[1]Department of Biostatistics, School of Public Health and Curriculum in Operations Research and Systems Analysis.

[2]Analytic Services, Inc., Falls Church, Virginia.

THE GRADIENT LINEAR OPTIMIZATION METHOD

by

Richard H. Shachtman[1] and Eileen P. Neely[2]

Department of Biostatistics
University of North Carolina at Chapel Hill

Institute of Statistics Mimeo Series No. 1041

November 1975

THE GRADIENT LINEAR OPTIMIZATION METHOD

Richard H. Shachtman[1] and Eileen P. Neely[2]

1. Introduction

Due to the notable expense necessarily incurred in the solution of large-scale linear programs with the algorithms which are currently available, efforts to streamline the solution techniques continue. Some studies have suggested that the key to more efficient solution may be the decision rule (rules) which is (are) used to determine the pivot element for change of basis [3,4,5]. This paper presents an algorithm which is a modification of the Bounding Hyperplane Method (BHPM), [4]. In particular, a different criterion is suggested for selection of the vector which is leaving the basis.

The proposed Gradient Linear Optimization (GLO) approach is one member of the class of primal-dual algorithms for the solution of linear programs [1, Chp. 8]. In these algorithms, a decision is made at each iteration, based on the current state of the problem, whether to use a primal or a dual iteration to move to the next basic solution for the problem. The purpose of such a design is to reduce the number of iterations and time required for solution. As pointed out by Saksena and Cole [4], this type of procedure is not restricted to movement within the feasible region of the linear program. At each iteration, the algorithm can lead to a solution

---

[1]Department of Biostatistics, School of Public Health and Curriculum in Operations Research and Systems Analysis.

[2]Analytic Services, Inc., Falls Church, Virginia.

which is either primal feasible [1, chp. 1], dual feasible [1, chp. 8], both primal and dual feasible, or neither primal nor dual feasible. A primal feasible basic solution corresponds to an extreme point of the polytope generated by the constraints, while a dual feasible basic solution which is not also primal feasible does not lie on the polytope. A basic solution which is both primal and dual feasible is optimal and is an extreme point of the polytope. In some cases this increased freedom of movement may allow improved efficiency as compared to primal and/or dual algorithms. Another advantage introduced by this freedom of movement is that selection of an initial basic solution is easy since primal feasibility is not required.

2. The Linear Program

The Gradient Linear Optimization method is designed to be used for the solution of linear programs of the form:

$$\begin{aligned} &\text{Maximize} && \underline{c}'\underline{x} \\ &\text{Subject to} && A\underline{x} \leq \underline{b}, \\ & && \underline{x} \geq 0, \end{aligned}$$

where $\underline{x}$ is an $n \times 1$ vector, A is the $m \times n$ matrix of coefficients, and b is an $m \times 1$ vector. (Note that since the sign of $\underline{b}$ is unimportant, a feasible region defined by $A\underline{x} \geq \underline{b}$ is also admissible.) After the inequalities are converted to equalities by the introduction of a vector $\underline{s}$ of slack variables, where each slack variable has an associated cost of 0, the problem can be written as:

$$\begin{aligned} &\text{Maximize} && \underline{c}'\underline{x} + \underline{0}'\underline{s} \\ &\text{Subject to} && A\underline{x} + I\underline{s} = \underline{b}, \\ & && \underline{x} \geq \underline{0}, \\ & && \underline{s} \geq 0. \end{aligned}$$

At each iteration, the current solution is either primal feasible or dual feasible, or else the problem can be identified as infeasible. Assuming that the problem is not identified as being infeasible, then the dual class algorithm will be used in the event that the problem is dual feasible; otherwise, an algorithm designated as primal class[3] will be employed. In the event that a solution is both primal feasible and dual feasible, the solution is optimal and no further iterations are needed. Thus at any iteration, the choice between algorithms is clearly defined.

For purposes of discussion of the GLO method, it is necessary to define some notation. Throughout the paper, $\underline{x}$, A, $\underline{c}$, $\underline{Z}$ are used to represent the solution vector, matrix of constraint coefficients, the vector of costs, and the vector of reduced costs, respectively. (Here, using standard notation, an element of $\underline{Z}$ is defined as $Z_j \equiv z_j - c_j$. The vector $\underline{Z}$ can then be interpreted as the normal to a family of hyperplanes where each hyperplane is of the form $\sum_{j \in J} Z_j x_j = K$ for an arbitrary constant K.) In addition, let I and J be the set of indices of the rows and columns, respectively, of the matrix A. Let the subset of row indices $I_1 \subset I$ be defined by $I_1 = \{i \in I \mid x_i < 0\}$; furthermore, let the subset of column indices $J_1 \subset J$ be defined by $J_1 = \{j \in J \mid Z_j < 0\}$. Then I_1 contains the indices of constraints which are currently primal infeasible, while J_1 contains the indices of columns which are currently dual infeasible. Finally, let mod Z be the modified reduced cost hyperplane which is obtained by replacing the negative coefficients (if any) of the reduced cost hyperplane by their moduli;

---

[3]The term "primal class" will be used to designate the algorithm which is used for primal feasible problems. It should be noted that this algorithm is also used in circumstances where the problem is neither primal nor dual feasible.

i.e., mod Z is a typical hyperplane in the family of hyperplanes having the form $\sum_{j \in J_1} |Z_j| x_j = K$, where K is an arbitrary constant.

3. The Algorithm

GLO is an iterative procedure in which the first step in each iteration must be a determination of the current status of the solution. Because the primal-dual approach does not require that the solution at each iteration be primal feasible, it is always possible to take the initial basic solution directly from the problem. The solution then progresses according to the following rules:

(1) If the current solution is both primal feasible (for all $i \in I$, $x_i \geq 0$) and dual feasible (for all $j \in J$, $Z_j \geq 0$), then the current solution is optimal.

(2) If the current solution is not dual feasible, i.e., it is either primal feasible or else neither primal nor dual feasible, two cases are possible, i.e., for δ_i as defined below,

a. If $\delta_i \geq 0$ for all $i \in I$, then the primal is unbounded; Stop.

b. If there exists i such that $\delta_i < 0$, use the primal class algorithm to proceed to the next iteration.

(3) If the current solution is dual feasible but not primal feasible, again two cases are possible, i.e.,

a. If for some $i_o \in I$, $x_{i_o} < 0$ and $a_{i_o j} \geq 0$ for all $j \in J$, then the primal is infeasible. Stop.

b. If conditions (1) and (3a) are not met, but the current solution is dual feasible, proceed to the next iteration by using the dual class algorithm.

The GLO Primal Class Algorithm

If there exists $j \in J$ with $Z_j < 0$, a restricted problem is generated by removing all columns which correspond to variables which are dual feasible. In order to select the pivot row, the following value is calculated for each constraint in the reduced problem:

$$\delta_i = \left(\sum_{j \in J_1} a_{ij} Z_j \right) / \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} .$$

If θ_i represents the angle between the normal to the ith hyperplane and the normal to the reduced cost hyperplane, then

$$\cos \theta_i = \left(\sum_{j \in J_1} a_{ij} Z_j \right) / \left(\left(\sum_{j=1}^{n} a_{ij}^2 \right) \left(\sum_{j=1}^{n} Z_j^2 \right) \right)^{\frac{1}{2}} .$$

Since the second factor in the denominator is constant over i, it is clear that δ_i is proportional to $\cos \theta_i$. The exit vector is chosen by restricting consideration to the constraints for which $\delta_i < 0$ and then, from among these choosing that row for which $|\delta_i|$ is maximum. More concisely, if $T = \{i \in I \mid \delta_i < 0\}$, then the rth row is chosen as the pivot row if and only if $r \in T$ and $|\delta_r| = \max_{i \in T} |\delta_i|$. This selection results in movement to the hyperplane having the normal which forms the smallest angle with the normal to the negative reduced cost plane of the reduced problem. The enter vector is selected in the usual manner (i.e., as for the simplex algorithm) in order to maximize the increase in the objective function. (See, for example, [1, chp. 4].) Maxtrix calculations are then executed for the whole tableau.

The GLO Dual Class Algorithm

If $Z_j \geq 0$ for all $j \in J$ but there exists an $i \in I$ with $x_i < 0$, a restricted problem is generated by removing all rows corresponding to variables which are primal feasible. To select the exit vector for the dual class algorithm, we define

$$\alpha_i = \left(\sum_{j=1}^{n} a_{ij} Z_j \right) \Big/ \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} .$$

The kth vector is selected as the exit vector if and only if $|\alpha_k| = \max_{i \in I_1} |\alpha_i|$. Since α_i is proportional to $\cos \theta_i$, the effect of this decision rule is to move to the hyperplane characterized by having the normal which forms the largest angle with the normal to the reduced cost hyperplane. The enter vector is then chosen in the usual manner in order to maximize the decrease in the objective function. (See, for example, [1, chp. 8].)

4. Comparison of GLO and BHPM

The GLO procedure described above is consistent with BHPM [4] in that two classes of algorithms are defined, one of which is to be used in the event that the current solution is dual feasible, with the second algorithm to be used if the criterion of dual feasibility is not met. The two methods diverge, however, with regard to the specification of the decision rules for choosing the exit vector. In both procedures, the exit vector is selected first, whereas in the simplex algorithm the enter vector is selected first. In BHPM, the goal is to move to the nearest bounding hyperplane (as viewed from the current extreme point) in the direction of the normal to the reduced cost plane. In contrast, the GLO approach uses a positional criterion (as measured by the angle between normals) rather than a distance criterion.

5. Examples

Example 1. This example was included in [4] as Example 2 and is introduced here in order to point up the difference in the sequence of basic solutions which can occur based on the choice of decision rules. The problem is to

$$\text{Maximize} \quad 0.75x_1 - 150x_2 + 0.02x_3 - 6.0x_4$$

$$\begin{aligned} \text{Subject to} \quad & 0.25x_1 - 60x_2 - 0.04x_3 + 9.0x_4 \leq 0 \\ & 0.50x_1 - 90x_2 - 0.02x_3 + 3.0x_4 \leq 0 \\ & x_3 \leq 1 \\ & x_j \geq 0, \quad j=1,2,3,4. \end{aligned}$$

Saksena and Cole introduced this as an unpublished example of cycling (due to Beale) for the simplex method. Solution with the GLO approach requires two iterations, which is the same number required by the BHPM approach. The tableaus generated during the solution of this problem by the GLO approach are shown in Tables I-III. Comparison of the tableaus in Tables I-III of this paper with the corresponding tableaus in Tables V-VIII of [4] reveals a difference in the sequence of pivotal elements. (Note: In Example 1 as found in [4], use of the GLO method to find the optimal solution resulted in the same sequence of basic solutions as for BHPM.)

TABLE I

Initial Tableau for Example 1

	$P_j \rightarrow$	P_1	P_2	P_3	P_4	P_5	P_6	P_7		Calculations: Primal Algorithm		
$\downarrow i$	$c \rightarrow$	.75	-150	0.02	-6	0	0	0	$\underline{x}\downarrow$	δ_i	ANGLE	REMARKS
	Basis											
1	P_5	.25	-60	-.04	9	1	0	0	0.0	-0.00308	89.77°	$\exists\, j \in J$ ⋗ $z_j < 0$, so
2	P_6	0.5	-90	-.02	3	0	1	0	0.0	-0.00416	89.68°	Primal Class
3	P_7	0	0	1*	0	0	0	1	1.0	-0.01414	88.92°	Algorithm is used.
	z_j	-.75	150	-.02	6	0	0	0	0			

Note: The Pivotal Element is indicated by an asterisk.

TABLE II

Tableau after one iteration for Example 1

	$P_{j\rightarrow}$	P_1	P_2	P_3	P_4	P_5	P_6	P_7		Calculations: Primal Algorithm		
$\downarrow i$	$c\rightarrow$	.75	-150	0.02	-6	0	0	0				
	Basis								$\underline{x}\downarrow$	δ_i	ANGLE	REMARKS
1	P_5	0.25	-60	0	9	1	0	0.04	0.04	-0.00309	89.76°	$\exists\ j \in J\ \ni$ $Z_j < 0$, so Primal class Algorithm is used. With the BHPM Algorithm, the basis after one iteration included P_5, P_1 and P_7.
2	P_6	0.5*	-90	0	3	0	1	0.02	0.02	-0.00416	89.68°	
3	P_3	0	0	1	0	0	0	1.0	1.0	0.0	90.00°	
	Z_j	-0.75	150	0	6	0	0	0.02	0.02			

Note: The Pivotal Element is indicated by an asterisk.

TABLE III

Tableau after two iterations for Example 1

↓i	$P_{j\rightarrow}$	P_1	P_2	P_3	P_4	P_5	P_6	P_7		Calculations: None Required		
	c→	.75	-150	0.02	-6	0	0	0	$\underline{x}\downarrow$	δ_i	ANGLE	REMARKS
	Basis											
1	P_5	0	-15	0	7.5	1	-0.5	0.03	0.03			
2	P_1	1.0	-180	0	6	0	2	0.04	0.04			
3	P_3	0	0	1	0	0	0	1	1.0			Solution is optimal. Value of the objective function is 0.05.
	Z_j	0	15	0	10.5	0	1.5	0.05	0.05			

Example 2. Consider the linear program

$$
\begin{array}{lrcl}
\text{Maximize} & 1.0x_1 + 4.0x_2 & & \\
\text{Subject to} & -2x_1 - 7.0x_2 & \leq & -163 \\
& -5x_1 - 8.0x_2 & \leq & -227 \\
& -4x_1 - 1.0x_2 & \leq & -52 \\
& - 1.0x_2 & \leq & -17 \\
& -4x_1 + 3.0x_2 & \leq & 76 \\
& -1x_1 + 1.0x_2 & \leq & 28 \\
& -2x_1 + 3.0x_2 & \leq & 98 \\
& -1x_1 + 3.0x_2 & \leq & 118 \\
& -1x_1 + 7.0x_2 & \leq & 310 \\
& 3x_1 + 8.0x_2 & \leq & 520 \\
& 1x_1 + 1.0x_2 & \leq & 95 \\
& 5x_1 + 4.0x_2 & \leq & 433 \\
& 5x_1 + 3.0x_2 & \leq & 396 \\
& 4x_1 - 1.0x_2 & \leq & 208 \\
& 1x_1 - 1.0x_2 & \leq & 34 \\
& 1x_1 - 2.0x_2 & \leq & 14 \\
& x_1 & \geq & 0 \\
& x_2 & \geq & 0.
\end{array}
$$

The diagram of the feasible region for this problem is shown in Figure 1. Solution by means of the GLO approach required two iterations, while five iterations were requried using the BHPM approach. The simplex algorithm, where movement is restricted to the polytope of the feasible region shown in Figure 1, would require additional iterations due both to the Phase I removal of artificial variables as well as the requirement (which is inherent in the simplex algorithm) of moving to neighboring extreme points.

Figure 1

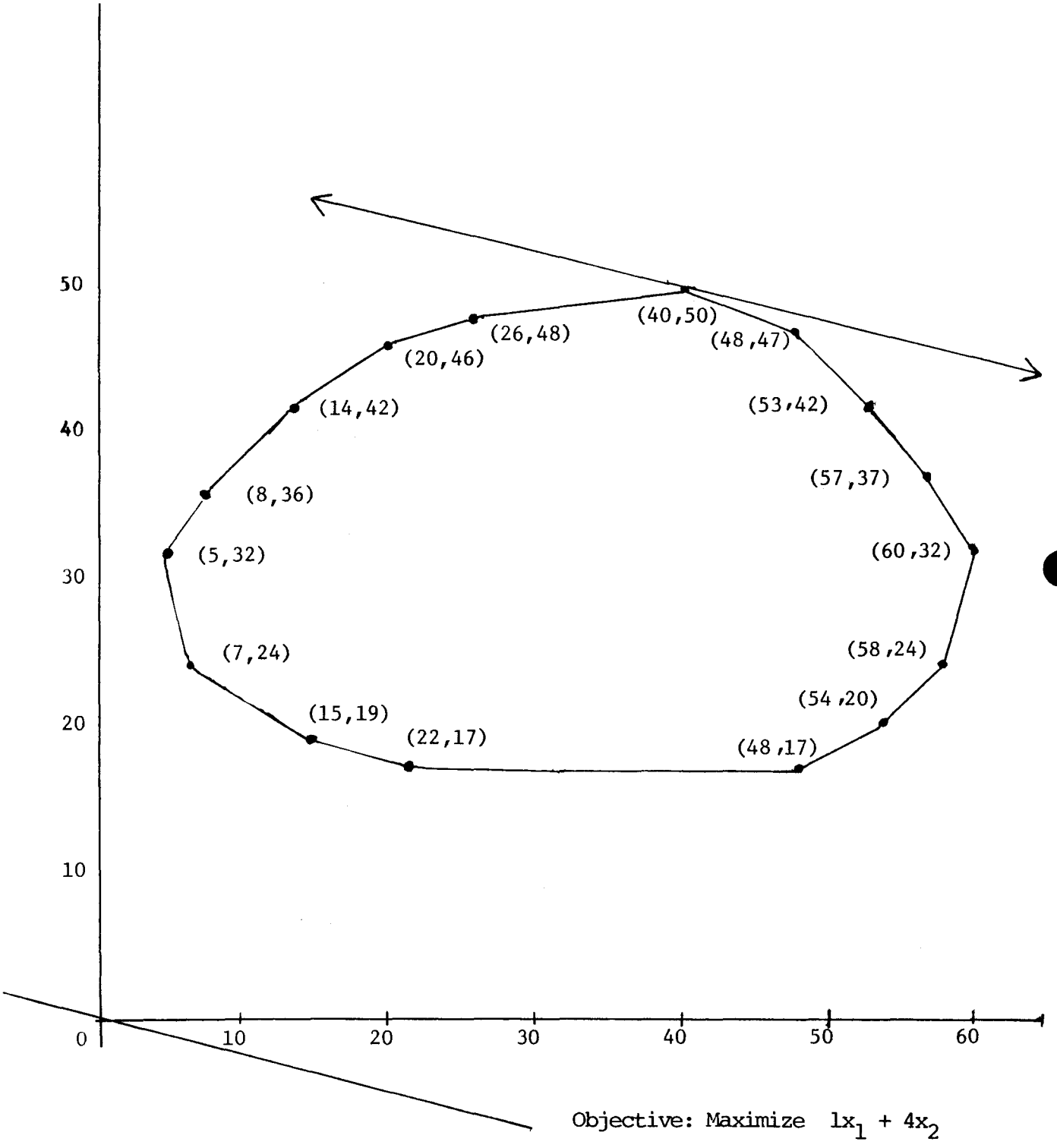
50
40
30
20
10
0
10
20
30
40
50
60
(40,50)
(48,47)
(26,48)
(20,46)
(14,42)
(53,42)
(57,37)
(8,36)
(5,32)
(60,32)
(7,24)
(58,24)
(54,20)
(15,19)
(22,17)
(48,17)
Objective: Maximize $1x_1 + 4x_2$

TABLE IV

Initial Tableau for Example 2

↓i	j→	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		Calculations : Primal Algorithm		
	c→	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\underline{x}$↓	δ_i	ANGLE	REMARKS
	Basis																						
1	P_3	-2	-7	1																-163			$\exists\ j \in J \ni z_j < 0$, so Primal class algorithm is used; P_{12} is selected to leave and P_2 to enter.
2	P_4	-5	-8		1															-227			
3	P_5	-4	-1			1														- 52			
4	P_6	0	-1				1													- 17			
5	P_7	-4	3					1												76	-1.569	67.63°	
6	P_8	-1	1						1											28	-1.732	65.16°	
7	P_9	-2	3							1										98	-2.673	49.59°	
8	P_{10}	-1	3								1									118	-3.317	36.45°	
9	P_{11}	-1	7									1								310	-3.781	23.51°	
10	P_{12}	3	8 *										1							520	-4.069	9.32°	
11	P_{13}	1	1											1						95	-2.887	45.56°	
12	P_{14}	5	4												1					433	-3.240	38.20°	
13	P_{15}	5	3													1				396	-2.874	45.82°	
14	P_{16}	4	-1														1			208			
15	P_{17}	1	-1															1		34			
16	P_{18}	1	-2																1	14			
	z_j	-1	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

Note. The Pivotal Element is indicated by an asterisk.

TABLE V

Tableau After One Iteration for Example 2

i	j→	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Calculations: Dual Algorithm		
	c_j→	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
	Basis																			x_i	α_i	ANGLE	REMARKS
1	P_3	.63		1									.88							292			$Z_j \geq 0$, $j \in J$, hence Dual Class Algorithm is used; P_{11} is selected to leave and P_1 to enter.
2	P_4	-2			1								1.0							293			
3	P_5	-3.63				1							.13							13			
4	P_6	.38					1						.13							48			
5	P_7	-5.13						1					-.38							-119	-0.5253	137.98°	
6	P_8	-1.38							1				-.13							-37	-0.4399	128.47°	
7	P_9	-3.13								1			-.38							-97	-0.5299	138.54°	
8	P_{10}	-2.13									1		-.38							-77	-0.5256	138.01°	
9	P_{11}	-3.63*										1	-.88							-145	-0.5828	145.50°	
10	P_2	.38	1										.13							65			
11	P_{13}	.63											-.13	1						30			
12	P_{14}	3.5											-.5		1					173			
13	P_{15}	3.88											.38			1				201			
14	P_{16}	4.38											.13				1			273			
15	P_{17}	1.38											.13					1		99			
16	P_{18}	1.75											.25						1	144			
	Z_j	0.5											0.5							260			

Note: The Pivotal Element is indicated by an asterisk.

TABLE VI

Tableau After Two Iterations for Example 2

↓i	j→	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	Calculations			
	c→	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
	BASIS																			$\underline{x}$↓	δ_i	ANGLE	REMARKS
1	P_3			1								.17	.72							267			
2	P_4				1							.55	1.48							373			Solution is optim
3	P_5					1						- 1	1							158			Value of the ob-
4	P_6						1					.1	.03							33			jective
5	P_7							1				-1.4	.86							86			function
6	P_8								1			-.38	.21							18			is 240.
7	P_9									1		-.86	.38							28			
8	P_{10}										1	-.59	.14							8			
9	P_1	1										-.28	.24							40			
10	P_2		1									.1	.03							50			
11	P_{13}											.17	-.28	1						5			
12	P_{14}											.97	-1.34		1					33			
13	P_{15}											1.07	-1.31			1				46			
14	P_{16}											1.21	-.93				1			98			
15	P_{17}											.38	-.21					1		44			
16	P_{18}											.48	-.17						1	74			
	Z_j											.138	.379							240			

TABLE VII

Number of Test Problems	MATRIX A		Vector b (RHS)	Vector c	ITERATION COUNT					CPU TIME (ITERATIONS ONLY)					CPU TIME (ENTIRE PROBLEM)				
					Mean			Std. Deviation		Mean in Seconds			Std. Deviation		Mean in Seconds			Std. Deviation	
	Size	Description			BHPM	GLO	BHPM+ GLO	BHPM	GLO	BHPM	GLO	BHPM+ GLO	BHPM	GLO	BHPM	GLO	BHPM+ GLO	BHPM	GLO
27	20x20	Fixed RHS 0% Sparse	1000	[-100,100]	53.370	26.593	2.007	22.592	5.213	0.820	0.384	2.147	0.366	0.082	0.909	0.474	1.918	0.365	0.085
25	20x20	Fixed RHS 34% Sparse	1000	[-100,100]	50.400 ⊕	28.880	1.745	19,472 ⊕	9.532	0.770 ⊕	0.442	1.742	0.316 ⊕	0.163	0.855 ⊕	0.526	1.625	0.345 ⊕	0.161
25	20x20	Fixed RHS 67% Sparse	1000	[-100,100]	43.080	20.360	2.116	20.613	10.480	0.635	0.268	2.369	0.324	0.152	0.720	0.350	2.057	0.321	0.152
5	20x20	Varying RHS 0% Sparse	[-1000,1000]	[-100,100]	65.600	29.200	2.247	19.667	6.611	1.046	0.436	2.399	0.290	0.106	1.150	0.530	2.170	0.304	0.114
25	20x40	Fixed RHS 0% Sparse	1000	[-100,100]	150.800 ⊕	24.920	6.051	36.507 ⊕	9.478	3.776	0.558	6.767	0.999 ⊕	0.228	4.052	0.689	5.881	1.107	0.234
26	20x40	Fixed RHS 34% Sparse	1000	[-100,100]	138.923 ⊕	36.846	3.770	37.550 ⊕	14.318	3.437	0.958	3.588	0.951 ⊕	0.401	3.678 ⊕	1.097	3.353	1.059 ⊕	0.396
25	20x40	Fixed RHS 67% Sparse	1000	[-100,100]	119.346 ⊕	20.308	5.877	44.349 ⊕	13.248	2.901	0.437	6.638	1.129 ⊕	0.329	3.100 ⊕	0.559	5.546	1.226 ⊕	0.330
30	20x40	Varying RHS 0% Sparse	[-1000,1000]	[-100,100]	131.433 ⊕	24.833	5.293	37.578 ⊕	8.921	3.332	0.577	5.775	1.025 ⊕	0.219	3.540	0.712	4.972	1.124	0.220

NOTE 1: In order to contain expenses for computer time, the maximum number of iterations allowed for any one problem was 175. No test problems were affected by this restriction when the GLO algorithm was used. Some problems, however, failed to converge in 175 iterations with the BHPM approach. The means and standard deviations which are artificially low due to this truncation are indicated in the table by the symbol ⊕ .

NOTE 2: In the case of a 20x20 matrix, the maximum number of iterations allowed was 100. Only one problem failed to converge in 100 iterations.

6. Test Results

The Gradient Linear Optimization Method was employed for the solution of a set of test problems of various sizes, and the results were compared with the results obtained when the same problems were solved by the BHPM approach. These tests were run on an IBM 370/165 using a series of programs designed mainly by James O. Kitchen of the University of North Carolina Computation Center [2]. Constraint coefficients were randomly generated in the interval [-100, 100]. The results of these tests are shown in Table VII.

Examination of the test results in Table VII reveals that for every group of test problems, on the average, the GLO algorithm required fewer iterations than did the BHPM algorithm. Similarly, the mean CPU times (for iterations only as well as for the entire problem) were consistently lower with GLO. In fact, on almost every individual problem, iteration count and CPU time were lower for GLO than BHPM. In addition to the results shown in Table VII, tests were also run on some larger problems, 35 x 55 or larger. Again, GLO was better, although time and money constrained us from running a sufficient number for statistical significance.

The results of test runs with the simplex algorithm are not included due to a marked disparity in coding techinques between the available simplex programs and those used for the BHPM and GLO algorithms. The latter two programs were coded similarly, which permitted a comparison of CPU times. Times should only be used for relative comparisons, in the sense that they were made on the same machine with the same coding techniques. That is, no effort was made to use the most efficient coding techniques. With respect to iteration count, the mean number of iterations required for solution by the GLO method ranged

from 1.015 m to 1.842 m. No particular relationship between the sparseness of the coefficient matrix and the efficiency of the GLO method was observed.

7. Degeneracy and Cycling

Although the authors do not have a proof that degeneracy and cycling can be avoided, in the hundreds of test problems which were run, no cycling was observed. Included in the testing were several problems which do cause cycling with other algorithms (e.g., Example 1 of this paper), and each of these converged with the GLO method.

REFERENCES

1. Hadley, G.: Linear Programming, Addison-Wesley, 1962.

2. Kitchen, J.O., Shachtman, R.H.: "A Linear Programming System". This is a system of seven programs designed to compare the BHPM method, variants of GLO methods and mixtures. It accepts the corresponding matrices for existing examples or generates random problems for testing.

3. Kuhn, H. W., Quandt, R. E.: "An Experimental Study of the Simplex Method", Proceedings of Symposia in Applied Math, 15 (1963) 107-124.

4. Saksena, C. P., Cole, A. J.: "The Bounding Hyperplane Method of Linear Programming", Operations Research, 19 (1971) 1-18.

5. Zionts, S.: "The Criss-cross Method for Solving Linear Programming Problems", Management Science, 15 (1969) 426-445.