A Revised Simplex Algorithm for Large-Scale Goal Programming Models

Kum-Sik Kang*

The modern decision environment is characterized by the presence of multiple objectives or goals, which are generally competing or conflicting. Therfore, it has been well established that virtually every decision making problem involves several key criteria. Mathematical programming techniques based on a single objective criterion such as cost minimization or profit maximization are restricted in application to real-world problems. Radford(14) contended that the goal of a global optimum solution should be discarded when considering complex and multiple objective decision problems. Under the modern complex decision environment, it is necessary to simultaneously consider all of the multiple and often conflicting objectives appropriately in selecting a best strategy or policy.

Among the various techniques which have been developed to handle multicriteria decision making problems, goal programming is perhaps the most promising approach as it is an appropriate, powerful, flexible, and pragmatic tool. This technique was originally introduced by Charnes and Cooper (2, 3), and further developed by Ijiri (8) and Lee (11).

The goal programming model can be solved through the use of a computer program based on an iterative algorithm. Currently, the most widely used computer program is Lee's program (11) written in Fortran. Another popular algorithm was developed by Ignizio (7). Lee's and Ignizio's programs were designed using the modified simplex method. Lee's and Ignizio's programs did not consider efficiency in terms of the running time and storage requirements. Computational inefficiency results from unnecessary information being computed from iteration to iteration.

Recently Arthur (1) attempted to design a more efficient algorithm. This algorithm was tested in comparison with Lee's algorithm in terms of computational time. Arthur's goal partitioning algorithm is more efficient than the other two algorithms because it reduces the number of computations by modifying the matrix size when the number of subproblems

* Associate Professor, Ajou University

increases and by eliminating unnecessary nonbasic variables. The critical disadvantage of this algorithm is its inability to provide the final optimal simplex tableau required to perform sensitivity analysis.

This paper presents a new efficient GP algorithm based on the product represenation of the revised simplex method in order to overcome deficiencies of computational inefficiency on the part of the Lee and Ignizio algorithms and the lack of the final simplex tableau on the part of the Arthur algorithm.

The Revised Simplex Method with Product Form

In the regular simplex method, the entire tableau of coefficients is recalculated and stored at each iteration. The only information needed for the successive iterations is the relative costs (Z_j-C_j) for the nonbasic variables, the coefficients of the incoming variables (pivot column) and the current right-hand side values of the equations. The revised simplex method is designed to compute and store only the information needed at the current iteration in a compact form. The revised simplex method employs the original data and the inverse of the courrent basis (B^{*-1}). The revised simplex method can reduce the bulk of the computational efforts by calculating and storing only necessary numbers at the current iteration.

The revised simplex method, originally developed by Dantzig and Orchard-Hays (5) has a number of advantages over the regular simplex method (4, 6, 10, 12, 13). They can be summarized as follows:

(1) The total amount of computation in the revised simplex method may be a great deal less than that required by the regular simplex method. In particular, computations at each iteration will be considerably less when the constraint marix is sparse, and/or the ratio of the number of variables over the number of constraints is relatively large.

(2) The machine round-off errors can be more controllable. Since only B^{*-1} changes at successive iteration, and more importantly, commercial codes periodically obtain direct inversion of the basis B^{*-1} by inverting the basis (B^*), these round-off errors can be controlled in the revised simplex method.

(3) The revised simplex method permits savings in storge because only essential data are stored in a compact form.

(4) Postoptimality analysis can be performed more conveniently, and an optimal dual program can be easily obtained by inspection.

Much attention has been given to methods for the inversion and updating of the basis matrix. Of these methods the Gauss-Jordan, or product form, of the inverse has been predominently used to maintain the sparsity of the inverse. In the product form representation of the inverse matrix, only a minimum amount of information such as the transformed column need be recorded while in the revised simplex method (with the explicit form) the full inverse matrix needs to be recorded after each iteration.

The product form is an efficient technique for storing and updating the inverse of the current basis B^{*-1}. Once element is determined, the new inverse of the current basis is obtained by premultiplying the previous B^{*-1} by an elementary tranformation matrix which is defined to be an identity matrix except for one column. The elementary matrix E is expressed as:

$$E=\begin{pmatrix} 1 & & & N_1 & & \\ & 1 & & N_2 & & \\ & & \ddots & \vdots & & \\ & & & N_r & & \\ & & & \vdots & 1 & \\ & & & N_{m+1} & & 1 \end{pmatrix} \tag{1-1}$$

where

$$N_i=\begin{cases} -\dfrac{A_{is}}{A_{rs}}, & i=1,\cdots\cdots,\ m+1,\ i\neq r \\ \dfrac{1}{A_{rs}}, & i=r \end{cases} \tag{1-2}$$

where A_{rs} is the pivot element, A_{is} includes the coefficients in the pivot column (excluding the pivot element), and r is the number of the equation containing the outgoing variable.

Let us denote the new B^{*-1} by B^{*-1} new and the B^{*-1} at the preceding iteration by B^{*-1} old. Then, the element in row i and column j of B^{*-1} new is obtained by (6):

$$(B^{*-1}\ \text{new})_{ij}=\begin{cases} (B^{*-1}\ \text{old})_{ij}-\dfrac{A_{is}}{A_{rs}}(B^{*-1}\ \text{old})_{rj}, & \text{if } i\neq r \\ \dfrac{1}{A_{rs}}\ (B^{*-1}\ \text{old})_{rj}, & \text{if } i=r \end{cases}$$

In matrix notation this is

$$B^{*-1}\ \text{new}=E\cdot B^{*-1}\ \text{old} \tag{1-3}$$

Since the initial basis is an identity matrix ($B^*=B^{*-1}=I$), the basis inverse at the t^{th} iteration can be formed by

$$B^{*-1}=E_tE_{t-1}\cdots\cdots E_2E_1B^{*-1}=E_tE_{t-1}\cdots\cdots E_2E_1 \quad (1\text{-}4)$$

The elementary matrices are stored by recording only the particular nonzero elements of the non-unit vector column and their row positions.

Since B^{*-1} is never computed explicitly, computation of the simplex multipliers and coefficients in the nonbasic varible columns is possible by using the elementary matrices.

$$\pi=C_BB^{*-1}=C_BE_tE_{t-1}\cdots\cdots E_1 \quad (1\text{-}5)$$

$$\bar{a}_j=B^{*-1}\bar{a}_j{}^*=E_tE_{t-1}\cdots\cdots E_1\bar{a}_j{}^* \quad (1\text{-}6)$$

Where C_B is the contribution rate per unit of each basic variable, $\bar{a}_j$ is the j^{th} column vector in the particular tableau, and $\bar{a}_j{}^*$ is the column vector in an original constraint matrix.

The Revised Simplex Method-Based Goal Programming Algorithm(RSMGP)

The revised simplex method-based goal programming algorithm has been developed by incorporating some efficient ideas of the revised simplex method with product form to Lee's algorithm based on the modified simplex method.

Computational Steps of the RSMGP

Step 1. Set up the initial tableau from the GP model.

As the initial solution is at the origin, all the negative deviational variables will become basic variables, and these variables will take on their corresponding initial RHS values. The preemptive factors and differential weights assigned to deviational variables in the objective function should be listed in the appropriate places. Unlike the modified simplex tableau, the RSMGP taleau requires $2\times n$ matrix, where n represents all the variable columns. For the two vectors, one is used for listing the priority number and the other is used for recording the corresponding weights. The new algorithm does not require the simplex criterion (Z_j-C_j) in order to save storage requirements. Instead, Z_j value is computed for priority being considered.

Step 2. Check for optimality.

Check if there is any underachievement of a priority being considered in the column

vector located in the left side of the solution base. If there is no underachievement, proceed to Step 6, as this priority has been achieved completely. Otherwise, proceed to Step 3.

Step 3. Determine the new incoming variable.

Once the highest priority level in the solution base is identified (K^{th} priority), an elimination procedure will be adopted if one of the following conditions is met in order to determine the new incoming variable.

(1) a column being considered is a basic variable column.

(2) the priority assigned to the nonbasic variable is higher than the priority being considered.

(3) these priorities are equivalent but the weight assigned to the nonbasic variable is equal to or higher than the largest weight associated with the priority being considered in the solution base.

The computational procedure for the coefficients of the nonbasic variable is possible through the use of (1-6).

The variable in the column that has the largest nonnegative Z_j value for K^{th} priority will be selected as a basic variable at the next iteration. The pivot column is designated as j'. If there is a tie between the columns, selection can be made on an arbitrary basis. However, there is an exception in the selection rule of the pivot column. If a nonbasic variable has a positive Z_j value and also is not associated with any priority level, an additional check should be made in order to determine if there is a negative Z_j value, at a higher priority level in the same column. If there is a negative Z_j value, then this nonbasic variable column can not be a candidate for a key column.

Step 4. Determine the outgoing variable from the solution base.

The minimum ratio rule is applied to all the constraints to determine the pivot row. The basic variable in the pivot row which has the minimum nonnegative value will be the outgoing variable. A tie in the selection of the pivot row may be broken by accepting a row that has the variable associated with the higher priority level. Designate the pivot row as i'. Next compute a transformed column through the use of (1-2).

Step 5. Determine the solution.

The new basic feasible solution is obtained as:

1. The new RHS corresponding to the pivot row:

$b_i'_{new}=b_i'_{old}/a_i'_{j'}$, where $a_i'_{j'}$ is pivot element

2. the RHS for all other rows:

$$b_{inew}=b_{iold}-(a_{ij}'*b_i'_{new})$$

Return to Step 2.

Step 6. Examine next lower priority level. Set $k=k+1$. If k exceeds the total number of priority levels the optimal solution is reached. Otherwise, return to Step 2.

Numerical Example of the RSMGP

Given the linear goal programming model:

$$\text{Min}Z=p_1(d_1^-+d_1^+)+p_2(d_2^++d_3^-)+p_3d_4^++p_4d_4^-$$

S.t.

$$5X_1+3X_2+d_1^--d_1^+=250$$
$$X_1+d_2^--d_2^+=60$$
$$X_2+d_3^--d_3^+=30$$
$$5X_1+3X_2+d_4^--d_4^+=200$$
$$X_1, X_2, d_1^-, d_2^-, d_3^-, d_4^-, d_1^+, d_2^+, d_3^+, d_4^+\geq 0$$

Iteration 1.

Step 1. Initialization.

The initial simplex tableau can be set up as Table 1-1. In Table 1-1, numbers in the 'P' vectors indicate the priority levels assigned to the deviational variables and numbers in the 'W' vectors indicate their corresponding weights.

Table 1-1. Initial Simplex Tableau

			P	1		2	4	1	2		3		
			W	1		1	1	1	1		1		
P	W	V	RHS	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_2^+	d_3^+	d_4^+	X_1	X_2
1	1	d_1^-	250	1				−1				5	3
		d_2^-	60		1				−1			1	
2	1	d_3^-	30			1				−1			1
4	1	d_4^-	200				1				−1	5	3

Step 2. Check for optimality.

Since the highest priority (P_1) is in the 'P' column vector, proceed to Step 3.

Step 3. Determination of the incoming variable.

Since this is the first iteration, Z_j values for P_1 for all the nonbasic variables are

obtained directly from the original model:

$$(d_2^+, d_3^+, d_4^+, X_1, X_2) = (0, 0, 0, 5, 3)$$

It should be noted here that d_1^+ cannot be a pivot column as priority level and weight assigned to it are equal to those assigned to the priority being considered (P_1 at this iteration). X_1 is the incoming variable as it has the largest Z_j value.

Step 4. Determination of the outgoing variable.

Application of the minimum-ratio rule indicates that d_4^- is the outgoing variable. The first transformed column (eta vector) is obtained using (1-2) as follows:

$$\begin{bmatrix} -5/5 = -1 \\ -1/5 = -1/5 \\ 0 = 0 \\ 1/5 = 1/5 \end{bmatrix}$$

Consequently, the first elementary matrix looks like:

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1/5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/5 \end{bmatrix}$$

It should be noted that only column 4 is different from the identity matrix as d_4^- (4th column of the basis) is the outgoing variable.

Step 5. Computation of the new basic feasible solution.

The basic feasible solution is computed as:

$$\begin{bmatrix} d_1^- \\ d_2^- \\ d_3^- \\ X_1 \end{bmatrix} = \begin{bmatrix} 250 - 40*5 \\ 60 - 40*1 \\ 30 - 40*0 \\ 200 \div 5 \end{bmatrix} = \begin{bmatrix} 50 \\ 20 \\ 30 \\ 40 \end{bmatrix}$$

Iteration 2.

Step 2.

Since P_1 is underachieved, proceed to Step 3.

Step 3.

The nonbasic variables which can be candidates for a pivot column are:

$$(d_4^-,\ d_2^+,\ d_3^+,\ d_4^+,\ X_2)$$

Using the formulae (1-6), the coefficients in the d_4^- column can be computed as follows:

$$d_4^-: \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1/5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1/5 \\ 0 \\ 1/5 \end{bmatrix}$$

Employing the same procedure we have:

$$d_2^+: \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \end{bmatrix}, \quad d_3^+: \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}, \quad d_4^+: \begin{bmatrix} 1 \\ 1/5 \\ 0 \\ -1/5 \end{bmatrix}, \quad \text{and } X_2: \begin{bmatrix} 0 \\ -3/5 \\ 1 \\ 3/5 \end{bmatrix},$$

Consequently, the new tableau containing only necessary information is shown in Table 1-2. From this table it is obvious that d_4^+ is the incoming variable as it has the largest Z_j value for P_1.

Table 1-2. Simplex Tableau Necessary for Interation 2

			P				4		2		3		
			W				1		1		1		
P	W	V	RHS	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_2^+	d_3^+	d_4^+	X_1	X_2
1	1	d_1^-	50				−1		0	0	1		0
		d_2^-	20				−1/5		−1	0	1/5		−3/5
2	1	d_3^-	30				0		0	−1	0		1
		X_1	40				1/5		0	0	−1/5		3/5

Step 4.

d_1^- is the outgoing variable since it has the minimum ratio. The second transformed column would be:

$$\begin{bmatrix} 1/1 = 1 \\ -1/5 = -1/5 \\ 0 = 0 \\ -(-1/5) = 1/5 \end{bmatrix}$$

The second elementary matrix will be:

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/5 & 0 & 0 & 1 \end{bmatrix}$$

Step 5.

The basic feasible solutions are computed as follows:

$$\begin{bmatrix} d_4^+ \\ d_2^- \\ 1_3^- \\ X_1 \end{bmatrix} = \begin{bmatrix} 50 \div 1 \\ 20 - 50*1/5 \\ 30 - 50*0 \\ 40 - 50*(-1/5) \end{bmatrix} = \begin{bmatrix} 50 \\ 10 \\ 30 \\ 50 \end{bmatrix}$$

Iteration 3.

Step 2.

Now P_1 has been completely achieved, proceed to Step 6.

Step 6.

$k \times k+1=2$. As k (or 2) is less than the total number of priority levels, return to Step 2.

Step 2.

Since P_2 is in the 'P' column vector, proceed to Step 3.

Step 3.

Nonbasic variables which can be a candidate for a pivot column are:

$$(d_4^-,\ d_3^+,\ X_2)$$

New coefficients for these nonbasic variables can be computed as follows:

$$d_4^-: \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/5 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1/5 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1/5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/5 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$d_3^+: \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} \text{ and } X_2: \begin{bmatrix} 0 \\ -3/5 \\ 1 \\ 3/5 \end{bmatrix}$$

Table 1-3 shows the following information:

Table 1-3. Simplex Tableau Necessary for Iteration 3

			P				4						
			W				1						
P	W	V	RHS	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_2^+	d_3^+	d_4^+	X_1	X_2
3	1	d_4^+	50				−1			0			0
		d_2^-	10				0			0			−3/5
2	1	d_3^-	30				0			−1			1
		X_1	50				0			0			3/5

Step 4.

X_2 is the new incoming variable.

d_3^- will be replaced by X_2. The third 'eta' vector is:

$$\begin{bmatrix} 0 \\ 3/5 \\ 1 \\ -3/5 \end{bmatrix}$$

The third elementary matrix will be:

$$E_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 3/5 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -3/5 & 1 \end{bmatrix}$$

Step 5.

The new basic feasible solutions are computed as:

$$\begin{bmatrix} d_4^+ \\ d_2^- \\ X_2 \\ X_1 \end{bmatrix} = \begin{bmatrix} 50-30*0 \\ 10-30*(-3/5) \\ 30\div 1 \\ 50-30*(3/5) \end{bmatrix} = \begin{bmatrix} 50 \\ 28 \\ 30 \\ 32 \end{bmatrix}$$

Iteration 4.

Step 2.

As P_2 is completely achieved, proceed to Step 6.

Step 6.

$k=k+1=3$. As k (or 3) is still less than the total number of priority levels, return to Step 2.

Step 2.

Since P_3 is underachieved, proceed to Step 3.

Step 3.

Nonbasic variables to be considered are d_4^- and d_3^+. New coefficients for these variables are obtained as:

$$d_4^-: \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } d_3^+: \begin{bmatrix} 0 \\ -3/5 \\ -1 \\ 3/5 \end{bmatrix}$$

Since Z_j values for P_3 are not positive in both columns, proceed to Step 6.

Step 6.

$k=k+1=4$. As k is equal to the total priorities, return to Step 2.

Step 2.

Since P_4 is achieved completely, return to Step 6.

Step 6.

$k+k+d=5$. As k is greater than the total priorities, the current solution is optimal.

$$\begin{bmatrix} d_4^+ \\ d_2^- \\ X_2 \\ X_1 \end{bmatrix} = \begin{bmatrix} 50 \\ 28 \\ 30 \\ 32 \end{bmatrix}$$

Only P_3 is underachieved by 50. Finally, the optimal simplex tableau (Table 1-4) can be obtained, using the procedure employed:

Table 1-4. Optimal Simplex Tableau

			P	1		2	4	1	2		3		
			W	1		1	1	1	1		1		
P	W	V	RHS	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_2^+	d_3^+	d_4^+	X_1	X_2
3	1	d_4^-	50	1	0	0	−1	−1	0	0	1	0	0
		d_2^-	28	−1/5	1	3/5	0	1/5	−1	−3/5	0	0	0
		X_2	30	0	0	1	0	0	0	−1	0	0	1
		X_1	32	1/5	0	−3/5	0	−1/5	0	3/5	0	1	0

Computational Results

RSMGP is written in Fortran language. In order to test the efficiency of this algorithm, ten different models of various sizes, structures, aud complexities have been selected from journal articles and a text (actual mcdel formulations are presented in (9)). The tests were made using a IBM VM 370 at the University of Nebraska-Lincoln.

Table 1-5 indicates the number of rows (NROW), the number of system constraints, the number of decision variables (NVAR), the number of priorities (NPRT), sparsity rate of each model, and the number of iterations required to reach the optimal solution in Lee's GP and RSMGP. Sparsity rate (1) is obtained by including the deviational variables while sparsity rate (2) does not.

The computational results are presented in Table 1-6. The columns 2, 3, and 4 in Table 1-6 refer to both time (in seconds) required to read input data and the execution time for Lee's modified simplex algorithm, Arthur's goal partitioning algorithm. and RSMGP, respectively. The last two columns show the relative efficiency of the RSMGP over Lee's and Arthur's algorithms, respectively. Based on the computational results, we can make the following observations.

1. RSMGP indicates a great superiority over Lee's method for all ten models tested. RSMGP compares favorable with Arthur's method for small models. However, RSMGP shows even a greater efficiency over Arthur's method for large problems.

2. Computational time of RSMGP and Lee's GP depends greatly on structure of the objective function. Total number of iterations depends on the number of negative deviational variables associated with preemptive priority factors in the objective function. For example, although model 5 has fewer constraints and an higher sparsity rate (obtained by

Table 1-5. General Information on Models

Model	System				Sqarsity	Sparsity	Iterations
	NROW	Constraints	NVAR	NPRT	Rate (1)	Rate (2)	
1	13	0	8	7	96.88%	69.2%	9
2	30	5	17	9	93.38	82.0	24
3	29	0	29	7	95.46	93.2	29
4	33	0	12	9	98.66	42.9	17
5	29	0	12	9	92.81	74.6	42
6	56	6	21	6	94.52	70.6	44
7	35	0	48	6	97.82	76.6	63
8	65	57	60	6	96.78	94.7	67
9	41	33	102	6	93.65	91.8	82
10	107	99	102	6	97.92	96.6	106

NOTE: 1) Sparsity Rate (1) = 1 − [(total coefficients) ÷ (number of rows × number of columns)]
2) Sparsity Rate (2) = 1 − [(total technological coefficients) ÷ (NROW × NVAR)]
3) Iterations required to reach an optimal solution in Lee's GP and RSMGP

considering only decision variables) than Model 4, it took much more computation time than Model 4, as it has a total of 25 negative deviational variables associated with priorities in the objective function while Model 4 has only 16 such negative variables. The total number of iterations required to reach the optimal solution in Model 5 is 42 in RSMGP and Lee's GP while Model 4 required 17 iterations.

3. The number of decision variables plays a more important role in computational time rather than the number of constraints in RSMGP. Model 7 has 48 rows while Model 6 has 21 rows, and Model 9 has 102 rows while Model 8 has 60 rows. However, Models 7 and 9 have more decision variables than Models 6 and 8. Therefore, Models 7 and 9 took 21 and 38 seconds, respectively, while Models 6 and 8 only 8 seconds. The number of decision variables affects the computational time since RSMGP computes Z_j values for all nonbasic variables whenever each priority is considered.

4. The sparsity rate of the model is closely related to computational time in RSMGP, For example. Model 8 has more constraints, more decision variables, and more negative deviaitonal variables in the objective function than Model 7. Model 10 has more constraints than Model 9. Models 8 and 10 have higher sparsity rates than Models 7 and 9, thus reducing computational time significantly.

5. In Arthur's algorithm, the number of constraints and higher rates of sparsity than

Models 4, 6, and 10, respectively. Conseqently, Models 5, 7, and 9 took 2, 5, and 61 seconds, respectively, while Models 4, 6, 10 took 5, 10, and 92 seconds respectively.

6. RSMGP has proven its efficiency over Arthur's goal partitioning algorithm for Models 8, 9, and 10 as these models include 57, 33, and 99 system constraints, respectively,

Table 1-6. Computational Results

Model	Lee(a)	Arthur(b)	RSMGP(c)	(c)/(a)	(c)/(b)
1	1	1	0	—	—
2	13	4	2	0.154	0.500
3	16	5	3	0.188	0.600
4	14	5	3	0.215	0.600
5	23	2	4	0.174	2.000
6	58	10	8	0.138	0.800
7	49	5	21	0.428	4.200
8	136	19	8	0.059	0.421
9	103	61	38	0.370	0.623
10	568	92	29	0.050	0.315

and have sparse matrices. In the goal partitioning algorithm the Phase I simplex method is performed on the system constraints before considering the goal constraints with the priority factor of P_1. During th Phase 1 solution procedure, unnecessary information is calculated at each iteration. Thus, the Phase 1 simplex method has disavantages similar to those of the regular simplex method. In contrast, RSMGP treats the system constraints as goal constraints by assigning the priority factor of P_1, and relegating the other priorities down by one priority level.

In addition to computer times, RSMGP has several advantages over both Lee's and Arthur's methods. RSMGP requires less storage requirements than Lee's GP. Lee's modified simplex method requires two $(k\times n)$ matrices for the simplex criterion, a $(k\times m)$ matrix for representing priorities and weights assigned to basic variables in the solution base, and an $(m\times n)$ matrix for computation of coefficients at each iteration. In contrast, RSMGP requires a $(2\times n)$ matrix for objective function input, an $(m\times 2)$ matrix for representing priorities and weights associated with basic variables, and an $(m\times n)$ matrix for inputing coefficients in the initial tableau. However, RSMGP computes coefficients for only nonbasic variables at each iteration, thus resulting in significant storage savings.

RSMGP produces more accurate solutions with respect to decision variables, coefficients,

and achievement level of each priority, because these algorithms not only use the original data and the transformed columns to update coefficients, but also employ smaller tolerances.

RSMGP has a great advantage over Arthur's algorithm as the former algorithm produces the final optimal simplex tableau required to perform sensitivity analysis while the latter does not. RSMGP also has superiority over Arthur's method with respect to accuracy, as the former algorithm employs smaller tolerance than the latter.

In summary, RSMGP has superiority over Lee's and Arthur's methods in terms of speed, accuracy, and storage savings. These advantages were generated from the use of the revised simplex method.

References

1. Arthur, J., *Contributions to the Theory and Applications of Goal Programming*, Unpublished doctoral dissertation, Purdue University, 1977.
2. Charnes, A., Cooper, W.W. and Ferguson, R., Optimal Estimation of Executive Compensation by Linear Programming, *Management Science*, 1955, 1(2), 138-151.
3. Charnes, A. and Cooper, W.W., *Management Models and Industrial Applications of Linear Programming*, New York: Wiley and Sons, 1961.
4. Cooper, L. and Steinberg, D., *Metheds and Applications of Linear Programming*, Philadelphia: Saunders Co., 1974.
5. Dantzig, G.B. and Orchard-Hays, W., Alternative Algorithm for the Revised Simplex Method. Rand RM-1268, November, 1953.
6. Hillier, F.S. and Lieberman, G.J., *Operations Research*, San Francisco: Holden-Day, 1974.
7. Ignizio, J.P., *Goal Programming and Extensions*, Lexington, Massachusetts: D.C. Heath and Co., 1976.
8. Ijiri, Y., *Management Goals and Accounting for Control*, Chicago: Rand-McNally, 1965.
9. Kang, Kum-Sik, *A Revised Simplex Algorithm for Large-Scale Goal Programming Models*, Unpublished doctoral dissertation, University of Nebraska-Lincoln, 1980.
10. Ladson, L.S., *Optimization Theory for Large Systems*, New York: McMillan, 1970.
11. Lee, S.M., *Goal Programming for Decision Analysis*, Philadelphia: Averback Publishers, 1972.
12. Loomba, N.P. and Turban, E., *Applied Programming for Management*, New York: Holt, Reinhart and Winston, Inc., 1974.
13. Phillips, D.T., Ravindran, A. and Solberg, J.J., *Operations Research: principles and practice*, New York: Wiley and Sons, 1976.
14. Radford, K.J., *Complex Decision Problems*, Virginia: Reston Publishing Co., 1977.