J. Soc. Indust. Appl. Math.
Vol. 12, No. 1, March, 1964
Printed in U.S.A.

A COMBINATORIAL ALGORITHM FOR LINEAR PROGRAMS IN THE GENERAL MIXED FORM*

R. T. ROCKAFELLAR†

1. Introduction. Let a_{ij} be real numbers for $i = 0, 1, \cdots, m$ and $j = 0, 1, \cdots, n$, and let the set of integers $\{1, 2, \cdots, m + n\}$ be partitioned into three disjoint subsets K_0, K_1, and K_2, any of which may be empty. A dual pair of problems can then be stated.

Problem I: Find a vector $X = (x_1, \cdots, x_{m+n})$ minimizing

$$\bar{x} = a_{00} + x_1 a_{10} + \cdots + x_m a_{m0}$$

subject to the constraints

$$x_{m+j} = a_{0j} + x_1 a_{1j} + \cdots + x_m a_{mj}, \qquad j = 1, \cdots, n,$$

$$x_k \geqq 0 \quad \text{for} \quad k \in K_0 \qquad \text{and} \qquad x_k = 0 \quad \text{for} \quad k \in K_1.$$

Problem II: Find a vector $Y = (y_1, \cdots, y_{m+n})$ maximizing

$$\bar{y} = a_{00} - a_{01} y_{m+1} - \cdots - a_{0n} y_{m+n}$$

subject to the constraints

$$y_i = a_{i0} - a_{i1} y_{m+1} - \cdots - a_{in} y_{m+n}, \qquad i = 1, \cdots, m$$

$$y_k \geqq 0 \quad \text{for} \quad k \in K_0 \qquad \text{and} \qquad y_k = 0 \quad \text{for} \quad k \in K_2.$$

When K_1 and K_2 are empty, I and II are elementary linear programs. In the general case, they can be viewed as linear programs involving a mixture of nonnegative and unrestricted variables, equality and inequality constraints. The two problems are dual to one another in the sense that the following version of the Gale-Kuhn-Tucker existence and duality theorems is valid (see [4, Part 3]).

THEOREM 1. *If any one of the following is true,*

(a) *the constraints in* I *are consistent and the minimand is bounded below,*

(b) *the constraints in* II *are consistent and the maximand is bounded above,*

(c) *the constraints are consistent both in* I *and in* II,

then all three are true, and in fact both I *and* II *have solutions. Moreover* $\min \bar{x} = \max \bar{y}$.

Problems I and II may be solved simultaneously by the well known simplex method of Dantzig [2] or one of its variants. Actually, however,

* Received by the editors January 14, 1963, and in revised form September 17, 1963.

† Computation Center, University of Texas, Austin, Texas. Work was done under grant AF-AFOSR-467-63 with the Computation Center.

such algorithms can not be applied directly except when

$$K_1 = \emptyset, \qquad K_2 = \{1, \cdots, m\}, \qquad K_0 = \{m+1, \cdots, m+n\}, \quad \text{and} \tag{1}$$
$$a_{i0} \geqq 0 \quad \text{for} \quad i = 1, \cdots, m.$$

Thus I and II would first have to be reformulated as dual problems I′ and II′ of this special type. Although this is always possible using standard tricks [4, p. 64 ff.], the new problems often require a much larger matrix. Extra "artificial variables" may still have to be introduced into I′ and II′ in order to get the algorithm started. Sometimes, as in the case of matrix games, I and II have a natural symmetry which is lost is I′ and II′. This can add to the inconvenience of translating results. The amount of reformulation necessary depends also on whether a given problem is initially expressed in form I or in form II (in other words, on whether it or its dual is taken as the primal problem for the algorithm). For example, consider the concave program:

$$\text{maximize} \qquad f(v_1, \cdots, v_r) = \min\left\{\sum_{j=1}^{r} c_{kj} v_j + d_k \mid k = 1, \cdots, s\right\} \tag{2}$$
$$\text{subject to} \qquad \sum_{j=1}^{r} a_{ij} v_j \leqq b_i, \qquad i = 1, \cdots, t.$$

One can readily express (2) as a linear program II with $m = s + t$ and $n = r + 1$. After a typical reformulation, however, one would have $m = s + t$ and $n = 2r + s + t + 2$. On the other hand, (2) can be also expressed in form I with $m = r + 1$ and $n = s + t$ in such a way that (1) is already satisfied. But then "artificial variables" must be added, so that n increases to $s + t + r$. Furthermore, some algorithms are inefficient in this case because most of the a_{i0} are zero.

The object of this paper is to explain a new simplex-type algorithm which can always be applied *directly* to I and II, thereby eliminating the bother and inconvenience of reformulation. Besides facilitating the solution by linear programming of many types of problems, such as (2), the algorithm can be used effectively to solve mixed systems of linear equations and inequalities (case II with $a_{0j} = 0; j = 0, 1, \cdots, n$).

The algorithm is based on the concept of combinatorial equivalence developed by Tucker [7, 9]. It invokes a succession of "pivot transformations" of the $(m + 1) \times (n + 1)$ matrix of coefficients. After finitely many iterations, it either furnishes solutions both to I and to II, or it determines that one of the problems is inconsistent and hence (by Theorem 1) that neither has a solution. There are three phases differing in the way that the pivot is chosen. Phase 0 automatically reduces the problems to a simpler form, Phase 1 then finds a vector satisfying the constraints of II,

TABLEAU 0.

	K_0	K_1	K_2	
I	$x_k \geqq 0$	$x_k = 0$	x_k arb.	$\bar{x}$ min
II	$y_k \geqq 0$	y_k arb.	$y_k = 0$	$\bar{y}$ max

TABLEAU 1.

	1	$-y_{m+j}$	
1	a_{00}	a_{0j}	$= \bar{y}$
x_i	a_{i0}	a_{ij}	$= y_i$
	$= \bar{x}$	$= x_{m+j}$	

TABLEAU $T(\pi)$.

	1	$-y_{\pi(m+j)}$	
1	$a_{00}(\pi)$	$a_{0j}(\pi)$	$= \bar{y}$
$x_{\pi(i)}$	$a_{i0}(\pi)$	$a_{ij}(\pi)$	$= y_{\pi(i)}$
	$= \bar{x}$	$= x_{\pi(m+j)}$	

and finally, Phase 2 calculates the extremal solutions. An auxiliary column is added to the matrix temporarily in Phase 1. The final phase essentially coincides with Tucker's version of the simplex method in [8].

2. Combinatorially equivalent representations. Problems I and II can be summarized neatly by means of Tableau 0 and Tableau 1.

While the requirements in Tableau 0 are invariant in form, the linear relations in Tableau 1 can be reexpressed in many different ways by solving for some of the variables in terms of others. Indeed, corresponding to various permutations π of $\{1, \cdots, m + n\}$ there will be tableaux $T(\pi)$ expressing dual systems of linear relations equivalent to those in Tableau 1 (which corresponds to the identity permutation $\pi = 1$).

The matrices $A(\pi)$ obtained this way are members of a *combinatorial equivalence class*. General formulas for $A(\pi')$ in terms of $A(\pi)$, where π and π' are two permutations, may be found in [7, 9]. We shall be concerned here only with a certain special case.

Fix $i_0 \in \{1, \cdots, m\}$ and $j_0 \in \{1, \cdots, n\}$, and let π' be the permutation obtained from a given π by

$$\begin{aligned} \pi'(i_0) &= \pi(m + j_0), \\ \pi'(m + j_0) &= \pi(i_0), \\ \pi'(k) &= \pi(k) \qquad \text{for all other} \quad k \in \{1, \cdots, m + n\}. \end{aligned} \tag{3}$$

Then, provided $a_{i_0 j_0}(\pi) \neq 0$, $A(\pi')$ exists and may be calculated from $A(\pi)$ by

$$
\begin{aligned}
a_{i_0 j_0}(\pi') &= 1/a_{i_0 j_0}(\pi), \\
a_{i j_0}(\pi') &= a_{i j_0}(\pi)/a_{i_0 j_0}(\pi), && i \neq i_0, \\
a_{i_0 j}(\pi') &= -a_{i_0 j}(\pi)/a_{i_0 j_0}(\pi), && j \neq j_0, \\
a_{ij}(\pi') &= a_{ij}(\pi) - a_{i j_0}(\pi) a_{i_0 j}(\pi)/a_{i_0 j_0}(\pi), && i \neq i_0 \text{ and } j \neq j_0.
\end{aligned} \tag{4}
$$

We shall refer to the process of passing from a given tableau $T(\pi)$ to the tableau $T(\pi')$ specified by these formulas as *pivoting on* (i_0, j_0).

Starting with Tableau 1, one can produce finitely many *combinatorially equivalent representations* of I and II by repeated pivoting. It might be hoped that some of these representations are of such a lucid character that solutions to I and II can be deduced from them at a glance. Suppose in fact that a tableau $T(\pi)$ has been calculated in which

$$
\begin{aligned}
&a_{i0}(\pi) = 0 \quad \text{when} \quad \pi(i) \leqslant K_2 \quad \text{and} \\
&a_{0j}(\pi) = 0 \quad \text{when} \quad \pi(m + j) \in K_1,
\end{aligned} \tag{5a}
$$

$$
\begin{aligned}
&a_{i0}(\pi) \geqq 0 \quad \text{when} \quad \pi(i) \in K_0 \quad \text{and} \\
&a_{0j}(\pi) \geqq 0 \quad \text{when} \quad \pi(m + j) \in K_0.
\end{aligned} \tag{5b}
$$

It must then be true that

$$
\min \bar{x} = a_{00}(\pi) = \max \bar{y} \tag{6a}
$$

and that the vectors $X(\pi)$ and $Y(\pi)$ given by

$$
\begin{aligned}
x_{\pi(i)} &= 0 \quad \text{and} \quad y_{\pi(i)} = a_{i0}(\pi), && i = 1, \cdots, m, \\
x_{\pi(m+j)} &= a_{0j}(\pi) \quad \text{and} \quad y_{\pi(m+j)} = 0, && j = 1, \cdots, n,
\end{aligned} \tag{6b}
$$

are solutions to I and II, respectively. Indeed, $X(\pi)$ and $Y(\pi)$ trivially satisfy all the constraints and yield $\bar{x} = a_{00}(\pi) = \bar{y}$. Theorem 1 then implies that $a_{00}(\pi)$ is actually the common minimum and maximum in I and II.

A representation satisfying (5a) and (5b) will be called *resolvent*. The algorithm we are about to describe is based on the following fact.

THEOREM 2. *Under the hypothesis of Theorem* 1, *a resolvent representation exists for* I *and* II.

This theorem will be proved constructively by showing that the algorithm produces a finite nonrepeating sequence of representations which terminates either in a resolvent representation, or in one which indicates that I or II is inconsistent.

There is another simple kind of equivalence which will be exploited in Phase 1 of the algorithm. Consider the problems I′ and II′ obtained from I and II by adding an auxiliary column of coefficients $a_{i,n+1}(\pi_0)$, $i = 0, 1, \cdots, m$, to some tableau $T(\pi_0)$, extending π_0 by $\pi_0(m + n + 1) = m + n + 1$ and placing $m + n + 1$ in K_2 in Tableau 0. These augmented problems are trivially equivalent to the original ones. In particular, I and II satisfy the hypothesis of Theorem 1 if and only if I′ and II′ do. Furthermore, the representations of I and II correspond one-to-one with those of I′ and II′ in which $\pi(m + n + 1) = m + n + 1$. This provides us with a useful device. Given a tableau $T(\pi_0)$, we can augment it as above, calculate any sequence of combinatorially equivalent augmented tableaux $T(\pi_0), T(\pi_1), \cdots, T(\pi_r)$ such that $\pi_r(m + n + 1) = \pi_r(m + n + 1)$ at the end, and finally delete the $(n + 1)$st column again. The result of this process is always another representation of the original pair of problems.

3. Statement of algorithm. For simplicity, the particular tableau on hand at the start of each step below is always denoted by $T(\pi)$. The algorithm ordinarily begins with the original representation of I and II in Tableau 1 (see Remark 1).

Step 0.1. If $a_{ij}(\pi) = 0$ for every (i, j) such that

$$\begin{aligned} &\pi(i) \subset K_2 \quad \text{and} \quad \pi(m + j) \in K_1, \\ &\text{or} \quad \pi(i) \in K_2 \quad \text{and} \quad \pi(m + j) \in K_0, \\ &\text{or} \quad \pi(i) \in K_0 \quad \text{and} \quad \pi(m + j) \in K_1, \end{aligned} \tag{7}$$

proceed to 0.3. Otherwise do 0.2.

Step 0.2. Pivot on any (i_0, j_0) satisfying (7) such that $a_{i_0 j_0}(\pi) \neq 0$ and return to 0.1. (See Remark 2.)

Step 0.3. If $a_{0j}(\pi) \neq 0$ for some j such that $\pi(m + j) \in K_1$, then I is inconsistent. If $a_{i0}(\pi) \neq 0$ for some i such that $\pi(i) \subset K_2$, then II is inconsistent. Otherwise proceed to 1.1.

Step 1.1. Proceed to 2.1 if $a_{i0}(\pi) \geqq 0$ for all i such that $\pi(i) \in K_0$. (See Remark 3.) Otherwise augment the tableau as described earlier taking $a_{i,n+1}(\pi) = 1$ if $\pi(i) \in K_0$ and $a_{i0}(\pi) < 0$, and $a_{i,n+1}(\pi) = 0$ for all other i. (See Remark 4.) Then choose i_1 such that $\pi(i_1) \subset K_0$ and $a_{i_1 0}(\pi)$ is as negative as possible, pivot on $(i_1, n + 1)$ and proceed to 1.2. (The i_1 referred to in 1.2 and 1.3 is the one used here.)

Step 1.2. If $a_{i_1 j}(\pi) \geqq 0$ for all j with $\pi(m + j) \subset K_0$, then II is inconsistent. Otherwise choose j_0 such that $\pi(m + j) \subset K_0$ and $a_{i_1 j_0}(\pi)$ is as negative as possible, and do 1.3.

Step 1.3. If there exist indices i such that $\pi(i) \in K_0$, $a_{ij_0}(\pi) > 0$ and

$$a_{i0}(\pi)/a_{ij_0}(\pi) < a_{i_1 0}(\pi)/a_{i_1 j_0}(\pi), \tag{8}$$

choose from among them an index i_0 minimizing the ratio on the left of (8). (See Remark 6.) Then pivot on (i_0, j_0) and return to 1.2. Otherwise pivot on (i_1, j_0), interchange column j_0 and column $n+1$ of the resulting tableau, delete the resulting column $n+1$ and proceed to 2.1. (See Remark 5.)

Step 2.1. If $a_{0j}(\pi) \geqq 0$ for all j such that $\pi(m+j) \in K_0$, the representation is resolvent and solutions to I and II are given by (6a) and (6b). Otherwise select j_0 such that $\pi(m+j_0) \in K_0$ and $a_{0j_0}(\pi)$ is as negative as possible, and proceed to 2.2.

Step 2.2. If $a_{ij_0}(\pi) \leqq 0$ for all j such that $\pi(i) \in K_0$, then I is inconsistent. Otherwise choose, from among the indices i such that $\pi(i) \in K_0$ and $a_{ij_0}(\pi) > 0$, an i_0 minimizing the ratio

$$a_{i0}(\pi)/a_{ij_0}(\pi). \tag{9}$$

(See Remark 6.) Then pivot on (i_0, j_0) and return to 2.1.

Remark 1. The algorithm can, of course, begin equally well with any tableau $T(\pi)$ representing I and II. For instance, suppose that, having already found a resolvent representation for I and II, one wants to solve the new problems obtained by changing some of the marginal coefficients in Tableau 1. Instead of starting all over again, one can apply the algorithm to the representation of the new problems obtained by modifying the old resolvent one according to the formulas in [7] or [9].

Remark 2. Phase 0 is more efficient if preference is given to pivots such that $\pi(i_0) \in K_2$ and $\pi(m+j_0) \in K_1$. It can be shown that then Phase 0 always terminates in the minimum number of iterations. For problems in which most of the coefficients in the left margin of the initial tableau are zero, the efficiency of the later phases can sometimes be increased by choosing i_0 such that $a_{i_0 0}(\pi) \neq 0$ whenever possible in 0.2.

Remark 3. When 1.1 is reached, the representation is such that I and II have been reduced essentially to dual subproblems:

$$\text{minimize} \quad \bar{x} = a_{00}(\pi) + \sum_{\pi(i) \in K_0} x_{\pi(i)} a_{i0}(\pi),$$

$$\text{subject to} \quad x_{\pi(i)} \geqq 0 \quad \text{for} \quad \pi(i) \in K_0, \quad \text{and}$$

$$0 \leqq x_{\pi(m+j)} = a_{i0}(\pi) + \sum_{\pi(i) \in K_0} x_{\pi(i)} a_{ij}(\pi) \quad \text{for} \quad \pi(m+j) \in K_0;$$

$$\text{maximize} \quad \bar{y} = a_{00}(\pi) - \sum_{\pi(m+j) \in K_0} a_{0j}(\pi) y_{\pi(m+j)},$$

$$\text{subject to} \quad y_{\pi(m+j)} \geqq 0 \quad \text{for} \quad \pi(m+j) \in K_0, \quad \text{and}$$

$$0 \leqq y_{\pi(i)} = a_{i0}(\pi) - \sum_{\pi(m+j) \in K_0} a_{ij}(\pi) y_{\pi(m+j)} \quad \text{for} \quad \pi(i) \in K_0.$$

If $a_{i0}(\pi) \geqq 0$ for all i with $\pi(i) \in K_0$, the vector $Y(\pi)$ in (6b) satisfies the constraints of II, and the algorithm moves directly to the optimizing phase. Otherwise Phase 1 first changes the representation to one having this extra property.

Remark 4. In augmenting the tableau in 1.1, one could also set $a_{i,n+1}(\pi) = 1$ for all i. This would be simpler and perhaps just as efficient.

Remark 5. Before pivoting on (i_1, j_0) in the last part of 1.3, there is a representation of the augmented problems in which $\pi(i_1) = m + n + 1$ by definition of i_1. The π' representation after pivoting has $\pi'(m + j_0) = m + n + 1$ according to (3). Interchanging column j_0 and column $n + 1$ then yields a π'' representation with $\pi''(m + n + 1) = m + n + 1$, so that when column $n + 1$ is deleted the tableau returns to a representation of the original problems, as explained earlier.

Remark 6. It may happen that the minimum of the ratio in 1.3 or in 2.2 is achieved on a set I_0 containing more than one index. If this degenerate situation occurs repeatedly, it is theoretically possible (although quite unlikely in practice) that the algorithm will cycle. The possibility is eliminated entirely if the following procedure is used to pick i_0 whenever I_0 contains two or more indices. Let π_0 be the permutation as the algorithm first reaches 1.2 in the case of Phase 1, or 2.1 in the case of Phase 2. Let $k_1, \cdots, k_r$ be the indices in K_0 in their order of appearance in π_0 (i.e., $\pi_0^{-1}(k_1) < \cdots < \pi_0^{-1}(k_r)$). Starting with the set I_0 define I_s recursively as follows. If $k_s = \pi(i')$ where $1 \leqq i' \leqq m$, let $I_s = I_{s-1}$ if $i' \notin I_{s-1}$ or let I_s be I_{s-1} with i' deleted if $i' \in I_{s-1}$. If $k_s = \pi(m + j')$ where $1 \leqq j' \leqq n$, let I_s be the subset of I_{s-1} minimizing the ratio $a_{ij'}(\pi)/a_{ij_0}(\pi)$. This process continues until I_s contains just one index which is then taken as the i_0.

Remark 7. Speed could be gained in machine computation by a technique closely resembling one used in the revised simplex method [3], in which the original tableau is stored but columns of later tableaux are calculated individually only as they are needed.

4. Justification of the algorithm. In order to prove that the algorithm works, it is enough to show that the various possible terminal tableaux do have the meanings ascribed to them, and that no tableau occurs twice during one of the three iterative phases. Since a given pair of problems has only finitely many combinatorially equivalent representations, this implies that the algorithm eventually finds solutions or determines that none exist.

Each iteration of step 0.2 strictly reduces the total number of indices k such that $k \in K_2$ and $\pi^{-1}(k) \leqq m$, or $k \in K_1$ and $\pi^{-1}(k) \geqq m + 1$. Hence no cycling is possible in Phase 0. In 0.3 we have $a_{ij}(\pi) = 0$ for

		$\pi(m+j) \in K_1$	$\pi(m+j) \in K_0$	$\pi(m+j) \in K_2$
	$a_{00}(\pi)$	0	$a_{0j}(\pi)$	
$\pi(i) \in K_2$	0	0	0	
$\pi(i) \in K_0$	$a_{i0}(\pi)$	0	$a_{ij}(\pi)$	
$\pi(i) \in K_1$				

FIG. 1

all (i, j) satisfying (7). Therefore, for each remaining j such that $\pi(m + j) \in K_1$, column j of the tableau represents a constraint

$$x_{\pi(m+j)} = a_{0j}(\pi) + x_{\pi(1)}a_{ij}(\pi) + \cdots + x_{\pi(m)}a_{mj}(\pi)$$

of I in which the coefficient of $x_{\pi(i)}$ is zero unless $\pi(i) \in K_1$. Since x_k is required to be zero for $k \in K_1$, and $\pi(m + j) \in K_1$, I is inconsistent if $a_{0j}(\pi) \neq 0$. A parallel argument shows that II is inconsistent unless $a_{i0}(\pi) = 0$ for all i such that $\pi(i) \in K_2$ in 0.3.

At the end of Phase 0, the matrix of the representation has the form indicated in Fig. 1, except for the ordering of its rows and columns. Suppose it is also true that

$$a_{i0}(\pi) \geqq 0 \quad \text{for all } i \text{ such that } \quad \pi(i) \in K_0. \tag{10}$$

This is the case where the algorithm passes from 1.1 directly to Phase 2; it will be demonstrated later that the representation also has all these properties after Phase 1 has been passed through the hard way.

The pattern of zeros in Fig. 1 is preserved under iterations of 2.2 since the pivots (i, j) are all such that $\pi(i) \in K_0$ and $\pi(m + j) \in K_0$. It is also immediate from (4) that, because of the way the pivot is chosen, (10) is preserved in 2.2 and $a_{00}(\pi)$ is never decreased. In particular, the terminal tableau in 2.1 would indeed be resolvent. Moreover, no tableau can occur twice in Phase 2, except perhaps at the beginning and end of a chain of tableaux in which $a_{00}(\pi)$ remains constant. The procedure described in Remark 6 makes such cyclic chains impossible; it is a straightforward adaptation of the well known procedure of Charnes [1] for getting rid of the same possibility in the ordinary simplex method. In the terminal tableau in 2.2, column j_0 expresses a constraint

$$x_{\pi(m+j_0)} = a_{0j_0}(\pi) + x_{\pi(1)}a_{1j_0}(\pi) + \cdots + x_{\pi(x)}a_{mj_0}(\pi)$$

of I in which $a_{0j_0}(\pi) < 0$, $a_{ij_0}(\pi) \leqq 0$ when $\pi(i) \in K_0$, and $a_{ij_0}(\pi) = 0$ when $\pi(i) \in K_2$. No vector X meeting the requirements in Tableau 0 could also satisfy this constraint, so I is indeed inconsistent in this case.

Only Phase 1 still needs to be explained. The motivating idea is as follows.

The variable y_{m+n+1} added to II when the tableau is augmented in Step 1.1 allows for an adjustment in the values of certain of the y_k. Putting $m + n + 1$ in K_2 requires the amount of adjustment ultimately to be zero. After the pivot transformation in 1.1, there is a representation whose corresponding vector $Y(\pi)$ in (6b) satisfies all the constraints of II′, except that the adjustment $y_{m+n+1} = a_{i_1 0}(\pi)$ involved is negative. The algorithm proceeds now just as in Phase 2, but with row i_1 (expressing y_{m+n+1}) playing the role otherwise allocated to the top row (expressing $\bar{y}$, which is temporarily ignored). Successive iterations improve the value of y_{m+n+1}, without violating other constraints, until it is possible to make $y_{m+n+1} = 0$. The latter is accomplished by the final pivot transformation in 1.3.

More formally, one may easily verify from (4) that after pivoting in 1.1 the tableau satisfies (10), has $a_{i_1 0}(\pi) < 0$, and has the pattern of zeros indicated in Fig. 1 except for its row i_1 (where $\pi(i_1) = m + n + 1 \in K_2$) which does nevertheless have $a_{i_1 j}(\pi) = 0$ when $\pi(m + j) \in K_1$. Moreover, these properties are maintained under the pivot transformations selected in the first part of 1.3 which also never decrease $a_{i_1 0}(\pi)$. Hence no representation can appear twice in Phase 1, at least not when the degeneracy routine in Remark 6 is used. When the last part of 1.3 is reached, the maintained properties of the tableau and the chosen properties of the pivot guarantee that the resulting tableau once again resembles the one in Fig. 1 and satisfies (10), and hence is suitable for Phase 2. When the algorithm terminates instead in 1.2, row i_1 of the tableau expresses a constraint

$$y_{m+n+1} = y_{\pi(i_1)} = a_{i_1 0}(\pi) - a_{i_1 1}(\pi) y_{\pi(m+1)} - \cdots - a_{i_1 n}(\pi) y_{\pi(m+n)}$$

of the augmented problem II′ in which $a_{i_1 0}(\pi) < 0$, $a_{i_1 j}(\pi) \geqq 0$ when $\pi(m + j) \in K_0$, and $a_{i_1 j}(\pi) = 0$ when $\pi(m + j) \in K_1$. Since $m + n + 1 \in K_2$ in II′, there can be no vector satisfying this constraint along with the other requirements in Tableau 0. Hence II′ is inconsistent in this case, implying the inconsistency of the original problem II, as pointed out before.

5. Example. The workings of the algorithm (though not its efficiency) can be inspected by solving the following problem, which is designed to make use of all of the steps.

$$\text{Minimize} \quad \max\{4 - 3x_1 - 7x_2 + 3x_3,\ 1 - 2x_1 - 4x_2 + 2x_3\}$$

$$\text{subject to} \quad x_2 \geqq 0, \quad x_1 + 2x_2 - x_3 \leqq 0 \quad \text{and} \tag{11}$$

$$x_1 + x_2 - x_3 = -1.$$

		1	$-y_5$	$-y_6$	$-y_7$	$-y_8$	
	1	0	1	0	−4	−1	$= \bar{y}$
$\pi_1 = (1234 \mid 5678)$	x_1	0	(1)	−1	3	2	$= y_1$
	x_2	0	1	−2	7	4	$= y_2$
	x_3	0	−1	1	−3	−2	$= y_3$
	x_4	1	0	0	1	1	$= y_4$
		$= \bar{x}$	$= x_5$	$= x_6$	$= x_7$	$= x_8$	

$K_0 = \{2, 6, 7, 8\}$
$K_1 = \{5\}$
$K_2 = \{1, 3, 4\}$

	0	−1	1	−7	−3
$\pi_2 = (5234 \mid 1678)$	0	1	−1	3	2
	0	−1	−1	4	2
	0	1	0	0	0
	1	0	0	(1)	1

	7	−1	1	7	4	0
$\pi_3 = (5237 \mid 1648 \mid 9)$	−3	1	−1	−3	−1	0
	−4	−1	−1	−4	−2	(1)
	0	1	0	0	0	0
	1	0	0	1	1	0

	7	−1	1	7	4	0
$\pi_4 = (5037 \mid 16482)$	−3	1	−1	−3	−1	0
	−4	−1	−1	−4	−2	1
	0	1	0	0	0	0
	1	0	0	1	(1)	0

	3	−1	1	3	−4	0
$\pi_5 = (5938 \mid 16472)$	−2	1	−1	−2	1	0
	−2	−1	(−1)	−2	2	1
	0	1	0	0	0	0
	1	0	0	1	1	0

	1	−2	1	1	−2	1
$\pi_6 = (5638 \mid 1247 \mid 9)$	0	2	−1	0	−1	−1
	2	1	−1	2	−2	−1
	0	1	0	0	0	0
	1	0	0	1	(1)	0

	3	−2	1	3	2
$\pi_7 = (5637 \mid 1248)$	1	2	−1	1	1
	4	1	−1	4	2
	0	1	0	0	0
	1	0	0	1	1

Solutions: $\min \bar{x} = \max \bar{y} = 3$,

$X(\pi_7) = (-2, 1, 0, 3, 0, 0, 0, 2)$, $Y(\pi_7) = (0, 0, 0, 0, 1, 1, 1, 0)$

FIG. 2

When (11) is expressed in form I, one obtains the first tableau of Fig. 2. Tableaux subsequently calculated by the algorithm have been abbreviated by giving only the matrix and the corresponding permutation. (π_r denotes the permutation yielding the rth tableau, while $\pi = (k_1 \cdots k_m \mid k_{m+1} \cdots k_{m+n})$ means $\pi(1) = k_1$, $\pi(2) = k_2$, etc.) The pivot element leading to the next representation has been marked in each case by parentheses. The tableau which would appear in the last part of Step 1.3, after pivoting but before interchanging columns, has been omitted (it would lie between the fifth and sixth). The final representation is resolvent and furnishes the solution $x_1 = -2$, $x_2 = 1$, $x_3 = 0$, with minimum 3.

REFERENCES

[1] A. CHARNES, *Optimality and degeneracy in linear programming*, Econometrica, 20 (1952), pp. 160–170.

[2] G. B. DANTZIG, *Maximization of a linear function of variables subject to linear inequalities*, Activity Analysis of Production and Allocation (T. C. Koopmans, ed.), John Wiley, New York, 1951, pp. 339–347.

[3] G. B. DANTZIG, A. ORDEN, AND P. WOLFE, *The generalized simplex method for minimizing a linear form under linear inequality restraints*, Pac. J. Math., 5 (1955), pp. 183–195.

[4] A. J. GOLDMAN AND A. W. TUCKER, *Theory of linear programming*, Linear Inequalities and Related Systems (H. W. Kuhn and A. W. Tucker, eds.), Ann. of Math. Study 38 Princeton University Press, Princeton, pp. 53–97.

[5] C. E. LEMKE, *The dual method of solving the linear programming problem*, Nav. Log. Q., 1 (1954), pp. 36–47.

[6] A. W. TUCKER, *Dual systems of homogeneous linear relations*, Linear Inequalities and Related Systems, Princeton University Press, Princeton, 1956, pp. 3–18.

[7] ———, *A combinatorial equivalence of matrices*, Combinatorial Analysis (R. Bellman and M. Hall, eds.), Proc. of Symposia in Appl. Math., 10 (1960), pp. 129–134.

[8] ———, *Solving a matrix game by linear programming*, IBM J. Res. Devel., 4 (1960), pp. 507–517.

[9] ———, *Combinatorial theory underlying linear programs*, Recent Advances in Mathematical Programming (L. Graves and P. Wolfe, eds.), McGraw-Hill, New York, 1963.