Journal of Natural Sciences Research www.iiste.org
ISSN 2224-3186 (Paper) ISSN 2225-0921 (Online)
Vol.5, No.7, 2015

Comparisons of Linear Goal Programming Algorithms

U.C. Orumie , D.W Ebong
Department of Mathematics/Statistics, University of Port-Harcourt, Nigeria

ABSTRACT
Lack of an efficient algorithm capable of reaching a compromised solution within a reasonable time is a major setback in the use of goal programming. Orumie and Ebong newly developed an alternative method of solving goal programming problem utilizing modified simplex procedures. This algorithm is compared in terms of accuracy and time requirements with existing algorithms by Lee and by Ignizio. Computational times for 10 goal programming models of various sizes are presented. Number of iteration per problem, total entries per problems is used as benchmark for the comparisons. The new method by Orumie and Ebong (2011) have better computational times in all the problem solution and proved the best since there is a reduction in computational time in all the problems solved.
Keywords: Goal Programming, Lee's modified simplex, Ignizio's Sequential, Orumie and Ebong method

INTRODUCTION

Goal programming (GP) technique was initially developed by Charnes and Cooper (1961) for linear system in which conflicting goals were included as constraints. Their goal programming models are restricted to only those utilizing a single objective priority. The model is thus; . $lexi\min z = p_1(d_i^- + d_i^+)\, such that \quad a_{ij}x_j + d_i^- - d_i^+ = b_i \;, (i = 1,2 \ldots m)$. Further development took place by Ijiri (1965) who defined a preemptive priority levels to handle goals in their order of importance. The objective function represents the sum of disutilities for a particular program, and it is this weighted sum which is minimized to give the optimal solution. Lee (1972) and Ignizio (1976) brought the technique into common usage as an operational research tool. Goal programming quickly rose to become the most popular technique within the field of multi-criteria decision making (MCDM). This led to large number of applications being reported in the literature from the mid-1970 onwards.

Ignizio (1978) described Lee (1972) and Ignizio (1976) approaches as a significant improvement over the sequential simplex technique, since it requires fewer pivots (in general) and eliminates the need for the construction of new constraints at each sequence. Authur and Ravindran (1978) described Lee (1972) as the widely used goal programming algorithm whereas Schniederjans and kwaks (1982) echoed that Lee (1972) and Ignizio (1976) are the most commonly used goal programming solution methods, , but that both methods require columns in the simplex tableau for positive and negative deviational variables. Both methods require separate objective function rows for each priority level, all of which add immensely to the computational time of their solution method. Ignizio (1976) developed a GP solution approach known as multiphase (linear) GP method that represents a significant improvement over the sequential simplex technique but still maintains the need for the construction of new constraints for the accomplished priority level of the deviation variables for each lower level priority model. At the first stage of this procedure, the only goals included in the LP model are the first-priority goals, and the simplex method is applied in the usual way. If the resulting optimal solution is unique, we adopt it immediately without considering any additional goals. However, if there are multiple optimal solutions with the same optimal value of *Z* (call it *Z**), move to the second stage and add the second-priority goals to the model. Apply the simplex method again, if there exist multiple optimal solutions, repeat the same process for any lower priority goals. It is an efficient method but, there is a construction of new constraint at each sequence. Authur and Ravindran (1978) presented an efficient algorithm for solving linear goal programming problems utilizing the hierarchical structure of pre-emptive models using partitioning and elimination procedures that starts 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. Hwang et al (1980) described this method as being an efficient solution method. But Schniederjans and kwaks (1982) stated that Arthur and Ravindran computational procedure are limited to problems that have priorities and /or do not have conflicting goal constraints that lose variables via the variable elimination process. They stated that conflicting goal constraints that are later added to an already optimized tableau (as described in their procedure) without an iterative adjustment run the risk of violating the original G.P. problem. Ignizio (1982) developed another procedure for solving GPP that reduced tableau element with the use of the condensed simplex tableau alongside with the concept of column dropping and reflected p -space to reduce storage. He utilized fully the positive deviational variables in the basis. But, Schniederjans and Kwak (1983) reported that, Ignizio (1982) algorithm requires more computational element manipulation than the Schniederjans and Kwak algorithm, and also fails to provide useful information that is commonly found in more popular G.P. algorithms, such as that of Lee (1972). Schniederjans and kwaks (1982) provided a new computing

procedure for goal programming problems based on Baumol's simplex method. Olson (1984) stated that the algorithm was found to be very fast when it identified an optimal solution. He however, pointed out that the procedure does not follow a path of guaranteed solution improvement. If the solution contains a large proportion of negative deviational variables, this method may require extra computational effort, whereas in models where an unsatisfied goal was in the final solution, cycling often occurred when the positive (non-optimal) solution basis was sacrificed in an attempt to improve satisfaction of the unsatisfied goal. But According to Ignizio (1983), the fact that there is no evidence that the Baumol method for Linear Programming is more efficient than the usual forms of the simplex method disqualifies their claim. Olson (1984) compared four goal programming methods (Schniederjans and kwaks (1982), Lee (1972), Authur and Ravindran (1978), and Olson (1984)). He also presented a revised simplex algorithm (RSM) for solving LGP problem that utilizes Schiederjans and Kwaks (1982) dual simplex rules applied for the calculation of new tableau elements, and took 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. In his result tests, he summarized that the dual simplex method appears to have superior computational times for models with a large proportion of positive deviational variables in the solution, whereas the revised simplex algorithm appears more consistent in time and accuracy for general goal programming models. He summarized that the Lee algorithm proved accurate in models tested, although extra iterations were required in some models, whereas the Arthur and Ravindran code was not tampered with other than to increase dimensions owing to unfamiliarity with the specific program. The Schniederjans and Kwak has computational efficiency over the revised simplex for models involving solutions with a high proportion of positive deviational variables in the solution as described by the author.

Ignizio (1985) developed a multidimensional dual simplex algorithm for solving GP problems (MDD). As first shown by Ignizio, a lexicographic GP has a dual similar to that of a linear programme. The main difference is that the right hand sides of the dual are multi-dimensional and lexicographically ranked. This result obviously follows from the fact that the objective rows in the primal are lexicographically ranked (Tamiz et al (1995)). However, having noted that the multi-dimensional dual is a linear programming problem with multiple and prioritized right-hand sides, each model in the series is identical to the previous, with the exceptions that: the right-hand side changes and certain of the constraints will be dropped dependent upon the solution obtained to the previous linear programmes; all non-binding dual constraints correspond to primal variables that are: (a) non-basic in the optimal solution for the k^{th} primal achievement vector (b) associated with negative shadow prices in the optimal tableau are removed(see Ignizio (1985). But Crowder and Sposito (1987) disqualified the algorithm. They argued that removal of non-binding constraints in the dual problem after one has obtained the optimum for the dual problem associated with priority level i, coincides with the removal of non-basic variables in the LGP primal at priority level i. This implies that pre-emptive priority structures are not maintained while solving MDD problem, otherwise incorrect solution will be obtained. They supported their argument by solving a problem as shown in Ignizio (1982) (pp. 408-410).Solving the LGP problem using SLGP yields $a_0^1 = (0,1,2), x_1^0 = 6, x_2^0 = 0$. However, if one solves the multidimensional-dual problem, changing right-hand sides and removing slack rows at each priority level as outlined by Ignizio (1985), one obtains the incorrect solution: $a_0^1 = (0,1,0), x_1^0 = 4, x_2^0 = 0$.

Shim and Chun (1991) presented used Resource Planning and Management Systems (RPMS) network approach to the goal programming (GP) problem as an alternative method. In RPMS-based GP, deviational variables are represented by resource nodes. But Ogryczak (2001) disqualifies his work. He argued that RPM defers from GP formation due to the use of negative weight and additional regularization of the min max aggregation.

Tamiz et al (1995) reviewed current literature on the branch of multi-criteria decision modeling known as GP. They summarized that out of 355 papers mentioned by Romero (1991), 226 used utilizing the concept of lexicographic GP which require the pre emptive ordering of priority levels.

Yu and Li (1996) proposed a solution algorithm for linear goal programming problems. This method starts by introducing artificial variables to the model and using the two phase method or big-m method. According to him, this method uses a smaller number of variables in computation and is efficient in solving lexicographic problem, but produces entirely different and wrong results if the problem is weighted.

Calvete and Mateo (1998), presented a lexicographic optimization of multi objective generalized network flow (LGNF) problem based on the underlying ideas of primal dual algorithms for the minimum cost generalized network flow (MGNF) problem. The algorithm is efficient in reaching optimality condition, but tedious in labeling process because of several nodes, arcs, paths which result in multiple solutions.

Baykasoglu et.al (1999) used multiple objective tabu search (TS) algorithm to solve linear GP models. The danger is that the algorithm may recycle old solutions and become trapped in a loop as indicated by the author, which implies that it does not handle all kinds of goal functions and constraints.

Kasana (2003) developed an alternative algorithm for solving LLGPP called grouping algorithm that

considers all goals and real constraints together as one group with the objective function being the sum of all the unwanted deviations, and solves a sequence of LP sub problems, each using the optimal solution of the previous sub problems. This algorithm is being dominated by the partitioning method as indicated by the author. He indicated that it is good and performs well only if a large number of goals are satisfied. In other word, if an unsatisfied goal is in the final tableau, it is inefficient. It utilized sequential method.

This study compares four approaches to the solution of general linear goal programming models. The three approaches considered are Lee (1972) modified simplex goal programming, Ignizio's sequential goal programmings and the new alternative method of solving goal programming by Orumie and Ebong (2011).

In section two, the four algorithms will be presented, followed by their comparisons in section three. Computational result is presented in section four .Section five includes summary of result and conclusion in section six.

2.The two most popular algorithm of linear goal programming with, Ignizio (1982) and the new algorithm by Orumie and Ebong(2011).

Consider the linear Goal programming model as defined in Nabendu and Manish (2012);

Find $x_i : i = 1,2,...,n$

Minimize $\sum_{i}^{m} p_i (w_i^{+} d_i^{+} + w_i^{-} d_i^{-})$ (2.0)

S.t

$$\sum_{j}^{n} a_{ij} x_j + d_i^{-} - d_i^{+} = b_i \quad (2.1)$$

where

$x \in F$ (2.2)

$x_j, d_i^{+}, d_i^{-} \geq 0$ (2.3)

and $d_i^{+} * d_i^{-} = 0$ (2.4)

for $j = 1,2,\ldots,n, \quad i = 1,2,\ldots,m$

Then the existing methods are summarized below;

(a) The Orumie and Ebong(2011)method

Consider the Goal programming model in equation (2.0-2.4) above; the table (2.1) below represents the initial tableau of the above problem. Let p_i be the i[th] priority level and p_{ir} ,i=1,2,..,L ,the row corresponding to the p_i, then;

step (1) Set i=1,

step (2)The entering variable is the variable with the highest $\{a_{ij}>0, \in p_{ir}\}$. If ∃ties $\in a_{ij}$ in p_{1r}, the variable with max {min ratio of $\frac{b_i}{c_j}$} will enter, where c_j is the column corresponding to the ties in a_{ij} .If ∃ ties $\in p_1$, variable with highest w_i will enter. If ties exist in w_i,the variable with the highest minimum ratio ($\frac{b_i}{s_i^*}$) will enter the basis. Where s_i^* is the supposed pivot column corresponding to the ties in w_i. Avoid 0 and negative a_{ij}.

Step (3) Leaving variable is the variable with the minimum ratio of ($\frac{b_i}{c_p}$)Where c_p is the Pivot column resulting from (2). If ties exist in the ratio ,the variable with the smaller right hand side leaves.

Step (4) Perform normal gauss Jordan operation to update the new tableau.

Step (5) if deviations corresponding to p_i=0 in basic, go to step (6) otherwise go to step 7. If $a_{ij} \leq 0 \ \forall j \in p_{ir}$ go to step (6), If $\exists a_{ij}>0 \in p_{ir,}$ but $a_{ij} \leq 0 \in p_{i=i+1}$,corresponding to the pivot column ,go to step (6).

Step (6) Set i=i+1,go to step (2) ,if i≤L ,otherwise go to (7).

Step (7) Optimality occurs when;

(*i*) all $a_{ij} \leq 0 \forall p_{ir}$

(*ii*) when $all\ p_i(d_i^+ + d_i^-) \in z = 0$

(*ii*) when $all\ p_i(d_i^+ + d_i^-) \in z \neq 0$, but cannot be achieved further..

(*iii*) when $\exists\ positve\ a_{ij} \in p_{i+1}\ but\ violates\ p_i$

The optimal solution is the vector of (x, d_i^+, d_i^-) in the last iteration.

Initial table of the new algorithm (TABLE 2.1)

Var in basis with p_i	X_1 X_2 … X_n $d_1^{(v)}$ $d_2^{(v)}$ . . .$d_k^{(v)}$	Solution value b_i
	a_{11} $a_{12.}$ ... a_{1n} $c_{11}^{(v)}$ $c_{11}^{(v)}$. . . $c_{1k}^{(v)}$	b_1
	a_{12} a_{22} ... a_{2n} $c_{21}^{(v)}$ $c_{22}^{(v)}$ . . .$c_{2k}^{(v)}$	b_2
	. . . . . .	.
	. . . . . .	.
	. . . . . .	.
	a_{m1} a_{m2} ... a_{mn} $c_{m1}^{(v)}$ $c_{m2}^{(v)}$. . . $_{mk}^{(v)}$	b_m

Where $c_j^{(v)}$ is the coefficient of $d_j^{(v)} = \begin{Bmatrix} 1 & if\ (v) = (-) \\ -1 & if\ (v) = (+) \end{Bmatrix}$

(b) Lee(1976) modified simplex method

The table below 2.2 represents the initial tableau of the above algorithm.

STEPS

Consider, the goal programming in equation (2.0)-(2.4) above, the steps are as follow;

1. Examine $c_j - z_j$ values in the p_1-row first. If all $c_j - z_j \geq 0$ at the highest priority levels in the same column, then the optimal solution is obtained, otherwise it is not optimal. If the target value of each goal in x_b column is zero, the solution is optimal.
2. To determine the variable to enter into the new solution mix, start examining (c_j-z_j) row of highest priority (p_1) and select the largest negative value. Otherwise, move to the next higher priority (p_2) and select the largest negative value break ties arbitrarily.
3. Apply usual procedure to calculate the minimum ratio to choose the variable to leave the current solution mix.
4. Any negative value in the (c_j-z_j) row which has negative (c_j-z_j) value under a lower priority row is ignored.

2.4 LEE MODIFIED SIMPLEX ALGORITHM OF 1972

Multiphase algorithms are more efficient as they approach the entire goal programming problem as a single model and do not require the additional "constraints" as in sequential during solution.

Consider the general Goal programming model in equation (1.10-1.130) above, the Table (2.3) below represents the initial tableau of the above algorithm.

STEPS

1. In the p_1-row examine c_j-z_j values first. If all $c_j-z_j \geq 0$ at the highest priority levels in the same column, then the optimal solution is obtained, otherwise it is not optimal. The solution is optimal if the target value of each goal in x_b column is zero.

2. To determine the variable to enter into the basis, start examining (c_j-z_j) row of highest priority (p_1) and select the highest negative value. Otherwise, move to the next higher priority (p_2) and select the highest negative value. Ties are broken arbitrarily.

3. To calculate the minimum ratio to determine the leaving variable, apply usual simplex procedure.

4. Ignore any negative value in the (c_j-z_j) row which has negative (c_j-z_j) value under a lower priority row ignored.

TABLE 2.3 INNITIAL TABLE OF LEE'S MODIFIED SIMPLEX METHOD FOR PROBLEM IN EQUATION (1.10-1.13)

			C_j	C_{11}	C_{12}	. . .	C_{1n}	C_{1n+1}	. . .	C_{1k}	
C_β	Var in basis	Solution value $b=X_b$	X_1	X_2	…	X_n	d_1^-	d_2^-	. . .	d_k^-	d_1^+ d_2^+ . . . d_k^+	MinRatio
		b_1	a_{11}	a_{12}	. . .	a_{1n}	1	0	. . .	0	-1 . . . 0	
		b_2	a_{12}	a_{22}	. . .	a_{2n}	0	1	. . .	0	0 -1 . . . 0	
		. . .	. . .									
		b_m	a_{m1}	a_{m2}		a_{mn}	0	0	1	0	0 -1	
		P_k	P_{k1}	p_{k2}	…	p_{kn}	p_{kn+1}	. . .			p_{kk}	
	c_j-z_j	.	. . .									
		P_2	P_{21}	p_{22}	_	p_{2n}	p_{2n+1}				p_{2k}	
		P_1	P_{11}	p_{12}	___	p_{1n}	p_{1n+1}	. . .			p_{mk}	

2.5 A GENERAL SEQUENTIAL GOAL PROGRAMMING ALGORITHM BY IGNIZIO (1967) AS PRESENTED IN IGNIZIO (1978)

Given the problem in equation (2.1) to (2.3), then the general, sequential goal programming algorithm is:

THE ALGORITHM

Step 1: Set k = 1 (where k is used to represent the priority level under consideration and K is the total of these).

Step 2: Establish the mathematical formulation [see equations (2.1), (2.2) and (2.3)] for priority level k only. That is, minimize $a_k = g_k(\bar{n}, \bar{p})$, subject to only the goals or constraints associated with k = 1. Such a problem is equivalent to a traditional, single-objective model.

Step 3: Solve the single-objective problem associated with priority level k via any appropriate computer code. Let the optimal solution to this problem be given as a*, where: a* is the optimal value of $g_k(\bar{n}, \bar{p})$.

Step4: Set k=k + 1. If $k \geq K$, go to Step 7.

Step 5: Establish the equivalent, single objective model for the next priority level (level k). This model is given by: minimize $a_k = g_k(\bar{n}, \bar{p})$ (1.5.1)

s.t.

$$f_t(\bar{x}) + n_t - p_t = b_t \quad (2.5.2)$$

$$g_s(\bar{n}, \bar{p}) = a_s^* \quad (2.5.3)$$

$$\bar{x}, \bar{n}, \bar{p}, \geq 0 \quad (2.5.4)$$

where:

s =1,…,K-1

t is the set of subscripts associated with those goals or constraints included in priority levels 1, 2, ..., k.

Step 6: Go to Step 3.

Step 7: The solution vector $\bar{x}^*$, associated with the last single objective model solved, is the optimal vector for the original goal programming.

2.6 IGNIZIO MODIFIED SIMPLEX ALGORITHM (1976)

Consider the general Goal programming model in equation (1.10-1.13) above, the Table (2.6) below represents the initial tableau of the above algorithm.

THE ALGORITHM

Step 1: Set k = 1 (where k is used to represent the priority level under consideration and K is the total of these). subject to all the constraints.

Step 3: Examine c_1-z_1 values in the p_1-row first. If all c_1-$z_1 \geq 0$ at the highest priority levels in the same column, then the optimal solution is obtained, otherwise it is not optimal. If the target value of each goal in x_b column is zero, the solution is optimal.

Step 4: To determine the variable to enter into the basis, start examining (c_1-z_1) row of p_1 and select the largest negative value. Break ties arbitrarily. Otherwise, go to step 7.

Step 5: Apply usual procedure to calculate the minimum ratio to choose the variable to leave the basis.

Step 6: perform Gauss Jordan's operation as usual. Let the optimal solution to this problem be given as a_1*, where: a_1* is the optimal value of p_k.
Step 7: Set k=k + 1. i.e., establish the equivalent, single objective model for the next priority level (p_k=2). If k ≥ K, go to Step 8. This model is given by: min p_{k+1} subject to all the constraints, given the p_1 value.
Step8: The solution associated with the last augmented objective solved, is the optimal vector for the original goal programming.

This algorithm requires the addition of the satisfied priority row to the exiting constraints before considering the next priority.

TABLE 2.6 INNITIAL TABLE OF IGNIZIO (1976) MODIFIED SIMPLEX METHOD FOR PROBLEM IN EQUATION (1.10-1.13)

			C_j				C_{11}	C_{12}		C_{1n}	C_{1n+1}		C_{1k}	
C_β	Var in basis	Solution value b=X_b	X_1	X_2	…	X_n	d_1^-	d_2^-		d_k^-	d_1^+	d_2^+	d_k^+	MinRatio
		b_1	a_{11}	a_{12}		a_{1n}	1	0		0	-1		0	
		b_2	a_{12}	a_{22}		a_{2n}	0	1		0	0	-1	0	
		. .	. .					.					.	
		b_m	a_{m1}	a_{m2}		a_{mn}	0	0		1	0	0	-1	
	c_j-z_j	P_k	P_{k1}	p_{k2}	…	p_{kn}	p_{kn+1}						p_{kk}	

2.7 IGNIZIO (1982) CONDENSED SIMPLEX AND COLUMN DROPPING METHOD (see Igizio (1983)).

Given a goal programming model in equation (2.1)-(2.3) above, the solution procedure is similar to that of Ignizio (1976), but just that the negative deviational variables are fully maintained as the basis variables and there are interchanges of entering and living variables, together with dropping of satisfied columns (i.e. check marks are placed above the columns which are no longer eligible for entry into the basis). The interior matrix under the positive deviation variables is always the negative of that which appears under the negative deviation variables (di¯s). Since this relationship exists for every tableau, there is no need of including the positive deviational variables columns.

THE ALGORITHM

Step 1: Set k = 1 (where k is used to represent the priority level under consideration and K is the total of these) subject to all the constraints.

Step 3: Examine c_k-z_k values in the p_k-row first. If all c_k-$z_k \leq 0$ $\forall k$, then the optimal solution is obtained, otherwise it is not optimal. If the target value of each goal in x_b column is zero, the solution is optimal, go to step 8.

Step 4: To determine the variable to enter into the new basis, start examining (c_1-z_1) row of p_1 and select the largest negative value. Break ties arbitrarily. Otherwise, go to step 7.

Step 5: Apply usual procedure to calculate the minimum ratio to choose the variable to leave the current solution mix.

Step 6: Interchange the variables and perform Gauss Jordan operation as usual except for the pivot column that falls under the positive deviation variables. This will always be the negative of that which appears under the negative deviation variables (di¯s) columns. Let the optimal solution to this problem be given as a_1*, where: a_1* is the optimal value of p_k.

Step 7: Set k=k + 1. i.e., establish the equivalent, single objective model for the next priority level (p_k=2). If k ≥ K, go to step 8. This model is given by:

min p_{k+1} (2.7.1)

subject to all the constraints, given the $p_1 = a_1$* (2.7.2).

Place check marks above the columns which are no longer eligible for entry into the basis and go to step 4.

Step8: The solution associated with the last augmented objective solved, is the optimal vector for the original goal programming.

TABLE 2.7 INNITIAL TABLE OF IGNIZIO (1982) CONDENSED SIMPLEX AND COLUMN DROPPING METHOD.

C_β	Var in basis	Solution value b=X_b	X_1 X_2 … X_n d_1^- d_2^- . . d_k^- . .
	d_1^-	b_1	a_{11} a_{12}. . . a_{1n} 1 0 . . . 0
	d_2^-	b_2	a_{12} a_{22}. . . a_{2n} 0 1 . . . 0
	. .	. .	. . . . .
	d_m^-	b_m	a_{m1} a_{m2} a_{mn} 0 0 1
	c_i-z_j	P_k	P_{k1} p_{k2} ... p_{kn} p_{kn+1} . . .

3. COMPARISON OF THE NEW ALGORITHMS WITH THE EXISTING ONES.

INTRODUCTION

Ten problems from standard published papers and texts of various sizes and complexities were solved to and also used to compare Orumie and Ebong(2011) new algorithm with already existing and most popular approaches by Lee(1972), Ignizio(1976) and Ignizio(1982). The models varied widely in the number of constraints, decision variables, deviational variables and pre-emptive priority levels and weights.

The number of iteration, rows and columns (total entries) are reasonable surrogate for the actual time. This surrogate is useful when comparing various algorithms within the same general class of algorithms for which the time per iteration can be expected to be about the same among the algorithms. The comparison is based on the number of rows per iteration, number of columns per iteration, number of deviational variables, table entries, rows, columns dispensed by the new method, number of iteration, number of table entries per problem, and percentage of the deviational variable columns, rows, and table entries, dispensed by the new method. These factors above and the result summary of the solved problems are presented in Table 4.1 below.

4.1 COMPUTATIONAL RESULT

The standard problems were solved using the Orumie and Ebong (2011) method and the most widely existing ones by Lee (1972) and Ignizio (1976), including Ignizio (1982), as summarized in table 4.1 above.

In the first problem, the maximum number of elements needed to be computed for all the iterations in the proposed method was 250. If the same problem were solved using the simplex method presented by Lee, it would require a maximum 825 elements. If the same problem were solved using Ignizio (1976) and Ignizio (1982) methods, they would require a maximum of 600 and 400 elements respectively as shown in column 5 of the same table. On the same question, the current algorithm required eleven columns and five rows per table, while Lee and Ignizios required 15 columns, 11 rows, 15 columns, 8 rows and 10 columns, 8 rows respectively. The current method eliminated 40% of the deviational variable columns on the table unlike Lee (1972) and Ignizio (1976) that utilized full deviational variables columns. But Ignizio (1982) method eliminated 50% of the deviational variable columns on the same question. The current method also dispensed 55% rows in Lee and 25% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 67%, 58%, 38% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

In problem 2, the maximum number of elements needed on the average to be computed for all the iterations in the proposed method was 84. If the same problem were solved using the simplex method presented by Lee, it would require a maximum 330 elements. If the same problem were solved using Ignizio (1976) and Ignizio (1982), it would require a maximum of 198 and 126 elements respectively. The current algorithm required 7 columns and 4 rows per table, while Lee required 11columns, 10 rows. Both Ignizio (1976) and Ignizio (1982) require 11 columns, 6 rows and 7 columns, 6 rows respectively. Both Ignizio (1982) and the current method dispensed 50% of the deviational variable columns on the table unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 60% rows in Lee and 33% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 68%, 58%, 19% entries on Lee, Ignizio(1976) and Ignizio(1982) respectively.

Also in problem 3, the maximum number of elements needed to be computed for all the iterations in the proposed method was 56. If the same problem were solved using the simplex method presented by Lee, it would require a maximum 108 elements whereas both Ignizio (1976) and Ignizio(1982) would require a maximum of 90 and 70 elements respectively to solve the same problem. The current algorithm required 7 columns and 4 rows per table, while Lee required required 9 columns, 6 rows. Both Ignizio(1976) and Ignizio(1982) would reqire 9 columns, 5 rows and 7 columns, 5 rows respectively. Both Ignizio(1982) and the current method eliminated 50% of the deviational variable columns on the table unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 33% rows in Lee and 20% rows in both,

Ignizio 1976 and 1982 respectively. It also dropped total 22%, 38%, 20% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

In problem 4, the maximum number of elements needed to be computed for all the iterations in the proposed method was 140. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 660 elements. If the same problem were solved using Ignizio (1976) and Ignizio (1982), they would require a maximum of 330 and 210 elements respectively. In the same question, the current algorithm required 7 columns and 4 rows per table, Lee required 11columns ,12 rows and both Ignizio (1976) and Ignizio (1982) would require 11columns ,6 rows and 7columns ,6 rows respectively to solve the same problem. Both Ignizio (1982) and the current method eliminated 50% of the deviational variable columns on the table unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 67% rows in Lee and 33% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 76%, 58%, 33% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

Furthermore, in problem 5, the maximum number of elements needed to be computed for all the iterations in the proposed method was 56. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 108 elements. If the same problem were solved using. If the same problem were solved using Ignizio (1976) and Ignizio (1982), it would require a maximum of 90 and 70 elements respectively. The current algorithm required 7 columns and 4 rows per table, while Lee required 9columns, 6 rows. Both Ignizio (1976) and Ignizio (1982) require 9 columns, 5 rows and 7 columns, 5 rows respectively. Both Ignizio (1982) and the current method dispensed 50% of the deviational variable columns on the table unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 33% rows in Lee and 20% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 19%, 38%, 20% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

In problem 6, the maximum number of elements needed to be computed for all the iterations in the proposed method was 135. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 363 elements, whereas if the same problem were solved using both Ignizio (1976) and Ignizio (1982), they would require a maximum of 231 and 147 elements respectively. In the same question the current algorithm required 9 columns and 5 rows per table, while Lee required 11columns. Both Ignizio (1976) and Ignizio (1982) require 11 columns, 7 rows and 7 columns, 7rows respectively. Both Ignizio(1982) and the current method dispensed 50% and 33% of the deviational variable columns on the table respectively, unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 55% rows in Lee and 28% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 63%, 42%, 8% entries on Lee, Ignizio(1976) and Ignizio (1982) respectively.

In the problem 7, the maximum number of elements needed to be computed for all the iterations in the proposed method was 72. If the same problem were solved using the simplex method presented by Lee, it would require a maximum 396 elements. If the same problem were solved using Ignizio (1976) and (1982) , they would require a maximum of 144 and 96 elements respectively. In the same question, the current algorithm required 6 columns and 3 rows per table, while Lee required 9 columns, 11 rows. Both Ignizio (1976) and Ignizio (1982) would require 9 columns ,4 rows and 6columns , 4 rows respectively to solve the same problem. Both Ignizio(1982) and the current method dispensed 50% and 33% of the deviational variable columns on the table respectively, unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 73% rows in Lee and 25% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 86%, 50%, 25% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

In problem 8, the maximum number of elements needed to be computed for all the iterations in the proposed method was 84. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 300 elements. If the same problem were solved using Ignizio's method of (1976) and (1982), they would require a maximum of 180 and 126 elements respectively. In the same question the current algorithm required 7 columns and 4 rows per table, while Lee required 10 columns, 10 rows. Both Ignizio(1976) and Ignizio(1982) require 10 columns, 6 rows and 7 columns, 6 rows respectively. Both Ignizio(1982) and the current method dispensed 50% of the deviational variable columns on the table, unlike Lee and Ignizio that utilized full deviational variables columns. The current method also dispensed 60% rows in Lee and 33% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 72%, 53%, 33% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

In problem 9, the maximum number of elements needed to be computed for all the iterations in the proposed method was 128. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 528 elements. If the same problem were solved using Ignizio (1976) and (1982), they would require a maximum of 308 and 196 elements respectively. The current algorithm required 8 columns and 4 rows per table, while Lee required 11columns, 12 rows. Ignizio (1976) and Ignizio (1982) require 11 columns, 7 rows and 7 columns, 7 rows respectively. Both Ignizio (1982) and the current method dispensed 50% and 38% of the deviational variable columns on the table, unlike Lee and Ignizio (1976) that utilized full

deviational variables columns. The current method also dispensed 67% rows in Lee and 42% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 76%, 58%, 35% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

Finally in problem10, the maximum number of elements needed to be computed for all the iterations in the proposed method gave 36. If the same problem were solved using the simplex method presented by Lee, it would require a maximum of 126 elements. If the same problem were solved using Ignizio (1976) and Ignizio (1982), they would require a maximum of 144 and 60 elements respectively. The current algorithm required 6 columns and 3 rows per table, while Lee required 9 columns, 7 rows. Both Ignizio (1976) and (1982) would require 9 columns, 5 rows and 6 columns, 5 rows respectively.

Both Ignizio (1982) and the current method dispensed 50% and 75% of the deviational variable columns on the table, unlike Lee and Ignizio (1976) that utilized full deviational variables columns. The current method also dispensed 57% rows in Lee and 57% rows in both, Ignizio 1976 and 1982 respectively. It also dropped total 71%, 75%, 43% entries on Lee, Ignizio (1976) and Ignizio (1982) respectively.

5. SUMMARY

Problems from standard published papers and texts were solved by both the proposed method and the existing methods to test their computational efficiency. The Orumie and Ebong(2011) method has computational advantages over both Lee's full simplex method and Ignizio methods although they are, all accurate in reaching optimality. In all the tested examples, all the methods have the same number of iteration. However Ignizio (1976) and Lee's utilized full deviational variable columns, unlike Ignizio (1982) that maintained only 50% deviational variable columns. Furthermore Lee (1972) and Ignizio (1976) require the same number of column per iteration as shown in column 2 of table 4.1, but different number of rows.

Column 7, 14 and 15 of table 4.1 shows that it is only Orumie and Ebong (2011) algorithm and Ignizio (1982) that dispensed some of the deviational variable columns. Average percentage of deviational variables (d) columns dropped by Orumie and Ebong (2011) method in Lee (1972) and Ignizio (1976) is 47.6% with standard deviation 12.7% as shown on column 11 of the same table. Average percentage of deviational variables (d) columns dropped by Ignizio (1982) in Lee (1972) and Ignizio (1976)= 50% with standard deviation 0.00 as shown on column 15 of table 4.1.

Column 3 of table 4.1 shows that it is only the Orumie and Ebong(2011) algorithm that dispensed the highest rows.

Average percentage row dropped by Orumie and Ebong(2011) method in Lee is 56% with standard deviation 13.43% as shown on column 11 of table 4.1. Average percentage row dropped by the current method in Ignizio is 31.6% with standard deviation 11.20 %(see also column 11). Average % row dropped by the current method in Ignizio (1982) is 31.60% with standard deviation 11.20%.

Column 5 of the same table shows that the Orumie and Ebong(2011) algorithm requires the least entries per table followed by Ignizio (1982), and Ignizio (1976). Lee has the highest entries per table.

Average percentage of entries dispensed by the Orumie and Ebong(2011) method per problem in Lee is 62% with the best case = 86.4%, worst case =18.5% and standard dev. =22.8%. Average % of entries dispensed by the Orumie and Ebong(2011) method per problem in Ignizio (1976) is 51% with the best case = 60%, worst case =37.78% and standard dev. =10%. Average percentage of entries dispensed by Orumie and Ebong(2011) method per problem in Ignizio (1982) is 27.44% with the best case = 43.33%, worst case = 8.16% and standard dev. = 10.72%.(see column 13 of table 4.1). Average percentage entries dispensed by Ignizio per problem in Lee is 36% with the best case = 64%, worst case =17% and standard dev. =15%. Average percentage entries dispensed by Ignizio (1982) per problem in Lee is 56% with the best case = 76%, worst case =33% and standard dev. =14% (see column 16 and 17 of table 4.1).

6 CONCLUSION

The Orumie and Ebong(2011) method has computational advantages over Lee's and Ignizio's methods since it yields a substantial reduction in the number of tableau elements computations in all the problems solved. It also eliminates on the average 56%, 31.6%, 31.6% rows on Lee's method, Ignizio (1976), Ignizio (1982) method respectively, and eliminates on the average 62%, 51%, 27.44% of total entries in on Lee's method, Ignizio (1976), Ignizio (1982) method respectively per problem as shown in Table 4.1 and, thus saves storage space. The Orumie and Ebong(2011) method eliminates on the average 47.5% deviational variables columns in both Lee (1972) and Ignizio (1976) method, but on the average utilize the same number of deviational variable columns with Ignizio (1982) method. Column 10 of table 4.1 above shows that Ignizi (1982) utilizes lesser deviational variable columns per problem than the Orumie and Ebong(2011) algorithm when the problem has more or all of the deviational variables in the objective function to be negative.

RECOMMENDATION

It is recommended that the Orumie and Ebong(2011) method should be used since it reduces computational time of solving a problem.

AREA FOR FURTHER RESEARCH

Further research should be carried out on the modification of the Orumie and Ebong(2011) to be able to solve other types of GPP such as integer, zero-one goal programming, and quadratic goal programming.

REFERENCES

Aouni, B. & O. Kettani, (2001), *Goal programming model*: A glorious history and promising future. European Journal Of operation of operational research 133,225-231.

Arthur J. L. & A. Ravindran, (1978), *An efficient goal programming algorithm using constraint partitioning and variable elimination*. Management Science. 24, 867-86.

Calvete. H. I. & P. M. Mateo (1998*): Lexicographic Optimisation in Generalised Network Flow Problems.* Journal of the Operational Research Society, Vol. 49, No. 5, pp. 519-529.

Charnes .A. & W. W. Cooper (1961): *Management Models and the Industrial Applications of Linear Programming,* Vols. 1 and 2. John Wiley, New York.

Cohon ,T.L(1978) Multi objective Programming, Academic press, New York.

Hannan, E. L(1985): *An assessment of criticism of goal programming*, computer and operation research 12(6),525-541.

Ignizio, J. P. (1967): Adaptive antenna array study, Boeing Company, RWA-5557.

Ignizio, J.P (1982) Linear programming in Single and Multiple Objective System. Prentice Hall,USA,page 408-410.

Ignizio J. P (1985): *An algorithm for solving the linear goal-programming problem by solving its dual*. Journal of operational Research Society. 36, 507-5 15.

Kasana ,H.S (2003):*Grouping Algorithm For Linear Goal Programming Problems*. Asia Pacific Journal Of Operational Research;20,191-220.

Lee, S. M. (1972): Goal Programming for Decision Analysis. Auer Bach, Philadelphia.

Lee, S. M. and E. R.Clayton (1972): *A goal programming model for academic resource allocation*. Management Science, Vol.18, No.8, pp.395-408.

Ogryczak, W. (2001), *Comments on Romero C, Tamiz M and Jones DF (1998); Goal Programming, Compromise Programming and Reference Point Method Formulations: Linkages and Utility Interpretations*, The Journal of the Operational Research Society, Vol. 52, No. 8, pp. 960-962.

Olson D. L. (1984): *Revised simplex method of solving linear goal programming problem.* Journal of the Operational Research Society Vol. 35, No. 4(1984) pp 347-354.

Schniederjans M. J. & N. K. Kwak (1982*): An alternative solution method for goal programming problems: a tutorial. /*. Operational Research Society. 33, 247-251.

Scniederjans M.J (1995) *Goal programming methodology and applications*. Kluwerpublishers, Boston.

Yu P.L & L. H L^1 (1996), *An efficient method of solving Linear Goal Programming Problems.* Journal of Optimization theory and applications vol.90,No. 2,pp465-467.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar