Computing a quasi-perfect equilibrium of a two-player game

Peter Bro Miltersen* and Troels Bjerre Sørensen*

February 9, 2007

Abstract

Refining an algorithm due to Koller, Megiddo and von Stengel, we show how to apply Lemke's algorithm for solving linear complementarity programs to compute a quasi-perfect equilibrium in behavior strategies of a given two-player extensive-form game of perfect recall. A quasi-perfect equilibrium is known to be sequential, and our algorithm thus resolves a conjecture of McKelvey and McLennan in the positive. A quasi-perfect equilibrium is also known to be normal-form perfect and our algorithm thus provides an alternative to an algorithm by von Stengel, van den Elzen and Talman. For the case of a zero-sum game, we devise variants of the algorithm that rely on linear programming rather than linear complementarity programming and use the simplex algorithm or other algorithms for linear programming rather than Lemke's algorithm. We argue that these latter algorithms are relevant for recent applications of equilibrium computation to artificial intelligence.

Keywords: Equilibrium computation, quasi-perfect equilibrium.
JEL classification: C63, C72

*Department of computer science, University of Aarhus, Aabogade 34, DK-8200 Aarhus N, Denmark. Email {bromille,trold}@daimi.au.dk.

1 Introduction

Koller, Megiddo and von Stengel (1996) showed how to apply Lemke's algorithm (Lemke, 1965) for solving linear complementarity programs (LCPs) to compute a Nash equilibrium in behavior strategies of a given two-player extensive-form game of imperfect information but perfect recall. Their algorithm avoids converting the extensive-form game to normal form and thus avoids an exponential blowup in the size of the representation of the game. Instead, the linear complementarity program they devise has roughly the same number of variables and constraints as the number of information sets of the game considered and can therefore be used in practice to solve rather large games. For the case of zero-sum games, Koller, Megiddo and von Stengel (1994) gave a variant of their algorithm based on linear programming rather than linear complementarity programming. This version of the algorithm is motivated by the fact that linear programs are more efficiently solvable than linear complementarity programs, both in theory and in practice. It has been used by artificial intelligence researchers (Billings et al., 2003; Gilpin and Sandholm, 2005), for constructing strategies for two-player poker games with millions of information sets. The computed strategies are used for computer programs capable of playing the poker games considered, for instance against human opponents. That is, they are to be understood and assessed as *prescriptive* strategies.

We shall henceforth refer to the algorithms of Koller, Megiddo and von Stengel as the *KMvS algorithms*. In this paper we show how to modify the KMvS algorithms for both the general-sum and the zero-sum case so that a *quasi-perfect* equilibrium is computed. Quasi-perfection is an attractive equilibrium refinement notion due to van Damme (1984). It refines sequential equilibrium as well as normal-form perfect equilibrium. It has been argued by Mertens (1995) that quasi-perfection is conceptually superior to Selten's notion of extensive-form perfection. In particular, Mertens presents a certain two-player voting game where any extensive-form perfect equilibrium involves using a weakly dominated strategy. That is, extensive-form perfection is in general inconsistent with *admissibility* while quasi-perfection *guarantees* normal-form perfection, which for equilibria of two-player games is equivalent to admissibility. Further discussion of quasi-perfection can be found in the survey of Hillas and Kohlberg (2002).

1.1 Background and motivation

Our algorithm addresses several issues previously raised by the computational game theory community as well as by the artificial intelligence community.

The KMvS algorithms are based on the *sequence form* for extensive form games due to von Stengel (1996). As a consequence of this, the equilibrium computed is in behavior *plans* rather than behavior *strategies*, i.e., no behavior is computed for information sets that are guaranteed not to be reached according to the plan. This is not an issue if one is interested only in computing a Nash equilibrium, as one can then assign arbitrary behavior to these information sets. One can even find a subgame perfect equilibrium simply by solving subgames separately. However, it did not seem obvious how to extend the sequence form approach to find, e.g., a sequential equilibrium (Kreps and Wilson, 1982) where meaningful behavior must be assigned to *all* information sets. McKelvey and McLennan (1996) discuss this difficulty but still conjecture "*that a suitable generalization of the sequence form will be the natural vehicle for computation of sequential equilibrium.*" It is exactly this conjecture that we confirm in this paper. The desired generalization guaranteeing the computation of meaningful behavior in all information sets turns out to be a *symbolic perturbation* of the linear complementarity programs describing the equilibria in sequence form.

The only previous algorithm we are aware of that modifies the KMvS algorithms so that some equilibrium refinement is computed is an algorithm due to von Stengel, van den Elzen and Talman (2002). Their algorithm is for the general-sum case and computes a normal-form perfect equilibrium. As quasi-perfection refines normal-form perfection, our algorithm provides an alternative to theirs. Interestingly, the algorithm of von Stengel, van den Elzen and Talman is based on applying Lemke's algorithm in a rather different way from the way it was originally applied by Koller, Megiddo and von Stengel. In contrast, our application is technically very close to this original application. As far as we know, the algorithm by von Stengel, van den Elzen and Talman has no variant based on linear programming rather than linear complementarity programming and applicable to the case of zero-sum games.

Finally, a main motivation of the present paper comes from the application of the zero-sum version of the KMvS algorithm by the artificial intelligence community to the computation of prescriptive strategies to be used by game playing computer programs. It is well known that the Nash equilibrium concept has serious deficiencies as a prescriptive solution concept, even for the case of zero-sum games where Nash equilibria are simply pairs of minimax strategies. That this deficiency shows up in the equilibria computed by the KMvS algorithm was discovered by Koller and Pfeffer (1997) who implemented the algorithm in their software package GALA and based on their computational experiences with GALA observed: "[Using maximin strategies] *can lead to very counterintuitive behavior. For example, assume that player 1 is guaranteed to win \$1 against an optimal player 2. But now, the latter makes a mistake which allows player 1 to immediately win \$10000. It is perfectly consistent with the definition for the 'optimal' (maximin) strategy to continue playing so as to win the \$1 that was the original goal.*". Clearly, insisting on an equilibrium in undominated strategies would eliminate the undesirable behavior thus alluded to by Koller and Pfeffer. Thus, we suggest that our algorithm may be preferable to the original KMvS algorithm for the artificial intelligence applications.

1.2 Outline of algorithm and organization of paper

Our algorithm is based on applying a perturbation to the game G considered, resulting in a game $G(\epsilon)$, parametrized by a parameter $\epsilon > 0$. While perturbed games are ubiquitous in the theory of equilibrium refinements, we believe that the particular perturbation model we suggest is new. We show that limit points of Nash equilibria of the perturbed games are quasi-perfect equilibria of the original game. We then show how to modify the KMvS algorithm so that a parametric expression of a Nash equilibrium of $G(\epsilon)$, valid for all sufficiently small $\epsilon > 0$ is computed. This parametric expression can be used to directly derive a quasi-perfect equilibrium of G.

The outline of the paper is as follows: In Section 2 we present our notation for extensive-form games and present the necessary facts about the sequence form. We also present van Damme's notion of quasi-perfect equilibria. In Section 3 we introduce our perturbation model and show how a parametric expression for a Nash equilibrium of the perturbed game yields a quasi-perfect equilibrium of the original game. In Section 4 we show how to apply Lemke's algorithm to compute such a parametric expression. In Section 5.1 we show how to adapt the approach to the zero-sum case, using the simplex algorithm rather than Lemke's algorithm. Finally, in Section 5.2, we show how to use generic solvers (i.e., solvers not necessarily based on the simplex algorithm) for *unparametrized* linear programs to solve the parametric linear programs we derive for the zero-sum case. As a corollary of this, we get a *polynomial time algorithm* for computing a quasi-perfect equilibrium of a given two-player extensive-form game with imperfect information but perfect recall.

2 Preliminaries

2.1 Extensive-form games and the sequence form

Throughout the paper, unless we mention otherwise, we consider a two-player, extensive-form game G with imperfect information but perfect recall. The game is given by a finite tree with payoffs at the leaves, information sets partitioning nodes of the tree and nodes of chance being allowed. Actions of a player are denoted by labels on edges of the tree. Perfect recall means that all nodes in an information set belonging to a player share the sequence of actions and information sets of that player that are visited on the path from the root to the nodes. A behavior *strategy* assigns probabilities to actions in a way consistent with the information sets. A behavior *plan* only assigns such probabilities to actions of information sets that may be reached if the plan is played. For details, see e.g., Koller, Megiddo and von Stengel (1996) or any textbook on game theory.

An important insight of von Stengel (1996) and Koller, Megiddo and von Stengel is that behavior strategies for games of perfect recall are for computational purposes often better represented in *sequence form*, which we describe next. Given a behavior strategy for one of the players, the *realization weight* of a leaf or node z in the tree is the product of behavior probabilities of actions in the *sequence* σ of actions made by that player on the path from the root of the game to z.

Because of perfect recall, all nodes in an information set belonging to a player share the same sequence of actions of that player, and are therefore assigned the same realization weight relative to that player. Further, we may assume without loss of generality that actions are labeled so that any particular label only occurs at one information set. Then, we can conveniently extend the terminology and let the realization weight of the *node* z as well as the realization weight of the *sequence* σ and the realization weight of the *information set* h to which z belongs all denote the same value.

A *realization plan* for a player is a vector of realization weights indexed by his possible sequences of actions. Note that a behavior probability is the ratio between two realization weights. If this ratio is $0/0$, the realization weights do not define a behavior probability. Still, the map between behavior *plans* and realization plans is a bijection, so given a realization plan, we may talk about the corresponding behavior plan and vice versa. Also, if a realization plan has strictly positive weight on all possible sequences, the corresponding behavior plan is fully mixed at every information set and is therefore, in fact, a behavior *strategy.*

For a game G of perfect recall, the statement that a vector is a valid realization plan can be expressed by a non-negativity constraint and a system of linear equations with no more equations than information sets. We let

$$Ex = e, x \geq 0 \tag{1}$$

denote the constraints expressing that x is a valid realization plan for Player I and we let

$$Fy = f, y \geq 0 \tag{2}$$

be the constraints expressing that y is a valid realization plan for Player II. Here, E and F are matrices containing entries from $\{-1, 0, 1\}$ only.

The key to the computational usefulness of sequence form is the following fact: If Player I plays by realization plan x and Player II plays by realization plan y, then the expected payoff for Player I is given by a *bilinear form* $x^\top Ay$. Similarly, the expected payoff for Player II is given by $x^\top Cy$. Here, A and C are matrices depending on G, and easily computed given the extensive form of G.

2.2 Quasi-perfect equilibria

Van Damme (1984) presents several characterizations of quasi-perfect equilibria. We shall adopt one of them as the definition of quasi-perfect equilibria in the present paper. We give the general definition but shall only use the two-player case in this paper.

Definition 1 Let G be an m-player game in extensive form with perfect recall.

Let $\gamma = (\gamma_i)$ be a behavior strategy profile for G which is fully mixed in each information set, γ_i being the strategy of Player i. Also, we let γ_{-i} denote $(\gamma_1, \ldots, \gamma_{i-1}, \gamma_{i+1}, \ldots, \gamma_m)$.

Fix any information set h in G. Let i be the player to which h belongs. An *h-local purification* of the strategy γ_i is a behavior strategy for Player i obtained by replacing the behavior of Player i at h and at every information set belonging to Player i that may be encountered by Player i *after* he encounters h, by behavior that puts all probability mass on some single action at each of these information sets. The behavior at all other information sets is left unchanged.

We say that an h-local purification γ'_i of γ_i is an *h-local best response* to γ_{-i} if it achieves the best expected payoff against γ_{-i} among all h-local purifications of γ_i.

We say that an h-local purification γ'_i of γ_i is *ϵ-consistent with* γ_i if γ_i assigns behavior probability strictly bigger than ϵ to the actions to which γ'_i assigns behavior probability 1 at h and subsequent information sets.

We say that the behavior strategy profile γ is *ϵ-quasi-perfect* if it is fully mixed and if for each player i and every information set h belonging to Player i, all h-local purifications of γ_i that are ϵ-consistent with γ_i are h-local best responses to γ_{-i}.

Finally, a behavior strategy profile for G (not necessarily fully mixed) is a quasi-perfect equilibrium if it is a limit point as $\epsilon \to 0+$ of ϵ-quasi-perfect behavior strategy profiles.

It was shown by van Damme that any game of perfect recall possesses a quasi-perfect equilibrium and that such an equilibrium is sequential as well as normal-form perfect. The proof of existence uses the existence of normal-form *proper* equilibria and the relationship between these equilibria and quasi-perfect equilibria established by van Damme. As far as we know, no very simple and direct proof of existence is known. For the two-player case, a constructive proof of existence follows from the results of this paper.

3 Perturbed games

In this section we define a certain perturbation $G(\epsilon)$ of G parametrized by $\epsilon > 0$, and show that a parametric expression for an equilibrium of $G(\epsilon)$ can be used to compute a quasi-perfect equilibrium of G. In later sections, we then turn to computing such a parametric expression.

Definition 2 Let G be a two-player game of perfect recall and let $\epsilon > 0$ be a parameter. We define the perturbed game $G(\epsilon)$ to be a game of exactly the same structure as G (i.e., same information sets, actions and payoffs) but with a restriction on the realization plans allowed: In a valid realization plan for either player in $G(\epsilon)$, the realization weight of any sequence σ of actions that can be played must be at least $\epsilon^{|\sigma|}$ where $|\sigma|$ is the number of actions in the sequence σ.

While perturbed games are ubiquitous in the theory of equilibrium refinements, we believe that the particular perturbation model of Definition 2 is novel. Note that the perturbation model is strongly tied to the sequence form and is analogous to but *not* equivalent to a Selten tremble which would put a lower bound on ϵ on the behavior probability of each action. In fact, while a limit point of equilibria of a game perturbed by such Selten trembles is by definition an extensive-form perfect equilibrium of the game, our main Lemma 3 below shows that a limit point of equilibria of a game perturbed as in Definition 2 is a quasi-perfect equilibrium. The example of Mertens (1995) of a two-player game where *no* extensive-form perfect equilibrium is quasi-perfect thus highlights that the two perturbation models are in fact very different, in spite of their superficial similarity.

To concretely illustrate this difference, consider the one-player game G of Fig. 1. Let $G(\epsilon)$ be the sequence form perturbation of G of Definition 2. Let $G'(\epsilon)$ be G perturbed by Selten trembles, i.e., with each behavior probability restricted to be at least ϵ. Note that the only non-optimal outcome of the games occur if the player first chooses R and then r. In the only equilibrium of $G'(\epsilon)$, the behavior probabilities of the actions R and r are both ϵ. This leads to the realization weights $x_R = \epsilon$ and $x_{Rr} = \epsilon^2$. The limit as $\epsilon \to 0$ has $x_R = 0$, and therefore any extensive-form perfect equilibrium chooses L with probability 1. Intuitively, in an extensive-form perfect equilibrium, a player takes the possibility of mistakes that may be made in the future into account, including his own potential future mistakes. In contrast, any strategy with $x_{Rr} = \epsilon^2$ is an equilibrium of $G(\epsilon)$, as any such strategy yields the optimal expected payoff of $-\epsilon^2$. In particular, the value of x_R and hence the behavior probability of the action R may in equilibrium be any value between ϵ and 1. Note that this is consistent with behavior in a quasi-perfect equilibrium: In such an equilibrium, a player takes the possibility of mistakes made in the future by *other* players into account, but he ignores his own potential future mistakes. Our main lemma confirms this intuitive connection

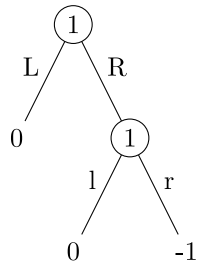

Figure 1: A one-player game

between the perturbed game $G(\epsilon)$ and quasi-perfect equilibria of G, for a general game G.

Lemma 3 *Let G be a two-player game of perfect recall. For any $\epsilon > 0$, any Nash equilibrium in behavior strategies of $G(\epsilon)$ is an ϵ-quasi-perfect behavior strategy profile for G.*

Proof Let $\gamma = (\gamma_1, \gamma_2)$ be a Nash equilibrium in behavior strategies of $G(\epsilon)$. According to the definition, we have to show that for every information set h in G, any h-local purification of γ that is ϵ-consistent with b is an h-local best response. So let an information set h be given. Assume without loss of generality that h belongs to Player I. Let an h-local purification γ_1' of γ_1 that is ϵ-consistent with γ_1 be given and let x' be the realization plan corresponding to γ_1'. Let γ_1^* be an arbitrary h-local purification of γ_1 and let x^* be the realization plan corresponding to γ_1^*. To show that γ_1' is an h-local best response, we need to show that $(x^*)^\top Ay \le (x')^\top Ay$.

Let x be the realization plan corresponding to γ_1. We claim that there is a $\delta > 0$ so that $\tilde{x} = x + \delta(x^* - x')$ is a valid realization plan for $G(\epsilon)$. To see this, we observe that since x, x^*, x'

are all valid realization plans, the linear equations $E\tilde{x} = e$ are satisfied and hence we only have to worry about the constraint stating that the realization weight of a sequence of d actions should be at least ϵ^d. The plans x^* and x' are identical on all sequences, except on sequences that contain an action taken at h, so we only have to check these. Among these sequences, we only have to worry about the ones to which x' assigns non-zero realization weight. But since γ_1' is ϵ-consistent with γ_1, a trivial induction reveals that the realization weight given by x to each of these sequences is *strictly* bigger than $\epsilon^{|\sigma|}$. Hence, the claim follows for some sufficiently small $\delta > 0$. Fix such a δ.

Let y be the realization plan corresponding to b_2. We have that

$$\begin{aligned}\tilde{x}^\top Ay &= (x + \delta(x^* - x'))^\top Ay \\ &= x^\top Ay + \delta((x^*)^\top Ay - (x')^\top Ay). \end{aligned} \tag{3}$$

Since $\tilde{x}$ is a valid realization plan for player 1 in $G(\epsilon)$ and $\gamma = (\gamma_1, \gamma_2)$ is a Nash equilibrium, we also have

$$\tilde{x}^\top Ay \leq x^\top Ay \tag{4}$$

and combining (3) and (4) we have $(x^*)^\top Ay - (x')^\top Ay \leq 0$, i.e., $(x^*)^\top Ay \leq (x')^\top Ay$, as desired. ♠

It is here worth pointing out that while extensive-from (trembling hand) perfect equilibria are *defined* as limit points of equilibria of perturbed games, we are not aware of any natural characterization of quasi-perfect equilibria in terms of limit points of equilibria of perturbed games. The present paper provides no such characterization either, but our main lemma above points out that a least a certain subset of the quasi-perfect equilibria can be obtained as such limit points, at least for the case of two-player games.

Our algorithms for computing a quasi-perfect equilibrium all first compute a parametric expression for an ϵ-quasi-perfect behavior strategy profile. More precisely, in the next section we shall prove the following fact.

Fact 4 *For any two-player game G of perfect recall, there is a $\delta > 0$ and a map γ mapping each positive real number ϵ to two vectors of real numbers $(\gamma_1(\epsilon), \gamma_2(\epsilon))$ so that*

1. *each entry of $\gamma_i(\epsilon)$, $i = 1, 2$, is a rational function, i.e. a ratio between two polynomials in ϵ,*

2. *for all values of $\epsilon \in (0, \delta]$, $\gamma(\epsilon)$ is a Nash equilibrium of $G(\epsilon)$ in behavior strategies.*

Furthermore, the proof is constructive and algorithmic: Given the game G, we show how to explicitly compute the coefficients of the polynomials defining γ. Given these expressions, it is now straightforward to compute a quasi-perfect equilibrium of G: Any rational function r of parameter ϵ so that $r(\epsilon) \in [0, 1]$ for sufficiently small ϵ has a unique limit as $\epsilon \to 0+$, and this limit can be easily computed from the coefficients of the two polynomials defining r; it is either 0 or the ratio between two coefficients of these polynomials. Thus, we can determine the limit as $\epsilon \to 0+$ of all behavior probabilities in $(\gamma_1(\epsilon), \gamma_2(\epsilon))$ and by Lemma 3 and by Definition 2, these values together define a quasi-perfect equilibrium of G in behavior probabilities.

Note that it is crucial that we compute the limit of the equilibria of $G(\epsilon)$ in *behavior strategies* and *not* the limit of the equilibria of $G(\epsilon)$ in realization plans. Only by doing the former do we ensure the computation of meaningful behavior at *all* information sets, including those having realization weight 0 in the computed equilibrium.

4 The general-sum case using Lemke's algorithm

In this section we show how to apply Lemke's algorithm to find a parametric expression for an equilibrium of $G(\epsilon)$, thus proving Fact 4. This is a technical modification of the original KMvS algorithm and we shall follow the approach of Koller, Megiddo and Stengel (1996) very closely. Thus, we shall in this section assume familiarity with their approach, including their excellent exposition of Lemke's algorithm.

First, we shall mimic Koller, Megiddo and von Stengel's derivation of a linear complementarity program characterizing the Nash equilibria of a game G in realization plans (Koller et al., 1996, Section 2) and derive a parametrized linear complementarity program describing the Nash equilibria of the perturbed game $G(\epsilon)$.

Theorem 5 *The equilibria of $G(\epsilon)$ are given by the solutions to the following linear complementarity program.*

$$\begin{aligned} Ex &= e \\ Fy &= f \\ u = E^\top p - Ay &\geq 0 \\ v = F^\top q - C^\top x &\geq 0 \\ x - k(\epsilon) &\geq 0 \\ y - l(\epsilon) &\geq 0 \\ u^\top (x - k(\epsilon)) + v^\top (y - l(\epsilon)) &= 0 \end{aligned} \tag{5}$$

where $k(\epsilon)$ is the vector indexed by sequences σ of Player I so that $k_\sigma(\epsilon) = \epsilon^{|\sigma|}$. Similarly, $l(\epsilon)$ is the vector indexed by sequences π of Player II so that $l_\pi(\epsilon) = \epsilon^{|\pi|}$.

The variables of the program are x (a realization plan for Player I), y (a realization plan for Player II), and p, q, u, v (auxiliary vector variables). The matrices A, C, E, F are as described in Section 2.1.

Proof Suppose that a strategy of Player II is fixed and given in sequence form by a realization plan y. A best response by Player I as a realization plan x is then given by

$$\begin{array}{ccc} \max\limits_{x} & x^\top (Ay) & \text{so that} \\ Ex & = & e \\ x & \geq & k(\epsilon). \end{array} \tag{6}$$

The dual of (6) is

$$\begin{array}{ccc} \min\limits_{p,u} & p^\top e - u^\top k(\epsilon) & \text{so that} \\ E^\top p & \geq & Ay + u \\ u & \geq & 0. \end{array} \tag{7}$$

Since $k(\epsilon)$ is positive at every entry, an optimal solution to (7) has $u = E^\top p - Ay$. Also, by linear programming duality, feasible solutions x to (6) and (p, u) to (7) are optimal if and only if the two objective function values are equal, i.e.,

$$x^\top (Ay) = p^\top e - u^\top k(\epsilon) \tag{8}$$

which, using $e = Ex$ and $E^\top p - Ay = u$ is equivalent to

$$u^\top (x - k(\epsilon)) = 0. \tag{9}$$

Similarly, suppose that a strategy of Player I is fixed and given in sequence form by a realization plan x. A best response by Player II as a realization plan y is then given by

$$\begin{array}{ccl} \max\limits_{y} & x^\top (Cy) & \text{so that} \\ Fy & = & f \\ y & \geq & l(\epsilon). \end{array} \tag{10}$$

The dual of (10) is

$$\begin{array}{ccl} \max\limits_{q,v} & q^\top f - v^\top l(\epsilon) & \text{so that} \\ F^\top q & \geq & C^\top x + v \\ v & \geq & 0. \end{array} \tag{11}$$

Since $l(\epsilon)$ is positive at every entry, an optimal solution to (11) has $v = F^\top q - C^\top x$. Also, feasible solutions y to (10) and (q, v) to (11) are optimal if and only if the two objective function values are equal, i.e.,

$$x^\top (Cy) = q^\top f - v^\top l(\epsilon) \tag{12}$$

which, using $f = Fy$ and $F^\top q - {}^\top x = v$ is equivalent to

$$v^\top (y - l(\epsilon)) = 0. \tag{13}$$

The Nash equilibrium condition of x and y being best responses to each other is then expressed by combining the constraints of (6), (7), (9), (10), (11) and (13), leading to the linear complementarity program (5). ♠

Still following Koller, Megiddo and Von Stengel, we now put the derived linear complementarity program in *standard form* in order to apply Lemke's algorithm. We define $\tilde{x} = x - k(\epsilon)$, $\tilde{y} = y - l(\epsilon)$, $\tilde{e} = e - Ek(\epsilon)$, $\tilde{f} = f - Fl(\epsilon)$, $\alpha = -Al(\epsilon)$ and $\beta = -C^\top k(\epsilon)$ and rewriting $p = p' - p''$ with $p', p'' \geq 0$ and $q = q' - q''$ with $q', q'' \geq 0$, we can write the system (5) as

$$\begin{array}{rcl} z & \geq & 0 \\ w = Mz + b(\epsilon) & \geq & 0 \\ z^\top w & = & 0 \end{array} \tag{14}$$

where

$$M = \begin{pmatrix} & -A & E^\top & -E^\top & & \\ -C^\top & & & & F^\top & -F^\top \\ -E & & & & & \\ E & & & & & \\ & -F & & & & \\ & F & & & & \end{pmatrix}, \tag{15}$$

$z = (\tilde{x}, \tilde{y}, p', p'', q', q'')^\top$, and $b(\epsilon) = (\alpha, \beta, \tilde{e}, -\tilde{e}, \tilde{f}, -\tilde{f})^\top$. For $\epsilon = 0$, the LCP is identical to the one derived by Koller, Megiddo and von Stengel (1996). In particular, M is the *same* matrix M as in their Equation (2.10). Thus, we have an LCP in standard form, completely analogous to the LCP described by Koller, Megiddo and von Stengel, the only difference being that our vector $b(\epsilon)$ is a formal perturbation of their vector b, parametrized by the parameter $\epsilon > 0$. Note that each entry of $b(\epsilon)$ is a polynomial in ϵ.

We now apply Lemke's algorithm to this parametrized program exactly as Koller, Megiddo and von Stengel applied Lemke's algorithm to their program, while keeping ϵ a symbolic indeterminate, representing a sufficiently small number. That is, we generalize the system (14) to

$$\begin{aligned} z &\geq 0 \\ w = Mz + dz_0 + b(\epsilon) &\geq 0 \\ z^\top w &= 0 \end{aligned} \tag{16}$$

where $z_0 \geq 0$ is a new scalar variable and $d = (1, 1, 1, .., 1)^\top$. Solutions to (14) are now exactly the solutions to (16) for which $z_0 = 0$. In the case of a non-degenerate LCP (we shall deal with the degenerate case later), Lemke's algorithm works by traversing a sequence of solutions to the system of linear equations

$$Iw - dz_0 - Mz = b(\epsilon) \tag{17}$$

where I is an $n \times n$ identity matrix. The solutions considered are *basic* solutions, that is, the vector $b(\epsilon)$ is represented as a linear combination of n linearly independent columns of the matrix $[I, -d, -M]$. These columns form a non-singular $n \times n$ submatrix B. The corresponding variables are the *basic variables* of the solution (the *basis*). The nonbasic variables all have value 0. For any basic solution and for any $1 \leq i \leq n$, at most one of z_i, w_i is nonbasic. Thus, a basic solution where z_0 is a nonbasic variable is a solution to (14). A basic solution where z_0 is a basic variable has for some i both z_i as w_i as nonbasic variables. Furthermore, it is an invariant of the algorithm that one of these (say, w_i) just left the basis. The other one (in that case, z_i) will enter the basis in the next iteration of the algorithm. This iteration is given by the following *pivoting operation*, changing the basis and the corresponding basic solution: Let h be the column of $[I, -d, -M]$ corresponding to the entering variable. To find the leaving variable, the algorithm picks the largest value of v so that

$$B^{-1}b(\epsilon) - B^{-1}hv \geq 0. \tag{18}$$

This makes some component of $B^{-1}b(\epsilon) - B^{-1}hv$ zero, and the corresponding basic variable leaves the basis. The matrix B is now updated according to the new basis.

In our case, each entry of $b(\epsilon)$ is a formal polynomial in ϵ, with ϵ to be interpreted as a sufficiently small number. We should check that the pivoting operation is still well-defined and can be performed algorithmically. Indeed, for all sufficiently small $\epsilon > 0$, the choice of leaving variable is independent of $\epsilon > 0$. Furthermore, the correct choice can be found by comparing the polynomials $f_j(\epsilon) = (B^{-1}b(\epsilon))_j / (B^{-1}h)_j$ and finding the smallest according to *lexicographic* order, i.e. the order $\leq_{\text{lex}}$ defined by

$$\sum a_i \epsilon^i \leq_{\text{lex}} \sum a'_i \epsilon^i \Leftrightarrow ([\forall i : a_i = a'_i] \vee [a_t < a'_t \text{ for } t = \min\{i | a_i \neq a'_i\}]) \tag{19}$$

and this comparison can indeed be performed algorithmically while keeping ϵ an indeterminate. Pivoting in this way, we have that if Lemke's algorithm terminates with a solution for which

$z_0 = 0$, this solution is an expression for a valid solution to (14) for all sufficiently small $\epsilon > 0$. The entries of vectors x, y found are formal polynomials in ϵ. The corresponding behavior probabilities are then rational functions in ϵ, as stated in Fact 4. Thus, to complete the description of the algorithm and the proof of Fact 4, we should ensure that Lemke's algorithm will indeed terminate with such a solution. As Koller, Megiddo and von Stengel, the concerns we have to deal with are *degeneracy* and *ray termination*.

Degeneracy occurs when the leaving variable is not uniquely defined, with two or more entries of $B^{-1}b(\epsilon) - B^{-1}hv$ being zero for the maximum v for which $B^{-1}b(\epsilon) - B^{-1}hv$ is non-negative. Degeneracy may cause Lemke's algorithm to cycle. The only known way for avoiding degeneracy in general is to apply a symbolic perturbation to the LCP. In particular, Koller, Megiddo and von Stengel replace their condition $B^{-1}b - B^{-1}hv \geq 0$ (the condition analogous to (18) but with b a vector of unparametrized values) with

$$B^{-1}b_\epsilon - B^{-1}hv \geq 0 \tag{20}$$

where $b_\epsilon = b + (\epsilon, \epsilon^2, \epsilon^3, \ldots, \epsilon^n)^\top$ and $\epsilon > 0$ is a formal parameter, representing a sufficiently small value, exactly as in our parametrized LCP. That is, even when solving *unparametrized* LCPs one has to consider formally perturbed programs and do pivoting operations using a lexicographic rule as we describe above[1]. It is easy to see that the perturbation of (20) ensures non-degeneracy if b is a vector of unparametrized values. When using the same method on our parametrized LCP, we have to deal with two perturbations, the one introduced by the parameter $\epsilon > 0$ in the formulation of our LCP and another introduced to eliminate degeneracy, the latter still being necessary as the program (16) may well be degenerate. We need to ensure that the latter technical perturbation does not interfere with the former meaningful one. The simplest way is to make the latter perturbation an order of magnitude smaller than the former and replace the condition (18) with

$$B^{-1}b'(\epsilon) - B^{-1}hv \geq 0 \tag{21}$$

where $b'(\epsilon) = b(\epsilon) + (\epsilon^{d+1}, \epsilon^{d+2}, \epsilon^{d+3}, \ldots, \epsilon^{d+n})^\top$ and d is the maximum degree of the polynomials in ϵ that define b. Again, it is easy to see that the perturbation (21) ensures non-degeneracy. Note that the original perturbation should appear in the solution we eventually output while the perturbation introduced to avoid degeneracy should not. That is, the solution we eventually produce and output is the one satisfying the equation $Iw - dz_0 - Mz = b(\epsilon)$ (with $z_0 = 0$) and not the one satisfying $Iw - dz_0 - Mz = b'(\epsilon)$.

Finally, we have to deal with the possibility of ray termination. By definition, this happens when $B^{-1}b - B^{-1}hv$ is non-negative for arbitrarily large values of v and hence no leaving variables can be found. Exactly the same argument as given in Theorem 4.1, Lemma 4.2, Lemma 4.3, and Theorem 4.4 of Koller, Megiddo and von Stengel (1996) shows that this will not happen when Lemke's algorithm is applied to the LCP given by (14), if a sufficiently large constant is subtracted from all entries of A and C and if the lexicographic perturbation method is used to eliminate degeneracy. Note that their LCP is a special case of ours. As their proof goes through without any modification whatsoever for our more general parametrized LCP, we shall not repeat it here.

This concludes our description of our application of Lemke's algorithm to solve the general-sum case.

[1] This also means that the modification necessary for turning a standard implementation of Lemke's algorithm into one solving LCPs parametrized as ours is in fact minor.

5 The zero-sum case

5.1 The zero-sum case using the simplex algorithm

In this section we consider the zero-sum case, i.e., we assume $C = -A$. We derive a variant of algorithm of the last section based on linear programming and the simplex algorithm rather than linear complementarity programming and Lemke's algorithm.

We mimic the derivation of Koller, Megiddo and von Stengel (1994) for the unperturbed case. If G is a zero-sum game, so is $G(\epsilon)$, and so the set of Nash equilibria of $G(\epsilon)$ is the Cartesian product of the sets of minimax strategies for each of the two players. First we derive the minimax strategy for Player II. Suppose that a strategy of Player II is fixed and given in sequence form by a realization plan y. As in the proof of Theorem 5, a best response by Player I as a realization plan x is then given by

$$\begin{array}{rcl} \max\limits_{x} & x^\top(Ay) & \text{so that} \\ Ex & = & e \\ x & \geq & k(\epsilon) \end{array} \tag{22}$$

whose dual program is

$$\begin{array}{rcl} \min\limits_{p,u} & p^\top e - u^\top k(\epsilon) & \text{so that} \\ E^\top p & \geq & Ay + u \\ u & \geq & 0. \end{array} \tag{23}$$

The value of the optimal solution to (23) expresses the expected payoff that Player I can achieve in the perturbed game $G(\epsilon)$ if Player II plays using strategy y. Player II wants to choose a valid realization plan y so that this is minimized. His minimax strategy is thus given by

$$\begin{array}{rcl} \min\limits_{p,u,y} & p^\top e - u^\top k(\epsilon) & \text{so that} \\ E^\top p & \geq & Ay + u \\ Fy & = & f \\ y & \geq & l(\epsilon) \\ u & \geq & 0. \end{array} \tag{24}$$

Reversing the roles of the two players and arguing completely analogously, we obtain the minimax strategy for Player I.

$$\begin{array}{rcl} \max\limits_{q,v,x} & q^\top f + v^\top l(\epsilon) & \text{so that} \\ F^\top q & \leq & A^\top x - v \\ Ex & = & e \\ x & \geq & k(\epsilon) \\ v & \geq & 0. \end{array} \tag{25}$$

Since (24) and (25) are dual linear programs they describe an equilibrium for the perturbed game $G(\epsilon)$. Also, because of this duality, we only need to apply the simplex algorithm to one of these programs to solve them both.

To easily apply the simplex algorithm to the program, we convert it into standard form. For convenience, by adding a constant to every leaf of the original extensive form, we may transform the game into one where every payoff for Player I is positive and hence every payoff for Player II is negative. Then, a reformulation of (24) is

$$\begin{array}{rcl} \max\limits_{p,u,y} & -p^\top e + u^\top k(\epsilon) & \text{so that} \\ -E^\top p + Ay + u & \leq & 0 \\ -Fy & \leq & -f \\ -y & \leq & -l(\epsilon) \\ p,u,y & \geq & 0. \end{array} \tag{26}$$

By introducing slack variables, we put (25) in the following standard form.

$$\begin{array}{rcl} \max\limits_{z} & c(\epsilon)^\top z & \text{so that} \\ w = Mz + b(\epsilon) & \geq & 0 \\ z & \geq & 0. \end{array} \tag{27}$$

Here, $b(\epsilon)$ and $c(\epsilon)$ are vectors with entries being either constants (i.e., not depending on ϵ) or some power ϵ^j.

For details about the simplex algorithm, we refer to Schrijver (1987). To solve (27), the simplex algorithm traverses a sequence of basic solutions to the system of linear equations

$$Iw - Mz = b(\epsilon) \tag{28}$$

where I is an $n \times n$ identity matrix. That is, the vector $b(\epsilon)$ is represented as a linear combination of n linearly independent columns of the matrix $[I, -M]$. These columns form a non-singular $n \times n$ submatrix B. The corresponding variables are the *basic variables* of the solution (the *basis*). The nonbasic variables all have value 0. As in Lemke's algorithm, we go from one basic solution to the next by a pivoting operation. In the simplex algorithm, the entering variable is chosen in the following way. Let $c_B(\epsilon)$ be the entries of $c(\epsilon)$ corresponding to basic variables. Given a non-basic variable, we let $d(\epsilon) = 0$ if the variable considered is a slack variable w_j and otherwise we let $d(\epsilon)$ be the corresponding entry of $c(\epsilon)$. We let h be the column of $[I, -M]$ corresponding to this variable. The variable is allowed to enter the basis if

$$d(\epsilon) - c_B(\epsilon)^\top B^{-1} h > 0. \tag{29}$$

In case of more than one choice of such variables, *Bland's rule* prescribes choosing the one with smallest index in some fixed enumeration of the variables. Bland's rule ensures termination of the simplex algorithm. If (29) fails to hold for all non-basic variables, the current solution is optimal. Having found a variable to enter the basis, the simplex algorithms picks the leaving variable in exactly the same way as Lemke's algorithm, i.e., by picking the largest value of v so that

$$B^{-1} b(\epsilon) - B^{-1} h v \geq 0. \tag{30}$$

This makes some component of $B^{-1}b(\epsilon) - B^{-1}hv$ zero, and the corresponding basic variable leaves the basis. Again, in case of a tie, Bland's rule prescribes choosing the one with smallest index.

The simplex algorithm as described will solve any given bounded linear program to optimality so our only concern is to check that the algorithm can be carried out on our parametric program (27). Here, $b(\epsilon)$ and $c(\epsilon)$ are formal polynomial in ϵ, with ϵ to be interpreted as a sufficiently small number. Indeed, the conditions (29) and (30) are both independent of ϵ when $\epsilon > 0$ is sufficiently small and both can be checked by comparing easily computed polynomials according to lexicographic order, exactly as in the case of Lemke's algorithm. Also, for the program at hand, it is straightforward to find an initial basis so that the corresponding basic solution is feasible for all sufficiently small $\epsilon > 0$. Thus, we can solve the parametrized program (27) to optimality using the simplex algorithm. The primal solution of the final tableau of the simplex algorithm yields a realization plan y for Player II and the dual solution of the final tableau yields a realization plan x for Player I, both being vectors of formal polynomials in ϵ. The corresponding behavior probabilities are then rational functions in ϵ, as stated in Fact 4.

This concludes our description of our application of the simplex algorithm to solve the zero-sum case.

5.2 The zero-sum case using generic linear programming solver

A major concern of computer science is the existence of *polynomial time algorithms* for solving computational problems. A polynomial time algorithm takes as input a string over $\{0, 1\}$ and produces another such string using a number of atomic Boolean operations (such as taking the logical AND of two bits) which is polynomial bounded in the length of the input string. For details, see, e.g., Chapter 2.4 of Schrijver (1987).

For the equilibrium computation problems of the present paper, the input string is a description of the extensive form game. In order to ensure finiteness of such a description, we assume that payoffs are rational numbers. If so, we can in fact assume without loss of generality that they are integers. The output is a description of the behavior probabilities of the equilibrium. For the two-player games we consider, these can be assumed to be rational numbers when payoffs are integers so that the output string is also finite.

While Lemke's algorithm and the Lemke-Howson algorithm are regarded as fairly efficient in practice, they are known *not* to be polynomial time algorithms (Savani and von Stengel, 2004). In fact, no polynomial time algorithm for computing a Nash equilibrium is known, not even for a bimatrix game. Also, the simplex algorithm is not a polynomial time algorithm, but alternative polynomial time algorithms for linear programming such as the ellipsoid algorithm (Khachiyan, 1979) and interior point algorithms (Karmarkar, 1984) are known. Thus, it follows from the work of Koller, Megiddo and von Stengel that a *Nash* equilibrium of a two-player extensive-form *zero-sum* game of perfect recall can be found in polynomial time.

Since the ellipsoid algorithm and interior point algorithms solve *unparametrized* linear programs, it does not follow immediately from the previous section that we have a polynomial time algorithm for finding a quasi-perfect equilibrium. However, in this section we point out that these algorithms can in fact also be used to solve a parametrized linear program of the form (27) in polynomial time. In particular, we prove:

Proposition 6 *A quasi-perfect equilibrium of a given two-player extensive-form zero-sum game of perfect recall can be found in polynomial time.*

Proof The idea is to apply an arbitrary polynomial time algorithm for solving linear programs to (27) with the parameter ϵ replaced with a concrete, very small value. We shall then argue that we can reconstruct the optimal parametric solution from the concrete optimal solution obtained.

Let V be the maximum absolute value of any coefficient occurring in (27). As in Section 5.1, we let n be the number of equations of (27), i.e., the size of the basis in a basic solution. Let $\epsilon^* = \frac{1}{2}n^{-n-1}V^{-2n-1}$. Consider the program (27) with the parameter ϵ substituted with the concrete value ϵ^*. It is easy to see that this unparametrized program is feasible. Let x' be a basic feasible and optimal solution. We claim that the corresponding basic solution (i.e., the solution with the same set of basic variables) to the parametrized program if also feasible and optimal for all values of $\epsilon \leq \epsilon^*$.

To verify the claim, we observe that a basic solution with basis matrix B is feasible for a particular value of ϵ if $B^{-1}b(\epsilon) \geq 0$. It is optimal if the inequality (29) is violated for all non-basic variables. By Cramer's rule and the Hadamard inequality, these feasibility and optimality conditions are equivalent to a system of inequalities

$$\forall i : \sum_{j=0}^{d} k_{ij}\epsilon^j \geq 0 \tag{31}$$

where k_{ij} are rational numbers that can be written as fractions with numerator of absolute value at most $n^{n/2+1}V^{n+1}$ and denominator $|\det B| \leq n^{n/2}V^n$. By multiplying the equations by $|\det B|$, we may in fact assume that k_{ij} are integers of absolute value at most $n^{n+1}V^{2n+1} = 1/2\epsilon^*$. Thus, for any $\epsilon \leq \epsilon^*$, the inequalities (31) are equivalent to the conditions

$$\forall i : [\forall j : k_{ij} = 0] \vee [k_{\min\{j|k_{ij}\neq 0\}} > 0] \tag{32}$$

which do not depend on ϵ, as was to be proved.

Our polynomial time algorithm works as follow. We run a polynomial time unparametrized linear programming solver on the program (27) with the parameter ϵ being fixed to the value ϵ^*. Note that the number of bits in the description of ϵ^* is polynomial in the size of the program, so this can be done in polynomial time. We obtain an optimal solution to the program. Given any such optimal solution, it is straightforward to efficiently obtain a *basic* optimal solution, including a partition of the variables into basic and non-basic ones. Then, we obtain the corresponding solution to the parametrized program (27) as a vector of polynomials in ϵ by computing the inverse of the basis matrix B^{-1} and using equation (28). By the claim, this corresponding solution is indeed feasible and optimal for all $\epsilon \leq \epsilon^*$ and hence a minimax strategy for Player II in $G(\epsilon)$ in realization weights for sufficiently small values of $\epsilon > 0$. We similarly compute a parametrized minimax strategy for Player I and derive a quasi-perfect equilibrium in behavior strategies as explained in Section 3. ♠

We remark that even though the algorithm in the proof of the proposition is polynomial time, it is quite impractical due to the small value of ϵ^* which leads to the resulting unparametermized linear programs containing coefficients with a very large number of digits. Thus, even though interior point methods is a practical alternative to the simplex method for solving unparametrized linear programs, we don't know how to apply these methods to practically compute a quasi-perfect equilibrium for a zero-sum game. In the algorithm of Section 5.1, ϵ is kept symbolic and coefficients are of the same order of magnitude as the payoffs of the game to be solved. This leads to a much more practical algorithm, even though it is not polynomial time.

Acknowledgements

We would like to thank Daphne Koller, Bernhard von Stengel, Andrew McLennan, Robert Wilson, Ted Turocy and the anonymous reviewers for very helpful discussions and comments on preliminary versions of this paper. One such version (Miltersen and Sørensen, 2006) was presented at the 17th annual ACM-SIAM Symposium on Discrete Algorithms and appears in the proceedings of that conference. The research of Peter Bro Miltersen is supported by BRICS, a center of the Danish National Research Foundation, by a grant from the Danish Science Research Council and by Center for Algorithmic Game Theory, funded by the Carlsberg foundation.

References

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., and Szafron, D.: Approximating Game-Theoretic Optimal Strategies for Full-scale Poker, in Proceedings of Eighteenth International Joint Conference on Artificial Intelligence: Springer 2003

van Damme, E.: 1984, A relation between perfect equilibria in extensive form games and proper equilibria in normal form games, International Journal of Game Theory **13**, 1–13 (1987)

van Damme, E.: Stability and Perfection of Nash Equlibria, 2nd ed: Springer-Verlag 1991

Gilpin, A. and Sandholm, T.: Finding Equilibria in Large Sequential Games of Incomplete Information, Technical Report CMU-CS-05-158: Carnegie Mellon University 2005

Hillas, J. and Kohlberg, E.: Foundations of Strategic Equilibria, in R. Aumann and S. Hart (eds.), Handbook of Game Theory, vol. 3: Elsevier Science 2002

Karmarkar, N.: A New Polynomial-Time Algorithm for Linear Programming, Combinatorica **4(4)**, 373–395 (1984)

Khachiyan, L.: A Polynomial Algorithm in Linear Programming, Soviet Mathematics Doklady **20**, 191–194 (1979)

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

Koller, D., Megiddo, N., and von Stengel, B.: Efficient Computation of Equilibria for Extensive Form Games, Games and Economic Behavavior **14**, 247–259 (1996)

Koller, D. and Pfeffer, A.: Representations and Solutions for Game-Theoretic Problems, Artificial Intelligence **94(1–2)**, 167–215 (1997)

Kreps, D. M. and Wilson, R.: Sequential Equilibria, Econometrica **50(4)**, 863–894 (1982)

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

McKelvey, R. and McLennan, A.: Computation of equilibria in finite games, in H. Amman, D. Kendrick, and J. Rust (eds.), Handbook of Computational Economics: Elsevier 1996

Mertens, J.-F.: Two examples of strategic equilibrium, Games and Economic Behavior **8**, 378–388 (1995)

Miltersen, P. B. and Sørensen, T. B.: Computing sequential equilibria for two-player games, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms: ACM 2006

Savani, R. and von Stengel, B.: Exponentially Many Steps for Finding a Nash Equilibrium in a Bimatrix Game, in Proceedings to 45th Annual IEEE Symposium on Foundations of Computer Science: IEEE Computer Society Press 2004

Schrijver, A.: Theory of Linear and Integer Programming: Wiley 1987

von Stengel, B.: Efficient computation of behavior strategies, Games and Economic Behavior **14**, 220–246 (1996)

von Stengel, B., van den Elzen, A., and Talman, D.: Computing Normal Form Perfect Equilibria for Extensive Two-Person Games, Econometrica **70(2)**, 693–715 (2002)