Construction of Generalized Capital Budgeting
Test Problems with Known Optimal Solutions

by
Jay M. Fleisher

Computer Sciences Technical Report #260

August 1975

The University of Wisconsin
Computer Sciences Department
1210 West Dayton Street
Madison, Wisconsin 53706

Received: August 28, 1975

Construction of Generalized Capital Budgeting Test Problems with Known Optimal Solutions

by

Jay M. Fleisher†

Technical Report #260

†This research was supported by National Science Foundation Grant MCS74-20584.

Abstract

This report presents a procedure for the construction of "generalized" capital budgeting problems with known optimal solutions. Generalized capital budgeting problems are pure integer programming problems with upper bounded variables and nonnegative data and thus provide a source of problems for testing integer programming codes. The data comprising these problems will be selected in such a manner that sufficient optimality conditions will be satisfied.

1. INTRODUCTION

1.1 Statement of the Problem

Generalized capital budgeting problems, as considered in this report, have the following form†:

(GCB) Maximize $c\,x$

subject to $A\,x \leqq b$

$0 \leqq x \leqq d$, x integer

where A is an $m \times n$ matrix with all $A_{ij} \geqq 0$, c, d, and $x \in R_+^n$, and $b \in R_+^m$. In addition, it is assumed throughout this report that the data for (GCB) is integer valued.

Note that the problem data of (GCB) consists entirely of nonnegative values and (GCB) necessarily has an optimal solution because $x = 0$ is feasible and the objective value is bounded above by cd. It is usually the case that n is larger than m: i.e., typically $n \geqq 2m$.

The usual type of capital budgeting problem [2, 3, 7] has the form (GCB) with $d = 1$ (vector of 1's). It arises in the following context:

A firm has n projects which it may or may not undertake and the j^{th} project has a net profit of c_j. If $x_j = 1$, project j is undertaken and if $x_j = 0$, project j is not undertaken. Project j requires an expenditure of A_{ij} units of the i^{th} resource and the availability of the i^{th} resource is b_i units.

---

†Vectors may be row vectors or column vectors. If A is an $m \times n$ matrix, x and $y \in R^n$, and $u \in R^m$, then xy will denote $\sum_{j=1}^{n} x_j y_j$ and uAx will denote $\sum_{i=1}^{m} \sum_{j=1}^{n} u_i\, A_{ij}\, x_j$.

Problem (GCB) arises in a more general context where a firm may undertake up to d_j projects of type j, each with the same cost structure.

Problem (GCB) is equivalent to the ordinary capital budgeting problem since integer programming problems with bounded variables can be reformulated so that all integer variables are 0-1 variables, as is shown in references [2] and [6]. However, this report considers the general case where variables may be allowed to assume values other than 0 or 1. As is discussed in reference [6], it is usually more efficient to solve problems with $d \neq 1$ directly, without conversion to 0-1 problems.

1.2 Basic Idea

The sufficient optimality conditions derived in [1] form the basis for construction of test problems of the form (GCB) with known optimal solutions. The data comprising (GCB) will be generated in such a manner that the optimality conditions will be satisfied at a specified solution x^*. Alternate optima are possible, however, the optimal value of (GCB) is $c\,x^*$, so it is possible to determine how close to optimality a feasible solution obtained from a code actually is.

The sufficient optimality criterion, which is derived in reference [1], is stated below for the case where problems are of the form (GCB). The quantities A, b, c, d will refer to data for (GCB) and m and n will denote respectively the number of constraints and the number of variables for (GCB).

(SOC) Sufficient Optimality Criteria:

Let x^* be an integer vector such that $0 \leqq x^* \leqq d$, $s^* \in R_+^m$, $c^{(k)}$ be an integer valued vector, $k = 1, 2, \ldots, p$,

$u^{(k)} \in R_+^m$, $k = 1, 2, \ldots, p$,
$v^{(k)}$, $w^{(k)} \in R_+^n$, $k = 1, 2, \ldots, p$,
$\lambda_k \geqq 0$, $k = 1, 2, \ldots, p$.

If

(1) $c^{(k)} = A^T u^{(k)} - v^{(k)} + w^{(k)}$, $k = 1, 2, \ldots, p$,
(Dual Feasibility)

(2) $c = \sum_{k=1}^{p} \lambda_k c^{(k)}$, $k = 1, 2, \ldots, p$,
(Composition)

(3) $b = Ax^* + s^*$,
(Primal Feasibility)

(4) $\delta_k = s^* u^{(k)} + x^* v^{(k)} + (d-x^*) w^{(k)} < \gamma_k$
where $\gamma_k =$ †gcd $(c_1^{(k)}, c_2^{(k)}, \ldots, c_n^{(k)})$, $k = 1, 2, \ldots, p$
(Quasicomplementarity)

then x^* solves (GCB).

The quantities x^* and s^* will be referred to respectively as the solution vector and the slack vector, $u^{(k)}$, $v^{(k)}$, and $w^{(k)}$ will be referred to as u-, v-, and w-multipliers, the $c^{(k)}$ vectors will be referred to as component cost vectors, the λ_k scalars will be referred

---

†Greatest common divisor. A generalized greatest common divisor, applicable when the arguments are rational numbers, is described in reference [1] which also lists properties of the generalized greatest common divisor.

to as component weights, δ_k will be referred to as the index of quasicomplementarity for the k^{th} component, and γ_k will be referred to as the critical index for the k^{th} component. In addition, the quantities $x^*v^{(k)} + (d-x^*)w^{(k)}$ and $s^*u^{(k)}$ will be referred to respectively as the solution quasicomplementarity index and the slack quasicomplementarity index for the k^{th} component.

The following theorem guarantees the existence of test problems of the form (GCB) with integer data such that the (SOC) conditions are satisfied at a solution x^* and such that x^* does not solve the continuous relaxation of (GCB).

Theorem 1:

Let m, n $\geqq$ 1, $x^*, d \in R^n$ be integer vectors such that $0 \leqq x^* \leqq d$ and $x^* \neq d$. Then there exists a nonnegative m x n integer matrix A and integer vectors $b \in R_+^m$ and $c \in R_+^n$ such that (GCB) with the data A, b, c, and d has the following properties:

(i) The conditions in (SOC) hold at x^* and hence, x^* solves (GCB).

(ii) The solution x^* is not optimal if the integrality requirements of (GCB) are removed.

Proof:

Since $x^* \neq d$, let r be an index such that $x_r^* \leq d_r - 1$.

$$\text{Set } A_{ij} = \begin{cases} 2 \text{ if } j = r,\ i = 1,2,\ldots,m, \\ 2k_{ij} \text{ if } j \neq r \text{ where } k_{ij} \geqq 0, \text{ integer, } i = 1, 2,\ldots,m, \end{cases}$$

(A contains all even entries and the r^{th} column contains all 2's)

$$b_i = \sum_{j=1}^{n} A_{ij}\, x_j^{*} + 1, \quad i = 1, 2, \ldots, m,$$

$$c_j = \sum_{i=1}^{m} A_{ij}, \ j = 1, 2, \ldots, n,$$

$$u_i^{(k)} = \begin{cases} 1 \text{ if } i = k \\ 0 \text{ if } i \neq k, \ k = 1, 2, \ldots, m, \end{cases}$$

$$v^{(k)} = w^{(k)} = 0, \ k = 1, 2, \ldots, m,$$

$$c_j^{(k)} = a_{kj}, \ j = 1, 2, \ldots, n, \quad k = 1, 2, \ldots m,$$

$$\lambda_k = 1, \ k = 1, 2, \ldots m, \text{ and}$$

$$s_i^{*} = 1, \quad i = 1, 2, \ldots m.$$

Then $\gamma_k = \gcd(\{c_j^{(k)}\}) = 2$ since $c_j^{(k)} = a_{kj}$ is even and $a_{kr} = 2$ and $\delta_k = \sum_{i=1}^{m} u_i^{(k)} s_i^{*} + \sum_{j=1}^{n} v_j^{(k)} x_j^{*} + \sum_{j=1}^{m} w_j^{(k)} (d_j - x_j^{*}) = 1 + 0 + 0 = 1$, $k = 1, 2, \ldots, m$.

Hence, $\delta_k < \gamma_k$, $k = 1, 2, \ldots, m$, so (4) of (SOC) holds. Verification that (1), (2) and (3) of (SOC) hold is straightforward and follows from the choice of $c^{(k)}$, c, and b.

To show x^{*} is not optimal if the integrality requirements are dropped, consider the vector x^{o} where

$$x_j^{o} = \begin{cases} x_j^{*} + \frac{1}{2} & \text{if } j = r \\ x_j^{*} & \text{if } j \neq r. \end{cases}$$

Then $b_i = \sum_{j=1}^{n} A_{ij}\, x_j^{*} + 1 = \sum_{j=1}^{n} A_{ij}\, x_j^{o}$ since $a_{ir} = 2$ and $cx^{o} - cx^{*} = \frac{1}{2} c_r = m > 0$ since $c_r = 2m$. Thus x^{o} is feasible for (GCB) with the integrality requirements removed and yields a higher objective value than x^{*}. ∎

Theorem 1 settles the question of existence of problems (GCB) with a solution x^* such that (SOC) holds at x^* and x^* does not solve the continuous relaxation of (GCB). However, the problems exhibited in Theorem 1 have the undesirable property that the rows can be simplified: i.e., since $A_{ij} \equiv 0 \pmod 2$, b_i could be replaced by $\sum_{j=1}^{n} A_{ij} x_j^*$ without removing any integer lattice points from the feasible region of (GCB). For this reason, Theorem 1 does not provide a source of good test problems, and, in fact, has not been used for problem generation.

The test problem generator to be described in Section 2 is a procedure for generating data and other quantities appearing in (SOC) for problems of the form (GCB) such that (SOC) holds at a solution vector x^*. The quantities x^* and d are specified at the beginning. For each component k, $k = 1, 2, \ldots, p$, a subset of the rows of A and s^* as well as the vectors $u^{(k)}$, $v^{(k)}$, $w^{(k)}$, and $c^{(k)}$ are generated in such a manner that (1) and (4) will hold for the k^{th} component. Finally, the component weights λ_k are generated and c and b are set according to (2) and (3). With (SOC) satisfied, it is guaranteed that x^* is optimal for (GCB).

A number of heuristic rules have been incorporated into the construction procedure in an attempt to prevent the generation of "trivial" problems. Judging from the difficulty of the problems obtained, these heuristic rules have been successful.

2. GENERATION OF TEST PROBLEMS

2.1 Properties of Test Problem Generators

The generator to be described guarantees that the conditions in (SOC) are satisfied. In addition, the following considerations have also been taken into account:

(5) The data for (GCB) are integer valued. This will eliminate rounding errors in the data which could possibly result in the generation of a test problem with the conditions in (SOC) not satisfied and x^* being non-optimal. Furthermore, source data for test problems generally requires less space when the data values are all integer valued.

(6) The test problem generator should be able to produce problems which are fairly difficult to solve. In particular, the solution to the continuous relaxation of (GCB) should not solve (GCB).

(7) The test problems should not contain rows such that (GCB) can be readily simplified by inspection. For example, a row $\sum_{j=1}^{n} \alpha_j x_j \leqq \beta$ should not have any $\alpha_j > \beta$, since this would imply x_j can be eliminated from (GCB) because $x_j \geqq 1$ is infeasible, and β should be an integral multiple of $\gamma = \gcd(\alpha_1, \alpha_2, \ldots, \alpha_n)$ since otherwise, the row could be replaced by the tighter row (in the continuous sense) $\sum_{j=1}^{n} \frac{\alpha j}{\gamma} x_j \leqq \lfloor \frac{\beta}{\gamma} \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer which does not exceed x.

(8) There should be at least two components in the generation of the test problems. If there were only one component, then, as shown in reference [1], the gap between the objective values of (GCB) and its continuous relaxation would have to be less than $\gamma = \gcd(c_1, c_2, \ldots c_n)$. Since the

objective value of (GCB) must be an integral multiple of γ, an optimal solution to (GCB) could be recognized by a code without continuing the search if an optimality test based on the gcd were incorporated in the code. Such a test appears in the ENUMER8 code [5, 6] of Trotter and Shetty.

2.2 Procedure for Generating Test Problems

The procedure to be described below is a systematic way of generating data and the other quantities appearing in (SOC) for problems of the form (GCB) such that (SOC) holds at a solution vector x^*. The procedure generates at least 5 rows ($m \geqq 5$), but there is theoretically no upper limit on the number of rows or columns. (At least two components are needed for generating good test problems and the procedure to be described requires at least 3 rows for the first component plus 2 rows for each subsequent component.) All data generated for (GCB) will be integer valued and the A matrix coefficients will be roughly uniformly distributed on $\{0, 1, 2, \ldots, a_0 - 1\}$ where a_0 is a specified parameter.

In what follows, Latin letters will refer to quantities which assume integer values and Greek letters refer to quantities which may assume any real values. The expression $e(\alpha)$ will denote $\lfloor \alpha + 0.5 \rfloor$: i.e., the value of α "rounded" to the nearest integer.

Input Parameters

m = number of constraints; $m \geqq 5$.

n = number of variables; $n \geqq 5$ is recommended.

a_o = parameter determining range of A matrix coefficients; $0 \leqq A_{ij} \leqq a_o-1$; $a_o \geqq 10$ is recommended. The critical index for every component will be an integral multiple of a_o and usually be equal to a_o.

c_o = parameter determining relation of a_o to the approximate average value for c_j; $\bar{c}_j \stackrel{\sim}{\sim} c_o \cdot \bar{A}_{ij} \stackrel{\sim}{\sim} c_o \cdot a_o/2$; $2 \leqq c_o \leqq 10$ seems reasonable.

†d = vector of upper bounds; $d > 0$.

†x^* = vector which will be a solution to the problem generated. $0 \leqq x^* \leqq d$.

α = ratio of the index of quasicomplementarity of each component to its maximum allowed value of a_o-1. The index of quasicomplementarity for <u>every</u> component will be $e(\alpha(a_o-1))$; $0 \leqq \alpha \leqq 1$ (Larger values of α tend to generate more difficult test problems.)

β = maximum ratio of the solution quasicomplementarity index to the total index; $0 \leqq \beta \leqq 1$ (For a fixed α, smaller values of β tend to result in more slack in the rows of A.) If $\beta = 0$, the algorithm sets the v and w multipliers to 0.

p = number of components; $2 \leqq p \leqq \lfloor\frac{m-1}{2}\rfloor$.

u_o = maximum allowed u-multiplier value; $3 \leqq u_o \leqq 7$ is recommended.

<u>Procedure P1:</u> (see Figure 1)

1. [Initializations]

   Set K_i, $i = 1, 2, \ldots m$ to 0 (K_i will indicate the component corresponding to row i). Set R_t, $t = 1, 2, \ldots m$ to a random permutation of

---

†d and x^* may be input explicitly or generated according to prespecified parameters. See the appendix.

$\{1, 2,\ldots,m\}$ (used to indicate the order in which the rows will be generated). For the first component, r_1 rows of A are generated and for the other components, r_2 rows of A are generated where $r_2 = \lfloor (m-1)/p \rfloor$ and $r_1 = m - r_2(p-1)$. Note that $(r_1, r_2) \geqq (3, 2)$.

2. [Generation of A, s^*, multipliers, and component cost vectors].

   Set $t = 1$.

   For $k = 1, 2,\ldots,p$ do steps 2.1 through 2.8.

2.1 [Initializations]

If $k = 1$, set $r = r_1$; otherwise, set $r = r_2$ (r is the number of rows generated). Set $K_i = k$, $i = R_t, R_{t+1},\ldots, R_{t+r-1}$ and $Q = e(\alpha\cdot(a_0-1))$.

2.2 [Generate v and w multipliers for component k] If $\beta = 0$, set $v^{(k)}$ and $w^{(k)}$ to 0 and go to step 2.3; otherwise generate $v^{(k)}$ and $w^{(k)}$ such that $Q_1 \equiv x^* v^{(k)} + (d-x^*)w^{(k)} \leqq \lfloor \beta\cdot Q \rfloor$ and $0 \leqq v_j^{(k)}, w_j^{(k)} \leqq a_0-1$ (see the Appendix). Set $Q = Q-Q_1$.

2.3 [Generate u multipliers for rows not corresponding to component k].

Set $u_i^{(k)} = 0$ for $K_i = 0$ (rows corresponding to subsequent components). If $k = 1$ go to step 2.4; otherwise generate $u_i^{(k)}$ previously generated $(1 \leq K_i < k)$ such that $Q_2 \equiv \sum\limits_{1 \leqq K_i < k} u_i^{(k)} s_i^* \leqq \lfloor \frac{1}{3} Q \rfloor$ and $0 \leqq u_i^{(k)} \leqq u_0$ (See the Appendix).

Set $Q = Q-Q_2$.

2.4 [Generate u multipliers and s^* components for rows corresponding to component k].

Generate $u_i^{(k)}$ and s_i^* for $i = R_t, R_{t+1},\ldots,R_{t+r-1}$ such that $\sum\limits_{K_i=k} u_i^{(k)} s_i^* = Q$, $1 \leqq u_i^{(k)} \leqq u_0$, and $u_{t+r-1}^{(k)} = 1$ (See the Appendix). This completes generation of $u^{(k)}$, $v^{(k)}$, and $w^{(k)}$ such that the index of quasicomplementarity for the k^{th} component will be $e(\alpha\cdot(a_o-1))$.

2.5 [Generate r-1 free rows of A corresponding to component k]

Generate rows $i = R_t, R_{t+1},\ldots,R_{t+r-2}$ such that A_{ij} is a random number uniformly distributed on $\{0, 1, 2,\ldots,a_o-1\}$ for $j = 1, 2,\ldots,n$. These rows will be referred to as free rows since they may be generated freely.

2.6 [Determine component cost vector $c^{(k)}$]

Set $\hat{c}_j^{(k)} = \sum_{i\neq R_{t+r-1}} A_{ij}\, u_i^{(k)} - v_j^{(k)} + w_j^{(k)}$; $j = 1, 2,\ldots,n$.
and $c_j^{(k)} = a_o\cdot[(\hat{c}_j^{(k)} + a_o - 1)/a_o]$, $j = 1, 2,\ldots,n$. Then $c_j^{(k)}$ is an integral multiple of a_o, and thus $\gcd(c_1^{(k)}, c_2^{(k)},\ldots,c_n^{(k)}) \geqq a_o$. Since $A_{ij} \geqq 0$ and $0 \leqq v_j^{(k)}, w_j^{(k)} \leqq a_o-1$, $c_j^{(k)} \geqq 0$.

2.7 [Generate fixed row of A corresponding to component k] Generate row $i = R_{t+r-1}$ such that $A_{ij} = c_j^{(k)} - \hat{c}_j^{(k)}$, $j = 1, 2,\ldots,n$. This row is referred to as a fixed row because the constraint $A_{ij} = c_j^{(k)} - \hat{c}_j^{(k)}$ determines A_{ij}. It may be verified that $0 \leqq A_{ij} \leqq a_o-1$ and $c_j^{(k)} = \sum_{i=1}^{m} A_{ij}\, v_i^{(k)} - v_j^{(k)} + w_j^{(k)} \equiv 0 \pmod{a_o}$.

2.8 [End of loop in step 2]

Set $t = t+r$.

3. [Generate b and c]

Set $b = Ax^* + s^*$.

For $k = 1, 2,\ldots,p$, generate λ_k's which are nonnegative integral multiples of $\frac{1}{a_o}$ and set $c = \sum_{k=1}^{p} \lambda_k c^{(k)}$. The λ_k's should be generated so that $\bar{c} = \sum_{j=1}^{n} c_j/n \approx \frac{1}{2}\, a_o\, c_o$ (see the Appendix). Set $z^* = cx^*$, the optimal value of the problem. This completes the generation process.

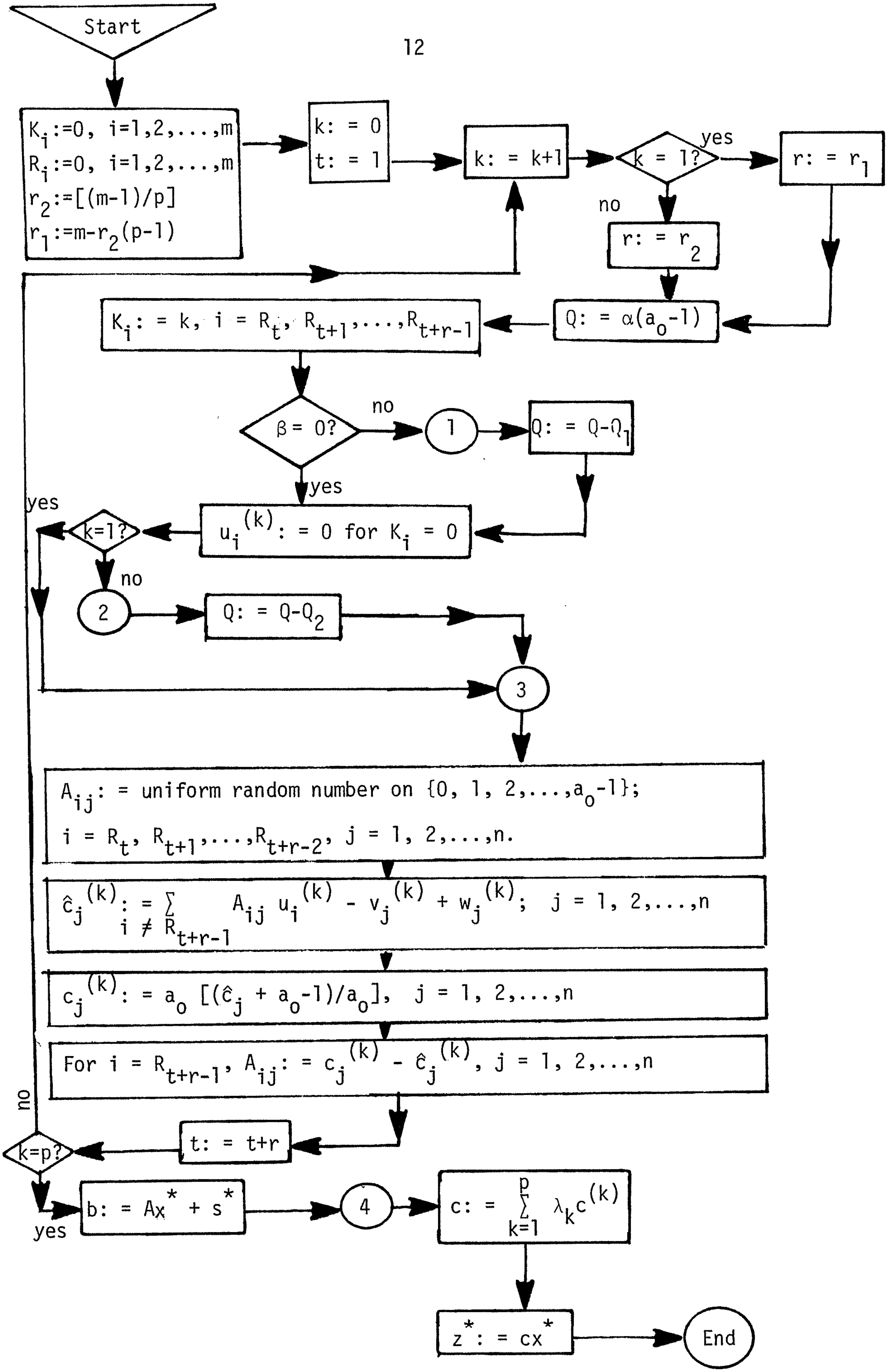

Figure 1: Flow Chart for the Test Problem Generator

Key to Procedures in Flowchart:

1. Generate $v^{(k)}$ and $w^{(k)}$ such that

   $Q_1 \equiv x^* \ v^{(k)} + (d-x^*)\ w^{(k)} \leqq \lfloor \beta \cdot Q \rfloor$ and $0 \leqq v_j^{(k)},\ w_j^{(k)} \leqq a_o - 1$

2. Generate $u_i^{(k)}$ for $1 \leqq K_i < k$ such that

   $Q_2 \equiv \sum_{1 \leqq K_i < k} u_i^{(k)}\ s_i^* \leqq \lfloor \frac{1}{3} Q \rfloor$ and $0 \leqq u_i^{(k)} \leqq u_o$.

3. Generate $u_i^{(k)}$ and s_i^* for $i = R_t, R_{t+1}, \ldots, R_{t+r-1}$ such that

   $\sum_{K_i = k} u_i^{(k)} s_i^* = Q$, $1 \leqq u_i^{(k)} \leqq u_o$, and $u_{t+r-1} = 1$.

4. Generate λ_k's which are nonnegative integral multiples of $\frac{1}{a_o}$ such that $\sum_{j=1}^{n} \sum_{k=1}^{p} \lambda_k \ c_j^{(k)} \approx \frac{n}{2} a_o c_o$.

2.3 Validity of Procedure:

In steps 2.2-2.4, we generate $u^{(k)}$, $v^{(k)}$, and $w^{(k)}$ multipliers, as well as s_i^* for rows i corresponding to the k^{th} component, such that

$$(9) \quad \sum_{i=1}^{m} s_i^* \ u_i^{(k)} + \sum_{j=1}^{n} x_j^* \ v_j^{(k)} + \sum_{j=1}^{n} (d_j - x_j^*)\ w_j^{(k)} = G,$$

where $G \leq a_o - 1$. (This may be verified by summing the quasicomplementarity relations in steps 2.2-2.4.) In steps 2.5-2.7, we generate rows of A as well as $c^{(k)}$ such that

$$(10) \quad c_j^{(k)} = \sum_{i=1}^{m} A_{ij} \ u_i^{(k)} - v_j^{(k)} + w_j^{(k)} = t_j \ a_o,$$

where t_j is a nonnegative integer.

(Note that if row i corresponds to component k', where k'>k, corresponding terms in (9) and (10) are 0 because $u_i^{(k)} = 0$.) From (10) we have $\gcd(c_1^{(k)}, c_2^{(k)}, \ldots, c_n^{(k)}) \geqq a_o > G$ where the first inequality holds as equality whenever the t_j's are relatively prime. Thus, from (9) and (10) it follows that each time steps 2.2-2.7 are executed conditions (1) and (4) of (SOC) are satisfied for a fixed component k so that upon completion of step 2, (1) and (4) are satisfied.

In step 3, we generate nonnegative component weights λ_k and set $b = Ax^* + s^*$ and $c = \sum_{k=1}^{p} \lambda_k c^{(k)}$ which are, respectively, conditions (3) and (2) of (SOC). Thus, at the conclusion of the procedure, (SOC) is satisfied. ∎

From (SOC), an upper bound on the differential between the optimal objective values of (GCB) and its continuous relaxation is given by $\sum_{k=1}^{p} \lambda_k \delta_k$ since the gap between the optimal objective values of (GCB_k) (problem (GCB) with $c = c^{(k)}$) and its continuous relaxation is bounded above by δ_k [1]. Since $\delta_k = e(\alpha\cdot(a_0-1))$ in Procedure P1, it is suggested that the parameter α should be set at or near 1 for constructing difficult test problems, although a large gap needn't imply a difficult problem or vice-versa.

For a variety of test problems constructed using Procedure P1, the actual gap was usually greater than 70% of this upper bound (see Section 3). A sufficient condition for the gap to be <u>positive</u> and hence, for the solution to the continuous relaxation of (GCB) to <u>not</u> solve (GCB), is $s^* > 0$ and for some r, $c_r > 0$ and $x_r^* < d_r$. If these conditions hold, $\theta = \min \{s_i^*/a_{ir} | a_{ir} > 0, d_r - x_r^*\} > 0$. Then x^0 is feasible for the continuous relaxation of (GCB) where

$$x_j^0 = \begin{cases} x_j^* + \theta & \text{if } j = r \\ x_j^* & \text{if } j \neq r \end{cases}$$

and $cx^o - cx^* = \theta c_r > 0$. Thus, x^o is feasible for the continuous relaxation of (GCB) and yields a higher objective value than x^*.

3. COMPUTATIONAL EXPERIENCE

3.1 Generation of Test Problems

A variety of integer programming problems of the form (GCB) were generated using Procedure P1 described in Section 2.2 and the Appendix. The parameters used in generating the problems are given in Table I and are as described in Section 2.2 except that d_0 and ξ_0, used to generate d and x^*, are described in the Appendix.

Table I

Parameters For Test Problems Generated

Problem	m	n	a_0	c_0	d_0	ξ_0	α	β	p	u_0
1	7	12	50	3	-3	.500	1.000	.100	3	7
2	7	15	100	6	3	1.000	1.000	.500	3	3
3	7	10	60	3	10	.500	1.000	.250	3	6
4	11	15	120	5	2	.600	1.000	.200	3	6
5	11	15	120	5	2	.600	1.000	.200	5	6
6	10	20	100	4	1	.500	1.000	.150	4	5
7	7	30	100	5	1	.333	1.000	.400	3	5
8	9	30	100	5	-3	.333	1.000	.400	4	5
9	15	50	100	5	1	.250	1.000	.250	4	5
10	15	50	100	5	1	.250	1.000	.250	7	5

The test problems generated, along with the solution and slack vectors, are listed in the Appendix. Brief characteristics of the test problems generated are listed in Table II where

Nodes = an upper bound on the number of integer solutions

$$= \prod_{j=1}^{n} (d_j+1),$$

z^* = the optimal objective value of the GCB problem,

z^o = the optimal objective value of the corresponding continuous relaxation,

Gap = $z^o - z^*$,

Bound = a priori upper bound on $z^o - z^*$ based on results from the generator,

% Gap = 100 times $(z^o - z^*)/z^*$,

% Bound = 100 times $(z^o - z^*)$/Bound.

Table II

Characteristics of Test Problems Generated

Problem	m	n	Nodes	z^*	z^o	Gap	Bound	% Gap	% Bound
1	7	12	3.32×10^{5}	722	752.50	30.50	31.36	4.22	97.3
2	7	15	1.07×10^{9}	5305	5438.74	133.74	199.98	2.52	66.9
3	7	10	1.35×10^{10}	2375	2414.44	39.44	51.13	1.66	77.1
4	11	15	1.43×10^{7}	3990	4057.01	67.01	82.31	1.68	81.4
5	11	15	1.43×10^{7}	4222	4275.52	53.52	80.33	1.27	66.6
6	10	20	1.05×10^{6}	2139	2221.82	82.82	96.03	3.87	86.2
7	7	30	1.07×10^{9}	2460	2539.99	79.99	120.78	3.25	66.2
8	9	30	6.34×10^{13}	3615	3681.62	66.62	95.04	1.84	70.1
9	15	50	1.13×10^{15}	3046	3087.06	41.06	49.55	1.35	82.9
10	15	50	1.13×10^{15}	3082	3177.65	95.65	112.86	3.10	84.8

3.2 Testing the Problems

The test problems generated were run on the integer programming codes IPMIXD and IPDNUM available at the Madison Academic Computing Center using a Univac 1110 computer. IPMIXD is a linear programming based branch and bound algorithm based on the method of Land and Doig [4] for solving pure and mixed integer programming problems. IPDNUM is a pure integer programming code developed from ENUMER8 [5,6] which uses an implicit enumeration algorithm with an assortment of fathoming tests. Only Problem 6 was tested with Univac's FMPS level 6.0 branch and bound 0-1 code, since the code is unsatisfactory in its current state for these test problems due to excessive output generated.

Table III lists computational results where

Code = 1 for IPMIXD, 2 for IPDNUM, 3 for FMPS,

Terminated = explanation of how run terminated,

Best Objective = objective value of best feasible solution obtained,

% Error = 100 x (z^* - Best Objective)/z^*, where z^* is the optimal objective value (see Table II),

Iterations = number of nodes explicitly analyzed,

Time = solution time in seconds,

Best Node # = node number at which best solution was found,

Incumbents = number of successive <u>feasible</u> solutions found with strictly increasing objective value.

(For many of the problems, optimal solutions other than x^* were found, as alternate optima may exist for problems of the form (GCB). No attempt has been made to collect data on these alternate optima.)

Table III

Computational Results

Problem	Code	Terminated	Best Objective	% Error	Iterations	Time	Best Node #	Incumbents
1	1	Optimal	722	.00	1333	8.2	13	2
2	1	Time Limit*	5305	.00	30315	180.0+	346	3
2	2	Time Limit*	5305	.00	33576	180.0+	14803	13
3	1	Optimal	2375	.00	4422	25.2	179	4
3	2	Optimal	2375	.00	20544	51.4	12904	36
4	1	Optimal	3990	.00	1052	10.9	988	10
5	1	Optimal	4222	.00	107	2.0	16	1
6	1	Optimal	2139	.00	1675	21.8	20	1
6	2	Time Limit	2096	2.01	4226	60.0+	2690	8
6	3	Page Limit**	—	—	104	61.6	—	0
7	1	Time Limit*	2460	.00	19633	180.0+	166	3
8	1	Time Limit	3598	.47	18383	180.0+	11152	9
9	1	Time Limit	2862	6.04	8288	180.0+	3282	7
10	1	Time Limit	3031	1.65	9002	180.0+	7717	12

These results indicate that Procedure P1 constructs difficult test problems of the form (GCB). Even moderate sized problems have required examination of several thousand nodes to verify optimality, although the IPMIXD code usually arrives at a good solution early in the search for this problem class.

---

*Best feasible solution found is optimal, but optimality not verified within time limit.

**The FMPS code did not find a feasible solution before generating 100 pages of unsuppressable output.

Appendix

Procedures For Generating u-, v-, and w-Multipliers, and Component Weights

Listed here are some procedures required by Procedure P1 of Section 2.2 for generating u-, v-, and w-multipliers, and component weights. These procedures were used in conjunction with Procedure P1 for generating the problems discussed in Section 3. The same conventions used in Section 2.2 will apply here, and urn(p, q) will denote a random integer uniformly distributed on {p, p+1, p+2,...,q}.

A1. Generation of $v^{(k)}$ and $w^{(k)}$ such that $x^* v^{(k)} + (d-x^*)w^{(k)} \leqq g$ and $0 \leqq v_j^{(k)}, w_j^{(k)} \leqq a_0-1$.

This procedure is used in step 2.2 of Procedure P1, where $g = [\beta Q]$.

1. [Initialization]

   Set $M_j = \text{urn}(0, 1)$, $j = 1, 2,\ldots,n$

   (If $M_j = 0$, $v_j^{(k)} \geqq 0$ and $w_j^{(k)} = 0$;
   if $M_j = 1$, $v_j^{(k)} = 0$ and $w_j^{(k)} \geqq 0$. If conditions (1) and (4) of (SOC) can be satisfied, they can be satisfied with $v^{(k)} w^{(k)} = 0$).

2. [Compute trial multipliers]

   For $j = 1, 2,\ldots,n$,

   if $M_j = 0$, set $\hat{w}_j^{(k)} = 0$ and $\hat{v}_j^{(k)} = \text{urn}(0, g)$,

   if $M_j = 1$, set $\hat{v}_j^{(k)} = 0$ and $\hat{w}_j^{(k)} = \text{urn}(0, g)$.

3. [Scale the trial multipliers]

   Compute $\hat{g} = x^* \hat{v}^{(k)} + (d-x^*)\hat{w}^{(k)}$. If $\hat{g} = 0$ go to step 4, otherwise set

   $v_j^{(k)} = \max(\lfloor \frac{g}{\hat{g}} \hat{v}_j^{(k)} \rfloor, a_0-1)$, $j = 1, 2,\ldots,n$,

   $w_j^{(k)} = \max(\lfloor \frac{g}{\hat{g}} \hat{w}_j^{(k)} \rfloor, a_0-1)$, $j = 1, 2,\ldots,n$.

   (This guarantees $x^* v^{(k)} + (d-x^*)w^{(k)} \leqq g$.)

4. [Increase quasicomplementarity where possible]

Set $h = g - x^* v^{(k)} + (d-x^*)w^{(k)}$.

For $j = 1, 2, \ldots, n$, do steps 4.1 through 4.4.

4.1 [v or w multiplier?]

If $M_j = 0$ go to step 4.2, otherwise go to step 4.3.

4.2 [Check v-multiplier]

†If $1 \leqq x_j^* \leqq h$, set $v_j^{(k)} = v_j^{(k)} + 1$ and $h = h - x_j^*$ (v-multiplier increased by 1 only if it increases quasicomplementarity, but not above g). If $h = 0$ go to step 5, otherwise go to step 4.4.

4.3 [Check w-multiplier]

†If $1 \leqq x_j^* - d_j \leqq h$, set $w_j^{(k)} = w_j^{(k)} + 1$ and $h = h + x_j^* - d_j$ (w-multiplier increased by 1 only if it increases quasicomplementarity, but not above g). If $h = 0$ go to step 5, otherwise go to step 4.4.

4.4 [End of loop]

(This completes generation of $v_j^{(k)}$ and $w_j^{(k)}$).

5. [Compute actual solution quasicomplementarity]

Set $Q_1 = x^* v^{(k)} + (d-x^*)w^{(k)}$ ($Q_1 \leqq g$ and is usually close to g). ∎

A2. Generation of $u_i^{(k)}$ for rows already generated ($1 \leqq K_i < k$) such that $\sum_{1 \leqq K_i < k} u_i^{(k)} s_i^* \leqq g$ and $0 \leqq u_i^{(k)} \leqq u_0$.

---

†The contribution to the index of quasicomplementarity for this column alone is a positive integral multiple of $v_j^{(k)}$ (or $w_j^{(k)}$) so the multiplier value can never exceed $a_0 - 1$.

This procedure is used in step 2.3 of Procedure P1.

1. [Compute trial multipliers]

   For i = 1, 2,...,m, if $1 \leqq K_i < k$, set $u_i^{(k)} = \text{urn}(0, u_0)$.

2. [Scale trial multipliers]

   Compute $\hat{g} = \sum_{1 \leqq K_i < k} u_i^{(k)} s_i^*$. If $\hat{g} = 0$ go to step 3, otherwise compute $u_i^{(k)} = \min.\ (v_0, \lfloor \frac{g}{\hat{g}}\, \hat{u}_i^{(k)} \rfloor)$, $1 \leqq K_i < k$ (This guarantees $\sum_{1 \leqq K_i < k} u_i^{(k)} s_i^* \leqq g$).

3. [Increase quasicomplementarity where possible].

   Set $h = g - \sum_{1 \leqq K_i < k} u_i^{(k)} s_i^*$.

   For $1 \leqq K_i < k$, if $s_i^* \leqq h$ and $u_i^{(k)} \leqq u_0$ set $u_i^{(k)} = u_i^{(k)} + 1$ and $h = h - s_i^*$ (u-multiplier increased by 1 if possible)

4. [Compute actual quasicomplementarity]

   Set $Q_2 = \sum_{1 \leqq K_i < k} u_i^{(k)} s_i^*$. ∎

The u-multipliers for rows already generated will generally be smaller than the u-multipliers for rows currently being generated (see Procedure A3) because of the allowable quasicomplementarity factor of $\frac{1}{3}Q$ in step 2.3 of Procedure P1.

A3. Generation of $u_i^{(k)}$ and s_i^* for the r rows currently being generated such that $\sum_{K_i = k} u_i^{(k)} s_i^* = g$ and $1 \leqq u_i^{(k)} \leqq u_0$.

This procedure is used in step 2.4 of Procedure P1.

1. [Determine products of u-multipliers and slacks so the q^{th} product is roughly proportional to $r - q + 1$].

   Set $y_q = \lfloor \frac{2(r-q+1)g}{r(r+1)} \rfloor$, $q = 1, 2, \ldots, r-1$

   and $y_r = g - \sum_{q=1}^{r-1} y_q$.

2. [Split products for the r-1 free rows.] For $q = 1, 2, \ldots, r-1$, do steps 2.1 through 2.6.

2.1 [Get row index]

   Set $i = R_{t+q-1}$

2.2 [Test for zero product]

   If $y_q = 0$, set $u_i^{(k)} = 1$ and $s_i^* = 0$ and go to step 2.6, otherwise go to step 2.3.

2.3 [Generate trial multiplier which won't exceed slack or u_0]

   Set $\hat{u}_i^{(k)} = \min(\lfloor\sqrt{y_q}\rfloor, \mathrm{urn}(1, u_0))$.

2.4 [Find multiplier which will make slack integer valued]

   Set $u_i^{(k)} = \max.(h\colon 1 \leqq h \leqq u_i^{(k)}$ and $y_q \equiv 0 \pmod h)$)

2.5 [Determine slack for row]

   Set $s_i^* = y_q/u_i^{(k)}$.

2.6 [End of step 2]

   (This completes generation of $u_i^{(k)}$ and s_i^*.)

3. [Make sure all of the $u_i^{(k)}$s for every row but the fixed row are relatively prime]. If $\gcd(\{u_i^{(k)} \mid i \neq R_{t+r-1}\}) \neq 1$, set $u_i^{(k)} = 1$ and $s_i^* = y_{r-1}$ for $i = R_{t+r-2}$.

4. [Fixed row has u-multiplier of 1]

For $i = R_{t+r-1}$ set $u_i^{(k)} = 1$ and $s_i^* = y_r$. ∎

Procedures A2 and A3 usually generate u-multiplier values which are mostly 0 or 1 with a few values in the range 2 to u_0, for $3 \leqq u_0 \leqq 7$. A good assortment of u-multiplier values usually results in a better assortment of objective coefficients generated. On the other hand, large u-multiplier values result in large component cost vectors which would result in a small relative gap between the optimal objective values of (GCB) and its continuous relaxation.

Procedure A3 guarantees that the value of $u_i^{(k)}$ where i ranges over all of the rows except the fixed row corresponding to component k be relatively prime. Such prevents the coefficients A_{ij} of the fixed row i from being forced to an integral multiple of $g = \gcd_{t \neq i}(a_0, \{u_t^{(k)}\}) \geqq 2$ when no positive v or w multipliers are generated. Such could result is a row which could be simplified as is discussed in (7) of Section 2.1.

Procedure A3 usually generates $s^* > 0$. Then for <u>any</u> column j, $c_j > 0$ and $x^*_j < d_j$ will guarantee that the solution to the continuous relaxation of (GCB) won't solve (GCB). Positive slack values will usually be generated for all rows unless $Q = a_0 \cdot \alpha \cdot (1-\beta)$ is small.

A4. Generation of component weight values, λ_k, such that the objective value coefficients will be about c_0 times the A matrix coefficients, as well as integer valued. This procedure is used in step 3 of Procedure P1.

1. [Generate trial multipliers]

   Set $\hat{\lambda}_k = 1 + \frac{k(k-1)}{p(p-1)}(\sqrt{p}-1)$, $k = 1, 2, \ldots, p$

   (This will generate p weights, the ratio of the smallest to the largest being about $\sqrt{p}$, and the spacings between them being about in the ratios 1: 2:...:p-1. The objective here is to generate a rich set of c_j values in $c = \sum_{k=1}^{p} \lambda_k c^{(k)}$).

2. [Scale the weights]

   Set $\hat{\lambda}_k = \hat{\lambda}_k \cdot \dfrac{n \cdot a_o \cdot c_o}{2 \sum_{k=1}^{p} \sum_{j=1}^{n} \hat{\lambda}_k c_j^{(k)}}$, $k = 1, 2, \ldots, p$.

3. [Round to multiples of $\frac{1}{a_o}$].

   Set $\lambda_k = e(a_o \cdot \hat{\lambda}_k)/a_o$, $k = 1, 2, \ldots, p$.

4. [Make sure at least 2 λ_k's are positive].

   If $\hat{\lambda}_{p-1} = \hat{\lambda}_p = 0$, stop and trigger an error condition. (Such would not result in a good test problem being generated with effectively less than 2 components. Only small values of a_o and c_o could cause this problem. Note $\lambda_1 \leqq \lambda_2 \leqq \cdots \leqq \lambda_p$ at this point.)

5. [Make $\lambda_k \cdot a_o$ values relatively prime].

   If gcd $(\{\lambda_k \cdot a_o\}) = 1$ terminate, otherwise, set $\lambda_{p-1} = \lambda_{p-1} - \frac{1}{a_o}$, $\lambda_p = \lambda_p + \frac{1}{a_o}$, and repeat this step. (This must terminate because $\lambda_{p-1} \cdot a_o$ is reduced to 1 in a finite number of iterations.) ∎

It is desirable to have the values of $\lambda_k \cdot a_o$ relatively prime so that the values of c_j won't be forced to assume an integral multiple of $g = \gcd(\{\lambda_k \cdot a_o\}) \geqq 2$. In practice, it is more convenient to work with _relative_ component cost vectors, $C^{(k)} = \frac{1}{a_o} c^{(k)}$ and _relative_ component weights, $L^{(k)} = a_o \lambda^{(k)}$. Then $c = \sum_{k=1}^{p} L^{(k)} C^{(k)}$ and both $L^{(k)}$ and $C^{(k)}$ are integer valued which circumvents rounding problems in steps 3-5 of the above procedure.

Extensions to the Test Problem Generator

When using Procedure P1 to generate test problems of the form (GCB), it is convenient to be able to specify an upper bounds vector d and a solution vector x^* via single parameters, d_0 and ξ_0. For the test problem generator used to generate the problems in Section 3, the parameters d_0 and ξ_0 were used as follows:

$d_0 > 0$ - set all upper bounds to do.

$d_0 = 0$ - input the d vector explicitly.

$d_0 < 0$ - create mixed upper bounds from 1, 2,...,$-d_0$ in nondecreasing order with about an equal number of each. Specifically, $d_j = 1 + \lfloor (0.5-j) \cdot d_0/n \rfloor$

$\xi_0 \geqq 1$ - set each x_j^* to urn(0, d_j).

$0 < \xi_0 < 1$ - set the fraction of nonzero solution vector values to about ξ_0 such that the nonzero values alternate among 1, $e(\frac{2+d_j}{3})$, $e(\frac{1+2d_j}{3})$, and d_j. Specifically, $x_j^* = 0$ for $j \neq e(\frac{k}{\xi_0})$, j, k integer, $1 \leqq j \leqq n$, the k^{th} nonzero $x_j^* = e((d_j - 1)(k \bmod 4)/3)$.

$\xi_0 \leqq 0$ input the x^* vector explicitly.

It is desirable to generate d and x^* in such away as to avoid problems caused by most solution vector values being 0 or most solution vector values being at upper bound. Values of ξ_0 which lie in (0.0, 0.1) or (0.9, 1.0) are not recommended.

If very few $x_j^* > 0$, the right hand side values, $b = Ax^* + s^*$, may be so small that some $A_{ij} > b_i$ would occur, making $x_j \neq 0$ infeasible. If $\sum_{j=1}^{n} x_j^* = 10$, b_i has an expected value of $5(a_0-1) + s_i^*$ and it is highly unlikely that b_i will be less than some A_{ij} since $A_{ij} \leqq a_0-1$. If very few $x_j^* < d_j$, any solution of the continuous relaxation of (GCB) couldn't differ much from a solution of (GCB), with most variables being at their upper bounds.

After a problem of the form (GCB) has been generated, it is desirable for the solutions $x_j = \begin{cases} d_j & \text{if } j=r,\ r=1, 2,\ldots,n \\ 0 & \text{if } j\neq r \end{cases}$ to be feasible since otherwise, the problem could be somewhat simplified by reducing some upper bounds d_j. The largest value x_j can assume in a feasible solution is given by $\hat{d}_j = \min_{i:\ A_{ij}\neq 0} \lfloor b_i/A_{ij} \rfloor$, $j = 1, 2,\ldots n$.

For the test problem generation used to generate the problems in Section 3, an option is provided to reset $d_j = \min.(d_j, \hat{d}_j)$, $j = 1, 2,\ldots,n$, so that the problem generated will have upper bounds on the variables which cannot be reduced by inspection of the data. If d_j is reduced to 0, however, an error message is triggered because the variable x_j would be superfluous with $x_j \neq 0$ infeasible. Choosing ξ_0 (or x^*) such that the expected sum of the x_j^*'s is at least 10 will make this undesirable phenomena highly unlikely.

Data For the Test Problems

The data generated for the problems described in Section 3 along with solution and slack vectors is as follows where A, B, C, D, X, S, denote respectively the constraint matrix, right hand side vector, objective function vector, upper bounds vector, solution vector, and slack vector for the problem.

Problem 1

S = 8 11 11 21 11 21 5

B = 224 195 205 282 259 194 335

X	C	D	A-TRANSPOSE						
0	72	1	7	25	31	31	25	37	31
1	89	1	48	13	34	49	46	20	20
0	47	1	8	35	5	24	3	44	12
1	81	1	48	6	47	26	8	18	29
0	112	2	34	32	39	33	43	41	43
2	72	2	13	11	4	44	10	37	39
0	49	2	16	8	17	49	15	8	18
1	72	2	8	31	29	26	16	25	37
0	72	3	46	11	16	16	19	21	22
2	88	3	22	16	6	26	45	16	46
0	63	3	1	30	4	4	13	28	41
2	80	3	21	40	32	10	34	2	37

Problem 2

S = 23 12 24 15 8 29 10

B = 630 1013 914 917 610 1036 887

X	C	D	A-TRANSPOSE						
0	202	3	2	47	64	53	14	30	9
0	404	3	90	45	53	69	74	57	1
1	202	3	18	70	43	13	6	84	44
1	341	3	86	70	0	46	52	0	93
0	290	3	48	88	73	59	14	24	2
1	367	3	17	6	70	75	64	62	98
0	316	3	8	26	31	63	71	65	29
0	202	3	45	42	0	15	29	54	44
3	404	3	23	99	55	76	30	96	90
3	290	3	93	83	61	28	13	60	10
3	367	3	8	10	55	82	88	30	66
3	202	3	29	32	20	38	20	42	36
0	404	3	57	63	17	80	42	80	95
0	316	3	31	31	35	38	66	95	31
3	202	3	9	61	68	32	9	59	12

Problem 3

S = 11 25 9 22 12 13 15

B = 1026 320 876 697 1026 793 711

X	C	D	A-TRANSPOSE						
0	104	10	55	26	5	18	46	39	4
4	117	10	50	10	49	44	26	59	31
0	52	10	14	12	3	8	38	34	29
7	65	10	47	7	10	4	9	5	15
0	88	10	32	3	20	32	56	25	36
10	88	10	23	1	49	39	58	36	25
0	75	10	27	11	2	40	53	22	4
1	104	6	32	52	15	53	35	37	41
0	94	10	36	28	56	7	17	56	52
4	117	8	56	36	24	7	58	28	44

Problem 4

S =	9	31	12	9	6	12	14	18	25	19	40
B =	741	763	738	884	858	948	808	738	803	640	808

X	C	D	A-TRANSPOSE										
0	140	2	17	81	89	3	7	45	43	18	33	94	44
1	404	2	102	107	26	80	77	107	45	112	81	40	19
2	337	2	57	69	29	99	88	67	69	38	47	13	90
0	415	2	111	86	110	59	112	78	90	27	70	112	109
2	321	2	29	50	70	49	110	83	59	89	32	27	18
0	420	2	115	115	105	113	82	16	99	77	31	53	85
1	347	2	57	50	56	93	73	72	7	89	95	16	65
1	166	2	27	66	80	4	13	51	91	7	52	69	98
0	228	2	103	42	16	61	6	18	65	69	76	7	3
2	311	2	34	55	32	103	44	118	50	15	108	75	42
0	270	2	95	37	92	24	59	6	104	39	117	114	30
2	254	2	104	40	91	74	11	25	75	16	17	57	69
1	342	2	91	57	90	30	78	45	64	111	102	73	36
0	259	2	16	103	10	7	37	92	119	112	52	113	45
1	285	2	7	24	30	18	105	75	81	85	40	79	112

Problem 5

S =	16	8	22	12	24	24	8	24	42	24	48
B =	985	1058	909	1114	668	853	1026	782	805	904	752

X	C	D	A-TRANSPOSE										
0	158	2	13	5	41	33	93	113	39	106	22	115	86
1	353	2	111	17	102	118	116	74	112	39	16	86	59
2	318	2	68	44	85	100	85	8	91	63	74	111	113
0	384	2	38	97	119	89	106	80	116	10	108	66	52
2	338	2	95	94	57	70	11	93	77	71	39	106	13
0	330	2	22	104	47	41	73	97	106	76	81	39	45
1	318	2	24	80	105	118	25	24	87	92	62	86	14
1	218	2	18	72	51	104	93	63	7	109	93	23	52
0	222	2	88	105	16	20	85	32	2	35	29	103	91
2	333	2	109	112	72	73	85	54	48	92	19	30	43
0	249	2	71	52	9	96	92	97	36	22	71	41	81
2	381	2	113	82	80	71	2	105	106	9	100	76	38
1	337	2	10	119	30	82	26	102	96	15	106	25	113
0	285	2	33	65	66	27	36	83	103	3	93	94	67
1	256	2	36	98	11	52	18	46	72	33	22	14	52

Problem 6

S =	39	39	17	9	20	17	20	20	25	39
B =	463	451	623	493	551	647	624	511	595	526

X	C	D	A-TRANSPOSE									
0	245	1	77	21	72	76	11	28	32	77	53	41
1	177	1	12	85	58	14	62	85	42	18	83	68
0	291	1	57	87	72	86	93	3	54	87	68	47
1	237	1	7	82	93	37	56	31	90	61	50	16
0	114	1	21	26	3	12	80	93	59	64	86	5
1	237	1	20	61	93	48	52	44	98	37	20	9
0	194	1	85	36	17	54	93	93	94	41	19	35
1	211	1	52	1	54	58	38	73	88	39	58	52
0	231	1	72	14	92	32	89	35	59	3	48	88
1	211	1	90	19	79	15	33	28	86	18	99	73
0	97	1	62	74	12	6	17	11	10	20	57	64
1	168	1	71	26	20	54	32	37	0	55	67	96
0	174	1	93	85	24	32	35	99	59	50	21	9
1	134	1	8	24	30	32	76	73	15	30	36	62
0	308	1	79	56	99	71	51	91	76	51	41	79
1	271	1	68	63	79	61	31	67	97	92	13	27
0	131	1	68	0	15	29	40	7	55	88	34	16
1	211	1	44	12	11	92	73	95	87	55	94	24
0	97	1	13	1	8	20	42	15	70	60	50	30
1	282	1	52	39	89	73	78	97	1	86	50	60

Problem 7

S = 10 14 10 14 26 10 26

B = 483 418 411 544 414 522 440

X	C	D	A-TRANSPOSE						
0	160	1	48	14	16	95	14	2	23
0	268	1	12	62	71	65	4	89	50
1	191	1	75	3	6	15	31	54	16
0	306	1	65	61	75	53	19	52	2
0	153	1	5	56	11	59	37	83	26
1	237	1	13	57	32	69	0	93	96
0	244	1	70	49	29	14	52	22	53
0	359	1	79	83	87	6	33	30	48
1	282	1	99	51	27	84	19	11	29
0	213	1	41	70	14	95	20	6	77
0	366	1	96	59	56	55	48	17	89
1	153	1	5	6	51	36	49	19	36
0	153	1	2	13	54	51	48	21	35
0	275	1	71	42	74	57	0	2	8
1	299	1	15	55	98	63	1	95	29
0	275	1	34	76	65	73	63	36	21
0	344	1	94	50	48	56	52	81	10
1	160	1	3	72	28	91	97	5	1
0	275	1	46	86	28	16	86	84	41
0	153	1	14	28	60	10	64	5	1
1	313	1	87	6	19	71	55	80	85
0	251	1	84	99	7	78	53	0	10
0	237	1	5	87	92	59	31	8	16
1	359	1	87	53	88	2	19	24	67
0	237	1	25	20	61	11	33	48	93
0	275	1	20	98	32	74	90	90	49
1	275	1	20	75	46	65	91	74	55
0	184	1	6	31	57	37	54	63	1
0	275	1	6	30	94	96	90	6	68
1	191	1	69	26	6	34	26	57	0

Problem 8

S = 15 13 5 14 10 13 30 14 6

B = 718 363 749 572 650 653 1017 789 629

X	C	D	A-TRANSPOSE								
0	314	1	78	94	20	16	5	51	15	97	99
0	402	1	66	55	88	32	85	37	62	70	92
1	348	1	75	46	46	91	60	44	55	92	84
0	367	1	35	50	97	68	18	92	95	81	50
0	285	1	65	33	88	81	21	63	27	20	5
1	175	1	6	20	43	12	32	45	97	61	24
0	291	1	49	13	70	51	89	75	31	33	38
0	190	1	54	45	2	52	9	2	85	39	67
1	155	1	1	10	55	24	0	65	79	14	9
0	293	1	59	66	59	52	28	95	34	89	11
0	187	2	48	12	40	90	30	3	8	3	36
1	224	2	12	46	45	53	54	2	65	36	58
0	263	2	90	60	37	52	76	13	74	48	19
0	274	2	72	40	36	13	90	93	66	6	40
1	305	2	57	69	81	21	31	47	47	68	26
0	161	2	86	2	8	0	36	24	29	58	34
0	305	2	76	59	67	40	62	71	69	15	19
2	141	2	65	0	5	31	41	25	38	80	18
0	235	2	97	50	1	4	52	16	44	71	70
0	198	2	18	14	5	51	81	83	98	13	59
3	328	3	75	8	94	17	46	22	78	89	57
0	138	3	25	2	38	33	31	66	16	7	1
0	328	3	54	97	44	8	62	92	61	19	68
1	176	3	5	37	6	16	13	41	66	7	95
0	263	3	84	78	46	17	55	1	61	69	23
0	204	3	38	10	50	93	97	0	4	11	23
2	178	3	17	9	16	47	48	75	88	16	44
0	269	3	98	35	28	5	93	2	94	68	56
0	286	3	76	14	73	25	92	25	16	2	52
2	305	3	79	40	72	67	67	65	46	19	16

Problem 9

S = 4 6 10 5 10 10 7 8 4 24 17 6 7 7 12

B = 609 602 616 565 620 488 527 634 592 501 680 581 688 541 468

X	C	D	A-TRANSPOSE														
0	196	1	2	13	11	92	7	25	97	14	25	74	61	33	61	67	61
0	249	1	29	73	92	30	62	61	17	45	23	22	53	89	43	50	80
0	255	1	45	39	36	98	76	83	2	9	24	9	50	60	67	12	68
1	279	1	88	94	11	17	33	10	69	15	46	73	39	57	86	36	24
0	342	1	91	77	5	51	30	54	55	47	98	53	54	75	34	52	94
0	224	1	85	37	51	49	2	45	8	46	24	80	33	47	32	20	66
0	259	1	49	89	15	68	33	14	14	72	84	10	82	26	0	67	49
1	237	1	65	46	82	25	37	51	3	24	70	65	12	32	44	53	42
0	281	1	33	76	2	6	60	12	41	49	97	35	67	79	60	81	62
0	333	1	63	88	3	35	63	75	86	49	71	62	19	69	84	1	95
0	248	1	2	20	64	87	79	55	68	57	1	43	55	86	83	64	27
1	306	1	97	79	55	63	38	95	0	23	75	70	9	21	95	1	18
0	180	1	80	5	27	22	22	29	25	46	2	34	64	12	92	60	56
0	259	1	3	88	90	86	21	74	85	10	61	68	78	2	96	2	5
0	210	1	33	3	69	36	32	93	57	91	14	70	86	5	86	65	76
1	262	1	36	69	75	32	88	14	71	86	24	5	18	97	32	11	33
0	266	1	15	65	31	49	79	43	40	53	57	32	62	86	42	69	33
0	244	1	8	96	6	24	99	68	22	68	49	71	45	42	29	80	49
0	360	1	68	92	93	67	42	23	89	46	70	57	20	79	90	7	74
1	204	1	24	75	35	41	36	31	30	99	17	21	86	24	47	56	37
0	267	1	14	70	13	86	16	38	29	1	98	8	30	69	63	80	40
0	271	1	7	25	28	97	88	43	77	35	82	27	85	68	11	10	37
0	260	1	89	36	36	97	69	28	10	29	62	25	33	31	3	6	15
1	263	1	51	21	49	0	76	4	69	48	95	24	58	88	23	14	54
0	154	1	9	17	13	5	33	16	41	17	8	90	11	71	73	63	47
0	283	1	83	9	46	88	7	31	86	25	26	63	44	83	66	98	56
0	292	1	70	93	87	9	71	33	39	86	48	78	48	38	26	86	91
1	218	1	10	61	81	50	81	55	65	65	6	94	51	3	56	89	24
0	209	1	17	49	75	49	40	66	93	11	39	14	65	39	25	29	38
0	204	1	23	56	55	63	87	55	32	54	4	1	14	27	60	59	5
0	259	1	52	33	30	27	96	12	87	93	89	5	38	24	9	76	20
1	313	1	83	3	15	73	30	49	72	81	98	4	41	75	62	80	14
0	257	1	9	77	1	95	70	0	4	61	93	76	56	32	9	43	4
0	279	1	91	85	46	35	56	33	61	3	19	11	79	98	14	1	93
0	247	1	51	54	85	52	48	88	74	10	35	70	32	22	46	24	61
1	259	1	6	86	53	88	40	59	33	10	57	0	74	50	61	14	84
0	220	1	81	7	51	48	69	41	24	20	30	23	22	29	65	33	31
0	318	1	93	42	23	48	98	27	27	54	32	69	35	96	94	91	9
0	238	1	5	63	59	9	48	18	61	92	50	16	12	93	95	1	2
1	276	1	11	15	33	97	82	25	30	59	38	10	89	88	84	90	74
0	163	1	13	34	45	7	57	25	43	23	24	48	92	55	13	34	47
0	222	1	54	49	47	17	29	94	9	85	8	72	16	72	51	29	65
0	260	1	93	16	18	92	67	64	62	42	1	9	6	71	29	85	31
1	147	1	49	0	28	15	21	7	72	36	10	48	94	9	67	32	14
0	229	1	47	33	57	94	38	49	4	61	31	6	41	4	52	76	96
0	153	1	1	19	62	0	97	67	24	48	28	9	85	30	34	46	11
0	267	1	92	77	8	47	25	79	81	36	46	67	43	2	26	47	88
1	282	1	85	47	89	59	48	78	6	80	52	63	92	31	24	58	38
0	277	1	75	87	34	46	66	20	4	83	31	69	74	60	30	49	33
0	248	1	88	18	60	15	9	0	93	92	47	22	97	22	67	68	73

Problem 10

S = 41 21 18 13 18 41 37 21 18 25 9 18 18 9 18

B = 690 664 683 626 434 587 616 662 721 636 596 510 752 607 475

X	C	D	A-TRANSPOSE													
0	286	1	38	81	86	84	44	34	63	78	23	52	44	11	4	83	73
0	286	1	18	65	32	79	39	56	65	4	9	56	74	82	77	51	16
0	231	1	23	11	94	47	50	40	5	29	87	48	60	54	18	74	16
1	299	1	90	53	7	18	53	28	83	94	4	99	99	99	24	85	40
0	264	1	11	15	91	33	31	50	39	54	13	28	63	6	90	98	58
0	172	1	29	26	36	67	46	7	94	2	9	40	7	91	41	34	56
0	245	1	51	44	75	68	23	88	52	79	8	80	83	24	16	45	40
1	273	1	88	93	3	69	48	37	16	42	90	16	65	58	75	1	28
0	180	1	0	0	9	31	87	70	35	69	26	33	3	9	46	98	49
0	310	1	38	97	56	45	86	56	37	17	53	19	29	96	54	62	97
0	214	1	80	60	6	25	26	13	45	96	49	28	3	46	96	34	40
1	227	1	26	33	74	41	80	25	61	33	83	98	22	30	68	48	1
0	276	1	6	95	29	60	16	45	1	34	76	38	97	40	32	19	53
0	272	1	84	98	68	23	7	80	76	94	9	2	99	85	8	8	85
0	146	1	51	28	95	34	30	40	71	14	20	94	10	37	17	19	30
1	216	1	13	11	65	43	89	48	9	46	45	50	75	54	68	31	1
0	209	1	4	28	21	12	20	59	60	84	82	27	88	77	48	13	7
0	298	1	80	79	96	32	58	89	78	86	31	89	16	93	63	56	58
0	272	1	96	56	87	49	46	92	88	53	32	61	24	74	35	64	79
1	243	1	73	29	42	55	19	18	22	73	89	24	38	10	80	57	94
0	213	1	1	74	14	48	65	79	66	51	38	86	15	20	2	53	48
0	313	1	19	90	79	68	15	44	15	14	22	17	74	55	77	58	66
0	231	1	28	21	54	28	18	51	27	44	85	44	65	30	24	81	8
1	276	1	69	97	95	15	11	89	91	17	86	94	23	10	47	52	31
0	258	1	28	84	79	54	81	61	86	19	61	60	9	97	60	9	95
0	214	1	93	62	59	4	10	33	6	2	98	90	7	2	67	45	43
0	286	1	70	54	0	63	43	83	91	67	64	45	76	1	73	62	17
1	251	1	47	87	39	74	37	39	12	79	1	13	19	30	92	43	11
0	361	1	83	69	91	37	8	94	39	81	84	24	81	43	94	85	64
0	231	1	13	90	12	3	86	7	16	84	25	81	56	3	53	24	60
0	312	1	75	43	39	17	97	41	63	9	80	19	90	71	93	54	96
1	243	1	72	22	91	77	23	1	14	51	25	9	14	84	97	72	28
0	222	1	64	91	24	14	5	94	62	21	77	23	1	8	1	71	55
0	247	1	5	94	39	80	63	25	1	15	18	19	36	26	40	42	69
0	217	1	64	83	80	29	17	90	80	8	40	92	25	56	30	9	37
1	220	1	18	18	67	91	14	91	65	91	45	44	67	44	19	29	92
0	246	1	33	88	41	74	51	38	43	92	15	84	40	14	49	18	82
0	217	1	2	26	90	0	54	74	86	98	51	14	5	19	27	96	72
0	294	1	97	68	41	89	68	43	64	13	27	46	41	55	89	48	77
1	224	1	85	59	62	73	18	68	76	42	66	51	31	11	15	45	9
0	273	1	51	85	79	48	40	68	15	2	46	36	19	21	47	84	24
0	239	1	50	82	7	68	89	49	91	87	77	40	32	8	55	3	65
0	202	1	77	43	21	46	29	11	16	77	78	38	32	25	16	59	0
1	306	1	1	81	43	38	12	81	39	60	75	22	77	28	96	44	69
0	165	1	55	0	92	24	31	76	78	20	55	97	17	36	82	2	90
0	302	1	5	77	50	28	87	95	52	66	83	20	59	92	16	68	57
0	168	1	72	10	42	67	43	23	55	61	66	77	37	27	31	12	58
1	304	1	67	60	77	19	12	21	91	13	94	91	57	34	53	91	53
0	261	1	49	65	79	7	19	27	53	45	46	40	54	86	75	39	42
0	335	1	77	71	74	82	21	47	87	41	94	31	27	74	77	89	41

REFERENCES

1. J. M. Fleisher and R. R. Meyer, A New Class of Sufficient Optimality Conditions for Integer Programming, Computer Sciences Technical Report No. 248, University of Wisconsin, Madison, April, 1975.

2. R. Garfinkel and G. Nemhauser, Integer Programming, John Wiley & Sons, N.Y., 1972.

3. R. G. Jeroslow and T. H. Smith, Experimental Results on Hillier's Search Imbedded in a Branch and Bound Algorithm, Management Sciences Research Report No. 326, Graduate School of Industrial Administration, Carnegie-Mellon University, November, 1973.

4. R. Shareshain, Branch and Bound Mixed Integer Programming - BBMIP, IBM, New York Scientific Center, Share Program Library, 360D-15.2.005, 1967.

5. L. Trotter, Jr., User's Instructions for the Integer Programming Code ENUMER8, MRC Technical Summary Report No. 1347, Mathematics Research Center, University of Wisconsin, Madison, December, 1973.

6. L. Trotter, Jr. and C. Shetty, An Algorithm for the Bounded Variable Integer Programming Problem, MRC Technical Summary Report No. 1355, Mathematics Research Center, University of Wisconsin, Madison, December, 1973.

7. H. Wagner, Principles of Operations Research, Prentice Hall, Inc., N. J., 1969.