ORiON, Vol. 8, No. 2, pp. 69-89 *ISSN 0259-191-X*

TECHNIQUES THAT STRIVE TO COMBAT THE INFLUENCE OF DEGENERACY IN LINEAR PROGRAMMING PROBLEMS

J H Nel[1]
Department of Computer Science
University of the North, Sovenga

and

J M Hattingh
Department of Computer Science
Potchefstroom University for CHE, Potchefstroom

SUMMARY

Degeneracy can cause enormous problems when solving large scale linear programming problems. This is not only because there is a possibility that the problem can cycle, but also because a large number of iterations can be executed that do not improve the objective.

In this article a procedure which utilizes derived reduced costs is discussed. The derived reduced cost of a non-basic variable is defined in such a way that it makes the introduction of the non-basic variable into the basis unattractive if such a decision fails to improve the objective. The procedure deliberately strives to combat degeneracy using derived reduced costs, but it also utilizes the advantageous properties of the classical gradient methods.

Results achieved with the method are also reported.

1. Paper read at the Joint Annual Conference of the South African Statistical Society and the Operations Research Society of South Africa, November 1989, WITS.

1. INTRODUCTION

The solution of any programming problem consists of the following two choices for moving from one extreme point to another: the direction which is taken, and the length of the step. In linear programming problems the direction of movement is determined by the choice of the pivot column, while the length of the step is influenced by both the choice of the pivot column and pivot row.

Two useful techniques to select the pivot row are those of Wolfe [19] and Wolfe and Cutler [20]. Ryan [16] stresses that the problem of degeneracy cannot be ignored and demonstrates how, by simply applying Wolfe's [19] algorithm, the number of pivots on a degenerate point can be drastically reduced. Wolfe and Cutler [20] have also suggested that should the selection of a pivot row be at issue, the pivot row should be selected such that the pivot is as large as possible. This promotes numerical stability.

The purpose of most anti-degeneracy procedures (row selection procedures) is purely to prevent the problem from cycling. These procedures do not take performance into account at all. (See also Charnes [3] and Dantzig, Orden and Wolfe [5]).

Column selection procedures, on the other hand, can ensure good performance, but usually do not take into account the presence of degeneracy. (Harris [9] and Crowder and Hattingh [4]). Other column selection procedures that prevent the problem from cycling and which are practical to implement are those of Bland [2] and Gill [7].

Examples of a few anti-degeneracy techniques (column selection procedures) that do take performance into account are those of Welman [18], Janse van Vuuren [11] and Nel [15]. It is, however, a cause for concern to realise that, in practice, few if any anti-degeneracy techniques are implemented in computer programs. Degeneracy, nevertheless, remains a topical subject for research purposes. [1]

Research is continuously undertaken to discover whether alternative methods exist for the solving of linear programming problems. Amongst those techniques which have

1. During 1990 an article by Gal [6] was published in ORION that examines degeneracy in general by means of the theory of graphs. This theory has not yet been examined fully and constitutes a new field of research.

been developed are those by Khachiyan [13] and Karmakar [12]. The expectation was that these methods could lead to the development of techniques that were as efficient as the simplex method, if not more so. The debate (see Kozlov [14]) concerning the effectiveness of these methods is still continuing and the methods are not yet being implemented in linear programming packages on a large scale. Kozlov [14] criticises Karmarkar's method as follows: "Because no one has been able to duplicate Karmakar's results in solving real problems, many in the scientific community remain skeptical" and he adds: "Much of the controversy seems to stem from Karmarkar's and Bell Labs' refusal to provide precise details of the algorithm's implementation".

The above-mentioned techniques are still, therefore, in a developmental phase and it cannot yet be proved that they perform better than simplex related techniques. The latter remain the most important techniques for the solution of large scale linear programming problems. It is for this reason that simplex related techniques and their refinements are still being examined.

The purpose of this study is to develop techniques which will counter the effect of degeneracy by reducing the number of iterations and solution time. The technique employs the principle of derived reduced costs.

In the second paragraph derived reduced costs are discussed according to the method proposed by Welman [18]. In the third paragraph a modification to the method is proposed. Results and conclusions are extrapolated from the modified method. The final conclusions are arrivec at and recommendations are made in the fourth paragraph.

2. DERIVED REDUCED COSTS

2.1 THE RATIONALE OF DERIVED REDUCED COSTS

Degeneracy occurs when at least one of the basic variables is zero. When the minimum ratio is computed to select the variable that must leave the basis, a degenerate pivot row is selected when the entry in the pivot column corresponding to that row is positive. This results in a degenerate step with no improvement in the objective.

When no positive entries exist in a pivot column opposite a zero on the right hand side the pivot row is selected such that the corresponding value on the right hand side in the pivot row is positive. An improvement in the value of the objective function will result.

The method discussed in this article deliberately attempts to select the pivot column with as few as possible positive entries opposite the zeroes on the right hand side. The ideal is to select a column for which there are no positive entries in the column opposite zeroes on the right hand side, so that the value of the objective function will improve.

The method can be summarised as follows (the normal notation which is used in Hadley [8] and others is used here):

Let $x_{\delta i} = \dfrac{1}{\delta + x_{Bi}}$; with $\delta > 0$ and very small and

x_{Bi} the i-th component of the present basic feasible solution, $i = 1, 2, \ldots, m$.

If we denote the $x_{\delta i}$ as a vector $\underline{x}_\delta$, we define the derived reduced cost associated with column j as:

$$\begin{aligned}(c_j - z_j)^* &= c_j - (\underline{c}_B^*)^T B^{-1}\underline{a}_j \\ &= c_j - (\underline{c}_B^T - \underline{x}_\delta^T)B^{-1}\underline{a}_j \quad (\text{where } \underline{c}_B^* = (\underline{c}_B - \underline{x}_\delta)) \\ &= c_j - \underline{c}_B^T B^{-1}\underline{a}_j + \underline{x}_\delta^T B^{-1}\underline{a}_j \\ &= c_j - z_j + \underline{x}_\delta^T B^{-1}\underline{a}_j \quad (\text{for minimization}).\end{aligned}$$

The minimum of $(c_j - z_j)^*$ for the j for which

$$c_j - z_j = c_j - \underline{c}_B^T B^{-1}\underline{a}_j < 0 \text{ for } j = 1, 2, \ldots, n$$

is used to determine the index of the pivot column k.

The pivot row, row r, is determined by using the normal minimum ratio test, that is

$$\Theta = \frac{x_{Br}}{y_{rk}} = \min_i \{\frac{x_{Bi}}{y_{ik}};\ y_{ik} > 0;\ i = 1, 2, \ldots, m\},$$

where y_{rk} is element r of $\underline{y}_k = B^{-1}\underline{a}_k$.

The following example is used to illustrate the use of derived reduced costs.

The first table (Table 2.1) of Beale's problem [1], which cycles if it is solved by means of the simplex method, is as follows:

	$\underline{c}_B$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	b
x_5	0	$\frac{1}{4}$	-60	$\frac{-1}{25}$	9	1	0	0	0
x_6	0	$\frac{1}{2}$	-90	$\frac{-1}{50}$	3	0	1	0	0
x_7	0	0	0	1	0	0	0	1	1
$c_j - z_j$		$\frac{-3}{4}$	150	$\frac{-1}{50}$	6	0	0	0	

Table 2.1

Columns $\underline{a}_1$ and $\underline{a}_3$ are candidates for the pivot column. Furthermore,

$$\underline{x}_\delta{}^T = [\frac{1}{\delta}; \frac{1}{\delta}; \frac{1}{1+\delta}]$$

and $(c_1 - z_1)^* = -\frac{3}{4} + \frac{3}{4\delta}$; $(c_3 - z_3)^* = -\frac{1}{50} - \frac{3}{50\delta} + \frac{1}{1+\delta}$.

If $\delta = 0{,}001$ (for example), then $(c_1 - z_1)^* = 749.25$ and $(c_3 - z_3)^* = -59.021$.

The third column of Table 2.1 will then become basic in the next step. As there are only negative entries in the third column opposite the zeroes on the right hand side, $\underline{x}_\delta$ causes the pivot row to be selected as the row for which $x_{Bi} \neq 0$, that is the third row. Degeneracy is eliminated in the next step.

The role played by $\underline{x}_\delta$ in the selection of the pivot column is substantiated as follows:

Let $I = \{i / x_{Bi} = 0\}$.

If $x_{Bi} = 0$, then $x_{\delta i} = \dfrac{1}{\delta + x_{Bi}} = \dfrac{1}{\delta}$ is a large positive number.

Thus, for a non-basic column j, $\underline{x}_\delta B^{-1}\underline{a}_j$ is large positive if a $y_{ij} > 0$ exists for $i \in I$. Therefore, $(c_j - z_j)^* = c_j - \underline{c}_B{}^T B^{-1}\underline{a}_j + \underline{x}_\delta{}^T B^{-1}\underline{a}_j$ has a high positive value and makes the selection of column j as pivot column unattractive.

If $y_{ij} < 0$ and $x_{Bi} = 0$, the term $y_{ij}x_{\delta i}$ makes a large negative contribution to the value of $(c_j - z_j)^*$, and it increases the attractiveness of the column as pivot column. Should both negative and positive entries occur in the columns opposite the zeroes on the right hand side, the column where the number of negative entries predominate will be selected.

3. DOUBLE BLOCK DERIVED REDUCED COSTS (DBDRC)

3.1 INTRODUCTION

Many experiments have been undertaken using derived reduced costs in various forms. From the experiments it became clear that aspects such as multiple pricing and the use of gradient methods cannot be ignored. A method [15] was developed in which the influence of the gradient methods and the derived reduced costs can be controlled.

The method is the following: a large multiple pricing block (size p) is selected by means of either the simplex or the partially normalized method [4]. Let the indices of ~~the columns selected in this manner be:~~

$$J = \{j_1, j_2, \ldots, j_p\}.$$

For each of the columns with indices in J, the reduced costs $d_j = (c_j - z_j)$ and $\pi_j = \underline{x}_\delta{}^T B^{-1}\underline{a}_j$ are calculated. Let $Q = \{\pi_{j1}, \pi_{j2}, \ldots, \pi_{jp}\}$.

From this large block a smaller block of size q is selected by choosing the columns that correspond with the q smallest values in Q. If $q = 1$, the problem reduces to single pricing, with a derived criterion for the choice of the pivot column.

This second block is optimized entirely by means of the simplex method. The order in which the columns in the second block were selected was not subjected to experimentation. It is assumed that the optimization of the second block occurs mainly in the main memory of the computer and that relatively few read and write activities, which are usually slow, take place at this stage.

The purpose of Double Block Derived Reduced Costs (DBDRC) is the following: in the selection of the first block one ensures that the columns which are eventually used give profitable directions. Both the simplex and the partially normalized methods are gradient methods (Crowder and Hattingh [4]), and are therefore used to achieve "profitable directions". In the selection of the second block, degeneracy is taken into account. Indeed, in the construction of this second block, care is taken to include columns that tend to improve the objective wherever possible.

Experiments were conducted with various sizes of the second block. The more the size of the first block approximates that of the second, the more limited the effect of the derived reduced costs will be.

3.2 IMPLEMENTATION OF THE METHOD

The steps for the DBDRC method are as follows:

Step 1

Calculate a (feasible) solution by means of $\underline{x}_B = B^{-1}\underline{b}$.

Step 2

For every non-basic column j calculate

$$d_j = c_j - z_j \text{ and}$$

$$\pi_j = (\underline{x}_j^T B^{-1})\underline{a}_j$$

with $\underline{x}_\delta$ defined as above.

Step 3

If all $c_j - z_j$ values ≥ 0, (for minimization), stop, the solution is optimal, else go to step 4.

Step 4

Select columns p[1] by means of the simplex or the partially normalized methods. The columns constitute a multiple pricing block. Note the corresponding indices of these columns, as well as the corresponding π values, that is

$$P = \{j_1, j_2, \ldots, j_p\} \text{ and}$$

$$Q = \{\pi_{j1}, \pi_{j2}, \ldots, \pi_{jp}\}.$$

Step 5

Select a smaller block with size q, using the smallest values in Q.

Step 6

Update all the columns of the second block.

Step 7

This block, then, constitutes the multiple pricing block, which is optimized by means of the simplex method. Pivot until the block selected in step 6 becomes optimal.

Step 8

Go to step 1.

1. $p = 20$ was used in experimentation, and we assume that 20 columns constitute a sufficiently "large" block.

Remark

The method is very effective in terms of time for the following reason: the values of $\pi_j = (\underline{x}_\delta{}^T B^{-1})\underline{a}_j$ are calculated simultaneously with the values of $z_j = (c_B{}^T B^{-1})\underline{a}_j$. The values of π_j can be calculated without calculating the entries of the non-basic columns $(B^{-1}\underline{a}_j)$.

3.3 EXPERIMENTAL RESULTS

3.3.1 TEST PROBLEMS AND NOTATION

Four problems were solved by means of the DBDRC method. ADLITTLE and ISRAEL are well-known problems in the literature. The following references suggest some examples where they were used: Gill [7][1], Crowder & Hattingh [4] and Kozlov [14].

On the surface it appears that only a few test problems were used for experimental purposes. However, to determine the effectiveness of the DBDRC method a large number of permutations were taken into account. To generate Table 3.3, for example, a total of 364 runs were made which varied from 30 minutes to 3 hours per run.

The dimensions of the test problems are the following:

PROBLEM	ROWS	COLUMNS	NON-ZERO ELEMENTS	%DENSITY
ADLITTLE1	57	138	424	5,4%
ADLITTLE2	57	138	424	5,4%
ISRAEL1	175	316	2443	4,4%
ISRAEL2	175	316	2443	4,4%

Table 3.1

The various values used for q are: 6, 8, 10, 12, 14, 16 and 18.

1. Gill acknowledges D.M. Gay for making available the test problems. He also refers to Gay's article 'Electronic mail distribution of linear programming test problems' in *Mathematical Programming Society COAL Newsletter*, 13, 10-12 (1985).

The thirteen values used for δ are displayed in the following table:

NUMBER	VALUE OF δ	NUMBER	VALUE OF δ
1	0,9		
2	0,8	8	$0,1 \times 10^{-2}$
3	0,7	9	$0,1 \times 10^{-4}$
4	0,6	10	$0,1 \times 10^{-6}$
5	0,5	11	$0,1 \times 10^{-8}$
6	0,1	12	$0,1 \times 10^{-10}$
7	0,01	13	$0,1 \times 10^{-12}$

Table 3.2

The entries in the tables that follow indicate the average number of major iterations relative to the simplex method without multiple pricing for each of the above-mentioned four problems. Averages for the DBDRC method for all values of δ and for all values of q (the size of the second multiple pricing block) are also reported in the tables. The corresponding figures for the simplex method without the derived reduced costs are also produced in the tables.

3.3.2 COMPARISONS WITH THE SIMPLEX METHOD WITH MULTIPLE PRICING

The four test problems were solved with the DBDRC method. The first block with a size of $p = 20$ columns was selected by means of the simplex method and the second block with q columns was selected according to the minimum of the π_j values. The result is found in Table 3.3. In Table 3.3 comparisons of the simplex method with multiple pricing are made with the DBDRC method with corresponding block sizes.

METHOD: DBDRC (FIRST BLOCK SELECTED WITH SIMPLEX METHOD)
NUMBER OF MAJOR ITERATIONS RELATIVE TO SIMPLEX WITHOUT MULTIPLE PRICING

NUM=	q							AVG.		AVG.
BER	6	8	10	12	14	16	18	DBDRC		SIMP
1	,264	,244	,232	,216	,230	,174	,170	,219	X	,212
2	,244	,245	,209	,196	,223	,174	,170	,209	|	,212
3	,238	,219	,203	,174	,195	,169	,170	,196	|	,212
4	,270	,208	,180	,187	,188	,168	,170	,196	|	,212
5	,265	,257	,184	,197	,188	,178	,171	,206	|	,212
6	,284	,225	,219	,189	,177	,163	,175	,204	|	,212
7	,271	,228	,200	,199	,157	,173	,197	,204	|	,212
8	,264	,247	,197	,200	,156	,173	,187	,203	|	,212
9	,260	,250	,194	,199	,164	,161	,184	,202	|	,212
10	,263	,226	,190	,185	,163	,161	,184	,196	|	,212
11	,263	,228	,190	,185	,163	,161	,184	,196	|	,212
12	,263	,228	,190	,185	,163	,161	,184	,196	|	,212
13	,268	,224	,190	,189	,163	,172	,182	,198	X	,212
AVG.										
DBDRC	,263	,233	,198	,192	,179	,168	,179			
SIMP	,302	,268	,206	,206	,176	,164	,163			

Table 3.3

Duncan's [17] method of multiple comparisons is used to determine whether there are significant differences between the averages of the DBDRC method. The results are as follows:

```
q                 6       8       10      12      14      18      16

AVG.(DBDRC)     ,263    ,233    ,198    ,192    ,179    ,179    ,168

                                 X----------------------------X

                                                 X---------------------X
```

According to Duncan's method ($\alpha = 0{,}05$) the DBDRC method with the size of the second block taken at 10, 12, 14, and 18 produces almost similar results, while the same method with block sizes of 14, 16 and 18 produces almost the same results.

The same test was done on the averages for all values of q for each of the 13 cases to determine the influence of δ. According to Duncan's test ($\alpha = 0{,}05$) there is no significant difference in the 13 cases and therefore within the range of values of δ we have chosen. It thus does not seem to matter what value is used.

The DBDRC method produces better results than the simplex method with corresponding values for q = 6 (12,9% better), 8 (13,1% better), 10 (3,9% better) and 12 (6,8% better).

A TRU ("Total resource unit") [10] value was also printed by the computer for each run made, and the value is calculated as follows:

TRU = ,005 x processor time consumed
+ ,002 x drum page transfers
+ ,002 x disc page transfers
+ ,010 x filestore transfers
+ ,001 x mainstore occupancy
+ ,012 x spooled records.

For example, a job for which processor time consumed = 2800, disc page transfers = 370, and mainstore occupancy = 1500, will be charged

$$(2800 \times 0{,}005) + (370 \times 0{,}002) + (1500 \times 0{,}001) = 16 \text{ TRUs.}$$

The TRU value for each main pivot (TRU/number of major iterations) was calculated for each problem. The results for ISRAEL1 and ISRAEL2 are given separately, and were only calculated for the final 6 cases. It is impractical to determine the average TRU value/major iteration for all the test problems as the variation is too large. The results are only given for ISRAEL1 and ISRAEL2 as the test problems are relatively large and give a good indication of the behaviour of the method for large scale linear programming problems in general.

ISRAEL1: TRU/MAJOR ITERATIONS

NUM=				q				AVG.		AVG.
BER	6	8	10	12	14	16	18	DBDRC		SIMP
8	60,5	59,3	64,0	64,4	67,4	94,5	113,2	74,7	X	83,5
9	46,6	60,2	69,2	84,5	64,7	74,0	115,6	73,5	|	83,5
10	56,8	56,7	55,9	73,9	86,2	85,5	107,0	74,6	|	83,5
11	55,9	54,8	58,7	69,8	89,7	83,9	115,3	75,4	|	83,5
12	53,9	53,2	58,9	69,7	86,8	82,0	110,3	73,5	|	83,5
13	58,5	58,0	58,8	74,5	89,1	83,1	106,5	75,5	X	83,5
AVG. DBDRC	55,4	57,0	60,9	72,8	80,6	83,8	111,3			
	X	——	X							
					X	X				
SIMP	50,5	70,4	105,1	96,2	95,9	82,5	83,9			

Table 3.4

According to Duncan ($\alpha = 0{,}05$), the average TRU value for each major iteration is indistinguishable for q = 6, 8 and 10. It is evident that the greater the number of columns in the second block, the more expensive the method becomes. There is no difference in the TRU values if the 6 cases are examined, which indicates that δ does not exercise any significant influence on the TRU value.

For q = 8 (19,0% less), 10 (42,1% less), 12 (24,3% less) and 14 (16,0% less) the TRU value for each major iteration of the DBDRC method is less than the TRU value for each major iteration of the simplex method with corresponding values of q for ISRAEL1.

ISRAEL2: TRU/MAJOR ITERATIONS

NUM= BER	q 6	8	10	12	14	16	18	AVG. DBDRC		AVG. SIMP
8	57,4	71,6	75,1	94,5	85,0	95,6	126,4	86,5	X	98,1
9	67,7	82,7	79,8	105,3	85,2	120,8	112,8	93,5	|	98,1
10	61,7	54,1	82,9	103,3	105,5	118,2	111,3	91,0	|	98,1
11	62,5	56,4	90,5	97,5	91,1	110,7	111,1	88,5	|	98,1
12	61,4	54,4	92,0	88,6	94,7	112,9	113,6	88,2	|	98,1
13	58,2	54,7	88,6	96,8	102,3	110,5	114,9	89,4	X	98,1
AVG. DBDRC	61,5	62,3	84,8	97,7	93,4	111,4	115,0			
	61,5	62,3	84,8	93,4	97,7	111,4	115,0			
	X——	——X	X——	——X						
				X——	——X	X——	——X			
SIMP	54,6	73,5	77,7	105,0	118,0	134,3	123,7			

Table 3.5

According to Duncan ($\alpha = 0{,}05$) the average TRU value for each major iteration is indistinguishable for q = 6 and 8; 10 and 14; 14 and 12; 16 and 18. It is also very evident here that the greater the dimension of the second block the more expensive the method becomes. There is no difference in the TRU values if the 6 cases are examined, which indicates that δ does not have a significant influence on the TRU value.

For q = 8 (15,2% less), 12 (7,0% less), 14 (20,8% less), 16 (17,1% less) and 18 (7,0% less) the TRU value for each major iteration is less for the DBDRC method than the TRU value for each major iteration of the simplex method with corresponding values of q for ISRAEL2.

3.3.3 CONCLUSIONS FOR THE SIMPLEX METHOD

Tables 3.4 and 3.5 are included to give an indication of the "costs" associated with the sizes of the second block. These blocks clearly indicate that it is not cost effective to select a "too large" dimension for the second block. Table 3.3 indicates that the number of iterations do decrease as the size of the second block increases but the costs also increase dramatically!

Figures 3.1 and 3.2 clearly indicate for ISRAEL1 and ISRAEL2, jointly, the relationship between the number of major iterations, TRU value and the size of the multiple pricing block. The bigger the second block the fewer the number of major iterations and the more expensive the method becomes.

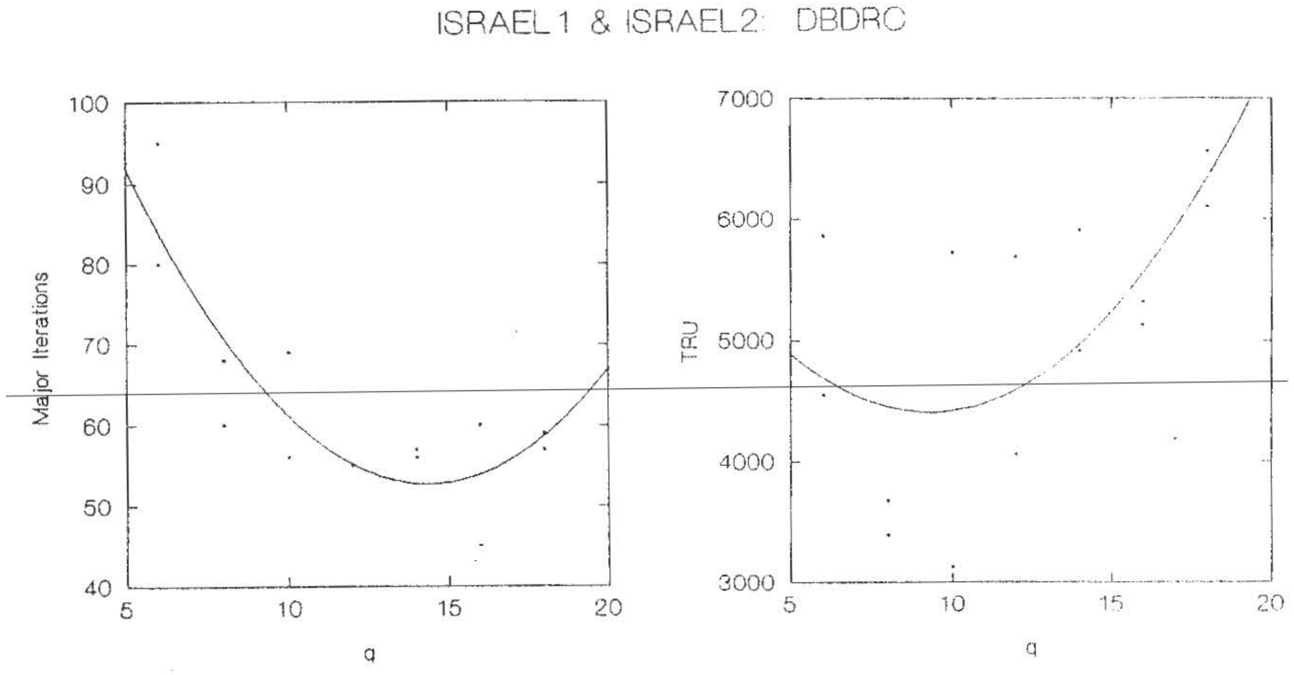

Figure 3.1

Figure 3.2

Finding the point of equilibrium, the point where the method is still relatively cheap and the number of major iterations are relatively low, is important.

The DBDRC method with q = 10 and $\delta = 0,1 \times 10^{-6}$ is recommended as the method which gives the best results in terms of iterations and time. The reason is the following: the average number of major iterations for $\delta = 0,1 \times 10^{-6}$ for all values of q is the smallest, and it appears that the average number of major iterations stabilises from this value of δ and smaller. The average time for each major iteration is almost the same for q = 6, 8 and 10 and this group provides the best results. From this group, q = 10 has the minimum number of average major iterations.

3.3.4 THE PARTIALLY NORMALIZED METHOD

A complete description of the partially normalized method can be found in [4], as discussed by Crowder and Hattingh. For the purposes of this study the first block was selected with a size of 20 columns by means of the partially normalized method and the number of rows included was set at 6 throughout. The second block was selected according to the q smallest values of π_j $(= \underline{x}_\delta{}^T B^{-1} \underline{a}_j)$. The result is reflected in Table 3.6. Experiments were conducted where the π_j values were divided by the norm, but poorer results were obtained and these are not reported here.

In Table 3.6 comparisons of the partially normalized method are made with the DBDRC method with corresponding block sizes.

METHOD: DBDRC (FIRST BLOCK SELECTED WITH PARTIALLY NORMALIZED METHOD)
NUMBER OF MAJOR ITERATIONS RELATIVE TO SIMPLEX WITHOUT MULTIPLE PRICING

NUM=	q							AVG.		AVG.
BER	6	8	10	12	14	16	18	DBDRC		NORM
1	,305	,243	,236	,211	,201	,177	,158	,219	X	,204
2	,341	,249	,250	,200	,192	,181	,158	,224		,204
3	,284	,300	,225	,192	,180	,163	,158	,215		,204
4	,311	,225	,225	,208	,181	,177	,165	,213		,204
5	,315	,257	,210	,207	,212	,163	,171	,219		,204
6	,306	,247	,220	,183	,232	,171	,182	,220		,204
7	,269	,275	,237	,212	,170	,174	,157	,213		,204
8	,269	,279	,220	,202	,169	,182	,170	,213		,204
9	,278	,280	,227	,205	,167	,186	,172	,216		,204
10	,276	,285	,210	,207	,171	,178	,172	,214		,204
11	,276	,285	,210	,207	,171	,178	,172	,214		,204
12	,276	,285	,210	,207	,171	,178	,172	,214		,204
13	,287	,285	,210	,207	,171	,165	,167	,213	X	,204
AVG. DBDRC	,292	,269	,222	,204	,184	,175	,167			
			X	X						
				X	X					
					X		X			
NORM	,268	,262	,188	,205	,161	,173	,174			

Table 3.6

According to Duncan's method ($\alpha = 0{,}05$), q = 10 and 12; 12 and 14, and 14, 16 and 18 produce similar results. There is no difference in the average number of main pivots with regard to the 13 cases.

It appears that not much success is achieved with the DBDRC method in conjunction with the partially normalized method, especially not in cases where the size of the second block is relatively small.

3.4 FINAL COMPARISONS

In Table 3.7 the following methods are compared:

(i) the simplex method with multiple pricing with a block size of q;

(ii) the DBDRC method where the simplex method with multiple pricing of 20 was used to select the first block, and a second block with a size of q was selected for which the π_j values were a minimum and $\delta = 0{,}1 \times 10^{-6}$;

(iii) the partially normalized method with multiple pricing with a block size of q, and the number of rows considered of 6, and

(iv) the DBDRC method where the first block was selected by means of the partially normalized method with multiple pricing of 20 and the second block was selected with a size of q for which the π_j values were a minimum and $\delta = 0{,}1 \times 10^{-6}$.

SUMMARY OF RESULTS

METHOD	q: 6	8	10	12	14	16	18	AVG.
(i) SIMPLEX	,302	,268	,206	,206	,176	,164	,163	,212
(ii) SIMP/DBDRC	,263	,226	,190	,185	,163	,161	,184	,196
(iii) PAR. NORM	,268	,262	,188	,205	,161	,173	,174	,204
(iv) NORM/DBDRC	,276	,285	,210	,207	,171	,178	,172	,214

Table 3.7

The average results for the DBDRC method ($\delta = 0{,}1 \times 10^{-6}$) are even better than the average results of the partially normalized method (3,9%).

3.4 GRAPHS

Graphs are drawn after each major iteration of the rate at which the value of the objective function decreases (minimization is involved here). This is done to demonstrate the influence of derived reduced costs on the number of major iterations and the value of the objective function for a specific problem.

Figures 3.3 and 3.4 are examples representative of the graphs resulting from experimentation.

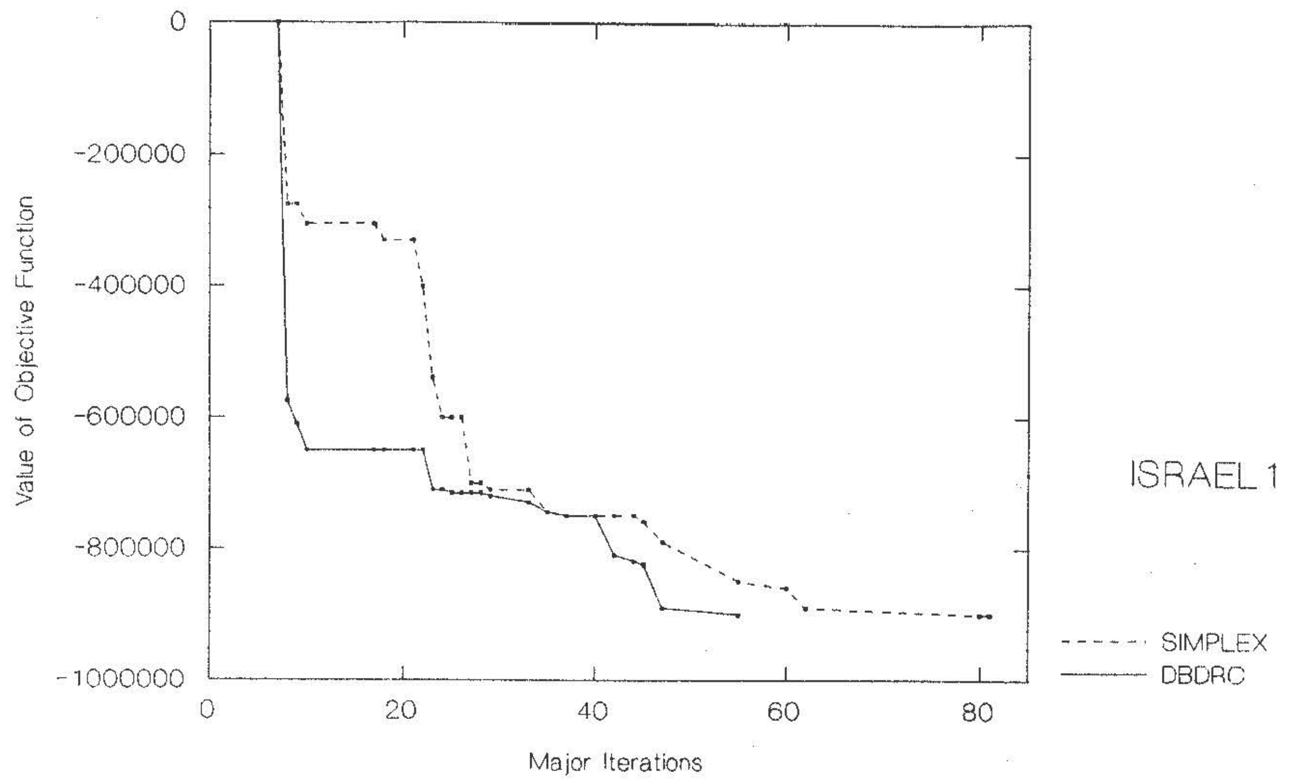

Figure 3.3

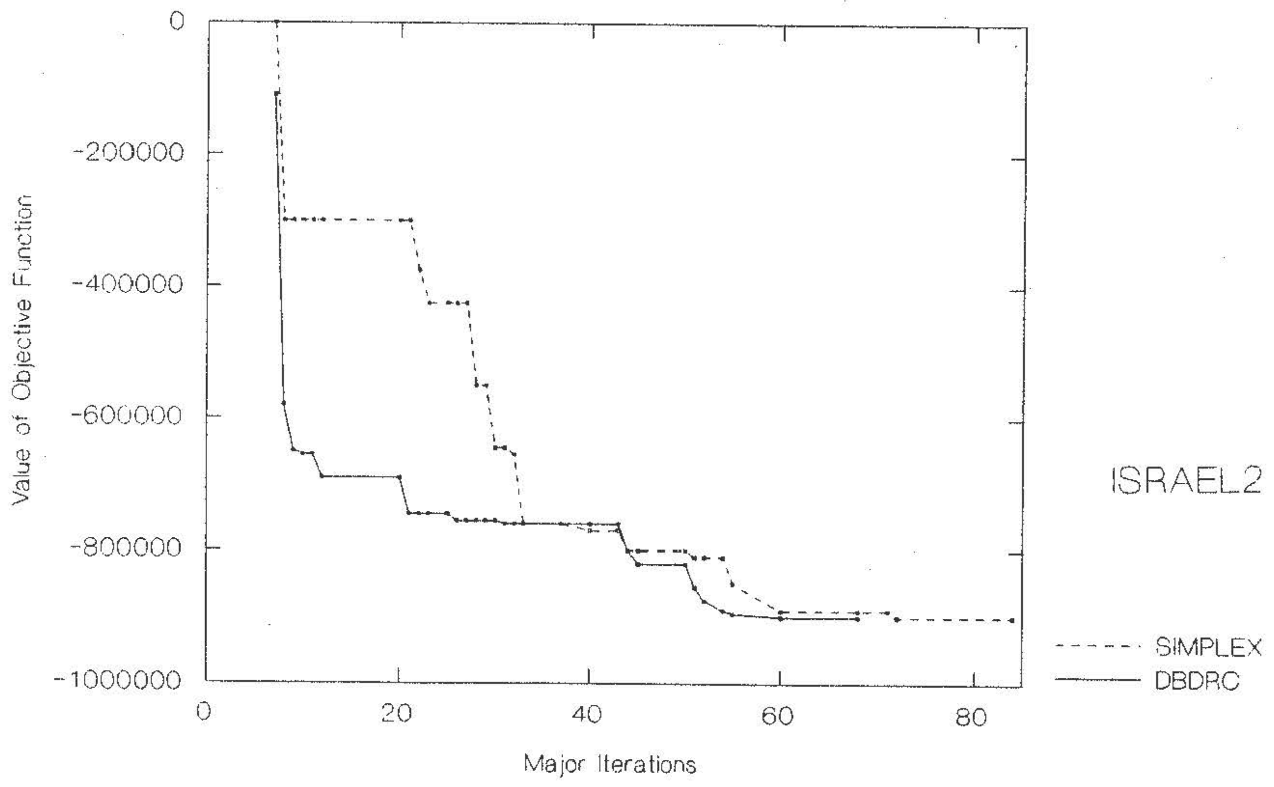

Figure 3.4

Discussion

The value of the objective function initially decreases faster for the DBDRC method, but later the decrease in the value of the objective function is less, which results in longer "tails".

The tendency initially to rapidly decrease the value of the objective function is a very important characteristic of the variations of derived reduced costs. This characteristic can be usefully employed in the solution of problems which are highly degenerate as the number of zeroes on the right hand side are already reduced considerably early in the solution process.

Long "tails" may also arise from situations where, at a certain stage, all pivot columns have at least one positive entry corresponding to a zero on the right hand side. In such a case, a pivot leading to degeneracy is inevitable.

In the past a number of experiments have been conducted with the occurrence of long "tails" and great success has been achieved by, amongst others, Harris [9] and Crowder and Hattingh [4]. Relatively speaking, there is not much literature available concerning the initial rapid decline in the value of the objective function. This study reveals that much promise resides in this field. One of the possibilities is to commence the solving process by means of the DBDRC method, and as soon as the decline in the objective function is no longer exceptionally high, to switch to a method (Crowder and Hattingh [4] or Harris [9], for example) that ensures that an optimal solution is arrived at fairly rapidly.

4. CONCLUSIONS

The DBDRC method produces, generally speaking, good results in terms of the number of iterations and in terms of time. The reason for this is the fact that the method employs the good characteristics of gradient directions and that degeneracy is, to a large extent, counteracted early in the solution process.

This method can be employed with success in the solution of large scale linear programs, especially when the problems are highly degenerate.

Bibliography

[1] E.M.L. BEALE, Mathematical programming in practice, John Wiley, New York, (1968).

[2] R.G. BLAND, New finite pivoting rules for the simplex method, *Mathematics of operations research*, 2(2), 103 - 107 (1977).

[3] A. CHARNES, Optimality and degeneracy in linear programming, *Econometrica*, 20, 160 - 172 (1952).

[4] H. CROWDER en J.M. HATTINGH, Partially normalized pivot selection in linear programming, *Mathematical programming study*, 4, 12 - 25 (1975).

[5] G.B. DANTZIG, A. ORDEN, P WOLFE, The generalized simplex method for minimizing a linear form under linear inequality constraints, *Pacific journal of mathematics*, 5(2), 183 - 195 (1955).

[6] T. GAL, Degeneracy problems in mathematical programming and degeneracy graphs, *Orion*, 6(1), 3 - 36 (1990).

[7] P.E. GILL, W. MURRAY, M.A. SAUNDERS, M.H. WRIGHT, A practical anti-cycling procedure for linearly constrained optimization, *Mathematical programming*, Series B, 45(3), 437 - 474 (1989).

[8] G. HADLEY, Linear programming, Addison Wesley, Massachusetts, (1978).

[9] P.J.M. HARRIS, Pivot selection methods of the devex LP code, *Mathematical programming study*, 4, 39 - 57 (1975).

[10] ICL, Accounting, Charging and budgeting, R00135/03, International Computers Limited, 1985.

[11] J.H. JANSE VAN VUUREN, Die invloed wat kolomseleksieprosedures op ontaarding en die werkverrigting van lineêre programmeringsalgoritmes het,

Dissertation (M.Sc.) - PU for CHE, Potchefstroom (1981).

[12] N. KARMARKAR, A new polynomial-time algorithm for linear programming, *Combinatorica*, 4(4), 373 - 389 (1984).

[13] L. KHACHIYAN, A polynomial algorithm in linear programming, *Dokl. akad. nauk SSSR*, 224, 1093 - 1096 (1979). (English translation in *Soviet math. dokl.*, 20, 191 -194 (1979)).

[14] A. KOZLOV, The Karmarkar algorithm, is it for real?, *Siam News*, 18(6), 1 - 14 (1985).

[15] J.H. NEL, Bydraes tot ry- en kolomseleksieprosedures vir ontaarde lineêre programme, Thesis (D.SC.) - PU for CHE, Potchefstroom (1986).

[16] D.M. RYAN, M.R. OSBORNE, On the solution of highly degenerate linear programmes, *Mathematical programming*, 41(3), 385 - 392 (1988).

[17] R.G.D. STEEL, J.H. TORRIE, Principles and procedures of statistics, second edition, McGraw-Hill, 187 - 188, Auckland, (1981).

[18] P. du B. WELMAN, Contributions to the theory and implementation of column selection and methods for systematic column generation for linear programming, Thesis (D.Sc.) - PU for CHE, Potchefstroom (1979).

[19] P. WOLFE, A technique for resolving degeneracy in linear programming, (in Recent advances in mathematical programming, Graves, R.L. & Wolfe, P. eds) McGraw-Hill, New York (1963).

[20] P. WOLFE, L. CUTLER, Experiments in linear programming (in Recent advances in mathematical programming, Graves, R.L. & Wolfe, P. eds) McGraw-Hill, New York (1963).