International Journal of Game Theory (1996) 25: 73–92

Finding Mixed Strategies with Small Supports in Extensive Form Games[1]

DAPHNE KOLLER

Computer Science Division, 387 Soda Hall, University of California, Berkeley, CA 94720, USA

NIMROD MEGIDDO

IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099, and School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel

Abstract: The complexity of algorithms that compute strategies or operate on them typically depends on the representation length of the strategies involved. One measure for the *size* of a mixed strategy is the number of strategies in its *support* – the set of pure strategies to which it gives positive probability. This paper investigates the existence of "small" mixed strategies in extensive form games, and how such strategies can be used to create more efficient algorithms. The basic idea is that, in an extensive form game, a mixed strategy induces a small set of *realization weights* that completely describe its observable behavior. This fact can be used to show that for any mixed strategy μ, there exists a realization-equivalent mixed strategy μ' whose size is at most the size of the game tree. For a player with imperfect recall, the problem of finding such a strategy μ' (given the realization weights) is NP-hard. On the other hand, if μ is a behavior strategy, μ' can be constructed from μ in time polynomial in the size of the game tree. In either case, we can use the fact that mixed strategies need never be too large for constructing efficient algorithms that search for equilibria. In particular, we construct the first exponential-time algorithm for finding all equilibria of an arbitrary two-person game in extensive form.

1 Introduction

When attempting to implement various game-theoretic ideas, one often needs to construct algorithms that operate on strategies in various ways. For example, it may be necessary to compute strategies satisfying certain criteria (for example, equilibrium strategies), to store strategies for future use, and to use strategies for further computation or for playing the game. Clearly, the space required to represent the strategy often affects the complexity of the resulting algorithms. In this paper, we present a technique for reducing the amount of space required to represent mixed strategies for *extensive-form games.*

[1] Research supported in part by ONR Contract N00014-91-C-0026, by the Air Force Office of Scientific Research (AFSC) under Contract F49620-91-C-0080, and by a University of California President's Postdoctoral Fellowship. Some of this work was done while Daphne Koller was at Stanford University. The United States Government is authorized to reproduce and distribute reprints for governmental purposes.

0020-7276/96/1/73-92$2.50 © 1996 Physica-Verlag, Heidelberg

The extensive form is a natural way of representing a game: as a decision tree with information structure. For this form, three types of strategies for a player (say player 1) have been defined.[2] A *pure strategy* determines completely the moves of player 1 at each of his information sets. A *mixed strategy* defines a probability distribution over the entire set of pure strategies. When a mixed strategy is played, a pure strategy is chosen according to this distribution before the game starts, and then followed throughout the game. A *behavior strategy* randomizes locally rather than globally: for each information set, it prescribes a probability distribution over the possible moves.

These different types of strategies differ considerably in terms of the space required to represent them. A pure strategy has one entry for each information set of the player. Its size is clearly linear in the size of the game tree. A behavior strategy assigns a probability to each possible decision of the player. Thus, it also can be represented in linear space. A mixed strategy assigns a probability to each pure strategy. The number of pure strategies is usually exponential in the size of the game tree [7, 4]. Thus, the size of a mixed strategy may be exponential in the size of the tree.

In many useful mixed strategies, however, only a small number of pure strategies receive positive probabilities. For a mixed strategy μ, the set of pure strategies which receive positive probability under μ is called the *support* of μ. We use the *size* of μ to denote the number of pure strategies in the support of μ. Mixed strategies whose support is "small" can be specified with a sparse representation. The relatively "manageable" size of these strategies may reduce the space and time complexities of algorithms handling them. This idea was first utilized by Wilson [14] in his modified version of the Lemke-Howson algorithm [9]. This algorithm searches the space of mixed strategies pairs for an equilibrium of a general two-person game. Wilson's variant represents the strategies encountered in the search sparsely; i.e. it maintains only the pure strategies in their support, and generates additional pure strategies, as needed, directly from the game tree. Wilson justifies his algorithm as follows:

> The advantages of such a modification will derive from the incidence of validity of three propositions commonly verified in computational experience on practical problems: (1) in practice most games arise in extensive form; (2) even simply described extensive forms commonly generate normal forms of enormous size, since the number of pure strategies increases exponentially with the number of information sets...; (3) the frequency of equilibria using only a very few of the available pure strategies is very high...

Wilson's motivation, as we see, arose from "computational experience on practical problems." In this paper, we describe and formally prove a result that, among other things, implies Wilson's proposition (3).

[2] From here on, without loss of generality, we describe strategies in terms of player 1.

We begin by observing that a mixed strategy contains a large amount of information, most of which is irrelevant. The only relevant aspect of a mixed strategy is the observable behavior that it induces on the nodes of the game tree. This motivates the following definition. A strategy μ of player 1 uniquely defines a *weight distribution* on the nodes of the game tree. Intuitively, the *realization weight* given to a node a by μ is the probability that a is not ruled out by player 1 when playing μ (note that a can still be ruled out by the strategy of one of the other players). This is exactly the probability that a pure strategy chosen according to μ is consistent with the player 1 choices on the path to a. Given the weight distribution defined by μ, and any strategy combination of the other players, we can determine the probability of reaching any node in the tree without referring back to μ. That is, the weight distribution on the nodes completely specifies the observable behavior of μ. Realization weights were first introduced by Koller and Megiddo [4] in the context of perfect recall games. In their algorithm, and in the more recent ones of von Stengel [12] and of Koller, Megiddo, and von Stengel [6], the realization weights corresponding to equilibrium mixed strategies are computed directly. This relies on the fact that, for games with perfect recall, realization weights can be described using a small system of linear equations. As we discuss below, this is not the case for games with imperfect recall, a fact that prevents the application of these techniques to the general case. In this paper, we utilize realization weights in a very different way. Rather than using them directly as a new strategic representation, we use them indirectly, as a tool for constructing small mixed strategies. This allows us to apply our techniques to arbitrary games.

Our main result is that mixed strategies need never be too large. That is, for every mixed strategy μ there exists an equivalent "small" mixed strategy. We define two strategies of player 1 to be *equivalent* if for every payoff function and every strategy of the other players, they result in the same payoff.[3] It can easily be shown that two strategies are equivalent if and only if they induce the same weight distribution over the leaves of the game tree. Hence, a weight distribution represents an entire class of equivalent strategies. This observation is the basis for our result. In order to state it formally, let T be a game tree, and let Z be the set of leaves in T. We measure the size of the tree in terms of the cardinality of Z and denote that number by $|T|$; this is justified since if each decision node has at least two choices, then more than half of all the nodes of T are leaves. In Section 2 we show that for any mixed strategy μ for player 1, there exists an equivalent strategy μ' whose size does not exceed $|T|$ (the number of leaves in the tree). Moreover, there exist games where some mixed strategies do not have equivalent strategies of smaller size. We therefore define a mixed strategy to be *small* if its size does not exceed $|T|$.[4]

[3] This standard concept is sometimes called *payoff equivalence* and sometimes *realization equivalence*.

[4] Our result is actually stronger, in that the size of the strategies is often even smaller than $|T|$. For simplicity, we chose to define "small" based on $|T|$. See Section 2 for further discussion.

Next, we investigate the issue of finding a small strategy that is equivalent to some given mixed strategy. Given μ, this can be done quite easily by a process that iteratively removes strategies one by one from the support of μ. Unfortunately, the size of μ can be exponential in the size of the game tree, so that this process is typically of little practical use. In Section 3 we investigate an alternative approach: finding a small mixed strategy directly from the weight distribution on the leaves. We show that, in general, although the size of the input is now small (linear in the size of the tree), this does not help with respect to the complexity of the problem. The problem of deciding whether there even exists a mixed strategy that induces a given weight distribution is NP-complete.

The situation is different, however, for the case of behavior strategies. Unlike general mixed strategies, it is easy to decide whether there exists a behavior strategy that induces a given weight distribution. In fact, such a behavior strategy can be computed in polynomial time from the weight distribution (see also [4]). In many interesting problems, behavior strategies play an important role. A player is said to have *perfect recall* if she remembers throughout the play everything she has known and done. Kuhn [8] showed that for a player with perfect recall, every mixed strategy has an equivalent behavior strategy. Thus, for such a player, it suffices to investigate only behavior strategies. In [4], we also make the point that for a player with imperfect recall, the use of an arbitrary mixed strategy (one not derived from a behavior strategy) may violate the spirit of the imperfect recall requirement. Thus, in this case one also often wishes to restrict attention to behavior strategies.

In Section 4 we show how a small mixed strategy can be computed in polynomial time from a behavior strategy (or from the weight distribution induced by the behavior strategy). In particular, for any strategy of a player with perfect recall, a small mixed strategy can be found in polynomial time. For example, using the results of [4, 12], a small *optimal* mixed strategy can be found for any player in a two-person zero-sum game with perfect recall. Similarly, using the results of [6], an equilibrium pair of small mixed strategies can be found in any two-person game with perfect recall.

In general, the representation of a small mixed strategy might not be smaller than the representation of the behavior strategy that induces it. However, as many algorithms are based on mixed strategies, the ability to construct efficiently a small mixed strategy is often useful. In [5], for example, we demonstrate how the techniques of this paper can be applied to de-randomization of algorithms. But even the fact itself that mixed strategies need never be too large can help us in constructing more efficient algorithms. In Section 5 we demonstrate this by constructing an exponential time algorithm for enumerating all equilibria in two-person general-sum games. To our knowledge, this was the first exponential-time algorithm for this problem, and is still the only one that works for games with imperfect recall. (For the case of perfect recall games, the recent algorithms of [12] and [6] are better.) We also briefly discuss how this result can be used to construct an efficient variant of Wilson's algorithm [14]. In general, we believe that these techniques can also be used to construct efficient algorithms for many problems involving mixed strategies.

2 Existence of Small Strategies

As we discussed in the introduction, in extensive form games, mixed strategies contain a large amount of irrelevant information. This leads us to hope that we might be able to find "small" mixed strategies where this redundancy is eliminated. In this section, we formalize this intuition. Let T be a game tree for some number of players (the number does not matter). For the rest of the paper, we restrict attention to player 1 (often referred to as "the player"), and to the problem of constructing small mixed strategies for player 1. Assume that the player's information sets in T are $u_1, \ldots, u_n$, and that the set of choices available at information set u_j is C_j. Let $\Sigma = C_1 \times C_2 \times \cdots \times C_n$ denote the set of pure strategies for the player. For a pure strategy $s \in \Sigma$, let $s_j \in C_j$ denote the choice made by s at information set u_j, so $s = \langle s_1, \ldots, s_n \rangle$. For example, in the game tree T^* of Figure 1, player 1 has two information sets, u_1 and u_2, with $C_1 = \{l, r\}$ and $C_2 = \{c, d\}$. Therefore, $\Sigma = \{l, r\} \times \{c, d\}$. Note that the tree T^* is not completely specified. We did not describe the payoffs at the leaves nor the player to move at the root of the tree; these will be irrelevant to our discussion.

Given a strategy μ for player 1 in the game T, we would like to find an *equivalent* mixed strategy that is more compact.

Definition 2.1: For a mixed strategy μ and a pure strategy s, let $\mu(s)$ denote the probability that μ assigns to the pure strategy s. The *support* of μ, denoted $\mathrm{Supp}(\mu)$, is the set of pure strategies $\{s \in \Sigma : \mu(s) > 0\}$. The *size* of μ is the cardinality $|\mathrm{Supp}(\mu)|$ of its support.

For example, consider a mixed strategy μ^* over T^* such that:

$$\mu^*(\langle l, c \rangle) = 1/7$$
$$\mu^*(\langle l, d \rangle) = 2/7$$
$$\mu^*(\langle r, c \rangle) = 4/7$$
$$\mu^*(\langle r, d \rangle) = 0.$$

Here, $\mathrm{Supp}(\mu^*) = \{\langle l, c \rangle, \langle l, d \rangle, \langle r, c \rangle\}$.

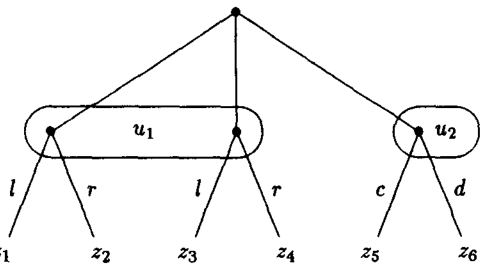

Fig. 1. A game tree T^*.

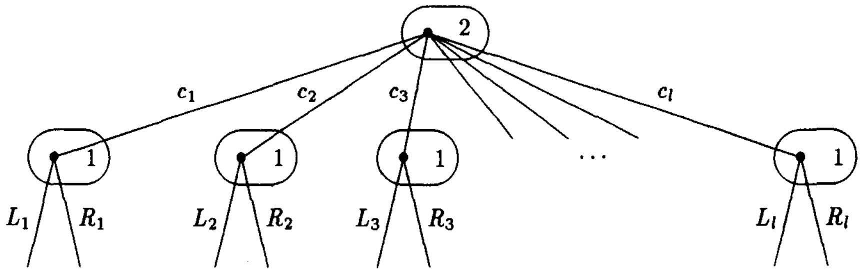

Fig. 2. A game tree with exponentially many pure strategies.

Our goal is, given a mixed strategy μ, to find another mixed strategy with a "small" support, that performs the same as μ in all situations.

Definition 2.2: Two mixed strategies μ_1 and μ_2 are said to be *equivalent* if for all payoff functions, all probability distributions at the chance nodes, and all strategies of the other players, the payoffs to all players under μ_1 and under μ_2 are identical.

At first glance it seems very difficult to check whether two mixed strategies are equivalent; it appears that it might be necessary to check infinitely many (or at least very many) variants of the game, corresponding to different assignments of payoffs and probabilities of the chance moves. However, it turns out that equivalence can be defined in terms of the following simple concepts.

Consider the information sets intersected by the path from the root to some node a. In each of these information sets, only one decision leads to a. We can therefore characterize the set of pure strategies which may reach the node a as follows.

Definition 2.3: We say that a pure strategy s *potentially reaches* a node a if, for every information set u_j on the path from the root to a, the strategy s at u_j takes the decision s_j leading to a. Let $R(a)$ denote the set of pure strategies $\{s \in \Sigma$:*s potentially reaches a*$\}$.

Note that even if s potentially reaches a, it is not necessarily the case that a is actually reached in a particular play of the game. This depends, of course, on the strategies of the other players and on the chance moves. The fact that s potentially reaches a means only that s does not unilaterally rule out the possibility of reaching a. For example, consider the strategy $s = \langle ld \rangle$ over the tree T^*. This strategy potentially reaches z_1, z_3, and z_6. Clearly, the node that is actually reached in a particular game depends on the move taken at the root. However, this is independent of the strategy of player 1. It is easy to see that:

$$R(z_1) = R(z_3) = \{\langle l,c \rangle, \langle l,d \rangle\}$$
$$R(z_2) = R(z_4) = \{\langle r,c \rangle, \langle r,d \rangle\}$$
$$R(z_5) = \{\langle l,c \rangle, \langle r,c \rangle\}$$
$$R(z_6) = \{\langle l,d \rangle, \langle r,d \rangle\}.$$

The notion of potential reachability is often used to reduce the space of pure strategies. Consider some information set u. If a pure strategy s does not potentially reach any node in u, then the decision taken by s at u is *irrelevant*. The information set u is never reached in play, so the decision there cannot affect the outcome of the game. Pure strategies that differ only in choices at irrelevant information sets have been called *equivalent* by Kuhn [8] and even identified [7]. This identification can be done by leaving the choices at the irrelevant information sets blank. In this formulation, a pure strategy would represent an entire equivalence class of pure strategies. This reduction of the space of pure strategies forms the basis for the *reduced normal form* of an extensive game. Further reductions for a particular given payoff function have been considered by Dalkey [3] and by Swinkels [11], but these provide no additional savings in the context of a generic game.

Unfortunately, the savings provided by the reduced normal form are often limited. Consider a game where the player has *parallel information sets* – ones where the player's actions cannot affect which is reached. For example u_1 and u_2 in T^* are parallel information sets. For a more extreme example, consider Figure 2, where player 1 has l parallel information sets. The number of pure strategies in this case is 2^l, and none can be eliminated as equivalent to another. Hence, even the reduced normal form is often exponential in the size of the game tree.

Our approach to reducing the size of mixed strategies is completely different. Rather than eliminating or identifying pure strategies, we define equivalence classes of *mixed strategies*. This process will allow us to find, in each equivalence class, a representative mixed strategy which is "small". We note that our technique also recognizes equivalence of pure strategies (in the above sense), so that it can be seen as generalizing the ideas behind the reduced normal form. The main idea we use is the following.

Definition 2.4: Let μ be a mixed strategy, and let a be some node. We define the *realization weight that μ induces on a* by

$$\mu[a] = \mu(R(a)) = \sum_{s \in R(a)} \mu(s). \tag{1}$$

The realization weight that a mixed strategy μ induces on a is simply the total probability (according to μ) of the pure strategies that potentially reach a; that is, it is the probability according to μ of potentially reaching a. Note that potential reachability is defined purely in terms of player 1's moves. Hence, the realization weight that μ induces on some node a depends only on μ, and *not* on the strategies of the other players or on the probabilities of the chance moves. For example, for μ^* defined above,

$$\mu^*(z_1) = \mu^*(z_3) = \mu^*(\langle l, c\rangle) + \mu^*(\langle l, d\rangle) = 1/7 + 2/7 = 3/7$$

(since $R(z_1) = R(z_3) = \{\langle l, c\rangle, \langle l, d\rangle\}$).

Now, let z be the set of leaves in T. The assignment of realization weights to all the leaves in the tree is called a *weight distribution*. As the following lemma shows, the weight distribution completely specifies all the relevant information about a mixed strategy.

Lemma 2.5: Two mixed strategies μ and μ' are equivalent if and only if they induce the same weight distribution, i.e. for every $z \in Z$, $\mu[z] = \mu'[z]$.

Proof: Suppose that μ and μ' are equivalent, and let z be an arbitrary leaf. Choose the strategies of the other players and the probabilities at the chance nodes so that they assign probability one to any decision on the path from the root to z. Choose the payoff function so that player 1 receives a payoff of one at z, and zero at every other leaf. The expected payoff to player 1 under any mixed strategy μ is exactly $\mu[z]$. Since the payoffs under μ and μ' must be equal, we have $\mu[z] = \mu'[z]$.

To prove the converse, suppose that $\mu[z] = \mu'[z]$ for every leaf $z \in Z$. Fix an arbitrary vector s^{-1} of pure strategies for all the other players, and an arbitrary probability distribution at each chance node. For any mixed strategy μ for player 1, we can define the probability that z is actually reached under μ and s^{-1}. Let $\pi_\mu(z)$ denote that probability. If on the path from the root to z there is at least one decision not taken under s^{-1}, then $\pi_\mu(z) = 0$ for every μ; otherwise, $\pi_\mu(z)$ is clearly equal to the product of $\mu[z]$ and the probabilities of the chance moves along the path to z. By assumption, $\mu[z] = \mu'[z]$, and therefore $\pi_\mu(z) = \pi_{\mu'}(z)$ for all z. The associated payoffs for μ and μ' are thus also equal. Since a mixed strategy for the other players is a probability distribution over pure strategies, this must also hold if we allow the other players to play mixed strategies. $\square$

This lemma asserts that equivalence of strategies is completely determined by the weight distribution they induce on the leaves of the tree. Hence, a weight distribution represents an entire equivalence class of mixed strategies. Note that the space of weight distributions has a much smaller dimension than the space of mixed strategies. This observation is at the heart of our proof that small mixed strategies are sufficiently expressive. Our definition of "small" is based on the precise dimension of this space, which is at most the number of leaves in T. This lemma provides the motivation for our definition (in the introduction) of the *size of* T as the number of leaves in T.

Theorem 2.6: For any mixed strategy μ, there exists an equivalent strategy μ' whose size is at most $|T|$.

Proof: Let $|\Sigma| = m$, and denote the strategies in Σ by $s^1, \ldots, s^m$. Let r_z denote $\mu[z]$. In order for a mixed strategy η to be equivalent to μ, it has to satisfy $\eta[z] = r_z$ for all leaves z. We can represent these constraints in terms of a system of linear equations and inequalities, as follows. Let the variable x_i represent the probability assigned by a mixed strategy η to the pure strategy s^i ($i = 1, \ldots, m$). The vector $\mathbf{x} = (x_1, \ldots, x_m)^T$ describes an appropriate mixed strategy η if and only if it satisfies the system:

$$\sum_{i:s^i \in R(z)} x_i = r_z \quad \text{(for every leaf } z\text{)} \tag{2}$$

$$\sum_{i=1}^{m} x_i = 1 \tag{3}$$

$$x_i \geq 0 \quad (i = 1, \ldots, m). \tag{4}$$

The number of constraints in equations (2) and (3) is $|T| + 1$. We now show that the constraint (3) is redundant if the numbers r_z are induced by some mixed strategy μ. For each node not belonging to player 1 (even for the chance nodes), select the first edge leading out of the node. Denote this set of choices by s^{-1}, and let Z' be the set of leaves not ruled out by s^{-1}. It is clear that for any $z \in Z'$ and any $s^i \in \Sigma$, s^i potentially reaches z ($s^i \in R(z)$) iff s^i reaches z under s^{-1} (since s^{-1} does not rule out any of the decisions on the path to z). Moreover, any strategy $s^i \in \Sigma$ is in $R(z)$ for exactly one $z \in Z'$: the leaf z reached by the combined strategy (s^i, s^{-1}). Thus the collection of the sets $\{R(z): z \in Z'\}$ constitutes a partition of Σ. Since the weights r_z are induced by μ, we deduce that:

$$\sum_{z \in Z'} r_z = \sum_{z \in Z'} \sum_{s^i \in R(z)} \mu(s^i) = \sum_{s^i \in \Sigma} \mu(s^i) = 1.$$

Thus, for any vector x satisfying constraint (2):

$$\sum_{i=1}^{m} x_i = \sum_{z \in Z'} \sum_{s^i \in R(z)} x_i = \sum_{z \in Z'} r_z = 1.$$

Therefore, equation (3) is redundant.

Let $\boldsymbol{Ax} = \boldsymbol{b}$ be the matrix representation of the system of linear equations described in constraint (2) ($\boldsymbol{A} \in \mathbb{R}^{|T| \times m}$, $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{b} \in \mathbb{R}^{|T|}$). Let $\boldsymbol{x}$ be the vector corresponding to μ: for all i, define x_i to be $\mu(s^i)$ so that by (1), $\mathbf{x}$ satisfies the constraints (2) and (4). Thus, the system $\boldsymbol{Ax} = \boldsymbol{b}$ has a feasible solution $\mathbf{x} \geq \mathbf{0}$. A classical theorem in linear programming asserts that, in this case, there exists a *basic solution* to the same system. A basic solution is a vector $\boldsymbol{x}' \geq \mathbf{0}$ such that $\boldsymbol{Ax}' = \boldsymbol{b}$ and the columns of $\boldsymbol{A}$ corresponding to the positive components of $\mathbf{x}'$ are linearly independent. Since $\boldsymbol{A}$ has $|T|$ rows, there are at most $|T|$ linearly independent columns in $\boldsymbol{A}$. Therefore, the system $\boldsymbol{Ax} = \boldsymbol{b}$, $\boldsymbol{x} \geq \mathbf{0}$ has a basic solution $\boldsymbol{x}'$ with at most $|T|$ positive x_i's. Such a solution describes a mixed strategy μ' as we need, by setting $\mu'(s^i) = x_i'$. □

This result is the motivation for our definition of "small".

Definition 2.7: A mixed strategy μ is said to be *small* if $|\text{Supp}(\mu)| \leq |T|$.

We note that our result is, in fact, stronger than implied by the statement of the theorem. Consider two leaves z and z' that are reachable using the same sequence of decisions of player 1; for example, in the tree T^*, z_1 and z_3 are both reached precisely when the player's decision at u_1 is 1. For two such nodes z, z', necessarily $R(z) = R(z')$ and thus also $\mu[z] = \mu[z']$ for any mixed strategy μ. In particular, the two nodes z and z' induce identical equations in (2). Hence, the number of distinct equations in (2) is at most the number of distinct *sequences* (decision-paths) of player 1 in the tree. We could have defined "small" based on this number, instead of on $|T|$. We choose to use the less precise definition for the sake of simplicity. The definition of sequences and realization plans over sequences also appears in the work of von Stengel [12], who utilizes them to define and solve a generalized *sequence form* of a multi-player game. We observe that, in the worst case, the number of distinct sequences matches its upper bound of $|T|$.

3 Reducing the Size of a Mixed Strategy

In this section we investigate the problem of finding a small mixed strategy μ' that is equivalent to some given mixed strategy μ. Our first result is obtained by re-examining the proof of Theorem 2.6. Recall that the mixed strategy μ can be viewed as a non-negative solution to some linear system of equations; the small mixed strategy μ', whose existence we proved in Theorem 2.6, is a basic solution to the same system. Hence, we can reduce μ to μ' using any standard algorithm for *basis crashing* – converting a non-basic solution to a linear system into a basic one. In our context, the most standard algorithm essentially removes pure strategies from the support of μ one at a time, while maintaining equivalence. Assuming that μ is given to us in sparse representation, the standard basis-crashing algorithm requires $O(|\text{Supp}(\mu)|\cdot|T|^2)$ arithmetic operations for this conversion.[5] We can speed up this construction somewhat using a faster basis-crashing algorithm due to Beling and Megiddo [1], resulting in the following theorem:

Theorem 3.1: Given a mixed strategy μ in sparse representation, it is possible to construct a small equivalent strategy μ' using $O(|\text{Supp}(\mu)|\cdot|T|^{1.62})$ arithmetic operations.

Note that the running time of this algorithm depends linearly on $|\text{Supp}(\mu)|$, which is often exponential in the size of the game tree. Hence, this algorithm is not an effective solution to the problem of constructing small mixed strategies. We would like to find a more efficient approach. As we observed, we cannot improve

[5] If the number of pure strategies is exponential in the size of the tree, and if μ is not represented sparsely, then it clearly requires exponential time simply to scan all of μ.

the running time if we need exponential time simply to read in μ. This might often be the case even if μ is represented sparsely. Hence, we must look for an alternative representation of our input. The obvious choice is the weight distribution: it can be represented very compactly, while still being a complete description of the desired properties of our output μ'.

In this section we therefore investigate the following problem: given some weight distribution, find a small mixed strategy that induces it. Unfortunately, we are able to show that this problem is, in general, NP-hard, thus justifying the exponential behavior of our algorithm above. In fact, we even show that the problem of deciding whether a given weight distribution is induced by *any* mixed strategy is NP-complete. We begin with the lower bound.

Theorem 3.2: Given a game tree T with Z the set of its leaves, and a weight distributions $\{r_z\}_{z \in Z}$, it is NP-hard to decide whether there exists a mixed strategy μ such that $\mu[z] = r_z$ for all $z \in Z$.

Proof: We prove the theorem by reduction from the 3-colorability problem for graphs, which is defined as follows: Given an undirected graph $G = (V, E)$, find a coloring $\gamma : V \to \{1, 2, 3\}$, so that for any edge $(i, j) \in E$, $\gamma(i) \neq \gamma(j)$. Given an instance of this problem, we construct the following game tree. Intuitively, the game has two stages: First, some other player picks some edge $(i, j) \in E$. Then player 1 must (separately) choose a color for i and a color for j. The player has $n = |V|$ information sets $u_1, \ldots, u_n$, corresponding to the vertices of G. The decision of the player at u_i corresponds to a choice of color for vertex i. Clearly, any pure strategy for player 1 corresponds precisely to a (possibly illegal) coloring of the nodes of the graph. We now define the weight distribution for this game. For an edge $e = (i, j) \in E$, and $a, b \in \{1, 2, 3\}$, let $z(e, a, b)$ be the leaf reached by choosing e at the root, a at the information set u_i, and b at the information set u_j. We define:

$$r_{z(e,a,b)} = \begin{cases} 0 & a = b \\ \frac{1}{6} & a \neq b \end{cases}$$

It remains only to prove that there is a mixed strategy μ generating this weight distribution if and only if there exists a legal 3-coloring for the original graph.

Assume that μ is a mixed strategy generating this weight distribution, and let s be any pure strategy in the support of μ; that is, $\mu(s) > 0$. We have already shown that s describes a coloring in G by $\gamma(i) = s_i$. Now, consider any edge $e = (i, j) \in E$. If $s_i = s_j = a$, then s potentially reaches $z(e, a, a)$. Since s has positive probability in μ, by definition $\mu[z(e, a, a)] > 0$; since $r_{z(e,a,a)} = 0$, this contradicts the assumption that μ induces this weight distribution. Thus, for every edge $(i, j) \in E$, $s_i \neq s_j$, and therefore also $\gamma(i) \neq \gamma(j)$; i.e. γ is a legal coloring.

Now, assume that there exists a legal coloring γ for G. Let $\gamma_1, \ldots, \gamma_6$ denote the six colorings resulting from γ by permuting the colors. More precisely, each γ_k is equal to $\gamma \circ \pi$ where π is some permutation of $\{1, 2, 3\}$. Let s^k be the pure strategy representing $\gamma_k : s_i^k = \gamma_k(i)$. Let μ be the mixed strategy such that $\mu(s^k) = \frac{1}{6}$ for

$k = 1, \ldots, 6$ and $\mu(s) = 0$ for all other $s \in \Sigma$. We want to prove that μ induces the desired weight distribution. Since each s^k represents a legal coloring, it does not potentially reach any leaf of the form $z(e, a, a)$. Therefore, $\mu[z(e, a, a)] = 0$ for any such leaf. On the other hand, a leaf $z(e, a, b)$, where $e = (i, j)$ and $a \neq b$ is potentially reached by s^k if and only if $\gamma_k(i) = a$ and $\gamma_k(j) = b$. Because of our definition of $\gamma_1, \ldots, \gamma_6$ as different permutations of γ, this latter condition holds for precisely one coloring γ_k. The vertex $z(e, a, b)$ is potentially reached by the corresponding s^k but not by any $s^{k'}$, $k' \neq k$. Therefore, $\mu[z(e, a, b)] = \frac{1}{6}$ for all leaves $z(e, a, b)$ where $a \neq b$. Therefore, this mixed strategy μ generates the desired weight distribution. □

We note that this result relies on the fact that the player has imperfect recall. In the next section, we show that if the player has perfect recall, there exists a polynomial time algorithm for constructing a small mixed strategy from a weight distribution. On the other hand, the hardness result applies even to a very restricted class of games with imperfect recall: those where the player makes at most two decisions on every path, and where all information sets have at most three choices. In fact, the result also holds if we restrict to games where there are two decisions on each path and all information sets have *two* choices; see [5, Theorem 3.3] for more details.

We now prove the matching upper bound to the lower bound of Theorem 3.2.

Theorem 3.3: Given a game tree T with Z its leaves, and a weight distribution $\{r_z\}_{z \in Z}$, it is NP-complete to decide whether there exists a mixed strategy μ such that $\mu[z] = r_z$.

Proof: We proved NP-hardness in the previous theorem. It remains only to prove that the problem is in NP. Recall that in Theorem 2.6, we represented the constraints on a mixed strategy μ inducing a particular weight distribution as a set of linear equations and non-negativity constraints. That theorem also shows that if there exists any mixed strategy satisfying these constraints, then there exists one over a support whose size is at most $|T|$.

We can use these facts to construct a nondeterministic polynomial time algorithm for this problem. First, the algorithm nondeterministically chooses a set $\Sigma' \subset \Sigma$ of at most $|T|$ pure strategies ($|\Sigma'| \leq |T|$). This Σ' is one possibility for the support of an appropriate mixed strategy μ. The algorithm then attempts to solve the system of equations and non-negativity constraints in Equations (2), (3), and (4), over the subset of the variables x_i for $s^i \in \Sigma'$ (the others are set to 0). This is a linear programming problem (of a very simple type) with polynomially many equations and polynomially many variables; it can therefore be solved in polynomial time using any polynomial time linear programming algorithm. We know that there exists some mixed strategy μ generating this weight distribution if and only if there is a solution to one of these systems. Since the algorithm described is an NP algorithm, the problem is in NP. □

4 From Behavior Strategies to Small Mixed Strategies

In the previous section we analyzed the complexity of finding a small mixed strategy that is equivalent to a given mixed strategy. We first explained why this problem is uninteresting when the original mixed strategy is represented explicitly. We then showed that the problem is hard when the mixed strategy is represented succinctly, using its induced weight distribution. In this section, we investigate the same problem in the context of the more restricted class of *behavior strategies.* As we explained in the introduction, the class of behavior strategies is a very important subclass of mixed strategies. Kuhn [8] showed that for player with perfect recall, every mixed strategy has an equivalent behavior strategy. Hence, in those cases where the player has perfect recall, our results from this section apply to any mixed strategy of the player. On the other hand, in [4] we argued that for a player with imperfect recall, the use of an arbitrary mixed strategy may violate the spirit of the imperfect recall requirement. Hence, even in the case of imperfect recall, behavior strategies are of particular interest.

In this section, we present a polynomial-time algorithm that receives a behavior strategy and produces a small mixed strategy equivalent to it. Our algorithm applies only to games where each information set intersects each path from the root to a leaf at most once. However, this restriction is not specific to our approach, but to the result itself. In games where this does not hold, there exist behavior strategies that do not have any realization-equivalent mixed strategy.[6]

We present our algorithm as constructing a small mixed strategy from a behavior strategy represented in the standard way: as a tuple of probability distributions on moves at the different information sets. However, it works equally well when the behavior strategy is represented as a weight distribution. The reason is that, given a weight distribution $\{r_z\}_{z\in Z}$ (where $0 \leq r_z \leq 1$ for all z), we can compute in polynomial time a behavior strategy β inducing it, if one exist. The procedure is as follows. First, by working up from the leaves, we define r_a for each node a in the tree. This weight is already defined for the leaves in the tree. Let $a_1, \ldots, a_l$ be the children of a node a in the tree, and assume that we have already defined $r_{a_1}, \ldots, r_{a_l}$. If the node a belongs to player 1, we define $r_a = \sum_{i=1}^{l} r_{a_i}$. Otherwise, define $r_a = r_{a_1}$ if $r_{a_1} = \cdots = r_{a_l}$. If this is not the case, then the weight distribution is not derived from a legal behavior strategy. Given this extended weight distribution, we can now define the appropriate behavior strategy. Consider any information set u_j of player 1. We wish to define a probability distribution β_j over the set C_j of choices at u_j. For any choice $c \in C_j$ and any $a \in u_j$, let a^c be the child of a reached by taking the decision c. If for some nodes $a, b \in u_j$

[6] Consider a one-player game where a player has two decision nodes belonging to the same information set. If he plays R at the first node, he receives a payoff of 0; if he plays L he arrives at the second decision node, where playing R results in a payoff of 1000, and playing L in a payoff of 0. Any pure strategy, and hence any mixed strategy, results in a payoff of 0, whereas there exist behavior strategies that given positive payoff.

and a choice c it is the case that $r_{ac} \neq r_{bc}$ then again the weight distribution is not derived from a legal behavior strategy. Otherwise, there are two cases. If $r_a = 0$ for some (and thus all) $a \in u_j$, then u_j is an irrelevant information set, so that we can pick β_j arbitrarily. Otherwise, we define $\beta_j(c)$ to be (r_{ac}/r_a) (this definition is now clearly independent of our choice of a). See [4, 12] for further details and proof of correctness.

A behavior strategy β induces a mixed strategy $\bar{\mu}$ in the obvious way:

$$\bar{\mu}(s) = \prod_{j=1}^{n} \beta_j(s_j). \tag{5}$$

Unfortunately, when β is highly mixed (i.e. when $\beta_j(c) > 0$ for many j, c), $|\mathrm{Supp}(\bar{\mu})|$ will be very large (exponential in $|T|$). In Section 2, we proved that there exists a mixed strategy μ equivalent to β, such that $|\mathrm{Supp}(\mu)| \leq |T|$. We now describe a polynomial time algorithm for constructing such a mixed strategy.

The algorithm proceeds by incrementally constructing mixed strategies over more and more information sets. We define a sequence of trees $T^0, T^1, T^2, \ldots, T^n$, such that T^k is identical to T, except that all the nodes in the information sets $u_{k+1}, \ldots, u_n$ belonging to player 1 in T belong to some other arbitrary player in T^k. Note that in the tree T^n, all of $u_1, \ldots, u_n$ belong to player 1, so that T^n is equal to T. Recall that the ownership of the nodes not belonging to player 1 is irrelevant to the construction of equivalent strategies. Since only the ownership of the nodes in the tree changes, $|T^k| = |T|$ for all k. A *projected pure strategy* for T^k has the form $\langle s_1, \ldots, s_k \rangle$, and is denoted s^k. Similarly, a *projected behavior strategy* β^k is $\langle \beta_1, \ldots, \beta_k \rangle$. A projected mixed strategy μ^k is a mixed strategy over the tree T^k. That is, it is a probability distribution over pure strategies that assign decision values only to the information sets $u_1, \ldots, u_k$. We define $R^k(z)$ to be $R(z)$ in the tree T^k.

We recursively construct a sequence of mixed strategies μ^k for $k = 0, 1, .., n$, such that:

Condition 1: μ^k is equivalent to β^k,
Condition 2: $|\mathrm{Supp}(\mu^k)| \leq |T|$.

Initially, $k = 0$; there are no nodes belonging to player 1 in T^0. Obviously, $\beta^0 = \langle\ \rangle$, the empty tuple consisting of no decisions. The only legal pure strategy in this tree is also $\langle\ \rangle$; we therefore define $\mu^0(\langle\ \rangle) = 1$. Clearly, both conditions are satisfied.

Now, suppose we have defined μ^k that is equivalent to β^k, such that $|\mathrm{Supp}(\mu^k)| \leq |T|$. We will first define an intermediate mixed strategy η^{k+1} that will turn out to be equivalent to β^{k+1}:

$$\eta^{k+1}(s^k \circ s_{k+1}) = \mu^k(s^k) \cdot \beta_{k+1}(s_{k+1}),$$

for any s^k and $s_{k+1} \in C_{k+1}$, where for $s^k = \langle s_1, \ldots, s_k \rangle$, $s^k \circ s_{k+1} = \langle s_1, \ldots, s_k, s_{k+1} \rangle$.

Lemma 4.1: If μ^k is equivalent to β^k, then η^{k+1} is equivalent to β^{k+1}.

Proof: Let z be any leaf in T^{k+1}. We distinguish two cases:

Case I: On the path from the origin to z there is a node from information set u_{k+1}. Let c be the decision chosen at that node along the path. Since the information set u_{k+1} can intersect the path to z at most once, we have

$$\beta^{k+1}[z] = \beta^k[z] \cdot \beta_{k+1}(c),$$

where $\beta^k[z]$ is taken over T^k. On the other hand, by the assumption of the lemma,

$$\beta^k[z] = \mu^k[z],$$

and therefore, by Equation 1,

$$\beta^k[z] = \sum_{s^k \in R^k(z)} \mu^k(s^k).$$

For any projected pure strategy s^k, if s^k reaches z in the tree T^k, and if we extend s^k by the move c at the information set u_{k+1}, then the extended strategy reaches z in the tree T^{k+1}. Similarly, if any strategy s^{k+1} reaches z in T^{k+1}, then $s^{k+1}_{k+1} = c$ and s^k reaches z in T^k. Therefore, using an obvious notation,

$$R^{k+1}(z) = R^k(z) \circ c. \tag{6}$$

Note that by the definition of η^{k+1}:

$$\mathrm{Supp}(\eta^{k+1}) \subseteq \mathrm{Supp}(\mu^k) \circ c. \tag{7}$$

Using Equations (6) and (7), and the definition of η^{k+1}, we obtain that

$$\begin{aligned} \eta^{k+1}[z] &= \sum_{s^{k+1} \in R^{k+1}(z)} \eta^{k+1}(s^{k+1}) \\ &= \sum_{s^k \in R^k(z)} \eta^{k+1}(s^k \circ c) \\ &= \sum_{s^k \in R^k(z)} \mu^k(s^k) \cdot \beta_{k+1}(c) \\ &= \beta^{k+1}[z]. \end{aligned}$$

Case II: The information set u_{k+1} does not intersect the path to z. In this case

$$\beta^{k+1}[z] = \beta^k[z]. \tag{8}$$

For any projected pure strategy s^k which reaches z in T^k, all projected pure strategies $s^k \circ c$ ($c \in C_{k+1}$) reach z in T^{k+1}. Therefore, $R^{k+1}(z) = \cup_{c \in C_{k+1}} (R^k(z) \circ c)$. We can thus deduce that:

$$\begin{aligned}\sum_{s^{k+1} \in R^{k+1}(z)} \eta^{k+1}(s^{k+1}) &= \sum_{s^k \in R^k(z)} \sum_{c \in C_{k+1}} \eta^{k+1}(s^k \circ c) \\ &= \sum_{s^k \in R^k(z)} \sum_{c \in C_{k+1}} \mu^k(s^k) \cdot \beta_{k+1}(c) \\ &= \left(\sum_{s^k \in R^k(z)} \mu^k(s^k) \right) \cdot \left(\sum_{c \in C_{k+1}} \beta^{k+1}(c) \right) \\ &= \mu^k[z].\end{aligned}$$

Using the lemma assumption and Equation (8), we deduce:

$$\eta^{k+1}[z] = \mu^k[z] = \beta^k[z] = \beta^{k+1}[z].$$

Since one of the two cases holds for any leaf z, the claim follows from Lemma 2.5. □

We have established that η^{k+1} satisfies Condition 1. However, we only know that $|\mathrm{Supp}(\eta^{k+1})| \leq |\mathrm{Supp}(\mu^k)| \cdot |C_{k+1}|$, and therefore, in general, Condition 2 will not be satisfied. Thus η^{k+1} does not suffice for our purposes. Note, however, that η^{k+1} is a nonnegative solution to the system of constraints (2)–(4) described in the proof of Theorem 2.6. Using the algorithm of Theorem 3.1, we can find a small mixed strategy μ^{k+1} that is equivalent to η^{k+1}. It follows from Theorem 2.6 that $|\mathrm{Supp}(\mu^{k+1})| \leq |T^{k+1}| = |T|$. Thus, μ^{k+1} satisfies Condition 2. Furthermore, μ^{k+1}, η^{k+1}, and β^{k+1} are all equivalent, so that μ^{k+1} satisfies Condition 1. Hence, μ^{k+1} satisfies the requirements of the $(k+1)^{st}$ step of the recursive construction.

Theorem 4.2: Given a behavior strategy β, an equivalent small strategy μ can be found in strongly polynomial time performing $O(n \sum_{k=1}^{n} |C_k| |T|^{2.62}) = O(n|T|^{3.62})$ arithmetic operations.

Proof: For $k = n$, μ^k is a small mixed strategy over T which is equivalent to β. Let c_k denote $|C_k|$. The construction requires n iterations. Iteration k ($k = 1, \ldots, n$) requires $O(c_k |T|)$ operations to create η^k from μ^{k-1}, and $O(|\mathrm{Supp}(\eta^k)| \cdot |T|^{1.62}) = O(c_k |T|^{2.62})$ operations for transforming η^k into μ^k. The entire algorithm therefore requires $O(n \sum_{k=1}^{n} c_k |T|^{2.62}) = O(n|T|^{3.62})$ arithmetic operations. □

5 Solving Extensive Games Efficiently

Our original motivation for investigating the existence of small mixed strategies is their potential usefulness in algorithms. Having shown that small mixed strategies are as expressive as arbitrary mixed strategies, we now use this fact to construct much faster algorithms for solving extensive games. The basis for our approach is the following corollary to Theorem 2.6.

Corollary 5.1: Consider an N-player extensive-form game T. For any equilibrium payoff vector $\boldsymbol{h} = (h^1, \ldots, h^N)$, there exists an equilibrium strategy combination $\boldsymbol{\mu} = (\mu^1, \ldots, \mu^N)$ yielding $\boldsymbol{h}$ where all of the μ^i's are small mixed strategies.

Proof: Consider some equilibrium combination resulting in the payoff vector $\boldsymbol{h}$, and take the equivalent small mixed strategy for each player's strategy separately. □

This corollary allows us to search for an equilibrium over the space of small mixed strategies rather than over all mixed strategies. For each player $i = 1, \ldots, N$, let Σ^i denote the set of pure strategies of player i. Our general approach takes a standard algorithm for computing equilibria in normal form games, and uses it to construct equilibria in small mixed strategies. The general outline is as follows.

(i) For each i, choose a subset S^i of Σ^i of size at most $|T|$.
(ii) For each such choice, enumerate all the *candidate equilibria* – equilibria over the normal form game derived from T by restricting each player i to the pure strategies in S^i.
(iii) For each such candidate equilibrium $\boldsymbol{\mu} = (\mu^1, \ldots, \mu^N)$, check whether it is also an equilibrium in the full game. In the worst case, this can be accomplished by generating each possible pure strategy for each player i, and checking its payoff against $\boldsymbol{\mu}^{-i}$. If that payoff is better than $h^i(\boldsymbol{\mu})$ then $\boldsymbol{\mu}$ is not an equilibrium.[7]

We could apply this scheme to any of the algorithms for solving N-player normal form games, for example, the algorithms of Rosenmüller [10] or Wilson [13]. This would result in an algorithm for finding equilibria in N-player extensive-form games.

We now show how this scheme can be used to find equilibria in two-person extensive-form games. In this case, the problem of finding equilibria in a normal-form game can be described as a *linear complementarity probelm* (*LCP*) (see [2]). There are a number of standard algorithms for finding such equilibria. One possibility is to enumerate all the possible supports for a mixed strategy pair (a support for each of the two players), and attempt to find an equilibrium over that support pair. It is straightforward to show that, in the two-player case, an equilibrium over a given support pair is the solution to a system of linear equations [2, p. 17]. The approach above modifies this construction by traversing

[7] Note that although this is an expensive procedure, its cost is negligible relative to the exhaustive enumeration of all possible supports.

only *small* supports for the two players. Corollary 5.1 shows that this can be done without loss of generality. Any equilibrium payoff that can be found using the standard exhaustive enumeration algorithm can also be found by enumerating only small supports. This allows us to construct an algorithm for finding equilibria of two-player extensive-form games. Note that, unlike previous algorithms, the running time of our algorithm is exponential *in the size of the game tree*, rather than in the size of the corresponding normal-form game.

Theorem 5.2: The algorithm outlined above finds all payoffs, and strategies generating them, corresponding to the basic equilibria of a two-player game in extensive form. The algorithm runs in time exponential in the size of the game tree.

Proof: Let m_1, m_2 denote $|\Sigma^1|$ and $|\Sigma^2|$ respectively, and let t denote $|T|$. The number of supports examined by the algorithm for player i is $\sum_{j=1}^{t} \binom{m_i}{j}$, which is at most tm_i^t. For each pair of supports, one for each player, the algorithm attempts to find an equilibrium over that pair. In this case, this procedure reduces to solving a set of linear equation, and can therefore be done in polynomial time. Finally, for each resulting candidate equilibrium (μ^1, μ^2), the algorithm needs to check whether it is, in fact, an equilibrium; i.e. whether μ^1 is a best response to μ^2 and vice versa. This is done by checking, for any pure strategy s^i of player i, whether the payoff achieved by s^i against μ^j $(j \neq i)$ is better than that achieved by μ^i. For any s^i this requires at most $O(t^2)$ operations: t for enumerating the pure strategies s^j in the support of μ^j, and $O(t)$ for computing i's payoff given a pair of pure strategies s^i, s^j. Hence, we obtain an algorithm whose running time is:

$$O(m_1^t \cdot m_2^t \cdot \text{poly}(t) \cdot (m_1 + m_2) \cdot t^2) = O(m_1^{t+1} m_2^{t+1} \text{poly}(t)). \quad \square$$

An alternative approach to the problem of finding equilibria was proposed by Lemke and Howson [9]. Their algorithm searches for a single equilibrium, and cannot be used to enumerate all of them. The algorithm generates a sequence of basic solutions to the underlying system of linear equations. Each basic solution supports a pair of mixed strategies (one for each player) which are "almost in equilibrium" (almost complementary). The algorithm moves between bases using pivoting operations similar to those used by the simplex algorithm. The algorithm terminates when a pair of supports defining an equilibrium is found. Using the same techniques described above , it is possible to modify this algorithm so that only small supports are traversed. This procedure was done by Wilson [14]. His algorithm "restrict(s) the computation to the ordinarily small portion corresponding to the strategies actually used by the players." Like the original Lemke-Howson algorithm, Wilson's variant traverses the space of mixed strategies. However, rather than maintaining the entire representation of the intermediate mixed strategies in his search, it maintains only those pure strategies in their supports. Wilson also shows how to avoid searching the entire space of pure strategies when the algorithm calls for a new pure strategy to enter the support.

He shows that "in games with perfect recall these strategies can be generated as needed from an auxiliary analysis of the players' decision trees derived from the extensive form of the game."

As suggested by our quote from Wilson's paper, the motivation for this algorithm is derived from the observation that, typically, mixed strategies have a small support. Since this was, at the time, only a "rule of thumb", the running time of the algorithm could not have been analyzed formally. Our results can be viewed as providing a formal justification for Wilson's algorithm. We can, in fact, use our techniques to construct a variant of Wilson's algorithm whose running time is guaranteed to be exponential in the size of the game tree, in the worst case. As for the Lemke-Howson algorithm, in many cases the algorithm will not need to traverse of all the possible small supports, so that it should often run faster than our complete enumeration algorithm above. We chose not to present this revised algorithm and the associated analysis since, for the case of perfect recall games, a much better algorithm already exists [6]. There, we show how to use the notion of realization weights to construct a small LCP that can be represented explicitly and solved using the standard Lemke-Howson algorithm

Acknowledgements: The authors wish to acknowledge Moni Naor for providing a greatly simplified proof of Theorem 3.2, and an anonymous referee for useful comments. They also thank Bernhard von Stengel for many useful comments and references.

References

[1] Beling PA, Megiddo N (1993) Using fast matrix multiplication to find basic solutions. Tech Report RJ 9234, IBM Research Division
[2] Cottle RW, Pang J-S, Stone RE (1992) The linear complementarity problem. Academic Press, San Diego
[3] Dalkey N (1953) Equivalence of information patterns and essentially indeterminate games. In: contributions to the theory of games II (Princeton) Kuhn HW, Tucker AW (eds) Princeton, University Press, 217–243
[4] Koller D, Megiddo N (1992) The complexity of two-person zero-sum games in extensive form. Games and Economic Behavior 4: 528–552
[5] Koller D, Megiddo N (1994) Finding small sample spaces satisfying given constraints. SIAM Journal on Discrete Mathematics: 260–274
[6] Koller D, Megiddo N, von Stengel B. Fast algorithms for finding randomized strategies in game trees. Proceedings of the 26th ACM Symposium on Theory of Computing 750–759
[7] Kuhn HW (1950) Extensive games. Proc National Academy of Sciences of the USA 36: 570–576
[8] Kuhn HW (1953) Extensive games and the problem of information. In: Contributions to the theory of games II (Princeton) Kuhn HW, Tucker AW (eds) Princeton University Press 193–216
[9] Lemke CW, Howson Jr JT (1964) Equilibrium points in bimatrix games. Journal of the Society for Industrial and Applied Mathematics 12: 413–423
[10] Rosenmüller J (1971) On a generalization of the Lemke-Howson algorithm to noncooperative N-person games. SIAM Journal on Applied Mathematics 21: 73–79
[11] Swinkels J (1989) Subgames and the reduced normal form. Tech Report 344 Econometric Research Program, Princeton University

[12] von Stengel B (1993) LP representation and efficient computation of behavior strategies. Tech Report S-9301 University of the Federal Armed Forces at Munich

[13] Wilson R (1971) Computing equilibria of N-person games. SIAM Journal on Applied Mathematics 21: 80–87

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

Received March 1992
Revised version September 1994