Efficient Computation of Equilibria for Extensive Two-Person Games

Daphne Koller* daphne@cs.berkeley.edu

Nimrod Megiddo† megiddo@almaden.ibm.com

Bernhard von Stengel‡ stengel@icsi.berkeley.edu

February 8, 1995

Revised version, submitted to GAMES AND ECONOMIC BEHAVIOR

Abstract. The Nash equilibria of a two-person, non-zero-sum game are the solutions of a certain linear complementarity problem (LCP). In order to use this for solving a game in extensive form, it is first necessary to convert the game to a strategic description such as the normal form. The classical normal form, however, is often exponentially large in the size of the game tree. In this paper we suggest an alternative approach, based on the *sequence form* of the game. For a game with perfect recall, the sequence form is a linear sized strategic description, which results in an LCP of linear size. For this LCP, we show that an equilibrium is found by Lemke's algorithm, a generalization of the Lemke-Howson method.

Keywords. Equilibrium, extensive game, Lemke-Howson algorithm, linear complementarity, sequence form.

*Computer Science Division, University of California, Berkeley, CA 94720; and IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120

†IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120; and School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel

‡International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704-1105; and Informatik 5, University of the Federal Armed Forces at Munich, 85577 Neubiberg, Germany

List of Symbols

A	payoff matrix player 1
B	payoff matrix player 2
B^{-1}	inverse of basis matrix for pivoting
b	constant LCP vector
C_u	choice set
c	choice
d	LCP covering vector
E	constraint matrix for player 1
e	right hand side of player 1's constraints
F	constraint matrix for player 2
f	right hand side of player 2's constraints
h	entering column for pivoting
I	$n \times n$ identity matrix
M	LCP matrix
n	LCP dimension
p	dual vector for player 1
q	dual vector for player 2
u	information set
v_B	vector of basic variables
v_i	entering variable for pivoting
w	LCP vector of slack variables
$\tilde{w}$	(w with tilde) direction of secondary ray
x	vector of decision variables for player 1
y	vector of decision variables for player 2
z	LCP vector of variables
$\tilde{z}$	(z with tilde) direction of secondary ray
z_0	auxiliary variable for augmented LCP
$\tilde{z}_0$	(z_0 with tilde) direction of secondary ray

Greek letters

ε	perturbation for nondegeneracy
σ_u	sequence leading to u
$\sigma_u c$	sequence with last choice c at u

1. Introduction

In this paper, we consider extensive two-person games with general payoffs, where the players have perfect recall. Until recently, most methods for computing equilibria for extensive games involved converting the game to its normal form. While efficient solution algorithms exist for normal-form games, the conversion itself typically incurs an exponential blowup, since the number of pure strategies, even in the reduced normal form, is often exponential in the size of the game tree.

The normal form and the associated blowup can be avoided by considering *sequences* of choices instead of pure strategies. Instead of mixed strategy probabilities, the realization probabilities for playing these sequences can serve as strategic variables of a player. The number of these variables is linear instead of exponential in the size of the game tree. They were introduced by Koller and Megiddo (1992), who used them for one of the players in the game. The *sequence form* of an extensive game, described in the paper by von Stengel (1995) in this journal issue, is a strategic description where all players are treated symmetrically. The equilibria of a two-person non-zero-sum game are the solutions to a small *linear complementarity problem* (LCP) corresponding to the sequence form. (For a summary of these and other results, including some material of the present paper, see Koller, Megiddo, and von Stengel 1994.)

The LCP arising from a (normal form) bimatrix game can be solved by the algorithm by Lemke and Howson (1964), which is said to be efficient in practice; for a nice exposition see Shapley (1974). That algorithm finds a solution to a certain LCP with arbitrary nonnegative variables. The LCP solutions correspond to equilibria of the bimatrix game if the variables are normalized so that they represent mixed strategy probabilities. Unfortunately, the standard Lemke-Howson algorithm cannot be applied to the LCP resulting from the sequence form, since realization probabilities for sequences are defined by more complicated equations. This problem is solved in the present paper. Instead of the Lemke-Howson method, we use the related but more general algorithm by Lemke (1965). Since Lemke's algorithm is also said to be efficient in practice, this provides an effective algorithm for finding equilibria of general two-person games in extensive form.

The present paper is self-contained and partly expository. In Section 2, we briefly define the sequence form for an extensive two-person game, and derive the corresponding LCP. In Section 3, we give an exposition of Lemke's algorithm since it is not widely known to game theorists, and since the treatment of degenerate problems has to be supplemented. We have drawn most of the technical material on linear complementarity from the book by Cottle, Pang, and Stone (1992). In Section 4, we prove that Lemke's algorithm terminates with a solution for our application. In the concluding Section 5, we compare our result with earlier work.

2. The Sequence Form for Extensive Two-Person Games

We use the following conventions for extensive games; for details see von Stengel (1995). An extensive game is given by a tree, payoffs at the leaves, chance moves, and information sets partitioning the set of decision nodes. The *choices* of a player are denoted by labels on tree edges. We assume for simplicity that any labels corresponding to different choices are distinct. For a particular player, any node of the tree defines a *sequence* of choices given by the respective labels (for his moves only) on the path from the root to the node. We assume that both players have *perfect recall.* By definition, this means that all nodes in an information set u of a player define for him the same sequence σ_u of choices. Under that assumption, each choice c at u is the last choice of a unique sequence $\sigma_u c$. This defines all possible sequences except for the empty sequence $\emptyset$.

The *sequence form* of an extensive game is a strategic description similar to the normal form, but where sequences replace pure strategies. The probabilities for playing these sequences and the resulting payoffs are specified as follows.

For player 1, a nonnegative vector x, called a *realization plan*, represents the realization probabilities for the sequences of player 1 when he plays a mixed strategy. These can be characterized by the equations $x(\emptyset) = 1$ and

$$-x(\sigma_u) + \sum_{c \in C_u} x(\sigma_u c) = 0$$

for all information sets u of player 1, where $x(\emptyset)$ and $x(\sigma_u c)$ for all sequences $\sigma_u c$ are the components of x, and C_u is the set of choices at u. (A realization plan x satisfying these equations corresponds to the *behavior strategy* that makes the choice c at u with probability $x(\sigma_u c)/x(\sigma_u)$ if the denominator of this term is positive, and arbitrarily otherwise.) A realization plan y for player 2 is characterized analogously. We abbreviate these equations for the nonnegative vectors x and y using the *constraint matrices* E and F and right hand sides e and f by

$$Ex = e \qquad \text{and} \qquad Fy = f. \tag{2.1}$$

The first row of these matrix equations represents the realization probability one for the empty sequence, so e and f are equal to the vector $(1, 0, \ldots, 0)^T$ of appropriate dimension. The other rows correspond to the information sets of the respective player. A typical constraint matrix is

$$E = \begin{pmatrix} 1 & & & & & & & \\ -1 & 1 & 1 & & & & & \\ & -1 & & 1 & 1 & & & \\ & -1 & & & & 1 & 1 & 1 \end{pmatrix} \tag{2.2}$$

for a player 1 with three information sets which have two, two, and three choices, respectively, and where the first choice at the first information set precedes both the second and third information set.

The payoffs to player 1 and 2 are represented by matrices A and B, respectively. Each row corresponds to a sequence of player 1, each column to a sequence of player 2. Each leaf of the game tree defines a pair of sequences. Pairs of sequences not defined by a leaf have matrix entry zero. For a pair of sequences defined by a leaf, the player's payoff is his payoff at the leaf if there are no chance moves. If there are chance moves, a pair of sequences may correspond to more than one leaf. The payoff entry is then the sum, over all leaves that define the given pair of sequences, of the payoff at the respective leaf times the probability that chance moves allow reaching it. The resulting payoff matrices A and B are sparse and have a linear number of nonzero entries. For realization plans x and y, the expected payoffs to player 1 and 2 are then x^TAy and x^TBy, respectively.

Using these expected payoffs and the linear constraints (2.1), we can characterize an *equilibrium* of the game as a solution to a certain LCP. An equilibrium is a pair x, y of mutual best responses. In particular, if the realization plan y is fixed, then x is a best response to y if and only if it is an optimal solution of the linear program

$$\begin{array}{ll} \underset{x}{\text{maximize}} & x^T(Ay) \\ \text{subject to} & x^TE^T = e^T, \\ & x \geq 0\,. \end{array} \tag{2.3}$$

The dual LP to (2.3) has an unconstrained vector p of variables and reads

$$\begin{array}{ll} \underset{p}{\text{minimize}} & e^Tp \\ \text{subject to} & E^Tp \geq Ay\,. \end{array} \tag{2.4}$$

Feasible solutions x, p of these two LPs are optimal if and only if the two objective function values are equal, that is, $x^T(Ay) = e^Tp$. By the constraints in (2.3) this is equivalent to $x^T(Ay) = x^TE^Tp$ or

$$x^T(-Ay + E^Tp) = 0\,. \tag{2.5}$$

This condition is known as 'complementary slackness' in linear programming. It states that two nonnegative vectors are orthogonal, which means that they are complementary in the sense that they cannot both have a positive component in the same position.

In the same way, y is a best response to x if and only if it satisfies the constraints

$$Fy = f, \qquad y \geq 0 \tag{2.6}$$

and there exists a vector q such that

$$F^Tq \geq B^Tx \tag{2.7}$$

and

$$y^T(-B^Tx + F^Tq) = 0\,. \tag{2.8}$$

The expected payoff to player 2 is $y^T(B^Tx)$.

Thus, any equilibrium x, y is part of a solution x, y, p, q to the constraints in (2.3)–(2.8). These constraints define a *linear complementarity problem* (LCP). An LCP in standard form is specified by a pair b, M with a vector b in $\mathbb{R}^n$ and an $n \times n$ matrix M (see Cottle, Pang and Stone 1992, p. 1). The problem is to find $z \in \mathbb{R}^n$ so that

$$\begin{aligned} z &\geq 0 \\ b + Mz &\geq 0 \\ z^T(b + Mz) &= 0\,. \end{aligned} \tag{2.9}$$

In order to translate our LCP into this standard form, we introduce nonnegative vectors p', p'' and q', q'' of the same dimension as the unconstrained vectors p and q, respectively, and represent the latter by $p = p' - p''$ and $q = q' - q''$. The nonnegative vector z of LCP variables is then $z = (x, y, p', p'', q', q'')^T$. Furthermore, we let

$$M = \begin{pmatrix} & -A & E^T & -E^T & & \\ -B^T & & & & F^T & -F^T \\ -E & & & & & \\ E & & & & & \\ & -F & & & & \\ & F & & & & \end{pmatrix} \qquad \text{and} \qquad b = \begin{pmatrix} 0 \\ 0 \\ e \\ -e \\ f \\ -f \end{pmatrix}. \tag{2.10}$$

Then, $b + Mz \geq 0$ is obviously equivalent to the constraints: $E^Tp \geq Ay$ as in (2.4), (2.7), and (2.1). Finally, the complementarity condition $z^T(b + Mz) = 0$ in (2.9) is equivalent to (2.5) and (2.8) since the remaining conditions $p'^T(e - Ex) = 0$ etc. are implied by (2.1). To this LCP, we will apply Lemke's algorithm.

3. Lemke's Algorithm

Lemke (1965) described a complementary pivoting algorithm for finding a solution to an LCP of the general form (2.9). We describe it briefly in this section; for more detailed expositions see Murty (1988, pp. 63–84) and Cottle, Pang and Stone (1992, pp. 270–280 and 336–342).

For Lemke's method, the system (2.9) is rewritten and generalized as follows. Let I be the $n \times n$ identity matrix and d be an n-vector with positive components (for example, $d = (1, \ldots, 1)^T$). Using an auxiliary variable z_0, the term $b + Mz$ in (2.9) is replaced by $b + dz_0 + Mz$, which is denoted by the n-vector w. The problem generalizing (2.9) is that of finding $w \geq 0$, $z_0 \geq 0$, and $z \geq 0$ so that

$$Iw - dz_0 - Mz = b \tag{3.1}$$

and $z^T w = 0$ hold. A solution w, z_0, z to this problem defines a solution to (2.9) if and only if $z_0 = 0$.

In (3.1), the vector b is represented as a nonnegative linear combination of certain columns of the matrix $[I, -d, -M]$. Like the simplex algorithm for linear programming, Lemke's algorithm traverses *basic* solutions of this system. These are nonnegative solutions where only n linearly independent columns, called *basic columns*, of the matrix are linearly combined. The corresponding coefficients are n *basic variables* and represent the *basis*, a subset of $\{w_1, \ldots, w_n, z_0, z_1, \ldots, z_n\}$. All nonbasic variables have value zero. The system (3.1) is called *nondegenerate* if basic variables are always positive.

Basic solutions are changed by the following *pivoting operation.* The n basic columns of $[I, -d, -M]$ define a nonsingular $n \times n$ submatrix B, so that the vector v_B of basic variables is $v_B = B^{-1}b$. The algorithm chooses (see below) some nonbasic variable v_i as *entering* variable; let h denote the corresponding matrix column. The algorithm then moves to a new basic solution where v_i is a basic variable. It lets v_i become positive and preserves the equation (3.1), that is, $B\,v_B + h\,v_i = b$, or equivalently

$$v_B = B^{-1}b \;-\; B^{-1}h\,v_i\,. \tag{3.2}$$

In the standard case, the entering column $B^{-1}h$ has at least one positive component. Then there is a maximum choice of v_i in (3.2) so that v_B stays nonnegative, while some component of this vector becomes zero. This is made the *leaving* variable. It leaves the basis and is replaced by the entering variable v_i. That operation is called a *pivot* and is easily computed from $B^{-1}b$ and $B^{-1}h$. It requires an update of the basis and of B^{-1}.

For $1 \le i \le n$, the variables z_i and w_i are called *complementary.* Lemke's algorithm computes with *almost complementary* basic solutions, where the basis contains at most one variable of each complementary pair z_i, w_i for $1 \le i \le n$, and may also contain z_0. For an almost complementary basic solution, $z^T w = 0$. (A nonbasic solution w, z_0, z to (3.1) with $z^T w = 0$ is also called almost complementary.) If z_0 is nonbasic, then the LCP is solved. If z_0 is one of the n basic variables, then there is a complementary pair z_i, w_i where both variables are nonbasic, and either can be made an entering variable. This leads to the following algorithm.

For initialization, let $z = 0$, so $z^T w = 0$. If $w = b + dz_0$ and z_0 is sufficiently large, then w is nonnegative since $d > 0$, and (3.1) is satisfied. The set of these almost complementary solutions is called the *primary ray.* Let z_0 be minimal such that $w = b + dz_0 \ge 0$. Unless $b \ge 0$ (in which case the LCP is solved immediately), z_0 is positive and some component w_i of w is zero. The resulting basis $\{w_1, \ldots, w_{i-1}, w_{i+1} \ldots, w_n, z_0\}$ defines the initial almost complementary basic solution to (3.1). Decreasing z_0 from infinity until the endpoint of the primary ray is reached where w_i becomes zero can be thought of as a pivot where z_0 has entered and then w_i leaves the basis. Next, the complement z_i of the variable w_i that has just left is chosen to enter the basis; this starts the main loop of the algorithm.

In the main step of the algorithm, the entering variable v_i is increased in (3.2) until some basic variable becomes zero, which is made the leaving variable. Then, a pivot is performed. If the leaving variable was z_0, then the LCP is solved. If the leaving variable was not z_0, choose its *complement* has the new entering variable and repeat the step. (This is known as the 'complementary' pivoting rule.)

This algorithm solves the LCP (2.9) except for two possible problems: *ray termination* and *degeneracy*. Geometrically, the nonnegative solutions to (3.1) define a polyhedron where the basic solutions represent vertices. Increasing v_i in (3.2) means moving along an edge to an adjacent vertex. In that way, the algorithm traces a path consisting of almost complementary edges beginning with the primary ray. A *secondary ray* results if the entering column $B^{-1}h$ in (3.2) has no positive component since then v_i can be increased indefinitely. (The analogous phenomenon occurs with the simplex algorithm for an unbounded LP objective function.) For certain LCPs, ray termination can be excluded, which will be the case in our application.

The second problem is *cycling*, that is, an almost complementary basis is repeated in the computation. In that case, the corresponding vertex on the computed path is met by three or more almost complementary edges (two on the path where the vertex appeared the first time, the third when it is encountered again). At such a vertex, several edges can be followed, so that there must be a tie as to which variable should leave the basis. Since only one of them can be chosen to leave the basis, after pivoting the other will still be basic but have zero value. This means (3.1) is degenerate. Thus, if we can eliminate degeneracy, the leaving variable is unique, no basis is revisited, and the algorithm must terminate.

Degeneracy is avoided if the vector b is slightly perturbed by replacing it by $b(\varepsilon) = b + (\varepsilon, \ldots, \varepsilon^n)^T$, where ε is positive but very small. As in (3.2), the value of the entering variable v_i is then chosen to be the maximum subject to

$$B^{-1}b + B^{-1} \cdot (\varepsilon, \ldots, \varepsilon^n)^T \;-\; B^{-1}h\, v_i \;\geq\; 0\,. \tag{3.3}$$

We will show that the increase of v_i is blocked (if at all) by a unique row in (3.3): Consider any two rows j and k of the inequalities (3.3) where the components c_j and c_k, say, of the entering column $B^{-1}h$ are positive (only such rows matter). Denote the jth and kth row of $[B^{-1}b, B^{-1}]$ by $(a_{j0}, a_{j1}, \ldots, a_{jn})$ and $(a_{k0}, a_{k1}, \ldots, a_{kn})$, respectively. The corresponding inequalities in (3.3) are

$$a_{j0} + a_{j1}\varepsilon + a_{j2}\varepsilon^2 + \cdots + a_{jn}\varepsilon^n - c_j\, v_i \geq 0,$$
$$a_{k0} + a_{k1}\varepsilon + a_{k2}\varepsilon^2 + \cdots + a_{kn}\varepsilon^n - c_k\, v_i \geq 0.$$

It is easy to see that if ε is sufficiently small, then row j blocks the increase of v_i earlier than row k if and only if the row vector $1/c_j \cdot (a_{j0}, a_{j1}, \ldots, a_{jn})$ is *lexicographically smaller* than $1/c_k \cdot (a_{k0}, a_{k1}, \ldots, a_{kn})$, that is, it is smaller in the first component where the vectors differ; furthermore, these vectors are not equal since B^{-1} is nonsingular. In that way, the leaving variable is uniquely determined by a 'lexico-minimum ratio test' (which is also known for the simplex algorithm; see, for example, Chvátal 1983, p. 36). Thereby, ε can be treated as if it is 'just vanishing' (that is, zero), so that the computed solutions are not changed. Interpreted for the perturbed system, the lexicographic rule preserves the invariant that all basic variables are positive (which implies nondegeneracy), although some of them may be vanishingly small.

4. Solving the LCP for the Sequence Form

We will apply Lemke's algorithm to the LCP derived from the sequence form. In order to show that the algorithm terminates with a solution in this case, we must show that it cannot terminate with a secondary ray. This latter possibility can be excluded when the vector and matrix defining the LCP have certain properties; such 'matrix classes' have been widely researched in the literature on LCPs. In our application, we use such a property stated in Theorem 4.4.13 by Cottle, Pang and Stone (1992, p. 277); this theorem is also implicit in earlier work by Lemke (1965) and Cottle and Dantzig (1968). We state this result in Theorem 4.1 below. The proof is not new, but we present it here in a single piece as a convenience to the reader; in the literature, various LCP matrix classes, ray termination, and degeneracy are often studied separately and with their own terminology that is not necessary here. Furthermore, we have slightly generalized the theorem to *degenerate* LCPs.

For a degenerate LCP, cycling is avoided by the lexicographic method. However, the mentioned Theorem 4.4.13 could, at first glance, fail because its proof considers a basic solution (the endpoint of a secondary ray) where z_0 is a basic variable with positive value. In a degenerate problem, z_0 may be zero, and the conclusion of the theorem is invalid if degeneracy is ignored completely, as Example 4.4.16 in Cottle, Pang and Stone (1992, p. 279) shows. This poses no difficulty since in a basic solution where the variable z_0 is basic but zero, it can be chosen to leave the basis (before invoking the lexicographic rule) and a solution to the LCP is at hand. As a slight generalization of known results, we show that no harm is done if the lexicographic rule is used alone; other than in this respect, the following proof is not new.

Theorem 4.1. If (i) $z^TMz \geq 0$ for all $z \geq 0$, and (ii) $z \geq 0$, $Mz \geq 0$ and $z^TMz = 0$ imply $z^Tb \geq 0$, then Lemke's algorithm computes a solution of the LCP (2.9) and does not terminate with a secondary ray.

Proof. Suppose M and b satisfy (i) and (ii), and assume to the contrary that Lemke's algorithm terminates with a secondary ray. Let (w, z_0, z) denote the endpoint of the ray. This is a basic solution of (3.1), where the vector v_B of basic variables includes z_0 since it would otherwise solve the LCP. We assume first that z_0 is positive.

Ray termination means that the entering column $B^{-1}h$ in equation (3.2) is nonpositive. The elements of the secondary ray result if v_i in that equation takes any nonnegative value. They can be written as $(w, z_0, z) + \lambda(\tilde{w}, \tilde{z}_0, \tilde{z})$ for $\lambda \geq 0$ (with $\lambda = v_i$). The vector $(\tilde{w}, \tilde{z}_0, \tilde{z})$ is nonnegative; its components are the components of $-B^{-1}h$, a one in the place of the entering variable, and zero otherwise.

Since the elements of the secondary ray are solutions to (3.1), this equation for $\lambda = 0$ and $\lambda = 1$ implies

$$\tilde{w} = d\tilde{z}_0 + M\tilde{z}\,. \tag{4.1}$$

Furthermore, it is easy to see that $\tilde{z} \neq 0$ since the secondary ray is not the primary ray (Cottle, Pang and Stone 1992, p. 275). Because its elements are almost complementary, one can infer

$$0 = \tilde{z}^T\tilde{w} = \tilde{z}^T d\tilde{z}_0 + \tilde{z}^T M\tilde{z}\,.$$

This equation has been stated by Lemke (1965, p. 687, equation (20) with $\tilde{z}_0 = u_0$, $\tilde{z} = u$), and by Cottle and Dantzig (1968, p. 116, equation (37)). It implies $\tilde{z}_0 = 0$ since $\tilde{z}$ is nonnegative and nonzero and $d > 0$, and since the last term is nonnegative by assumption (i). Thus, $\tilde{z}^T M\tilde{z} = 0$, and by (4.1), $\tilde{w} = M\tilde{z} \geq 0$. Assumption (ii) therefore implies $\tilde{z}^T b \geq 0$. We derive a contradiction to this conclusion as follows, where the inequality follows from (i):

$$\begin{aligned} 0 &= (z + \lambda\tilde{z})^T(w + \lambda\tilde{w}) \\ &= (z + \lambda\tilde{z})^T(b + dz_0 + M(z + \lambda\tilde{z})) \\ &\geq (z + \lambda\tilde{z})^T(b + dz_0) \\ &= z^T(b + dz_0) + \lambda\tilde{z}^T(b + dz_0). \end{aligned}$$

The last term is nonpositive for all $\lambda \geq 0$ only if $\tilde{z}^T(b + dz_0) \leq 0$, or equivalently, $\tilde{z}^T b \leq -\tilde{z}^T(dz_0) < 0$, contradicting (ii).

Permitting degeneracy, let the endpoint (w, z_0, z) of the secondary ray be such that z_0 is a basic variable but has value zero. Because this basic solution has been computed using lexicographic degeneracy resolution, there is a perturbation of (3.1) where b is replaced by $b(\varepsilon) = b + (\varepsilon, \ldots, \varepsilon^n)^T$ for some small positive ε, and the same basis defines a (perturbed) solution that is nondegenerate so that z_0 is positive. For the perturbed system, there is still a secondary ray since the nonpositive entering column $B^{-1}h$ in (3.2) does not depend on b. With the same argument as before, we can now conclude $\tilde{z}^T b(\varepsilon) < 0$, which is again a contradiction to (ii) since $\tilde{z}^T b(\varepsilon) = \tilde{z}^T b + \tilde{z}^T(\varepsilon, \ldots, \varepsilon^n)^T > \tilde{z}^T b$. This shows that the theorem holds even if Lemke's algorithm encounters degenerate solutions, provided it uses the lexicographic method. $\square$

We apply this theorem to the LCP defined by (2.10) using the following two lemmas, where the first is immediate from the structure of the constraint matrices, as example (2.2) illustrates.

Lemma 4.2. The only nonnegative solutions x and y to $Ex = 0$ and $Fy = 0$ are $x = 0$ and $y = 0$.

Lemma 4.3. If $E^T p \geq 0$ and $F^T q \geq 0$ then $e^T p \geq 0$ and $f^T q \geq 0$.

Proof. Consider the following LP: maximize 0 subject to $Ex = e$, $x \geq 0$. It is feasible, so the value 0 of its objective function is a lower bound for the objective function of the dual LP: minimize $e^T p$ subject to $E^T p \geq 0$. Hence, if $E^T p \geq 0$ then $e^T p \geq 0$. Similarly, $F^T q \geq 0$ implies $f^T q \geq 0$. $\square$

Theorem 4.4. If $A \leq 0$ and $B \leq 0$, then M and b in (2.10) satisfy all assumptions of Theorem 4.1.

Proof. Let $z = (x, y, p', p'', q', q'')^T \geq 0$ and $p = p' - p''$, $q = q' - q''$ as above. Then $z^T M z = x^T(-A - B)y \geq 0$, satisfying (i). For (ii), $Mz \geq 0$ is equivalent to $-Ay + E^T p \geq 0$, $-B^T x + F^T q \geq 0$, $Ex = 0$ and $Fy = 0$. This implies, by Lemma 4.2, $x = 0$ and $y = 0$, and therefore $E^T p \geq 0$ and $F^T q \geq 0$, so that by Lemma 4.3, $e^T p \geq 0$ and $f^T q \geq 0$. We conclude that $z^T b = b^T z = e^T p + f^T q \geq 0$. (Note that we did not use the assumption $z^T M z = 0$.) $\square$

The conditions $A \leq 0$ and $B \leq 0$ can be assumed without loss of generality, by subtracting a constant from the payoffs to the players at the leaves of the tree so that these become nonpositive. This transformation does not change the game. The same assumption is made for the algorithm by Lemke and Howson (1964). Without this condition, easy examples show that Lemke's algorithm may terminate with a secondary ray instead of an LCP solution.

5. Conclusions and Comparison with Related Work

We have shown that Lemke's algorithm, applied to our LCP, terminates with a solution. Since all solutions to the LCP are equilibria, this shows that our algorithm finds some equilibrium of the game in extensive form. Our algorithm can also be used to solve bimatrix (i.e., normal form) games. The game is represented as an extensive game in the standard way, where each player has only one information set and his choices are his strategies. The sequence form of that game has essentially the same payoff matrices as the normal form. Clearly, there is a direct correspondence between the equilibria in the two representations of the game, so that our algorithm, applied to the sequence form, also constructs an equilibrium for the bimatrix game. For such games, however, the algorithm by Lemke and Howson (1964) also finds an equilibrium. It is known that certain equilibria of bimatrix games may be 'elusive' to the Lemke-Howson method (Aggarwal 1973). Since the two algorithms operate similarly, we conclude that certain equilibria may be elusive to our method as well.

The size of the sequence form is linear in the size of the extensive game, whereas the size of the normal form is generally exponential. Therefore, our algorithm is exponentially faster than the standard Lemke-Howson method applied to the normal form. Our method also needs exponentially less space if the entire normal form is stored. There are two other algorithms for solving two-person extensive-form games that avoid the conversion to normal form and the associated exponential blowup. These are based not on the idea of sequences, but on the idea of mixed strategies with small supports. The *support* of a mixed strategy is the number of pure strategies to which it gives positive probability.

Wilson (1972) presented a variant of the Lemke-Howson algorithm for solving a two-player game with perfect recall that uses the extensive form directly. There are two important differences between Wilson's variant and the original Lemke-Howson algorithm. First, Wilson's method never deals with the entire LCP. Rather, it generates pivoting columns for the Lemke-Howson algorithm directly from the game tree. These columns are best-response pure strategies, and can be found by backward induction, using the perfect recall structure of the information sets. This leaves the problem of storing an intermediate solution during the search for an equilibrium, which still requires exponential space in the size of the game tree. In order to avoid this problem, Wilson's algorithm only maintains a subset of the basic variables at each point, namely those variables corresponding to mixed-strategy probabilities. The basic variables corresponding to the 'slack variables' are not stored explicitly, but are recomputed as needed.

Wilson did not prove formally why his algorithm should provide significant savings. He just observed empirically that "the frequency of equilibria using only a very few of the available pure strategies is very high." Koller and Megiddo (1995) proved that Wilson's approach (or a slight variant of it) is efficient because it suffices to consider mixed strategies with small support. They showed that two mixed strategies with the same realization probabilities for the leaves are realization equivalent. This implies that any mixed strategy has a realization equivalent mixed strategy whose support is at most the number of possible sequences (and is hence linear in the size of the game tree). In addition to showing that Wilson's empirical observation was justified, Koller and Megiddo constructed an algorithm for finding all equilibria of an extensive two-person game that runs in exponential time in the size of the game tree (and *not* in the large size of the normal form). Their algorithm enumerates all small supports for both of the players, and attempts to construct an equilibrium over that support pair. Unlike Wilson's algorithm and the method presented here, their algorithm constructs all equilibria, and works in exponential time even if the game has imperfect recall. However, since it is based on complete enumeration, its running time is exponential in all cases, not just in the worst case.

The sequence form can also be used in an algorithm that enumerates all equilibria. It can be shown that all equilibria of a game can be found by enumerating the complementary basic solutions to (3.1) (where $z_0 = 0$). Thereby, each of the 2^n sets of variables containing one variable of each complementary pair z_i, w_i for $1 \leq i \leq n$ is tested for being a nonnegative basic solution to (3.1). If this is the case, it solves the LCP (2.9). Mangasarian (1964) showed that in the case of bimatrix games this suffices to derive all equilibria. It is possible to show that the same argument applies also to the LCP defined by the sequence form.

The running time of our algorithm is also at worst exponential in the size of the extensive game (this is known for Lemke's algorithm even if applied to zero-sum games). However, this seems to be a rare case, like the exponential worst-case behavior of the simplex algorithm. In practice, it is likely that our method, like the simplex method, will be much faster. The complexity of constructing some equilibrium of a bimatrix game is currently unknown; this is a difficult open question. Related problems, such as finding an equilibrium with maximum payoff for a player, were shown to be NP-hard by Gilboa and Zemel (1989). The problems they discussed can be solved by a process that enumerates all equilibria.

As a topic for further research, it may be interesting to study further the computation by Lemke's algorithm in terms of the extensive game. Wilson (1972) interpreted the entering columns in the Lemke-Howson algorithm as best responses against the current pair of mixed strategies. In the case of the sequence form, the components of p and q in (2.4) and (2.7) can be interpreted as payoff contributions of optimal choices at information sets (von Stengel 1995, Section 6). It is therefore quite possible that, as in Wilson's algorithm, the entering columns can be interpreted as choices at information sets that are best responses against the current pair of realization plans. This might allow us to use the sequence form to construct equilibria satisfying certain local optimality conditions, such as subgame perfection.

Acknowledgements

We would like to thank Richard Cottle and Lloyd Shapley for stimulating discussions. Research supported in part by ONR Contracts N00014-91-C-0026 and N00014-94-C-0007, by the Air Force Office of Scientific Research (AFSC) under Contract F49620-91-C-0080, by a University of California President's Postdoctoral Fellowship, by the Volkswagen Foundation, and by the German-Israeli Foundation for Scientific Research and Development (G.I.F.). The United States Government is authorized to reproduce and distribute reprints for governmental purposes.

References

V. Aggarwal (1973), On the generation of all equilibrium points for bimatrix games through the Lemke-Howson algorithm. Mathematical Programming 4, 233–234.

V. Chvátal (1983), Linear Programming. Freeman, New York.

R. W. Cottle and G. B. Dantzig (1968), Complementary pivot theory of mathematical programming. Linear Algebra and Its Applications 1, 103–125.

R. W. Cottle, J.-S. Pang and R. E. Stone (1992), The Linear Complementarity Problem. Academic Press, San Diego.

I. Gilboa and E. Zemel (1989), Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior 1, 80–93.

D. Koller and N. Megiddo (1992), The complexity of two-person zero-sum games in extensive form. Games and Economic Behavior 4, 528–552.

D. Koller and N. Megiddo (1995), Finding mixed strategies with small supports in extensive games. International Journal of Game Theory, to appear.

D. Koller, N. Megiddo, and B. von Stengel (1994), Fast algorithms for finding randomized strategies in game trees. Proceedings of the 26th ACM Symposium on Theory of Computing, 750–759.

C. E. Lemke (1965), Bimatrix equilibrium points and mathematical programming. Management Science 11, 681–689.

C. E. Lemke and J. T. Howson, Jr. (1964), Equilibrium points in bimatrix games. Journal of the Society for Industrial and Applied Mathematics 12, 413–423.

O. L. Mangasarian (1964), Equilibrium points in bimatrix games. Journal of the Society for Industrial and Applied Mathematics 12, 778–780.

K. G. Murty (1988), Linear Complementarity, Linear and Nonlinear Programming. Heldermann Verlag, Berlin.

L. S. Shapley (1974), A note on the Lemke-Howson algorithm. Mathematical Programming Study 1: Pivoting and Extensions, 175–189.

B. von Stengel (1995), Efficient computation of behavior strategies. Games and Economic Behavior, this issue.

R. Wilson (1972), Computing equilibria of two-person games from the extensive form. Management Science 18, 448–460.