Anti-cycling lexicografic simplex algorithm

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics

PROBABILITY THEORY, STATISTICS AND OPERATIONS RESEARCH GROUP

Memorandum-COSOR 78-02

Anti-cycling lexicografic simplex algorithm

by

J.F. Benders

Eindhoven, January 1978

The Netherlands

Anti-cycling lexicografic simplex algorithms

by

J.F. Benders

Summary. The main subject of this paper is to derive some insight in the relation between the lexicografic simplex algorithm that employs a lexicografic nonnegative righthand side matrix of size (m,m+1) and Wolfe's anti-cycling algorithm that uses apart from the right hand vector b only one extra column.

0. Introduction. Although operational linear programming programs in general do not contain any precautions for avoiding cycling, it is common opinion that cycling never occurs when using them for solving practical problems. When solving linear complementarity problems by the Lemke-Howson algorithm, which uses also the simplex transformation, the possibility of cycling in case of degenerated problems may not be neglected. In [1] the Lemke-Howson algorithm is treated in a lexicografic fashion quite similar to the lexicografic simplex algorithm [ ]. This treatment however needs an extension of the right hand side column b to an (m,m+1) right hand side matrix. Recently [3] the question has been posed whether Wolfe's anti-cycling algorithm for linear programming problems [4], which uses only one extra column instead of an extra (m,m) matrix can be applied for avoiding cycling in the Lemke-Howson algorithm. As a contribution to answering this question it is shown in this paper that Wolfe's anti-cycling algorithm is actually an economized version of a generalized lexicografic simplex method.
For completeness in section 1 the usual lexicografic simplex algorithm is developed, followed in section 2 by an other lexicografic algorithm that is more general in the way the right hand side matrix is constructed.
Finally, is section 3 Wolfe's anti-cycling algorithm is shown to be equivalent to this generalized lexicografic algorithm.

1. The lexicografic simplex algorithm. The lexicografic simplex algorithm uses a simplex tableau in which the initial right hand side vector b is replaced by an (m,m+1) right hand side matrix P. The columns of P will be denoted by $p_{.1},\ldots,p_{.m+1}$, the rows by $p_{1.},\ldots,p_{m.}$. The matrix P must be chosen such that

1) $p_{.1} := b$.
2) its rows are linearly independent.
3) its rows are lexicografically positive.

The "value coefficient" d_0 is replaced by an $(m+1)$-value vector p_0, defined by

$$p_{01} := c_E'b, \ \forall_{i=2,\ldots,m+1}, \ p_{0i} := c_E'p_{.i}$$

where c_E is that part of the objective vector that belongs to the initial unit basis. Given a pivot a_{rs}, the simplex tableau including the $m+1$ right hand side columns is transformed in the usual way.
In each iteration the pivot column $a_{.s}$ is chosen such that $d_s < 0$. If such a column is not present, the tableau is optimal.
The pivot row is chosen in such a way that after the simplex transformation the right hand side rows remain lexicografically positive, i.e. r is taken such that

$$\frac{p_{r.}}{a_{rs}} = \text{minlex}\{\frac{p_{i.}}{a_{is}} \mid a_{is} > 0\} .$$

If such an index r is not present, the problem has an infinite solution.
Now the following properties hold:

1) the right hand side rows remain linearly independent; for the right hand side matrix is multiplied in each iteration by a non singular matrix. As a consequence, there is never an all zero right hand side column.
2) the right hand side rows remain lexicografically positive; for by the rule for chosing the pivot row the right hand side rows after each transformation are lexicografically nonnegative. By the preceding property however also nonzero.

A simplex transformation with pivot a_{rs} transforms the value vector by

$$\bar{p}_{0.} := p_{0.} - \frac{d_s}{a_{rs}} p_{r.}$$

hence, since $d_s < 0$ and $p_{r.} >$ lex 0,

$$\bar{p}_{0.} > \text{lex } p_{0.} .$$

It follows that the value vector increases lexicografically in each iteration.
If B_t is the basis in the t-th iteration, clearly

$$(p_{0.}^t)' = c_B'B_t^{-1}P ,$$

Thus, p_0^t is completely determined by the basis B_t, and it follows from the lexicografic increase of $p_{0.}^t$ in each iteration that no basis can reappear when applying this lexicografic simplex algorithm. This proves that this algorithm always ends in a finite number of steps (there are only a finite number of bases) with the conclusion that the problem at hand has m infinite solution, or with an optimum solution if one exists.

Remark. A convenient choice for the initial right hand side matrix P is the matrix (b,E) where E is the (m,m) unit matrix. Obviously, after the t-th iteration, governed by the basis B_t, the right hand side matrix in updated form becomes $(B^{-1}b,B^{-1})$. Hence all information required is contained in the updated right hand side and in the inverse basismatrix. It follows that in case a straightforward or an explicit inverse algorithm is used, the addition of an extra right hand side matrix is not necessary.

2. A generalized lexicografic simplex algorithm. The generalized lexicografic simplex algorithm presented here is also performed in a simplex tableau with a right hand side matrix P. However, contrary to the matrix P in the preceding algorithm, now it depends on the iteration number. In fact, only the first column of P is known in advance: $p_{.1} = b$ all other columns being defined and sometimes redefined in the course of the operations. The number of columns of P is even not fixed; it may change from iteration to iteration but will never exceed m + 1. For ease of presentation, this number will be taken exactly equal to m + 1. In the initial simplex tableau the first column is taken to be equal to b, all others are zero columns. Again the value coefficient d_0 is replaced by a value vector $p_{0.}$ such that

$$p_{01} := c_E'b; \ \forall_{i=2,\ldots,m+1} \ p_{0i} = 0$$

where c_E is that part of the objective vector that corresponds to the initial basis.

Given a pivot a_{rs} the simplex tableau including the m + 1 right hand side columns are transformed in the usual way. In each iteration the pivot column $a_{.s}$ is chosen such that $d_s < 0$. If such a column is not present the tableau is optimal.

In the course of the operations the row i will be marked by a nonnegative integer marker k_i. When searching for the pivot row index only those rows i are taken into account for which k_i is maximum. This maximum value k may alter in each iteration; it indicates the column $p_{.k}$ of the right hand side matrix which is used in the actual iteration for the selection of the pivot row index. Before starting the algorithm all markers k_i and also their maximum value k are set equal to 1.

Now assume that the t-th iteration has been performed and that B_t is the actual basis matrix. Then give a pivot column $a_{.s}$ the search for the pivot row starts by inspection of the k-th right hand side column in the updated simplex tableau. Three situations may occur

1) $\forall_{i,k_i=k}\ a_{is} \leq 0$.
   Then if k = 1, the algorithm stops; the problem has an infinite solution; if k > 1, the pivot row choice must be based on the preceding right hand side column; hence $\forall_{i,k_i=k}\ k_i := k_i - 1$ and $k := k - 1$.

2) there is an index r such that $k_r = k$ and

$$0 < \frac{p_{rk}}{a_{rs}} = \min\left\{\frac{p_{ik}}{a_{is}} \mid a_{is} > 0,\ k_i = k\right\} .$$

   Then a_{rs} is the pivot element; all markers remain unchanged.

3) there is an index i, such that

$$k_i = k,\ a_{is} > 0,\ p_{ik} = 0 .$$

   Then the pivot row choice must be based on the next right hand side column which must now be defined. The column $p_{.k+1}$ may be chosen arbitrarily apart from the fact that $\forall_{i,k_i=k}\ p_{i,k+1} \geq 0$, but not all equal to 0. For case of presentation, here $p_{.k+1}$ is defined by $\forall_{i,k_i<k}\ p_{i,k+1} = 0$, $\forall_{i,k_i=k}\ p_{i,k+1} = 1$ and $p_{0,k+1} = c_B' p_{.k+1}$. Moreover, $\forall_{i,k_i=k}\ k_i := k_i + 1$ and $k := k + 1$. This definition of $p_{.k+1}$ has the advantage that a pivot row choice now based on $p_{.k+1}$ leads to simplex transformation with pivot a_{rs} such that the value of $p_{0,k+1}$ increases by the positive amount $-(p_{rk}/a_{rs})d_s$.

The elements p_{0i}, $i \leq k+1$ remain unchanged. It follows that the value vector $p^t_{0.}$ increases lexicografically in each iteration.

The structure of the updated right hand side matrix is shown in figure 1.

It will be shown that this algorithm is finite, i.e. that it ends in a finite number of steps either with an indication that the problem has an infinite solution or with an optimum solution. As in the case of the usual simplex algorithm, the prove is completed if it can be shown that no basis can be met more then once during the simplex operations.

The right hand side columns, when introduced, are defined above in terms of the updated simplex tableau. It is more convenient however to define them in terms of the initial tableau:
If $p_{.k}$ is introduced in a simplex tableau characterized by the basis B, then this k-th column is reflected in the initial tableau by $Bp_{.k}$. Now, assume that during the t-th iteration, k right hand side columns are present. If the τ-th right hand side column is introduced in a simplex tableau characterized by a basis B_τ, then the initial right hand side matrix would have been

$$P^\tau := (b, B_1 p_{.1}, \ldots, B_\tau p_\tau, \ldots, B_k p_{.k}, 0, 0) .$$

Is B_t the actual basis, then the t-th value vector is

$$c'_{B_t} B_t^{-1} P^t = (c'_{B_t} B_t^{-1} b, c'_{B_t} B_t^{-1} B_1 p_{.1}, \ldots, c'_{B_t} B_t^{-1} B_k p_{.k}, 0, 0) .$$

Let the basis B_n reappear after v iterations, i.e.

$$B_{u+v} = B_u \quad \text{for some } v \geq 2 .$$

Then consider the sequence of intermediate bases

$$B_u, B_{u+1}, \ldots, B_{u+v} ,$$

and of intermediate right hand sides

$$P^u, P^{u+1}, \ldots, P^{u+v} .$$

The algorithm is such that value vector increases lexicografically in each iteration, i.e.

$$\forall_{\tau=0,\ldots,v-1} \; c'_{u+\tau} B^{-1}_{u+\tau} P^{u+\tau} < \text{lex } c'_{u+\tau+1} B^{-1}_{u+\tau+1} P^{u+\tau+1} \; .$$

All intermediate right hand side matrices have the same first column b. Suppose their common part consists of the first τ columns

$$b, B_1 p_{.1}, \ldots, B_\tau p_{.\tau} \; .$$

Clearly $1 \leq \tau \leq k$ where k is the number of nonzero right hand side columns in P^u.

If $\tau \leq k - 1$, there is at least one iteration $u + \rho$, $1 \leq \rho \leq v$ in which the pivot row selection was based on the right hand side column $B_\tau p_{.\tau}$. Hence in that iteration the corresponding component $p^\tau_{0,\tau}$ of the value vector $p^\tau_{0.1}$ has been increased. This, however, would mean that

$$c'_{B_u} B^{-1}_u B_\tau p_{.\tau} < c'_{B_{u+p}} B^{-1}_{u+\rho} B_\tau p_{.\tau} \leq c'_{B_{u+v}} B^{-1}_{u+v} B_\tau p_{.\tau}$$

which is impossible if $B_{u+v} = B_u$.

If $\tau = k$ and $B_{u+v} = B_u$ then the simplex tableau up to the first k columns of the right hand side matrix at the end of iteration $u+v$ is identical to that at the end of iteration u. Also $B^{-1}_{u+v} B_{k+1} p_{.k+1}$. Hence, when chosing the pivot row in the $(u+v)$-th iteration one was confronted with exactly the same situation as when chosing the pivot row in the u-th iteration. Therefore, this is based either on the k-th or on the $(k+1)$th right hand side column, resulting in an increase of the corresponding component of the, valuation vector; i.e.

$$P^{u+v}_{0,k} = c'_{B_{u+v}} B^{-1}_{u+v} B_k p_{.k} > c'_{B_u} B^{-1}_u B_k p_{.k}$$

or

$$P^{u+v}_{0,k+1} = c'_{B_{u+v}} B^{-1}_{u+v} B_{k+1}, p_{.k+1} > c'_{B_{u+1}} B^{-1}_{u+1} B_{k+1} p_{.k+1}$$

which again is impossible if $B_{u+v} = B_u$.

The contradictions obtained prove that reappearance of a basis B_u when applying the algorithm, is impossible.

3. The relation between Wolfe's anti-cycling algorithm and the generalized lexicografic simplex algorithm. Considering the generalized lexicografic algorithm one observes that in each iteration, both for the pivot choice and for updating the tableau, either a right hand side column $p_{.k}$ is not relevant, hence set equal to zero, or only those elements p_{ik} of $p_{.k}$ are of importance for which $p_{i,k-1} = 0$. Noting, now that $p_{i,k-1} = 0$ implies $\forall_{\tau \leq k-1}\ p_{i,k-\tau} = 0$ it follows that all relevant information concerning the right hand sides may be stored in one m-column, provided for each row a marker is used, saying to which right hand side the i-th component of this column actually belongs. Doing this, however, Wolfe's anti-cycling algorithm has been obtained. It follows that Wolfe's algorithm follows exactly the same steps as the generalized lexicografic algorithm. Hence it is also finite and solves a linear programming problem without any danger for cycling.

References.

[1] B. Curtes Eaves, The linear complementarily problem. Management Science 17 (1971), 612-634.

[2] M. Sumonard, Programmation lineaire Dunod, Paris, 1962.

[3] J.J.M. Evers, Private communication.

[4] Ph. Wolfe, A technique for resolving degeneracy in linear programming. Journal SIAM, 11 (1963), 205-211.

$B_t^{-1}b$	$B_t^{-1}B_2p_{.2}$	$B_t^{-1}B_3p_{.2}$	$B_t^{-1}B_4p_{.4}$	$p_{.5}$		w_i	k_i
a	*	*	*			a	1
a	*	*	*			a	1
	b	*	*			b	2
	b	*	*			b	2
		c	*			c	3
			d			d	4
				1		1	5
				1		1	5
				1		1	5
		c	*			c	3
		c	*			c	3
	b	*	*			b	2
a	*	*	*			a	1

Figure 1. The structure of the updated right hand side matrix immediately after the introduction of the 5-th right hand side column $p_{.5}$.

Entries indicated by letters are nonnegative.

* indicates possibly nonzero elements; they are irrelevant in the course of the calculations.

The ω-column represents the right hand side column in Wolfe's anti-cycling algorithm; the k-column contains the row markers both in the generalized lexicografic as in Wolfe's algorithm.