Mathematical Programming 5 (1973) 1–28. North-Holland Publishing Company

PIVOT SELECTION METHODS OF THE DEVEX LP CODE

Paula M.J. HARRIS
The British Petroleum Company Limited, London, Great Britain

Received 17 April 1972
Revised manuscript received 8 February 1973

Pivot column and row selection methods used by the Devex code since 1965 are published here for the first time. After a fresh look at the iteration process, the author introduces dynamic column weighting factors as a means of estimating gradients for the purpose of selecting a maximum gradient column. The consequent effect of this column selection on rounding error is observed. By allowing that a constraint may not be positioned so exactly as its precise representation in the computer would imply, a wider choice of pivot row is made available, so making room for a further selection criterion based on pivot size. Three examples are given of problems having between 2500 and 5000 rows, illustrating the overall time and iteration advantages over the standard simplex methods used today. The final illustration highlights why these standard methods take so many iterations. These algorithms were originally coded for the Atlas computer and were re-coded in 1969 for the Univac 1108.

1. Introduction

Devex, the word, comes from the Latin *devexus* – steep. It is referred to as archaic in the Shorter Oxford Dictionary but it appears also in Roget's Thesaurus, which should make it an acceptable English word. It is not a concocted word, though some have suggested that it DEVeloped from simplEX.

It has long been known that the standard simplex techniques for pivot selection are not ideal. At the very inception of linear programming, Dantzig [1] realised that the criterion of most negative reduced cost for selecting a new basic variable, chosen for computational ease, was not necessarily the best. He preferred the "greatest change" criterion, which was (and remains) impractical for any but the smallest problems.

Many other techniques have subsequently been suggested, notably the

"positive normalized" procedure of Dickson and Frederick [2] with its many variations. Computational experiments by Wolfe and Cutler [5] and Kuhn and Quandt [4] showed that both the greatest change and particularly the normalized procedures were indeed much superior to the criterion of most negative reduced cost.

However, all these improvements were devised using the tableau form simplex method and they had to be discarded as impractical when the product form simplex method superseded it.

The standard simplex technique for column selection has been described as taking a path of maximum gradient. On first studying L.P. with two- and three-dimensional models, one felt instinctively that this should mean the steepest path followed by one's finger as it traverses the edges of the model from the origin to the optimal solution vertex. This, however, is not the case. The Devex code attempts, by an approximation method, to find this steepest path.

The standard simplex technique for row selection works well when tackling by hand the simple examples used for demonstration. The two-dimensional diagram with which one illustrates one's early attempts at L.P. is, however, very different from the picture of things as "seen" by the computer. The pencil stroke of the illustration on a page represents the problem more realistically than the meticulous detail held within the machine. Consider a set of near concurrent lines (Fig. 1). The greater the number of binary places relied upon, the less likely are the near concurrent lines to "appear" concurrent, and the more iterations will be required to negotiate a corner. Simulating a pencil stroke obviates this.

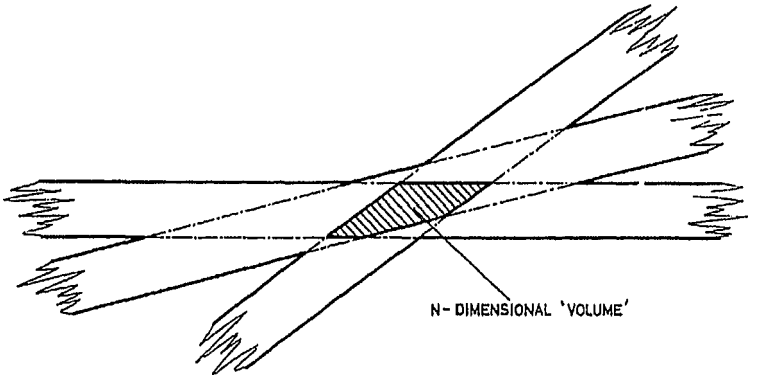


Fig. 1. Magnified intersection.

Since the user would be happy to have solutions obtained by graphical means (were that possible), it seems to the author that methods based on a graphical model should be more successful than methods which rely upon accuracy and which improperly treat the initial data as warranting that accuracy. The initial data are in part subject to uncertainty, but even where a fraction is known it cannot necessarily be expressed in a fixed punching field. Even should the decimal data be near correct, by the time it has been converted to binary much has been lost. These errors should not be treated as having a physical reality.

The constraints in a graphical model have breadth. Each solution vertex obtained is an n-dimensional volume. This is simulated in Devex by allowing a tolerance on all solution values.

Both row and column selection in Devex are thus developed from consideration of the practical model. It is most gratifying that these considerations have held good in the wider multi-dimensional space.

2. The iteration process

The author has an unusual view of the iteration process which she developed back in 1957 when disentangling the simplex method from a computer program, and believes that this clarifies the basis of the algorithm.

Since the algorithm requires, at certain iterations, that the out-of-basis variables be marked in some way for future reference, it is convenient for the description of the algorithm to refer to such an iteration as datum. No distinction is made between the row and column variables of the original problem.

The in-basis and out-of-basis variables at some datum iteration are renamed, using names R_i, $i = 1, ..., m$, for the m in-basis variables and C_j, $j = 1, ..., n$, for the n out-of-basis variables. The values of the R_i, in-basis variables, form the column C_0, called here the Quantity column.

To the set of m equations

$$a_{i0} = \sum_{j=1}^{n} a_{ij} C_j + R_i , \qquad i = 1, ..., m ,$$

are added the n temporary equations holding the non-basic variables at zero, $0 = -C_j$, $j = 1, ..., n$, making the matrix of coefficients at every iteration identical to its inverse (see Tableau 1).

Tableau 1

	C_1	C_2	C_3	C_4	C_5	R_1	R_2	R_3
C_1	−1							
C_2		−1						
C_3			−1					
C_4				−1				
C_5					−1			
R_1	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}	1		
R_2	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}		1	
R_3	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}			1

The iteration process is in three stages:

Stage 1: *Relax.* When the decision is made to increase the value of the variable C_q from zero and reduce the value of the variable R_p to zero, i.e., to replace C_q by R_p as a non-basic variable, the temporary constraint preventing such a move, $0 = -C_q$, is relaxed (see Tableau 2).

Tableau 2

	C_1	C_2	C_3	C_q	C_5	R_1	R_p	R_3	C_q from Tableau 1
C_1	−1								
C_2		−1							
C_3			−1						
C_q				0					−1
C_5					−1				
R_1	a_{11}	a_{12}	a_{13}	a_{1q}	a_{15}	1			a_{1q}
R_p	a_{p1}	a_{p2}	a_{p3}	a_{pq}	a_{p5}		1		a_{pq}
R_3	a_{31}	a_{32}	a_{33}	a_{3q}	a_{35}			1	a_{3q}

Stage 2. *Move.* Multiples α_j of the selected column C_q of Tableau 1 are subtracted from all the columns of the Tableau 2 and C_0, where $\alpha_j = a_{pj}/a_{pq}$ for all j, giving Tableau 3.

Tableau 3

	C_1	C_2	C_3	C_q	C_5	R_1	R_p	R_3
C_1	−1							
C_2		−1						
C_3			−1					
C_q	α_1	α_2	α_3	1	α_5		$1/a_{pq}$	
C_5					−1			
R_1	a'_{11}	a'_{12}	a'_{13}	0	a'_{15}	1	$-a_{1q}/a_{pq}$	
R_p	0	0	0	0	0		0	
R_3	a'_{31}	a'_{32}	a'_{33}	0	a'_{35}		$-a_{3q}/a_{pq}$	1

Stage 3: *Impose.* It only remains now to impose a temporary constraint on the new out-of-basis variable R_p, $0 = -R_p$, to make the set of equations once again define a solution vertex, making the quantity column C_0 contain the new solution values.

Tableau 4 shows the redistribution of coefficients one interation from the datum. Asterisks represent possible non-zero elements, both in the matrix of coefficients and in the quantity column C_0.

Tableau 4

	C_0	C_1	C_2	C_3	C_4	C_5	R_1	R_2	R_3
C_1		−1							
C_2			−1						
C_3				−1					
C_4	*	*	*	*	1	*		*	
C_5						−1			
R_1	*	*	*	*		*	1	*	
R_2								−1	
R_3	*	*	*	*		*		*	1

The iteration is now complete.

In general, the matrix of coefficients at some datum iteration as shown in Tableau 5 becomes as in Tableau 6 when temporary constraints on the column variables of E are replaced by temporary constraints on the row variables of E.

2.1. *Potential pivot columns*

Every column currently having a negative cost entry in the objective function is potentially a pivot column for the next iteration. Entries in the column represent the Quantity column changes that would take place on changing the value of the objective function z_0 by the amount z_j.

2.1.1. *Gradient.* The most negative z_j would make the greatest change to z_0 per unit of vector j, but changes in the other variables should be taken into account when considering gradient.

Let the subset β of the $m + n$ variables of the problem from an orthogonal framework in which distance, S_j say, can be measured, then we have

$$S_j = \sqrt{(\Sigma_{i \in \beta}\, a_{ij}^2)}$$

Tableau 5

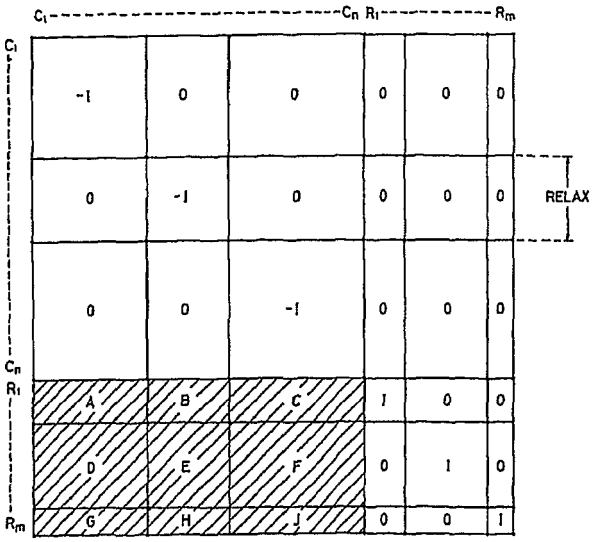
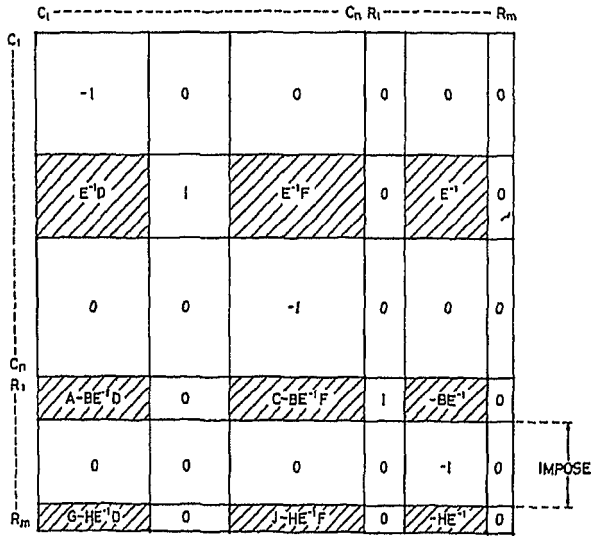


Tableau 6



from which the profit gradient can be calculated:

$$\text{GRADIENT} = -z_j/S_j \, .$$

Only when the current non-basic variables form the orthogonal framework will the most negative z_j necessarily give us the maximum gradient column.

Since standard simplex methods take the size of the negative cost row element z_j as directly indicating gradient, they can be regarded as measuring gradient in the framework of the current non-basic variables.

3. Pivot column selection

The author became aware of a fallacy in selecting negative cost row elements of large modulus when working with the primal integer-point problem associated with her mixed integer linear programming algorithm [3] in 1964.

3.1. The following integer example may illustrate the problem. X_1, X_2 and X_3 are the original integer variables, and Y_1, Y_2 and Y_3 are the non-negative integer variables introduced to create a vertex at an integer-point:

	Q	Y_1	Y_2	Y_3
Cost row Z		−1	−8	−2
X_1	3	−1	7	1
X_2	2	1	−10	−5
X_3	5	2	−18	3

As cost row elements become more "attractive" in size, they become generally less useful. A whole unit must be taken to have any effect at all, and the larger cost row elements are usually associated with larger elements elsewhere in the vector, preventing an ascent.

3.1.1. *Staircase analogy.* Steepness of the ascent path in the integer space is measured by dividing the RISE by the TREAD – using a staircase analogy. RISE represents the ascent if one whole unit of the column could be used, and TREAD represents the distance that would then be moved in the space of the original integer variables X_1, X_2 and X_3 – these being regarded as orthogonal.

In the above example we now have:

	Y_1	Y_2	Y_3
Rise	1	8	2
Tread	$\sqrt{(1^2+1^2+2^2)}$	$\sqrt{(7^2+10^2+18^2)}$	$\sqrt{(1^2+5^2+3^2)}$
Steepness	0.41	0.37	0.34

making Y_1 the most attractive column to consider, and in the event the most useful.

Feeling certain that this concept of steepness within a fixed framework could be equally useful in the solution of continuous L.P., the author wrote a simple program for very small L.P. requiring the complete tableau at each iteration and measuring gradient within the fixed framework of axes of the original out-of-basis variables.

3.1.2. *The steeper path.* The following example illustrates the change in the path taken when profit gradients are measured in the fixed framework of the original out-of-basis variables.

We are required to

$$\begin{aligned} &\text{maximize } Z = 2C_1 + C_2 + \tfrac{2}{3}C_3, \\ &\text{subject to } 2 \geqslant C_j \quad \text{for } j = 1, 2 \text{ and } 3, \\ &\qquad 1 \geqslant C_1 - C_2 - 2C_3, \\ &\qquad 11\tfrac{1}{3} \geqslant 5C_1 + C_2 - C_3, \\ &\qquad 17 \geqslant 5C_1 + 3C_2 + C_3. \end{aligned}$$

(Constraints not relevant to the path taken have been omitted.)

The iterations are displayed in conventional tableau form. TREAD has been calculated for all out-of-basis variables and GRADIENT for those having negative cost elements. See Fig. 2 for path comparison. In Tableaux 8, 10 and 12, which correspond with vertices A, C and E, the choice of pivot column is at variance with the standard method. In each case, when the column having most negative cost is used, two iterations are required to reach the next tableau position. Two sides of a triangle are traversed instead of one. The six pivots used to reach the optimal solution have been marked thus P.

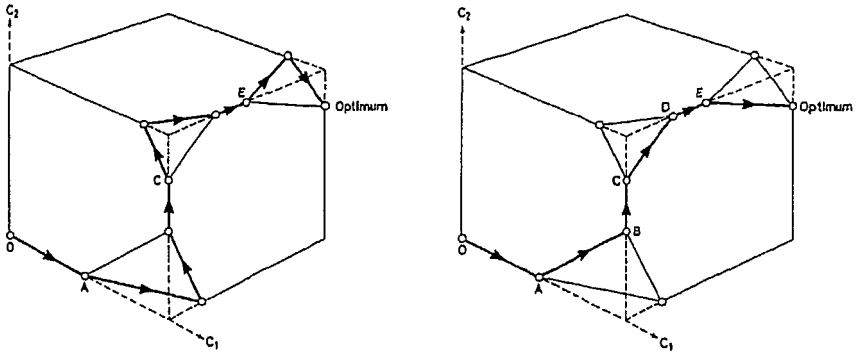


Fig. 2. Path comparison. (a) Changing framework. (b) Fixed framework.

Tableau 7

	Q	C_1	C_2	C_3
Z_0		−2	−1	$-\frac{2}{3}$
R_1	2	1		
R_2	2		1	
R_3	2			1
R_4	1	$\frac{1}{5}$	−1	−2
R_5	$11\frac{1}{3}$	$\frac{1}{5}$	1	−1
R_6	17	5	3	1
TREAD		1	1	1
GRADIENT		2	1	0.67

Tableau 8

	Q	R_4	C_2	C_3
Z_A	2	2	−3	$-4\frac{2}{3}$
R_1	1	−1	$\frac{1}{1}$	2
R_2	2			
R_3	2			1
C_1	1	1	−1	−2
R_5	$6\frac{1}{3}$	−5	6	9
R_6	12	−5	8	11
TREAD		1	$\sqrt{2}$	$\sqrt{5}$
GRADIENT			2.12	2.09

Tableau 9

	Q	R_4	R_1	C_3
Z_B	5	−1	3	$1\frac{1}{3}$
C_2	1	−1	1	2
R_2	1	1	−1	−2
R_3	2			1
C_1	2		1	
R_5	$\frac{1}{3}$	$\frac{1}{3}$	−6	−3
R_6	4	$\frac{1}{3}$	−8	−5
TREAD		1	$\sqrt{2}$	$\sqrt{5}$
GRADIENT		1		

Tableau 10

	Q	R_5	R_1	C_3
Z_C	$5\frac{1}{3}$	1	−3	$-1\frac{2}{3}$
C_2	$1\frac{1}{3}$	1	−5	−1
R_2	$\frac{2}{3}$	−1	5	$\frac{1}{1}$
R_3	2			
C_1	2		1	
R_4	$\frac{1}{3}$	1	−6	−3
R_6	3	−3	10	4
TREAD		1	$\sqrt{26}$	$\sqrt{2}$
GRADIENT			0.59	1.18

Tableau 11

	Q	R_5	R_1	R_2
Z_D	$6\frac{4}{9}$	$-\frac{2}{3}$	$5\frac{1}{3}$	$1\frac{2}{3}$
C_2	2			1
C_3	$\frac{2}{3}$	-1	5	1
R_3	$1\frac{1}{3}$	1	-5	-1
C_1	2		1	
R_4	$2\frac{1}{3}$	-2	9	3
R_6	$\frac{1}{3}$	$\underline{1}$	10	-4
TREAD		1	$\sqrt{26}$	$\sqrt{2}$
GRADIENT		0.67		

Tableau 12

	Q	R_6	R_1	R_2
Z_E	$6\frac{2}{3}$	$\frac{2}{3}$	$-1\frac{1}{3}$	-1
C_2	2			1
C_3	1	1	-5	-3
R_3	1	-1	5	$\underline{3}$
C_1	2		1	
R_4	3	2	-11	-5
R_5	$\frac{1}{3}$	1	-10	-4
TREAD		1	$\sqrt{26}$	$\sqrt{10}$
GRADIENT			0.26	0.32

3.1.3. *The ever changing framework.* When gradients are compared in the space of the current non-basic variables, the framework is changed at every iteration. Measured in the current framework, the preceding iteration may not be of good gradient. Measured in the framework of the preceding iteration, the current iteration may not be of good gradient. But which of these two reference frameworks is the more valid? If the earlier iteration was chosen for its steepness, then why change the definition of steepness and discount that move? If instead we continue to measure steepness in the earlier framework, our decisions remain valid.

It appears not to matter which reference framework is used so long as it is adhered to. To re-define steepness at every iteration is a continual process of discounting earlier decisions.

Kuhn and Quandt [4] experimented with different reference frameworks, but only when they used the space of all the variables of the problem was the framework not changed at each iteration. Unfortunately it is not practical to calculate gradients in a fixed framework when using the product form simplex method and this column selection technique was not pursued.

Results from the simple program using calculated gradients quickly confirmed the importance of the concept. Knowing from this that current selection methods were far from ideal, the author sought a practical compromise requiring only information readily available using the product form simplex method, to help in the assessment of steepness.

3.1.4. *Failure to indicate steepness.* Failure of the cost row elements to indicate steepness is most readily observable after the use of a small pivot. In the example which follows, the pivot row elements range from smaller to larger than the pivot. Equal initial costs have been used to show up the influence of pivot row element size on future pivot column choice. On adjusting the pivot row element sizes, by scaling their columns, to make no element larger than the pivot, we observe that the scaled cost row elements give a better indication of steepness. In this example, only the cost row and pivot row need be considered. The original out-of-basis variables $C_1, \ldots, C_8$ form the reference framework.

Initially all the columns are equally attractive.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_j	R_p
Cost row	−1	−1	−1	−1	−1	−1	−1	−1	−1	
Pivot row	1	$-\frac{1}{8}$	$-\frac{1}{4}$	$-\frac{1}{2}$	−1	−2	−4	−8	$-a_{pj}$	1

But one iteration later, when R_p has replaced C_1, we have

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_j	R_p
Cost row		$-\frac{9}{8}$	$-\frac{5}{4}$	$-\frac{3}{2}$	−2	−3	−5	−9	$\alpha_j - 1$	1
Old pivot row	1	$-\frac{1}{8}$	$-\frac{1}{4}$	$-\frac{1}{2}$	−1	−2	−4	−8	α_j	1

C_8 now appears to be the most useful column, but let us consider its usefulness in terms of gradient.

Using the staircase analogy we have

GRADIENT = RISE/TREAD

RISE = $1 - \alpha_j$, the increase in profit if $C_j = 1$.

TREAD = $\sqrt{(1 + \alpha_j^2)}$, the distance moved.

We cannot change the value of C_j by 1 without at the same time changing the value of C_1 by α_j (see Fig. 3).

We now have:

	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_j
RISE	$\frac{9}{8}$	$\frac{5}{4}$	$\frac{3}{2}$	2	3	5	9	$1 - \alpha_j$
TREAD	$\sqrt{(1+\frac{1}{64})}$	$\sqrt{(1+\frac{1}{16})}$	$\sqrt{(1+\frac{1}{4})}$	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{17}$	$\sqrt{65}$	$\sqrt{(1+\alpha_j^2)}$
GRADIENT	$9/\sqrt{65}$	$5/\sqrt{17}$	$3/\sqrt{5}$	$2/\sqrt{2}$	$3/\sqrt{5}$	$5/\sqrt{17}$	$9/\sqrt{65}$	$(1-\alpha_j)/\sqrt{(1+\alpha_j^2)}$

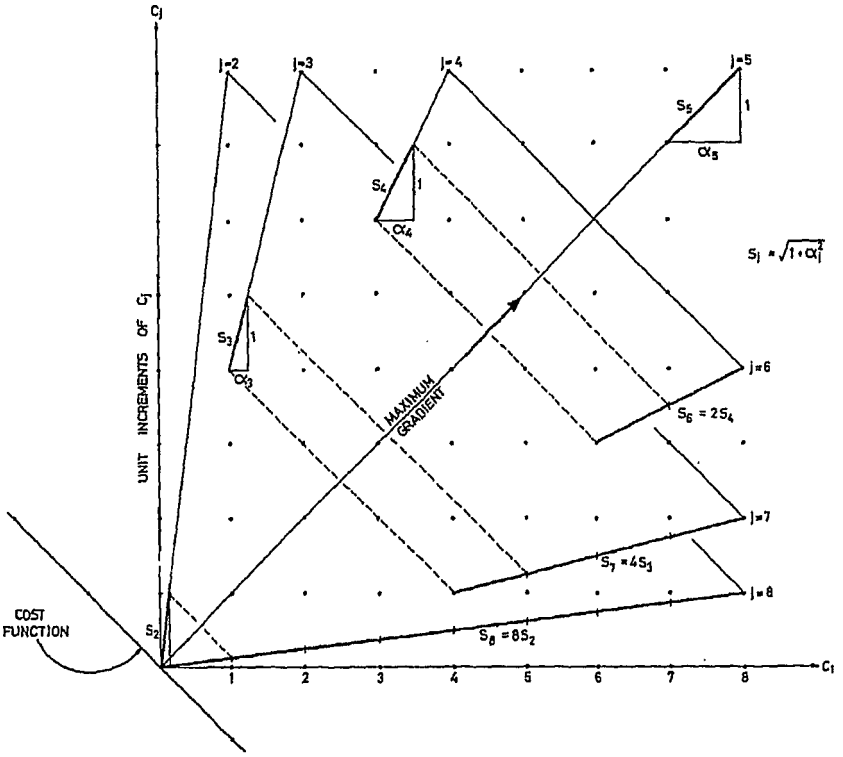


Fig. 3. The TREAD (S_j) effect.

As we would expect from considerations of the geometry of the problem, C_5 is the maximum gradient column. The cost row elements on columns C_6, C_7 and C_8, for which $|\alpha_j|$ exceeds unity, are most misleading as indicators of steepness. C_8, which appeared most useful when size of cost element alone was considered, is now shown to have the smallest gradient.

3.1.5. *Making the pivot element the largest in the row.* By using very rough approximations to TREAD as column scaling factors, the same

end is achieved as by actually calculating the gradients. For $|\alpha_j| \leqslant 1$, let $\sqrt{(1+\alpha_j^2)} \simeq 1$; for $|\alpha_j| \geqslant 1$, let $\sqrt{(1+\alpha_j^2)} \simeq |\alpha_j|$. The offending columns C_6, C_7 and C_8 in the example are thus scaled down so that no $|\alpha_j|$ exceeds unity, i.e., no element in the pivot row exceeds the pivot in magnitude:

	C_2	C_3	C_4	C_5	C_6	C_7	C_8
Scaling factor	1	1	1	1	2	4	8
Cost row	$-\frac{9}{8}$	$-\frac{5}{4}$	$-\frac{3}{2}$	-2	$-\frac{3}{2}$	$-\frac{5}{4}$	$-\frac{9}{8}$
Old pivot row	$-\frac{1}{8}$	$-\frac{1}{4}$	$-\frac{1}{2}$	-1	-1	-1	-1

The cost row elements of C_6, C_7 and C_8 are now the same as those of C_4, C_3 and C_2, respectively, making the cost elements correctly reflect steepness.

3.1.6. *The approximation to* TREAD. The algorithm used in the Devex code employs a mixture of these two techniques to obtain the TREAD of column j:

(i) accurately calculating the pivot column TREAD, $T_q = S_q$, from the subset of pivot column elements (see Section 3.2.1) ($j = q$);

(ii) scaling down variables, by increasing their TREAD to make the pivot element equal to the largest in the row ($j \neq q$).

These approximations T_j to TREAD are called weighting factors and are carried on all out-of-basis variables for application whenever the elements of a row are compared.

3.1.7. *Updating the weighting factors.* The old pivot row elements required for updating these weighting factors are extracted after the eta vector has been formed, and are therefore conveniently in pivot-divided form α_j. As each element α_j becomes available, both the weighting factor and the cost element of column j are updated, and the steepness of column j is assessed. The weighting factor T_j is replaced by $|\alpha_j| T_q$ unless T_j already exceeds this value; here $T_q = S_q$, the calculated pivot column TREAD. The cost row element z_j in the core-held cost row, is replaced by $z_j - \alpha_j z_q$, from which the steepness can be assessed:

$$\text{GRADIENT} \simeq \frac{-z_j + \alpha_j z_q}{\max(T_j, |\alpha_j| T_q)}.$$

3.1.8. *When to establish a new datum.* In practice, the ratio between the calculated and the estimated pivot column TREAD rarely exceeds 2.0, its most usual value lying between 0.7 and 1.3. In the current Devex code, values below 0.5 are taken as an indication of failure of the approximation although when failure does occur the ratio usually falls below 0.2. When failure occurs, a new reference framework is set up and all the weighting factors are reset to unity. This process is called RECTIFY in the Devex code. The new reference framework is made to agree initially with the old by choosing to rectify when the most negative cost column has an estimated gradient exceeding half the maximum.

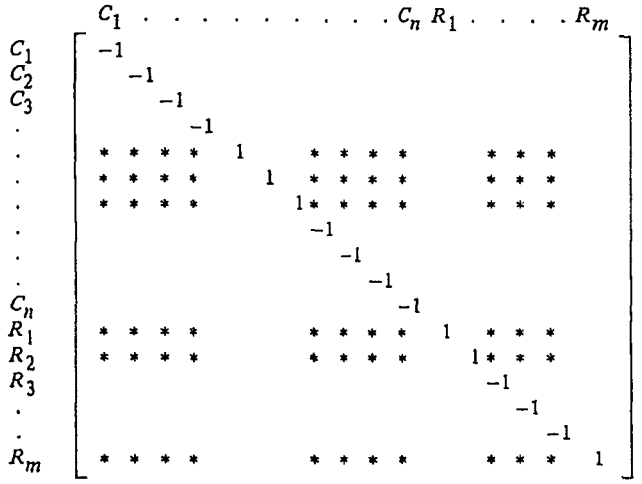
3.2. *Setting up the reference framework*

Let the current iteration be the datum. Then the renamed variables C_1 to C_n define the space in which gradients are in future to be compared (see Tableau 13).

Tableau 13a
The datum iteration tableau

	C_1	.	.	.	.	.	.	.	.	C_n	R_1	.	.	.	.	R_m
C_1	-1															
C_2		-1														
C_3			-1													
.				-1												
.					-1											
.						-1										
.							-1									
.								-1								
.									-1							
.										-1						
C_n											-1					
R_1	*	*	*	*	*	*	*	*	*	*	*	1				
.	*	*	*	*	*	*	*	*	*	*	*		1			
.	*	*	*	*	*	*	*	*	*	*	*			1		
.	*	*	*	*	*	*	*	*	*	*	*				1	
.	*	*	*	*	*	*	*	*	*	*	*					1
R_m	*	*	*	*	*	*	*	*	*	*	*					

Tableau 13b
Several iterations later



3.2.1. *Computational note.* The elements of the pivot column produced by the machine form a column of the reduced and shuffled tableau containing only the *** rows of Tableau 13b. It is therefore necessary to refer to markers, set at the last rectification on all out-of-basis variables, to determine whether a coefficient is required in the calculation of the TREAD $T_q (= S_q)$. If the pivot column itself carries a marker, then $(-1)^2$ must be added to the sum of squares, since the element -1 is missing from the machine produced column.

3.3. *Derivation of the approximation*

The first n rows of the $m + n$ rows of the complete Tableau 13b are occupied by the variables which were out of the basis when the reference framework was set up (Tableau 13a). At each iteration, multiples α_j of the pivot column q (see Section 2, Stage 2) are subtracted from all the columns of the tableau, where

$$\alpha_j = a_{pj}/a_{pq} \quad \text{for all } j . \tag{1}$$

To calculate the GRADIENT, the RISE, given by the negative of the control row element z_j, must be divided by the TREAD T_j defined as

$$T_j = \sqrt{(\Sigma_{i=1}^{n} a_{ij}^2)} . \tag{2}$$

But, since elements a_{ij} are not available, only an approximation to T_j can be used.

Assuming that T_j was a fair approximation for the TREAD at iteration r and that from the first n pivot column elements we have calculated

$$T_q = \sqrt{(\Sigma_{i=1}^{n} a_{iq}^2)} ,$$

then, for all j,

$$T_j' = \sqrt{(\Sigma_{i=1}^{n} a_{ij}'^2)} , \quad \text{where } a_{ij}' = a_{ij} - \alpha_j a_{iq} ,$$

becomes

$$T_j' = \sqrt{(\Sigma_{i=1}^{n} a_{ij}^2 + \alpha_j^2 \Sigma_{i=1}^{n} a_{iq}^2 - 2\Sigma_{i=1}^{n} a_{ij} [a_{ij} - a_{ij}'])} ,$$

from which we have

$$T_j' \simeq \sqrt{(T_j^2 + \alpha_j^2 T_q^2)} \qquad \text{for all } j \neq q ,$$

which roughly approximates to

$$T_j' = T_j \text{ or } |\alpha_j| T_q , \tag{3}$$

whichever is the larger.

With this approximation for T_j', we have

$$T_j' \geqslant |\alpha_j| T_q . \tag{4}$$

From (1) and (4) it follows that

$$|a_{pq}|/T_q \geqslant |a_{pj}|/T_j \qquad \text{for all } j ,$$

which is the very condition required to ensure that no element in the weighted pivot row shall exceed the weighted pivot in magnitude at iteration r.

No exception need be made in the approximation of the TREAD T_p of the out-going variable. Using the complete tableau, we see at once that

$$T_p' = |\alpha_p| T_q .$$

Using the more conventional reduced tableau it becomes necessary to introduce constants, k_p and k_q say, which take the values 1 or 0 depending upon whether the pivot row p and the pivot column q, respectively, belong to variables of the reference framework or not.

If the first s rows hold the variables of the reference framework, we have

$$T_q = \sqrt{(\Sigma_{i=1}^{s} a_{iq}^2 + k_p\, a_{pq}^2 + k_q)}\,, \qquad i \neq p\,,$$

and after the iteration, we have

$$T'_p = \sqrt{(\Sigma_{i=1}^{s} a_{iq}^2/a_{pq}^2 + k_q/a_{pq}^2 + k_p)}\,, \qquad i \neq p\,,$$

from which it follows that

$$T'_p = T_q/a_{pq} \quad \text{or} \quad T'_p = |\alpha_p|\, T_q\,,$$

so that the approximation formula (3) gives in this instance, where $j = p$, the exact value required.

It has been found in practice that it is inadvisable to have weighting factors less than unity, which might arise when a variable leaves the basis using a large pivot. To obviate this, in-basis variables are given unit weight, $T'_q = 1$, so that $T'_p = 1$ or $|\alpha_p|\, T_q$, whichever is the larger.

3.3.1. *The need to* RECTIFY. Since by this process weights increase from unity and are never decreased, only underestimates of weight are self-correcting. When the weights become too large, it is necessary to set up a new reference framework. Delay in rectifying causes standard simplex type gradients since T'_p is probably the only corrected weight.

3.4. *Concerning standard simplex*

3.4.1. *Rounding error.* Dickson and Frederick [2] mention the increase in digital accuracy that they achieved with the improved column selection technique, but supposed this to be due to the reduction in the number of iterations – a major factor in the days prior to the product-form simplex method. It would appear now that a still more important factor is the tendency not to accumulate rounding error when the snowball effect, inherent in the standard simplex technique, has been removed.

A large weighting factor on a column is an indication that a large multiple of a previous pivot column has contributed to its composition. If a column having an unattractive gradient is selected by the standard simplex method, then it carries a large multiple of the rounding errors from a previous iteration. If a multiple of this column, greater than one, is sufficient to cause the next pivot column to be selected, then this next pivot column will carry a greater multiple of these errors. Should this happen successively, then the rounding errors will rapidly build up.

3.4.2. *Large product of pivots.* Finding large pivots in these inflated columns may temporarily halt the "snowball" but will unfortunately contribute substantially to the monstrous product of pivots (see Fig. 7b). Since the same product of pivots must be achieved, no matter what route is taken to the final solution basis, any tendency to force the use of large pivots must increase the number of iterations.

3.4.3. *Partial pricing.* When only a small section of the matrix is priced at each iteration using standard simplex, it has been observed that the iterations are more effective. This observation gives yet further confirmation of the importance of not seeking out negative costs of large modulus. The iteration reduction when using partial pricing is small compared with the reduction obtained by using Devex (see Figs. 5 and 6 for plots of Devex compared with Ilona standard simplex which uses partial pricing.)

4. Primal/dual algorithms

Whatever has been said here about column weighting factors T_j in the primal simplex algorithm will apply equally to row weighting factors D_i in the dual algorithm. But, if both row and column weighting factors are carried, then row selection in the primal and column selection in the dual may also be improved. Whenever in the primal algorithm a choice has to be made between rows making an equal ascent, this can be better made by considering steepness in the dual space, and vice versa.

Selecting, from those rows for which a_{i0}/a_{iq} is constant, the row p for which the dual gradient is greatest,

$$a_{p0}/D_p = \max a_{i0}/D_i \,,$$

gives the largest suitable pivot a_{pq}/D_p in the weighted pivot column as well as a pivot which compares favourably with other elements in its row.

In large L.P. problems, the number of rows competing for equal ascent is larger than most L.P. codes will allow. The standard simplex technique requires the deletion of small quantities a_{i0} in order to widen the pivot row choice when making zero ascents. Unfortunately, deleting small positive elements here makes the constraints more restrictive, often with disastrous results. The constraints spring back into place at the next re-solve (invert) carrying the current solution vertex with them. Deleting small negative quantities is much less disastrous, and leads to a wider and better pivot selection. The creation of near zeros, when using the standard simplex method of row selection, is an indication that these rows have been excluded unnecessarily from the selection. Once near zeros have been created, the number of near-zero ascents that ensue is an indication of weakness in the pivot row selection technique.

Linear programming is a practical technique and not a mathematical exercise. Negative quantities which are too small to be printed on the solution output (printed to three decimal places, say) do not exist any more than do the small positive quantities which are not printed, and should therefore be equally tolerated throughout computation.

5. Pivot row selection

Fig. 4(a) illustrates the increase in permitted movement from zero of the selected variable C_q when negative quantities down to $-\delta$ are tolerated. The constraint imposed by row i on the value of C_q has shifted a distance δ/a_{iq} from the simplex barrier which permits no negative quantities. This shift is inversely proportional to the pivot column element in row i.

Each constraint illustrated in Fig. 4(b) would, if used as pivot row, increase C_q within the limits imposed by the other constraints. R_3 would be selected as pivot row using standard simplex. R_5 imposes the most severe limit on C_q-increase when negative quantities down to $-\delta$ are tolerated. R_6 would be selected by an algorithm which chose the largest of all these possible pivots.

5.0.1. *Pivot tolerance.* It is often the case when the standard simplex technique is used, that an ascent is prevented by a small element a_{iq}

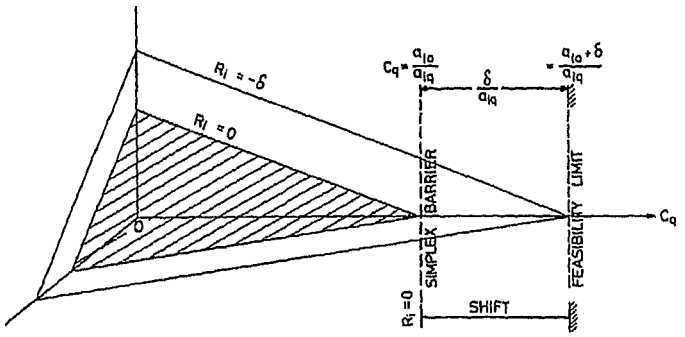


Fig. 4(a).

which may even be too small to be considered as a pivot. The shift can then be sufficiently large to include good sized pivots making useful ascents. Because Devex exploits this shift, smaller pivot tolerance can be used with impunity. The pivots used are, in practice, rarely either very large or very small.

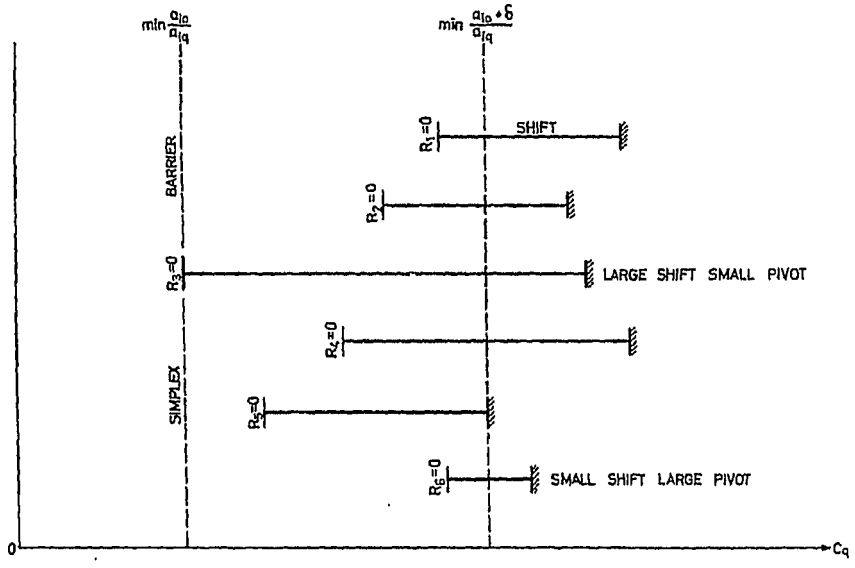


Fig. 4(b).

5.1. *The row selection rule*

The usual simplex restriction

$$a'_{i0} \geqslant 0 \quad \text{for all } i\,,$$

may be relaxed to

$$a'_{i0} \geqslant -\delta \quad \text{for all } i\,.$$

The rule for pivot row selection now becomes

$$\frac{a_{p0}}{a_{pq}} \leqslant \frac{a_{i0} + \delta}{a_{iq}} \quad \text{for all } i\,.$$

Only when there exists a row p for which

$$\frac{a_{p0} + \delta}{a_{pq}} < \frac{a_{i0}}{a_{iq}} \quad \text{for all } i \neq p\,,$$

will this selection rule uniquely define p. In a code which does not carry row weighting factors, the largest of the possible pivots should be chosen.

Two complete passes through the pivot column are required. In large problems, row p is rarely uniquely defined.

Pass 1. To find the limit L, given by

$$L = \min_i \{(a_{i0} + \delta)/a_{iq}\} \quad \text{for } a_{i0} \geqslant -\delta \text{ and } a_{iq} > \text{tol}\,.$$

Pass 2. To find row p for which

$$a_{pq} = \max_s \{a_{sq}\}\,,$$

where

$$a_{s0}/a_{sq} \leqslant L \quad \text{for } a_{s0} \geqslant -\delta \text{ and } a_{sq} > \text{tol}\,.$$

If the quantity a_{p0} on the chosen pivot row is negative, then this is reset to zero and a zero ascent is made. Whenever this selection technique does not select the same row as the standard simplex technique, a larger ascent is made using a better sized pivot.

5.2. *Row weighting factors in the primal*

Although the Devex algorithms include the use of row weighting factors, these are not yet available in the Univac 1108 code. Work is now proceeding to introduce these dual weighting factors and a further reduction in the number of iterations of up to 20% or 30% is expected, depending on the structure and degeneracy of the problem.

The TREAD D_i in the space of the dual variables is defined as follows:

$$D_i = \sqrt{(\Sigma_{j=n+1}^{m+n} a_{ij}^2)} .$$

The estimates of TREAD are updated by the elements of the pivot column and the current estimate of the pivot row TREAD D_p using the same argument as for column weighting factors, thus $D_i' = D_i$ or $|a_{iq}/a_{pq}|D_p$, whichever is the larger. The TREAD D_p of the pivot row variable is not calculated from the pivot row elements because these are not available at the time. The row weighting factors are reset to unity and a new dual reference framework is set up whenever RECTIFY is dictated by the column weighting factors.

These estimates of TREAD are only a guide and are used by modifying the row selection rule after pass 1, making:

Pass 2. To find the row p for which

$$a_{pq} = \max_r \{a_{rq}\} ,$$

where

$$a_{rq}/D_r \geqslant \max_s \{a_{sq}/D_s\} - 0.00005 ,$$

$$a_{s0}/a_{sq} \leqslant L , \qquad \text{for } a_{s0} \geqslant -\delta \text{ and } a_{sq} > \text{tol} .$$

By this selection rule, the chance of selecting a pivot which is small for its row is reduced. The rule reverts to the largest possible pivot rule when treads become large or possible pivots become small.

5.3. *Column weighting factors in the dual*

The Atlas Devex program carries both row and column weighting factors and uses them for row and column selection both in the primal and in the dual.

In the dual, the TREAD T_q of the pivot column variable is not calculated because the elements of the pivot column are not available at the time of updating the column weights. Instead, the TREAD D_p of the

pivot row variable is calculated and this replaces the estimated value before updating the row weighting factors. Whenever a dual RECTIFY is dictated by failure of this estimate to approximate to TREAD, the column weighting factors also are reset to unity and a new primal reference framework is set up.

Experience on Atlas with the dual Devex algorithms, using both row and column weighting factors for pivot selection, indicates that the reduction of iterations in the dual is even greater than the reduction achieved in the primal. Until these algorithms are available for solving larger problems than Atlas can handle, the full extent of this economy will not be known.

6. Conclusions

Success with large problems using only single length arithmetic and storage has proved these techniques to be invaluable in getting fast, practical solutions. The solution is printed to three decimal places to omit the tolerated negative values. In many instances, even this may be more places than the user requires. Great accuracy in the solution values is certainly not required by the user, and concentration on accuracy during compute would seem to be the cause of many niggling basis changes en route. The most significant improvement in number of iterations occurs in the tail, in which the standard simplex method appears loath to terminate. As the problem size grows, it becomes increasingly important for the user to supply good feasible starting bases; hence the computation is driven further into the tail. The algorithms of the Devex code take advantage of this and demonstrate considerable reductions in the number of iterations and compute times required.

Appendix

These algorithms, devised by the author over the years from 1957, have been used since 1965 in production programs. Experience with these algorithms is now extensive and continues to demonstrate the superiority of the Devex code.

The current Devex code uses single length, 27 bit mantissa, on the Univac 1108, on problems of up to 7 000 rows. The negative quantity tolerance used is $\delta = 0.0005$. This tolerance is the same for all variables.

It is left to the user to take advantage of this tolerance by scaling his variables so that the three decimal place printed solution gives him the precision that his problem warrants.

A.1. *Current program performances*

The following figures, not corrected for machine speeds, give an indication of current program performances on a problem of size 2 345 X 5 167 having 47 126 non-zero elements, started from a given basis.

Machine	Program	Minutes in dedicated machine	Eta formations
UNIVAC 1108	ILONA 8.4	120	1 821
UNIVAC 1108	DEVEX	68	819
IBM 370/165	MPSX	59	2 667

A.2. *The three plots*

Three plots of profit against iteration number illustrate typical patterns for comparison with the standard simplex methods used today (see Figs. 5, 6, 7).

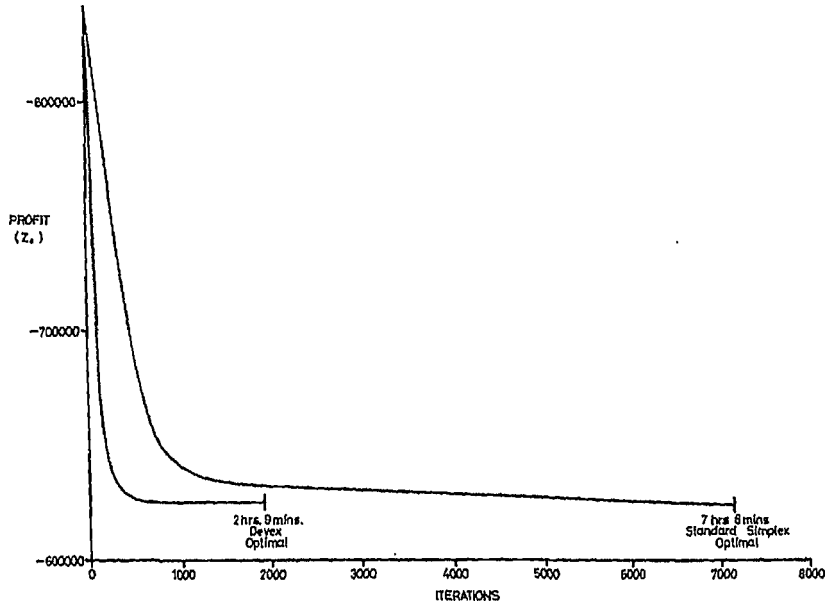


Fig. 5. Problem 1: a very long tail (2549 X 5415)

(1) A very long tail. (The tail is long even with Devex.)
(2) A usual Devex tail. (Time was not available to pursue the standard simplex tail, estimated at 40 hours.)
(3) A tail tip. (A large job, 37 Devex iterations from optimum.)

A.3. *Directly comparable*

All the runs took place on the same machine, the Univac 1108. All three problems started from good feasible starting bases, thus avoiding the comparison of different methods of reaching a first feasible solution.

A.4. *Summary of runs*

Problem	Matrix size	Devex	Standard simplex
1	2549 X 5415	1920 iterations 2 hours 9 minutes	7112 iterations 7 hours 6 minutes
2	2774 X 5725	5037 iterations 6 hours	(30000 iterations) (40 hours)
3	4713 X 10146	37 iterations 6 minutes	93 iterations 11 minutes

The standard simplex program used on Problems 1 and 2 used double pricing and partial scan. This program was unable to handle the size of Problem 3. Problem 3 was run on a version of the Devex program which ignored the weighting factors, to simulate standard simplex. Figures in brackets were estimated from previous runs on similar models, and appear to agree with the progress being made at $10\frac{3}{4}$ hours (see Fig. 6).

A.5. *Iteration rates*

The simulated standard simplex run on Problem 3 is of particular interest because it demonstrates the effect of weighting factors alone. When the weighting factors were ignored, not only were considerably more iterations required but each iteration was slower due to an increase in the length of the eta vectors, taking on average 4.4 seconds instead of 3. These rates exclude the inverts which accounted for approximately 4 minutes of each run.

A.6. *Eta vector length*

The average length of an eta vector rose from 1500 non-zeros using

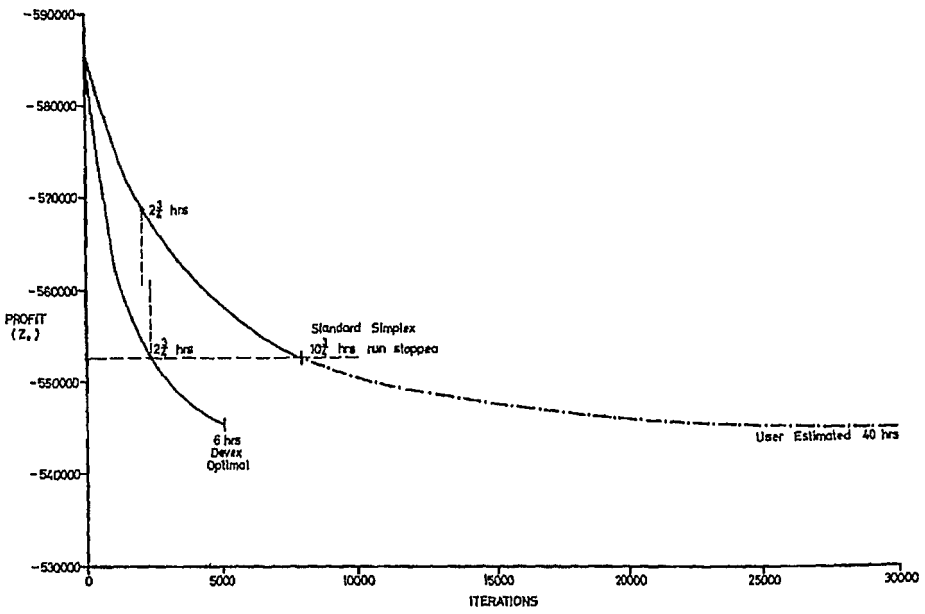


Fig. 6. Problem 2: a usual devex tail (2774 × 5725; 55483 non-zeros).

Devex to 2200 non-zeros when the weighting factors were ignored. Some of these were spurious small elements due to the rounding errors mentioned earlier, but many were due to an increased interaction between adjacent iterations. This augers well for the use of multiple pricing with Devex. Columns chosen for steepness seem less likely to be interdependent, and the use of one vector should not so frequently undermine the usefulness of another. Double pricing in Devex can be implemented easily by extracting the pair of pivot rows between the writing of the two etas.

4.7. *Iteration effectiveness*

The increased interaction between adjacent iterations becomes more explicable when we consider the unnecessarily tortuous path of basis changes and the number of variables unnecessarily involved (see Fig. 7(c)). By the 45th iteration, 36 different variables have entered the basis, and yet we see from the plot that by then we are further from the optimal basis than we were at the start, and further from the starting basis than we shall be at the finish. Fig. 7(c) together with Fig. 7(b)

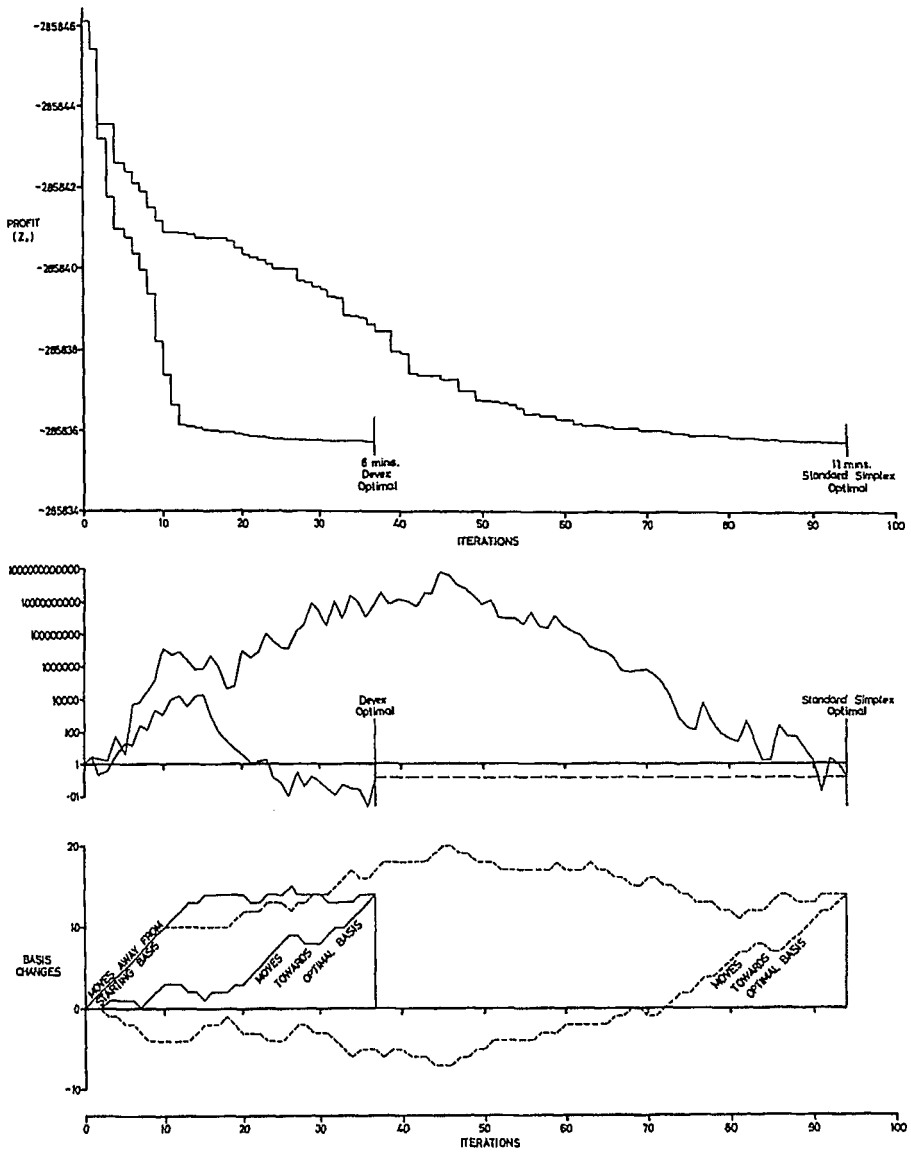


Fig. 7. Problem 3: a tail tip (4713 X 10146; 90274 non-zeros.)
(a) Movement of the profit function. (b) Product of pivots from the starting basis to the optimal basis. (c) Differences from starting and optimal bases.

gives us an insight into how the standard method manages to take so many iterations.

The monstrous product of pivots is caused by the column selection technique seeking out columns in which the elements tend to be large. Since it is important that a pivot element should not be small for its column, finding compensatory small pivots is both difficult and undesirable when presented with inflated columns. The fight to bring down this product of pivots continues throughout the run. During the 94 simplex iterations, 73 different variables enter the basis. By iteration 72, the basis is no closer to the optimal basis than it was at iteration 0, and is closer to the starting basis, by iteration 81, than at any iteration since iteration 19. In contrast, during the 37 Devex iterations, 35 different variables enter the basis and never is the basis unnecessarily far from both the starting and optimal bases.

Acknowledgments

The author wishes to thank the Chairman and Directors of The British Petroleum Company Limited for permission to publish this paper, and J. Tomlin of Scientific Control Systems Ltd. for his encouragement and helpful comments during the writing of this paper.

References

[1] G.B. Dantzig, "Minimization of a linear function of variables subject to linear inequalities", in: *Activity analysis of production and allocation*, Ed. T.C. Koopmans (Wiley, New York, 1951) 339–347.

[2] J.C. Dickson and F.P. Frederick, "A decision rule for improved efficiency in solving linear programming problems with the simplex algorithm", *Communications of the Association for Computing Machinery* 3 (1960) 509–512.

[3] P.M.J. Harris, "An algorithm for solving mixed integer linear programmes", *Operational Research Quarterly* 15 (2) (1964) 117–132.

[4] H.W. Kuhn and R.E. Quandt, "An experimental study of the simplex method", in: *Proceedings of Symposia in Applied Mathematics* 15 (American Mathematical Society, Providence, R.I., 1963) 107–124.

[5] P. Wolfe and L. Cutler, "Experiments in linear programming", in: *Recent advances in mathematical programming*, Eds. R. Graves and P. Wolfe (McGraw-Hill, New York, 1963).