Comparison of Four Goal Programming Algorithms

David L. Olson

The Journal of the Operational Research Society, Volume 35, Issue 4 (Apr., 1984), 347-354.

Stable URL:
http://links.jstor.org/sici?sici=0160-5682%28198404%2935%3C347%3ACOFGPA%3E2.0.CO%3B2-5

J. Opl Res. Soc. Vol. 35, No. 4, pp. 347–354, 1984
Printed in Great Britain.

0160-5682/84 $3.00 + 0.00

Comparison of Four Goal Programming Algorithms

DAVID L. OLSON
Texas A&M University

A major limitation in the use of goal programming has been the lack of an efficient algorithm for model solution. Schniederjans and Kwak recently published a proposal for more efficient solution of goal programming models utilizing dual simplex procedures. A goal programming algorithm based upon that method has been coded, as well as a revised, simplex-based algorithm. These algorithms are compared in terms of accuracy and time requirements with algorithms previously presented by Lee and by Arthur and Ravindran. Solution times for a series of 12 goal programming models are presented. The dual simplex method appears to have superior computational times for models with a large proportion of positive deviational variables in the solution. The revised simplex algorithm appears more consistent in time and accuracy for general goal programming models.

Key words: goal programming, revised simplex, dual simplex

INTRODUCTION

GOAL PROGRAMMING has been a popular theoretical method for dealing with multiple objective decision-making problems. Charnes and Cooper,[1] Lee,[2] Ignizio[3] and many others have been instrumental in the development of various forms of goal programming. Lin[4] presents an impressive list of articles which propose or apply goal programming. However, a major limitation in applying the method has been the lack of an algorithm capable of model solution in reasonable time. Hwang *et al.*[5] cited a number of limitations found in existing algorithms. Charnes and Cooper have restricted their goal programming models primarily to those utilizing a single objective priority, enabling use of efficient and readily available linear programming packages. Lee and Ignizio have presented simplex based algorithms capable of dealing with pre-emptive sets of goals. However, Hwang *et al.* (p. 28) cited these algorithms as being time-consuming, even for moderate sized problems. Dauer and Krueger[6] published a procedure iteratively utilizing linear programming codes to solve general (pre-emptive) goal programming models. Ignizio and Perlis[7] outlined the difficulties in such an iterative procedure, along with a technique for implementation. Arthur and Ravindran[8] extended this idea into a goal programming package utilizing the hierarchical structure of pre-emptive models. Hwang *et al.* cited Arthur and Ravindran's code as being an efficient means of solution.

Schniederjans and Kwak[9] recently proposed a new approach to solving general goal programming models. Lemke[10] originally presented the dual simplex algorithm for linear programming. Schniederjans and Kwak's solution procedure utilizes this method for goal programming, thus eliminating up to one half of the deviational variable columns in the simplex basis relative to Lee's full simplex approach, and does away with the $Z_j - C_j$ section of the tableau. This was cited by Schniederjans and Kwak as a source of significant savings in computational time.

This research tests these alternative goal programming algorithms on a series of 12 models. Evaluation of performance stresses model dependability as well as time requirements.

The general goal programming formulation considered for n variables, m constraints and K pre-emptive priority levels is:

$$\begin{aligned}
&\text{Min } P_1(w_{i1}^- d_{i1}^- + w_{i1}^- d_{i1}^+); \quad \text{for} \quad i = 1, \ldots, m \\
&\text{Min } P_2(w_{i2}^- d_{i2}^- + w_{i2}^- d_{i2}^+); \quad \text{for} \quad i = 1, \ldots, m \\
&\qquad \vdots \qquad\qquad\qquad \vdots \\
&\text{Min } P_k(w_{ik}^- d_{ik}^- + w_{ik}^+ d_{ik}^+) \quad \text{for} \quad i = 1, \ldots, m \\
&\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j + d_i^- - d_i^+ = b_i \quad \text{for} \quad i = 1, \ldots, m \\
&\qquad x_j, d_i^- = 1, d_i^+ \geqslant 0 \\
&\qquad P_1 \gg P_2 \gg \ldots \gg P_k .
\end{aligned}$$

In the special case of one priority level, this model is a linear programming formulation.

ALGORITHMS

This study compares four approaches to the solution of general goal programming models. The four approaches considered are Lee's[2] modified goal programming simplex, a revised simplex program developed by the author based upon Lee's routine, Arthur and Ravindran's[8] hierarchical solution procedure, and Schniederjans and Kwak's[9] dual simplex approach. The author modified a code presented by Lee (LEESGP), and programmed REVSIM (revised simplex) and DUALSIM (dual simplex). Arthur and Ravindran's code was obtained and used as was, other than an increase in dimensional capacity of the code.

Lee's modified simplex

Lee's[2] algorithm treats the full simplex tableau, expanding the evaluation section $(Z_j - C_j)$ to a row for every pre-emptive priority. The selection rules for this algorithm follow conventional primal linear programming.

Arthur and Ravindran's method

Arthur and Ravindran[8] took advantage of the hierarchical structure of pre-emptive goal programming models to decompose the problem into a series of essentially linear programming problems. This algorithm begins by considering only those constraints affecting required structural constraints and the first priority goals. Should multiple optimal solutions exist for this model, constraints affecting the next priority are added, and the new model solved. This procedure continues until a single optimal solution is found. Computational economies are gained by considering only rows and columns affecting the most important unsatisfied goal. For models with few priorities, or where all goals can be satisfied, little theoretical computational advantage is expected with this method. For goal programming models with tight constraints at high priorities, this approach should prove attractive.

Schniederjans and Kwak's method

This recently proposed approach utilizes a dual simplex procedure. Computational efficiency is gained by not maintaining the identity matrix column elements in the simplex tableau. Column labels are exchanged with row labels as the algorithm exchanges variables.

This dual simplex procedure eliminates up to one half of the deviational variable columns in the simplex tableau. Additional computational advantage can be gained by the possibility of fewer iterations. The procedure does not follow a path of guaranteed solution improvement. If the optimal solution includes a high proportion of positive deviational variables, the dual simplex method can be expected to be relatively faster than other approaches. However, if the solution contains a large proportion of negative deviational variables, this method may require extra computational effort.

The algorithm proposed by Schniederjans and Kwak[9] was found to be very fast when it identified an optimal solution. However, in models where an unsatisfied goal was in the final solution (usually the case, or else linear programming would be appropriate), cycling often occurred when the positive (non-optimal) solution basis was sacrificed in an attempt to improve satisfaction of the unsatisfied goal.

Therefore, the algorithm as presented was modified to take advantage of quick initial performance until a feasible basic solution was obtained, and reversion to conventional $Z_j - C_j$ evaluation was utilized from that point on. This approach still utilizes the efficiencies of not maintaining identity columns while gaining the systematic security of searching for the optimal solution utilizing only feasible solutions once feasibility is obtained. The variable exchange procedure as adopted is as follows.

If all right hand sides in the current basis are non-negative, go to 3.

(1) Select the pivot row by identifying the most negative right-hand side.

(2) Select the pivot column by:
   (A) eliminating variables violating structural requirements;
   (B) considering variables that do not violate any goals, and selecting the column with the greatest positive A_{ij} element in the pivot row;
   (C) if (B) is unsuccessful, picking the variable violating the lowest priority, breaking ties by the ratio of priority weight to A_{ij} element;
   (D) if (C) is also unsuccessful, selecting the next most negative right-hand side as the pivot row.

   Should (D) prove unsuccessful, the model is infeasible (mandatory constraints are mutually exclusive). Whenever a column is selected, perform the simplex operation and go back to the beginning.

(3) Select the pivot column by conventional $Z_j - C_j$ calculation.

(4) Select the pivot row by conventional r.h.s.$_i/A_{ij}$ calculation.

In either path (1–2 or 3–4), dual simple rules presented by Schniederjans and Kwak apply for calculation of new tableau elements.

Revised simplex

The revised simplex method economizes on the number of simplex columns by relying upon necessary relationships concerning simplex tableau elements. The negative deviational variable columns are fully maintained, but other variable columns are developed only as necessary. The current positive deviational variable entries in the simplex tableau are by necessity the negative of the negative deviational variable entries. Real variable columns are generated as needed for evaluation by pre-multiplying the original technological coefficient matrix A_{ij} by the current basis.

Selection rules are identical to Lee's modified simplex routine. Barring rounding error, the identical path to solution is taken. Figure 1 shows maintained simplex columns and the columns where information is generated only as required.

To demonstrate the revised simplex goal programming procedure, a very simple goal programming model is presented:

$$\text{Min } P_1 d_1^+; \quad P_2(2d_2^- + 1d_3^-)$$

$$\text{s.t. } X_1 + X_2 + d_1^- - d_1^+ = 10$$

$$X_1 + d_2^- - d_2^+ = 6$$

$$X_2 + d_3^- - d_3^+ = 5.$$

The revised simplex procedure operates with the initial identity matrix, updating other columns only as required. The objective function can be calculated for the most important unsatisfied objective level, and the contribution potential for each non-basic column can be generated without the need to compute the updated columns for other variables. A flag row is used to identify those variables which could not enter the solution at the next

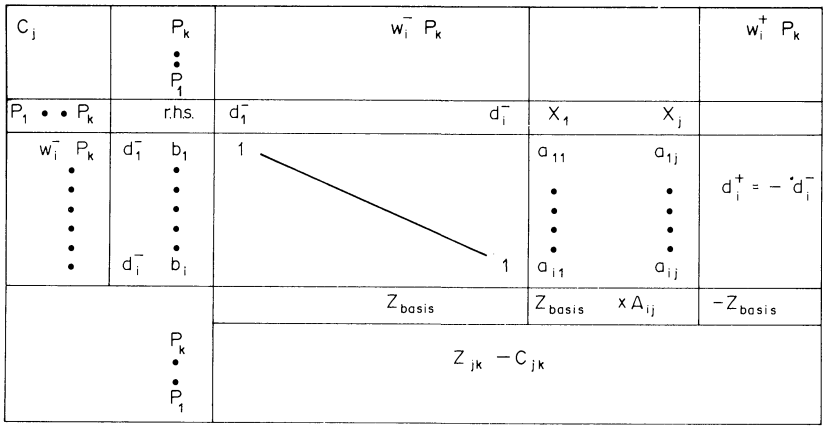

FIG. 1. *Initial tableau—revised simplex.*

iteration (either because they are already in the basis, or because entering that variable would conflict with a higher objective level). Because the positive deviational variables are mirror images of corresponding negative deviational variables, it is not necessary to compute positive deviational variable coefficients (they will always be the negative of their corresponding d_i^-). The initial tableau for the model above is shown in Figure 2.

Because the first objective (minimize d_1^+) is accomplished by the initial basis, there is no need to consider the first objective level, as long as d_1^+ is kept from entering the basis. This is assured by flagging the d_1^+ column. The second priority level is not satisfied, as both X_1 and X_2 are below their target levels. Z_j at the second objective level is calculated for the basis and pre-multiplied for all non-basic real variable columns to obtain $Z_j - C_j$. Positive deviational variable Z_j values are obtained by multiplying corresponding Z_j for each d_i^- by -1. $Z_j - C_j$ is calculated for each unflagged variable, and the entering variable in the first iteration is X_1, because it has a contribution rate of 2, larger than other columns. The second iteration updates only the identity matrix of the original tableau, substituting X_1 for d_2^- according to normal simplex procedure. The second tableau is given in Figure 3. Z_j for the second iteration is calculated for all of the d_i^- columns. This is used to generate Z_j for the lowest unsatisfied objective level for all unflagged columns. The Z_j value for d_2^- is directly obtained from this Z_j vector (0). The Z_j value for X_2 is obtained by premultiplying the Z_j vector [0, 0, 1] times the original X_2 column [1, 0, 1], yielding 1. The Z_j value for d_2^+ is the negative of the d_2^- entry in the Z_j vector (0), while the value for d_3^+ is -1. $Z_j - C_j$ for all deviational variables is calculated by subtraction. In the second tableau, only X_2 has potential for improving satisfaction of the second objective level. The

Initial tableau

C_j			P_2	–	2	1			–	–	–
			P_1	–	–	–			1	–	–
P_1	P_2		r.h.s.	d_1^-	d_2^-	d_3^-	X_1	X_2	d_1^+	d_2^+	d_3^+
–	–	d_1^-	10	1	–	–	1	1			
–	2	d_2^-	6	–	1	–	(1)	–			
–	1	d_3^-	5	–	–	1	–	1			
flag				X	X	X			X		
Z_j (P_2)				0	2	1	2	1		–2	–1
$Z_j - C_j$ (P_2)							(2)	1		–2	–1

FIG. 2. *Initial tableau—example.*

Second iteration

C_j		P_2	–	2	1			–	–	–
P_2		r.h.s.	d_1^-	d_2^-	d_3^-	X_1	X_2	d_1^+	d_2^+	d_3^+
–	d_1^-	4	1	–1	–		(1)			
–	X_1	6	–	1	–		0			
1	d_3^-	5	–	–	1		1			
flag			X		X	X		X		
Z_j (P_2)			0	0	1		1		0	–1
$Z_j - C_j$ (P_2)				–2			(1)		0	–1

FIG. 3. *Second tableau—example.*

leaving variable in the third iteration requires generating the current X_2 column. This is done by pre-multiplying the basis matrix times the original X_2 column.

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

The leaving variable is determined by dividing the current r.h.s. column by this X_2 column, and selecting the minimum non-negative ratio. Figure 4 shows the final tableau. The third iteration requires updating the current basic matrix, reflecting substitution of X_2 for d_1^+. The Z_j vector at P_2 is again calculated for this matrix. Z_j for each non-flagged variable are obtained, and C_j subtracted. The third iteration in this example yields negative $Z_j - C_j$ values for all unflagged variables, indicating no further possible improvement for the model.

TEST RESULTS

A series of nine goal programming models encountered in the author's research, plus three randomly generated models were used to compare these four approaches. The models varied widely in the number of constraints, decision variables and pre-emptive priority levels. They also varied in structure, such as the density of the initial decision variable A_{ij} matrix, as well as in the number of unsatisfied goals in the optimal solution. All of these factors can impact the computational time required for model solution. Model characteristics are given in Table 1.

Third iteration

C_j		P_2	–	2	1			–	–	–
P_2		r.h.s.	d_1^-	d_2^-	d_3^-	X_1	X_2	d_1^+	d_2^+	d_3^+
–	X_2	4	1	–1	–					
–	X_1	6	–	1	–					
1	d_3^-	1	–1	1	1					
flag					X	X	X	X		
Z_j (P_2)			–1	1	1				–1	–1
$Z_j - C_j$ (P_2)			–1	–1					–1	–1

FIG. 4. *Third tableau—example.*

TABLE 1. GOAL PROGRAMMING MODELS TESTED

Model	Constraints	Decision variables	Pre-emptive priorities	A_{ij} density	Comments
1	8	3	3	0.92	A blending models
2	9	3	4	0.85	Modified Model 1
3	23	4	9	0.33	A budgeting model
4	21	15	4	0.33	A budget allocation model
5	13	18	9	0.45	A capital budgeting model
6	19	9	4	0.97	A blending model
7	38	33	3	0.23	A scheduling model (sparse)
8	48	63	2	0.08	A modified L.P. model, mostly equalities
9	100	5	2	1.00	A least absolute value regression
10	20	20	10	0.99	Randomly generated
11	20	100	10	0.59	Randomly generated
12	120	100	10	0.11	Randomly generated

*A_{ij} density calculated as (number of non-zero A_{ij} entries)/(rows × columns).

Evaluation criteria

Algorithms should be dependable. It is of no use to be extremely efficient computationally if the optimal solution cannot be identified. The accuracy of solution is considered the paramount criterion.

Computer resources are expensive. Therefore, the time and core storage requirements of a program are of importance. Measures of C.P.U. time required and storage region are flexible, depending upon the computer system used. This study used an Amdahl V-8 central processing unit. Table 2 presents the storage required for the programs used in the tests. All programs were dimensioned for 150 constraints, 150 decision variables and 10 priority levels. A revised simplex linear programming package is included in the table for comparison.

Accuracy

Double precision was incorporated in the codes. Inaccuracies can upset algorithm operation in models requiring a large number of iterations. The Lee algorithm proved accurate in models tested, although extra iterations were required in some models. The Arthur and Ravindran code was not tampered with other than to increase dimensions owing to unfamiliarity with the specific program. Accuracy was found to be a problem for this code in larger models tested. This confirms the findings reported by Ignizio and Perlis[7] that codes not utilizing advanced techniques are limited to models involving fewer than 50 iterations.

Time

Time required for model solution is given in Table 3. Iterations are given in parentheses. The Arthur and Ravindran code was competitive in time for those models where it obtained the correct solution. For larger models run, incorrect solutions were obtained. This may be a function of the code. The original code used, written for a C.D.C. system, was highly inaccurate, possibly owing to word length differences. Dr Arthur was kind enough to provide a code written for I.B.M. systems, which was used in these tests.

LEESGP should be slower than the revised simplex code, because they share common operation, except for the maintained arrays. The primary theoretical interest of the study

TABLE 2. PROGRAM STORAGE REQUIREMENTS

Algorithm	Code	Required
Lee's G.P.	LEESGP	684K*
Revised simplex G.P.	REVSIM	472K
Arthur and Ravindran's G.P.	PARTGP	585K
Schniederjans and Kwak's G.P.	DUALSIM	480K
Revised simplex L.P.	LPREVSIM	420K

*Program capacities for the goal programming models are 150 constraints, 150 variables and 10 pre-emptive priority levels. The linear programming code (150 constraints and 150 variables) is shown for comparison.

TABLE 3. TEST RESULTS

Model	LEESGP sec (iter)	REVSIM sec (iter)	PARTGP sec	DUALSIM sec (iter)	in solution d^+	d^-
1	0.28 (7)	0.22 (7)	0.28 (6)	0.16 (8)	3	2
2	0.32 (7)	0.22 (7)	0.26 (6)	0.16 (9)	3	3
3	1.96 (20)	0.40 (19)	0.32 (8)	0.23 (17)	10	7
4	1.10 (23)	0.42 (21)	0.40 (21)	0.24 (18)	5	1
5	0.81 (25)	0.35 (25)	0.37 (15)	0.47 (29)	3	4
6	0.83 (25)	0.32 (25)	0.48 (27)	0.37 (26)	5	9
7	1.84* (31)	1.06 (29)	0.95 (31)	1.66 (93)	2	16
8	6.85* (67)	4.36 (72)	†	1.13 (69)	0	3
9	48.42* (113)	12.95 (72)	Not run	4.15 (85)	47	48
10	2.77 (49)	0.93 (49)	0.70 (43)	1.58 (69)	6	6
11	9.95 (50)	1.84 (50)	†	8.70 (71)	7	7
12	Not run	26.61 (150)	Not run	95.96 (312)	10	13

*Rounding error.
†Solution not optimal.
Times in C.P.U. seconds, Amdahl V8 (simplex iterations in parentheses).

is a comparison of the revised simplex and dual simplex codes. Resulting times vary, reflecting in part the different number of iterations required. In Model 9, the dual simplex code proved impressively faster. Model 9 was a least absolute value regression, with 47 positive deviational variables of 100 in the optimal solution. The dual simplex code took less than one third the time required by the revised simplex code. Model 8 included 40 structural equalities, requiring both revised simplex and dual simplex codes to undergo a number of required iterations. The dual simplex code proved much faster for that model. Model 7, on the other hand, had only two positive deviational variables in the final solution, yielding a much longer path for the dual simplex code.

The most significant test results finding the dual simplex code to be unfavourable were the randomly generated models (Models 10–12). The A_{ij} matrix for Model 10 was filled with randomly generated, two-digit numbers. Right-hand sides were based upon expected values of the constraints, with initial priority levels easy to satisfy, while later priorities were successively more difficult. The full 20 by 20 matrix was filled in Model 10. Model 11 added entries for 80 additional variables in the first 10 constraints, with right-hand sides adjusted accordingly. Model 12 added 100 constraints, limiting each of the 100 decision variables to 1.

CONCLUSION

The computational efficiency of the dual simplex code presented by Schniederjans and Kwak over the revised simplex code can be substantial for models involving solutions with a high proportion of positive deviational variables in the solution. If this is expected before solution (applications such as least absolute value regression would be one example), the dual simplex approach is attractive.

Both revised simplex and dual simplex appear to have computational advantages over Lee's full simplex code and Arthur and Ravindran's partitioning code. Both the Lee code and the Arthur and Ravindran code seem to lose accuracy for models involving over 50 iterations. Tests on randomly generated models indicate that the revised simplex code is a steadier procedure.

The limited tests conducted in this study merit replication before solid conclusions can be made. However, it is hoped that identification of relatively efficient goal programming codes as well as model features favouring one procedure over others can further the use of multiple objective optimization techniques.

REFERENCES

[1] A. CHARNES, W. W. COOPER and R. O. FERGUSON (1955) Optimal estimation of executive compensation by linear programming. *Mgmt Sci.* **1**, 138–151.
[2] S. M. LEE (1972) *Goal Programming for Decision Analysis.* Auerbach, Philadelphia.
[3] J. P. IGNIZIO (1976) *Goal Programming and Extensions.* D. C. Heath, Lexington, Massachusetts.
[4] W. T. LIN (1980) A survey of goal programming applications. *Omega* **8**, 115–117.

[5] C. L. HWANG, S. R. PAIDY, K. YOON and A. S. M. MASUD (1980) Mathematical programming with multiple objectives: a tutorial. *Comput. Opns Res.* **7,** 5–31.
[6] J. P. DAUER and R. J. KRUEGER (1977) An iterative approach to goal programming. *Opl Res. Q.* **28,** 671–681.
[7] J. P. IGNIZIO and J. H. PERLIS (1979) Sequential linear goal programming: implementation via MPSX. *Comput. Opns Res.* **6,** 141–145.
[8] J. L. ARTHUR and A. RAVINDRAN (1978) An efficient goal programming algorithm using constraint partitioning and variable elimination. *Mgmt Sci.* **24,** 867–868.
[9] M. J. SCHNIEDERJANS and N. K. KWAK (1982) An alternative solution method for goal programming problems: a tutorial. *J. Opl Res. Soc.* **33,** 247–251.
[10] C. E. LEMKE (1954) The dual method of solving the linear programming problem. *Naval Res. Logist. Q.* **1,** 36.