying the dual simplex method, starting with the LP-optimal basis for (6), or by parametric programming on the new bounds.

would remain to return to the list of partitioned problems - this list so far contains only the problem (10) - and examine each in turn to see if there exist any better integer solutions.

If the optimal LP solution to (9) does not satisfy the integer restrictions, the feasible set F_1 is itself partitioned in a similar fashion to the partitioning of F_0. As before, one of the two associated problems is added to the list, to be considered later, while the other is examined next. This process - reoptimization, partitioning, etc. - is continued until some partitioned set F_j is obtained for which the associated LP optimum is integer-feasible as well, or it is demonstrated that the optimal solution to (6) cannot lie in the set F_j. In branch-and-bound terminology, the partition F_j is said to be fathomed, and at this point a new problem is extracted from the list and the search for a better integer solution to (6) is continued in exactly the same way. At any given point in the search, the best integer solution found "so far" is called the incumbent. The search is terminated when the list of partitioned problems has been exhausted, and the final incumbent solution must then be optimal. (However, if no incumbent solution has ever been obtained, it follows that (6) must have no feasible solutions.)

The task of demonstrating that a given partition F_j cannot contain an improved integer solution to (6) is accomplished by computing bounds on the optimal objective value. If $\bar{x}$ is the current incumbent solution, the corresponding objective value $c\bar{x}$ provides a lower bound on the maximal value in (6). Upper bounds on the maximum objective value attainable by any feasible integer solution in a given partition F_j of F_0 are easily derived. The optimal objective value obtained in solving the

LP problem associated with F_j provides one such bound. If this bound is not greater than $c\bar{x}$, the partition F_j is fathomed, since then F_j clearly cannot contain a better integer solution to (6) than $\bar{x}$. Alternatively, F_j can also be fathomed if the associated linear program is shown to be infeasible, since this in turn implies $F_j = \emptyset$. In either case, the analysis of the partition F_j is complete.

The enumerative procedure outlined above can be conveniently diagrammed using a tree-search structure in which each of the nodes represents a partitioned subset F_j. (See Figure 1.) The partitioning process is represented by branches in the tree which connect a given set (node) to its two partitioned subsets (descendent nodes). The generic name "branch-and-bound" derives from this visualization of the procedure, in conjunction with the bounding methods discussed above. While the discussion of the branch-and-bound method has been within the context of the integer programming problem (6), it should be kept in mind that this type of "divide and conquer" search strategy is in fact much broader in its range of applicability. (See Mitten (1970) for a formulation of the branch-and-bound procedure in more general terms.) The branch-and-bound concepts introduced here will be drawn upon again in a different context in later chapters.

In addition to the references above, introductory expositions and further details on any or all of the integer programming methods discussed may be found in Garfinkel and Nemhauser (1972), Greenberg (1971), or Hu (1969). Surveys of integer programming algorithms have been presented by Balinski (1965), Lawler and Wood (1966), and Geoffrion and Marsten (1972). This last survey deserves special mention in that there is also

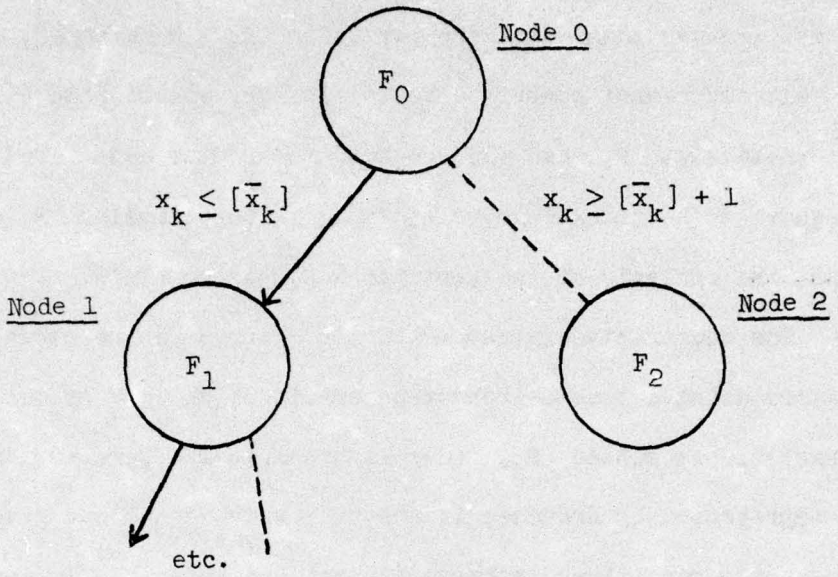

Figure 1: Branch-and-bound tree structure for the general integer programming problem

a general framework developed which serves to both unify and contrast the ideas underlying the three classes of integer programming algorithms. The present discussion has in part followed the presentation in the Geoffrion and Marsten survey.

Despite the wide variety of algorithmic approaches which have been proposed, the art of integer programming today still cannot claim the same degree of success as has been enjoyed in the field of linear programming. Computational surveys of existing computer codes, such as those by Trauth and Woolsey (1969) and Geoffrion and Marsten (1972), indicate that even small or moderately-sized integer programming problems can sometimes require disproportionately large amounts of computation time. Moreover, it has generally been accepted that, even with the best of the available computer codes, solution times for integer

programming problems increase exponentially with the number of problem variables. While it has not yet been demonstrated that it is impossible to construct an integer programming algorithm whose solution time can be bounded by a polynomial function of the number of variables, nonetheless, the inception of a general-purpose integer programming algorithm with the same overall degree of efficiency as the simplex method seems an eternally elusive goal.

By contrast, some particularly notable successes have been enjoyed by special-purpose algorithms specifically designed to treat limited classes of specially-structured integer programming problems. See, for example, Geoffrion and Graves (1973), Healy (1964), Reardon (1974), or Tomlin (1970). The classes considered are generally of significant practical importance, and often the work has been undertaken with a particular application in mind. In the face of the current state of the art of general integer programming algorithms, and, on the other hand, the results obtained in applications using highly specialized algorithms, this latter approach would seem a most viable alternative. In fact, these considerations have served as the philosophical basis for, and have greatly motivated the present work.

1.3 The Decomposition Principle: A Historical Perspective

Linear programming models with the block angular structure arise frequently in many large-scale applications. In a multidivisional corporation, for example, the activities in each of the subdivisions are often quite independent of the activities of other subdivisions. If complete independence holds, of course, the optimal operation of each of the subdivisions can be treated as a separate problem in its

own right. Most commonly, however, the subdivisions must compete for a few resources (e.g., capital, labor) on which there are corporate-wide limitations. This situation naturally leads to mathematical models with the block angular structure.

As practical block angular problems tend to be rather large in size, sometimes involving thousands of constraints and many thousands of variables overall, solution algorithms which proceed by decomposing the problem into its (smaller) subproblems and a single, coordinating "Master Problem" can be of great value. Since the computational effort required to solve linear programming problems increases approximately as the cube of the number of rows in the problem, decomposition algorithms often provide more efficient solution procedures for large, structured linear programs than do the more straightforward linear programming techniques. Moreover, even with today's largest computers, the extreme size of many block angular problems can exceed available in-core (high speed) memory capacity. Since accessing disks, drums, or tapes is a much slower process than in-core memory retrieval, computation time can quickly become excessive for such problems. By breaking the problem into the smaller, more tractable subproblems, however, decomposition algorithms can substantially reduce the number of accesses required of external storage devices, thereby further enhancing computational efficiency.

Because of the practical importance of block angular mathematical programs, and the inherent advantages of decomposition algorithms, much attention has been devoted towards exploiting this special structure when all the problem variables are assumed to be continuous. Analysis

began with the pioneering work of Dantzig and Wolfe (1960), who considered the problem defined by (1)-(4), that is, the linear programming problem associated with (P). Decomposition is achieved by expressing all feasible subproblem solutions as convex combinations of the extreme points of the convex set of such solutions.

Briefly, the central ideas of the Dantzig-Wolfe algorithm are as follows. Let $F^{(j)}$ denote the set of feasible solutions to the j^{th} subproblem, namely,

$$F^{(j)} \equiv \{x^{(j)} | B^{(j)}x^{(j)} \leq \beta^{(j)}, \quad x^{(j)} \geq 0\}, \quad j = 1,2, \ldots , N . \quad (11)$$

We assume for the present discussion that $F^{(j)}$ is bounded.[1/] Let $\gamma_k^{(j)}$ $(k = 1, 2, \ldots , E_j)$ denote the k^{th} extreme point of $F^{(j)}$, where E_j is the (finite) number of such extreme points. Then any $x^{(j)} \in F^{(j)}$ can be expressed as

$$x^{(j)} = \sum_{k=1}^{E_j} w_k^{(j)} \gamma_k^{(j)} \quad (12)$$

for some set of weights $w_k^{(j)}$, $k = 1, 2, \ldots , E_j$, such that

$$\sum_{k=1}^{E_j} w_k^{(j)} = 1 \text{ and } w_k^{(j)} \geq 0, \text{ for } k = 1, 2, \ldots , E_j . \quad (13)$$

Moreover, since any point $x^{(j)}$ which can be expressed in this form must be in $F^{(j)}$, it is seen that (12), (13) provide an equivalent representation for the set $F^{(j)}$.

Thus, letting

[1/] There are minor modifications to the procedure when $F^{(j)}$ is unbounded. See Dantzig and Wolfe (1960).

$$p_k^{(j)} \equiv c^{(j)}\gamma_k^{(j)}, \quad \text{for } j = 1, 2, \ldots, N; \quad k = 1, 2, \ldots, E_j, \tag{14}$$

and

$$\alpha_k^{(j)} \equiv A^{(j)}\gamma_k^{(j)}, \quad \text{for } j = 1, 2, \ldots, N; \quad k = 1, 2, \ldots, E_j, \tag{15}$$

(1)-(4) can be re-written as,

$$\text{(M) maximize} \quad \sum_{k=1}^{E_1} p_k^{(1)} w_k^{(1)} + \sum_{k=1}^{E_2} p_k^{(2)} w_k^{(2)} + \cdots + \sum_{k=1}^{E_N} p_k^{(N)} w_k^{(N)}, \tag{16}$$

subject to

$$\sum_{k=1}^{E_1} \alpha_k^{(1)} w_k^{(1)} + \sum_{k=1}^{E_2} \alpha_k^{(2)} w_k^{(2)} + \cdots + \sum_{k=1}^{E_N} \alpha_k^{(N)} w_k^{(N)} \leq b \tag{17}$$

$$\left.\begin{aligned} \sum_{k=1}^{E_1} w_k^{(1)} &= 1 \\ \sum_{k=1}^{E_2} w_k^{(2)} &= 1 \\ &\ddots \quad \vdots \\ \sum_{k=1}^{E_N} w_k^{(N)} &= 1 \end{aligned}\right\} \tag{18}$$

$$w_k^{(j)} \geq 0 \quad \text{for } j = 1,2, \ldots, N; \; k = 1,2, \ldots, E_j. \tag{19}$$

Problem (M), which Dantzig and Wolfe have called the Master Problem, has only $m + N$ rows, rather than the $m + \sum_{j=1}^{N} m_j$ rows of (P), although in general it may have many more columns than (P). This difficulty is circumvented, however, by using the revised simplex method and generating the columns of (M) only as they are needed. More specifically, in pricing out the current basic feasible solution of (M) at each iteration, the simplex method requires that the column of (M) with lowest relative cost

(if negative)[1] be introduced into the basis next. This column, call it α^*, is determined by

$$\pi\alpha^* = \min_{1 \le j \le N} \{ \min_{1 \le k \le E_j} \pi\alpha_k^{(j)} \} , \tag{20}$$

where π is the current vector of dual multipliers associated with the rows (17) of (M). The outer minimization in (20) is trivial, while the inner minimization is accomplished for each j, using (15), by

$$\min_{1 \le k \le E_j} \pi\alpha_k^{(j)} = \min_{1 \le k \le E_j} (\pi A^{(j)})\gamma_k^{(j)} = \min \rho^{(j)} x^{(j)} , \quad \text{subject to} \quad B^{(j)} x^{(j)} \le \beta^{(j)} \tag{21}$$

where $\rho^{(j)} \equiv \pi A^{(j)}$. From the theory of linear programming, it is well known that at least one optimal solution $x^{(j)*}$ to the j^{th} subproblem (21) will indeed be an extreme point $\gamma_k^{(j)}$ of the set $F^{(j)}$ for some $k = 1, 2, \ldots, E_j$.

Balas (1966) has presented an alternative decomposition algorithm for block angular linear programming problems. In Balas' work, the linking constraints are at first ignored and the subproblems solved separately. Any infeasibilities in the resulting solution are then used to develop a modified set of price vectors for the subproblems. The subproblems are resolved with the new objective functions, and the procedure continues to iterate in this manner until, after a finite number of such iterations, the optimal solution is obtained.

---

[1] If all columns are found to have nonnegative relative costs, the current solution is optimal.

Balas' decomposition algorithm bears many similarities conceptually to the algorithm of Dantzig and Wolfe. Both, for example, provide a means for decentralized planning in a multi-divisional organization through the use of an internal pricing system. However, while in Dantzig and Wolfe's procedure, a feasible solution is obtained first and then iteratively improved upon until the optimum is achieved, Balas' algorithm proceeds by obtaining an "optimal" (infeasible) solution first, then iteratively reducing the infeasibility of the generated solutions while maintaining optimality.

Others who have considered the problem of decomposing structured linear programs are Bennett (1966), Hartman and Lasdon (1970), Ritter (1967), and Rosen (1964). To date, however, little consideration has been given to block angular models in which the decision variables are required to take on only integer values. Yet, especially in integer programming problems, where computation time may increase exponentially with the number of problem variables, it appears that substantial savings can be achieved by solving a judiciously chosen sequence of the smaller subproblems, rather than attacking the problem as a whole. For example, it should be possible to solve ten ten-variable subproblems in much less computer time than would be required to solve one one-hundred variable problem. Moreover, as with its linear programming counterpart, an algorithm for solving large, block angular integer programs which treated only one subproblem at a time could render solvable problems whose overall size would otherwise result in prohibitive in-core storage requirements. These considerations have motivated the present work.

At this point, previous and related results in the area of decomposing

integer programs deserve some mention. Schrage (1973) was apparently the first to consider the advantages of a decomposition algorithm for specially structured integer programming problems. Schrage's paper was primarily concerned with problems in which the constraint matrix possesses a dual angular structure, that is, where the subproblems are tied together by a set of linking columns rather than linking constraints. In a linear programming sense, this is the dual structure to the one considered here.

Reardon (1974) showed how certain capital budgeting problems could be formulated with the dual angular structure, and, developing and refining Schrage's ideas, designed a decomposition algorithm to treat problems with this structure. Schrage's suggestion was that, in using a partial enumeration solution procedure for dual angular integer programs, one should give priority to fixing the values of the linking variables first. Once these columns have been fixed, the problem decomposes of its own accord into independent subproblems. Reardon's idea, then, was to separate the problem into subproblems from the start, thereby relaxing the restriction that the linking variables (called "first stage" variables) must take on the same values in each subproblem. This restriction is implicitly re-imposed by stipulating that each of the subproblems must have identical first-stage search trees. In computational tests of the algorithm, Reardon obtained an approximately linear increase in solution time with problem size, a vast improvement over the roughly exponential growth observed with a non-decomposition approach!

It is natural to consider using Lagrangean relaxation as a means of decomposing block angular integer programs. Such an approach would

be analogous to Balas' infeasibility pricing algorithm for decomposing block angular linear programming problems. Geoffrion (1974) discusses the uses of Lagrangean relaxation in integer programming. The central idea is to simplify an otherwise difficult problem by using a suitably chosen set of multipliers to take any "complicating" constraints (in the present context, these would be the linking constraints) up into the objective function. Thus, using the m-vector of multipliers $\lambda \geq 0$, (P) would be transformed to,

$$\text{maximize} \quad \sum_{j=1}^{N} c^{(j)}x^{(j)} + \lambda(b - \sum_{j=1}^{N} A^{(j)}x^{(j)}) , \tag{22}$$

subject to

$$\left.\begin{aligned} B^{(1)}x^{(1)} &\leq \beta^{(1)} \\ B^{(2)}x^{(2)} &\leq \beta^{(2)} \\ \ddots &\quad \vdots \\ B^{(N)}x^{(N)} &\leq \beta^{(N)} \end{aligned}\right\} \tag{23}$$

$$x^{(j)} \geq 0, \qquad \text{for } j = 1, 2, \ldots, N \tag{24}$$

$$x^{(j)} \text{ integer, for } j = 1, 2, \ldots, N . \tag{25}$$

By choosing specific values λ^* for the multipliers, (22)-(25) then immediately decomposes into the N subproblems:

$$\text{maximize} \quad (c^{(j)} - \lambda^* A^{(j)})x^{(j)} ,$$

subject to

$$B^{(j)}x^{(j)} \leq \beta^{(j)}$$

$$x^{(j)} \geq 0, \text{ integer} ,$$

for $j = 1, 2, \ldots, N$.

Lasdon (1973) has considered an approach along these lines in a production scheduling application.[1/] The difficulty with such an approach, as Lasdon points out, is that in general, due to the non-convexity of the feasible region, there is no guarantee that there exists a choice of values for the multipliers λ that leads to an optimal solution to (P). Worse, one cannot in general even be assured of finding a feasible solution - that is, one satisfying the linking constraints as well as the subproblem constraints - for (P) in this way. In view of Balas' results with block angular linear programs, this state of affairs is somewhat disappointing. However, Lagrangean relaxation may still be useful as a heuristic approach in integer programs where it is not strictly required that all of the constraints be met.

1.4 Overview

In the remaining chapters, then, an exact algorithm for the decomposition of block angular integer programs is developed. A conceptual outline of the algorithm is presented in Chapter 2. Decomposition occurs through resource allocation, and a branch-and-bound search for the optimal allocation of linking resources to subproblems is established.

Details of the algorithm are discussed in Chapters 3-5. Specifically, Chapter 3 deals with the question of how the branching is to be done, while Chapters 4 and 5 develop procedures for computing bounds at each node in the search tree. In Chapter 4, the case in which (P) has only a single linking constraint is considered. In modeling the operation of a multi-divisional organization, such a constraint might represent

---

[1/] Lasdon's paper, as well as Geoffrion's, also contains an excellent discussion of the concepts relating to generalized Lagrange multipliers.

the overall corporate budgeting limitation to which all of the subdivisions (subproblems) are subject. The simplifications that result in this case give rise to a specially structured master problem, and an efficient solution algorithm which exploits this structure is developed. The problem of computing bounds in the presence of more than one linking constraint is addressed in Chapter 5.

Computational results obtained with a Fortran code of the decomposition algorithm on an IBM 370/168 are presented in Chapter 6. Finally, in Chapter 7, we summarize our results and discuss some suggestions for areas of future research.

Some of the results of Chapter 4 are generalized in Kochman (1976), where a class of nonlinear integer programming problems, of which the master problem developed in Chapter 4 is a member, is introduced. A general solution procedure for problems in this class is discussed, and two illustrative examples are presented.

CHAPTER 2

RESOURCE DECOMPOSITION: A CONCEPTUAL OUTLINE

The algorithms of Dantzig and Wolfe (1960) and Balas (1966) discussed in Chapter 1 both utilize pricing mechanisms to achieve the decomposition of block angular linear programs. Lasdon's generalized Lagrange multiplier approach to the integer programming problem might also fall into the category of "price decomposition" algorithms. The difficulty with such approaches in an integer programming context, however, is that, in sharp contrast to the linear programming case, there does not exist any comparably well-defined and well-behaved system of shadow prices. Consequently, an alternative approach is considered here, and the resulting algorithm proceeds through what might be called resource decomposition. In this chapter, we present a conceptual basis for the decomposition algorithm.

2.1 Resource Decomposition

Consider again the problem (P) introduced in Chapter 1:

$$(P) \quad \text{maximize} \quad \sum_{j=1}^{N} c^{(j)}x^{(j)}, \tag{1}$$

subject to

$$\sum_{j=1}^{N} A^{(j)}x^{(j)} \leq b \tag{2}$$

$$B^{(j)}x^{(j)} \leq \beta^{(j)}, \quad \text{for } j = 1, 2, \ldots, N \tag{3}$$

$$x^{(j)} \geq 0, \quad \text{for } j = 1, 2, \ldots, N \tag{4}$$

$$x^{(j)} \text{ integer, for } j = 1, 2, \ldots, N. \tag{5}$$

Let $\hat{x}^{(j)}$, $1 \leq j \leq N$, denote an optimal solution to (P), and, correspondingly, define the m-vectors $\hat{b}^{(j)}$ by

$$\hat{b}^{(j)} \equiv A^{(j)}\hat{x}^{(j)}, \qquad j = 1, 2, \ldots, N . \tag{6}$$

Conceptually, the linking constraints (2) can be viewed as representing common resources which the subproblems must share. Under this interpretation, the $\hat{b}^{(j)}$ in (6) are seen to be in some sense an optimal allocation of these resources to the subproblems. Clearly, were the values $\hat{b}^{(j)}$, $j = 1, 2, \ldots, N$, known beforehand, (P) could be decomposed immediately and the optimal solution $\hat{x}^{(j)}$, $j = 1, 2, \ldots, N$, obtained simply by solving the N subproblems,

$$\begin{aligned} (S_j) \qquad & \text{maximize} \quad c^{(j)}x^{(j)} , \\ & \text{subject to} \\ & A^{(j)}x^{(j)} \leq \hat{b}^{(j)} \\ & B^{(j)}x^{(j)} \leq \beta^{(j)} \\ & x^{(j)} \geq 0, \ \text{integer} , \end{aligned}$$

for $j = 1, 2, \ldots, N$. These subproblems (S_j), $j = 1, 2, \ldots, N$, could be solved by any of the various standard integer programming algorithms discussed in Chapter 1.

Unfortunately, in general an optimal allocation $\hat{b}^{(j)}$, $j = 1, 2, \ldots, N$, is not known until after (P) has been solved. However, in view of the reduction which results, it appears that it may be profitable to "guess" at values for the $\hat{b}^{(j)}$, $j = 1, 2, \ldots, N$, or rather, to devise a systematic means of determining a sequence of such guesses which leads to an optimal allocation in a finite number of steps.

To this end, instead of considering the $\hat{b}^{(j)}$, $j = 1, 2, \ldots, N$, as fixed vectors, we introduce into the problem the additional <u>allocation variables</u> $b^{(j)} = (b_{1j}, b_{2j}, \ldots, b_{mj})^T$, $j = 1, 2, \ldots, N$, where each b_{ij} has the interpretation

$$b_{ij} = \text{amount of } i^{th} \text{ linking resource allocated to } j^{th} \text{ subproblem, } 1 \leq i \leq m, \quad 1 \leq j \leq N .$$

The resulting (equivalent) formulation of the original problem (P) is

(P')

$$\text{maximize} \quad c^{(1)}x^{(1)} + c^{(2)}x^{(2)} + \cdots + c^{(N)}x^{(N)} , \tag{7}$$

subject to

$$b^{(1)} + b^{(2)} + \cdots + b^{(N)} \leq b \tag{8}$$

$$\left.\begin{array}{rcccccc}
A^{(1)}x^{(1)} - b^{(1)} & & & & & & \leq 0 \\
B^{(1)}x^{(1)} & & & & & & \leq \beta^{(1)} \\
 & A^{(2)}x^{(2)} - b^{(2)} & & & & & \leq 0 \\
 & B^{(2)}x^{(2)} & & & & & \leq \beta^{(2)} \\
 & & \ddots & & & & \vdots \\
 & & & & A^{(N)}x^{(N)} - b^{(N)} & & \leq 0 \\
 & & & & B^{(N)}x^{(N)} & & \leq \beta^{(N)}
\end{array}\right\} \tag{9}$$

$$x^{(j)} \geq 0, \text{ integer, for } j = 1, 2, \ldots, N . \tag{10}$$

The transformed problem (P') is itself a block angular integer program, in which the linking constraints are now of the form (8). These constraints on the $b^{(j)}$, $j = 1, 2, \ldots, N$, are necessary to insure that

the linking resources are not over-allocated, and are implied by the inequalities (2) in (P).

Although they may be required to be nonnegative in certain applications (e.g., when they actually represent resource allocations), the allocation variables $b^{(j)}$, $j = 1, 2, \ldots, N$, are not, in general, restricted in sign. We shall assume throughout, however, that each of the matrices $A^{(j)}$, $j = 1, 2, \ldots, N$, has all integer entries. This assumption implies that any feasible integer solution $x^{(j)}$ to the j^{th} subproblem of (P) can use only integeral amounts of the linking resources. Consequently, the search for an optimal allocation of these resources to the subproblems can be restricted to integer values of the allocation variables b_{ij}.[1/]

Reordering the variables in (P'), and writing the problem in block form leads to the representation,

(P")

$$\text{maximize} \quad \left[{}_0T_b^{(1)} \;\middle|\; {}_0T_b^{(2)} \;\middle|\; \cdots \;\middle|\; {}_0T_b^{(N)} \right] \left[c^{(1)}x^{(1)} \;\middle|\; c^{(2)}x^{(2)} \;\middle|\; \cdots \;\middle|\; c^{(N)}x^{(N)} \right] \tag{11}$$

subject to

$$\begin{bmatrix} I & I & \cdots & I \\ -I & & & \end{bmatrix} \leq b \tag{12}$$

$$\begin{bmatrix} A^{(1)} & & & \\ B^{(1)} & & & \\ -I & A^{(2)} & & \\ & B^{(2)} & \ddots & \\ & & A^{(N)} & \\ -I & & B^{(N)} & \end{bmatrix} \leq \begin{bmatrix} 0 \\ \beta^{(1)} \\ 0 \\ \beta^{(2)} \\ \vdots \\ 0 \\ \beta^{(N)} \end{bmatrix} \tag{13}$$

[1/] Note that the assumption of integrality of the entries in each $A^{(j)}$,

$$x^{(j)} \geq 0, \text{ integer, for } j = 1, 2, \ldots, N \tag{14}$$

$$b^{(j)} \text{ integer, for } j = 1, 2, \ldots, N . \tag{15}$$

In this form, (P') takes on the dual angular structure considered by Reardon (1974), with the allocation variables forming the linking columns. In this sense, the algorithm developed here is a specialized procedure for decomposing dual angular integer programs in which the linking columns possess the special structure exhibited in (P"). The outstanding feature of this structure is that each of the linking variables b_{ij} has a non-zero coefficient in exactly one subproblem. Whereas in the general dual angular problem all linking columns must be assigned fixed values before decomposition occurs (at which point the decomposition is total), in (P") partial decomposition occurs whenever all the allocation variables $b^{(j)}$ associated with one given subproblem are fixed. Once values $b^{(j)} = b^{(j)*}$ (possibly nonoptimal) have been specified, the subproblem (S_j) can be solved immediately, independently of the others, and the information obtained used to improve the allocations in the remaining subproblems. This point is clarified in Chapter 3.

2.2 A Branch-and-Bound Search for an Optimal Allocation

Using the concepts of resource decomposition described in the preceding section, the block angular integer program (P) is transformed into the N "separate" subproblems (S_j) by specifying values $b^{(j)*}$ for

---

$j = 1, 2, \ldots, N,$ is sufficient for this conclusion, but not necessary. It may happen in certain applications, for example, that the nature of the linking resources is such that they can be meaningfully divided among the subproblems only in integral amounts.

the allocation variables $b^{(j)}$, $j = 1, 2, \ldots, N$. The problem of finding an optimal solution $x^{(j)*}$, $j = 1,2,\ldots,N$, for (P) thus becomes that of determining an optimal allocation of linking resources to subproblems. The subproblems are tied together, and the possible choices of values $b^{(j)} = b^{(j)*}$, $j = 1, 2, \ldots, N$, are restricted by the inequalities (8). Moreover, under the assumption of integrality in the matrices $A^{(j)}$, $j = 1, 2, \ldots, N$, only integer values for the allocation variables need be considered.

In many practical situations, previous experience with the problem may provide a good initial estimate for an optimal allocation. The difficulty, of course, is that although the subproblem optimal solutions $x^{(j)*}$ corresponding to a given allocation $b^{(j)*}$, $j = 1, 2, \ldots, N$, may constitute a very good - perhaps even optimal - solution to (P), there is no guarantee that these solutions will be optimal, nor even any simple means of determining their optimality or nonoptimality. Consequently, it will generally be necessary to examine a sequence of estimates for the optimal allocation. Obviously, it is desirable to design this sequence in such a manner that only a "reasonable" number of allocations must be examined in all.

The procedure developed here utilizes a branch-and-bound type search strategy to locate an optimal allocation, and is similar in concept to that presented in Section 1.2 in the context of the general integer programming problem. In the present context, the feasible set F_0 to be searched is defined by

$$F_0 \equiv \{(b^{(1)},\ldots,b^{(N)}) \mid \sum_{j=1}^{N} b^{(j)} \leq b, \text{ and } \underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij}, \; b_{ij} \text{ integer } \forall i,j\}, \tag{16}$$

where $b^{(j)} = (b_{ij}, \ldots, b_{mj})^T$, $j = 1, 2, \ldots, N$, $\underline{b}_{ij}$, $\overline{b}_{ij}$ represent lower and upper bounds, respectively, on the allocation variable b_{ij}, $1 \leq i \leq m$, $1 \leq j \leq N$, and b is the vector of linking constraint right-hand sides in (2). Procedures for computing the initial values of the $\underline{b}_{ij}$ and $\overline{b}_{ij}$ are presented in Section 2.3.

The basic structure of the search tree in this case is shown in Figure 1. To begin the search, an allocation $b^{(j)^*} = (b^*_{1j}, \ldots, b^*_{mj})^T$, $j = 1, 2, \ldots, N$, is chosen from the set F_0 and substituted into the subproblems (S_j). Each of the subproblems is solved in turn to obtain its integer optimum $x^{(j)^*}$, $j = 1, 2, \ldots, N$ ("optimal" relative to the allocation $b^{(j)^*}$). If all the subproblems are feasible under the allocation $b^{(j)^*}$, $j = 1, 2, \ldots, N$, the corresponding objective value $z^* = \sum_{j=1}^{N} c^{(j)} x^{(j)^*}$ immediately provides a lower bound for the optimal objective value.

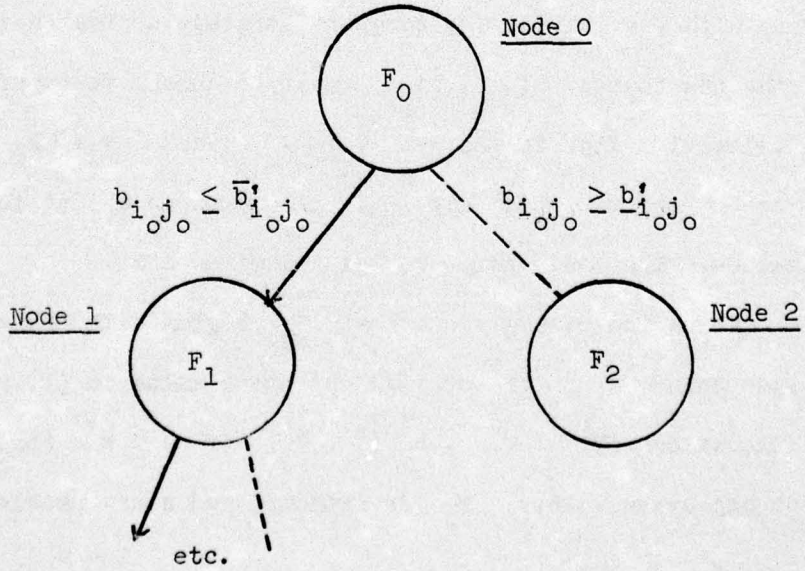

Figure 1: Search tree structure for locating an optimal allocation

At this point it is unknown whether $x^{(j)^*}$, $j = 1, 2, \ldots, N$, is optimal for (P) in an overall sense. We therefore seek an improved solution, or, by failing to find one, verification of the optimality of $x^{(j)^*}$, $j = 1, 2, \ldots, N$. Assuming, then, that $x^{(j)^*}$, $j = 1, 2, \ldots, N$, is not optimal for (P), at least one of the allocations b^*_{ij} must be nonoptimal. Hence, one variable, $b_{i_o j_o}$ say, is chosen as the branching variable, and F_0 is partitioned as indicated in Figure 1. On one branch, a new lower bound on $b_{i_o j_o}$, say $\underline{b}'_{i_o j_o}$ (where $\underline{b}'_{i_o j_o} \geq \underline{b}_{i_o j_o}$), is specified. On the other branch, the upper bound on $b_{i_o j_o}$ is revised to $\overline{b}'_{i_o j_o}$ (where $\overline{b}'_{i_o j_o} \leq \overline{b}_{i_o j_o}$). Thus, the sets F_1, F_2 are defined by

$$F_1 \equiv F_0 \cap \{(b^{(1)}, \ldots, b^{(N)}) | b_{i_o j_o} \leq \overline{b}'_{i_o j_o}\} \tag{17}$$

and

$$F_2 \equiv F_0 \cap \{(b^{(1)}, \ldots, b^{(N)}) | b_{i_o j_o} \geq \underline{b}'_{i_o j_o}\} . \tag{18}$$

In keeping with the "divide and conquer" strategy of branch-and-bound search, the new bounds $\underline{b}'_{i_o j_o}$, $\overline{b}'_{i_o j_o}$ must be chosen to ensure that an optimal allocation lies in the set $F_1 \cup F_2$, while $F_1 \cap F_2 = \emptyset$.[1/] As before, one of the two sets, say F_2, is stored on a list for later investigation. The other set, F_1, is examined next.

The examination of a given subset F_j begins with the computation of an upper bound $\overline{z}$ on the optimal objective value in (P) attainable by any allocation $(b^{(1)}, \ldots, b^{(N)}) \in F_j$. If $\overline{z} \leq z^*$ (the current incumbent objective value), F_j is fathomed and a new problem is taken

---

[1/] In particular, we must have $\underline{b}'_{i_o j_o} > \overline{b}'_{i_o j_o}$. The branching process, including the selection of these revised bounds, is discussed in Chapter 3.

from the list instead. If $\bar{z} > z^*$, however, F_j may contain an allocation leading to an improved objective value in (P). In this case, a promising new allocation $(b^{(1)'}, \ldots, b^{(N)'})$ is chosen from F_j and substituted into the subproblems. From this point, the procedure for F_j is the same as that described above for F_0, with the exception that the new solution $x^{(j)'}$, $j = 1, 2, \ldots, N$, and corresponding objective value $z' = \sum_{j=1}^{N} c^{(j)} x^{(j)'}$ is retained as the (new) incumbent solution only if $z' > z^*$.

We summarize these remarks with a conceptual outline of the algorithmic procedure:

Rudimentary Branch-and-Bound Algorithm (RBBA)

Step 1. Initialize the list to contain only the original set F_0.
Set $z^* = -\infty$.

Step 2. If the list is empty, stop:
If $z^* = -\infty$, (P) is infeasible.
Otherwise, the incumbent is optimal and z^* is the maximal objective value.
If the list is not empty, select one of the sets F_j to be searched next, and remove F_j from the list.

Step 3. (The bounding step.) Compute an upper bound U on the maximal objective value attainable by any allocation in F_j.
If $U \leq z^*$, go to Step 2. (F_j is fathomed.)
Otherwise, select a promising allocation $(b^{(1)^*}, \ldots, b^{(N)^*}) \in F_j$ and go to Step 4.

Step 4. Using the allocation $b^{(j)*}$, solve the j^{th} subproblem (S_j) to obtain the integer optimum $x^{(j)*}$ (if it exists), $j = 1, 2, \ldots, N$.

(i) If any subproblem is infeasible, or if the current allocation $(b^{(1)*}, \ldots, b^{(N)*})$ can otherwise be recognized as nonoptimal, go (immediately) to Step 5. (Don't solve the remaining subproblems.)

(ii) If all subproblems (S_j) are feasible, and (S_k) is unbounded for some $1 \leq k \leq N$ (i.e., (S_k) has no finite optimal solutions), stop. (Problem (P) is unbounded and the current allocation $(b^{(1)*}, \ldots, b^{(N)*})$ is clearly optimal.)

(iii) If all subproblems (S_j) have finite optimal solutions $x^{(j)*}$, $j = 1, 2, \ldots, N$, and if the corresponding objective value $\sum_{j=1}^{N} c^{(j)}x^{(j)*} > z^*$, reset $z^* = \sum_{j=1}^{N} c^{(j)}x^{(j)*}$ and install $x^{(j)*}$, $j = 1,2,\ldots,N$, as the new incumbent. (If $\sum_{j=1}^{N} c^{(j)}x^{(j)*} \leq z^*$, the current incumbent is retained.)

Go to Step 5.

Step 5. (The branching step.) Choose a branching variable $b_{i_o j_o}$ and partition the set F_j as discussed above: e.g.,

(i) In the first subset, F_{j_1}, reset $\underline{b}_{i_o j_o} = \underline{b}^{*}_{i_o j_o}$.

(ii) In the second subset, F_{j_2}, reset $\overline{b}_{i_o j_o} = \overline{b}^{*}_{i_o j_o}$.

Add F_{j_1} to the list.

Then go to Step 3, where now F_{j_2} takes the place of F_j.

The description of the branch-and-bound algorithm above is far from complete. Procedures for computing the objective function bounds required at Step 3 are the subjects of Chapters 4 and 5. It will be seen there that the problem of locating a "promising" new allocation in an unfathomed set F_j in Step 3 is closely connected with the bounding procedures. In fact, the allocation selected for use in Step 4 will be the one in F_j which gives the greatest upper bound on the overall objective function value in (P).

In Step 5, the two points requiring further specification are deciding which allocation variable should be selected as the branching variable, and, once chosen, specifying the bounds $\underline{b}'_{i_o j_o}$, $\bar{b}'_{i_o j_o}$ which define the partitions F_{j_1}, F_{j_2}. These are discussed in the next chapter.

Finally, some decision rule is needed in Step 2 regarding the selection of the subset F_j from the list. In the algorithm developed here, the last-in-first-out (LIFO) rule is used. Under the LIFO strategy, the last set F_j added to the list in Step 5 is always the first one to be chosen for examination next in Step 2. This rule facilitates programming the algorithm, and, more importantly, can increase algorithmic efficiency by reducing the computational effort required in reoptimization. For these reasons, use of the LIFO strategy is a standard practice in many branch-and-bound type search procedures.

The advantages of the decomposition procedure become apparent in Step 4. For the reasons discussed in Chapter 1, the most significant feature of the algorithm is the fact that only the subproblems (S_j) ever need be solved explicitly. Moreover, it will be seen that the subproblems may be solved by any linear-programming based method. This

is advantageous if the subproblems themselves possess structures that can be efficiently exploited by specialized solution procedures. Finally, it will usually be possible to recognize that an allocation $b^{(j)*}$, $j = 1, 2, \ldots, N$, is nonoptimal without solving all of the subproblems. When this situation occurs, the computations in Step 4 are terminated, and we proceed immediately to Step 5.

2.3 Computing Bounds on the Allocation Variables

For most problems, the initial specification of the bounds $\underline{b}_{ij}$, $\overline{b}_{ij}$ $(1 \leq i \leq m,\ 1 \leq j \leq N)$ defining the set F_0 is not crucial, provided of course that the intervals $[\underline{b}_{ij}, \overline{b}_{ij}]$ are chosen sufficiently wide to ensure that an optimal allocation is included. In this section we present some simple procedures for computing valid initial bounds on the allocation variables.

Suppose first that we have been given upper and lower bounds $U^{(j)}$, $L^{(j)}$, respectively, on the vector of decision variables $x^{(j)}$ in the j^{th} subproblem:

$$L^{(j)} \leq x^{(j)} \leq U^{(j)}, \qquad j = 1, 2, \ldots, N. \tag{19}$$

The bounds $L^{(j)}$, $U^{(j)}$ in (19) often can be readily inferred from the functional constraints, and, inasmuch as many practical integer programming algorithms require the specification of such bounds prior to solution of the problem, the assumption is not a very restrictive one.

Define the index sets P_{ij}, N_{ij} by

$$P_{ij} = \{1 \leq k \leq n_j \,|\, A_{ik}^{(j)} \geq 0\}, \quad \text{and} \quad N_{ij} = \{1 \leq k \leq n_j \,|\, A_{ik}^{(j)} < 0\}, \tag{20}$$

for $i = 1, 2, \ldots, m$, and $j = 1, 2, \ldots, N$, where $A_{ik}^{(j)}$ denotes

the entry in the i^{th} row, k^{th} column of the matrix $A^{(j)}$ of linking constraint coefficients for the j^{th} subproblem. Then for $1 \leq i \leq m$, $1 \leq j \leq N$,

$$\sum_{k=1}^{n_j} A_{ik}^{(j)} x_k^{(j)} = \sum_{k \in P_{ij}} A_{ik}^{(j)} x_k^{(j)} + \sum_{k \in N_{ij}} A_{ik}^{(j)} x_k^{(j)}$$

$$\geq \sum_{k \in P_{ij}} A_{ik}^{(j)} L_k^{(j)} + \sum_{k \in N_{ij}} A_{ik}^{(j)} U_k^{(j)}, \quad \text{by (19) and (20)}.$$

Hence,

$$\underline{b}_{ij} \equiv \sum_{k \in P_{ij}} A_{ik}^{(j)} L_k^{(j)} + \sum_{k \in N_{ij}} A_{ik}^{(j)} U_k^{(j)}, \qquad 1 \leq i \leq m, \; 1 \leq j \leq N, \quad (21)$$

represents a valid lower bound on the allocation variable b_{ij}.

Similarly, it is easy to see that

$$\overline{b}_{ij} \equiv \sum_{k \in P_{ij}} A_{ik}^{(j)} U_k^{(j)} + \sum_{k \in N_{ij}} A_{ik}^{(j)} L_k^{(j)}, \qquad 1 \leq i \leq m, \; 1 \leq j \leq N, \quad (22)$$

defines a valid upper bound for b_{ij}. In (21), (22), if either of the sets P_{ij}, N_{ij} is empty, the corresponding sum is taken to be zero.

Both the bounds (21), (22) are valid in the sense that the bounds $L^{(j)}$, $U^{(j)}$ on the vector of decision variables $x^{(j)}$ imply that the j^{th} subproblem must "use" at least $\underline{b}_{ij}$ units of the i^{th} linking resource, and cannot possibly use more than $\overline{b}_{ij}$ units. Thus, there must exist an optimal allocation of linking resources to subproblems with $\underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij}$ for all $i = 1, 2, \ldots, m$, and $j = 1, 2, \ldots, N$.

Having already computed the lower bounds $\underline{b}_{ij}$ from (21), the upper bounds $\overline{b}_{ij}$ in (22) can possibly be tightened, using (8), since

$$\sum_{k=1}^{N} b_{ik} \leq b_i \Longrightarrow b_{ij} \leq b_i - \sum_{k \neq j} b_{ik}$$

$$\leq b_i - \sum_{k \neq j} \underline{b}_{ik} .$$

Thus,

$$\overline{b}'_{ij} \equiv \min\{\overline{b}_{ij},\ b_i - \sum_{k \neq j} \underline{b}_{ik}\}, \qquad 1 \leq i \leq m,\ 1 \leq j \leq N , \tag{23}$$

gives at least as tight an upper bound on b_{ij} as (22) alone.

There are three special cases of (21), (22) which deserve further comment due to the frequency with which they arise. Firstly, if, as will usually be the case, we have $L^{(j)} \equiv 0$ for $j = 1, 2, \ldots, N$ (nonnegativity), (21), (22) reduce to

$$\underline{b}_{ij} = \sum_{j \in N_{ij}} A_{ik}^{(j)} U_k^{(j)}, \text{ and } \overline{b}_{ij} = \sum_{k \in P_{ij}} A_{ik}^{(j)} U_k^{(j)}, \quad 1 \leq i \leq m,\ 1 \leq j \leq N . \tag{24}$$

Moreover, in many practical problems the decision variables $x^{(j)}$ are restricted to take on only the values 0 or 1. In this case, (24) simplifies even further to

$$\underline{b}_{ij} = \sum_{j \in N_{ij}} A_{ik}^{(j)}, \text{ and } \overline{b}_{ij} = \sum_{k \in P_{ij}} A_{ik}^{(j)}, \quad 1 \leq i \leq m,\ 1 \leq j \leq N . \tag{25}$$

Finally, in many applications (e.g., in capital budgeting problems) all the constraint coefficients are nonnegative. From (24), it is clear that if $N_{ij} = \emptyset$ for some j and $1 \leq i \leq m$ (i.e., $A^{(j)} \geq 0$), and there are nonnegativity restrictions on the $x^{(j)}$ ($L^{(j)} \equiv 0$), then $\underline{b}_{ij} = 0$ for $1 \leq i \leq m$.

So far we have considered only the bounds on the decision variables, and, to some extent, the implications of the linking constraints (8) in computing upper and lower bounds on the allocation variables. Tighter bounds than those given by (21) or (22) may be obtained (at greater computational expense, of course) by using the subproblem constraints (3) as well. For $1 \leq i \leq m$ and $1 \leq j \leq N$, consider the problem,

$$\text{maximize} \quad \sum_{k=1}^{n_j} A_{ik}^{(j)} x_k^{(j)} , \tag{26}$$

subject to

$$B^{(j)} x^{(j)} \leq \beta^{(j)} \tag{27}$$

$$L^{(j)} \leq x^{(j)} \leq U^{(j)} \tag{28}$$

$$x^{(j)} \text{ integer} . \tag{29}$$

By the same reasoning as before, the optimal solution $x^{(j)*}$ to (26)-(29) provides a valid upper bound on b_{ij} by

$$\overline{b}_{ij} = \sum_{k=1}^{n_j} A_{ik}^{(j)} x_k^{(j)*} . \tag{30}$$

Similarly, if the objective function (26) is minimized in (26)-(29), the resulting optimal solution $\hat{x}^{(j)}$ provides a lower bound on b_{ij} by

$$\underline{b}_{ij} = \sum_{k=1}^{n_j} A_{ik}^{(j)} \hat{x}_k^{(j)} . \tag{31}$$

While the bounds $\underline{b}_{ij}$, $\overline{b}_{ij}$ given in (30), (31) will generally be much tighter than those provided by (21), (22), the extra computational effort required can be considerable, since it involves solving $2mN$ integer programs. The work required can be mitigated to some extent if an enumerative algorithm is used. In particular, note that for fixed

j the feasible region defined by (27)-(29) is the same for all i, so feasible solutions found in the course of solving (26)-(29) for one i can be used as initial incumbents in solving (26)-(29) for different i. Nonetheless, for most problems it is doubtful that the work expended to generate the bounds (30), (31) will prove worthwhile. A more tractable alternative might be to drop the integer restrictions (29) and solve the linear program (26)-(28) instead. In this case, the upper and lower bounds $\bar{b}_{ij}$, $\underline{b}_{ij}$ are given, respectively, by the greatest integer not exceeding the maximal objective value obtained in (26)-(28), and the least integer not less than the minimal objective value.

A further relaxation of (26)-(28) which may be useful can be obtained by taking a weighted nonnegative sum of the constraints (27) to transform (26)-(28) into a knapsack problem:

$$\left.\begin{matrix}\text{maximize}\\ \text{minimize}\end{matrix}\right\} \quad \sum_{k=1}^{n_j} A_{ik}^{(j)} x_k^{(j)} \ ,$$

subject to (32)

$$(\lambda B^{(j)}) x^{(j)} \leq \lambda \beta^{(j)}$$

$$L^{(j)} \leq x^{(j)} \leq U^{(j)} \ .$$

Here, $\lambda \geq 0$ is an m_j-vector of weights which may be chosen in any reasonable manner, and we shall not elaborate on this point. (The choice $\lambda_k = 1$, $1 \leq k \leq m_j$ - in which case the functional constraint in problem (32) is simply the sum of the constraints (27) - will usually suffice.) Since linear programming knapsack problems can be solved almost by

inspection,[1/] the bounds $\underline{b}_{ij}$, $\overline{b}_{ij}$ obtained from (32) require little more computational effort than those provided by (21), (22), and will generally be significantly tighter.

We conclude this section with a word of caution concerning one <u>invalid</u> procedure for computing upper bounds $\overline{b}_{ij}$ on the allocation variables. Consider the problem,

$$\begin{aligned} &\text{maximize} && c^{(j)}x^{(j)} , \\ &\text{subject to} && \\ & && B^{(j)}x^{(j)} \leq \beta^{(j)} \\ & && x^{(j)} \geq 0, \ \text{integer} , \end{aligned} \tag{33}$$

for $j = 1, 2, \ldots, N$. Problem (33) is the relaxation of (S_j) obtained by assigning arbitrarily large values to $b^{(j)*}$, or, equivalently, by dropping the linking constraints entirely. Denoting the (finite) optimal solution (if one exists) to (33) by $\overline{x}^{(j)}$, it is tempting to take $\overline{b}^{(j)} \equiv A^{(j)}\overline{x}^{(j)}$ as providing, simultaneously, upper bounds on b_{ij} for all $i = 1, 2, \ldots, m$. The reasoning is that given (in effect) infinite amounts of the linking resources, the j^{th} subproblem uses only $A^{(j)}\overline{x}^{(j)}$ units of these resources in an optimal solution. Hence, there must exist an optimal allocation $b^{(j)*}$, $j = 1, 2, \ldots, N$, of the (finite) vector of linking resources b, with $b^{(j)*} \leq A^{(j)}\overline{x}^{(j)}$ for each $j = 1, 2, \ldots, N$.

Unfortunately, this line of reasoning is erroneous, and it is easy to construct counter examples. The difficulty arises from what might

---

[1/] The solution of bounded-variable linear programming knapsack problems is discussed in Chapter 4.

be called a substitution effect among the linking resources. Thus, while only $\sum_{k=1}^{n_j} A_{ik}^{(j)}\bar{x}_k^{(j)}$ units of the i^{th} linking resource are used when unlimited amounts of all m linking resources are available, an allocation of $b_{\ell j}^{*} < \sum_{k=1}^{n_j} A_{\ell k}^{(j)}\bar{x}_k^{(j)}$ for some $1 \leq \ell \leq m$ can cause <u>increased</u> consumption of the other resources in the corresponding optimal solution to the j^{th} subproblem. In fact, the substitution of other resources as an adjustment to the decreased availability of a given scarce resource is a common phenomenon in mathematical programming problems.

Of course, when (P) has only a single linking constraint the substitution effect is eliminated. In this case, the upper bound $\bar{b}^{(j)} \equiv A^{(j)}\bar{x}^{(j)}$ obtained from (33) is indeed valid.

CHAPTER 3

BRANCHING IN THE RUDIMENTARY BRANCH-AND-BOUND ALGORITHM

The branching step (Step 5) in the Rudimentary Branch-and-Bound Algorithm (RBBA) presented in Chapter 2 is incomplete insofar as no decision rules have been given to guide the selection of the branching variable $b_{i_o j_o}$ or the specification of the revised bounds associated with the partitioning process. Before addressing these issues, however, it will be useful to introduce some preliminary concepts and notation.

3.1 Preliminaries

We begin with a key definition.

Definition: An allocation $b_{ij} = b^*_{ij}$, $1 \le i \le m$, $1 \le j \le N$, is feasible if $(b^{(1)^*}, \ldots, b^{(N)^*}) \in F_o$, where $b^{(j)^*} = (b^*_{1j}, \ldots, b^*_{mj})$, $j = 1, 2, \ldots, N$, and F_o is as given by equation (2.16).

Note that the feasibility of an allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ is defined in terms of membership in the set F_o, without regard to whether the subproblems (S_j) are feasible under $(b^{(1)^*}, \ldots, b^{(N)^*})$.

Next, let $v(S_j | b^{(j)^*})$ denote the maximal objective value in the j^{th} subproblem (S_j) when the linking constraint right-hand sides in (S_j) are set to $b^{(j)} = b^{(j)^*} = (b^*_{1j}, \ldots, b^*_{mj})$. Similarly, let $v(P|(b^{(1)^*}, \ldots, b^{(N)^*}))$ denote the maximal objective value in problem (P) under the allocation $b_{ij} = b^*_{ij}$, $1 \le i \le m$, $1 \le j \le N$. That is,

$$v(P|(b^{(1)^*}, \ldots, b^{(N)^*})) = \sum_{j=1}^{N} v(S_j | b^{(j)^*}).$$

Definition: A feasible allocation $(b^{(1)*}, \ldots, b^{(N)*})$ is dominated if there exists another distinct feasible allocation $(b^{(1)'}, \ldots, b^{(N)'})$ such that $v(P|(b^{(1)*}, \ldots, b^{(N)*})) \leq v(P|(b^{(1)'}, \ldots, b^{(N)'}))$.

Also, a set S of feasible allocations will be called dominated if every allocation $(b^{(1)}, \ldots, b^{(N)}) \in S$ is dominated. Clearly, any allocation or set of allocations which can be shown to be dominated will not have to be examined further in the search for an optimal allocation.

Finally, from the definitions of the sets F_0, F_1, F_2, etc., in Chapter 2 (see equations (2.16), (2.17), (2.18)), it is seen that each of the subsets F_n differs from the original set of feasible allocations F_0, and other subsets F_k, $k \neq n$, only in the values of the bounds $\underline{b}_{ij}$, $\overline{b}_{ij}$ on the allocation variables. Suppressing the common requirements $\sum_{j=1}^{N} b^{(j)} \leq b$, $b^{(j)}$ integer, for $j = 1, 2, \ldots, N$, the sets F_n may be identified by these bounds. Thus, in the following discussion, we adopt the notation shown in Figure 1 for representing the nodes in the search tree. The node depicted in Figure 1 would correspond to the set F_n defined by $F_n = \{(b^{(1)}, \ldots, b^{(N)}) | \sum_{j=1}^{N} b^{(j)} \leq b, \ \underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij}, \ b_{ij}$ integer, for $1 \leq i \leq m, 1 \leq j \leq N\}$.

3.2 Branching in the RBBA

The branching rule developed in this section is motivated primarily by the desire to keep as small as possible the number of allocations which must be examined overall in the course of the search. This objective is accomplished by first recognizing that non-zero slacks in the optimal subproblem solutions can be used to identify dominated allocations, and

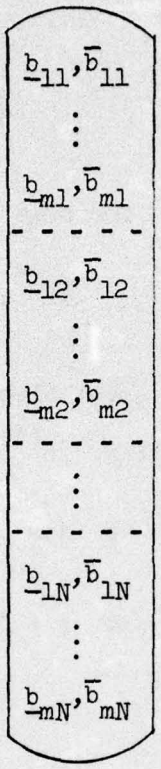

Figure 1: Representation of node n in the search tree

then branching in such a way as to eliminate these allocations from further consideration. The procedure is derived from the following simple results.

Theorem 1: Consider the block angular problem (P), and let $(b^{(1)^*}, \ldots, b^{(N)^*})$ be a feasible allocation. Suppose subproblem (S_{j_0}) is solved using the vector of linking constraint right-hand sides $b^{(j_0)^*}$. Assuming (S_{j_0}) is feasible under the allocation $b^{(j_0)^*}$, let $x^{(j_0)^*}$ denote the optimal solution, and $s^{(j_0)^*} = (s^*_{1j_0}, \ldots, s^*_{mj_0})$ the corresponding vector of slacks in

the linking constraints. That is,

$$s^{(j_0)^*} \equiv b^{(j_0)^*} - A^{(j_0)} x^{(j_0)^*} \geq 0 .$$

Then if any $s^*_{ij_0} > 0$ for some $i = 1, 2, \ldots, m$, the allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ is dominated.

Proof: Let j_1 index some other subproblem (S_{j_0}), where $1 \leq j_1 \leq N$, $j_1 \neq j_0$, and consider the allocation $(b^{(1)'}, \ldots, b^{(N)'})$ defined by

$$b^{(j)'} = \begin{cases} b^{(j_1)^*} + s^{(j_0)^*}, & \text{for } j = j_1 \neq j_0 \\ b^{(j_0)^*} - s^{(j_0)^*}, & \text{for } j = j_0 \\ b^{(j)^*}, & \text{for } j = 1,2,\ldots,N,\ j \neq j_0, j_1 . \end{cases} \tag{1}$$

We will show that the allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ is dominated by $(b^{(1)'}, \ldots, b^{(N)'})$.

Firstly, under the hypothesis that $s^*_{ij_0} > 0$ for some $i = 1, 2, \ldots, m$, it is clear that $(b^{(1)^*}, \ldots, b^{(N)^*})$, $(b^{(1)'}, \ldots, b^{(N)'})$ are distinct. Hence, consider the problem,

$$(S'_{j_0}) \qquad \text{maximize} \quad c^{(j_0)} x^{(j_0)} ,$$

$$\text{subject to} \quad A^{(j_0)} x^{(j_0)} \leq b^{(j_0)'}$$

$$B^{(j_0)} x^{(j_0)} \leq \beta^{(j_0)}$$

$$x^{(j_0)} \geq 0, \ \text{integer} .$$

Since $b^{(j_0)'} \leq b^{(j_0)^*}$, the set of feasible solutions to (S'_{j_0}) is

a subset of the set of feasible solutions to (S_{j_0}). However, since $A^{(j_0)} x^{(j_0)^*} = b^{(j_0)^*} - s^{(j_0)^*} = b^{(j_0)'}$, $x^{(j_0)^*}$ is still feasible for (S'_{j_0}), and hence must also be optimal for (S'_{j_0}).

Thus, since $b^{(j_1)'} \geq b^{(j_1)^*}$, and since $x^{(j_0)^*}$ is optimal for both (S_{j_0}) and (S'_{j_0}), it follows from (1) that

$$v(S_j|b^{(j)^*}) \begin{cases} = v(S_j|b^{(j)'}) \ , & \text{for } j = 1,2,\ldots,N,\ j \neq j_0, j_1 \\ = v(S_{j_0}|b^{(j_0)'}), & \text{for } j = j_0 \\ \leq v(S_{j_1}|b^{(j_1)'}), & \text{for } j = j_1 \ . \end{cases} \tag{2}$$

Summing over j in (2), $\sum_{j=1}^{N} v(S_j|b^{(j)^*}) \leq \sum_{j=1}^{N} v(S_j|b^{(j)'})$, or, equivalently, $v(P|(b^{(1)^*}, \ldots, b^{(N)^*})) \leq v(P|(b^{(1)'}, \ldots, b^{(N)'}))$.

It follows by definition that the allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ is dominated.

Q.E.D.

Corollary: Under the same hypotheses as in Theorem 1, and with the allocation $(b^{(1)'}, \ldots, b^{(N)'})$ defined by (1), the set of allocations

$$S \equiv \{(b^{(1)}, \ldots, b^{(N)}) \in F_0 | b^{(j_0)'} \leq b^{(j_0)} \leq b^{(j_0)^*}, b^{(j_0)} \neq b^{(j_0)'}\} \tag{3}$$

is dominated.

Proof: Arguing as in Theorem 1, each allocation $(b^{(1)}, \ldots, b^{(N)}) \in S$ can be shown to be dominated by the allocation $(b^{(1)'}, \ldots, b^{(N)'})$.

Q.E.D.

For completeness, it is also necessary to consider the case where one of the subproblems (S_j) is discovered to be infeasible under a given allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$.

Theorem 2: Consider the allocation $(b^{(1)^*}, \ldots, b^{(N)^*}) \in F_0$, and suppose the subproblem (S_{j_0}) is found to be infeasible when solved using the vector of linking constraint right-hand sides $b^{(j_0)^*}$. Then $(b^{(1)^*}, \ldots, b^{(N)^*})$ is dominated.

Proof: Adopting the standard convention that $v(S_j | b^{(j_0)^*}) = -\infty$ if (S_{j_0}) is infeasible under $b^{(j_0)^*}$, it follows that $v(P|(b^{(1)^*}, \ldots, b^{(N)^*})) = -\infty$. Hence $(b^{(1)^*}, \ldots, b^{(N)^*})$ is dominated by any other allocation $(b^{(1)'}, \ldots, b^{(N)'}) \in F_0$.

Q.E.D.

Corollary: Under the same hypotheses as in Theorem 2, the set of allocations

$$S \equiv \{(b^{(1)}, \ldots, b^{(N)}) \in F_0 | b^{(j_0)} \leq b^{(j_0)^*}\} \tag{4}$$

is dominated.

Proof: For each allocation $(b^{(1)}, \ldots, b^{(N)}) \in S$, (S_{j_0}) must also be infeasible under $b^{(j_0)}$. The result follows by Theorem 2.

Q.E.D.

Theorems 1, 2, and their corollaries are the main results on which the branching rule is based. Specifically, we wish to branch in such a way as to rule out of further consideration any set of allocations S

satisfying either (3) or (4). Suppose at a given node n in the RBBA, the allocation $(b^{(1)*}, \ldots, b^{(N)*}) \in F_n$ is selected to be examined next, and subproblem (S_{j_0}) is solved using $b^{(j_0)*}$. For $i = 1,2,\ldots,m$, let

$$s^*_{ij_0} = \begin{cases} b^*_{ij_0} - \sum_{k=1}^{n_{j_0}} A_{ik}^{(j_0)} x_k^{(j_0)*}, & \text{if } (S_{j_0}) \text{ is feasible under } b^{(j_0)*} \\ M, & \text{if } (S_{j_0}) \text{ is infeasible under } b^{(j_0)*}, \end{cases} \quad (5)$$

where $x^{(j_0)*}$ is as in Theorem 1 and M is an arbitrarily large number. Then using the notation in Figure 1, the branching at node n is done as indicated by Figure 2. The bounds $\underline{b}_{ij_0}$, $\bar{b}_{ij_0}$, $i = 1, 2, \ldots, m$, in subproblem (S_{j_0}) at node $n+1$ are revised as a result of the branching procedure. Each of the right-hand branches added to the search tree in Figure 2 defines a new subset F_k to be added to the list and examined later. However, the subset F_{n+1} represented by node $n+1$ is considered next.

The new bounds on the allocation variables b_{ij_0}, $i = 1,2,\ldots,m$, in (S_{j_0}) at node $n+1$ are given by

$$\underline{b}_{ij_0} = b^*_{ij_0} - s^*_{ij_0} \quad \text{and} \quad \bar{b}_{ij_0} = b^*_{ij_0}, \quad \text{for } i = 1, 2, \ldots, m, \quad (6)$$

where $s^*_{ij_0}$, $i = 1, 2, \ldots, m$, is given by (5). If (S_{j_0}) was feasible under the vector of linking constraint right-hand sides $b^{(j_0)*}$, then $\underline{b}_{ij_0} = b'_{ij_0}$, $i = 1, 2, \ldots, m$, where $b^{(j_0)'}$ is as defined in (1). In the set F_{n+1}, we have in this case that $b^{(j_0)'} \leq b^{(j_0)} \leq b^{(j_0)*}$,

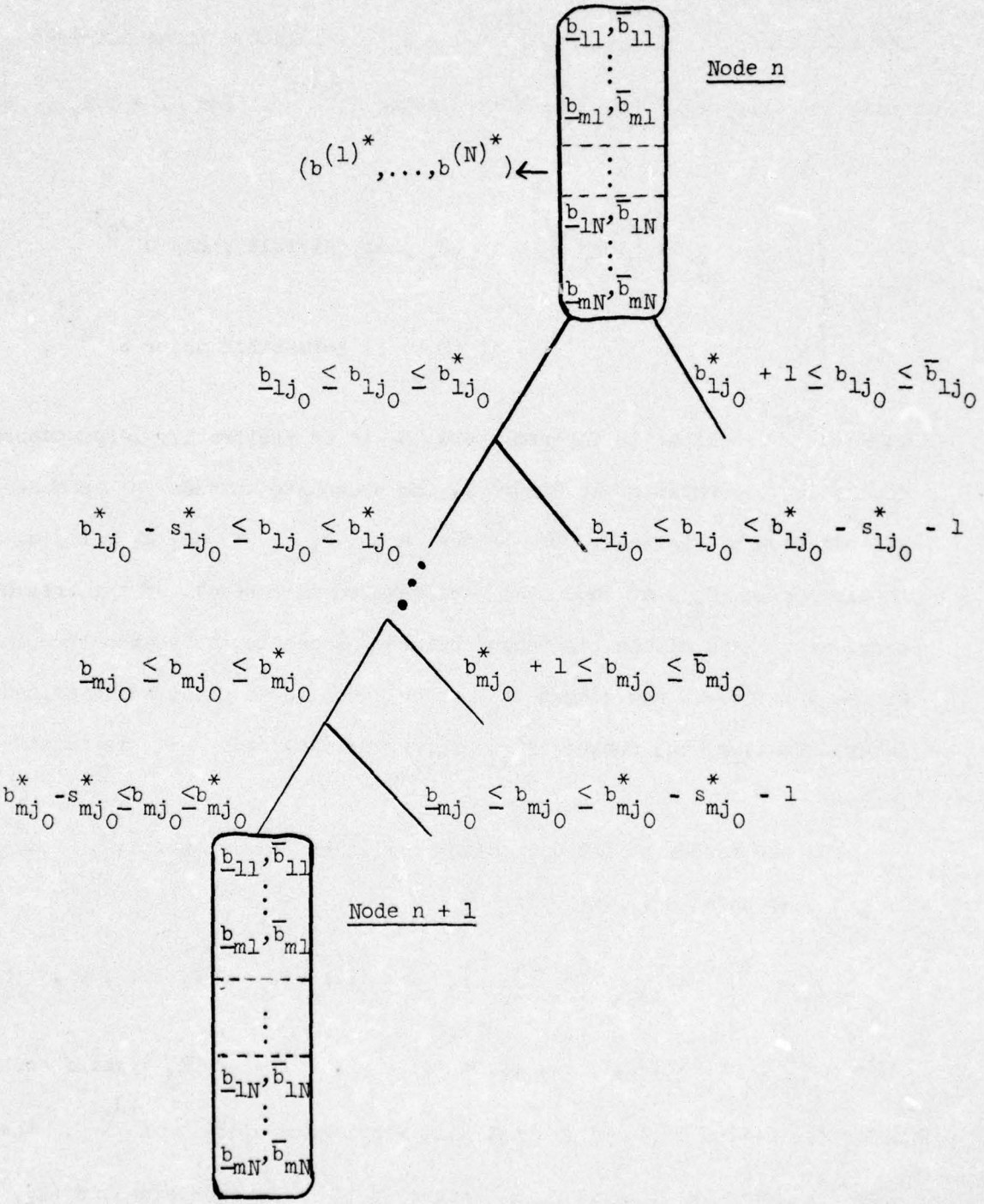

Figure 2: General branching procedure at node n

and it follows from the corollary to Theorem 2 that every allocation in F_{n+1} is dominated, except those with $b^{(j_0)} = b^{(j_0)'}$.[1] Consequently, we can fix the allocation variables in subproblem (S_{j_0}) at the values $b^{(j_0)} = b^{(j_0)'}$ at node n+1, and also in any nodes descendant from node n+1, thereby ruling out of further consideration the set of allocations S defined by (3). Moreover, since, as seen in Theorem 1, the optimal solution $x^{(j_0)*}$ obtained in (S_{j_0}) under the allocation $b^{(j_0)*}$ remains optimal under $b^{(j_0)'}$, it will not be necessary to resolve the j_0^{th} subproblem at node n+1, nor at any of its descendants.

Similar remarks apply if (S_{j_0}) was infeasible under the vector of linking constraint right-hand sides $b^{(j_0)*}$. In this case, it is sufficient in (5) to choose

$$M \geq \max_{1 \leq i \leq m} (b^*_{ij_0} - \underline{b}_{ij_0}) , \tag{7}$$

where here $\underline{b}_{ij_0}$ is the initial lower bound on b_{ij_0}, i = 1,2,...,m, used in defining the set F_n. Then after branching as in Figure 2, the set F_{n+1} corresponding to node n+1 consists of all allocations $(b^{(1)}, \ldots, b^{(N)}) \in F_n$ with $b^{(j_0)} \leq b^{(j_0)*}$. By the corollary to Theorem 2, all these allocations are dominated; hence, F_{n+1} is fathomed. In this way, the desired result is achieved: the set S of dominated allocations defined in (4) is ruled out of further consideration.

---

[1] Note that if $b'_{ij_0} < \underline{b}_{ij_0}$ for some i = 1, 2, ... , m (where here $\underline{b}_{ij_0}$ is the lower bound on b_{ij_0} in the initial set F_n), the set F_{n+1} consists entirely of dominated allocations. In this case, node n+1 is immediately fathomed.

The branching procedure is clarified with the aid of a small example. Consider the problem,[1/]

$$\text{maximize} \quad 8x_1 + 5x_2 + 6x_3 + 9x_4 + 7x_5 + 9x_6 + 6x_7 + 5x_8 ,$$

subject to

$$\begin{aligned}
5x_1 + 3x_2 \qquad + 2x_4 \qquad + 3x_6 + 4x_7 + 6x_8 &\leq 15 \\
2x_1 \qquad + 4x_3 + 3x_4 + 7x_5 \qquad + x_7 \qquad &\leq 10 \\
2x_1 + 4x_2 + 3x_3 \qquad\qquad\qquad\qquad &\leq 10 \\
7x_1 + 3x_2 + 6x_3 \qquad\qquad\qquad\qquad &\leq 15 \\
5x_1 \qquad + 3x_3 \qquad\qquad\qquad\qquad &\leq 12 \qquad (8) \\
3x_4 + x_5 + 2x_6 \qquad\qquad &\leq 7 \\
2x_4 + 4x_5 + 3x_6 \qquad\qquad &\leq 9 \\
8x_7 + 5x_8 &\leq 25 \\
7x_7 + 9x_8 &\leq 30 \\
6x_7 + 4x_8 &\leq 20
\end{aligned}$$

$$x_j \geq 0, \text{ integer}, \quad j = 1, 2, \ldots, 8 .$$

Problem (8) is a block angular integer program of the form (P), with $m = 2$ linking constraints and $N = 3$ subproblems. Using equations (24), (23) of Chapter 2, the initial bounds $0 \leq b_{ij} \leq 15$, $0 \leq b_{ij} \leq 10$, $j = 1, 2, 3$, can be derived to define the feasible set F_0.

Suppose the initial allocation $(b^{(1)^*}; b^{(2)^*}; b^{(3)^*}) = (2,10; 9,0; 4,0)$ is selected for examination at node 0. Problem (8) decomposes, and the first subproblem becomes

---

[1/] This example is taken from Hillier and Lieberman (1974), p. 145. The right-hand sides in the linking constraints have been halved, and integer restrictions imposed on x_j, $j = 1, 2, \ldots, 8$.

$$(S_1) \qquad \text{maximize} \quad 8x_1 + 5x_2 + 6x_3 ,$$

subject to

$$\begin{aligned} 5x_1 + 3x_2 \qquad &\leq 2 \\ 2x_1 \qquad + 4x_3 &\leq 10 \\ 2x_1 + 4x_2 + 3x_3 &\leq 10 \\ 7x_1 + 3x_2 + 6x_3 &\leq 6 \\ 5x_1 \qquad + 3x_3 &\leq 12 \\ x_j \geq 0, \text{ integer}, \; j &= 1,2,3 . \end{aligned}$$

The optimal solution to (S_1) is easily seen to be $x^{(1)^*} = (x_1^*, x_2^*, x_3^*) = (0,0,2)$, so that in this case the slack vector is $s^{(1)^*} = (s_{11}^*, s_{21}^*) = (2,2)$. By the corollary to Theorem 1, the set of allocations

$$S = \{(b^{(1)}, b^{(2)}, b^{(3)}) \mid (0,8) \leq (b_{11}, b_{21}) \leq (2,10), \; (b_{11}, b_{21}) \neq (0,8)\}$$

is immediately seen to be dominated, and the partitioning of F_0 is carried out as shown in Figure 3. At node 1 in Figure 3, and also in any of its descendant nodes, the allocation variables in the first subproblem may be fixed at $b_{11} = 0$, $b_{21} = 8$, since all other allocations in the set F_1 represented by node 1 are dominated. As in Theorem 1, $x^{(1)^*} = (0,0,2)$ remains optimal for the first subproblem in the case $b^{(1)^*} = (0,8)$, and consequently, subproblem 1 will not need to be resolved at any nodes descendant from node 1.

3.3 Additional Comments on the Branching Procedure

The above example also serves to illustrate that often it will not be necessary to add to the search tree all of the branches indicated in Figure 2. The general branching procedure generates two new subsets for each allocation variable b_{ij_0}, $i = 1, 2, \ldots, m$, in the subproblem

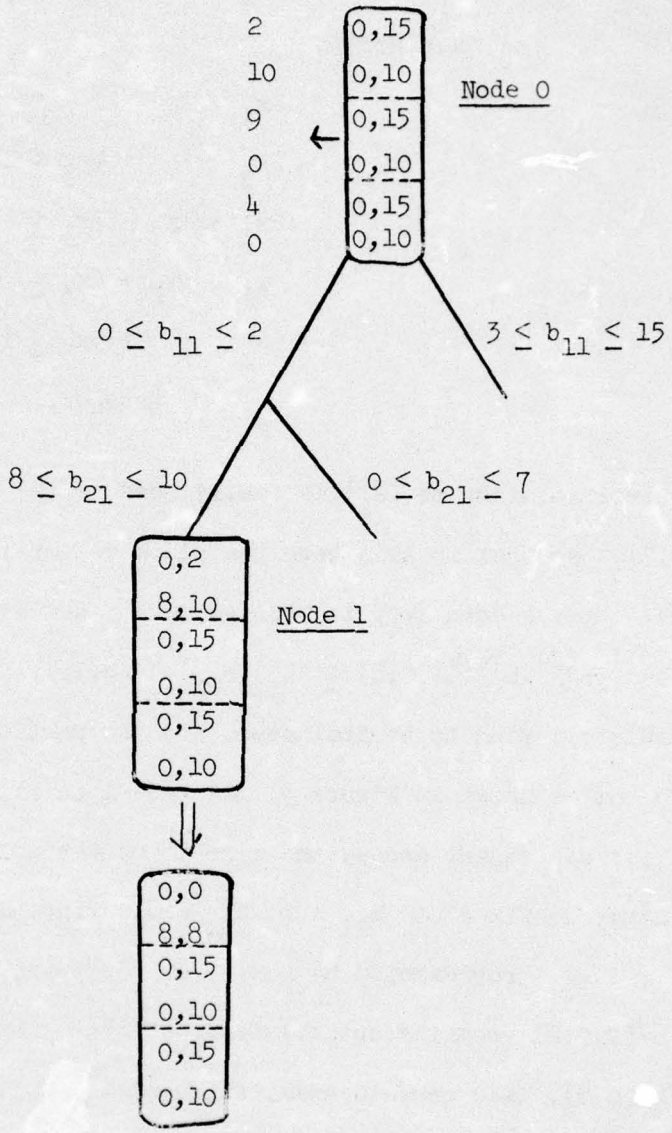

Figure 3: An example of the branching process

(S_{j_0}) which has just been solved using the allocation $\mathbf{b}^{(j_0)^*}$; for fixed $i = i_0$, F_{n_1}, say, corresponds to the set of allocations with $b_{i_0 j_0} \geq b^*_{i_0 j_0} + 1$, and F_{n_2} corresponds to the set of allocations with

$b_{i_0j_0} \leq b'_{i_0j_0} - 1 = b^*_{i_0j_0} - s^*_{i_0j_0} - 1$. However, if $b^*_{i_0j_0} = \bar{b}_{i_0j_0}$ (where $\bar{b}_{i_0j_0}$ represents the upper bound on $b_{i_0j_0}$ in the initial set F_n), the set F_{n_1} is obviously empty, since in this case $b_{i_0j_0}$ would be required to satisfy $b^*_{i_0j_0} + 1 \leq b_{i_0j_0} \leq \bar{b}_{i_0j_0} = b^*_{i_0j_0}$. Similarly, if $b'_{i_0j_0} \equiv b^*_{i_0j_0} - s^*_{i_0j_0} = \underline{b}_{i_0j_0}$, the set F_{n_2} is empty. In either case, the corresponding branch in Figure 2 is omitted. In terms of the example, note that neither the branch $b_{11} \leq b^*_{11} - s^*_{11} - 1$ $(= -1)$ nor the branch $b_{21} \geq b^*_{21} + 1$ $(= 11)$ is included in Figure 3.

It should be noted that the allocation $(b^{(1)'}, \ldots, b^{(N)'})$ defined by (1) is used only to facilitate the proof of Theorem 1. In general, the slacks $s^{(j_0)*}$ may be re-allocated among <u>several</u> other subproblems in a more promising manner than in (1). This point is developed in Chapters 4 and 5.

In any event, the definition of the allocation $(b^{(1)'}, \ldots, b^{(N)'})$ in Theorem 1 presupposes the existence of another subproblem (S_{j_1}) to which the unused resources $s^{(j_0)*}$ may be re-allocated. By the time the subproblem (S_N) is solved, however, the allocations to the previous N-1 subproblems have already been fixed in accordance with the branching procedure described above, and cannot be revised. This difficulty is resolved by noting that once the allocations $b^{(j)}$, $j = 1, 2, \ldots, N-1$ are fixed, there is only one feasible allocation of interest in (S_N), namely,

$$b^{(N)} = b - \sum_{j=1}^{N-1} b^{(j)} . \qquad (9)$$

Any allocation which uses less than all of the remaining linking resources

in (S_N) is dominated by the allocation in (9). Consequently, no branching is required following the solution of the N^{th} subproblem in Step 4 of the RBBA. Step 5 is bypassed, and, jumping to Step 2 instead, a new subset F_k is selected for examination next.

The advantages of adopting the branching procedure described in Section 3.2 are several. Firstly, the procedure enables the search tree to be proved substantially (and quickly) by eliminating searches over the dominated sets S as defined in (3) or (4). This, of course, has been the primary motivating force in its development. Secondly, the procedure provides an easy decision rule for use in Step 5 in the RBBA presented in Chapter 2. The choice of the branching variables and the specification of the bounds used in the partitioning process result directly from the subproblem solutions obtained in Step 4, with only minor additional computational effort required. Thirdly, in general only one subproblem (S_j) will be solved in Step 4 in the RBBA before the given allocation is recognized to be dominated. Jumping then to Step 5 and branching as described above, the unused resources $s^{(j)*}$ discovered in solving (S_j) can be reallocated to the remaining subproblems, thereby possibly yielding an improved allocation overall. Finally, the branching procedure allows the objective function bounds computed at subsequent nodes to be tightened, and thereby can aid in fathoming in the RBBA. Since in the resultant node n+1 in Figure 2 the allocation to subproblem (S_{j_0}) is fixed at $b^{(j_0)} = b^{(j_0)'} \equiv b^{(j_0)*} - s^{(j_0)*}$, the exact (integer programming) objective value obtained in (S_{j_0}) can be used when computing the upper bound U on the objective function value in (P) in Step 3. This of course holds for all nodes descendant from

node $n+1$ as well, since the allocation to the j_0^{th} subproblem remains fixed at these nodes. These last remarks are clarified in Chapters 4 and 5, where the bounding procedures are discussed.

CHAPTER 4

COMPUTING BOUNDS: SINGLE LINKING CONSTRAINT CASE

In this chapter we are concerned with Step 3 of the RBBA presented in Chapter 2, i.e., the problem of computing an upper bound U on the maximal objective value attainable in (P) under any allocation $(b^{(1)}, \ldots, b^{(N)})$ in a given subset F_n of feasible allocations. To this end, a Master Problem associated with (P) is introduced in Section 4.1. The solution to this problem provides the desired bound U. In addition, in the case that the upper bound U does not fathom F_n, the solution of the Master Problem generates a promising allocation - as required in the second part of Step 3 - which may be selected next for further examination in Step 4.

When the original block angular problem (P) has only a single linking constraint (as might be the case, for example, for a multidivisional corporation in which the subdivisions are tied together only by a corporate-wide capital budgeting limitation), the Master Problem itself takes on a special structure which gives rise to a specialized and very efficient solution procedure.[1/] For this reason, the discussion of the bounding procedure in this chapter is devoted entirely to the single linking constraint case. Although the fundamental concepts involved in generating the objective function bounds and trial allocations in the multiple linking constraints case are quite similar to those developed

---

[1/] In fact, in this case the Master Problem is seen to belong to the broader class of concave-separable integer programs discussed in Kochman (1976).

here, discussion of the details of the bounding procedure when there is more than one linking constraint is deferred to Chapter 5.

4.1 The Master Problem

The transformed block angular problem (P') given by (2.7)-(2.10) might itself be rewritten in the form,

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{N} f_j(b^{(j)}) , \\ \text{subject to} \quad & \\ & \sum_{j=1}^{N} b^{(j)} \leq b \qquad (1) \\ & \underline{b}^{(j)} \leq b^{(j)} \leq \overline{b}^{(j)}, \; b^{(j)} \text{ integer for } j = 1,2,\ldots,N , \end{aligned}$$

where for each $j = 1, 2, \ldots, N$, $\underline{b}^{(j)} = (\underline{b}_{1j}, \ldots, \underline{b}_{mj})$, $\overline{b}^{(j)} = (\overline{b}_{1j}, \ldots, \overline{b}_{mj})$, and $f_j(b^{(j)})$ is the maximal objective function value in subproblem (S_j), as a function of the vector $b^{(j)}$ of linking resources allocated to (S_j). That is,

$$\begin{aligned} f_j(b^{(j)}) = \quad \max \quad & c^{(j)}x^{(j)} , \\ \text{subject to} \quad & \\ & A^{(j)}x^{(j)} \leq b^{(j)} \qquad (2) \\ & B^{(j)}x^{(j)} \leq \beta^{(j)} \\ & x^{(j)} \geq 0, \text{ integer} , \end{aligned}$$

for $j = 1, 2, \ldots, N$.

From the definition of the set F_0 of feasible allocations in (2.16), it is apparent that problem (1) can be more compactly expressed as

$$\underset{(b^{(1)},\ldots,b^{(N)})\in F_0}{\text{maximize}} \quad \sum_{j=1}^{N} f_j(b^{(j)}) \ . \tag{3}$$

Moreover, as discussed in Section 3.1, the subsets F_n of F_0 generated in the course of the RBBA differ from each other only in the values of the upper and lower bounds $\bar{b}_{ij}$, $\underline{b}_{ij}$ on the allocation variables b_{ij}, $1 \leq i \leq m$, $1 \leq j \leq N$. Consequently, the problem of determining the maximum objective value attainable in (P) by any allocation in a given subset F_n can be represented by

$$\underset{(b^{(1)},\ldots,b^{(N)})\in F_n}{\text{maximize}} \quad \sum_{j=1}^{N} f_j(b^{(j)}) \tag{4}$$

by revising the bounds $\underline{b}^{(j)}$, $\bar{b}^{(j)}$, $j = 1, 2, \ldots, N$, in (1) appropriately.

In order to derive an upper bound U on the optimal objective value in (4), consider the relaxation of (4) obtained by replacing the objective function components $f_j(b_j)$,[1/] $j = 1, 2, \ldots, N$, with their linear-programming approximations. That is, for $j = 1, 2, \ldots, N$, let

$$\begin{aligned} z_j(b_j) \equiv \quad \max \quad & c^{(j)}x^{(j)} , \\ \text{subject to} \quad & \\ & A^{(j)}x^{(j)} \leq b_j \\ & B^{(j)}x^{(j)} \leq \beta^{(j)} \\ & x^{(j)} \geq 0 , \end{aligned} \tag{5}$$

[1/] When the block angular problem (P) has only a single linking constraint, the matrices $A^{(j)}$ contain only one row, and the vector of allocation variables $b^{(j)}$ reduces to a scalar. In identifying the individual allocation variables b_{ij}, we therefore suppress the first subscript, and write $b^{(j)} = (b_{ij}) \equiv b_j$ in the following discussion.

and consider the problem,

$$\text{(MP)} \qquad \text{maximize} \quad \sum_{j=1}^{N} z_j(b_j) \ ,$$

subject to

$$\sum_{j=1}^{N} b_j \leq b$$

$$\underline{b}_j \leq b_j \leq \bar{b}_j, \ b_j \ \text{integer, for} \ \ j = 1,2,\ldots,N \ .$$

Insofar as problem (MP) serves to coordinate the activities of the subproblems, we refer to (MP) as the Master Problem, borrowing loosely from the terminology of Dantzig and Wolfe (1960). It is clear, since (5) is a relaxation of (2), that $z_j(b_j) \geq f_j(b_j)$ for all $\underline{b}_j \leq b_j \leq \bar{b}_j$, $j = 1, 2, \ldots, N$, and so the maximal objective value in (MP) indeed generates an upper bound for the objective value in (4).

4.2 Solving the Master Problem

In the presence of a single linking constraint, problem (5) may be considered as a parametric linear program, and can be solved by standard linear programming techniques. It is well-known that the resulting functions $z_j(\cdot)$ defined in (5) will be concave and piecewise linear; that is, each $z_j(b_j)$, $j = 1, 2, \ldots, N$, has the form shown in Figure 1. As such, and because of the integer restrictions on the allocation variables b_j, $j = 1, 2, \ldots, N$, (MP) is a concave separable integer program. This class of problems is defined and discussed in greater generality in Kochman (1976). The algorithm developed in this section for solving the problem (MP) is a specialized version of the procedures presented there.

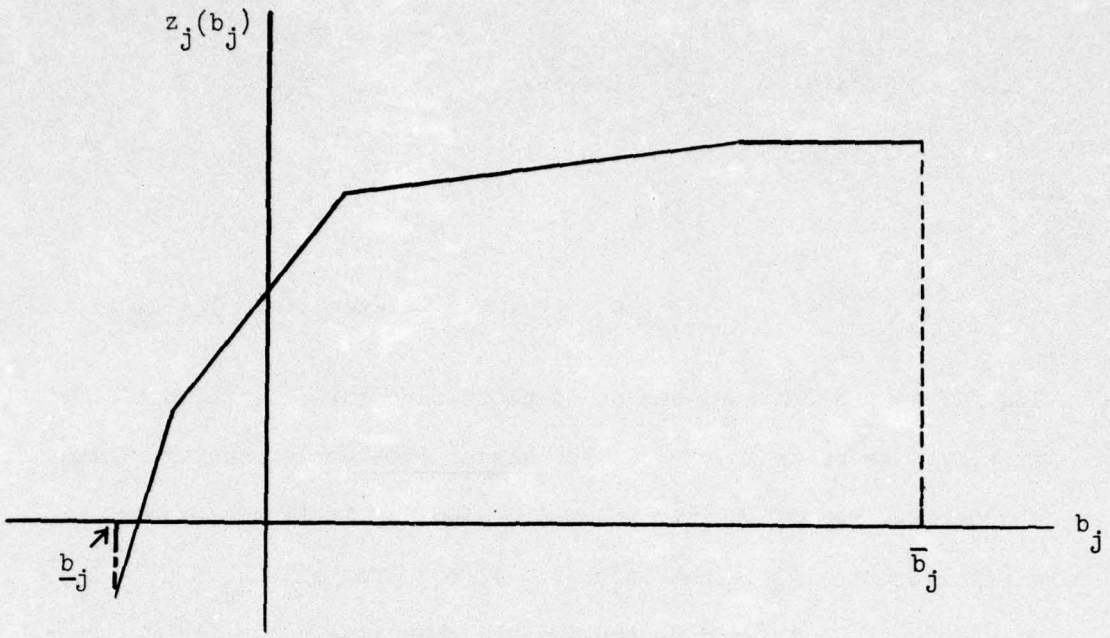

Figure 1: Form of the concave and piecewise linear function $z_j(b_j)$

Ignoring the integer restrictions on the b_j, $j = 1, 2, \ldots, N$, for the moment, (MP) becomes a concave-separable linear program in bounded variables. As shown in Figure 2, let p_{jk} denote the k^{th} breakpoint of $z_j(b_j)$, including the lower and upper bounds $\underline{b}_j$, $\overline{b}_j$. Let K_j denote the number of such breakpoints, so that

$$p_{j1} = \underline{b}_j, \quad \text{and} \quad p_{jK_j} = \overline{b}_j \, . \tag{6}$$

Without loss of generality, we may assume that p_{j1}, p_{jK_j} are integer-valued.

Let s_{jk} $(k = 1, 2, \ldots, K_j - 1)$ denote the slope of the k^{th} segment of $z_j(b_j)$, and α_{jk} the intercept. Then, for $j = 1,2,\ldots,N$, $z_j(b_j)$ may be written as

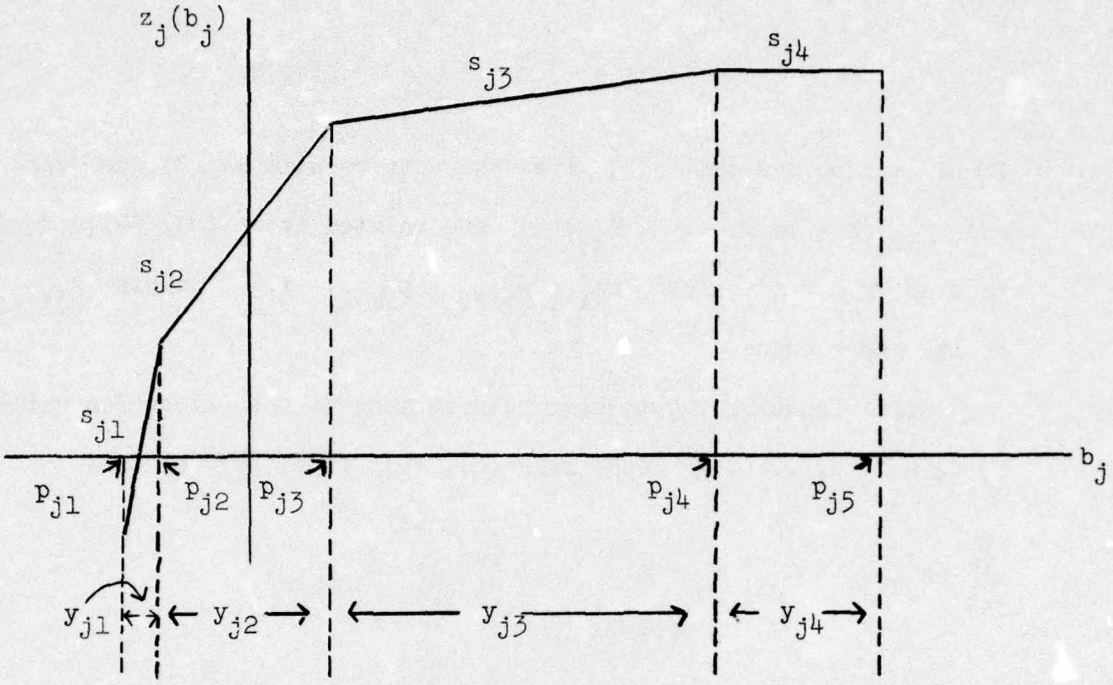

Figure 2: Breakpoints p_{jk} and auxiliary variables y_{jk} associated with $z_j(b_j)$

$$z_j(b_j) = s_{jk}b_j + \alpha_{jk}, \quad \text{for} \quad p_{jk} \leq b_j \leq p_{jk+1}, \quad k = 1,2,\ldots,K_j-1 \ . \tag{7}$$

To linearize (MP), the auxiliary variables y_{jk} ($k = 1, 2, \ldots, K_j-1$), $j = 1, 2, \ldots, N$) are introduced as indicated in Figure 2. We can then write

$$b_j = p_{j1} + \sum_{k=1}^{K_j-1} y_{jk}, \qquad j = 1, 2, \ldots, N, \tag{8}$$

where each y_{jk} is bounded by

$$0 \leq y_{jk} \leq p_{jk+1} - p_{jk}, \qquad k = 1,2,\ldots,K_j, \quad j = 1,2,\ldots,N \ . \tag{9}$$

Also, for $j = 1, 2, \ldots, N$,

$$z_j = s_{j1}p_{j1} + \alpha_{j1} + \sum_{k=1}^{K_j-1} s_{jk}y_{jk} . \qquad (10)$$

It is easy to see that (10) gives the same results as (7) whenever b_j and y_{jk} $(k = 1, 2, \ldots, K_j - 1)$ are related as in (8), (9), provided that no $y_{jk} > 0$ unless $y_{jk-1} = p_{jk} - p_{jk-1}$, i.e., unless y_{jk-1} is at its upper bound.

Still ignoring the integer restrictions on the allocation variables b_j $(j = 1, 2, \ldots, N)$ and using (8), (9), (10), (MP) becomes

$$\sum_{j=1}^{N} (s_{j1}p_{j1} + \alpha_{j1}) + \max \sum_{j=1}^{N} \sum_{k=1}^{K_j-1} s_{jk}y_{jk} ,$$

subject to

$$\sum_{j=1}^{N} \sum_{k=1}^{K_j-1} y_{jk} = b - \sum_{j=1}^{N} p_{j1} \qquad (11)$$

$$0 \leq y_{jk} \leq p_{jk+1} - p_{jk}, \; j = 1,2,\ldots,N, \; k = 1,2,\ldots,K_j-1 .$$

It is a standard result of the theory of concave-separable linear programs that the optimal solution to (11) will naturally satisfy the condition that no $y_{jk} > 0$ unless $y_{jk-1} = p_{jk} - p_{jk-1}$. Thus, (11) is equivalent to the LP relaxation of the Master Problem (MP).

Note that (11) itself is a linear programming, bounded-variable knapsack problem, and so its solution is almost trivial. The general LP bounded-variable knapsack problem is of the form,

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j \ ,$$

$$\text{subject to}$$

$$\sum_{j=1}^{n} a_j x_j \leq b \tag{12}$$

$$0 \leq x_j \leq \bar{x}_j \ ,$$

where, without loss of generality, $c_j > 0$, $a_j > 0$, $j = 1,2,\ldots,n$.[1/]

The solution to (12) is obtained by first computing the ratios c_j/a_j, $j = 1, 2, \ldots, n$, and reordering the variables x_j so that

$$c_{j_1}/a_{j_1} \geq c_{j_2}/a_{j_2} \geq \cdots \geq c_{j_n}/a_{j_n} \ . \tag{13}$$

Then, starting with x_{j_1} and proceeding sequentially to x_{j_2}, x_{j_3}, etc., each variable x_{j_i} is set to its upper bound $\bar{x}_{j_i}$ until, for some index p,

$$\sum_{i=1}^{p-1} a_{j_i} \bar{x}_{j_i} \leq b, \quad \text{but} \quad \sum_{i=1}^{p} a_{j_i} \bar{x}_{j_i} > b \ . \tag{14}$$

Next, x_{j_p} is set to $(b - \sum_{i=1}^{p-1} a_{j_i} x_{j_i})/a_{j_p}$, and x_{j_i} is set to zero for $i = p+1, p+2, \ldots, n$. Thus, if p is that index for which (14) holds, the optimal solution to (12) is

---

[1/] If any $c_k > 0$ and $a_k \leq 0$, it is always optimal to set $x_k = \bar{x}_k$, thereby eliminating x_k from the problem. Similarly, if any $c_k \leq 0$ and $a_k \geq 0$, set $x_k = 0$. Finally, if any $c_k \leq 0$ and $a_k < 0$, make the transformation $x_k' = \bar{x}_k - x_k$. Substituting x_k' for x_k in (12) then renders $c_k' \geq 0$, $a_k' > 0$.

$$x_{j_i} = \begin{cases} \bar{x}_{j_i} & , \quad 1 \le i \le p-1 \\ \dfrac{1}{a_{j_p}} \left(b - \sum_{i=1}^{p-1} a_{j_i} \bar{x}_{j_i} \right), & i = p \\ 0 & , \quad p+1 \le i \le n \, , \end{cases} \tag{15}$$

where the indexing sequence j_i is defined by (13).

Problem (11) is a special case of problem (12) in which each constraint coefficient is identically equal to 1. Thus, the sequence j_i in (13) in this case can be obtained simply by reordering the s_{jk}, $1 \le j \le N$, $1 \le k \le K_j - 1$, in non-increasing order since no divisions are necessary.

If (11) is solved by the procedure outlined above, it is clear from (15) that at most <u>one</u> y_{jk}, say y_{rp}, will not be at its upper or lower bound in the optimal solution $y^* = (y^*_{jk})$, $1 \le j \le N$, $1 \le k \le K_j - 1$. Hence, if the upper bound (9) on each y_{jk} is integer, each y^*_{jk} $(jk \neq rp)$ will be integer. But then the single functional constraint of (11) shows that

$$y^*_{rp} = b - \sum_{j=1}^{N} p_{j1} - \sum_{\substack{j=1 \\ jk \neq rp}}^{N} \sum_{k=1}^{K_j - 1} y^*_{jk} = \text{integer} \, .$$

Therefore, when each upper bound $p_{jk+1} - p_{jk}$ in (9) is integral, it follows that there exists an optimal LP solution to (11) which is naturally integer. By the equivalence between (11) and the Master Problem (MP), this solution yields the optimal LP solution to (MP) via (8). Furthermore, from (8) and the fact that each y_{jk}, $j = 1,2,\ldots,N$, $k = 1, 2, \ldots, K_j - 1$, is naturally integer in the optimal solution to (11), it follows that each b_j, $j = 1, 2, \ldots, N$, will also be

integer-valued at the LP optimum to (MP). The discussion above has thus established

Theorem 1: Whenever $p_{jk+1} - p_{jk}$ is integral for all $j = 1, 2, \ldots, N$, $k = 1, 2, \ldots, K_j - 1$, the optimal integer solution to the Master Problem (MP) can be obtained from the optimal linear programming solution to the bounded-variable knapsack problem (11).

We next show how to modify the parametric objective functions $z_j(b_j)$, $j = 1, 2, \ldots, N$, in such a way as to obtain an equivalent problem (MP') to (MP) in which all the breakpoints p_{jk} are integral. This condition obviously implies that $p_{jk+1} - p_{jk}$ will be integral for all j, k, and so the solution procedure described above may be applied to (MP').

The procedure for modifying $z_j(b_j)$ is the same for each $j = 1, 2, \ldots, N$. Hence, consider $z_j(b_j)$ for some arbitrary but fixed j. Recalling that, without loss of generality, p_{j1} and p_{jK_j} may be assumed to be integral, let $p_{j\ell}$ $(2 \leq \ell \leq K_j - 1)$ be any non-integer breakpoint of $z_j(\cdot)$. We seek to remove $p_{j\ell}$ as one of the breakpoints of $z_j(\cdot)$ by creating two new breakpoints at $[p_{j\ell}] \equiv$ greatest integer $\leq p_{j\ell}$, and $< p_{j\ell} >$ = least integer $\geq p_{j\ell}$. This is accomplished by replacing $z_j(b_j)$ for $[p_{j\ell}] \leq b_j \leq < p_{j\ell} >$ with the straight line segment from the point $([p_{j\ell}], z_j([p_{j\ell}]))$ to the point $(< p_{j\ell} >, z_j(< p_{j\ell} >))$, as shown in Figure 3. It is easy to see that this modification preserves the concavity of $z_j(\cdot)$. Equally important (for reasons that will be seen shortly), the value of $z_j(b_j)$ is changed under the modification procedure only between integer values of b_j.

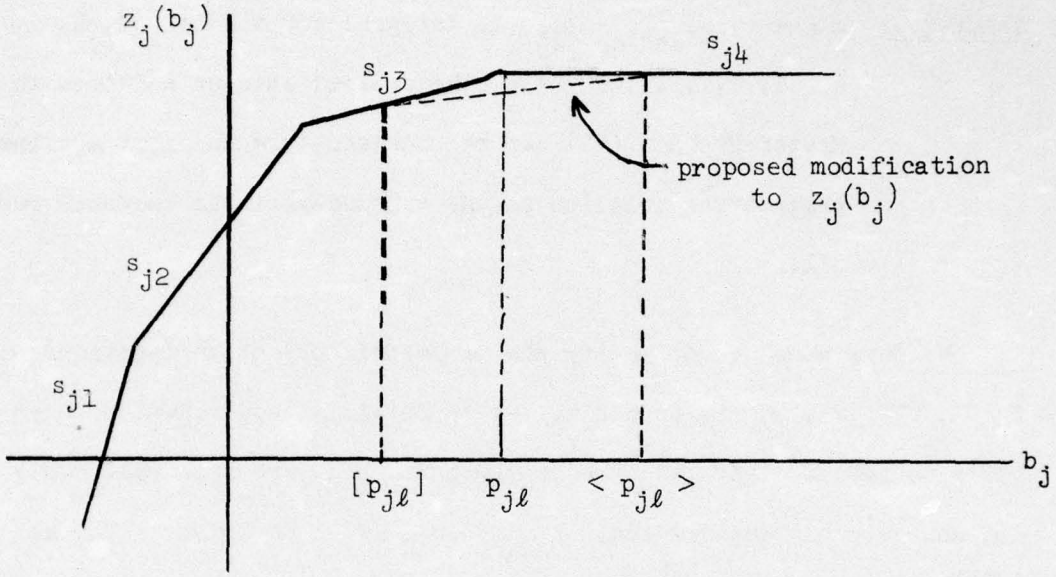

Figure 3: Modification to $z_j(b_j)$ to eliminate non-integer breakpoints

The slope s and intercept α of the modified segment of $z_j(\cdot)$ are easily computed:

$$s = \frac{z_j(< p_{j\ell} >) - z_j([p_{j\ell}])}{< p_{j\ell} > - [p_{j\ell}]} = z_j(< p_{j\ell} >) - z_j([p_{j\ell}]) , \qquad (16)$$

since $< p_{j\ell} > - [p_{j\ell}] = 1$ for non-integer $p_{j\ell}$. Also, since the value of $z_j(b_j)$ at $b_j = [p_{j\ell}]$ is unchanged,

$$\alpha = z_j([p_{j\ell}]) - s[p_{j\ell}] . \qquad (17)$$

Letting $z_j'(b_j)$ denote the function which results from modifying $z_j(b_j)$ as in Figure 3, we have, specifically,

$$z_j'(b_j) = \begin{cases} z_j(b_j), & \text{for } p_{j1} \leq b_j \leq [p_{j\ell}] \\ sb_j + \alpha, & \text{for } [p_{j\ell}] \leq b_j \leq < p_{j\ell} > \\ z_j(b_j), & \text{for } < p_{j\ell} > \leq b_j \leq p_{jK_j}, \end{cases} \tag{18}$$

where $z_j(b_j)$ is given by (7).

The modified function $z_j'(b_j)$ does not have a breakpoint at $b_j = p_{j\ell}$. Instead, the integer points $b_j = [p_{j\ell}]$ and $b_j = < p_{j\ell} >$ are breakpoints of $z_j'(b_j)$, while they may not have been breakpoints of the original function $z_j(b_j)$. Usually, therefore, the modification procedure results in increasing the total number of breakpoints by one for each non-integer breakpoint of $z_j(b_j)$. If, however, $[p_{j\ell}]$ or $< p_{j\ell} >$ or both are natural breakpoints for $z_j(b_j)$, or if the interval $[p_{j\ell}] \leq b_j \leq < p_{j\ell} >$ contains more than one breakpoint of $z_j(b_j)$ (so that all are eliminated simultaneously by the same segment), then the number of breakpoints can remain the same or even decrease following the modification.

The modification procedure is the same for each non-integer breakpoint $p_{j\ell}$ of $z_j(b_j)$. The resulting function $z_j'(b_j)$ is concave, piecewise linear, and has all-integer breakpoints. Repeating the procedure for each $j = 1, 2, \ldots, N$ leads to the problem,

$$\text{(MP')} \qquad \text{maximize} \quad \sum_{j=1}^{N} z_j'(b_j),$$

$$\text{subject to} \quad \sum_{j=1}^{N} b_j \leq b$$

$$\underline{b}_j \leq b_j \leq \bar{b}_j, \; b_j \text{ integer}, \; j = 1,2,\ldots,N.$$

Since only integer solutions to (MP) are of interest, and $z_j'(b_j) = z_j(b_j)$ at all integer values of b_j, $\underline{b}_j \leq b_j \leq \overline{b}_j$, $j = 1,2,\ldots,N$, it is clear that (MP') and (MP) must have the same optimal solutions. We may therefore solve (MP') in place of (MP). However, since all breakpoints of $z_j'(b_j)$, $j = 1, 2, \ldots, N$, are integer, it follows by the arguments above that the integer restrictions on b_j, $j = 1, 2, \ldots, N$, in (MP') are redundant and may be dropped. Moreover, since the $z_j'(b_j)$, $j = 1, 2, \ldots, N$, remain concave and piecewise linear under the modification procedure, (MP') is a concave separable, bounded-variable LP knapsack problem, and may be efficiently solved using the procedure described above.

In practice, it is not necessary to introduce the auxiliary variables y_{jk} explicitly in solving (MP'). Because each constraint coefficient in (MP') is unity, the LP solution procedure for problem (11) reduces simply to sequentially choosing the largest (remaining) s_{jk} and setting the corresponding variable y_{jk} to its upper bound. But since each $z_j'(b_j)$ is concave, $j = 1, 2, \ldots, N$, we have for each j that

$$s_{jk} \geq s_{jk+1}, \qquad k = 1, 2, \ldots, K_j - 2 . \tag{14}$$

Thus, in terms of the original variables b_j, $j = 1, 2, \ldots, N$, the solution procedure amounts to searching over j for that (remaining) segment k of $z_j(\cdot)$ with largest slope s_{jk}, and increasing the corresponding variable b_j from p_{jk} to p_{jk+1}. That is, b_j is increased to its next larger (integer) breakpoint value. When such an increase would result in making $\sum_{j=1}^{N} b_j > b$, the chosen variable b_r is set to $b_r = b - \sum_{j \neq r} b_j$ instead, and the procedure is terminated.

The procedure for solving the Master Problem (MP) in the case that the original block angular problem (P) has only a single linking constraint is now presented in algorithmic form.

Algorithm for Solving (MP): Single Linking Constraint

Step 0. For $j = 1, 2, \ldots, N$, solve subproblem (S_j) parametrically in b_j to obtain the functions $z_j(b_j)$. Modify $z_j(b_j)$ as discussed above to eliminate all non-integer breakpoints of $z_j(b_j)$.

Step 1. (Initialization.) For $j = 1, 2, \ldots, N$:

Set $b_j^* = p_{j1} = \underline{b}_j$.

Set $k_j = 1$.

Step 2. Find an r $(1 \leq r \leq N)$ such that $b_r^* < \overline{b}_r$ and $s_{rk_r} \geq s_{jk_j}$ $\forall j = 1, 2, \ldots, N$ such that $b_j^* < \overline{b}_j$.

Step 3. Reset $k_r = k_r + 1$.

Then, reset $b_r^* = p_{rk_r}$.

Step 4. (Check for termination.)

If $\sum_{j=1}^{N} b_j^* \geq b$, reset $b_r^* = b - \sum_{j \neq r} b_j^*$ and stop

$(b_j^*, j = 1, 2, \ldots, N,$ is optimal for (MP)).

Otherwise, go to Step 2.

In Step 1 of the algorithm, each variable b_j is initially set to its lower bound $\underline{b}_j$. The k_j $(j = 1, 2, \ldots, N)$ are used to index the segment of $z_j(b_j)$ which is currently of interest. In the search

for the largest (remaining) s_{jk} in Step 2, we know by (19) that if $b_j^* = p_{jk_j}$, $j = 1, 2, \ldots, N$, then only the slope s_{jk_j} (and not $s_{jk_j+1}, s_{jk_j+2}, \ldots, s_{jK_j-1}$) need be considered for each j. The index k_r is updated appropriately in Step 3.

In Step 2, only those j $(1 \leq j \leq N)$ are considered for which $b_j < \bar{b}_j$, since any b_j which is already equal to its upper bound obviously cannot be feasibly increased further. As discussed above, the variable b_r which is chosen to be increased next is reset to its next larger breakpoint value p_{rk_r}. (Note that k_r is updated first in Step 3.) The breakpoints p_{rk_r} used in Step 3 refer of course to the (all-integer) breakpoints of the modified functions $z_j'(b_j)$, $j = 1, 2, \ldots, N$.

The subsets F_n which are to be searched differ from one another in the values of the bounds $\underline{b}_j$, $\bar{b}_j$, $j = 1, 2, \ldots, N$, on the allocation variables. After the initial set F_0, therefore, we may not have $\underline{b}_j = p_{j1}$ for all $j = 1, 2, \ldots, N$ in each subset F_n, and hence some care must be exercised in initializing b_j^* and, correspondingly, the index k_j, $j = 1, 2, \ldots, N$, in Step 1. Given that this is properly accomplished, however, the algorithm above can be used to resolve (MP) at each node n in the search tree.

4.3 Using the Master Problem in the RBBA

The bounding step (Step 3) in the Rudimentary Branch-and-Bound Algorithm now can be described in greater detail. Recall that the objective in Step 3 of the RBBA is to determine whether the current set F_n can possibly contain any allocations which lead to a better

overall solution to (P) than the incumbent, and if so, to select one such allocation for further examination. This objective is accomplished by constructing and solving the Master Problem as discussed in Sections 4.1 and 4.2.

Suppose at a given node n in the search tree, the subset F_n of feasible allocation is to be considered next. Using the algorithm of Section 4.2, the optimal (integer) solution b_j^*, $j = 1, 2, \ldots, N$, to (MP) is easily obtained.[1/] The optimal objective value,

$$U = \sum_{j=1}^{N} z_j(b_j^*) , \tag{20}$$

provides an upper bound on the maximal objective value in problem (4). If $U \leq z^*$, where z^* is the current incumbent objective value, the set F_n is fathomed. If $U > z^*$, however, the allocation $b_j = b_j^*$ $(j = 1, 2, \ldots, N)$ immediately presents itself as a natural choice for the next allocation to be considered in Step 4 of the RBBA.

Consider again now the branching procedure presented in Chapter 3. Suppose that when subproblem (S_{j_0}) is solved in Step 4 of the RBBA, it is discovered that the allocation $b_j = b_j^*$ $(j = 1, 2, \ldots, N)$ is dominated. As discussed in Chapter 3, this is the case whenever the optimal integer solution $x^{(j_0)*}$ in (S_{j_0}) uses an amount $b'_{j_0} \equiv A^{(j_0)} x^{(j_0)*} < b_{j_0}^*$ of the linking resource. After branching on b_{j_0} in Step 5 of the RBBA,

---

[1/] This solution is optimal for (MP) in the sense that it generates the maximal objective value which (MP) can attain over the set F_n. Note, however, that the allocation $b_j = b_j^*$, $j = 1, 2, \ldots, N$, still may not be optimal for (P); that is, it is possible that $v(P|(b_1^*, \ldots, b_N^*)) < v(P|(b_1', \ldots, b_N'))$ for some other allocation $(b_1', \ldots, b_N') \in F_n$, even though $v(MP|(b_1^*, \ldots, b_N^*)) > v(MP|(b_1', \ldots, b_N'))$.

the allocation in subproblem (S_{j_0}) is "fixed" at the level $b_{j_0} = b'_{j_0}$ at the next entry to Step 3. Since each of the slopes s_{jk} and breakpoints p_{jk} of $z'_j(b_j)$ is unchanged, it is clear that in resolving (MP) now, the variables b_r chosen to be increased in Step 2 of the algorithm above will be selected in the same order (and increased by the same amounts in Step 3) as before, with the exception that this time the allocation to (S_{j_0}) cannot be raised above $b_{j_0} = b'_{j_0}$. Consequently, if the previously optimal allocation was saved, the Master Problem need not be resolved from scratch. Instead, starting with the $b_j = b^*_j$, $j \neq j_0$, and $b_{j_0} = b'_{j_0}$ (fixed), (MP) can be efficiently reoptimized simply by using the algorithm above to reallocate the "additional" units $b^*_{j_0} - b'_{j_0}$ of the linking resource among the other subproblems (S_j), $j \neq j_0$.

The fact that the allocations to certain subproblems are fixed as a result of the branching procedure also allows the upper bound U defined in (20) to be tightened. Consider an arbitrary node n in the search tree, and let

$$J_1 \equiv \{1 \leq j \leq N | \text{The allocation to } (S_j) \text{ is fixed in } F_n\} . \qquad (21)$$

Correspondingly, let

$$J_2 \equiv \{1, 2, \ldots, N\} - J_1 . \qquad (22)$$

The task of finding an optimal solution to the Master Problem at node n in effect reduces to that of optimally allocating the remaining amount, $b - \sum_{j \in J_1} b^*_j$, of the linking resource among the unfixed subproblems, where b^*_j $(j \in J_1)$ represents the level at which the allocation to (S_j)

is fixed. The bound (20) generated by the optimal solution $b_j = b_j^*$ $(j = 1, 2, \ldots, N)$ to (MP) at node n may therefore be replaced by the tighter bound,

$$U' = \sum_{j \in J_1} v(S_j | b_j^*) + \sum_{j \in J_2} z_j(b_j^*) , \qquad (23)$$

where, $v(S_j | b_j^*)$, $j \in J_1$, is the (known) optimal objective value in (S_j) under the allocation $b_j = b_j^*$. Since $U' \leq U$, and U' can be obtained at approximately the same computational expense as U, it is clearly preferable to use the bound U' in attempting to fathom F_n in Step 3 of the RBBA.

Finally, it should be noted that Step 0 in the algorithm for solving the Master Problem, included for the sake of completeness above, need only be executed once in the course of the RBBA. Specifically, prior to the start of the RBBA (as described in Chapter 2), the subproblems (S_j), $j = 1, 2, \ldots, N$, are solved as linear programs parametrically in b_j to obtain the functions $z_j(b_j)$. These functions are then modified in accordance with the procedure presented in Section 4.2. By storing the slopes s_{jk} and breakpoints p_{jk} of the revised functions $z_j'(b_j)$, it will not be necessary to recompute these values each time the Master Problem is solved.

CHAPTER 5

COMPUTING BOUNDS: MULTIPLE LINKING CONSTRAINTS CASE

The procedures presented in Chapter 4, while very efficient for computing the required objective function bounds in the presence of a single linking constraint, are inappropriate for this task when the block angular integer program (P) has more than one linking constraint. The major difficulty which arises with $m > 1$ linking constraints is that the parametric linear programming problems (4.5) now become the multi-parametric linear programs (MPLP),

$$\begin{aligned} z_j(b^{(j)}) \equiv \max \quad & c^{(j)}x^{(j)} , \\ \text{subject to} \quad & A^{(j)}x^{(j)} \leq b^{(j)} \\ & B^{(j)}x^{(j)} \leq \beta^{(j)} \\ & x^{(j)} \geq 0 , \end{aligned} \tag{1}$$

$j = 1, 2, \ldots, N$, where $b^{(j)} = (b_{1j}, \ldots, b_{mj})$ is now an m-vector of parameters. Although several multi-parametric linear programming algorithms have recently been proposed,[1/] in general, the MPLP is a much more difficult problem than the standard (single-parameter) parametric linear program. Consequently, direct methods for solving problems such as (1) are computationally too cumbersome for use within the present context.

[1/] See Gal and Nedoma (1972), or Van de Panne (1975). Duffin's (1974) modified Fourier-Motzkin elimination procedure can also be used for this purpose.

A second complication which arises in the presence of multiple linking constraints is in the structure of the Master Problem itself:

$$\text{maximize} \quad \sum_{j=1}^{N} z_j(b^{(j)}) ,$$

subject to

$$\sum_{j=1}^{N} b^{(j)} \leq b \tag{2}$$

$$\underline{b}^{(j)} \leq b^{(j)} \leq \bar{b}^{(j)}, \; b^{(j)} \text{ integer}, \quad j = 1,2,\ldots,N .$$

While it is again easy to see that the $z_j(b^{(j)})$ defined in (1) are concave and piecewise linear, each is now a function of several variables. Therefore, the objective function in problem (2) is not separable, and the solution procedure developed in Chapter 4 can no longer be used.

The bounding method proposed in this chapter for the multiple linking constraints case is nonetheless quite similar in spirit to the method discussed in Chapter 4. As before, a Master Problem of the form (2) is introduced, and its use in the RBBA is entirely analogous to the use of problem (MP) in the single linking constraint case. For the reasons mentioned above, however, the functions $z_j(b^{(j)})$ are not wholly computed in advance; rather, relevant "segments" of each $z_j(b^{(j)})$ are generated in the course of the algorithm. Moreover, because of the loss of separability, a different linearization procedure for the concave integer program (2) is required.

5.1 The Bounding Procedure

In order to compute an upper bound on the maximum objective value which (P) can attain over a given subset F_n of feasible allocations, we utilize the dual multipliers associated with the linking constraints

in the LP-optimal solutions for the subproblems.

Theorem 1: Let $\bar{z}_j$ denote the optimal LP objective value in the j^{th} subproblem of (P), given the allocation $b^{(j)^*} = (b^*_{1j},\ldots,b^*_{mj})$. Let λ_{ij} ($1 \leq i \leq m$) denote the LP-optimal dual multiplier associated with the constraint $\sum_{k=1}^{n_j} A^{(j)}_{ik} x^{(j)}_k \leq b^*_{ij}$. Then, for any other allocation

$$b'_{ij} \equiv b^*_{ij} + \Delta b_{ij}, \qquad 1 \leq i \leq m , \tag{3}$$

we have

$$v(S_j | b^{(j)'}) \leq \bar{z}_j + \sum_{i=1}^{m} \lambda_{ij} \Delta b_{ij} . \tag{4}$$

Proof: Consider the LP relaxation of the j^{th} subproblem under the allocation $b^{(j)^*}$. This may be written as

$$\begin{aligned} &\text{maximize} \quad c^{(j)} x^{(j)} , \\ &\text{subject to} \\ &\qquad M^{(j)} x^{(j)} \leq \rho^{(j)} \\ &\qquad x^{(j)} \geq 0 , \end{aligned} \tag{5}$$

where

$$M^{(j)} = \begin{pmatrix} A^{(j)} \\ B^{(j)} \end{pmatrix}, \quad \text{and} \quad \rho^{(j)} = \begin{pmatrix} b^*_{1j} \\ \vdots \\ b^*_{mj} \\ \beta^{(j)} \end{pmatrix} .$$

The dual of (5) is

$$\begin{aligned} &\text{minimize} \quad y^{(j)}\rho^{(j)} , \\ &\text{subject to} \\ &\qquad y^{(j)}M^{(j)} \geq c^{(j)} \\ &\qquad y^{(j)} \geq 0 \end{aligned} \tag{6}$$

Let $\bar{x}^{(j)}$ and $\bar{y}^{(j)}$ denote the optimal solution to (5) and (6) respectively. By strong duality,

$$\bar{z}_j = c^{(j)}\bar{x}^{(j)} = \bar{y}^{(j)}\rho^{(j)} \tag{7}$$

Now suppose that the right-hand side in (5) is changed to $\rho^{(j)}$, where

$$\rho^{(j)} = \rho^{(j)} + (\Delta b_{1j}, \ldots, \Delta b_{mj}, 0, \ldots, 0)^T , \tag{8}$$

and where Δb_{ij}, $1 \leq i \leq m$, is given by (3). The dual to this revised problem is

$$\begin{aligned} &\text{minimize} \quad y^{(j)}\rho^{(j)} , \\ &\text{subject to} \\ &\qquad y^{(j)}M^{(j)} \geq c^{(j)} \\ &\qquad y^{(j)} \geq 0 \end{aligned} \tag{9}$$

Then, since $\bar{y}^{(j)}$ is still feasible for (9), it follows by weak duality that $\bar{z}_j \leq \bar{y}^{(j)}\rho^{(j)}$, where $\bar{z}_j$ is the optimal objective function value in the revised version of problem (5). Thus,

$$\begin{aligned} \bar{z}_j &\leq \bar{y}^{(j)}\rho^{(j)} \\ &= \bar{y}^{(j)}\rho^{(j)} + \sum_{i=1}^{m} \bar{y}_i^{(j)}\Delta b_{ij}, \quad \text{from (8)} \\ &= \bar{z}_j + \sum_{i=1}^{m} \bar{y}_i^{(j)}\Delta b_{ij} , \quad \text{from (7)} \end{aligned}$$

Since $\bar{y}^{(j)}$ is the optimal solution to (6), the dual of (5), the first m components of $\bar{y}^{(j)}$ must be $\bar{y}_i^{(j)} = \lambda_{ij}$, $i = 1, 2, \ldots, m$. This gives

$$\bar{z}_j' \leq \bar{z}_j + \sum_{i=1}^{m} \lambda_{ij} \Delta b_{ij} ,$$

and (4) follows by noting that the optimal integer solution in the j^{th} subproblem (corresponding to the allocation $b^{(j)'}$) must yield an objective value less than or equal to $\bar{z}_j'$.

Q.E.D.

The upper bound on $v(S_j | b^{(j)'})$ given by (4) can be rewritten as follows.

$$\bar{z}_j + \sum_{i=1}^{m} \lambda_{ij} \Delta b_{ij} = \bar{y}^{(j)} \rho^{(j)} + \sum_{i=1}^{m} \lambda_{ij} (b'_{ij} - b^*_{ij}), \quad \text{from (7) and (3)}$$

$$= \sum_{i=1}^{m} \bar{y}_i^{(j)} b^*_{ij} + \sum_{i=1}^{m_j} \bar{y}_{m+i}^{(j)} \beta_i^{(j)}$$

$$+ \sum_{i=1}^{m} \bar{y}_i^{(j)} (b'_{ij} - b^*_{ij}), \quad \text{from the definition of } \rho^{(j)}$$

$$= \sum_{i=1}^{m_j} \bar{y}_{m+i}^{(j)} \beta_i^{(j)} + \sum_{i=1}^{m} \bar{y}_i^{(j)} b'_{ij} .$$

Now define

$$z_{0j} \equiv \sum_{i=1}^{m_j} \bar{y}_{m+i}^{(j)} \beta_i^{(j)} , \tag{10}$$

so that z_{0j} is independent of the allocation $b^{(j)'}$. Again noting that $\bar{y}_i^{(j)} = \lambda_{ij}$ for $1 \leq i \leq m$, we have from (4) and the last equation above that

$$v(S_j|b^{(j)'}) \leq z_{0j} + \sum_{i=1}^{m} \lambda_{ij} b'_{ij} \qquad (11)$$

Once the constant z_{0j} has been computed, the bound (11) is a function only of the allocation $b^{(j)'}$. Hence, in practice it will be more convenient to use (11) than (4). Since the allocation $b^{(j)'}$ in (11) is arbitrary, by summing over $j = 1, 2, \ldots, N$, the maximal objective value in (P) under any allocation $(b^{(1)}, \ldots, b^{(N)})$ can be bounded by[1/]

$$v(P|(b^{(1)}, \ldots, b^{(N)})) \leq \sum_{j=1}^{N} (z_{0j} + \sum_{i=1}^{m} \lambda_{ij} b_{ij}) . \qquad (12)$$

In general, different allocations $b^{(j)*}$ lead to different LP-optimal bases in (S_j), and, consequently, to different sets of optimal dual multipliers $\{\lambda_{ij}, 1 \leq i \leq m\}$. Each of those sets can be used to generate an upper bound on $v(S_j|b^{(j)})$ as in (11). Indexing the different sets by the superscript k, and letting L_j denote the total number of such sets associated with the subproblem (S_j), we have

$$v(S_j|b^{(j)}) \leq \min_{1 \leq k \leq L_j} (z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij}) , \qquad (13)$$

where $z_{0j}^{(k)}$ is computed from (10), for $k = 1, 2, \ldots, L_j$. Hence, defining $z_j(b^{(j)})$ by

$$z_j(b^{(j)}) = \min_{1 \leq k \leq L_j} (z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij}), \quad j = 1,2,\ldots,N , \qquad (14)$$

[1/] We note that, by the way in which (12) was obtained, the sets of dual multipliers $\{\lambda_{ij}, 1 \leq i \leq m\}$ need not be related to each other for different $j = 1,2,\ldots,N$. In particular, the allocations $b^{(j)*}$ used to obtain $\{\lambda_{ij}, 1 \leq i \leq m\}$, $j = 1,2,\ldots,N$, (as in Theorem 1) are not required to satisfy $\sum_{j=1}^{N} b^{(j)*} \leq b$.

leads to the Master Problem (2).

To convert the concave (non-separable) problem (2) into a standard mixed-integer linear programming format, the objective function components $z_j(b^{(j)})$ are replaced by the auxiliary variables z_j, $j = 1, 2, \ldots, N$. The z_j are not restricted in sign, but from (14), are required to satisfy

$$z_j \leq z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij} \quad \text{for each } k = 1, 2, \ldots, L_j:$$

$$\text{(MP)} \quad \text{maximize} \quad \sum_{j=1}^{N} z_j , \tag{15}$$

subject to

$$\sum_{j=1}^{N} b_{ij} \leq b_i \quad , \quad i=1,2,\ldots,m \tag{16}$$

$$z_j - \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij} \leq z_{0j}^{(k)}, \quad k=1,2,\ldots,L_j, \; j=1,2,\ldots,N \tag{17}$$

$$\underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij}, \; b_{ij} \text{ integer}, \quad i=1,2,\ldots,m \; , \; j=1,2,\ldots,N. \tag{18}$$

Since the objective (15) in (MP) is to maximize the sum of the auxiliary variables z_j, it is clear that in the optimal solution $(z_j^*, b^{(j)*})$, $j = 1, 2, \ldots, N$, the implied equality in (14) will be naturally satisfied; i.e., we will have

$$z_j^* = \min_{1 \leq k \leq L_j} (z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij}^*) \equiv z_j(b^{(j)*}), \qquad j = 1,2,\ldots,N \; ,$$

as desired. Thus, the Master Problem (MP) above is equivalent to problem (2) when the concave functions $z_j(b^{(j)})$ are as given by (14).

Each of the slopes s_{jk} of the different segments of $z_j(b_j)$ defined in Chapter 4 is exactly the dual multiplier associated with the (single) linking constraint in the basis that is optimal for (S_j) at $b_j = p_{jk}$.

Thus, there is a strong analogy between the Master Problem (MP) above and that introduced in Chapter 4 for the single linking constraint case. Each of the sets of multipliers $\{\lambda_{ij}^{(k)},\ 1 \leq i \leq m\}$, $k = 1, 2, \ldots, L_j$, defines a different segment (hyperplane) of the concave and piecewise linear function $z_j(b^{(j)})$. In fact, the task of explicitly computing the $z_j(b^{(j)})$ in (14) is tantamount to solving the multiparametric linear program (1). As has been previously noted, however, this can be a computationally unattractive approach for our purposes.

Rather than solve the MPLP (1) directly, therefore, we note that the necessary multipliers $\lambda_{ij}^{(k)}$ (and constants $z_{0j}^{(k)}$) can easily be obtained as a by-product of the subproblem solutions computed in the course of the RBBA. Specifically, suppose that at some point in the algorithm, ℓ_j distinct sets of multipliers $\{\lambda_{ij}^{(k)},\ 1 \leq i \leq m\}$ ($k = 1, 2, \ldots, \ell_j$) have been obtained in the j^{th} subproblem, where $1 \leq \ell_j \leq L_j$. A "relaxed" master problem of the form (MP) can be constructed in which some of the constraints (17) are omitted. The Relaxed Master Problem is

$$\text{(RMP)} \quad \text{maximize} \quad \sum_{j=1}^{N} z_j \,, \tag{19}$$

subject to

$$\sum_{j=1}^{N} b_{ij} \leq b_i \qquad , \quad i=1,2,\ldots,m \tag{20}$$

$$z_j - \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij} \leq z_{0j}^{(k)}, \quad k=1,2,\ldots,\ell_j,\ j=1,2,\ldots,N \tag{21}$$

$$\underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij},\ b_{ij} \text{ integer}, \quad i=1,2,\ldots,m, \quad j=1,2,\ldots,N\ . \tag{22}$$

In considering the set F_n of feasible allocations, the optimal solution $(z_j^*, b^{(j)^*})$, $j = 1, 2, \ldots, N$, to (RMP) provides a valid upper bound $U = \sum_{j=1}^{N} z_j^*$ on the maximal objective value which can be attained in (P) under any allocation in F_n. If F_n is not fathomed by U, the allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ may be used in solving the subproblems in Step 4 of the RBBA. If any linear programming based solution procedure is used (e.g., a branch-and-bound method such as was outlined in Section 1.2), it is well-known that the dual multipliers λ_{ij}, $1 \leq i \leq m$, appear in the optimal LP solution to (S_j) as the reduced cost coefficients associated with the linking constraint slack variables. Thus, while it may be advantageous to derive a few sets $\{\lambda_{ij}^{(k)}, 1 \leq i \leq m\}$ of multipliers initially in each subproblem, remaining sets can be obtained at little additional computational expense as the search proceeds. In this way, much of the work involved in explicitly computing the functions $z_j(b^{(j)})$ defined in (14) can be avoided.

One procedure by which suitable initial sets of multipliers $\{\lambda_{ij}^{(k)}, 1 \leq i \leq m\}$ may be determined is begun by specifying an arbitrary allocation $b^{(j)^*}$, $j = 1, 2, \ldots, N$, in each subproblem. These $b^{(j)^*}$ should be chosen to ensure that each of the subproblems (S_j) has an LP-optimal solution which is finite, but it is not necessary that the overall allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ itself be feasible; that is, it is not necessary that the constraint

$$\sum_{j=1}^{N} b^{(j)^*} \leq b$$

be satisfied. When each of the subproblems is solved as a linear program,

the resulting sets of dual-optimal multipliers can be used to construct a first relaxed master problem, say (RMP_1). This problem may in turn be solved to generate a second (feasible) allocation, thereby establishing an iterative routine. In particular, on the k^{th} iteration $(k = 1,2,\ldots)$, the subproblems are resolved as linear programs, using the optimal allocation from (RMP_k). Any new sets of multipliers discovered in the process are used to augment (RMP_k) to obtain the next relaxed master problem (RMP_{k+1}). The procedure terminates at iteration K when no new sets of multipliers result from the optimal allocation in (RMP_K). The final problem (RMP_K) then serves as the initial relaxed master problem in the RBBA.

In many practical situations, a good starting allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ may be readily available from previous experience with the problem. If no such allocation can be easily determined, however, one may begin by specifying $b^{(j)} = \bar{b}^{(j)}$, $j = 1, 2, \ldots, N$, where $\bar{b}^{(j)}$ is the initial vector of upper bounds on the linking resource allocations in the j^{th} subproblem. (Methods for computing $\bar{b}^{(j)}$ have been presented in Section 2.3.) If any subproblem (S_j) is then found to be infeasible, it is clear that the block angular integer program (P) must itself be infeasible.

Note that $\bar{z}_j$, the optimal LP objective value in (S_j) under the allocation $b^{(j)^*}$, must satisfy

$$\bar{z}_j \leq z_j^* = \min_{1 \leq k \leq \ell_j} \left(z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij}^* \right) . \tag{23}$$

If equality holds in (23), the bound z_j^* on $v(S_j | b^{(j)^*})$ obtained from (RMP) is as tight as that which would have been produced by solving

the full Master Problem (MP). If $\bar{z}_j < z_j^*$, however, it follows by strong duality that (adopting the notation of Theorem 1 again)

$$\bar{y}^{(j)}\rho^{(j)} < \min_{1 \leq k \leq \ell_j} (z_{0j}^{(k)} + \sum_{i=1}^{m} \lambda_{ij}^{(k)} b_{ij}^*) . \qquad (24)$$

The inequality (24) in turn implies that the set of dual multipliers $\{\lambda_{ij}^{(\ell_j+1)}, 1 \leq i \leq m\}$ obtained in solving (S_j) is distinct from any of the previous sets $\{\lambda_{ij}^{(k)}, 1 \leq i \leq m\}$, $k = 1, 2, \ldots, \ell_j$. Moreover, since

$$\bar{z}_j = \bar{y}^{(j)}\rho^{(j)} = z_{0j}^{(\ell_j+1)} + \sum_{i=1}^{m} \lambda_{ij}^{(\ell_j+1)} b_{ij}^* ,$$

this new set of multipliers gives rise to precisely that (binding) constraint of (MP) which was "missing" from (RMP). Thus, the new multipliers generated in the course of the RBBA are in some sense exactly (and only) those which are most relevant. By the same argument, it also seems reasonable to expect that the iterative procedure described above will indeed lead to suitable initial sets of multipliers for (RMP).

A second advantage in utilizing the relaxed master problem (RMP) rather than constructing the full master problem (MP) is that, in general, (RMP) has fewer constraints than does (MP). Practical experience has indicated that the computational effort required to solve linear programming problems increases approximately as the cube of the number of functional constraints. Since most integer programming algorithms are based on linear programming techniques, it is reasonable to expect that (RMP) will require less solution time than would (MP). Of course, this savings is obtained at the cost of possibly reduced fathoming capabilities

in the RBBA, since the objective function bound resulting from (RMP) may not be as tight as that provided by (MP). However, since most fathoming will tend to occur at later stages in the search, and since (RMP) may be augmented with the relevant additional constraints (17) as new sets of multipliers $\{\lambda_{ij}^{(k)},\ 1 \leq i \leq m\}$ are discovered in the course of the search, the loss may well be minimal.

The relaxed master problem (RMP) may be solved by any standard mixed-integer programming algorithm. Because the auxiliary variables z_j are not sign-restricted, however, the constraints (21) serve merely to supply upper bounds on the values which these variables may assume. In this sense, the only functional constraints which appear in (RMP) are the linking constraints (20). Consequently, integer-feasible solutions to (RMP) are easily obtained, and this fact can be exploited to improve the overall efficiency of solution.

In particular, consider the situation when an enumerative algorithm is used to solve (RMP). The integer restrictions on the allocation variables b_{ij} are relaxed, and (RMP) is first solved as a linear program. Let $(\tilde{z}_j, \tilde{b}^{(j)})$, $j = 1, 2, \ldots, N$, denote the optimal LP solution. If each $\tilde{b}_{ij}$, $1 \leq i \leq m$, $1 \leq j \leq N$, is integer-valued, $(\tilde{z}_j, \tilde{b}^{(j)})$ is optimal for (RMP), and we are done. Otherwise, however, let $f_{ij} = \tilde{b}_{ij} - [\tilde{b}_{ij}]$, $1 \leq i \leq m$, $1 \leq j \leq N$, denote the fractional part of b_{ij}, and define

$$I_i = \sum_{j=1}^{N} f_{ij}, \qquad i = 1, 2, \ldots, m . \tag{25}$$

Then, for $i = 1, 2, \ldots, m$,

$$b_i = \sum_{j=1}^{N} \tilde{b}_{ij}$$

$$= \sum_{j=1}^{N} [\tilde{b}_{ij}] + \sum_{j=1}^{N} f_{ij}$$

$$= \sum_{j=1}^{N} [\tilde{b}_{ij}] + I_i \ ,$$

and it follows that I_i must be an integer. Moreover, by (25), since $0 \leq f_{ij} < 1$, $j = 1, 2, \ldots, N$, it is clear that $0 \leq I_i < N$, $i = 1, 2, \ldots, m$. For each i, then, I_i of the b_{ij} $(j = 1,2,\ldots,N)$ may be arbitrarily selected to be rounded up to $[\tilde{b}_{ij}] + 1$, while the others are rounded down to $[\tilde{b}_{ij}]$. One reasonable criterion by which to decide which I_i of the b_{ij} to round up is to choose those $\tilde{b}_{ij}$ with the largest fractional parts f_{ij}:

$$\tilde{b}'_{ij} = \begin{cases} [\tilde{b}_{ij}] + 1, & \text{if } f_{ij} \text{ is among the } I_i \text{ greatest members of } \{f_{ij},\ 1 \leq j \leq N\} \\ [\tilde{b}_{ij}], & \text{otherwise,} \end{cases} \tag{26}$$

for $i = 1, 2, \ldots, m$. Since

$$\sum_{j=1}^{N} \tilde{b}'_{ij} = \sum_{j=1}^{N} [\tilde{b}_{ij}] + I_i = \sum_{j=1}^{N} \tilde{b}_{ij} = b_i \qquad (i = 1, 2, \ldots, m) \ ,$$

the rounded solution $b_{ij} = \tilde{b}'_{ij}$ $(1 \leq i \leq m,\ 1 \leq j \leq N)$ satisfies the constraints (20) of (RMP).

The maximal values $\tilde{z}'_j$ which each auxiliary variable z_j may then assume can be readily computed from the constraints (21). The integer-feasible solution $(\tilde{z}'_j,\ \tilde{b}^{(j)'})$, $j = 1, 2, \ldots, N$, immediately provides an initial incumbent solution for (RMP) (with objective value $\sum_{j=1}^{N} \tilde{z}'_j$),

which may then be used to enhance the power of subsequent fathoming tests. Non-integer solutions produced at other nodes in the course of solving (RMP) may be similarly rounded in the hope of quickly locating improved integer feasible solutions.

In limited preliminary computational tests, this simple rounding procedure was found to be surprisingly effective. For block angular problems with all-integer cost coefficients $c^{(j)}$ $(j = 1, 2, \ldots, N)$ - in which case the auxiliary variables z_j may be integer-restricted as well in (RMP) - the rounded LP-optimal solution $\tilde{b}'_{ij}$ was usually optimal. More importantly, with only rare exceptions it was found that whenever the bound $U^* = \sum_{j=1}^{N} z_j^*$ was sufficient for fathoming a given subset F_n of feasible allocations, the bound $\tilde{U} = \sum_{j=1}^{N} \tilde{z}_j$ was also sufficient for this purpose. Consequently, little fathoming power is lost by terminating the solution procedure for (RMP) as soon as the optimal LP solution has been obtained. If the bound $\tilde{U}$ does not fathom F_n, the (integer-feasible) allocation $\tilde{b}^{(j)'} = (\tilde{b}'_{ij}, \ldots, \tilde{b}'_{mj})$, $j = 1, 2, \ldots, N$, given by (26) may be selected next for examination in Step 4 of the RBBA. The computational effort required to solve the mixed-integer programming problem (RMP) is thereby reduced to little more than solving its linear programming relaxation.

Except for the differences in solution procedures, the relaxed master problem (RMP) is used in the RBBA in exactly the same way as is the corresponding master problem for the single linking constraint case. Remarks analogous to those in Section 4.3 apply here as well. In particular, we note that the objective function bound obtained by solving (RMP) can be tightened by using the known integer programming objective values

corresponding to any fixed subproblem allocations $b^{(j)*}$. As before, in considering a given subset of allocations F_n, let

$$J_1 = \{j \mid \text{The allocation to subproblem } (S_j) \text{ is fixed in } F_n\},$$
$$J_2 = \{j \mid 1 \leq j \leq N\} - J_1 .$$

By appropriately adjusting the right-hand sides in the constraints (20) of (RMP), the variables $(z_j, b^{(j)})$, $j \in J_1$, may be dropped from the problem, together with the corresponding constraints (21). The resulting (reduced) problem,

$$\text{maximize} \quad \sum_{j \in J_2} z_j ,$$

subject to

$$\sum_{j \in J_2} b_{ij} \leq b_i - \sum_{j \in J_1} b^*_{ij}, \quad i = 1, 2, \ldots, m \tag{27}$$

$$z_j - \sum_{i=1}^{m} \lambda^{(k)}_{ij} b_{ij} \leq z^{(k)}_{0j} \quad , \quad k = 1, 2, \ldots, \ell_j, \; j \in J_2$$

$$\underline{b}_{ij} \leq b_{ij} \leq \overline{b}_{ij}, \; b_{ij} \text{ integer} \quad , \quad i = 1, 2, \ldots, m, \; j \in J_2 ,$$

is computationally less expensive to solve than the full problem (RMP) itself. The maximal objective value which (P) can attain over the set F_n can then be bounded by

$$U = \sum_{j \in J_1} v(S_j \mid b^{(j)*}) + \sum_{j \in J_2} z^*_j .$$

Alternatively, if the solution of problem (27) is terminated after the LP-optimal solution $(\tilde{z}_j, \tilde{b}^{(j)})$, $j \in J_2$, is obtained (as suggested above), we have

Decomposition Algorithm for the Block Angular Integer Program (P)

Phase 1: Construction of Master Problem

Step 0. Input problem data. Compute upper and lower bounds $\bar{b}^{(j)}$, $\underline{b}^{(j)}$ on the allocation variables $b^{(j)}$ in the j^{th} subproblem, $j = 1,2,\ldots,N$. If (P) has only a single linking constraint, go to Step 1. Otherwise, go to Step 2.

Step 1. (Single linking constraint.) For $j = 1, 2, \ldots, N$, solve subproblem (S_j) as a linear program parametrically in the amount b_j of the linking resource allocated to (S_j), $\underline{b}_j \leq b_j \leq \bar{b}_j$. Modify the resulting parametric functions $z_j(b_j)$ to eliminate non-integer breakpoints. Go to Phase 2, Step 0.

Step 2. (Multiple linking constraints.) Set $\ell_j = 0$, $j = 1, 2, \ldots, N$. If an allocation $(b^{(1)^*}, \ldots, b^{(N)^*})$ (not necessarily feasible) is known for which each subproblem has a finite LP-optimal solution, set $b^{(j)} = b^{(j)^*}$, $1 \leq j \leq N$. Otherwise, set $b^{(j)} = b^{(j)^*} \equiv \bar{b}^{(j)}$, for $j = 1, 2, \ldots, N$. Set $n = 0$, $j = 1$, and go to Step 3.

Step 3. If $j > N$, go to Step 6. Otherwise, solve subproblem (S_j) as a linear program, using the allocation $b^{(j)} = b^{(j)^*}$ in the linking constraints. Go to Step 4.

Step 4. Let $\bar{y}^{(j)}$ be the $(m+m_j)$-vector of optimal dual multipliers for (S_j), as obtained in Step 3. If no finite optimal $\bar{y}^{(j)}$ exists, or if for some $1 \leq k \leq \ell_j$,

$$\sum_{i=1}^{m_j} \bar{y}^{(j)}_{m+i} \beta^{(j)}_i = z^{(k)}_{0j} \quad \text{and} \quad \bar{y}^{(j)}_i = \lambda^{(k)}_{ij}, \qquad 1 \leq i \leq m,$$

then reset $j = j + 1$ and go to Step 3. Otherwise, go to Step 5.

Step 5. Reset $\ell_j = \ell_j + 1$, and set

$$z_{0j}^{(\ell_j)} = \sum_{i=1}^{m_j} \bar{y}_{m+i}^{(j)} \beta_i^{(j)} \quad \text{and} \quad \lambda_{ij}^{(\ell_j)} = \bar{y}_i^{(j)}, \qquad 1 \leq i \leq m .$$

Then reset $n = n + 1$, $j = j + 1$, and go to Step 3.

Step 6. If $n = 0$, go to Phase 2, Step 0. Otherwise, solve the linear programming relaxation of (RMP), using multipliers $\{\lambda_{ij}^{(k)}, 1 \leq i \leq m\}$ and corresponding constants $z_{0j}^{(k)}$ $(k = 1, 2, \ldots, \ell_j, j = 1, 2, \ldots, N)$ to construct the constraints (21). For any $1 \leq j \leq N$ such that $\ell_j = 0$, use $z_{0j}^{(1)} = +\infty$, $\lambda_{ij}^{(1)} = 0$, $1 \leq i \leq m$, in (21). Reset $(b^{(1)*}, \ldots, b^{(N)*})$ to the LP-optimal solution to (RMP). Reset $n = 0$, $j = 1$, and go to Step 3.

Phase 2: Search for Optimal Allocation

Step 0. If an integer solution to (P) is known *a priori*, set z^* equal to its objective value and store the solution as the current incumbent. Otherwise, set $z^* = -\infty$. Set $J_1 = \phi$, $J_2 = \{1, 2, \ldots, N\}$. Go to Step 2.

Step 1. If the list of stored subsets F_k is empty, stop; the current incumbent solution is optimal. (If none has been found, (P) is infeasible.) Otherwise, select the last subset F_n on the list as the next one to be examined. Update the allocation variable bounds $\underline{b}^{(j)}$, $\bar{b}^{(j)}$ appropriately to correspond to those defining the subset of feasible allocations F_n. Correspondingly, update the sets J_1, J_2; specifically, if the allocation

to subproblem (S_j) was fixed previously, but is not fixed in F_n, reset $J_1 = J_1 - \{j\}$, $J_2 = J_2 \cup \{j\}$. Go to Step 2.

Step 2. (a) Single Linking Constraint. Solve the concave-separable master problem (MP) (see Sections 4.1, 4.2), using the current bounds $\underline{b}_j$, $\overline{b}_j$. Set $(b_1^*, \ldots, b_N^*)$ equal to the optimal integer solution. Set

$$U = \sum_{j \in J_1} v(S_j | b_j^*) + \sum_{j \in J_2} z_j(b_j^*) ,$$

and go to Step 3.

(b) Multiple Linking Constraints. Using the multipliers $\{\lambda_{ij}^{(k)}, 1 \leq i \leq m\}$ and corresponding constants $z_{0j}^{(k)}$ $(k = 1, 2, \ldots, \ell_j, j \in J_2)$, construct the relaxed master problem (27). For any $j \in J_2$ for which $\ell_j = 0$, use $z_{0j}^{(1)} = +\infty$, $\lambda_{ij}^{(1)} = 0$, $1 \leq i \leq m$, in (27). For $j \in J_2$, set $(\tilde{z}_j, \tilde{b}^{(j)})$ equal to the optimal LP solution, and set

$$U = \sum_{j \in J_1} v(S_j | b^{(j)^*}) + \sum_{j \in J_2} \tilde{z}_j ,$$

where $b^{(j)^*}$, $j \in J_1$, is the current level at which the allocation to (S_j) is fixed. Go to Step 3.

Step 3. (a) Single Linking Constraint. If $U \leq z^*$, go to Step 1. Otherwise, go to Step 4.

(b) Multiple Linking Constraints. If $U \leq z^*$, go to Step 1. Otherwise, round the allocations $\tilde{b}^{(j)}$, $j \in J_2$, as in (26), and set $b^{(j)^*}$, $j \in J_2$, equal to the resulting integer-feasible values. Go to Step 4.

Step 4. Compare the current allocations $b^{(j)^*}$ with the stored list of previously examined allocations $b^{(j)'}$ (if any) in subproblem (S_j), $j \in J_2$. If some $j_0 \in J_2$ yields $b^{(j_0)'} - s^{(j_0)'} \leq b^{(j_0)^*} \leq b^{(j_0)'}$

(where $b^{(j_0)'} - s^{(j_0)'}$ represents the level of linking resources used in the optimal solution to (S_{j_0}) under the allocation $b^{(j_0)'}$), then set $s^{(j_0)^*} = b^{(j_0)^*} - (b^{(j_0)'} - s^{(j_0)'})$ and go to Step 10. Otherwise, go to Step 5.

Step 5. Select a subproblem (S_{j_0}), $j_0 \in J_2$, by either[1/]

(i) $j_0 = \min\{j | j \in J_2\}$; or

(ii) $j_0 = \arg\min_{j \in J_2} T_j$, where T_j is the average solution time for (S_j) observed in the previous solutions of of (S_j) in Phase 2; or

(iii) $j_0 = \arg\max_{j \in J_2} G_j$, where G_j is the average gap between the LP bound $z_j(b^{(j)^*})$ obtained in Step 2 and the actual corresponding integer-programming value $v(S_j | b^{(j)^*})$ for (S_j) observed in the previous solutions of (S_j) in Phase 2; or

(iv) $j_0 = \arg\max_{j \in J_2} X_j$, where X_j is the average sum-of-slacks $\sum_{i=1}^{m} s_i^{(j)^*}$ observed in the linking constraints in previous solutions of (S_j) in Phase 2.

Then go to Step 6.

[1/] The notation $j_0 = \arg\min_{j \in J_2} T_j$ $(\arg\max_{j \in J_2} G_j, \arg\max_{j \in J_2} X_j)$ is used to indicate that the index j_0 is to be chosen so that $T_{j_0} = \min_{j \in J_2} T_j$ $(G_{j_0} = \max_{j \in J_2} G_j, X_{j_0} = \max_{j \in J_2} X_j)$.

Step 6. Compare the current allocation $b^{(j_0)^*}$ with the stored list of previously determined levels of optimal linking resource usage $b^{(j_0)'} - s^{(j_0)'}$ in (S_{j_0}). Let $B = \{b^{(j_0)'} \mid b^{(j_0)^*} \geq b^{(j_0)'} - s^{(j_0)'}\}$, and set

$$\underline{z} = \max_{b^{(j_0)'} \in B} v(S_{j_0} \mid b^{(j_0)'}) .$$

If $B = \emptyset$, set $\underline{z} = -\infty$. Go to Step 7.

Step 7. Solve (S_{j_0}) using the allocation $b^{(j_0)^*}$. If a branch-and-bound method is used, the value of $\underline{z}$ computed in Step 6 provides an initial lower bound on the optimal objective value. Then,

(i) If (S_{j_0}) is infeasible, set $s^{(j_0)^*} = +\infty$, $z^*_{j_0} = -\infty$, and go to Step 9.

(ii) If there exists a finite optimal solution $x^{(j_0)^*}$ to (S_{j_0}), set $s^{(j_0)^*} = b^{(j_0)^*} - A^{(j_0)} x^{(j_0)^*}$ and $z^*_{j_0} = c^{(j_0)} x^{(j_0)^*}$. If $m = 1$ (single linking constraint case), go to Step 9. If $m > 1$, go to Step 8.

(iii) If (S_{j_0}) is feasible, but the optimal solution is unbounded, let $x^{(j_0)^*}$ be an endpoint of the ray generating the unbounded solution, and set $s^{(j_0)^*} = b^{(j_0)^*} - A^{(j_0)} x^{(j_0)^*}$. (See discussion below.) Set $z^*_{j_0} = +\infty$. Go to Step 9.

Step 8. (Used in multiple linking constraint case only.) Let $\underline{y}^{(j_0)}$ be the $(m+m_{j_0})$-vector of LP-optimal dual multipliers for (S_{j_0}) under the allocation $b^{(j_0)^*}$ (as determined in Step 7). If for some

$k = 1, 2, \ldots, \ell_j$,

$$\sum_{i=1}^{m_{j_0}} \underline{y}_{m+i}^{(j_0)} \beta_i^{(j_0)} = z_{0j_0}^{(k)} \quad \text{and} \quad \underline{y}_i^{(j_0)} = \lambda_{ij_0}^{(k)}, \qquad 1 \leq i \leq m,$$

go to Step 9. Otherwise, reset $\ell_j = \ell_j + 1$, and set

$$z_{0j_0}^{(\ell_{j_0})} = \sum_{i=1}^{m_{j_0}} \underline{y}_{m+i}^{(j_0)} \beta_i^{(j_0)}, \quad \text{and} \quad \lambda_{ij_0}^{(\ell_{j_0})} = \underline{y}_i^{(j_0)}, \qquad 1 \leq i \leq m.$$

Then go to Step 9.

Step 9. Add $b^{(j_0)^*}$ to the list of examined allocations. Correspondingly, add $b^{(j_0)^*} - s^{(j_0)^*}$ and $z_{j_0}^* = v(S_j | b^{(j_0)^*})$ to the lists of optimal linking resource usage levels and associated objective values, respectively. Go to Step 10.

Step 10. If $|J_2| \leq 1$, go to Step 11. Otherwise, branch on the allocation variables $b^{(j_0)}$ as described in Chapter 3. (See especially Figure 2 of Chapter 3.) Add each of the newly formed subsets F_k to the list of such subsets. If $\underline{b}_{ij_0} > b_{ij_0}^* - s_{ij_0}^*$ for some $1 \leq i \leq m$, go to Step 1. Otherwise, revise the bounds on $b^{(j_0)}$ to $\underline{b}^{(j_0)} = b^{(j_0)^*} - s^{(j_0)^*}$, $\overline{b}^{(j_0)} = b^{(j_0)^*}$,[1/] and reset $J_1 = J_1 \cup \{j_0\}$, $J_2 = J_2 - \{j_0\}$. (The allocation in subproblem (S_{j_0}) is then temporarily fixed at the level $b^{(j_0)} = b^{(j_0)^*} - s^{(j_0)^*}$.) Go to Step 2.

[1/] If in Step 4 of Phase 2 it was discovered that $b^{(j_0)'} - s^{(j_0)'} \leq b^{(j_0)^*} \leq b^{(j_0)'}$ for some previously examined allocation $b^{(j_0)'}$, it will be more efficient to reset $\overline{b}^{(j_0)} = \min(\overline{b}^{(j_0)}, b^{(j_0)'}) \geq b^{(j_0)^*}$.

AD-A033 104 STANFORD UNIV CALIF SYSTEMS OPTIMIZATION LAB F/G 9/2
DECOMPOSITION IN INTEGER PROGRAMMING.(U)
SEP 76 G A KOCHMAN F44620-74-C-0079
UNCLASSIFIED SOL-76-21 AFOSR-TR-76-1213 NL

2 OF 2
AD A033104

END
DATE FILMED
1-77

Step 11. If $\sum_{j=1}^{N} z_j^* \leq z^*$, go to Step 1. Otherwise, reset $z^* = \sum_{j=1}^{N} z_j^*$, and install $(x^{(1)*}, \ldots, x^{(N)*})$ as the new incumbent solution; then go to Step 1.

The (optional) Steps 4, 6, 9 in Phase 2 of the decomposition algorithm can improve algorithmic efficiency in the following ways. In Step 4, if $b^{(j_0)'} - s^{(j_0)'} \leq b^{(j_0)*} \leq b^{(j_0)'}$, the solution which was optimal for (S_{j_0}) under the allocation $b^{(j_0)} = b^{(j_0)'}$ must also be optimal for the current allocation $b^{(j_0)} = b^{(j_0)*}$. In this case, therefore, the work to resolve (S_{j_0}) can be avoided, and instead we can jump immediately to the branching step (Step 10). Branching on the allocation variables $b^{(j_0)}$ is done in exactly the same way as if (S_{j_0}) had been resolved explicitly under the allocation $b^{(j_0)} = b^{(j_0)'}$.

At Step 6 in Phase 2, when the subproblem (S_{j_0}) must be resolved explicitly, the validity of the lower bound $\underline{z}$ on $v(S_{j_0}|b^{(j_0)*})$ follows from the observation that $v(S_{j_0}|b^{(j_0)})$ must be nondecreasing in $b^{(j_0)}$. If a branch-and-bound search procedure is used to solve (S_{j_0}) in Step 7, the lower bound $\underline{z}$ is immediately available for use in fathoming tests, and can thereby reduce the total computational effort of solution. However, in general, any subproblem solution method may be used in Step 7 in the single linking constraint case, and any method yielding the LP-optimal dual multipliers $\underline{y}^{(j_0)}$ as a by-product of the solution may be used in the multiple linking constraint case. If the method chosen for subproblem solution is not of the branch-and-bound type, it may be more efficient to omit Step 6 in Phase 2.

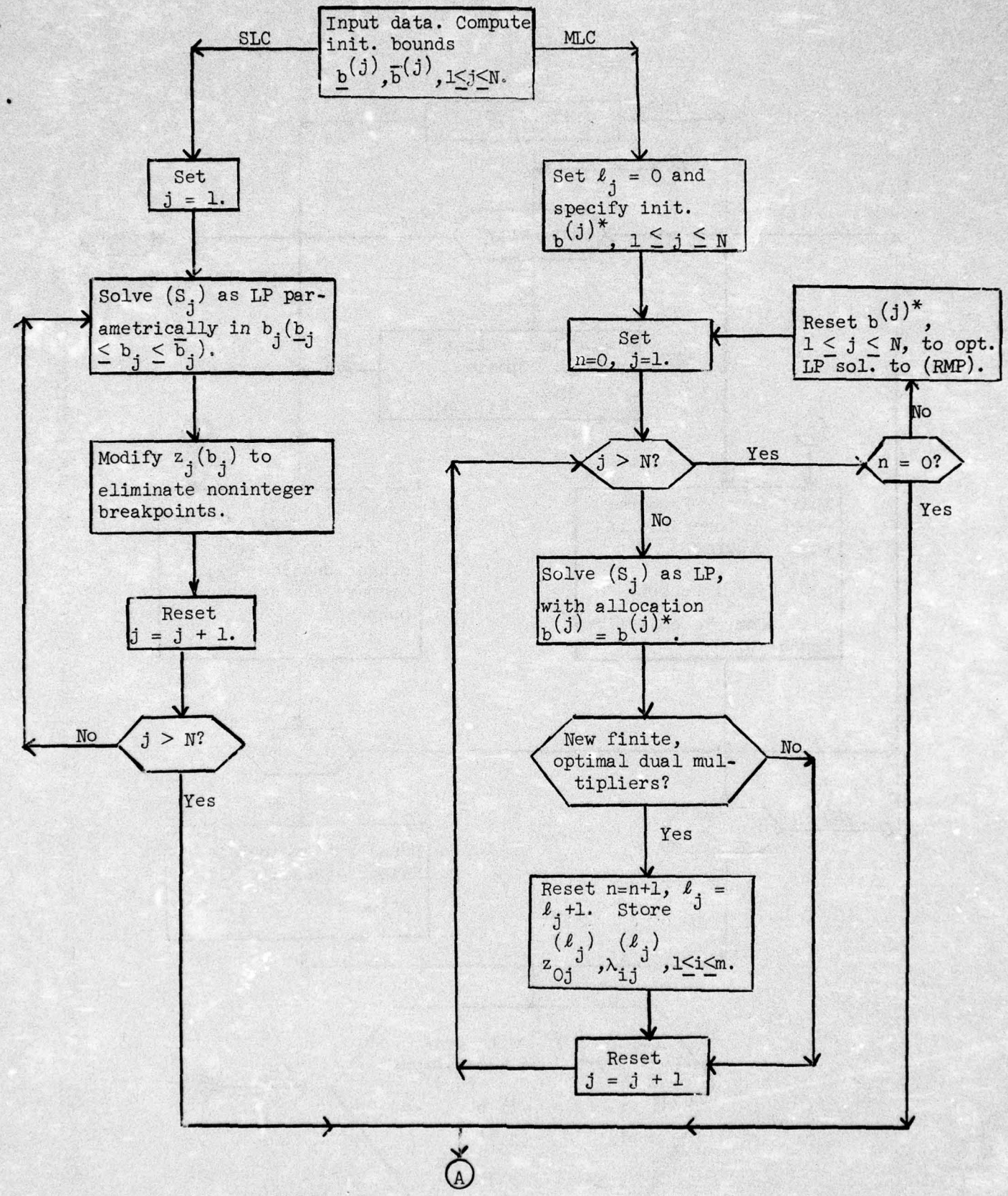

Figure 1: Flow diagram for Phase 1 of the decomposition algorithm

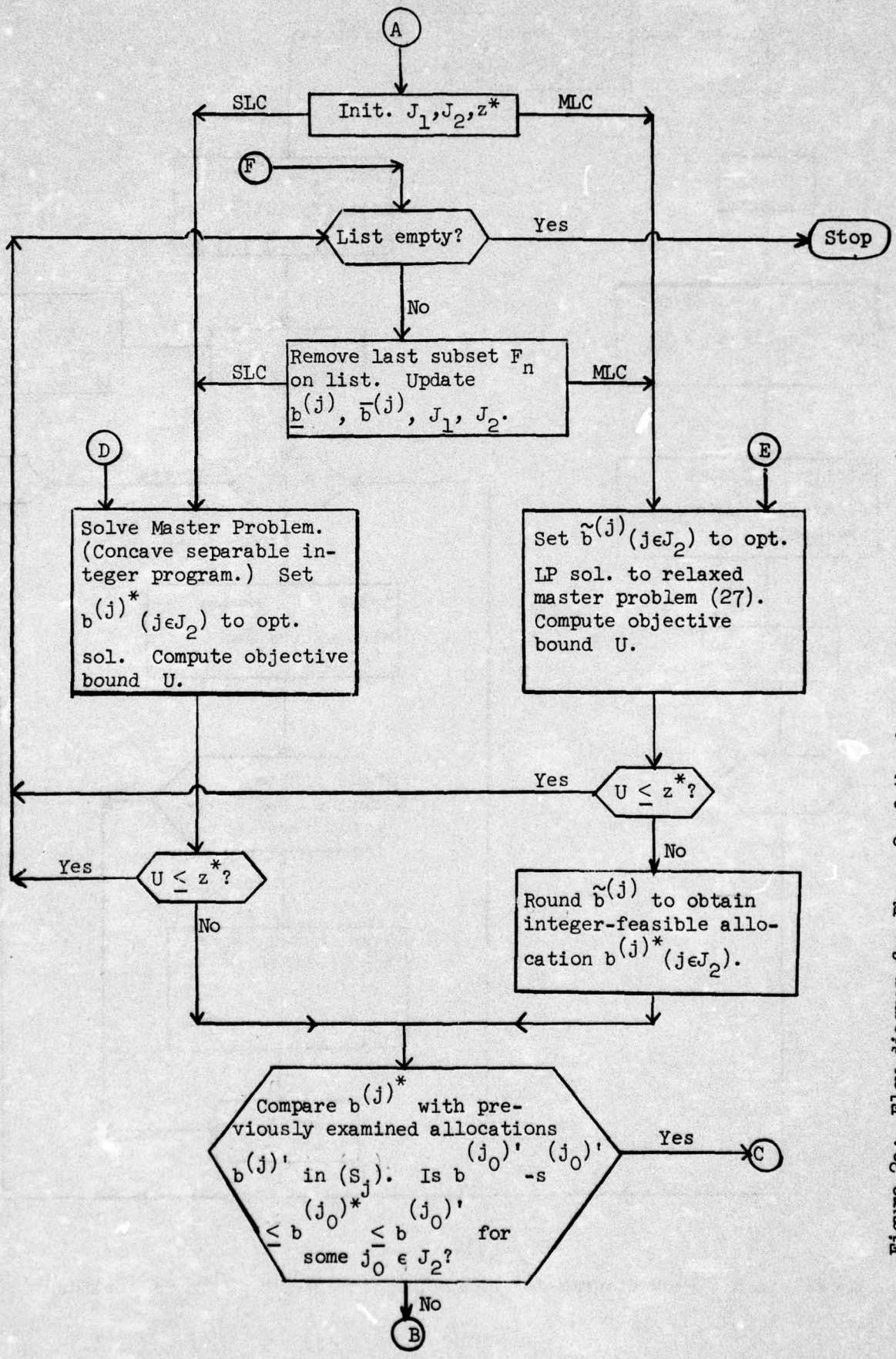

Figure 2a: Flow diagram for Phase 2 of the decomposition algorithm (continued in Figure 2b)

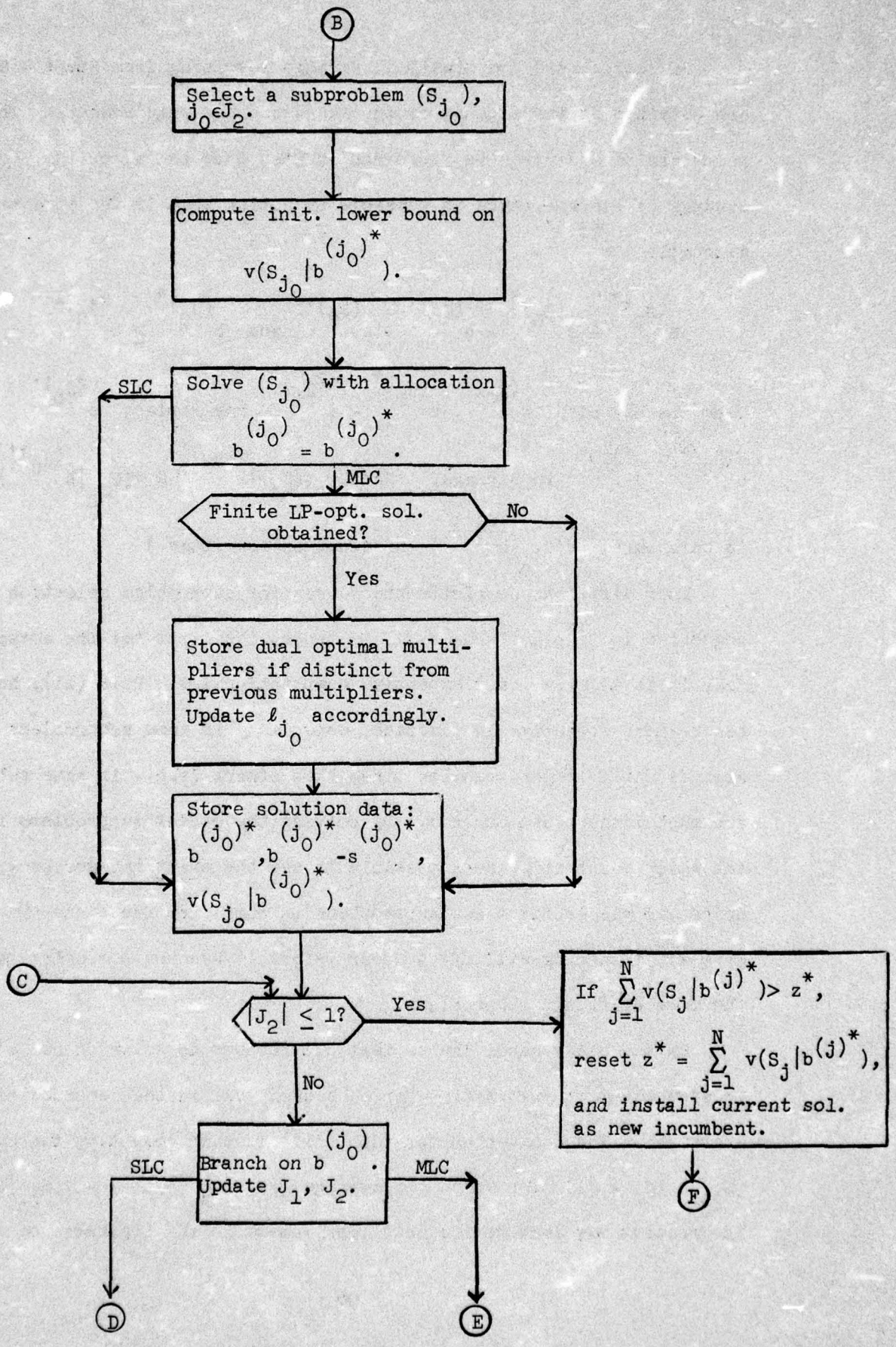

Figure 2b: Flow diagram for Phase 2 of the decomposition algorithm (continued from Figure 2a)

Of course, the computational savings resulting from Steps 4 and 6 are obtained at the expense of increased storage requirements. The necessary data storage is performed in Step 9 of the algorithm. Some savings of storage space is possible by noting that if for some previous allocation $b^{(j_0)'}$,

$$b^{(j_0)^*} - s^{(j_0)^*} = b^{(j_0)'} - s^{(j_0)'} \quad \text{and} \quad b^{(j_0)^*} \geq b^{(j_0)'},$$

then the new data $b^{(j_0)^*}$, $b^{(j_0)^*} - s^{(j_0)^*}$ may replace $b^{(j_0)'}$, $b^{(j_0)'} - s^{(j_0)'}$ in storage. (Since $v(S_{j_0}|b^{(j_0)^*}) = v(S_{j_0}|b^{(j_0)'})$ in this case, $v(S_{j_0}|b^{(j_0)^*})$ need not be rewritten.)

Four different heuristic procedures for subproblem selection are suggested in Step 5 of Phase 2. Of these, the first has the advantage that it is clearly least expensive computationally. Rule (ii), however, can be more effective in practice, especially if some subproblems require significantly longer solution times than others (e.g., if some subproblems are much larger than others). By solving the easier subproblems first (at which point it becomes possible to use the exact integer programming objective values for these subproblems in computing the bound U in Step 2), fathoming will often occur before it becomes necessary to solve the more difficult subproblems.

On the other hand, the easiest subproblems to solve do not always provide the most information for subsequent use in the decomposition algorithm. The motivation for rule (iii) is that selecting the subproblem (S_{j_0}) $(j_0 \in J_2)$ for which the average gap, $z_{j_0}(b^{(j_0)}) - v(S_{j_0}|b^{(j_0)})$, is greatest may lead to the best improvement in the tightness of the bound

U computed in Step 2 at the next iteration. The hope is, of course, that by enhancing the power of the fathoming test in Step 3, the total number of subproblem solutions required in the course of the search will be reduced. Alternatively, the use of rule (iv) in Step 5 can lead to the greatest inprovement in the allocations generated in Step 3 of the algorithm; following the solution of (S_{j_0}) in Step 7 and subsequent branching in Step 10, the slacks $s^{(j_0)^*}$ are reallocated to the remaining unfixed subproblems in Step 3. Hence, it also would seem reasonable to solve first those subproblems (S_j) which consistently give rise to the largest values of $s^{(j)^*}$.

Preliminary experimentation has indicated that the selection rules based on subproblem solution times (rule (ii)) and average gaps between the subproblems' linear and integer programming objective values (rule (iii)) may be the most advantageous in practice. Of these two, rule (iii) appeared to be more consistently effective. However, inasmuch as the subproblems in the test problems run were all of equal size, it is conceivable that rule (ii) may indeed be more favorable when the differences in subproblem solution times are more dramatic. Similarly, while rule (iv) was found to be little more effective than the rather arbitrary selection rule (i) (and sometimes much less effective), it should be pointed out that subproblem linking constraint coefficients were of approximately equal magnitude in all test problems run. Consequently, the resulting optimal slacks $s^{(j^*)}$ tended to be of equal magnitude for all subproblems.

For completeness, it is necessary in Step 7 of Phase 2 of the decomposition algorithm to consider the situation when the subproblem (S_{j_0})

is found to have an unbounded optimal solution. Hence, let

$$X(b^{(j_0)^*}) \equiv \{x^{(j_0)} \mid A^{(j_0)} x^{(j_0)} \leq b^{(j_0)^*}, B^{(j_0)} x^{(j_0)} \leq \beta^{(j_0)}, x^{(j_0)} \geq 0, x^{(j_0)} \text{ integral}\} .$$

Kaneko (1974) has shown that whenever the entries of the matrices $A^{(j_0)}$, $B^{(j_0)}$ are rational, the unboundedness of $X(b^{(j_0)^*})$ implies the existence of a "discrete ray" $R \subset X(b^{(j_0)^*})$, where

$$R = \{x^{(j_0)} \mid x^{(j_0)} = x^{(j_0)^*} + \theta d, \theta \in I\} ,$$

and where $x^{(j_0)^*} \in X(b^{(j_0)^*})$, the n_{j_0}-vector $d \neq 0$, and I is an infinite subset of the set of nonnegative numbers. Ideally, one would like to choose the endpoint $x^{(j_0)^*}$ in part (iii) of Step 7 to make each component of the slack-vector $s^{(j_0)^*} \equiv b^{(j_0)^*} - A^{(j_0)} x^{(j_0)^*}$ as large as possible, in the hope of subsequently improving the allocations to the remaining subproblems. However, the difficulty of determining an appropriate endpoint for this purpose may well be prohibitive, and for our purposes any integer-feasible endpoint $x^{(j_0)^*} \in R$ will suffice.

Finally, we remark that it is easy to see that if $\ell_j \geq 1$ for all $1 \leq j \leq N$ at the end of Phase 1 in the multiple linking constraint case, then the initial relaxed master problem (RMP) in Phase 2 is sufficient for solving the block angular linear programming problem associated with (P). Specifically, the first allocation $(\tilde{b}^{(1)}, \ldots, \tilde{b}^{(N)})$ obtained in Step 2 of Phase 2 of the decomposition algorithm must lead to the overall optimal LP-solution to (P) when each subproblem (S_j)

is solved as a linear program under the vector of linking constraint right-hand sides $\tilde{b}^{(j)}$, $j = 1, 2, \ldots, N$. Similarly, the Phase 1 procedure in the single linking constraint case solves the corresponding linear programming relaxation of (P), provided the parametric functions $z_j(b_j)$ are <u>not</u> modified to eliminate non-integer breakpoints.

5.3 Possible Modifications to the Algorithm

There are several possible modifications to the decomposition algorithm which may merit further investigation. It may be computationally more efficient, for example, not to resolve the Master Problem after every subproblem solution.[1/] Instead, after a given subproblem is solved, the slacks associated with the linking constraints can be arbitrarily reallocated to the remaining (unfixed) subproblems. One possible heuristic reallocation procedure that would be computationally less expensive is simply to assign all slacks to the next subproblem to be solved.

The Master Problem must at least be resolved periodically, of course, to test for fathoming and to identify promising new allocations when fathoming does not occur. Consequently, some computational experimentation with various (possibly dynamic) decision rules governing the frequency of Master Problem reoptimization would be necessary. The decision not to resolve the Master Problem at every node in the search tree inevitably results in the loss of fathoming capabilities at the "skipped" nodes, so that some of the computational savings obtained in by-passing the Master Problem may be lost in examining nodes which would

---

[1/] Since the special structure of the Master Problem in the single linking constraint case gives rise to the efficient solution procedure developed in Chapter 4, this modification is perhaps more relevant for the multiple linking constraints case.

otherwise have been fathomed. However, especially in large block angular problems, where fathoming is likely to occur only after many subproblems have been solved, the loss may well be minimal.

Secondly, it is not strictly necessary to branch on the corresponding allocation variables immediately after each subproblem is solved. An alternative strategy would be to successively reallocate any slacks appearing in the linking constraints in each subproblem solution (as is currently done), but to postpone branching until all subproblems have been solved, or until the current allocation can be recognized as being suboptimal. At this point, one subproblem may be chosen, and the corresponding allocation variables branched on in the usual way.

This approach would allow greater flexibility in the order in which the branching variables are selected. Concomitant with such an approach, therefore, might be the development of procedures for computing "penalties" associated with the branching process. These penalties, in analogy with those used in standard (nondecomposition) branch-and-bound algorithms, could be used to guide branch selection, and to tighten the objective function bounds associated with the stored nodes.

CHAPTER 6

COMPUTATIONAL RESULTS

Evaluation of the decomposition algorithm presented in Section 5.2 was carried out on Stanford University's IBM 370/168 computer, using programs written in FORTRAN IV by the author. In particular, a branch-and-bound procedure for the general integer programming problem, and two versions of the algorithm for decomposing block angular integer programs (corresponding to the single and multiple linking constraint cases, respectively) were coded.

The branch-and-bound code was written for use in generating the subproblem solutions required in Step 7 of Phase 2 of the decomposition procedures. In addition, this code also served to provide a guideline for evaluating the practical effectiveness of the decomposition algorithm, in that the test results obtained by the latter can be compared to those which might be obtained by a nondecomposition approach to the same problems. The decision, specifically, to implement a branch-and-bound routine for this dual purpose was made in consideration of the fact that branch-and-bound algorithms appear to be the most consistently effective of the various types of general integer programming procedures.

The branch-and-bound routine coded is based on the algorithm suggested by Dakin (1965). Tomlin's (1970) improved penalties are used to enhance the power of fathoming tests at each node, and to guide branch selection at unfathomed nodes. Specifically, the branching variable is always chosen to be the variable with the largest associated up- or

down-penalty, and the forward branch is made in the direction opposite to the maximum penalty. The node generated by the branch in the same direction as the maximum penalty is added to the list to be examined later. This branching strategy was first adopted by Little, Murty, Sweeney, and Karel (1963), and has since enjoyed considerable success. (See, for example, Beale and Small (1965).) Stored nodes are retrieved from the list in accordance with the last-in-first-out (LIFO) rule. (See Section 2.2.) For more details, the interested reader is referred to the references above, and also to Beale (1968).

The primal-simplex code LPM-1 was used to form the linear programming foundation for our three programs. This code uses the product-form of the inverse, and was developed by J. A. Tomlin of the Systems Optimization Laboratory, Department of Operations Research, Stanford University. In order to use LPM-1 within the context of a branch-and-bound procedure, the program was modified to handle variables with general upper and lower bounds, and appropriate row and column selection subroutines were added to facilitate operation as a dual-simplex method as well. Reoptimization following a branch in the forward direction is carried out by dual-simplex pivots, while primal-simplex pivoting is used for reoptimization when backtracking to examine a previously stored node. The LPM-1 code is also used to solve the relaxed master problem (RMP) in the decomposition program written for the multiple linking constraints case. The fact that some of the variables in (RMP) are not sign restricted presents no additional difficulties, since LPM-1 has been modified to treat variables with general upper and lower bounds.

The remaining portions of the codes were written in modular form according to the various tasks to be performed (e.g., branching, penalty

calculation, fathoming tests, etc., for the branch-and-bound code; master problem construction, master problem solution, branching on the allocation variables, etc., for the decomposition codes). The final version of the branch-and-bound code was compiled under the IBM FORTRAN-H-Extended compiler, using the optimization option at level two. While dimensioned to treat general integer programming problems with as many as 2000 nonzero constraint coefficients, the object program is approximately 113 K bytes in length. The similarly compiled final version of the decomposition procedure in the single linking constraint case, dimensioned to handle up to ten subproblems with a total of as many as 1000 nonzero constraint coefficients, is approximately 125 K bytes in length. For both programs, all instructions and data are in-core. While a code for the decomposition of block angular integer programs with multiple linking constraints was developed and debugged under the WATFIV compiler, no object code was generated for this program. The test results reported below are for the single linking constraint case only.

While the author does not claim the expertise of a professional programmer, it is hoped that by writing these codes himself, and by using the same branch-and-bound integer programming procedure throughout, the effects of discrepancies in programming sophistication on test results will be mitigated. Nonetheless, the results reported here should not necessarily be considered representative of those which might be obtained by professionally developed codes.

6.1 The Testing Procedure

There are several parameters which can affect the performance of the decomposition algorithm, including linking constraint tightness,

overall problem size, subproblem size, the number of linking constraints, etc. As limitations on time and computer funds prohibited the detailed examination of all such parameters, however, it was decided to restrict the computational testing of the algorithm primarily to study the effects of linking constraint tightness and overall problem size.

In order to test the effects of linking constraint tightness on solution times, some control over the tightness of all the constraints must be established. To accomplish this, in each test problem the coefficients in each subproblem constraint row were summed, and the right-hand side (RHS) for that row was set to a given multiple of the sum. Thus, in the notation used in the following sections, RHS = 300% (500%) indicates that the right-hand side in each of the subproblem constraints was set equal to 3 (5) times the sum of the coefficients in the corresponding row.

Once the right-hand sides in the subproblem constraints were fixed, the right-hand side for the linking constraint was similarly set equal to a given multiple of the sum of linking constraint coefficients. By varying this multiple with RHS fixed, the relative tightness of the linking constraint can be varied.

Definition: Let the right-hand sides of the subproblem constraints be fixed at a given multiple, say k_1, of the sum of constraint coefficients. Then, if the right-hand side in the linking constraint is set to k_2 times the sum of linking constraint coefficients, the tightness ratio TR is

$$TR \equiv k_2/k_1 .$$

The tightness ratio TR was used as a measure of the degree of linking constraint tightness in the computational results reported below; the smaller the value of TR, the tighter is the linking constraint relative to the subproblem constraints. Reasoning very intuitively, one might expect that the performance of the decomposition algorithm would improve relative to that of a nondecomposition approach as TR is increased. If the linking constraint is so tight that the subproblem constraints are redundant, the problem (P) is not really a block angular integer program at all, but in effect has been tricked into taking on a block angular appearance by the addition of redundant subproblems. Consequently, there would seem to be little advantage in using the decomposition procedure. At the other extreme, if TR is so large that the linking constraint is redundant, (P) consists of N independent problems which have been artificially forced together into one. In this case, the decomposition approach should be highly efficient! The purpose in examining the effects of linking constraint tightness, therefore, was to test the validity of this conjecture, and, if valid, to try to distinguish the ranges of TR in which each of the two approaches is superior.

The method used to test the effects of overall problem size on solution times is similar to the method reported by Reardon (1974). Once a given block angular integer program (P) was formulated (with, say, N subproblems in all), derivative problems with 2, 4, 6, ... , N subproblems were obtained by deleting various numbers of the subproblems of (P). Consequently, the derivative problems retained many of the characteristics of the source problem, and so were closely related in terms of their degree of computational difficulty. As Reardon pointed out,

this relationship enables the direct comparison of solution times for problems of increasing size, since differences in solution times may be attributed largely to the size differences alone.

In all, four test problems and their derivatives were solved. The data for the first two problems was obtained from a test problem of Reardon's, and attention was concentrated on the effects on solution times of various levels of linking constraint tightness in these problems. The data for the third test problem was taken from a capital budgeting application reported by Fleisher (1976). For this problem, both the effects of linking constraint tightness and problem size were investigated in turn. In the fourth problem, the data originated from an R & D project selection problem given by Petersen (1967). The primary objective in this case was to observe the dependence of solution times on problem size. In addition, for both the third and fourth test problems, internal timers were added to the two codes to monitor the progress of the solution with computation time. In particular, whenever a new incumbent solution was found, the objective value of the solution and the computation time required to reach that solution were printed out. This information provides insight into the effectiveness of the decomposition algorithm as a heuristic procedure for quickly generating good, though possibly nonoptimal, solutions for block angular problems which may be too large to solve optimally.

6.2 Test Problems 1 and 2

The first two test problems consisted essentially of subproblems one through four and subproblems five through eight, respectively, of Reardon's test problem number one. The original data for this problem

was derived from Petersen's R & D project selection problem 5. Reardon transformed the data appropriately to obtain an integer programming problem with the dual angular structure. By taking the transpose of the first linking column in Reardon's problem as the single linking constraint in ours, and dropping the remaining linking columns, we obtained a problem with the block angular structure suitable for use in testing the decomposition algorithm. Since the primary interest initially was in the performance of the decomposition algorithm in relation to the degree of linking constraint tightness, rather than problem size, the problem was arbitrarily divided into the two smaller test problems of four subproblems each. The savings in computation time which resulted from this reduction enabled more extensive testing of the effects of linking constraint tightness than would otherwise have been possible.

Inasmuch as each of the subproblem constraint matrices consisted of five rows and five columns, both Test Problems 1 and 2 were 21-row, 20-variable integer programs overall. These problems are exhibited in Tables A-I and A-II, respectively, in the appendix.

In Test Problem 1, the right-hand sides of the subproblem constraints were first fixed at three times the sum of the corresponding row coefficients (RHS = 300%). Attempts were made to solve the problem, both by the decomposition algorithm and by the standard branch-and-bound algorithm, for TR = 0.33, 1.00, 1.67. These test results are recorded in Table I, and are plotted in Figure 1. The right-hand sides of the subproblem constraints were next set equal to RHS = 500%, and the problem was rerun with TR = 0.2, 0.4, 0.6, 0.8. The solution times for these runs are shown in Table II, and are plotted in Figure 2.

TABLE I

Test Problem 1: RHS = 300%

SOLUTION TIMES AS A FUNCTION OF LINKING CONSTRAINT TIGHTNESS

	TIGHTNESS RATIO		
Algorithm	0.33	1.00	1.67
DSLC	2.45	6.15	6.08
BB	5.96	20.54	> 8.30[1/]

DSLC = Decomposition algorithm for single linking constraint case

BB = Branch-and-bound algorithm

TABLE II

Test Problem 1: RHS = 500%

SOLUTION TIMES AS A FUNCTION OF LINKING CONSTRAINT TIGHTNESS

	TIGHTNESS RATIO			
Algorithm	0.2	0.4	0.6	0.8
DSLC	1.79	4.88	4.44	> 20
BB	0.58	7.25	8.67	> 20

[1/] The two cases TR = 1.00, TR = 1.67 were attempted on the same run, with a combined time limit of 30 seconds for the job. WATFIV compilation required 1.16 seconds.

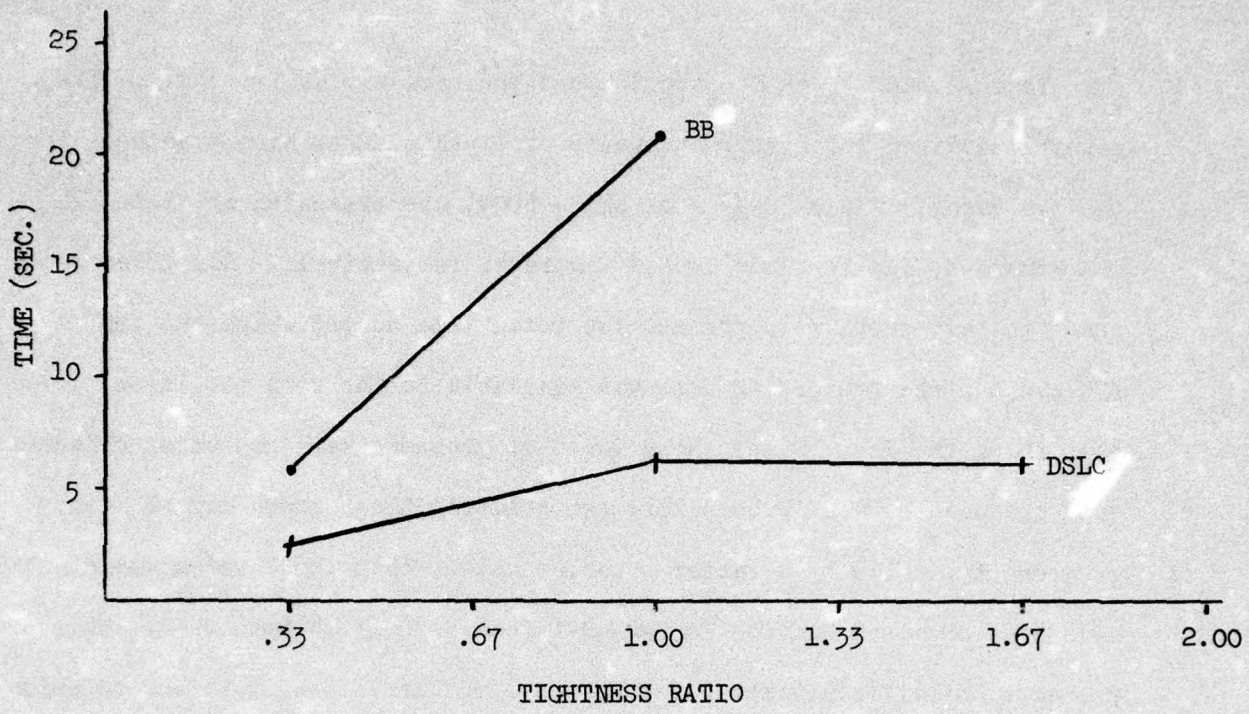

Figure 1: Solution times for Test Problem 1 (RHS = 300%)

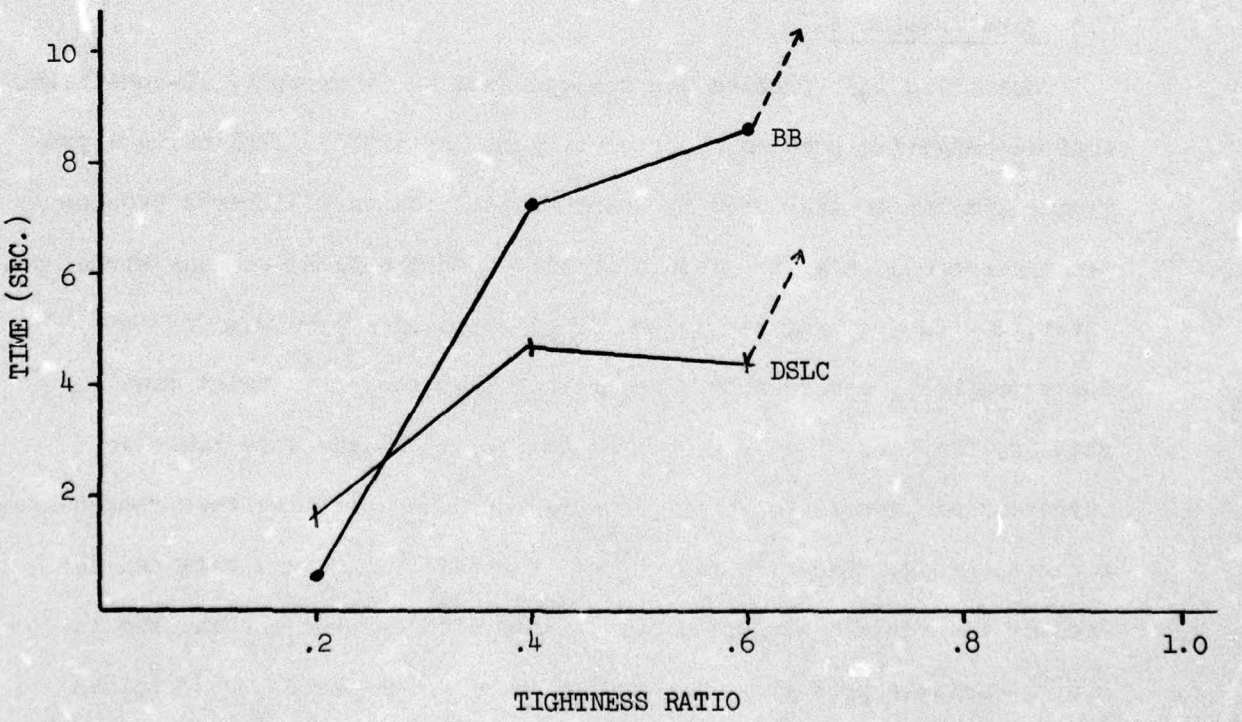

Figure 2: Solution times for Test Problem 1 (RHS = 500%)

Test Problem 2 was treated in much the same way as Problem 1. The results obtained under various levels of linking constraint tightness for the two cases, RHS = 300% and RHS = 500%, are presented in Table III and Figure 3, and in Table IV and Figure 4, respectively. All times shown for both test problems represent the total time during which the IBM 370/168 central processing unit was available to the core partition containing the job. Since these two test problems were run under object code produced by WATFIV compiler, the solution times shown may be much greater (typically by a factor of about seven) than those which would have been obtained had the FORTRAN-H-Extended compiler been used. This presents no difficulty for our purposes, however, since it is the relationships between these solution times, rather than the magnitude of the times themselves, that is of key importance.

6.3 Test Problem 3

Our third test problem was derived from a 20-variable, 10-constraint capital budgeting problem reported by Fleisher (1976). Following a procedure similar to that used by Reardon, the data in Fleisher's problem was transformed so as to yield a problem with the block angular structure. First, the twenty projects in the original problem were divided into forty projects, governed by five, rather than ten, constraint rows. That is, the first five rows of the constraint matrix were taken to represent one group of twenty projects, and the last five rows represented a second group. The original objective coefficients were retained for each of the revised projects; e.g., since both columns one and two in our 40-variable problem were obtained from column one of the original

TABLE III

Test Problem 2: RHS = 300%

SOLUTION TIMES AS A FUNCTION OF LINKING CONSTRAINT TIGHTNESS

	TIGHTNESS RATIO	
Algorithm	0.33	0.67
DSLC	3.84	9.89
BB	2.13	12.87

TABLE IV

Test Problem 2: RHS = 500%

SOLUTION TIMES AS A FUNCTION OF LINKING CONSTRAINT TIGHTNESS

	TIGHTNESS RATIO			
Algorithm	0.2	0.4	0.6	0.8
DSLC	1.95	4.19	5.14	10.72
BB	1.53	2.01	5.10	> 20

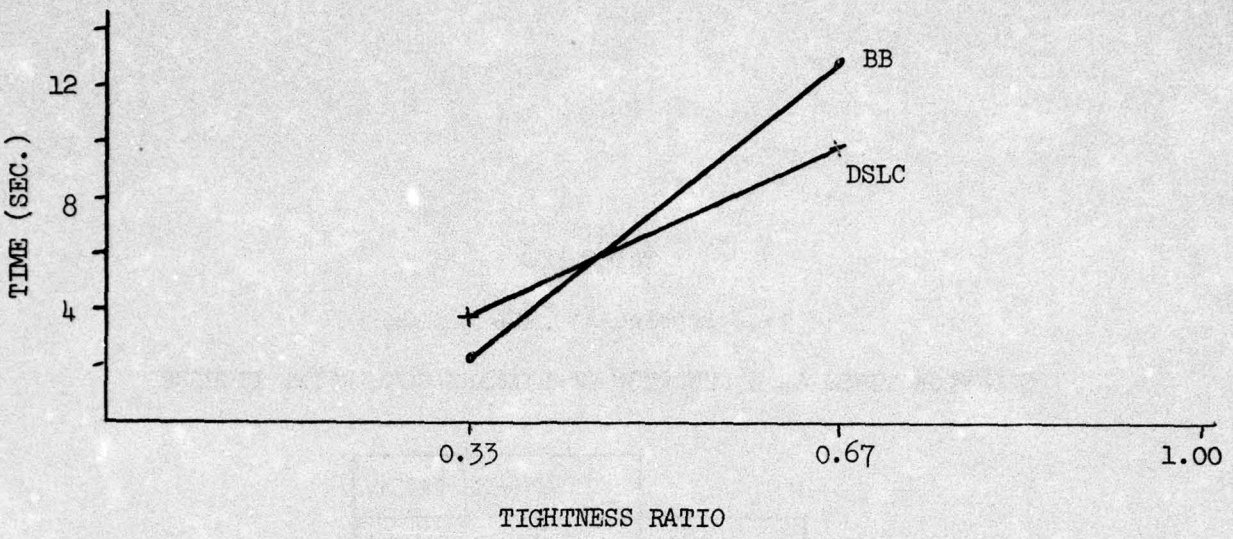

Figure 3: Solution times for Test Problem 2 (RHS = 300%)

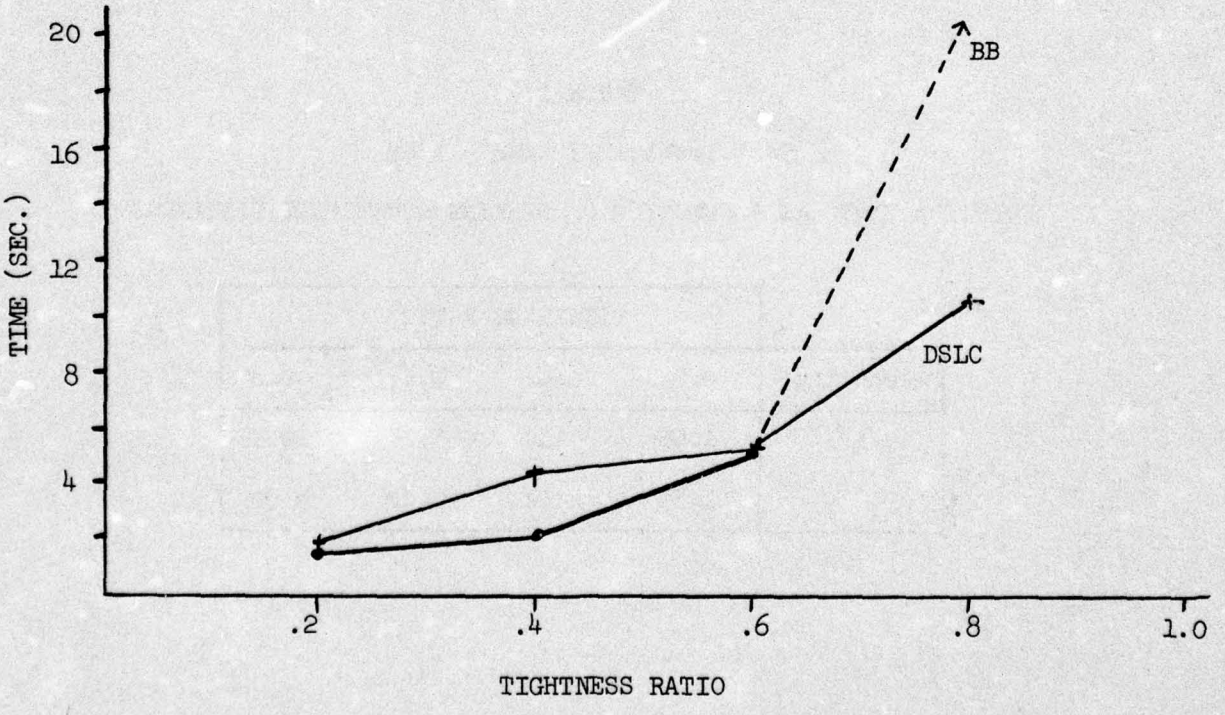

Figure 4: Solution times for Test Problem 2 (RHS = 500%)

problem, each was assigned the cost coefficient from column one of the original problem.

The forty revised projects were next grouped into eight subproblems of five variables and five constraint rows each. The coefficients in the last row of each subproblem were then removed and used to form the linking constraint instead. The resulting block angular integer program thus had four rows and five columns in each of eight subproblems, and one linking constraint. The coefficients defining this problem are given in Table A-III in the appendix.

The performance of the decomposition algorithm in relation to linking constraint tightness was tested on the derivative problem consisting of the first four subproblems of Test Problem 3. The right-hand sides in the subproblem constraints were set to three times the sum of the corresponding row coefficients, and four different values for the tightness ratio TR were tried in turn. The results obtained from both the decomposition and the branch-and-bound codes are presented in Table V and Figure 5.

TABLE V

Test Problem 3: RHS = 300%

SOLUTION TIMES AS A FUNCTION OF LINKING CONSTRAINT TIGHTNESS

	TIGHTNESS RATIO			
Algorithm	0.33	0.67	1.00	1.33
DSLC	1.74	5.03	8.24	6.83
BB	0.83	4.34	15.28	23.54

From these results, it appeared that while the standard branch-and-bound algorithm enjoyed a slight advantage over the decomposition approach for small values of TR, the advantage turned in favor of the latter as TR increased. Since the turning point was observed to be approximately TR = 0.67, it was decided to fix TR at this level in examining the effects on solution times of increasing problem size. Thus, with RHS = 300% again in each subproblem, and TR = 0.67 always, the two additional derivative problems consisting, respectively, of the first two and the first six subproblems of Test Problem 3 were solved. (The four-subproblem derivative, of course, had been solved previously.) The results are shown in Table VI, and are plotted in Figure 6. In view of the unexpectedly large increase in solution times observed in increasing the overall problem size from four to six subproblems, it was decided not to solve the eight-subproblem case.

In each of the six versions of Test Problem 3 that were solved, the quality of the successive incumbent solutions obtained by each of

TABLE VI

Test Problem 3: TR = 0.67

SOLUTION TIMES AS A FUNCTION OF PROBLEM SIZE

	NUMBER OF SUBPROBLEMS		
Algorithm	2	4	6
DSLC	1.34	5.03	26.60
BB	0.78	4.34	53.31

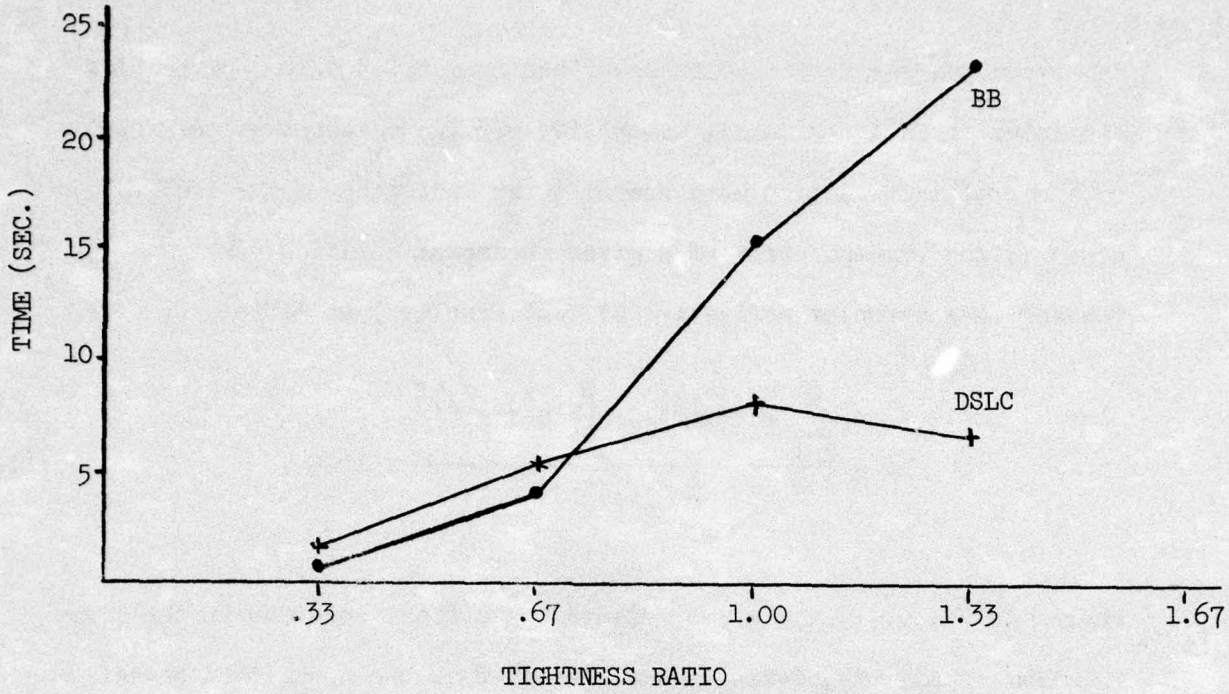

Figure 5: Solution times for Test Problem 3 (RHS = 300%)

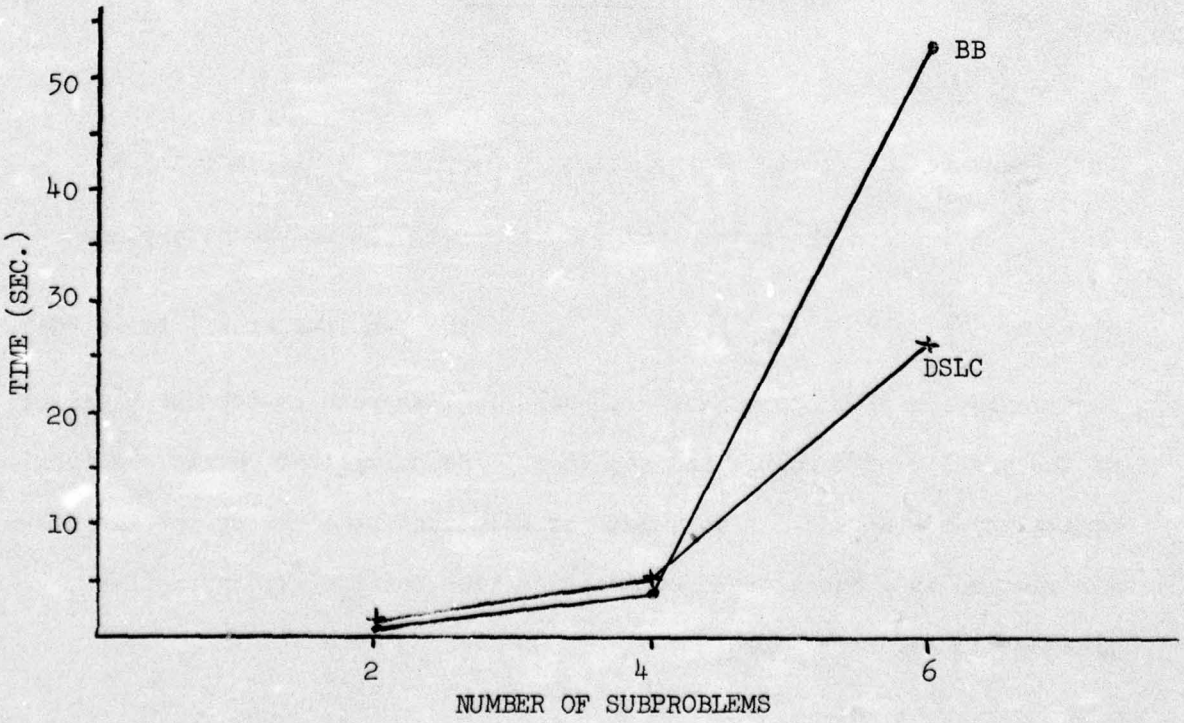

Figure 6: Solution times for Test Problem 3 (TR = 0.67)

the two codes was monitored as described in Section 6.1. The results are shown in Tables VII-XII, respectively. Two measures of the "quality" of a nonoptimal solution were computed, as indicated in the tables. In general, the percent error of a given incumbent solution $(x^{(1)^*},\ldots,x^{(N)^*})$ for any N-subproblem derivative of Test Problem 3 is defined by

$$e \equiv \frac{\sum_{j=1}^{N} c^{(j)}\hat{x}^{(j)} - \sum_{j=1}^{N} c^{(j)}x^{(j)^*}}{\sum_{j=1}^{N} c^{(j)}\hat{x}^{(j)}} \times 100\% , \qquad (1)$$

where $(\hat{x}^{(1)}, \ldots, \hat{x}^{(N)})$ represents the optimal solution to the given problem. Similarly, following Hillier (1969), the normalized deviation from optimality for $(x^{(1)^*}, \ldots, x^{(N)^*})$ is defined by

$$d \equiv \frac{\sum_{j=1}^{N} c^{(j)}\hat{x}^{(j)} - \sum_{j=1}^{N} c^{(j)}x^{(j)^*}}{\sqrt{\sum_{j=1}^{N} c^{(j)}\cdot c^{(j)}}} . \qquad (2)$$

It is easy to show that d represents the Euclidean distance in R^n $(n = \sum_{j=1}^{N} n_j)$ from the point $(x^{(1)^*}, \ldots, x^{(N)^*})$ to the hyperplane given by $\sum_{j=1}^{N} c^{(j)}x^{(j)} = \sum_{j=1}^{N} c^{(j)}\hat{x}^{(j)}$. Of the two, therefore, the normalized deviation from optimality represents a less problem-dependent measure of the quality of a nonoptimal solution. The normalized deviations from optimality of the successive incumbent solutions obtained by the two codes are plotted as a function of computation time for the six derivative problems in Figures 7-12.

TABLE VII

Test Problem 3: Four-Subproblem Derivative, TR = 0.33

QUALITY OF INCUMBENT SOLUTIONS

DSLC

Time	Objective Value	% Error	Norm. Dev. from Opt.
0.28	10397	1.75	.1885
0.39	10582	0.00	.0000

BB

Time	Objective Value	% Error	Norm. Dev. from Opt.
0.40	10505	0.73	.0784
0.56	10539	0.41	.0438
0.65	10573	0.09	.0091
0.67	10582	0.00	.0000

TABLE VIII

Test Problem 3: Four-Subproblem Derivative, TR = 1.00

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.46	14148	0.84	.1222	0.75	12733	10.76	1.5643
2.02	14150	0.83	.1202	0.75	13757	3.58	.5207
5.17	14161	0.75	.1090	0.76	13791	3.34	.4861
5.32	14187	0.57	.0825	0.77	13859	2.87	.4168
5.51	14215	0.37	.0540	0.78	13905	2.54	.3699
6.18	14229	0.27	.0397	0.78	13973	2.07	.3006
6.47	14260	0.06	.0081	0.97	14002	1.87	.2710
7.62	14268	0.00	.0000	1.02	14019	1.75	.2537
				1.11	14058	1.47	.2140
				1.88	14078	1.33	.1936
				3.21	14079	1.32	.1926
				3.22	14147	0.85	.1233
				4.83	14190	0.55	.0794
				5.25	14191	0.54	.0784
				5.41	14199	0.48	.0703
				7.03	14228	0.28	.0407
				7.20	14233	0.25	.0356
				7.64	14234	0.24	.0346
				7.90	14253	0.11	.0152
				11.36	14259	0.06	.0091
				11.78	14260	0.06	.0081
				12.19	14268	0.00	.0000

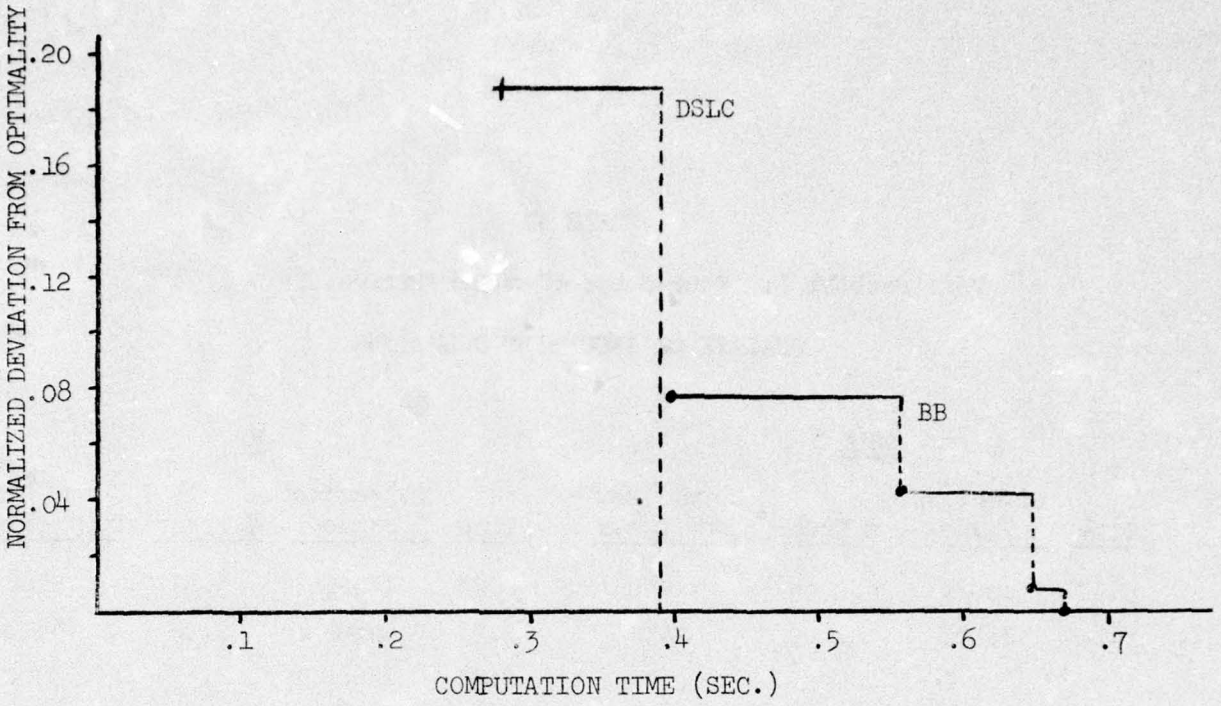

Figure 7: Quality of incumbent solutions, Test Problem 3 (four-subproblem derivative, TR = 0.33)

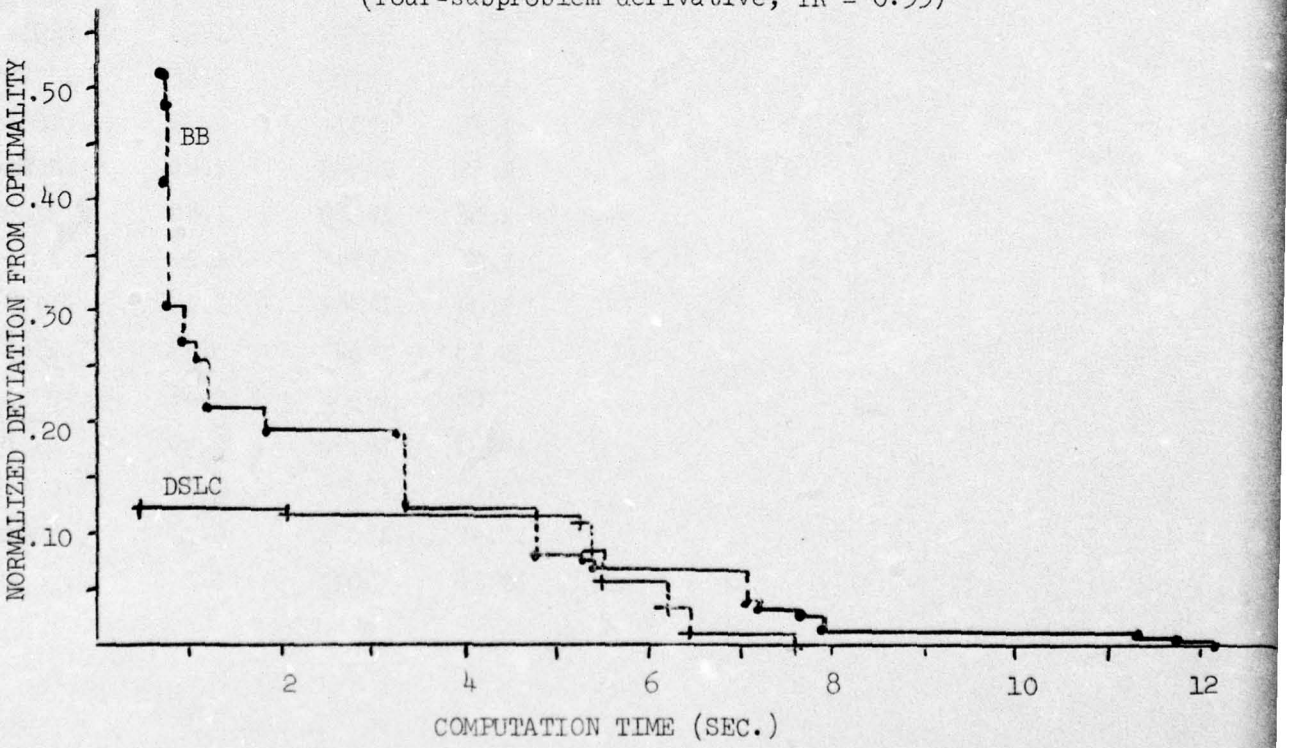

Figure 8: Quality of incumbent solutions, Test Problem 3 (four-subproblem derivative, TR = 1.00)

TABLE IX

Test Problem 3: Four-Subproblem Derviative, TR = 1.33

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.40	15025	0.05	.0081	0.73	14526	3.37	.5166
3.60	15033	0.00	.0000	0.75	14594	2.92	.4473
				0.78	14662	2.47	.3780
				0.87	14708	2.16	.3312
				0.94	14725	2.05	.3138
				1.03	14747	1.90	.2914
				1.25	14760	1.82	.2782
				1.27	14774	1.72	.2639
				1.37	14185	1.45	.2221
				1.68	14189	1.42	.2180
				1.76	14843	1.26	.1936
				5.04	14862	1.14	.1742
				5.43	14921	0.75	.1141
				7.62	14964	0.46	.0703
				10.83	14990	0.29	.0438
				10.87	15007	0.17	.0264
				17.40	15029	0.03	.0040
				17.58	15033	0.00	.0000

TABLE X

Test Problem 3: Two-Subproblem Derivative, TR = 0.67

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.77	6436	0.97	.0899	0.19	6141	5.51	.5112
0.79	6499	0.00	.0000	0.20	6209	4.46	.4141
				0.21	6255	3.75	.3484
				0.23	6340	2.45	.2270
				0.31	6352	2.26	.2099
				0.36	6369	2.00	.1856
				0.47	6377	1.88	.1742
				0.48	6445	0.83	.0771
				0.68	6482	0.26	.0242
				0.71	6499	0.00	.0000

TABLE XI

Test Problem 3: Four-Subproblem Derivative, TR = 0.67

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.36	13083	0.62	.0835	0.55	13034	1.00	.1335
1.65	13137	0.21	.0285	0.61	13057	0.82	.1100
1.72	13143	0.17	.0224	0.73	13077	0.67	.0896
1.83	13157	0.06	.0081	0.76	13151	0.11	.0142
3.70	13165	0.00	.0000	1.22	13165	0.00	.0000

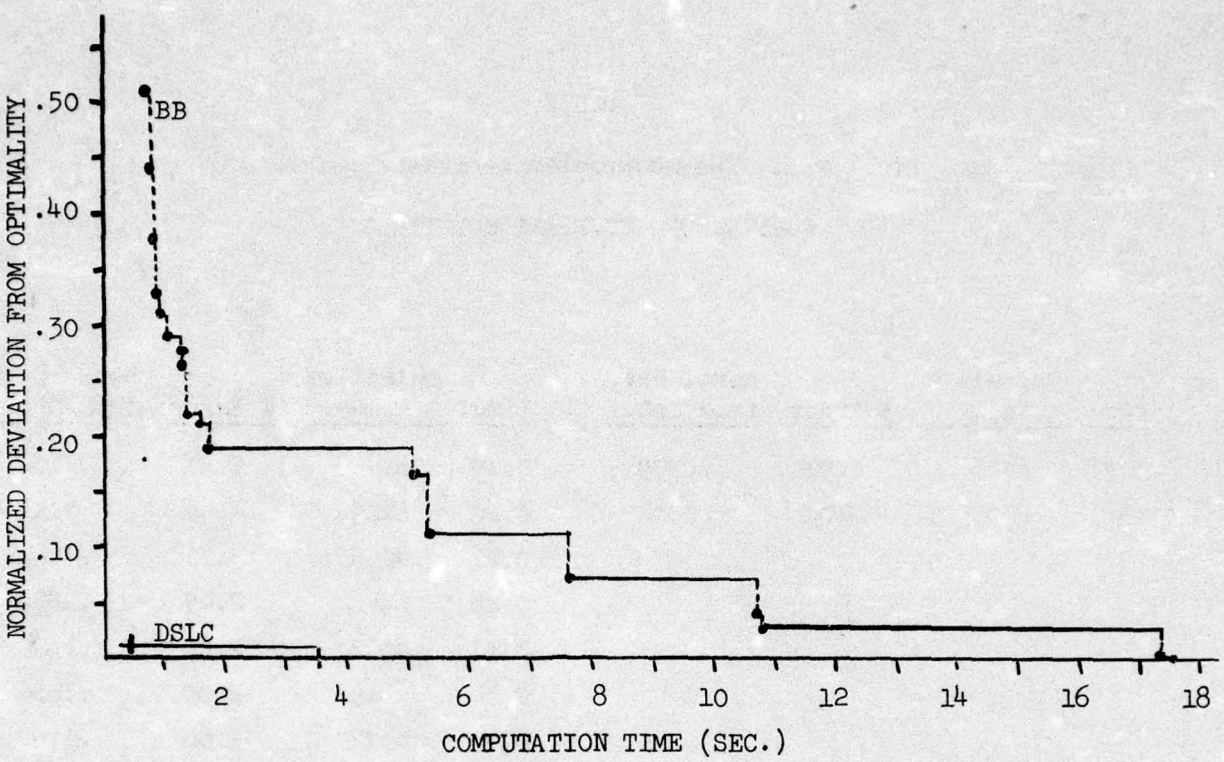

Figure 9: Quality of incumbent solutions, Test Problem 3 (four-subproblem derivative, TR = 1.33)

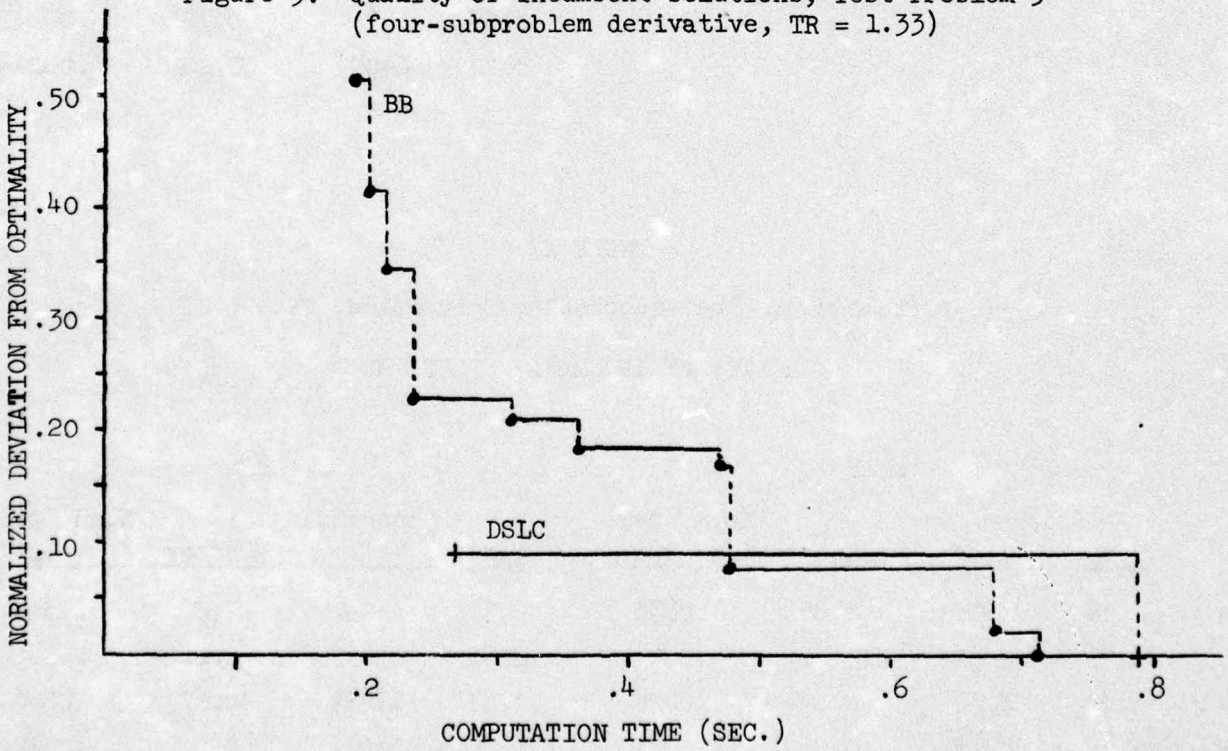

Figure 10: Quality of incumbent solutions, Test Problem 3 (two-subproblem derivative, TR = 0.67)

TABLE XII

Test Problem 3: Six-Subproblem Derivative, TR = 0.67

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.64	17923	1.46	.2302	1.29	16604	8.71	1.3764
1.21	17986	1.11	.1755	1.30	16938	6.87	1.0862
2.51	18037	0.83	.1312	1.32	17246	5.18	.8185
2.89	18151	0.20	.0321	1.33	17791	2.18	.3449
4.17	18188	0.00	.0000	1.35	17834	1.95	.3076
				1.35	17877	1.71	.2702
				1.36	18099	0.49	.0773
				1.37	18119	0.38	.0599
				2.04	18173	0.08	.0130
				4.36	18188	0.00	.0000

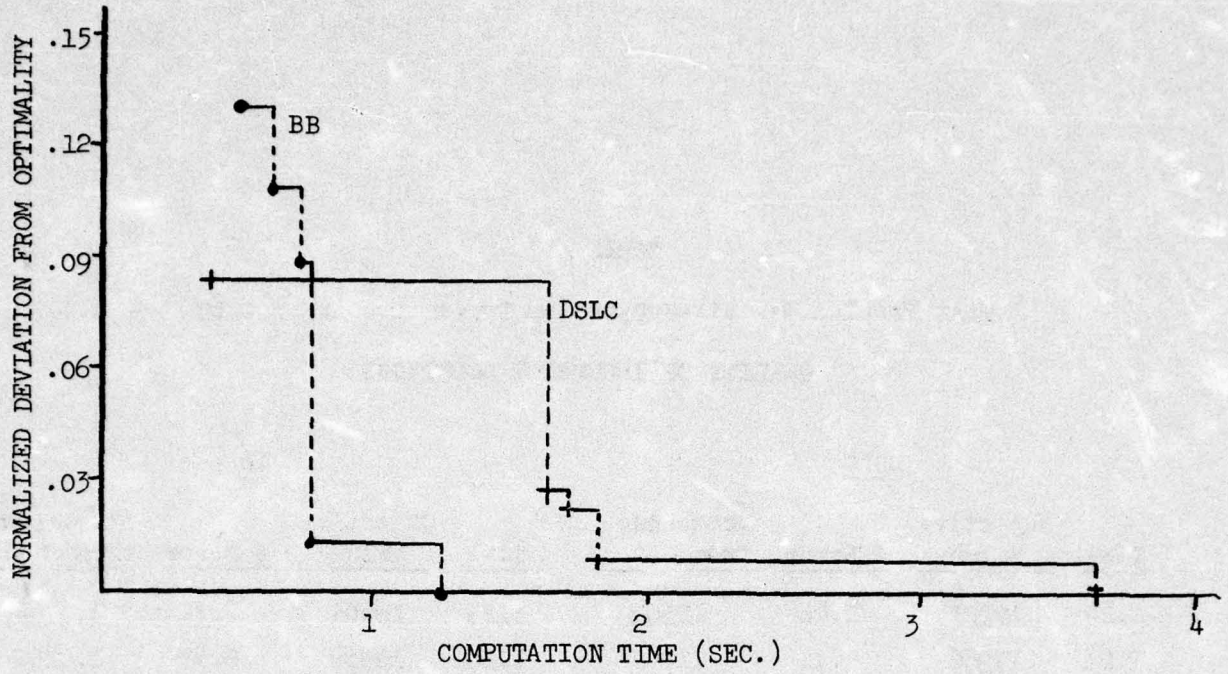

Figure 11: Quality of incumbent solutions, Test Problem 3 (four-subproblem derivative, TR = 0.67)

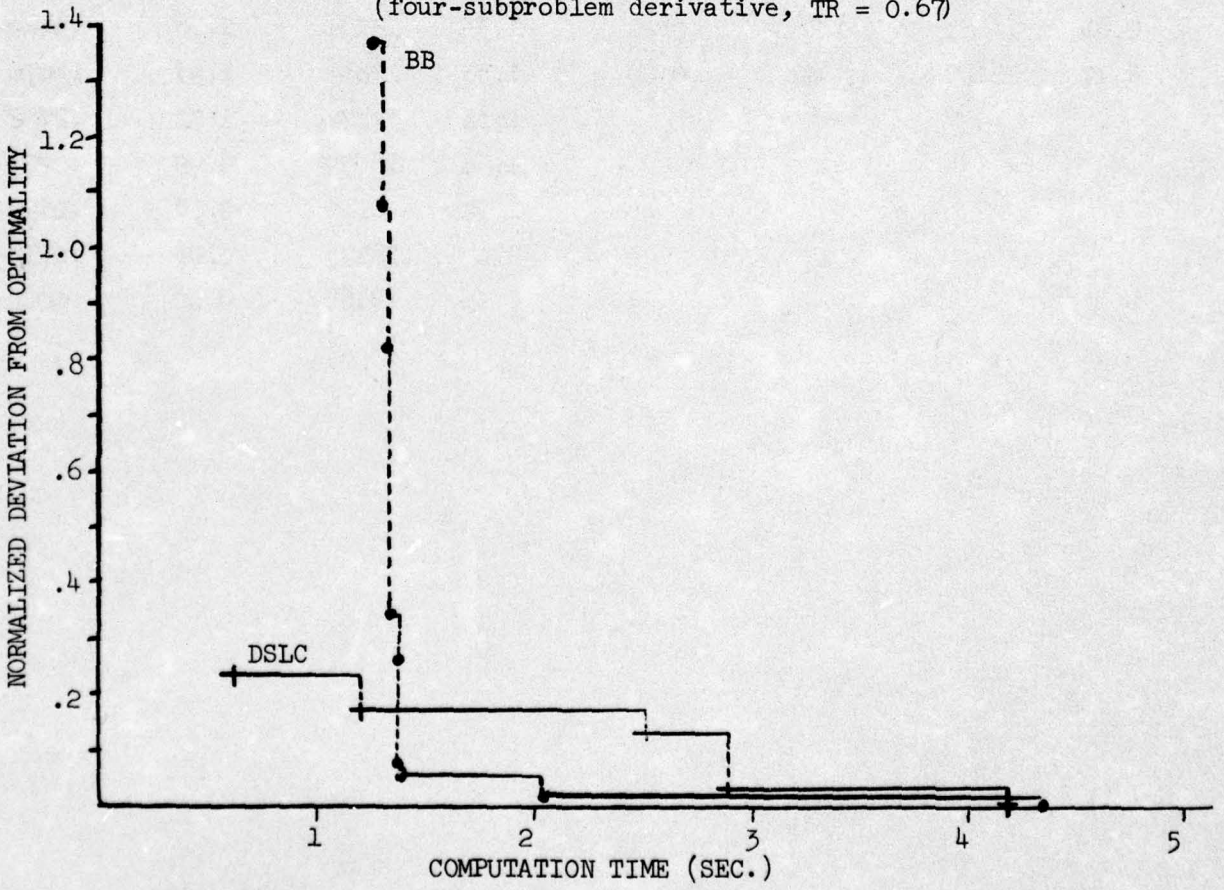

Figure 12: Quality of incumbent solutions, Test Problem 3 (six-subproblem derivative, TR = 0.67)

6.4 Test Problem 4

The data for the fourth test problem was taken from the seventh of the R & D project selection problems reported by Petersen (1967). To obtain a problem with the block angular structure, Petersen's 50-variable, 5-constraint problem was divided into ten subproblems of five columns and five rows each. The coefficients in the first constraint in each subproblem were then removed and used to form the linking constraint instead. Finally, each of the objective coefficients was divided by ten, and rounded to the nearest integer. Test Problem 4 is exhibited in Table A-IV in the appendix.

Because the decision variables in many practical integer programming problems are restricted to take on only the values zero or one, it was felt that the testing of the decomposition algorithm should include at least one such problem. Consequently, in each of the (five) derivative problems of Test Problem 4, unit upper bounds were specified on all variables. The subproblem constraint right-hand sides were fixed at (approximately) RHS = 70%. (This was the setting used in the zero-one test problems given by Reardon (1974).) Correspondingly, the linking constraint right-hand side was chosen always to be approximately 40% of the sum of the linking constraint coefficients, so that TR $\approx$ 0.57 in each of the derivative problems.

The five derivative problems solved consisted, respectively, of the first 2, 4, 6, 8 and 10 (all) of the subproblems of Test Problem 4. The total solution times required by the decomposition and the standard branch-and-bound algorithms are given in Table XIII, and these values are plotted in Figure 13. Although the decomposition procedure had

located the optimal solution for the last derivative problem in less than 31 seconds, it had still failed to verify the optimality of this solution after 60 seconds (total) of CPU usage, and computations were terminated prematurely.

As with Test Problem 3, the quality of the incumbent solutions produced by each of the two codes was observed in each of the five derivative problems. With "percent error" and "normalized deviation from optimality" again defined by (1), (2), these intermediate results are shown in Tables XIV-XVIII. The normalized deviation from optimality of the successive incumbent solutions is plotted as a function of computation time in Figures 14-18.

6.5 Summary of Computational Results

The results obtained in the first three test problems (specifically, those plotted in Figures 1-4) support the conjecture made in Section 6.1; for these problems, the decomposition algorithm has enjoyed its greatest computational advantage over the nondecomposition approach for the larger test values of the tightness ratio TR. In particular, in our limited experimental sample, the ratio TR = 0.67 emerged as the level of linking constraint tightness above which the decomposition procedure was consistently and significantly more effective than the standard branch-and-bound algorithm. However, the solution times required by the decomposition method were always competitive with, and often superior to, those required by the branch-and-bound algorithm for tightness ratios even as low as TR = 0.2.

The results of experiments with increasing problem size in Test Problems 3 and 4 were less conclusive. Figures 6 and 13 indicate that

TABLE XIII

Test Problem 4: TR = 0.57

SOLUTION TIMES AS A FUNCTION OF PROBLEM SIZE

	NUMBER OF SUBPROBLEMS				
Algorithm	2	4	6	8	10
DSLC	0.24	0.61	1.69	5.99	> 60
BB	0.15	0.87	5.16	16.64	29.43

TABLE XIV

Test Problem 4: Two-Subproblem Derivative, TR = 0.57

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.15	230	9.80	.0959	0.12	243	4.71	.0460
0.22	255	0.00	.0000	0.13	253	0.78	.0076
				0.14	255	0.00	.0000

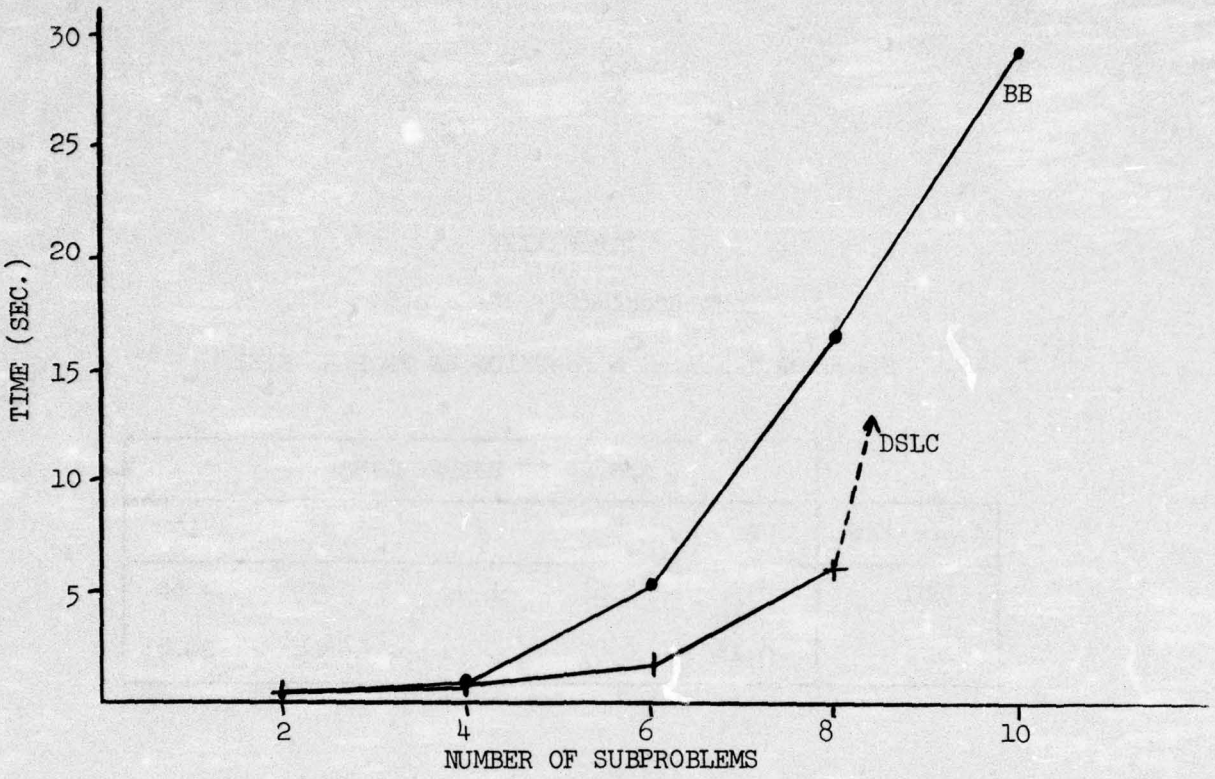

Figure 13: Solution times for Test Problem 4 (TR = 0.57)

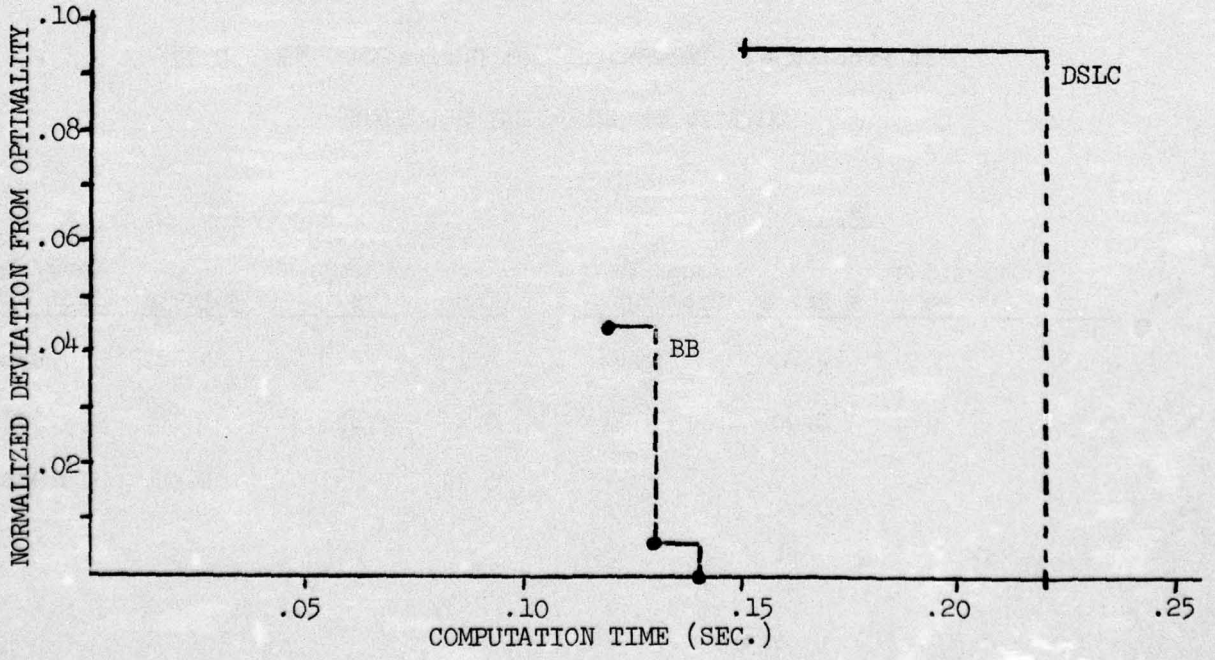

Figure 14: Quality of incumbent solutions, Test Problem 4 (two-subproblem derivative, TR = 0.57)

TABLE XV

Test Problem 4: Four-Subproblem Derivative, TR = 0.57

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.29	442	4.33	.0396	0.44	414	10.39	.0951
0.37	450	2.60	.0237	0.46	431	6.71	.0614
0.47	452	2.16	.0198	0.51	445	3.68	.0336
0.47	455	1.52	.0138	0.51	462	0.00	.0000
0.57	457	1.08	.0099				
0.60	462	0.00	.0000				

TABLE XVI

Test Problem 4: Six-Subproblem Derivative, TR = 0.57

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.40	537	3.94	.0430	0.96	559	0.00	.0000
0.73	541	3.22	.0352				
0.74	544	2.68	.0293				
1.19	545	2.50	.0273				
1.21	546	2.32	.0254				
1.27	552	1.25	.0136				
1.37	553	1.07	.0117				
1.38	554	0.89	.0097				
1.48	556	0.54	.0058				
1.51	559	0.00	.0000				

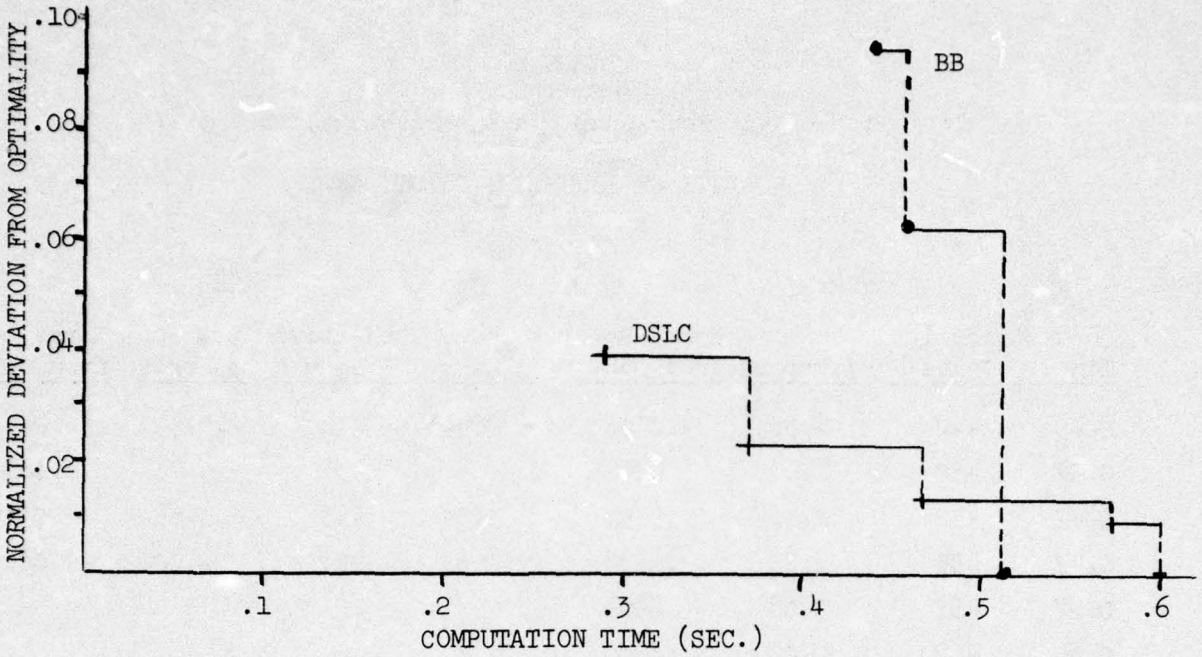

Figure 15: Quality of incumbent solutions, Test Problem 4 (four-subproblem derivative, TR = 0.57)

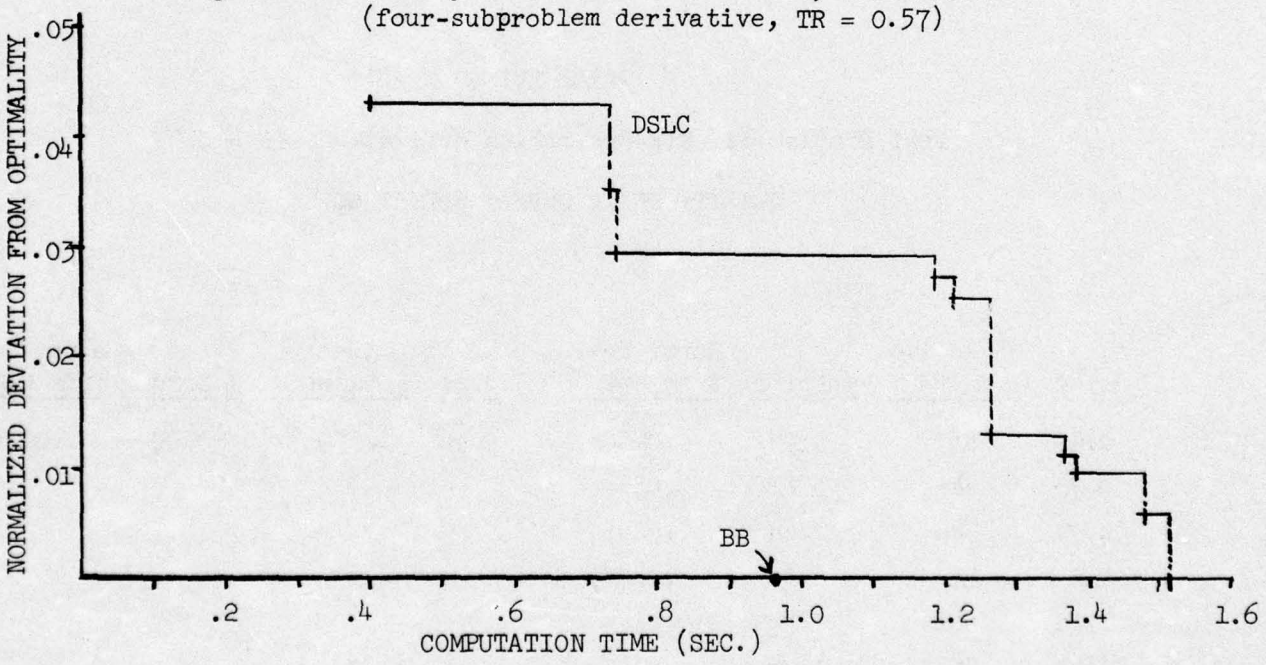

Figure 16: Quality of incumbent solutions, Test Problem 4 (six-subproblem derivative, TR = 0.57)

TABLE XVII

Test Problem 4: Eight-Subproblem Derivative, TR = 0.57

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm. Dev. from Opt.
0.52	667	2.78	.0365	1.85	675	1.60	.0211
0.80	670	2.33	.0307	1.95	677	1.31	.0173
1.08	672	2.04	.0269	1.96	680	0.87	.0115
1.31	674	1.75	.0230	2.72	681	0.73	.0096
1.64	675	1.60	.0211	2.85	684	0.29	.0038
1.72	676	1.46	.0192	5.56	686	0.00	.0000
1.76	677	1.31	.0173				
1.79	679	1.02	.0134				
1.90	680	0.87	.0115				
1.94	682	0.58	.0076				
3.07	683	0.44	.0057				
3.19	686	0.00	.0000				

TABLE XVIII

Test Problem 4: Ten-Subproblem Derivative, TR = 0.57

QUALITY OF INCUMBENT SOLUTIONS

DSLC				BB			
Time	Objective Value	% Error	Norm. Dev. from Opt.	Time	Objective Value	% Error	Norm Dev. from Opt.
0.65	984	5.93	.0973	2.84	900	13.96	.2292
0.72	1031	1.43	.0235	2.88	917	12.33	.2025
0.85	1034	1.15	.0188	2.97	922	11.85	.1947
0.96	1038	0.76	.0125	3.20	936	10.52	.1727
1.71	1040	0.57	.0094	3.32	953	8.89	.1460
3.93	1041	0.48	.0078	3.69	973	6.98	.1146
10.37	1042	0.38	.0062	3.71	975	6.79	.1114
10.39	1043	0.29	.0047	3.76	980	6.31	.1036
10.41	1045	0.10	.0015	4.59	981	6.21	.1020
30.80	1046	0.00	.0000	6.40	982	6.12	.1005
				8.63	984	5.93	.0973
				22.12	1008	3.63	.0596
				22.13	1041	0.48	.0078
				22.16	1043	0.29	.0047
				22.59	1045	0.10	.0015
				22.82	1046	0.00	.0000

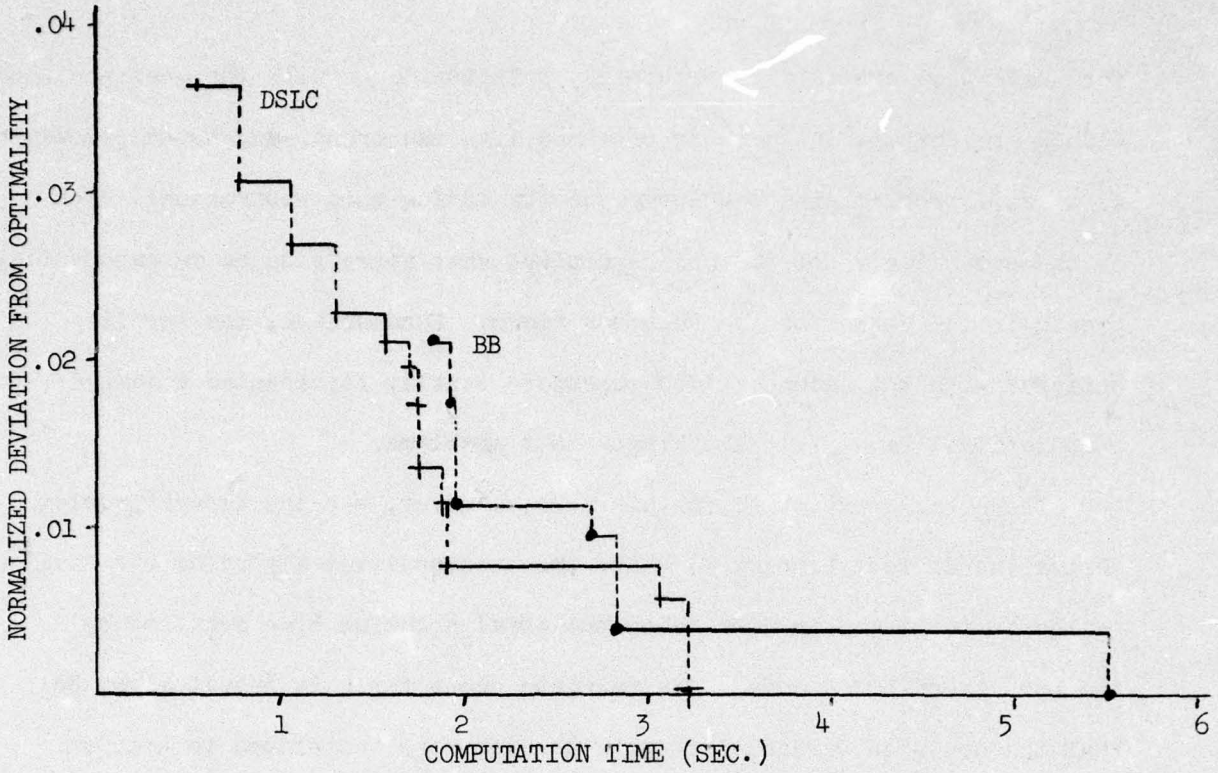

Figure 17: Quality of incumbent solutions, Test Problem 4 (eight-subproblem derivative, TR = 0.57)

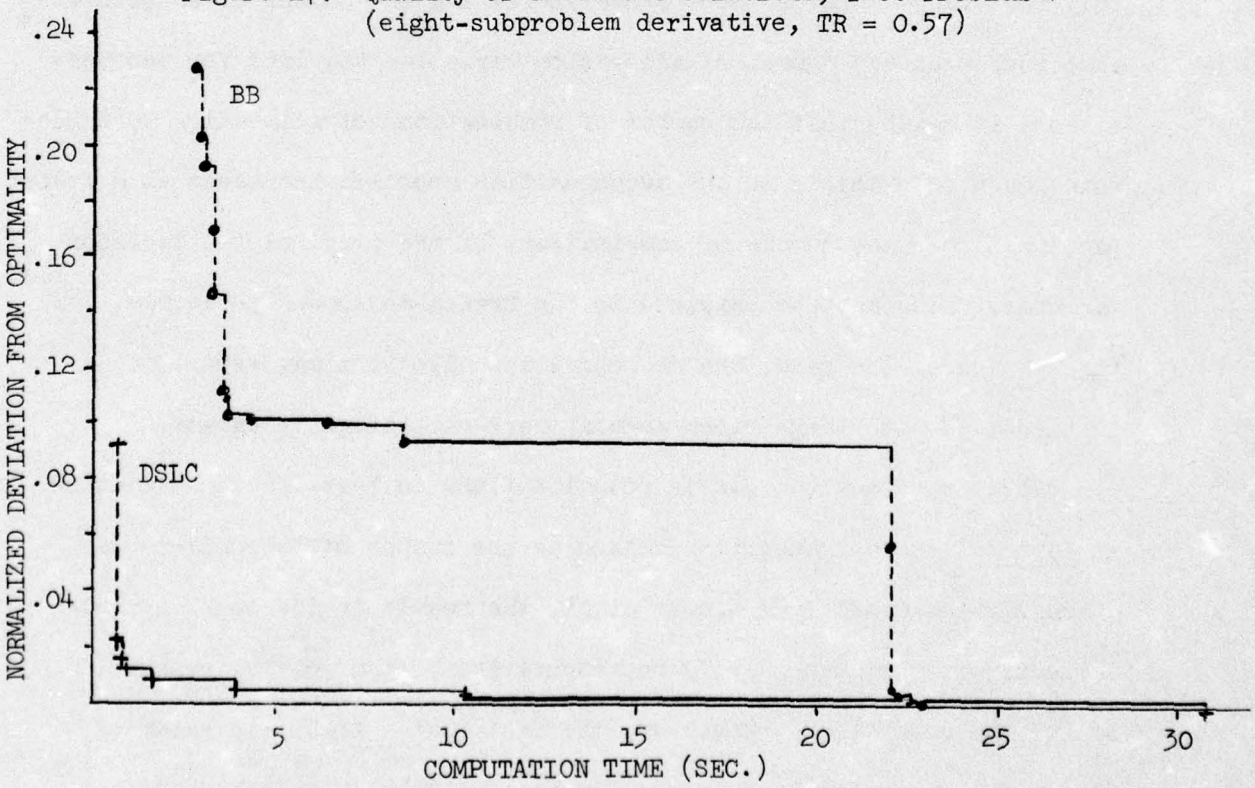

Figure 18: Quality of incumbent solutions, Test Problem 4 (ten-subproblem derivative, TR = 0.57)

for small problems (few subproblems), solution times with the decomposition method are comparable to those obtained with the branch-and-bound procedure. As overall problem size was increased (by adding more subproblems, each of the same size), both methods exhibited what appears to be an exponential growth in the corresponding solution times. Nonetheless, the results obtained with the decomposition procedure usually represented a marked computational savings in the larger test problems.

The notable exception to this rule, however, was the ten-subproblem derivative of Test Problem 4, where the decomposition algorithm had still failed to terminate in over twice the total solution time required by the branch-and-bound code. One possible cause for this result might be that the decision variables in this problem were restricted to take on only the values zero or one. Since increasing the number of subproblems also increases the number of allocation variables requires for decomposition, it may be that the number of combinations of allocation variables which must be examined in the decomposition approach increases at a faster pace than does the number of combinations of the original 0-1 decision variables which must be examined in the branch-and-bound procedure. If this is indeed the case, the decomposition algorithm may experience difficulties with large block angular zero-one integer programs.

However, since the gap in solution times in Test Problem 4 widened in favor of the decomposition method as the number of subproblems was gradually increased from two to eight, the result in the ten-subproblem derivative is not necessarily representative of the general behavior of the two algorithms. Moreover, the fact that a tightness ratio of TR = 0.57 was maintained in the fourth test problem must also be

considered. This level of linking constraint tightness is below the "critical" value of $TR = 0.67$ that was identified in the first three test problems; while the decomposition method was consistently superior to the branch-and-bound algorithm when $TR > 0.67$, neither method enjoyed a clear-cut advantage in test problems with $TR \leq 0.67$. In this light, the results in Test Problem 4 may equally well be considered as an indication that the degree of linking constraint tightness is a more crucial factor in determining the consistency of the effectiveness of decomposition than is problem size. Verification of this hypothesis in further computational tests has important implications for the user who must select a single method for a given application, since the tightness ratio TR is a simple measure which can be readily approximated from problem data alone.

Finally, the intermediate results obtained in Test Problems 3 and 4 (see Figures 7-12 and 14-18) would strongly encourage the use of the decomposition algorithm as a heuristic procedure for block angular integer programs which may be too large to solve optimally, or for which an exact optimal solution is not required. As a measure of the effectiveness of decomposition in quickly identifying good (suboptimal) solutions, let t_1 denote the computation time required by the decomposition algorithm to find a first feasible solution to a given problem, and correspondingly, let t_2 denote the time required by the branch-and-bound algorithm to reach a feasible solution of the same quality. In the various versions of Test Problems 3 and 4 that were solved, t_2 was observed to range from 2 to 43 times t_1.[1/] In particular, in the two largest derivatives

[1/] The single exception was the two-subproblem derivative of Test Problem 4. Because of its small size, however, this problem is so easily solved optimally that the use of a heuristic procedure would be questionable in any event.

of these problems - the first with 30 general integer variables and the second with 50 zero-one variables overall - the decomposition procedure reached a first feasible solution in 0.64 and 0.65 seconds, respectively. By comparison, the branch-and-bound algorithm required 1.35 and 8.63 seconds, respectively, to locate integer feasible solutions of approximately the same quality. These results are especially encouraging in view of the fact that the corresponding tightness ratios of TR = 0.67 and TR = 0.57 were fairly small.

In general, the decomposition procedure consistently and quickly produced initial feasible solutions of high quality. Moreover, its advantage over the branch-and-bound algorithm in this regard increased with increasing problem difficulty. In Test Problem 4 in particular, t_1 was observed to increase approximately linearly with increasing problem size. Recall that when the block angular integer program (P) is decomposed into its component subproblems, the work required to generate a first incumbent essentially involves the solution of N parametric linear programs, followed by the solution of N integer programming problems. If the total problem size is doubled, therefore, the work to find a first feasible solution involves not the solution of linear and integer programming problems of twice the original size, but rather the solution of twice as many linear and integer programs of the same size. Hence, there is reason to expect that, at least in the single linking constraint case, the observed linear increase in the computation time required by the decomposition algorithm to obtain a first feasible solution will extend to other, larger block angular integer programs as well.

CHAPTER 7

CONCLUSIONS, EXTENSIONS, AND AREAS FOR FUTURE RESEARCH

7.1 Conclusions

The computational results reported in Chapter 6 are too few to draw any final conclusions concerning the performance of the decomposition algorithm. Nonetheless, these first experiments would indicate that our procedure for the decomposition of block angular integer programs can be an attractive alternative to standard integer programming methods which ignore problem structure. In particular the decomposition approach has been most effective in treating problems in which the linking constraints are fairly loose relative to the subproblem constraints; in terms of the measure of relative linking constraint tightness TR defined in Section 6.1, solution times exhibited by the decomposition algorithm in problems with $TR > 0.67$ were 45-70% less than those required by a nondecomposition approach to the same problems. Thus, it may well be possible to determine in advance (by approximating TR from the problem data) which problems can be most efficiently solved through the decomposition approach.

Also encouraging was the observation that computation times required by the decomposition algorithm usually experienced a significantly less rapid growth with increasing problem size than did the corresponding times for the standard branch-and-bound procedure. There was one major exception to this trend, however, and more extensive computational testing of the algorithm on other and larger block angular problems would be

desirable. An investigation of the effects of different levels of linking constraint tightness on solution times in these larger problems is especially important. Moreover, since the results reported in Chapter 6 were obtained with block angular problems having only a single linking constraint, there is also the need to test the performance of the decomposition algorithm in the multiple linking constraints case.

Finally, our algorithm rapidly and consistently generated initial feasible solutions of high quality. Especially in the single linking constraint case, there is reason to believe that the computation time required to reach a first feasible solution will grow approximately linearly with the number of subproblems. Consequently, the decomposition procedure also can be valuable as a heuristic method for quickly locating good (suboptimal) solutions to large block angular integer programs.

7.2 Extensions and Areas for Future Research

It is hoped that the present work will be regarded as only a first step towards the development of efficient decomposition procedures for block angular integer programming problems, and as such, will serve to motivate continued and original efforts in this area. Several possible modifications to the decomposition algorithm already have been described in Section 5.3. Of these, the development of a system of penalties for use in association with the branching process is perhaps of greatest potential value. Historically, the introduction of the penalty concept in standard (nondecomposition) branch-and-bound integer programming methods has resulted in significant improvements in the efficiency of these methods. The derivation and incorporation of penalty calculations in the decomposition algorithm might best be combined with an investigation

of alternative branching strategies as well. One such alternative was discussed briefly in Section 5.3.

There is also the need for continued computational analysis of the performance of the decomposition algorithm. In addition to the problem parameters investigated in Chapter 6, two others which may have important effects on algorithmic efficiency are the number of linking constraints and (average) subproblem size. It would be extremely useful from a practical viewpoint to identify the ranges of these parameters for which the decomposition procedure is most effective.

The algorithm for decomposing block angular integer programs often can require that several different versions of each of the subproblems be solved. Since these versions will differ from one another only in the right-hand sides of the linking constraints, it may well turn out that significant computational savings can be achieved by incorporating techniques for solving these subproblems parametrically, or for conducting sensitivity analyses on the optimal subproblem solutions. Recently, many advances have been made in the areas of parametric integer programming and postoptimality analysis in integer programming.[1/] Such techniques may be particularly valuable for block angular integer programs in which the subproblems themselves are fairly large, since greater proportions of the total solution time for these problems will be spent in generating the subproblem solutions.

A closely related issue is the question of optimal degree of decomposition. In the algorithm developed here, each of the subproblems is

---

[1/] See, for example, Geoffrion and Nauss (1976), Piper and Zoltners (1975), (1976), Radke (1975), Roodman (1972), and Nauss (1975).

treated individually. In a sense, the straightforward branch-and-bound procedure is simply an alternative version of the decomposition algorithm in which all the subproblems are combined and treated as one. Intermediate versions between these two extremes are certainly possible as well. Since the straightforward branch-and-bound algorithm proved slightly superior to the decomposition approach in the two-subproblem derivatives of Test Problems 3 and 4, it seems reasonable to consider, for example, pairing individual subproblems and treating each such combined pair as a single subproblem when the decomposition algorithm is applied. In general, it would be interesting to compare solution times resulting from various different strategies for grouping the subproblems. Since partial decomposition approaches necessarily will lead to increased subproblem sizes, the incorporation of efficient parametric integer programming capabilities may well be an important factor in determing the relative effectiveness of different degrees of decomposition.

Underlying the branching procedure in the decomposition algorithm is the assumption that the allocation variables can take on only discrete values. In the present work, we have guaranteed that this will always be the case by assuming that 1) all linking constraint coefficients in the block angular integer program (P) are integral,[1/] and 2) all decision

---

[1/] Rational coefficients can always be made integral. Moreover, since digital computers can represent problem coefficients only with finitely many decimal places, in principle it is always possible to render all coefficients integral by scaling appropriately. However, because the magnitude of the resulting (integral) coefficients may be quite large, this approach to circumventing assumption 1) can lead to serious difficulties with the numerical stability of the algorithm.

variables are integer-restricted. As was previously mentioned, in certain applications physical considerations may naturally imply that the allocation variables can be meaningfully interpreted only at discrete levels (e.g., if they actually represent resources which can be subdivided only in integral amounts). In such cases, our method is applicable even when assumptions 1) and 2) are not satisfied. Nonetheless, a valuable extension in the range of problems for which the decomposition algorithm is suitable would be obtained if one or both of these conditions could be dropped. The second assumption in particular rules out the important class of mixed-integer programming problems. Benders (1962) has developed a procedure for decomposing general mixed-integer programming problems into their integer and linear programming component parts, and a similar approach might be considered here. The resulting integer programming component, however, must retain its block angular structure without violating the first assumption above.

The multi-stage integer program (Q) exhibited below is quite similar in structure to the block angular problem.

(Q)

$$\text{maximize} \quad c^{(1)}x^{(1)}+c^{(2)}x^{(2)}+c^{(3)}x^{(3)}+\cdots+c^{(N-1)}x^{(N-1)}+c^{(N)}x^{(N)} ,$$

subject to

$$\begin{aligned}
A^{(1)}x^{(1)}+B^{(1)}x^{(2)} &\leq \beta^{(1)} \\
A^{(2)}x^{(2)}+B^{(2)}x^{(3)} &\leq \beta^{(2)} \\
\ddots \qquad &\vdots \\
A^{(N-1)}x^{(N-1)}+B^{(N-1)}x^{(N)} &\leq \beta^{(N-1)}
\end{aligned}$$

$$x^{(j)} \geq 0, \text{ integer}, \quad j = 1, 2, \ldots, N .$$

By properly reordering the constraint rows, (Q) can be put into block angular form. This can be done, for example, by moving alternate levels $A^{(j)}x^{(j)} + B^{(j)}x^{(j+1)} \leq \beta^{(j)}$ (j = 1, 3, 5, etc.) to the top to form the linking constraints. The remaining levels $A^{(j)}x^{(j)} + B^{(j)}x^{(j+1)} \leq \beta^{(j)}$ (j = 2, 4, 6, etc.) then form the subproblems' constraints. The decomposition algorithm for block angular integer programs can thus be applied to problem (Q). Some effort should be made, however, to exploit the special structure of the linking constraints themselves in such problems. For example, each subproblem will have all-zero coefficients in many of the linking constraints. The total number of allocation variables required for decomposition can be reduced by noting that the allocation of the i^{th} linking resource to the j^{th} subproblem can obviously be fixed at zero if the coefficients corresponding to this subproblem are identically zero in the i^{th} linking constraint.

Alternatively, problem (Q) can be put into dual angular form by reordering the decision variables so that, for example, the columns corresponding to $x^{(j)}$ (j = 1, 3, 5, etc.) appear in the left-most positions in the revised constraint matrix. These variables then form the linking columns, while the subproblems are comprised of the remaining variables. Reardon (1974) has developed an efficient decomposition method for dual angular integer programming problems. The linking columns in the dual angular problem obtained from the multi-stage problem (Q) will again have a special structure of their own, and some effort should be made to exploit this fact. It would be interesting to compare the performance of Reardon's method with that of the algorithm presented here when each is applied to problems of the form (Q).

Finally, the doubly-coupled integer program is also closely related to both the block angular and dual angular integer programming problems. These problems are characterized by a block diagonal matrix of constraint coefficients in which both linking rows and linking columns are present. Consequently, the development of an algorithm combining Reardon's method with the one developed here would seem appropriate. One such approach might begin by treating the problem as a dual angular integer program with one (large) subproblem. In accordance with Reardon's algorithm, the linking variables are assigned values first, whereupon decomposition into independent subproblems occurs. In this case, of course, there is only a single subproblem. However, this subproblem itself has a block angular structure, and so may be solved by the method developed in the present work.

A P P E N D I X

TABLE A-I

Test Problem 1

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(1)}$	20	20	15	15	20	
$A^{(1)}$	8	8	3	5	5	
$B^{(1)}$	80	55	70	40	80	975
	100	10	100	4	90	912
	20	20	20	14	30	312
	40	30	30	29	40	507
	50	35	40	24	40	582

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(2)}$	20	10	10	6	6	
$A^{(2)}$	8	8	3	5	5	
$B^{(2)}$	40	45	21	15	9	390
	10	75	0	25	6	348
	20	8	6	3	12	147
	20	16	12	5	12	195
	20	19	16	7	15	231

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(3)}$	8	8	29	29	20	
$A^{(3)}$	8	8	3	5	5	
$B^{(3)}$	28	18	90	29	130	885
	28	0	120	6	130	852
	12	10	14	18	40	282
	18	10	24	30	60	426
	18	10	29	30	70	471

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(4)}$	20	7	7	5	5	
$A^{(4)}$	8	8	3	5	5	
$B^{(4)}$	70	32	21	20	17	480
	32	32	3	40	0	321
	42	6	9	3	12	216
	42	16	18	11	18	315
	42	21	20	17	18	354

RHS = 300%

	Tightness Ratio		
	0.33	1.00	1.67
Linking Constraint Right-hand Side	116	348	580

RHS = 500%

	Tightness Ratio			
	0.2	0.4	0.6	0.8
Linking Constraint Right-hand Side	116	232	348	464

[1] These right-hand sides represent the RHS = 300% case. The RHS = 500% case is obtained by multiplying these values by 5/3.

TABLE A-II

Test Problem 2

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(1)}$	65	65	33	33	16	
$A^{(1)}$	8	8	3	5	5	
$B^{(1)}$	120	35	40	25	30	750
	160	70	40	10	60	1020
	20	100	5	20	0	435
	30	110	25	20	10	585
	30	120	25	20	15	630

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(2)}$	16	24	24	4	4	
$A^{(2)}$	8	8	3	5	5	
$B^{(2)}$	20	20	25	6	5	228
	0	55	0	10	0	195
	5	5	6	3	4	69
	15	13	18	5	7	174
	20	25	22	5	7	237

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(3)}$	3	3	130	130	155	
$A^{(3)}$	8	8	3	5	5	
$B^{(3)}$	3	2	180	110	220	1545
	6	0	240	0	290	1608
	0	1	20	20	30	213
	1	2	80	40	60	549
	1	3	100	50	70	672

	COLUMNS					
	1	2	3	4	5	RHS[1]
$c^{(4)}$	155	55	55	48	48	
$A^{(4)}$	8	8	3	5	5	
$B^{(4)}$	70	50	55	30	20	675
	30	80	10	90	0	630
	50	40	30	10	5	405
	60	50	50	20	25	615
	60	55	55	20	25	645

RHS = 300%

	Tightness Ratio	
	0.33	0.67
Linking Constraint Right-hand Side	116	232

RHS = 500%

	Tightness Ratio			
	0.2	0.4	0.6	0.8
Linking Constraint Right-hand Side	116	232	348	464

[1] These right-hand sides represent the RHS = 300% case. The RHS = 500% case is obtained by multiplying these values by 5/3.

TABLE A-III

Test Problem 3

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(1)}$	245	245	177	177	291	
$A^{(1)}$	11	41	62	68	43	
$B^{(1)}$	77	28	12	85	57	777
	21	32	85	42	87	801
	72	77	58	18	72	891
	76	53	14	83	86	936

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(2)}$	291	237	237	114	114	
$A^{(2)}$	47	56	16	80	5	
$B^{(2)}$	3	7	31	21	93	465
	54	82	90	26	59	933
	87	93	61	3	64	924
	68	37	50	12	86	759

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(3)}$	237	237	194	194	211	
$A^{(3)}$	52	9	93	35	38	
$B^{(3)}$	20	44	85	93	52	882
	61	98	36	94	1	870
	93	37	17	41	54	726
	48	20	54	19	58	597

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(4)}$	211	231	231	211	211	
$A^{(4)}$	52	89	88	33	73	
$B^{(4)}$	73	72	35	90	28	894
	88	14	59	19	86	798
	39	92	3	79	18	693
	58	32	48	15	99	756

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(5)}$	97	97	168	168	174	
$A^{(5)}$	17	64	32	96	35	
$B^{(5)}$	62	11	71	37	93	822
	74	10	26	0	85	585
	12	20	20	55	24	393
	6	57	54	67	32	648

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(6)}$	174	134	134	308	308	
$A^{(6)}$	9	76	62	51	79	
$B^{(6)}$	99	8	73	79	91	1050
	59	24	15	56	76	690
	50	30	30	99	51	780
	21	32	36	71	41	603

Four-Subproblem Derivative

	Tightness Ratio			
	0.33	0.67	1.00	1.33
Linking Constraint Right-hand Side	1041	2082	3123	4164

Tightness Ratio = 0.67

	Number of Subproblems		
	2	4	6
Linking Constraint Right-hand Side	958	2082	3124

TABLE A-IV

Test Problem 4

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(1)}$	56	113	30	62	210	
$A^{(1)}$	40	91	10	30	160	
$B^{(1)}$	16	92	41	16	150	221
	38	39	32	71	80	182
	8	71	30	60	200	258
	38	52	30	42	170	232

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(2)}$	43	7	33	5	12	
$A^{(2)}$	20	3	12	3	18	
$B^{(2)}$	23	4	18	6	0	36
	26	5	40	8	12	64
	18	6	30	4	8	46
	9	7	20	0	3	27

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(3)}$	32	20	4	3	43	
$A^{(3)}$	9	25	1	1	10	
$B^{(3)}$	12	8	2	1	0	16
	30	15	0	1	23	48
	31	6	3	0	18	41
	21	4	1	2	14	29

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(4)}$	426	42	12	8	2	
$A^{(4)}$	280	10	8	1	1	
$B^{(4)}$	200	20	6	2	1	160
	100	0	20	3	0	86
	60	21	4	0	2	61
	310	8	4	6	1	230

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(5)}$	63	13	42	9	4	
$A^{(5)}$	49	8	21	6	1	
$B^{(5)}$	70	9	22	4	1	74
	40	6	8	0	6	42
	32	15	31	2	2	57
	18	15	38	10	4	60

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(6)}$	10	22	8	9	3	
$A^{(6)}$	5	10	8	2	1	
$B^{(6)}$	5	10	6	4	0	18
	4	22	4	6	1	26
	7	8	2	8	0	18
	8	6	0	0	3	12

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(7)}$	5	42	32	7	7	
$A^{(7)}$	0	10	42	6	4	
$B^{(7)}$	4	12	8	4	3	22
	5	14	8	2	8	26
	2	8	6	7	1	17
	0	10	6	1	3	14

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(8)}$	5	11	12	9	74	
$A^{(8)}$	8	0	10	1	40	
$B^{(8)}$	0	10	0	6	28	31
	0	20	0	0	6	18
	0	0	20	8	14	29
	0	3	5	4	0	8

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(9)}$	181	43	306	22	6	
$A^{(9)}$	86	11	120	8	3	
$B^{(9)}$	93	9	30	22	0	108
	12	6	80	13	6	82
	20	2	40	6	1	48
	30	12	16	18	3	55

	COLUMNS					
	1	2	3	4	5	RHS
$c^{(10)}$	30	62	42	5	8	
$A^{(10)}$	32	28	13	2	4	
$B^{(10)}$	36	45	13	2	2	69
	22	14	0	1	2	27
	14	20	12	0	1	33
	16	22	30	4	0	50

	Number of Subproblems				
	2	4	6	8	10
Linking Constraint Right-hand Side	155	293	338	386	509

REFERENCES

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

Balas, E., "Infeasibility Pricing Decomposition Method for Linear Programs," Operations Research, Vol. 14 (1966), pp. 847-873.

Balas, E., "Discrete Programming by the Filter Method," Operations Research, Vol. 15 (1967), pp. 915-957.

Balinski, M. L., "Integer Programming: Methods, Uses, Computation," Management Science, Vol. 12 (1965), pp. 253-313.

Beale, E. M. L., Mathematical Programming in Practice, Wiley and Sons, Inc., New York, 1968.

Beale, E. M. L., and R. E. Small, "Mixed Integer Programming by a Branch-and-Bound Technique," Proceedings of the Third IFIP Congress, Vol. 2 (1965), pp. 450-451.

Benders, J. F., "Partitioning Procedures for Solving Mixed Variables Programming Problems," Numerische Mathematik, Vol. 4 (1962), pp. 238-252.

Bennett, J. M., "An Approach to Some Structured Linear Programming Problems," Operations Research, Vol. 14 (1966), pp. 636-645.

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

Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

Dantzig, G. B., and P. Wolfe, "Decomposition Algorithm for Linear Programs," Operations Research, Vol. 8 (1960), pp. 101-111.

Duffin, R. J., "On Fourier's Analysis of Linear Inequality Systems," Mathematical Programming Study 1 (1974), pp. 71-95.

Everett, H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," Operations Research, Vol. 11 (1963), pp. 399-417.

Fleisher, J., Letter to the Editor in SIGMAP Newsletter, No. 20 (February 1976), pp. 6-8.

Gal, T., and J. Nedoma, "Multiparametric Linear Programming," Management Science, Vol. 18 (1972), pp. 406-422.

Garfinkel, R., and G. Nemhauser, Integer Programming, Wiley and Sons, Inc., New York, 1972.

Geoffrion, A. M., "An Improved Implicit Enumeration Approach for Integer Programming," Operations Research, Vol. 17 (1969), pp. 437-454.

Geoffrion, A. M., "Lagrangean Relaxation for Integer Programming," Mathematical Programming Study 2 (1974), pp. 82-114.

Geoffrion, A. M., and G. W. Graves, "Multicommodity Distribution System Design by Benders Decomposition," Working Paper No. 209, Western Management Science Institute, UCLA, August 1973.

Geoffrion, A. M., and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, Vol. 18 (1972), pp. 465-491.

Geoffrion, A. M., and R. M. Nauss, "Parametric and Postoptimality Analysis in Integer Linear Programming," Working Paper No. 246, Western Management Science Institute, UCLA, January 1976.

Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, Vol. 13 (1965), pp. 879-919.

Gomory, R. E., "An Algorithm for Integer Solutions to Linear Programs," in Recent Advances in Mathematical Programming, R. Graves and P. Wolfe, eds., McGraw-Hill, New York (1963), pp. 269-302.

Gomory, R. E., "On the Relation Between Integer and Noninteger Solutions to Linear Programs," Proceedings of the National Academy of Science, Vol. 53 (1965), pp. 260-265.

Gorry, G., and Shapiro, J., "An Adaptive Group Theoretic Algorithm for Integer Programming Problems," Management Science, Vol. 17 (1971), pp. 285-306.

Greenberg, H., Integer Programming, Academic Press, New York, 1971.

Hartman, J., and L. S. Lasdon, "A Generalized Upper Bounding Method for Doubly Coupled Linear Programs," Naval Research Logistics Quarterly, Vol. 17 (1970), pp. 411-429.

Healy, W. C., "Multiple Choice Programming," Operations Research, Vol. 12 (1964), pp. 122-138.

Hillier, F. S., "Efficient Heuristic Procedures for Integer Programming with an Interior," Operations Research, Vol. 17 (1969), pp. 600-637.

Hillier, F. S., and G. J. Lieberman, Introduction to Operations Research, second edition, Holden-Day, Inc., San Francisco, 1974.

Hu, T. C., Integer Programming and Network Flows, Addison Wesley, Reading Mass., 1969.

Kaneko, I., "On the Unboundedness of the Set of Integral Points in a Polyhedral Region," Technical Report SOL 74-12, Systems Optimization Laboratory, Department of Operations Research, Stanford University, September 1974.

Kaplan, S., "Solution of the Lorie-Savage and Similar Integer Programming Problems by the Generalized Lagrange Multiplier Method," Operations Research, Vol. 14 (1966), pp. 1130-1136.

Kochman, G. A., "On a Class of Concave-Separable Integer Programs," Technical Report, Department of Operations Research, Stanford University, September 1976.

Land, A. H., and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems," Econometrica, Vol. 28 (1960), pp. 497-520.

Lasdon, L. S., "Decomposition in Resource Allocation," in Decomposition of Large-Scale Problems, D. M. Himmelblau, ed., North Holland Publishing Company, Amsterdam, 1973.

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

Little, J. D. C., K. C. Murty, B. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11 (1963), pp. 972-989.

Martin, G. T., "An Accelerated Euclidean Algorithm for Integer Linear Programming," in Recent Advances in Mathematical Programming, R. Graves and P. Wolfe, eds., McGraw-Hill, New York, 1963.

Mitten, L. G., "Branch-and-Bound Methods: General Formulation and Properties," Operations Research, Vol. 18 (1970), pp. 24-34.

Nauss, R. M., "Parametric Integer Programming," Working Paper No. 226, Western Management Science Institute, UCLA, January 1975.

Nemhauser, G., and Z. Ullman, "A Note on the Generalized Lagrange Multipler Solution to an Integer Programming Problem," Operations Research, Vol. 16 (1968), pp. 450-453.

Petersen, C. C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R & D Projects," Management Science, Vol. 13 (1967), pp. 736-750.

Piper, C. J., and A. A. Zoltners, "Implicit Enumeration Based Algorithms for Postoptimizing Zero-One Programs," Naval Research Logistics Quarterly, Vol. 22 (1975), pp. 791-809.

Piper, C. J., and A. A. Zoltners, "Some Easy Postoptimality Analysis for Zero-One Programming," Management Science, Vol. 22 (1976), pp. 759-765.

Radke, M. A., "Sensitivity Analysis in Discrete Optimization," Working Paper No. 240, Western Management Science Institute, UCLA, September 1975.

Reardon, K. J., Decomposition of Dual Angular Integer Programs with Applications to Multi-Stage Capital Budgeting, Ph.D. Dissertation, Department of Operations Research, Stanford University, 1974.

Ritter, K., "A Decomposition Method for Linear Programming Problems with Coupling Constraints and Variables," MRC #739, Mathematics Research Center, University of Wisconsin, April 1967.

Roodman, G., "Postoptimality Analysis in Zero-One Programming by Implicit Enumeration," Naval Research Logistics Quarterly, Vol. 19 (1972), pp. 435-447.

Rosen, J. B., "Primal Partition Programming for Block Diagonal Matrices," Numerische Mathematik, Vol. 6 (1964), pp. 250-260.

Schrage, L., "Using Decomposition in Integer Programming," Naval Research Logistics Quarterly, Vol. 20 (1973), pp. 469-476.

Tomlin, J. A., "Branch-and-Bound Methods for Integer and Non-Convex Programming," in Integer and Nonlinear Programming, J. Abadie, ed., North Holland, Amsterdam, 1970.

Trauth, C. A., and R. E. Woolsey, "Integer Linear Programming: A Study in Computational Efficiency," Management Science, Vol. 15 (1969), pp. 481-493.

Van de Panne, C., "A Node Method for Multi-parametric Linear Programming," Management Science, Vol. 21 (1975), pp. 1014-1020.