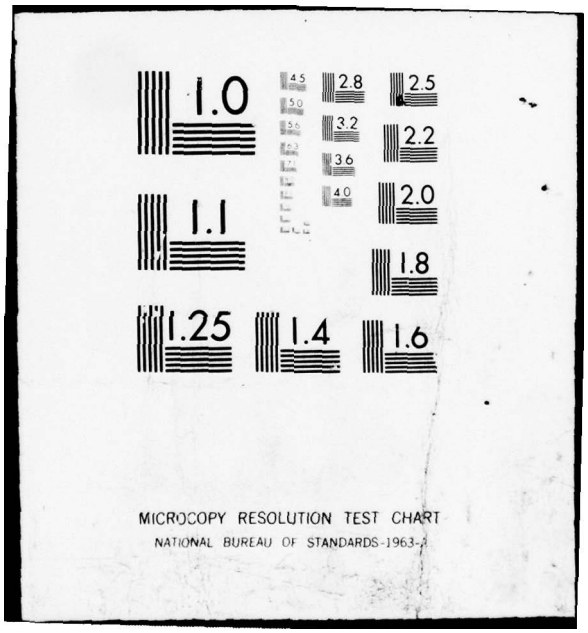
AD-A059 873 CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENC--ETC F/G 12/2
PIVOT AND COMPLEMENT - A HEURISTIC FOR 0-1 PROGRAMMING.(U)
FEB 78 E BALAS, C H MARTIN N00014-75-C-0621

UNCLASSIFIED MSRR-414 NL

1 OF 1
AD A059873

END
DATE FILMED
12-78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

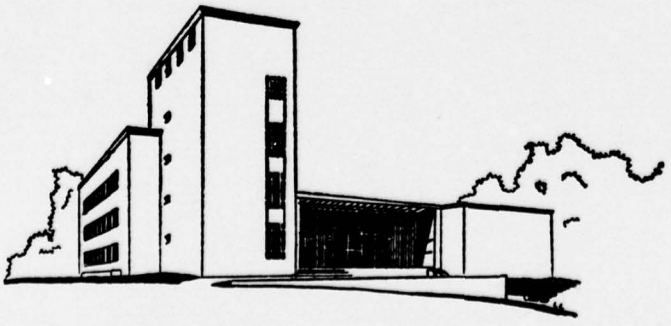
LEVEL II

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



(9) Management Sciences Research Report No. 414

(6) PIVOT AND COMPLEMENT - A HEURISTIC FOR 0-1 PROGRAMMING

by

(10) Egon Balas and Clarence H. Martin

(11) February 1978

(12) 41 p.

This report was prepared as part of the activities of the Management Science Research Group, Carnegie-Mellon University, under Grant MPS73-08534 A02 of the National Science Foundation and Contract N00014-75-C-0621 NR 047-048 with the U.S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

(15) NSF-MPS73-08534

Management Science Research Group
Gradate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

The pivot and complement procedure is a heuristic for finding approximate solutions to 0-1 programming problems. It uses the fact that a 0-1 program is equivalent to the associated linear program with the added requirement that all slack variables be basic. The procedure starts by solving the linear program; then performs a sequence of pivots aimed at putting all slacks into the basis at a minimal cost in dual feasibility, while taking care of occasionally arising primal infeasibilities by complementing some nonbasic 0-1 variables; finally, it attempts to improve the 0-1 solution obtained in this way by a local search based again on complementing certain sets of 0-1 variables.

The computational effort involved in the procedure is bounded by a polynomial in the number of constraints and variables. For the 67 test problems with 20-200 variables and 5-37 constraints on which the procedure was run, the time used to solve the linear program on the average considerably exceeded the time used for everything else, and the total time never exceeded a few seconds. As to the quality of the solutions obtained, in 23 cases, or one third of the total, the procedure found an optimal solution; for all the capital budgeting problems (positive coefficients everywhere), the solution found was within 0.15% of the optimum; for 58 of the 67 problems, it was within 1% of the optimum; and only in 1 case did the procedure not find a feasible solution.

PIVOT AND COMPLEMENT - A HEURISTIC FOR 0-1 PROGRAMMING

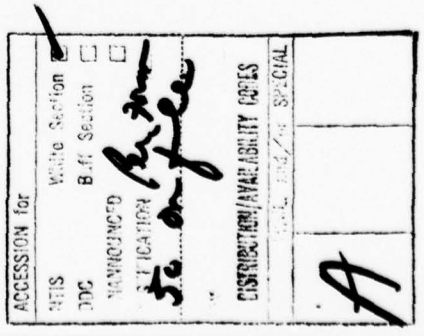
by

Egon Balas and Clarence H. Martin

1. Introduction

Since large integer programs are hard to solve, practitioners facing such problems and researchers trying to help them have always had an interest in heuristics as a means of finding "good" approximate solutions. The various enumerative algorithms for solving integer programs make ample use of heuristics, and the commercial codes based on these algorithms are often run in the "heuristic mode," i.e., they are stopped after producing a feasible solution that looks satisfactory. Several heuristic procedures for 0-1 or general integer programming were proposed as approximate solution methods in their own right, and tested with varying degree of success during the last 15 years (see, among others, the papers [24], [23], [5], [26], [8], [28], [9], [19], [21], [12], [27], [10]). Many more heuristic procedures were proposed for various special combinatorial structures, like scheduling and location problems, the traveling salesman problem, the quadratic assignment problem, etc.

While this research had strictly practical goals and its sole criterion of success was empirical, namely statistical performance on randomly generated test problems and/or some samples of real-world problems, a rather different and more recent approach is aimed at devising heuristics



with a guaranteed performance, namely procedures which are guaranteed to get within a given proximity of the optimum by a computational effort which is a polynomial function of the input. This approach, pioneered in the mid-sixties by Graham [7], has attracted a large number of researchers after 1972, mainly as a consequence of the NP-completeness results of Cook [3] and Karp [14], and has led to a plethora of theoretically interesting results, typified by (but by no means restricted to) [15], [13], [11], [25], [2], [4], etc. (see Garey and Johnson [6] and Korte [17] for two recent surveys).

This work has yielded (and continues to yield) many interesting insights; however, its underlying "worst-case analysis" approach has two important shortcomings. One consists in the fact that the proximity to the optimum that one is guaranteed to attain in polynomial time is usually unsatisfactory for practical purposes. The other one is that usually there is no strong correlation between the average and worst-case performance of heuristics (or, for that matter, exact algorithms). This situation has prompted yet another line of research, aimed at analyzing the expected, rather than worst-case, performance of algorithms. Work in this direction is typified by the papers of Karp [16] and Rabin [22]. The questions asked in this approach undoubtedly have more practical relevance than those involved in worst-case analysis, and though the answers seem harder to come by, there is little doubt that this line of research holds much promise and will in time yield some practical results. In the meantime, practitioners

will continue to judge approximate and exact solution procedures alike, by the empirical criterion of observed performance.

The heuristic procedure discussed in this paper, designed to "solve" (in the sense of approximating the optimum) arbitrary 0-1 programs, is mathematically unexciting, but has performed remarkably well, both from the point of view of the computational effort involved, and from that of the quality of the solutions obtained, on a variety of test problems, some randomly generated and some taken from the real world, ranging in size from 20 to 200 variables and from 5 to 37 constraints. The approach uses the fact that a 0-1 program (P) is equivalent to the associated linear program (LP), plus the requirement that all slack variables be basic. Thus the pivot and complement procedure starts by solving (LP), then performs a sequence of pivots aimed at putting into the basis all nonbasic slack variables at a minimal cost in dual feasibility, while eliminating occasional primal infeasibilities by complementing some variables; finally, it attempts to improve the 0-1 solution obtained in this way by a local search based again on complementing certain sets of variables.

The computational effort involved in the procedure is bounded by a polynomial in the number of variables and constraints; but this bound is a very weak one, and a more relevant piece of information is the fact that the time spent on solving the linear program on the average considerably exceeds the time spent on everything else.

As to the quality of the solutions obtained, there is no guarantee of it, not even of obtaining a feasible solution at all, except for the

class of problems with all coefficients nonnegative. But of the 67 test problems solved, the procedure found feasible solutions in 66 cases, and optimal solutions in 23 cases, one third of the total. Furthermore, in at least 58 cases the solution was within 1% of the optimum, and for the class of problems with all coefficients nonnegative, the solutions were on the average within 0.15% of the optimum.

In the next section (2) we give a detailed description of the pivot and complement procedure. Section 3 establishes a bound on the computational effort involved in the procedure, while section 4 discusses our computational experience. Finally, section 5 examines the effect of our heuristic upon a branch and bound procedure which uses it to find a starting solution.

2. Description of the Pivot and Complement Procedure

The problem we address in this paper is

$$\text{(ZOP)} \qquad \text{maximize} \quad \sum_{j \in N} c_j x_j$$

$$\text{subject to} \quad \sum_{j \in N} a_{ij} x_j \leq b_i, \qquad i = 1,\ldots,m$$

$$x_j = 0 \text{ or } 1 \qquad j \in N$$

where $N = \{1,2,\ldots,n\}$, and each c_j is assumed to be integer. The pivot and complement procedure begins by solving the linear programming relaxation of (ZOP), namely

$$\text{(LP)} \qquad \text{maximize} \quad \sum_{j\in N} c_j x_j$$

$$\text{subject to} \quad \sum_{j\in N} a_{ij} x_j \le b_i, \quad i = 1,\ldots,m$$

$$0 \le x_j \le 1, \quad j\in N,$$

by the simplex method for variables with upper bounds.

We will use the following representation of the optimal tableau:

$$x_i = a'_{io} + \sum_{j\in J} a'_{ij}(-x_j) \quad i\in\{0\}\cup I,$$

where I and J represent the basic and nonbasic sets respectively. Each variable at its upper bound of 1 is complemented (complementing x_j means substituting $x_j = 1 - x'_j$) and we have $a'_{io} \ge 0$, $i\in I$, $a'_{io} \le 1$, $i\in I\cap N$, and $a'_{oj} \ge 0$, $j\in J$.

The pivot and complement procedure has a search phase and an improvement phase. The first one attempts to reach a "good" feasible 0-1 point by pivoting and sometimes complementing variables, the second one attempts to improve the solution found in the first phase (if any) by complementing certain sets of variables. Before describing the procedure in its entirety, we introduce the concepts upon which it is based.

Pivots of Type 1

A pivot of type 1 is one that maintains primal feasibility and exchanges a nonbasic slack for a 0-1 variable in the basis. Thus we

seek a pivot in a column $q \in J \setminus N$ and a row $p \in I \cap N$ where p yields the minimum in

$$\min\{ \min_{\substack{i \in I \\ a'_{iq} > 0}} \{a'_{io}/a'_{iq}\} , \min_{\substack{i \in I \cap N \\ a'_{iq} < 0}} \{(a'_{io} - 1)/a'_{iq}\}\} .$$

If the minimum is obtained for $a'_{iq} > 0$, the 0-1 variable leaves the basis at 0; otherwise, it leaves at 1. Each time a pivot of type 1 is performed, the number of fractional 0-1 variables decreases by at least one.

Pivots of Type 2

A pivot of type 2 is one that maintains primal feasibility and satisfies two additional requirements. The first is that it leave unchanged the number of basic 0-1 variables. This restricts the nonbasic variable entering the basis and the basic variable leaving the basis to be either both 0-1 variables or both slack variables. The second requirement is that the pivot strictly improve some measure of the nearness of the solution to integrality. One measure is the sum of integer infeasibilities defined as

$$\sum_{i \in I \cap N} \min\{a'_{io}, 1 - a'_{io}\} .$$

A second measure is the absolute value of the determinant of the basis. Both measures were tested computationally and the results slightly favored the former. The results reported in this paper were obtained by requiring a pivot of type 2 to reduce the sum of integer infeasibilities by at least .01.

Pivots of Type 3

A pivot of type 3 is one that exchanges a nonbasic slack for a basic 0-1 variable, while sacrificing primal feasibility. The slack variable is restricted to enter the basis at a nonnegative level. Thus, if the pivot element is positive, the 0-1 variable leaves the basis at 0; otherwise, it leaves the basis at 1. The resulting solution is infeasible.

Complementing Variables in the Search Phase

By complementing a nonbasic 0-1 variable y_i (which may either be an original 0-1 variable, or its complement) we mean substituting $y_i = 1 - y_i'$. In the search phase we complement variables when the current solution is infeasible, to reduce the measure of infeasibility defined by

$$\sum_{i \in I} \max\{0, - a'_{io}\} .$$

Thus a set $S \subseteq J \cap N$ of nonbasic variables is a search phase candidate set for complementing if

$$\sum_{i \in I} \max\{0, - a'_{io}\} > \sum_{i \in I} \max\{0, - a'_{io} + \sum_{j \in S} a'_{ij}\}.$$

We restrict the cardinality of such sets S to 1 or 2, and require the decrease in the infeasibility measure to be at least .01.

Complementing Variables in the Improvement Phase

The improvement phase begins with the feasible zero-one solution identified in the search phase. We seek to improve this solution by

complementing some variables. Let N^1 and N^0 be the sets of 0-1 variables currently equal to 1 and 0 respectively, and define

$$d_j = \begin{cases} c_j & j \in N^0 \\ -c_j & j \in N^1 \end{cases} \qquad \text{and}$$

$$f_{ij} = \begin{cases} a_{ij} & j \in N^0 \\ -a_{ij} & j \in N^1 \end{cases} \qquad i = 1,\ldots,m.$$

We identify a set $S \subseteq N$ as an improvement phase candidate set for complementing if

$$\sum_{j \in S} d_j > 0 \qquad \text{and}$$

$$\sum_{j \in S} f_{ij} \leq b_i - \sum_{j \in N^1} a_{ij} \qquad i = 1,\ldots,m.$$

Complementing the variables in such a set thus yields a new feasible 0-1 solution with a strictly improved objective function value. We restrict the cardinality of these sets to 1, 2, or 3.

Variable Fixing

Let $\underline{z}$ be the objective function value of the current 0-1 solution, z_{LP} be the objective function value for the optimum solution to (LP), and $\bar{c}_j$ be the reduced cost of variable $j \in N$ at the LP optimum. Observe that $\bar{c}_j = 0$ if variable j is basic at the LP optimum, $\bar{c}_j \geq 0$ if it is nonbasic at 0,

and $\bar{c}_j \leq 0$ if it is nonbasic at 1. It is clear that if $|\bar{c}_j| > z_{LP} - \underline{z} - 1$ then in any 0-1 solution better than the current solution, variable j must assume the value it held in the LP optimum. Furthermore, if $|\bar{c}_j| > z_{LP} - \underline{z}$ then variable j must have this same value in the current solution. Thus whenever we encounter a 0-1 variable whose reduced cost exceeds $z_{LP} - \underline{z} - 1$ and which has the same value in the current solution to (ZOP) as in the original solution to (LP), this variable may be fixed at that value for the remainder of the procedure.

Rounding and Truncation Tests

At certain stages of the procedure the current basic solution is checked to determine whether rounding or truncating this solution will provide a feasible solution to (ZOP). The rounded solution is obtained by rounding all fractional 0-1 variables to the nearer integer value. If this solution is not feasible we then check the solution where all fractional 0-1 variables are truncated to 0.

The following is a statement of the pivot and complement procedure:

SEARCH PHASE

STEP 1: Solve (LP). If the solution is integer, STOP: it is optimal. Otherwise go to STEP 2.

STEP 2: Search for a pivot of type 1. If none exists, go to STEP 3. Otherwise perform the pivot of type 1 that results in the largest objective function value and to to STEP 4.

STEP 3: Search for a pivot of type 2. If none exists, go to STEP 5. Otherwise perform the first pivot of type 2 found and go to STEP 4.

STEP 4: Examine the current basic solution. If it is integer, go to IMPROVEMENT PHASE. Otherwise, go to STEP 2.

STEP 5: Check whether rounding or truncating the current basic solution yields a feasible integer solution. If so, go to IMPROVEMENT PHASE. If not, go to STEP 6.

STEP 6: Perform a pivot of type 3. Among all such possible pivots, choose the one that minimizes the resulting infeasibility. Go to STEP 7.

STEP 7: Search for a single nonbasic 0-1 variable that can be complemented to decrease the current value of the infeasibility measure. If none exist, go to STEP 9. Otherwise, complement the nonbasic 0-1 variable yielding the largest improvement in the measure of infeasibility and go to STEP 8.

STEP 8: If the current solution is infeasible, go to STEP 7. If it is feasible, check whether rounding or truncating the solution will yield a solution to (ZOP). If so, go to IMPROVEMENT PHASE. If not, go to STEP 2.

STEP 9: Search for a pair of nonbasic variables that may be complemented to decrease the current value of the infeasibility measure. If none exist, STOP, the procedure has failed. Otherwise, complement the first such pair identified and go to STEP 8.

IMPROVEMENT PHASE

STEP 1: Fix all free 0-1 variables that may be fixed on the basis of reduced cost and the difference between the objective function values of the (LP) optimum and the current 0-1 solution. Go to STEP 2.

STEP 2: Search for a single 0-1 variable that may be complemented to yield an improved 0-1 solution. If none exist, go to STEP 3. Otherwise, complement the 0-1 variable that yields the best 0-1 solution among all candidates and go to STEP 1.

STEP 3: Search for a pair of 0-1 variables that may be complemented to yield an improved 0-1 solution. If none exist, go to STEP 4. Otherwise, complement the first candidate pair identified and go to STEP 1.

STEP 4: Search for a triplet of 0-1 variables that may be complemented to yield an improved 0-1 solution. If none exist, STOP, the current 0-1 solution is the final solution of the procedure. Otherwise, complement the first candidate triplet identified and go to STEP 1.

To complete the description of the procedure, we furnish the following details of implementation.

At the beginning of the improvement phase, the 0-1 variables are ordered according to increasing absolute values of the reduced costs of the optimal (LP) tableau. Variables are fixed by simply removing those at the end of the list from further consideration. The search for improvement Phase complements proceeds from the beginning of the list.

When searching for potential improvement phase single and double complements, all variables and pairs of variables in the set of free variables are examined. However, when searching for potential triple complements, the index of the first variable of the triplet is restricted to the first third of the free variables ordered

by reduced costs. In searching for all three types of improvement phase complements, the objective function value improvement criteria are checked first. Then before testing for feasibility, the objective function improvement is checked to see if it would make the objective function larger than the objective function value of the LP optimum. If it does, the more expensive tests for feasibility are not needed since the resulting solution must be infeasible. Finally, when checking for feasibility, the constraints are examined in order of increasing s_i, where

$$s_i = b_i - \sum_{j \in N^1} a_{ij} \qquad i = 1,\ldots,m.$$

3. A Bound on the Computational Effort

Next we calculate a bound on the amount of computation the pivot and complement procedure may require after the solution of the linear program.

Let m be the number of constraints, n the number of 0-1 variables, and Δ the minimum acceptable decrease in the sum of integer infeasibilities (in step 3) or in the sum of common infeasibilities (in steps 7 and 9).

We start by listing the number of arithmetic operations required in the worst case by each step of the procedure, following the initial solution of the linear program.

Search Phase

Step 2a (trying a pivot of type 1):

m(m + 1) multiplications

m(m + 1) comparisons

Step 2b (executing the pivot):

mn multiplications

mn additions

Step 3a (trying a pivot of type 2):

$2mn$ multiplications

$m(2n + 1)$ additions

$(3m + 1)n + 2m$ comparisons

Step 3b (executing the pivot):

mn multiplications

mn additions

Step 4: $2m$ comparisons

Step 5: m^2 multiplications

$2m^2$ additions

$4m$ comparisons

Step 6: $m^2 + mn$ multiplications

$m^2 + mn$ additions

$m(m + 1)$ comparisons

Step 7: $2mn$ additions

$(m + 1)n$ comparisons

Step 8a (with return to Step 7):

m comparisons

Step 8b (with return to Step 2, or end of search phase):

m^2 multiplications

$2m^2$ additions

$4m$ comparisons

Step 9: $mn(n-1)$ additions

$\frac{1}{2}(m+1)n(n-1)$ comparisons

Improvement Phase

Step 1: n comparisons

Step 2: $2mn$ additions

mn comparisons

Step 3: $mn(n-1)$ additions

$\frac{1}{2}mn(n-1)$ comparisons

Step 4: $\frac{1}{9}mn(n-1)(n-2)$ additions

$\frac{1}{18}mn(n-1)(n-2)$ comparisons

The total number of pivots of types 1 and 3 cannot exceed the number m of slack variables. Thus there are at most m sequences of steps ending in step 2b or 6. The longest such sequence is one that starts (after step 6) with a subsequence S_1 of the form 7,9,8a,...,7,9,8a,...,7,9,8b,2 and continues with a subsequence S_2 of the form 2a,3,4,...,2a,3,4,...,2a,3,5,6. Thus, the computational effort required by the Search Phase of the procedure is at most m times the computational effort required by S_1 and S_2.

To evaluate the effort involved in a sequence S_1, note that the number of cycles 7,9,8a that can be executed consecutively *so as to* reduce total infeasibility each time by at least Δ, is bounded by I/Δ, where I *is an upper bound on* the sum of infeasibilities. Since infeasibility

only occurs as a result of a pivot of type 3, its maximal amount is at most

$$I = \max_j \sum_i |a_{ij}|.$$

Since the sequence S_1 consists of at most I/Δ cycles 7,9,8a, with 8a replaced in the last cycle by 8b, it requires at most

m^2 multiplications,

$I\,m\,n\,(n+1)/\Delta + 2m^2$ additions,

and

$I[(m+1)n(n+1) + 2m]/2\Delta + 3m$ comparisons.

To evaluate the effort involved in the sequence S_2, we observe that every application of step 4b reduces the sum of integer infeasibilities by at least Δ, and that this sum is at most $m/2$. Therefore S_2 contains at most $m/2\Delta$ cycles 2a,3,4 with step 4 replaced in the last cycle by steps 5,6; and the sequence S_2 requires at most

$[m^3 + m^2(3n+1)]/2\Delta + 2m^2 + mn$ multiplications,

$m^2(3n+1)/2\Delta + 3m^2 + mn$ additions,

and

$[m^3 + m^2(3n+5) + mn]/2\Delta + m^2 + 3m$ comparisons.

Thus the total computational effort required by the search phase is at most

$$[m^4 + m^3(3n + 6\Delta + 1) + 2m^2 n\Delta]/2\Delta \text{ multiplications,}$$

$$[m^3(3n + 10\Delta + 1) + 2m^2(In^2 + In + \Delta n)]/2\Delta \text{ additions,}$$

and

$$[m^4 + m^3(3n + 2\Delta + 5) + m^2(In^2 + In + n + 2I + 12\Delta)]/2\Delta$$

comparisons

As to the improvement phase, since the costs are integer, complementing a variable increases the value of the objective function by at least 1; hence $C = \sum_j |c_j|$ is an upper bound on the number of times such complementing can occur. The computationally most expensive way of complementing variables in this phase is to go through steps 1, 2, and 3 without finding anything to complement, and then complementing 3 variables at the end of step 4. If all complementing in the improvement phase occurs in this way (which is the worst possible case), the computational effort involved in this phase is bounded by

$$Cmn(n^2 + 6n + 11)/9 \text{ additions}$$

and

$$Cn[m(n - 1)(n + 7) + 18(m + 1)]/18 \text{ comparisons}$$

Thus the pivot and complement procedure as a whole requires in the worst case $0(m^4 + m^3 n)$ multiplications, $0(m^3 n)$ additions, and $0(m^4 + m^3 n + m^2 n^2)$ comparisons.

4. Computational Results

The performance of the pivot and complement procedure was evaluated using a set of 67 test problems. Information on these test problems, including their size, LP optimum, and 0-1 optimum (where known) is presented in Tables I, II, and III. The problems in the PET series are from Petersen [20], those in the ST series are from Senju and Toyoda [26], those in the JS series are from Jeroslow and Smith [12], while those in the CB series were generated by us. All these problems are of the capital budgeting type. Those in the PET series have a real world origin. Those in the ST, JS and CB series are randomly generated. The ST problems are published in full in [26].

The JS and CB problems were generated as follows. The costs c_j are random integers between 0 and 99; each a_{ij} is a random number between 0 and 9, multipled by $c_j/10$; while b_i is a random number between 5 and 9, multiplied by $s_i/10$, with $s_i = \sum_j a_{ij}$. The resulting capital budgeting problems are somewhat harder, and also more realistic, than problems whose coefficients are entirely random, with no relation between c_j and the corresponding a_{ij}.

Since all these problems have only nonnegative coefficients and therefore a feasible 0-1 solution can always be found by truncating a feasible LP solution, we generated the RG series by explicitly excluding all problems in which either truncating or rounding the LP solution yielded a feasible 0-1 solution. Subject to this condition, the problems in the RG series were generated as follows. The c_j were randomly drawn from

the integers between 2 and 11, the a_{ij} from the integers between -4 and 7, and the b_i were generated in the same way as for the JS and CB series.

The problems in the BM series are from Bouvier and Messoumian [1], where they are given in full. They are randomly generated, with positive and negative entries in both the constraint matrix and the right-hand side vector, and very tightly constrained, some with only a few feasible solutions.

Finally, the LS problems are from Lemke and Spielberg [18]. They are real-world problems and have positive and negative coefficients in both the coefficient matrix and the right-hand side vector.

The code for the pivot and complement procedure was written in FORTRAN IV and run with single precision arithmetic on the UNIVAC 1108 at CMU (which uses the operating system EXEC II), with the exception of the BM and LS series, which were run (on the same computer) with double precision. The code was run in two versions. The first version did not attempt triple complements in the improvement phase, while the second did. The results for the version without triple complements appear in Tables IV, V, and VI, and the results for the version with triple complements appear in Tables VII, VIII, and IX.

The following observations characterize the performance of the procedure:

1. The search phase found 0-1 solutions for 66 of the 67 test problems, the exception being BM 21.

2. The 0-1 solution found by the procedure was optimal for 22 (or 1/3) of the 66 problems solved, and was within 1% of the optimum

for 58 problems. For the problems with positive coefficients (PET, ST, JS, CB), the 0-1 solution found by the procedure was within 0.15% of the optimum.

3. The search phase found a 0-1 solution using only pivots of type 1 in 47 problems. These included all 41 of the capital budgeting problems (PET, ST, JS, and CB series) and 6 problems in the RG series.

4. The search phase needed pivots of type 3 only for problems in the BM and LS series. In the remaining problems, steps 1 through 5 were sufficient to produce 0-1 solutions.

5. The improvement phase improved upon the 0-1 solutions identified in the search phase in 58 of the 66 problems in which a 0-1 solution was identified. For 7 out of the remaining 8 problems the solution found in the search phase was optimal. Thus in all but one of the cases when the solution found in the search phase was not optimal, a better solution was found in the improvement phase.

6. The version that included triple complements produced better final solutions than the version that did not include triple complements in 31 out of 66 problems. The increase in run times was between 10 and 130%.

7. The procedure succeeded in fixing 45% of all the 0-1 variables in the 67 problems. However, at least for the problems in the JS, CB, and RG series, the number of variables fixed tended to decrease with the increase in the number of constraints.

8. The sum of the number of pivots (of all three types) and of the number of complements in the entire procedure was always less than twice the number of constraints.

TABLE I

TEST PROBLEM INFORMATION

Problem	Size		L.P.	I.P.
	M	N	Optimum	Optimum
PET 4	10	20	6155.3	6120
PET 5	10	28	12462.1	12400
PET 6	5	39	10672.3	10618
PET 7	5	50	16612.8	16537
ST A	30	60	7839.3	7772
ST B	30	60	8773.2	8722
JS 1	5	100	3324.5	3320
JS 2	5	100	3364.9	3363
JS 3	5	100	2727.5	2720
JS 4	5	100	3467.9	3467
JS 5	5	100	3983.9	3977
JS 6	5	100	3488.1	3483
JS 7	5	100	4530.7	4523
JS 8	5	100	3369.2	3366
JS 9	5	100	3948.1	3945
JS 10	5	100	3859.9	3854
JS 11	10	100	3507.4	3496
JS 12	10	100	3119.2	3108
JS 13	10	100	3629.7	3617
JS 14	10	100	3722.6	3717
JS 15	10	100	3416.4	3407
JS 16	10	100	3126.3	3120
JS 17	10	100	2939.5	2938
JS 18	10	100	3207.8	3201
JS 19	10	100	3028.4	3020
JS 20	10	100	3256.4	3250

PET - Petersen [20]

ST - Senju and Toyoda [26]

JS - Jeroslow and Smith [12]

TABLE 11

TEST PROBLEM INFORMATION

Problem	Size M	Size N	L.P. Optimum	I.P. Optimum
CB 1	5	200	7721.2	7718
CB 2	5	200	6955.7	6951
CB 3	5	200	7171.8	7170
CB 4	5	200	7923.8	7917
CB 5	5	200	9148.9	9143
CB 6	10	200	7115.6	7113
CB 7	10	200	6943.8	6940
CB 8	10	200	6693.0	6690
CB 9	10	200	6658.4	6654
CB 10	10	200	6642.9	6638
CB 11	20	200	7160.3	
CB 12	20	200	6081.6	
CB 13	20	200	6147.3	
CB 14	20	200	6650.6	
CB 15	20	200	6560.0	
RG 1	5	100	646.6	644
RG 2	5	100	564.7	563
RG 3	5	100	607.7	605
RG 4	5	100	605.0	603
RG 5	5	100	557.1	554
RG 6	10	100	528.7	525
RG 7	10	100	569.7	565
RG 8	10	100	566.7	564
RG 9	10	100	603.7	600
RG 10	10	100	602.3	599
RG 11	20	100	517.9	
RG 12	20	100	553.1	
RG 13	20	100	502.8	
RG 14	20	100	536.3	
RG 15	20	100	558.2	

CB - randomly generated with all positive coefficients.

RG - randomly generated with about 1/3 negative coefficients.

TABLE III

TEST PROBLEM INFORMATION

Problem	Size		L.P.	I.P.
	M	N	Optimum	Optimum
BM 19	25	20	-31.1	-47
BM 20	27	20	-33.9	-47
BM 21	20	23	-16.6	-35
BM 22	20	25	-19.3	-33
BM 23	20	27	-20.6	-34
BM 24	20	28	-25.8	-38
BM 25	20	30	-29.0	-43
LS B	28	35	-521.1	-550
LS C	12	44	-56.6	-73
LS D_2	37	74	638.6	540
LS E	28	89	-694.2	-1120

BM - Bouvier and Messoumian [1]

LS - Lemke and Spielberg [18]

TABLE IV

PIVOT AND COMPLEMENT PROCEDURE

WITHOUT TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Percent P and C Time[4]
PET 4	5920	6090	9	.12	33
PET 5	11140	11950	2	.18	42
PET 6	10479	10584	6	.31	50
PET 7	16235	16499	12	.66	21
ST A	7734	7761	22	1.42	41
ST B	8675	8722 +	25	1.60	32
JS 1	3299	3320 +	60	1.33	18
JS 2	3343	3359	70	1.39	15
JS 3	2702	2709	21	1.69	19
JS 4	3433	3462	76	.75	24
JS 5	3965	3976	65	1.22	20
JS 6	3445	3476	55	.96	25
JS 7	4439	4519	53	1.08	26
JS 8	3333	3366 +	86	1.15	28
JS 9	3914	3942	64	.82	27
JS 10	3800	3845	45	.96	30
JS 11	3438	3466	7	3.27	24
JS 12	3029	3106	30	2.89	20
JS 13	3505	3592	3	3.49	30
JS 14	3708	3713	54	2.33	13
JS 15	3313	3401	32	1.92	25
JS 16	3084	3109	24	2.94	14
JS 17	2908	2929	50	1.64	16
JS 18	3152	3191	38	2.54	19
JS 19	2910	3005	11	2.39	31
JS 20	3143	3250 +	71	1.87	22

\+ Optimal 0-1 solution

1 Value of the first 0-1 solution found

2 Value of the last 0-1 solution found

3 Includes some output time, but no input time

4 The remainder of the time was spent in solving the LP. For instance, the first number, 33, means that 67% of the total time was spent on solving the LP, and 33% on all the rest.

TABLE V

PIVOT AND COMPLEMENT PROCEDURE

WITHOUT TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Percent P and C Time[4]
CB 1	7698	7717	161	4.20	18
CB 2	6937	6948	123	4.17	18
CB 3	7153	7166	144	3.64	20
CB 4	7830	7895	28	5.06	30
CB 5	9107	9139	114	4.06	15
CB 6	7064	7105	102	7.85	12
CB 7	6889	6926	57	10.24	26
CB 8	6612	6677	60	9.44	18
CB 9	6540	6638	56	7.46	23
CB 10	6571	6638 +	149	8.71	11
CB 11	7100	7144	65	15.23	8
CB 12	6042	6063	26	22.94	10
CB 13	6009	6131	22	29.97	12
CB 14	6574	6633	41	21.43	10
CB 15	6507	6540	37	21.26	11
RG 1	638	644 +	82	1.15	33
RG 2	558	563 +	87	1.13	25
RG 3	605 +	605 +	81	1.17	18
RG 4	596	602	73	1.24	31
RG 5	551	553	50	1.39	22
RG 6	525 +	525 +	51	2.71	16
RG 7	565 +	565 +	48	2.43	21
RG 8	555	561	47	2.35	24
RG 9	591	595	19	1.88	32
RG 10	591	594	22	2.58	32
RG 11	510	510	10	5.73	22
RG 12	537	540	2	6.46	26
RG 13	477	483	0	7.00	28
RG 14	518	521	0	7.44	13
RG 15	545	548	6	4.33	26

+ Optimal 0-1 solution
1, 2, 3, 4 See Table IV

TABLE V

PIVOT AND COMPLEMENT PROCEDURE

WITHOUT TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Percent P and C Time[4]
BM 19	-47 +	-47 +	6	.91	64
BM 20	-47 +	-47 +	5	1.01	59
BM 21	None			.70	47
BM 22	-43	-41	0	.74	62
BM 23	-44	-42	0	.61	73
BM 24	-38 +	-38 +	1	1.09	48
BM 25	-44	-43 +	0	.97	42
LS B	-550 +	-550 +	10	.56	56
LS C	-81	-73 +	1	.74	83
LS D_2	538	540 +	3	8.54	81
LS E	-2030	-1153	3	6.58	73

+ Optimal 0-1 solution

1, 2, 3, 4 See Table IV

TABLE VII

PIVOT AND COMPLEMENT PROCEDURE

WITH TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Present P and C Time[4]
PET 4	5920	6120 +	12	.13	38
PET 5	11140	12400 +	15	.18	41
PET 6	10479	10588	6	.50	69
PET 7	16235	16499	12	.82	38
ST A	7734	7772 +	24	1.88	57
ST B	8675	8722 +	25	1.76	39
JS 1	3299	3320 +	60	1.53	30
JS 2	3343	3359	70	1.47	21
JS 3	2702	2716	40	2.68	50
JS 4	3433	3467 +	100	.81	31
JS 5	3965	3976	65	1.35	29
JS 6	3445	3478	63	1.22	42
JS 7	4439	4519	53	1.43	45
JS 8	3333	3366 +	86	1.14	28
JS 9	3914	3945 +	79	.91	39
JS 10	3800	3845	45	1.54	57
JS 11	3438	3495	38	4.86	50
JS 12	3029	3106	30	4.11	44
JS 13	3505	3611	20	7.34	68
JS 14	3708	3716	63	2.66	26
JS 15	3313	3403	33	3.71	62
JS 16	3084	3109	24	4.51	45
JS 17	2908	2931	62	2.00	33
JS 18	3152	3191	38	3.35	40
JS 19	2910	3005	11	4.95	68
JS 20	3143	3250 +	71	1.92	26

+ Optimal 0-1 solution

1, 2, 3, 4 See Table IV

TABLE VIII

PIVOT AND COMPLEMENT PROCEDURE

WITH TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Percent P and C Time[4]
CB 1	7698	7717	161	4.35	22
CB 2	6937	6950	136	6.44	49
CB 3	7153	7166	144	4.18	32
CB 4	7830	7913	95	11.88	71
CB 5	9107	9142	137	5.26	36
CB 6	7064	7110	138	10.52	36
CB 7	6889	6929	69	20.87	64
CB 8	6612	6683	86	19.40	61
CB 9	6540	6638	56	18.41	70
CB 10	6571	6638 +	149	8.89	16
CB 11	7100	7149	81	27.30	49
CB 12	6042	6064	28	49.80	59
CB 13	6009	6131	22	50.21	47
CB 14	6574	6634	45	36.30	47
CB 15	6507	6540	37	37.22	48
RG 1	638	644 +	82	1.12	33
RG 2	558	563 +	87	1.11	25
RG 3	605 +	605 +	81	1.17	20
RG 4	596	602	73	1.28	34
RG 5	551	553	50	1.99	39
RG 6	525 +	525 +	51	3.53	25
RG 7	565 +	565 +	48	2.97	36
RG 8	555	561	47	2.88	40
RG 9	591	598	46	3.94	69
RG 10	591	599 +	65	2.90	42
RG 11	510	510	10	8.44	49
RG 12	537	544	8	12.21	59
RG 13	477	490	0	16.79	71
RG 14	518	524	0	11.58	49
RG 15	545	550	20	6.94	55

+ Optimal 0-1 solution

1, 2, 3, 4 See Table IV

TABLE IX

PIVOT AND COMPLEMENT PROCEDURE

WITH TRIPLE COMPLEMENTS

Problem	First 0-1 Solution[1]	Final 0-1 Solution[2]	Number of Variables Fixed	Total CPU Time[3] (seconds)	Percent P and C Time[4]
BM 19	-47 +	-47 +	6	.93	65
BM 20	-47 +	-47 +	5	1.06	61
BM 21	None			.70	47
BM 22	-43	-41	0	.88	69
BM 23	-44	-41	0	.83	80
BM 24	-38 +	-38 +	1	1.28	56
BM 25	-44	-43 +	0	1.25	56
LS B	-550 +	-550 +	10	.72	67
LS C	-81	-73 +	1	1.13	89
LS D_2	538	540 +	3	18.45	91
LS E	-2030	-1153	3	10.32	83

+ Optimal 0-1 solution

1, 2, 3, 4 See Table IV

To compare the performance of our procedure with Toyoda's primal effective gradient method [27], which is an improvement over Senju and Toyoda [26], and is held by practitioners to be an efficient heuristic for 0-1 capital budgeting problems, we coded up Method I of [27] and solved with it the 41 capital budgeting problems in the PET, ST, JS, and CB series. Toyoda's procedure starts with all variables at 0, and through a dynamic ranking procedure sequentially selects variables to be set equal to 1 as long as feasibility can be maintained.

The results obtained with Toyoda's primal effective gradient method are shown in Table X. Table XI provides a summary comparison between the two versions of the pivot and complement procedure and Toyoda's method for the problems in the ST, JS, and CB series, which are grouped by the number of constraints in the problem. The reported results are averages for the problems in the set. The % optimum refers to the integer optimum, except for the set of 5 twenty constraint problems in the CB series for which the 0-1 optimum is not presently known, and is therefore replaced by the LP optimum.

The results appearing in Table XI show that the pivot and complement procedure without triple complements provides considerably better solutions than Toyoda's method in addition to providing information contained in the LP solution. Furthermore, the values of many variables in the optimal 0-1

TABLE X

TOYODA'S PRIMAL EFFECTIVE GRADIENT METHOD

Problem	Value of Solution Found	CPU Time[1]	Problem	Value of Solution Found	CPU Time[1]
PET 4	6010	.05	CB 1	7235	3.32
PET 5	11970	.11	CB 2	6643	3.53
PET 6	9888	.15	CB 3	6813	3.38
PET 7	15897	.22	CB 4	7515	3.66
ST A	7719	.68	CB 5	8897	3.66
ST B	8709	1.08	CB 6	6571	5.45
JS 1	3212	.86	CB 7	6633	5.22
JS 2	3319	.81	CB 8	6086	5.23
JS 3	2625	.81	CB 9	6215	5.22
JS 4	3255	.86	CB 10	6010	5.30
JS 5	3903	.95	CB 11	6507	9.37
JS 6	3366	.86	CB 12	5503	8.43
JS 7	4446	.97	CB 13	5695	8.96
JS 8	3281	.78	CB 14	6024	8.85
JS 9	3749	.96	CB 15	5960	9.64
JS 10	3759	.88			
JS 11	3216	1.38			
JS 12	2857	1.37			
JS 13	3433	1.31			
JS 14	3654	1.54			
JS 15	3278	1.28			
JS 16	2863	1.32			
JS 17	2707	1.31			
JS 18	2840	1.29			
JS 19	2914	1.21			
JS 20	2914	1.21			

1 Includes some output time (less than in the case of the pivot and complement procedure), but no input time.

TABLE XI

SUMMARY COMPARISON OF THE PIVOT AND COMPLEMENT PROCEDURE WITH TOYODA'S METHOD

Problem Series	Size M	Size N	Number in Sample	Pivot and Complement without Triple Complements		Pivot and Complement with Triple Complements		Toyoda	
				Time	%Optimum	Time	%Optimum	Time	%Optimum
ST	30	60	2	1.51	99.9	1.65	100.0	.88	99.6
JS	5	100	10	1.14	99.9	1.41	99.9	.87	96.9
JS	10	100	10	2.53	99.7	3.94	99.8	1.32	93.2
CB	5	200	5	4.23	99.9	6.42	99.9	3.51	95.3
CB	10	200	5	8.74	99.9	15.60	99.9	5.28	92.5
CB	20	200	5	22.17	99.7	40.20	99.7	9.05	91.1

solution are discovered. This is purchased at the cost of about twice the CPU time required for Toyoda's method. The version that includes triple complements provides some further improvements in the quality of the final solution, but the increases in CPU time are often substantial. Note however, that these times are still very low in comparison to those required by any exact algorithm.

5. The Heuristic Used With an Exact Algorithm

Besides its use as a heuristic to find approximate solutions for large 0-1 programming problems, the pivot and complement procedure can also be used to enhance any 0-1 programming algorithm whose performance depends on the quality of 0-1 solutions found early in the procedure. In order to test the pivot and complement procedure in this capacity, a branch and bound/implicit enumeration algorithm was implemented and tested on the above described problems with and without the heuristic. In the version which uses the heuristic, the pivot and complement procedure in its entirety is applied at the start to find an initial 0-1 solution and associated lower bound, and to remove from the problem all 0-1 variables whose reduced cost exceeds the gap between the bounds; and whenever a new 0-1 solution is found, step 2 of the improvement phase is used in an attempt to improve the solution just found. Apart from this feature, the code has the usual characteristics of many branch and bound/implicit enumeration codes. It follows a rigid depth first strategy until a node is fathomed by bounds, infeasibility or integrality. Then it backtracks flexibly to the node with the best projected value (weighted sum of objective function value and integer infeasibility). Branching occurs on the variable with highest up or down

penalty, provided the current value of the variable is not within 0.1 of 0 or 1. The better of the two newly created nodes is considered first. Monotone variables, as well as variables whose reduced cost exceeds the gap between the upper and lower bounds on the objective function value, are fixed as soon as they are discovered. At each node the truncated and rounded LP solutions are tested as possible new incumbent solutions. Logical tests are used at each node on each of the initial constraints, plus the four most recently obtained surrogate constraints.

Both the basic code and the one augmented by the pivot and complement procedure were run with double precision arithmetic on the UNIVAC 1108 computer of CMU for 53 of the test problems. Tables XII and XIII compare the performance of the two codes in terms of the nodes generated for finding an optimal solution, the nodes generated for proving optimality, and the total CPU time in seconds (excluding input but including some output) required to solve the problem.

A summary comparison of the performance of the two codes is presented in Table XIV. The average time required for solving the problems in the ST, JS, CB, and RG series is tabulated, with the results grouped by the number of constraints. Problems JS 13 and CB 7 are not included in the averages since they were not solved by the basic code. The fact that the augmented code almost uniformly dominates the basic code by a factor of about 2 indicates that the pivot and complement procedure is likely to be a very cost effective addition to any branch and bound/implicit enumeration code when used to solve 0-1 programming problems.

TABLE XII

PERFORMANCE OF TWO VERSIONS OF BRANCH AND BOUND

Basic: without the heuristic
Augmented: with the heuristic

Problem	Nodes to Find Optimum		Nodes to Prove Optimality		Total Time	
	Basic	Augmented	Basic	Augmented	Basic	Augmented
PET 4	11	1	11	6	.7	.4
PET 5	5	1	12	11	1.0	.8
PET 6	13	11	22	20	1.9	2.1
PET 7	31	16	39	24	3.8	3.1
ST A	108	1	159	79	101.3	38.6
ST B	89	1	397	360	257.7	158.0
JS 1	100	1	153	55	28.1	6.6
JS 2	57	15	57	15	10.5	3.0
JS 3	31	14	39	22	8.6	6.3
JS 4	9	1	9	1	1.8	1.2
JS 5	54	29	85	51	13.6	4.9
JS 6	85	56	93	64	12.8	5.3
JS 7	99	55	105	64	15.1	7.1
JS 8	28	1	36	6	5.7	1.7
JS 9	37	1	42	15	5.9	1.8
JS 10	10	10	28	23	4.4	4.2
JS 11	206	57	287	126	94.3	36.0
JS 12	465	195	561	375	206.0	109.8
JS 13	*	734	*	779	*	248.0
JS 14	33	8	87	59	24.5	10.6
JS 15	49	18	535	505	154.3	108.3
JS 16	53	94	103	129	37.2	37.9
JS 17	4	3	4	3	2.7	3.2
JS 18	65	47	99	77	29.4	19.0
JS 19	25	115	197	258	68.0	86.7
JS 20	3	1	36	36	10.2	5.7

* Problem solution exceeded node storage limits.

TABLE XIII

PERFORMANCES OF TWO VERSIONS OF BRANCH AND BOUND

Basic: without the heuristic
Augmented: with the heuristic

Problem	Nodes to Find Optimum		Nodes to Prove Optimality		Total Time	
	Basic	Augmented	Basic	Augmented	Basic	Augmented
CB 1	258	34	260	35	77.5	8.3
CB 2	44	33	47	35	19.1	12.4
CB 3	44	40	46	42	15.2	9.3
CB 4	378	385	650	669	261.3	160.8
CB 5	285	249	292	252	110.5	36.7
CB 6	332	8	369	56	174.3	23.0
CB 7	763	407	*	555	*	251.0
CB 8	69	107	83	118	55.1	69.0
CB 9	272	176	292	236	154.0	111.8
CB 10	219	1	242	56	131.2	20.5
RG 1	42	1	60	29	11.0	2.8
RG 2	4	1	7	6	2.6	1.7
RG 3	2	1	13	9	3.8	2.0
RG 4	118	60	118	60	21.0	5.0
RG 5	156	104	498	447	94.4	48.0
RG 6	218	1	397	226	134.0	43.1
RG 7	5	1	358	411	135.1	87.9
RG 8	64	27	100	63	32.9	14.8
RG 9	68	57	168	163	53.0	39.8
RG 10	124	1	180	58	21.1	12.9
BM 19	8	1	14	13	3.3	3.3
BM 20	14	1	17	10	4.9	3.2
BM 21	188	188	258	258	48.8	49.2
BM 22	90	83	131	125	24.7	25.3
BM 23	42	29	127	108	25.8	23.5
BM 24	89	1	149	76	33.8	21.0
BM 25	196	1	588	414	130.5	102.1

* Problem solution exceeded node storage limits

TABLE XIV

SUMMARY COMPARISON OF THE TWO VERSIONS OF BRANCH AND BOUND

Basic: without the heuristic
Augmented: with the heuristic

Problem Series	Size M	Size N	Number in Sample	Average Time	
				Basic Algorithm	Augmented Algorithm
ST	30	60	2	179.5	98.3
JS	5	100	10	10.7	4.2
JS	10	100	9	69.6	49.7
CB	5	200	5	96.7	45.5
CB	10	200	4	128.7	56.1
RG	5	100	5	26.6	11.9
RG	10	100	5	85.2	39.7

References

[1] B. Bouvier and G. Messoumian, "Programmes linéaires en variables bivalentes - algorithme de Balas." University of Grenoble, France, 1965.

[2] N. Christofides, "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem." GSIA, Carnegie-Mellon University, February 1976.

[3] S. A. Cook, "The Complexity of Theorem-Proving Procedures." Proceedings of the Third ACM Symposium on the Theory of Computing, 1971, p. 151-158.

[4] G. Cornuejols, M. L. Fisher and G. L. Nemhauser, "An Analysis of Heuristics and Relaxation for the Uncapacitated Location Problem." Discussion paper 7602, CORE, Louvain, Belgium.

[5] R. E. Echols and L. Cooper, "Solution of Integer Linear Programming Problems by Direct Search." JACM, 15, 1968, p. 75-84.

[6] M. R. Garey and D. S. Johnson, "Performance Guarantees for Heuristic Algorithms: An Annotated Bibliography." J. F. Traub (editor), Algorithms and Complexity: New Directions and Recent Results. Academic Press, 1976.

[7] R. L. Graham, "Bound for Certain Multiprocessing Anomalies." Bell System Technical Journal, 45, 1966, p. 1563-1581.

[8] G. W. Graves and A. B. Whinston, "A New Approach to Discrete Mathematical Programming." Management Science, 15, 1968, p. 177-190.

[9] F. S. Hillier, "Efficient Heuristic Procedures for Integer Linear Programming with an Interior." Operations Research, 17, 1969, p. 600-637.

[10] T. Ibaraki, T. Ohashi, H. Mine, "A Heuristic Algorithm for Mixed Integer Programming Problems." Mathematical Programming Study 2, December 1974, p. 115-136.

[11] O. H. Ibarra and C. E. Kim, "Fast Approximation Algorithms for the Knapsack and Sum of Subsets Problems." JACM, 22, 1975, p. 463-468.

[12] R. G. Jeroslow and T. H. C. Smith, "Experimental Results on Hillier's Linear Search." Mathematical Programming, 9, 1975, p. 371-376.

[13] D. S. Johnson, "Approximation Algorithms for Combinatorial Problems." Journal of Computer and Systems Sciences, 9, 1974, p. 256-278.

[14] R. M. Karp, "Reducibility Among Combinatorial Problems." R. E. Miller and J. W. Thatcher (editors), Complexity of Computer Computations, Plenum Press, 1972, p. 85-104.

[15] R. M. Karp, "The Fast Approximate Solution of Hard Combinatorial Problems." Proceedings of the Sixth Southeastern Conference on Combinatorics, Graph Theory and Computing. Utilitas Mathematica Publishing Co., Winnipeg, 1975.

[16] R. M. Karp, "The Probabilistic Analysis of Some Combinatorial Search Algorithms." J. F. Traub (editor), Algorithms and Complexity: New Directions and Recent Results. Academic Press, 1976, p. 1-19.

[17] D. Korte, "Approximate Algorithms for Discrete Optimization Problems." Report No. 7762-OR, Institut für Ökonometrie und Operations Research, University of Bonn, June 1977.

[18] C. E. Lemke and K. Spielberg, "Direct Search Algorithms for Zero-One and Mixed-Integer Programming." Operations Research, 15, 1967, p. 892-914.

[19] H. Mueller-Merbach, "Heuristic Methods: Structures, Applications, Computational Experience." B. W. Cottle and J. Krarup (editors), Optimization Methods for Resource Allocation, English Universities Press, 1974, p. 401-416.

[20] C. C. Petersen, "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R and D Projects." Management Science, 13, 1967, p. 736-750.

[21] C. C. Petersen, "A Capital Budgeting Heuristic Algorithm Using Exchange Operations." AIIE Transactions, 6, 1974, p. 143-150.

[22] M. O. Rabin, "Probabilistic Algorithms." J. F. Traub (editor), Algorithms and Complexity: New Directions and Recent Results. Academic Press, 1976, p. 21-39.

[23] S. Reiter and D. B. Rice, "Discrete Optimizing Solution Procedures for Linear and Nonlinear Integer Programming Problems." Management Science, 12, 1966, p. 829-850.

[24] S. Reiter and G. Sherman, "Discrete Optimizing." SIAM Journal, 13, 1965, p. 864-899.

[25] S. Sahni, "Approximate Algorithms for the 0-1 Knapsack Problem." JACM, 22, 1975, p. 115-124.

[26] S. Senju and Y. Toyoda, "An Approach to Linear Programming with 0-1 Variables," Management Science, 15, 1968, p. B196-207.

[27] Y. Toyoda, "A Simplified Algorithm for Obtaining Approximate Solutions to 0-1 Programming Problems," Management Science, 21, 1975, p. 1417-1427.

[28] C. A. Trauth and R. E. Woolsey, "MESA: A Heuristic Integer Linear Programming Technique." Res. Rep. SC-RR-68-299, Sandia Labs, Albuquerque, 1968.

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
Technical Report No. 414		

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
Pivot and Complement - A Heuristic for 0-1 Programming	Technical Report February 1978
	6. PERFORMING ORG. REPORT NUMBER MSRR 414

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
Egon Balas Clarence H. Martin	N0014-75-C-0621 MPS73-08534 A02 NSF

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Graduate School of Industrial Administration Carnegie-Mellon University Pittsburgh, Pennsylvania 15213	NR 047-048

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
Personnel and Training Research Programs Office of Naval Research (Code 434) Arlington, Virginia 22217	February 1978
	13. NUMBER OF PAGES 38

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is 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)

heuristics, 0-1 programming, approximation techniques, pivoting, search

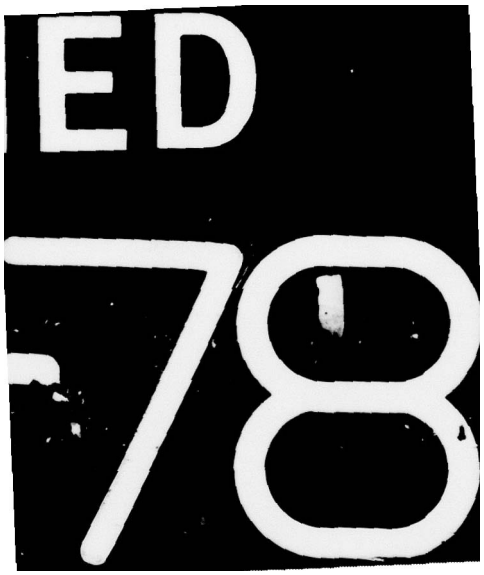
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The pivqt and complement procedure is a heuristic for finding approximate solutions to 0-1 programming problems. It uses the fact that a 0-1 program is equivalent to the associated linear program with the added requirement that all slack variables be basic. The procedure starts by solving the linear program; then performs a sequence of pivots aimed at putting all slacks into the basis at a minimal cost in dual feasibility, while taking care of occasionally arising primal infeasibilities by complementing some nonbasic

→ next page

0-1 variables; finally, it attempts to improve the 0-1 solution obtained in this way by a local search based again on complementing certain sets of 0-1 variables.

The computational effort involved in the procedure is bounded by a polynomial in the number of constraints and variables. For the 67 test problems with 20-200 variables and 5-37 constraints on which the procedure was run, the time used to solve the linear program on the average considerably exceeded the time used for everything else, and the total time never exceeded a few seconds. As to the quality of the solutions obtained, in 23 cases, or one third of the total, the procedure found an optimal solution; for all the capital budgeting problems (positive coefficients everywhere), the solution found was within 0.15% of the optimum; for 58 of the 67 problems, it was within 1% of the optimum; and only in 1 case did the procedure not find a feasible solution.


ED
78