Computational Intelligence, Volume 12, Number 1, 1996

PERFECT RECALL AND PRUNING IN GAMES WITH IMPERFECT INFORMATION[1]

JEAN R. S. BLAIR

Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, NY 10996
blair@eecs1.eecs.usma.edu

DAVID MUTCHLER

Department of Computer Science, Rose-Hulman Institute of Technology, 5500 Wabash Avenue, Terre Haute, IN 47803-3999
mutchler@cs.rose-hulman.edu

MICHAEL VAN LENT

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109
vanlent@eecs.umich.edu

Games with *imperfect information* are an interesting and important class of games. They include most card games (e.g., bridge and poker) as well as many economic and political models. Here we investigate algorithms for finding the simplest form of a solution (a pure-strategy equilibrium point) to imperfect information games expressed in their extensive (game tree) form. We introduce to the artificial intelligence community a classic algorithm, due to Wilson, that solves one-player games with perfect recall. Wilson's algorithm, which we call `IMP-minimax`, runs in time linear in the size of the game-tree searched. In contrast to Wilson's result, Koller and Meggido have shown that finding a pure-strategy equilibrium point in one-player games *without* perfect recall is $\mathcal{NP}$-hard. Here, we provide another contrast to Wilson's result—we show that in games *with* perfect recall but more than one player, finding a pure-strategy equilibrium point, given that such an equilibrium point exists, is $\mathcal{NP}$-hard.

Our second contribution is to present a pruning technique for Wilson's `IMP-minimax` algorithm to make the latter more tractable. We call this new algorithm `IMP-alpha-beta`. We provide a theoretical framework (model) and analyze `IMP-alpha-beta` in that model. `IMP-alpha-beta` is of direct value for one-player, perfect-recall games. It also has strong potential for other imperfect information games, as it is a natural (but as yet untested) heuristic in those cases.

Key words: game tree, imperfect information, perfect recall, heuristic search, pruning, alpha-beta, `IMP-minimax`, `IMP-alpha-beta`, $\mathcal{NP}$-hard, $\mathcal{NP}$-complete

1. INTRODUCTION

Games with *imperfect information* are an interesting and important class of games. They have been studied at length in the game theory literature and include many important applications, for example,

- Parlor games like bridge, poker, and Clue.
- Economic models of labor-and-management negotiation in which variables like future inflation rates are modeled probabilistically.
- A distributed computation in which the cooperating processors (each with its own input) make certain judgments whose efficacy is determined by the collective input (which is modeled probabilistically).

Game theorists have long known that imperfect information games are different from perfect information games in terms of solution theory and expressiveness (von Neumann 1928). More recently, computer scientists have examined the characteristics of imperfect in-

[1]The views expressed herein are those of the authors and do not purport to reflect the position of the United States Military Academy or the Department of the Army.

formation games from the *computational* viewpoint. They have found imperfect information games to be different from perfect information games in this respect also. Despite the differences, there is potential for transfer of ideas between the two classes (Levy 1989; Gambäck *et al.* 1991; Barr 1992; Gambäck *et al.* 1993; Smith and Nau 1993; Bampton 1994). Such transfer might enable the construction of better algorithms for perfect information games or better algorithms for imperfect information games.

This paper focuses on imperfect information games. It provides an overview of such games, extends the current understanding of their computational properties, and extends techniques for solving such games both heuristically and exhaustively. In particular:

1. We extend results of Koller and Meggido (1992) to show that solving imperfect information games is $\mathcal{NP}$-hard,[2] even when the input is the entire game tree to be searched. This stands in sharp contrast with perfect information games, where the classical minimax algorithm (Zermelo 1913; Shannon 1950; Kuhn 1953) is linear in the size of the game tree searched.
2. These results lead one to seek heuristics or special cases. To that end, we introduce to the artificial intelligence (AI) community a classic algorithm due to Wilson (1972) that solves a special case; we point out that it can be modified for use as a natural heuristic in the general case. Wilson's algorithm, which we call `IMP-minimax`, is to imperfect information games as minimax is to perfect information games. We further introduce and analyze `IMP-alpha-beta`, which computes the same value as does `IMP-minimax` but usually does so faster through *pruning* (i.e., not examining the value of some nodes). `IMP-alpha-beta` is to `IMP-minimax` as the powerful alpha-beta algorithm (Slagle and Dixon 1969; Knuth and Moore 1975; Luckhardt and Irani 1986; Korf 1991) is to minimax.

The above results are fundamental to an understanding of game-tree search in imperfect information games. The first result shows that to find pure strategy equilibrium points, which are the simplest form of "solution" for such games, we must seek special cases or heuristics. The second and third results complement the first, providing an efficient algorithm for a special case, that can be used as a natural heuristic in the general case.

The next section involves the study of imperfect information games by explaining in informal terms how they differ from the perfect information games more familiar to AI researchers. It also gives a simple, concrete example to show how games with imperfect information are different—from the computational viewpoint and from games with perfect information. Section 3 defines the standard game-theoretic terms we use herein, including a formal definition of imperfect information games. Section 4 presents the $\mathcal{NP}$-hardness results that are summarized near the beginning of that section. The remaining sections present the `IMP-minimax` and `IMP-alpha-beta` algorithms and an analysis of the latter's effectiveness at pruning.

2. WHAT IS IMPERFECT INFORMATION?

This section motivates the study of imperfect information games by explaining in informal terms how they differ from perfect information games. The next section presents the classic formal definition of an imperfect information game.

[2]By "solving," we mean "find a pure strategy equilibrium point, given that one exists"; these and other game-theoretic terms are defined in Section 3.

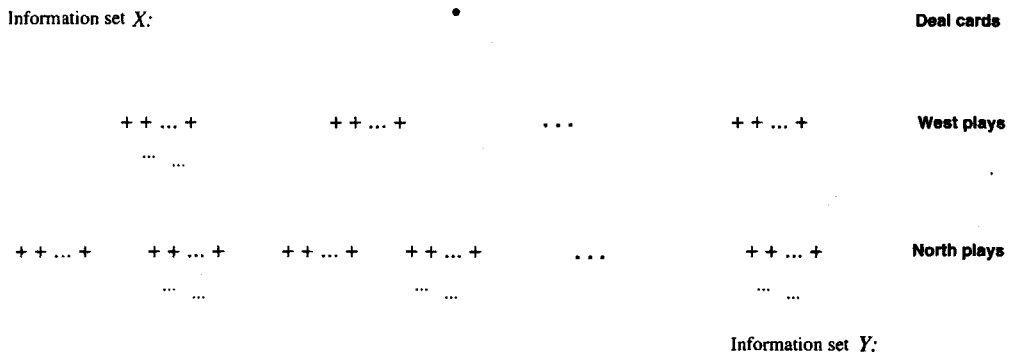

FIGURE 1. The top portion of the game tree for bridge.

Subsection 2.1 uses an example—the card game bridge—to explain, in informal terms, the notion of an imperfect information game. Subsection 2.2 gives a simple, concrete example to show how games with imperfect information are different, from the computational viewpoint, from games with perfect information. In Subsection 2.3, we summarize these differences, then conclude with a brief discussion of related AI work on imperfect information games.

2.1. An Informal Example—Bridge

Chess is the prototypical example of a game with perfect information—both players have equal knowledge of the game state. Backgammon is also perfect information but includes chance nodes (i.e., dice rolls). The card game bridge[3] is a good example of an imperfect information game. Bridge is played by four players organized into two teams, traditionally labeled North/South versus East/West. Figure 1 shows a partial sketch of the top portion of the game tree for bridge. The root is a chance node that represents dealing the shuffled deck of cards. At depth 1 there is a node, marked by a + sign, for each possible outcome. All of the depth 1 nodes corresponding to a single hand that West may have been dealt are grouped together in a single "information set," notated by drawing an ellipse around the set of nodes. Thus, there are $\binom{52}{13}$ information sets at depth 1, each containing $\binom{39}{13}\binom{26}{13}\binom{13}{13}$ nodes. Consider one such information set, call it X, in which West holds ♠9 ♡J743 ◇Q9643 ♣A84. For each node in X there are 13 alternatives (moves), corresponding to the 13 cards West could play. Because West does not yet see the other players' hands, *the rules of the game require that West select the same alternative from each node in set X*. This is called *imperfect information*. For example, West could choose the ninth alternative from each node in X, corresponding to playing ◇4.

The North/South team plays next, selecting a card from the North hand, after exposing the North hand (the so-called dummy) to all four players. At this point, 27 cards are visible to the North/South team: their own 26 cards plus the card West exposed. All the depth 2 nodes in which a particular set of 27 cards is visible to North/South are grouped into a single information set. Thus, there are $\binom{52}{27}$ information sets at depth 2, each containing $\binom{25}{12}\binom{13}{13}$ nodes. For example, in one such information set, call it Y, North holds ♠K62 ♡AQ82 ◇KJ85 ♣76, South holds ♠AQ8743 ♡9 ◇72 ♣KQJ2, and West has led the ◇4. For each node in Y

[3]We assume the reader has some familiarity with bridge. Our example ignores the so-called bidding phase that precedes the card play.

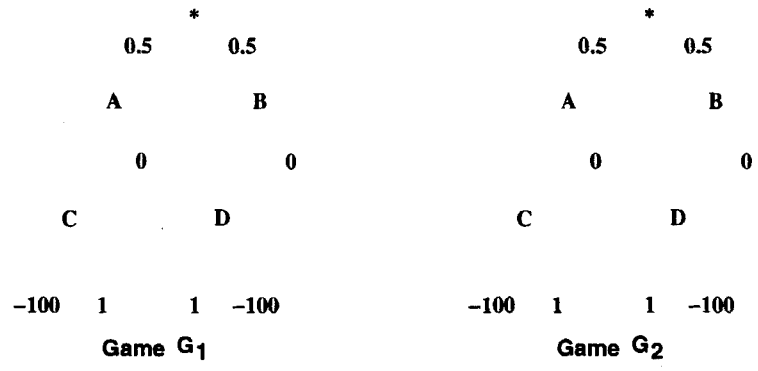

FIGURE 2. Two game trees, with perfect and imperfect information, respectively.

there are 4 alternatives, corresponding to the 4 diamonds the North/South team could play from the North (dummy) hand.[4] Again, *the rules of the game require that North/South select the same alternative from each node in set Y*. For example, North/South could choose the alternative from each node in Y that corresponds to playing $\diamondsuit$J. The game tree continues in like fashion.

Bridge cannot be modeled adequately as a perfect information game, even with the inclusion of chance nodes. In a perfect information game, at any point of the play, all players receive exactly the same description of the current state of the game. In bridge, such a description is impossible: cards seen by one player are unseen by other players.

2.2. Another Example—Perfect Information Versus Imperfect Information

To illustrate the computational difference between perfect information games and imperfect information games, first consider game tree G_1 on the left side of Fig. 2; this is a one-player game with perfect information. Initially, a chance event occurs with two possible outcomes, A or B, each equally likely. After outcome A, the sole player can either continue to position C, or quit the game with payoff 0. Likewise, after outcome B, the player can either continue to position D, or quit the game with payoff 0. From positions C and D, there are two choices, with payoffs -100 and 1 from C and 1 and -100 from D. The player seeks to maximize the expected payoff.

For games like G_1 that have perfect information (perhaps with chance nodes), a simple modification of minimax computes the value of the game tree (Ballard 1983):

$$\texttt{chance-minimax}\,(x) = \begin{cases} \text{payoff}\,(x) & \text{if } x \text{ is a leaf} \\ \text{F}\,\{\,\texttt{chance-minimax}(y) \mid y \text{ is a child of } x\,\} & \text{otherwise} \end{cases}$$

where function F is a *max* operator at nodes where the player moves and *average* at chance nodes.[5] Thus, a single traversal of the tree allows one to compute the value of the game tree as well as the optimal moves (by recording the *max* selections). This `chance-minimax`

[4]The rules of bridge require that North "follow suit" (i.e., play a card in the suit led).

[5]See Kuhn (1953), Luckhardt and Irani (1986), and Korf (1991) for the extension to more than two players.

algorithm correctly computes that the value of game G_1 is 1; the optimal strategy is to move left from nodes A, B, and D, and right from node C.

Now consider the game tree G_2 on the right side of Fig. 2. This is again a one-player game, but here we view the "player" as a team of two cooperating but uncommunicating agents. As in G_1, the chance event happens first, then the first agent makes the first decision (knowing the outcome of the chance event). But in G_2, the second agent makes the second decision *not knowing the outcome of the chance event.* Thus, the second agent can choose only left or right from C and D; the rules of the game require that the same decision must be made from both of these positions.

Both games G_1 and G_2 have chance nodes, but G_1 has perfect information while G_2 has imperfect information (because of the two-node information set containing nodes C and D). For game G_1, with perfect information, the `chance-minimax` algorithm correctly computes the value of the game. The `chance-minimax` algorithm performs the same computation on G_2 as it did for G_1, and, hence, computes both the wrong value for the game and a wrong strategy (regardless of how it resolves the conflicting decisions at C and D). In G_2, the optimal expected payoff is 0.5, obtained (for example) by moving left from position A, right from position B, and right from positions C and D. The optimality of this strategy can be seen by comparing it to all other strategies on the entire search space.

2.3. Informal Examples—Summary

The above examples show that imperfect information games are both interesting and different from perfect information games (even with chance nodes). In the first example (bridge), one cannot model imperfect information adequately by using perfect information games: there is simply no mechanism in perfect information games to model different players seeing different portions of the game state. The second example (games G_1 and G_2) shows that perfect information games are very different from imperfect information games, *from the computational viewpoint.* The `chance-minimax` algorithm solves the perfect information game G_1 in time linear in the size of the game tree. However, the algorithm gives an incorrect solution for the imperfect information game G_2. The $\mathcal{NP}$-hardness results of Section 4 show that such behavior is unavoidable, in general, unless $\mathcal{P} = \mathcal{NP}$.

Note that our complexity results are phrased in terms of the size of the game tree rather than the more common phrase "exponential in the depth of the game tree." Our results for imperfect information games imply that the associated algorithms are exponential in the size of the game tree (hence, doubly exponential in its depth), unless $\mathcal{P} = \mathcal{NP}$. Throughout, the size of the game tree is the size of the game tree searched. That is, nothing prevents the use of a static evaluator (Shannon 1950) to reduce the size of the game tree by treating selected interior nodes as leaves, with estimates for the values of the unsearched subtrees below them.

Much of the AI work on perfect information games has focused on computing a correct (game-theoretic) strategy, that is, the minimax strategy. In contrast, AI work on imperfect information games has avoided the task of computing a correct strategy (as described formally in the next section). For example, the classic work of Findler (1977) on poker focuses on the cognitive aspects of strategies. Quinlan (1979) describes particular heuristics for bridge, as does Gordon (1993) for scrabble. Bampton (1994), Barr (1992), Gambäck *et al.* (1991, 1993), Levy (1989), and Smith and Nau (1993) describe general purpose heuristics and reasoning methods applied to bridge. In the game-theory literature, the work of Wilson (1972) and Koller and Meggido (1992) are closely related to the results in this paper and are described further in Sections 4 and 5.

3. DEFINITIONS

This section presents the classic formal definition of an imperfect information game. There are four key ideas. The first is the notion, familiar to AI researchers, of describing a game by its *game tree*. The second is the mechanism called *information sets* for incorporating imperfect information into the game tree description. The third is the classic game-theoretic definition for what constitutes a solution to our games, namely, an *equilibrium point*. The fourth key idea is the definition of a certain subclass of imperfect information games called *perfect recall* games. This subclass is important both to the $\mathcal{NP}$-hardness results of Section 4 and to the correctness of the `IMP-minimax` and `IMP-alpha-beta` algorithms described in the rest of this paper.

Any game can be expressed in the so-called extensive (tree) form (Luce and Raiffa 1957; Shubik 1982; Rasmusen 1989) introduced in Kuhn (1953).[6] Here, each position reachable during the game is encoded by a node in the game tree, with the root node encoding the initial position of the game. From each node X in the tree, the arcs emanating from X encode the various alternatives (moves) available to the player who moves from the game position encoded by X. Each leaf of the game tree encodes a position in which the game has ended, and contains the payoffs to the players at that position. Formally, an n-player game Γ consists of

- A finite tree $\mathcal{K}$ called the game tree.[7] The edges below any interior node x in $\mathcal{K}$ are the alternatives from x.
- A partition of the interior nodes in $\mathcal{K}$ into $n + 1$ classes: the chance nodes and the player-k nodes, for k from 1 to n.
- For each chance node x in $\mathcal{K}$, a probability distribution on the alternatives from x.
- For each k, $1 \leq k \leq n$, a partition of the player-k nodes in $\mathcal{K}$ into information sets such that for any nodes x and y in the same information set:
  - —The number of children below x equals the number of children below y.
  - —If $x \neq y$, then neither node is an ancestor of the other.
- A payoff function h from the leaves of $\mathcal{K}$ to n-tuples of real numbers.

A zero-sum game is one in which at any leaf, the payoffs to the players sum to zero.[8]

To see that the above definition captures our informal notion of an n-player game, think of the root of $\mathcal{K}$ as the initial position of the game. To play the game means to follow a root-to-leaf path in the tree, with each edge on the path corresponding to a single move in the game. If a chance node x is encountered during the play, then "nature" will determine the edge below x in the root-to-leaf path, at random and according to the probability distribution associated with x. If a player-k node is encountered, then player k will choose the edge (next

[6]The extensive form contrasts with the so-called normal form. In the latter form, the game is reduced to a single-move game in which each player simultaneously selects a strategy for the game. The payoffs to the players are whatever payoffs would occur if the players were to use the selected strategies. In general, the extensive form of the game is an exponentially smaller representation of the game than the normal form and hence, more useful when the focus is on algorithms, as is the case here.

[7]By a tree, we mean a rooted tree with an associated parent function. With regard to trees, we use without explanation terms like *node*, *edge*, *path*, *depth*, *root*, *leaf*, *interior node*, *parent*, *child*, *ancestor*, and *descendant*. (A node is both an ancestor and descendant of itself.) See any standard text on data structures, for example, Aho *et al.* (1983), for definitions of these terms.

[8]For two-player games, this means that what is good for one player is equally bad for the other player. Most parlor games (for example, chess, bridge, and poker) are zero-sum games.

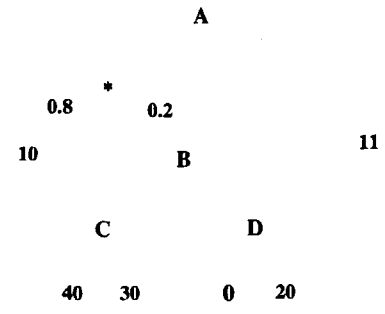

FIGURE 3. A two-player, zero-sum, imperfect information game tree. Player 1 moves at square-nodes; player 2 moves at circle-nodes. The sole chance node is marked by a diamond. At each leaf, the payoff to player 1 is given.

move). The outcome of the game for player k is the k^{th} component of the payoff vector $h(w)$ at the leaf w reached by the play. Figure 3 shows a simple two-player, zero-sum game tree, which we use to illustrate the definitions that follow.

A pure strategy π_k for player k on $\mathcal{K}$ is a function on the player-k nodes in $\mathcal{K}$, such that for any player-k nodes x and y in $\mathcal{K}$:

- $\pi_k(x)$ is a child of x.
- If x and y are in the same information set, $\pi_k(x)$ and $\pi_k(y)$ are the same alternative (i.e., if $\pi_k(x)$ is the j^{th} child of x, then $\pi_k(y)$ is the j^{th} child of y).[9]

A pure strategy π in an n-player game Γ is an n-element vector whose k^{th} component, π_k, is a pure strategy for player k.

"What a player knows" is reflected in the pure strategies available to the player, which are determined by the information sets. If two nodes x and y are in the same information set, then the player "cannot tell them apart," because by definition the player's pure strategy must be the same (choose the jth child, for some fixed j) on the two nodes. Thus, when there exists an information set with more than one node in it, the game is said to exhibit imperfect information.[10]

The quality of a pure strategy is measured by its expected payoff, which, in turn, depends on the probability of reaching leaf nodes. Given pure strategy π on game tree $\mathcal{K}$, the probability of node x under π, denoted $p_\pi(x)$, is defined to be the product of the probabilities of the arcs on the path from the root to x, with each arc below a nonchance node granted probability 1 or 0, depending on whether or not π selects that arc. The expected payoff to player k under pure strategy π, denoted $H_k(\pi)$, is defined to be $\sum p_\pi(w)\, h_k(w)$, where the sum is over all leaves w in $\mathcal{K}$ and $h_k(w)$ is the k^{th} component of payoff vector $h(w)$.[11]

A player-k pure strategy π_k is optimal for pure strategy π if for every player-k pure strat-

[9] In Fig. 3, player 2 has two pure strategies: go left or right from node B. Player 1 has four pure strategies: left/right from node A, and for each of those choices, left/right from the information set containing nodes C and D.

[10] Fig. 3 is a game with imperfect information because of the information set containing nodes C and D. Player 1 must either go left from both nodes C and D or right from both.

[11] In Fig. 3, consider the strategy in which all players move left at all choices. Under this strategy, the probability for each leaf is zero, except for the leaf labeled 10 (which has probability .8) and the leaf labeled 40 (which has probability .2). The expected payoff to player 1 under this strategy is $.8 * 10 + .2 * 40 = 16$.

egy α_k, we have $H_k(\pi) \geq H_k(\alpha)$, where α is the same as π except that the k^{th} component of π is π_k while the k^{th} component of α is α_k. A pure strategy π is a pure-strategy equilibrium point if each component π_k of π is optimal for π. Thus, in an equilibrium point, no player can strictly improve her expected payoff by a unilateral change in strategy.[12] All perfect information games and all one-player games have a pure strategy equilibrium point, but some imperfect information games with more than a single player have no pure strategy equilibrium point.

As we will see in Sections 4 and 5, imperfect information games with "perfect recall" are of particular interest. Informally, perfect recall means the player recalls her previous moves. More precisely, for information sets R and S in game Γ, we say that S follows R if $S \neq R$ and there exists nodes $r \in R$ and $s \in S$ such that s is a descendant of r. For any subset X of an information set, the i^{th} child of X is the set of all nodes y such that y is the i^{th} child of a node in X. A game Γ has perfect recall[13] if for every pair of player-k information sets R and S in Γ such that S follows R, there exists an i such that S lies entirely inside the forest of subtrees rooted at the i^{th} child of R.[14] Note that perfect recall is not the same as perfect information.

It is often possible to know from the phenomenon being modeled whether or not it has perfect recall. For example, consider the card game bridge. This game is played by four players organized into two teams. If one chooses to model the game as a four-player game, then it is a game with perfect recall, assuming that each player is capable of remembering the cards that have been played. Alternatively, one can model the game as a two-player (team) game. In this case, the game will not have perfect recall. After the initial lead, each agent sees the cards of the "dummy" (say, North) and her own cards. When the East/West team makes a play from the West hand, it "knows" the contents of the West and North hands. Later, when the same East/West team makes a play from the East hand, it has "forgotten" the contents of the West hand. In this two-team representation, the rules of the game require that the East/West team "forget" some of what it knew at its previous turn. Thus, the two-team representation of bridge lacks perfect recall.

4. FINDING PURE STRATEGY EQUILIBRIA IS $\mathcal{NP}$-HARD

The previous two sections describe imperfect information games and motivate the study of their complexity. This section presents our negative ($\mathcal{NP}$-hardness) results. The next section presents positive results, in the form of an algorithm (`IMP-minimax`), that is correct for a special case and can be used as a natural heuristic in the general case. Succeeding sections present an improvement to `IMP-minimax` called `IMP-alpha-beta` and an analysis of the magnitude of that improvement.

Subsection 4.1 explains what we mean by the "complexity" of games, presents the relevant complexity results known previously, and summarizes our new $\mathcal{NP}$-hardness result. Subsection 4.2 presents the formal statement of our new $\mathcal{NP}$-hardness result and its proof.

[12]The game in Fig. 3 has a single pure strategy equilibrium point, in which player 1 moves left from node A and right from information set C/D, while player 2 moves right from node B.

[13]It is easy to show that this definition of perfect recall, which is given by Thompson (1953), is equivalent to the definition given by Kuhn (1953).

[14]Game tree G_2 on the right side of Fig. 2 in Section 2 is an example of an imperfect information game that lacks perfect recall. The same game tree would have perfect recall if nodes A and B were placed into a single information set or if nodes A and B were player-1 moves while nodes C and D were player-2 moves. The game tree in Fig. 3 is another example of an imperfect information game with perfect recall.

Subsection 4.3 presents some corollaries to this result and then summarizes the consequences of these $\mathcal{NP}$-hardness results.

4.1. Background

An equilibrium point in a game is a collection π of strategies, one per player, such that no single player can unilaterally improve her payoff by changing her strategy from that specified by π. The notion of an equilibrium point is a natural and accepted solution concept for games that have equilibrium points. The most "simple" equilibrium point is one in which all strategies are pure strategies, that is, nonrandomized strategies. Here we investigate the complexity of finding pure-strategy equilibrium points when they exist.

AI research on machine game-playing has concentrated on perfect information games, like chess or backgammon. For such games, a pure-strategy equilibrium point always exists, and the classical minimax (Zermelo 1913; Shannon 1950; Kuhn 1953) and alpha-beta (Slagle and Dixon 1969; Knuth and Moore 1975; Luckhardt and Irani 1986; Korf 1991) algorithms find one in time linear in the size of the game tree searched. This contrasts starkly with imperfect information games, like poker or bridge, as the following result shows.

Theorem 1 (Koller and Meggido 1992). The problem of finding a pure-strategy equilibrium point in an imperfect information game is $\mathcal{NP}$-hard, even if there is only a single player (in which case a pure-strategy equilibrium point always exists).

An imperfect information game has *perfect recall* if, loosely speaking, the players never forget anything they once knew. (See Section 3 for a formal definition.) Games like bridge in which players are teams of cooperating but uncommunicating players, can be modeled as games without perfect recall, whereas games in which players are single-entity agents are typically modeled as games with perfect recall. For one-player games with imperfect information but perfect recall, there is a linear-time algorithm (`IMP-minimax`, as described in Section 5) for finding a pure-strategy equilibrium point (Wilson 1972; Koller and Meggido 1992). Hence, the perfect-recall property forms a natural dividing line for one-player games: the class of games without perfect recall is $\mathcal{NP}$-hard, while games with perfect recall can be solved in linear time.

Here we show that the perfect-recall property does not form such a dividing line for games with more than one player. We show that finding a pure-strategy equilibrium point in an imperfect information, perfect recall, n-player game, given that such an equilibrium point exists, is $\mathcal{NP}$-hard, for any $n \geq 2$. This provides an interesting contrast not only with the previous results for one-player games, but also with the fact that there is a polynomial-time algorithm for finding *behavior-strategy* (randomized) equilibrium points in two-player, imperfect information, perfect-recall games (Koller and Meggido 1992).

4.2. NP-Hardness Results

Our main result shows that it is $\mathcal{NP}$-complete to determine whether or not an n-player game ($n \geq 2$) has a pure-strategy equilibrium point, even if the game has perfect recall. In the format of Garey and Johnson (1979) the decision problem is:

PERFECT RECALL, PURE STRATEGY EQUILIBRIUM (PR-PSE):
Instance: An n-player perfect-recall game Γ.
Question: Does Γ have a pure-strategy equilibrium point?

Theorem 2. PR-PSE with $n = 2$ is $\mathcal{NP}$-hard.

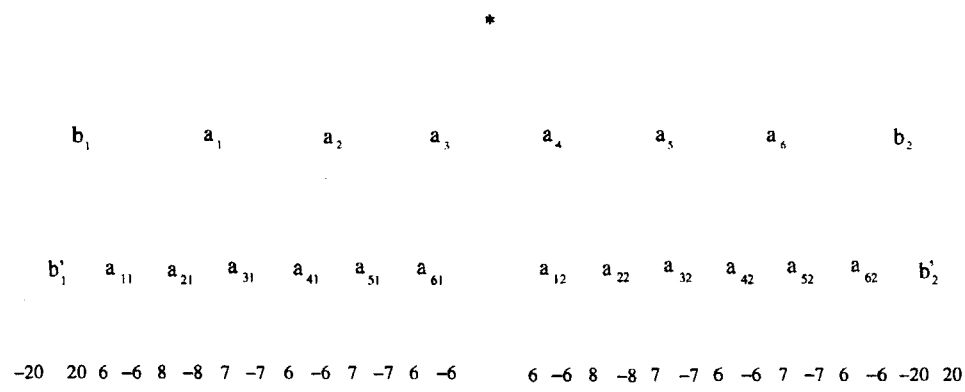

FIGURE 4. The constructed game tree for $m = 2$, sizes $s(a_1) = 6$, $s(a_2) = 8$, $s(a_3) = 7$, $s(a_4) = 6$, $s(a_5) = 7$, $s(a_6) = 6$, and $B = 20$. Payoff values to player 1 are shown at leaves.

Proof. The reduction is from 3-PARTITION (Garey and Johnson 1979, p. 224).

3-PARTITION:
Instance: Finite set A of $3m$ elements, bound $B \in Z^+$, and a "size" $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and such that

$$\sum_{a \in A} s(a) = mB.$$

Question: Can A be partitioned into m disjoint sets $S_1, S_2, \ldots, S_m$ such that

$$\sum_{a \in S_j} s(a) = B$$

for $1 \leq j \leq m$? (Notice that the constraints on the item sizes imply that every S_i in a valid partition must contain *exactly* three elements from A.)

Let $A = \{a_1, a_2, \ldots, a_M\}$ be the set of elements (where $M = 3m$), s the size function, and B the bound in an arbitrary instance of 3-PARTITION. We will construct a two-player, zero-sum, perfect-recall game Γ such that there is a pure-strategy equilibrium point $\pi = (\pi_1, \pi_2)$ for Γ if and only if there is a partition of A into m disjoint subsets $S_1, S_2, \ldots, S_m$ such that $\sum_{a \in S_j} s(a) = B$, for $1 \leq j \leq m$. It will follow that PR-PSE is $\mathcal{NP}$-hard for $n = 2$. An example of an instance Γ constructed in the manner specified in succeeding paragraphs is given in Fig. 4.

The root of the game tree for Γ is a chance node with $4m$ children, and each child below the root is equally likely.

The nodes immediately below the root are the player-1 nodes, each in an information set by itself. Of these $4m$ nodes, m are called *balancing nodes* and are labeled $b_1 \ldots b_m$. The remaining $3m$ nodes correspond to the $3m$ elements of A, and hence are labeled $a_1 \ldots a_M$, where $M = 3m$. Thus, we use a_i to refer both to the i^{th} element of set A and also to the i^{th} nonbalancing node at depth 1 of the tree.

Each balancing node b_j has exactly one child, labeled b'_j. Each nonbalancing node a_i has exactly m children, labeled $a_{i1} \ldots a_{im}$. These nodes below the player-1 nodes are all player-2 nodes. They are separated into m information sets, with the j^{th} information set I_j containing

$1 + 3m$ nodes: the sole child b'_j of player-1 node b_j and the j^{th} child a_{ij} of each player-1 node a_i, for i from 1 to $3m$.

There are two leaves below each player-2 node. The payoffs to player 1 at the leaves are as follows:

$$\begin{aligned} \text{payoff at the left child of } b'_j &= -B \\ \text{payoff at the right child of } b'_j &= B \\ \text{payoff at the left child of } a_{ij} &= s(a_i) \\ \text{payoff at the right child of } a_{ij} &= -s(a_i) \end{aligned}$$

The payoff to player 2 is the negative of the payoff to player 1. The game has perfect recall because there is only one level of nodes for each player.

It is easy to see that the construction of Γ can be accomplished in time polynomial in $|A|$. All that remains is to show that there is a pure-strategy equilibrium point $\pi = (\pi_1, \pi_2)$ for Γ if and only if there is a partition of A into m disjoint subsets $S_1, S_2, \ldots, S_m$ such that $\sum_{a \in S_j} s(a) = B$, for $1 \leq j \leq m$.

By construction of the game Γ, the expected payoff[15] $H(\pi_1, \pi_2)$ for the pure-strategy pair (π_1, π_2) is

$$\sum_x \frac{1}{4m} (\text{payoff at } x) = \frac{1}{4m} \sum_x (\text{payoff at } x)$$

where the sum is over all leaves x selected by (π_1, π_2). For any set X of nodes in the game tree, we use the phrase *contribution to* $H(\pi_1, \pi_2)$ *of* X to mean the sum of the payoffs at all leaves x below X selected by the pure-strategy pair (π_1, π_2).

First, suppose there exists a partition $S_1, \ldots, S_m$ satisfying the requirements of 3-PARTITION. Consider the player-2 strategy π_2 that goes right from every player-2 information set and the player-1 strategy π_1 that chooses the child a_{ij} below node a_i if and only if $a_i \in S_j$ in the partition. We claim that the pair of strategies (π_1, π_2) is an equilibrium point. To see this, consider what happens if we fix player 1's strategy at π_1 and allow player 2 to change his strategy to π'_2. For any player-2 information set I_j, the player-1 strategy π_1 selects exactly four nodes in I_j: node b'_j and the three nodes a_{pj}, a_{qj} and a_{rj}, where a_p, a_q, and a_r are the elements in set S_j. Hence, the contribution to $H(\pi_1, \pi'_2)$ of I_j is

$$-B + s(a_p) + s(a_q) + s(a_r)$$

if player 2 goes left at I_j under π'_2, and is

$$B - s(a_p) - s(a_q) - s(a_r)$$

if player 2 goes right at I_j under π'_2. Since $\sum_{a \in S_j} s(a) = B$, it follows that the contribution to $H(\pi_1, \pi'_2)$ of each player-2 information set I_j is zero, regardless of player 2's strategy π'_2. Hence, the total expected payoff $H(\pi_1, \pi'_2)$ is zero for every player-2 strategy π'_2; trivially, player 2 cannot improve his strategy.

Suppose now that we fix player 2's strategy at π_2 and allow player 1 to change her strategy to π'_1. Then, for each balancing node b'_j, the contribution to $H(\pi'_1, \pi_2)$ of b'_j is B. For each a_i, player 1 selects exactly one node a_{ik} below it, yielding payoff (under π_2) of $-s(a_i)$; hence,

[15] Throughout the remainder of this proof, by "payoff" we mean the payoff to player 1, and we use $H(\pi_1, \pi_2)$ to refer to the expected payoff $H_1(\pi_1, \pi_2)$ to player 1.

the contribution to $H(\pi_1', \pi_2)$ of a_i is $-s(a_i)$. Regardless of player 1's strategy π_1', the total expected payoff $H(\pi_1', \pi_2)$ is $\frac{1}{4m}$ times

$$mB - \sum_{a \in A} s(a) = mB - mB = 0,$$

and hence, trivially, player 1 cannot improve her strategy. It follows that (π_1, π_2) is an equilibrium point.

Second, consider the case when there is no partition satisfying the requirements of 3-PARTITION. We will show that there is no pure-strategy equilibrium point in Γ. To show this we first show that if player 1's strategy is arbitrarily fixed at, say, π_1, then there is a player 2 strategy π_2 for which $H(\pi_1, \pi_2)$ is strictly less than zero. Then we will show that if player-2's strategy is arbitrarily fixed at, say, π_2', then there is a player-1 strategy π_1' for which $H(\pi_1', \pi_2')$ is nonnegative. It will then follow that there is no pure-strategy equilibrium point.

Let π_1 be any player-1 strategy. Consider any player-2 information set I_j, and let $A_j = \{a_{ij} \mid \pi_1$ chooses a_{ij} from some $a_i\}$. The contribution to $H(\pi_1, \pi_2)$ of I_j is

$$-B + \sum_{a_{ij} \in A_j} s(a_i)$$

if player 2's strategy π_2 goes left from I_j; otherwise, the payoff will be the negative of the above. It follows that if player 2 chooses the strategy π_2 of going left at I_j when $\sum_{a_{ij} \in A_j} s(a_i) \leq B$, and right otherwise, then for each player-2 information set I_j, the contribution to $H(\pi_1, \pi_2)$ of I_j is no more than zero. Since there is no partition in the instance of 3-PARTITION with all sets having total size B, there must be at least one player-2 information set for which $\sum_{a_{ij} \in A_j} s(a_i) > B$. Thus, the contribution to $H(\pi_1, \pi_2)$ of that information set will be negative; hence, the total expected payoff $H(\pi_1, \pi_2)$ will be negative.

Now let π_2' be any player-2 strategy. We consider two cases.

Case 1: At all player-2 information sets I_j, strategy π_2' goes right. In this case, for each node b_j', the contribution to $H(\pi_1', \pi_2')$ of nodes below b_j' is B, regardless of the player-1 strategy π_1'. Furthermore, π_1' selects exactly one node a_{ij} from each node a_i, and the payoff below a_{ij} will be $-s(a_i)$. It follows that the total expected payoff $H(\pi_1', \pi_2')$ is $\frac{1}{4m}$ times

$$mB - \sum_{a \in A} s(a) = 0$$

and, hence, nonnegative.

Case 2: At some player-2 information set I_j, strategy π_2' goes left. Let x be the number of information sets at which player 2 goes left, and let I_j be any particular information set for which player 2 goes left. Consider the player-1 strategy π_1' in which for each node a_i, player 1 chooses a_{ij}. Then the contribution to $H(\pi_1', \pi_2')$ of I_j is

$$-B + \sum_{a \in A} s(a) = (m-1)B.$$

The contribution to $H(\pi_1', \pi_2')$ of each of the $m - x$ information sets at which player 2 goes right is B, and the contribution at each of the $x - 1$ information sets other than I_j at which player 2 goes left is $-B$. Thus, the total expected payoff $H(\pi_1', \pi_2')$ is $\frac{1}{4m}$ times

$$(m-1)B + (m-x)B - (x-1)B = (2m-2x)B.$$

Since x is not more than m, we have a total payoff that is nonnegative, completing the proof of the theorem. ■

4.3. Consequences of the $\mathcal{NP}$-Hardness Results

Previous results had shown that the perfect-recall property forms a natural dividing line for one-player games: the class of games without perfect recall is $\mathcal{NP}$-hard, whereas games with perfect recall can be solved in linear time. Our $\mathcal{NP}$-hardness result in the previous subsection shows that the perfect-recall property does not form such a dividing line for games with more than one player. The following three corollaries to Theorem 2 address further complexity issues for imperfect information games with more than one player. (See Blair and Mutchler (1995) for proofs of the corollaries.)

1. PR-PSE is $\mathcal{NP}$-complete for any $n \geq 2$, even if all players but one have perfect information and the game is zero-sum with only one level of decisions per player.
2. Given a zero-sum, perfect-recall game Γ in which a pure-strategy equilibrium point exists, it is $\mathcal{NP}$-hard to find such an equilibrium point.
3. Given a two-player, zero-sum, perfect-recall game Γ, it is $\mathcal{NP}$-hard to find a so-called max-min pure strategy (as opposed to a pure strategy equilibrium point).

Theorem 2, together with the $\mathcal{NP}$-hardness result of Koller and Meggido (1992), implies that for one-player games without perfect recall, and for games (with or without perfect recall) with more than one player there is no efficient algorithm for finding pure strategy equilibria (when they exist), unless $\mathcal{P} = \mathcal{NP}$. Several directions are available to resolve this dilemma. One can seek algorithms for special cases. Or, one can seek heuristics for the general case. Both these approaches are taken in the next section, where a classic algorithm (`IMP-minimax`) for finding pure strategy equilibria in one-player perfect-recall games is discussed. As explained in Section 5.3, this algorithm can be used as a natural heuristic in the general case. The attractiveness of `IMP-minimax` is further enhanced by its similarity to the minimax algorithm used so successfully (in the form of alpha-beta) for many perfect information games.

Another approach is to abandon pure-strategy equilibria and instead seek mixed-strategy (randomized) equilibria. A mixed strategy is any probability distribution over the set of pure strategies. When using a mixed strategy to play a game, the player selects a pure strategy according to the probability distribution specified by the mixed strategy. For example, if the game has 6 pure strategies $\pi_1 \ldots \pi_6$, then the mixed strategy (1/2, 1/6, 0, 0, 1/3, 0) would select π_1 with probability 1/2, π_2 with probability 1/6, and π_5 with probability 1/3. Pure strategies are simpler, more natural to humans, and (in general) smaller than mixed strategies, which is why we focus on pure strategies in this paper. However, mixed strategies have two advantages of their own. First, all games have mixed-strategy equilibria (Nash 1951), whereas not all games have pure-strategy equilibria. Second, there are polynomial-time algorithms (Koller and Meggido 1992; Koller *et al.* 1994) for finding small, mixed-strategy equilibria in the special case of two-player, zero-sum, perfect-recall games. This contrasts sharply with the results in this section, which show that there is no polynomial-time algorithm for finding pure-strategy equilibria in the same special case unless $\mathcal{P} = \mathcal{NP}$. The best of these polynomial-time algorithms for finding mixed-strategy equilibria first constructs a certain sparse $M + 1$ by $N + 1$ matrix. Here M is the sum, over the information sets U for player 1, of the number of alternatives ("choices") available at information set U, and N is defined likewise for player 2. The algorithm continues by solving a certain linear programming problem specified by this matrix. Any natural implementation of this algorithm will require

space at least linear in the size of the game tree. In practice then, these algorithms for finding mixed-strategy equilibria are slower than the linear-time logarithmic-space `IMP-minimax` heuristic presented in the next section.

5. IMP-minimax

The $\mathcal{NP}$-hardness results of the previous section and those of Koller and Meggido (1992) show that, in general, there is no efficient algorithm for finding pure strategy equilibria (when they exist), unless $\mathcal{P} = \mathcal{NP}$. The rest of this paper responds to this dilemma by introducing and analyzing algorithms for special cases that can also be used as natural heuristics in the general case. This section introduces the first such algorithm—a classic algorithm due to Wilson (1972), which we call `IMP-minimax`. Succeeding sections present an improvement to `IMP-minimax`, called `IMP-alpha-beta`, and an analysis of the magnitude of that improvement.

Subsection 5.1 states the formal properties of `IMP-minimax` that make it an efficient algorithm for a special case. Subsection 5.2 presents the algorithm itself. Subsection 5.3 contains examples illustrating `IMP-minimax` and its properties.

5.1. Properties of `IMP-minimax`

`IMP-minimax` has the following two properties (Wilson 1972; Blair *et al.* 1992; Koller and Meggido 1992):

- The run time of `IMP-minimax` is linear in the number of nodes in the game tree.
- If the game has only a single player and has perfect recall, then the strategy computed by `IMP-minimax` is optimal. Otherwise, the strategy might or might not be optimal.

Thus, `IMP-minimax` is an efficient algorithm for a special case.

5.2. Statement of `IMP-minimax`

In its simplest form, `IMP-minimax` is an algorithm for *one-player* games. Here we present `IMP-minimax` in its top-down, recursive form, as it would be programmed in a game-playing program.[16] Before describing `IMP-minimax`, we define the functions it relies on, using the example in Fig. 5 to illustrate them.

- Any subset X of an information set is a partial information set, abbreviated PI-set. The j^{th} child of PI-set X is the set of all immediate descendants of nodes in X reached via the j^{th} alternative. For a strategy π, to say "set $\pi(X)$ equal to the j^{th} child of PI-set X" means to set $\pi(x)$ to the j^{th} child of x for each x in the information set containing X.
- For any node x in $\mathcal{K}$, its reachable probability $prob(x)$ is the product of the probabilities below chance nodes on the path from the root to x.[17] We also permit function $prob$ to take a set X of nodes, in which case it returns $prob(x)$ summed over all nodes $x \in X$.
- Function $payoff(x)$ specifies the payoff (to the sole player) at leaf x of $\mathcal{K}$.

[16] Wilson (1972) and Koller and Meggido (1992) present `IMP-minimax` in a bottom-up fashion.

[17] For the example of Fig. 5, $prob$(leaf whose value is -100) is $1/6$, since the probability distributions below chance nodes are all the uniform distribution.

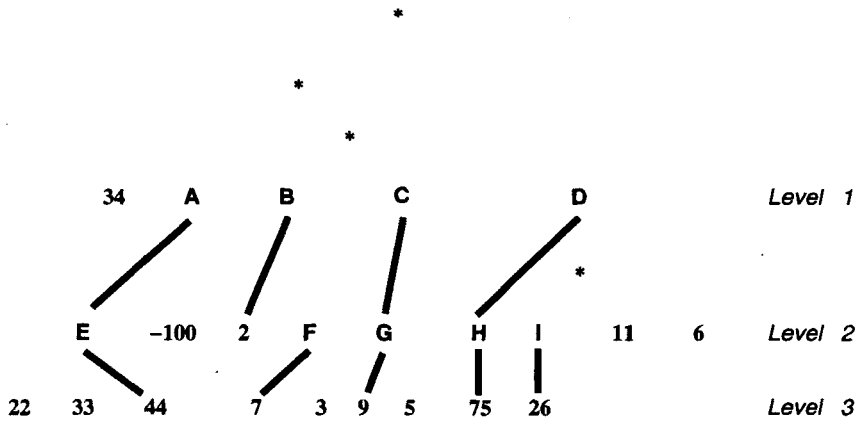

FIGURE 5. A one-player game tree. For any chance node (labeled *), the arcs directly below it are equally likely; player nodes are labeled with letters; and leaves are labeled with their respective payoff values. Ellipses are used to show information sets that contain more than one node. The thick lines indicate the strategy selected by IMP-minimax.

- Function *extend* takes a set X of nodes in $\mathcal{K}$ and returns the set obtained by recursively replacing each chance node in X by its children until the resulting set contains no chance nodes.[18]
- Function *partition* takes a set X of nodes in $\mathcal{K}$ and first separates the leaves from the nonleaves in X, then partitions the nonleaf nodes into their respective information sets.[19]

Figure 6 shows the IMP-minimax algorithm. Note that IMP-minimax both returns a value (per the V function) and computes a strategy π^* for $\mathcal{K}$.[20] It should be clear that IMP-minimax assigns an alternative to each information set in the tree and, thus, computes a strategy for $\mathcal{K}$.

5.3. Behavior of IMP-minimax

In general, the strategy computed by IMP-minimax is not an optimal strategy. For example, by tracing IMP-minimax on the game tree G_2 in Fig. 2, one can see that the information set containing C and D is encountered twice; once to compute $V(C)$ and later to compute $V(D)$. The action of π^* on that information set is determined by the computation of $V(D)$ (which overwrites the action determined by the computation of $V(C)$). Thus, the strategy has expected payoff $-99/2$, whereas an optimal strategy has payoff $1/2$.

The above example illustrates what can go wrong when using V to compute a strategy: information sets can be encountered more than once, with no effort to make the same choice each time. Wilson (1972) showed that such an event is impossible when there is perfect recall in a one-player game, in which case IMP-minimax computes an optimal strat-

[18]For the example of Fig. 5, *extend* ({*root of the tree*}) yields the five nodes labeled *Level 1* in the figure; *extend* applied to the children of the Level 1 nodes yields the Level 2 nodes, and so on.

[19]For the example of Fig. 5, letting -100 and 2 denote the leaves whose values are -100 and 2 respectively, we have that *partition* applied to $\{E, -100, 2, F, H, I\}$ returns $\{\{E\}, -100, 2, \{F\}, \{H, I\}\}$.

[20]The reader may find it instructive to trace the operation of IMP-minimax on the example in Fig. 5. There, the value returned by V applied to *extend* ({*root of* $\mathcal{K}$}) is $617/12$. The strategy π^* computed is shown in the figure by thick lines.

IMP-minimax: call V (*extend* ({*root of the game tree*})), where the recursive function $V(X)$ takes a set X of nodes in the game tree and is given by:

$$V(X) = \begin{cases} \max \{ V(\textit{extend}(Y)) \mid Y \text{ is a child of } X \} & \text{if } X \text{ is a PI-set} \\ \sum_{\substack{x \in \textit{partition}(X) \\ x \text{ a leaf}}} \textit{prob}(x)\ \textit{payoff}(x) + \sum_{\substack{x \in \textit{partition}(X) \\ x \text{ a PI-set}}} V(x) & \text{otherwise} \end{cases}$$

Each time in the recursion that the argument X is a single PI-set, set $\pi^*(X)$ equal to the child of X obtained via the alternative selected by the *max* operator in the calculation of $V(X)$.

FIGURE 6. IMP-minimax.

egy. Again, consider the game tree G_2 in Fig. 2, but with A and B in a single information set. (Thus, the game has imperfect information but perfect recall.) Here the computation of V(*extend* ({*root of modified* G_2})) returns the optimal payoff (0) and an optimal strategy (go right at $\{A, B\}$).

In Blair *et al.* (1992) we describe a class of one-player games that illustrates the magnitude of the improvement IMP-minimax can achieve over the naive, examine-all-strategies algorithm for games with perfect recall. For arbitrary game-tree depth d, there exists a game in the class that allows $2^{(4^{d-1}-1)/3}$ different strategies. The naive algorithm iterates over all possible strategies, computing the expected value of each and selecting the strategy whose expected value is maximal. Thus, its execution time is Ω (number of strategies). IMP-minimax, which takes advantage of the game having perfect recall, has execution time O(number of nodes in the tree). These quantities are sharply contrasted when considering game trees with increasing depth; for example, with depth 8 the number of nodes in the game tree is 16321, whereas the number of strategies is 8.4×10^{1643}.

IMP-minimax can be extended naturally to *two-player* zero-sum games with imperfect information by replacing the max operator in IMP-minimax with a min operator at the player 2 PI-sets. This natural heuristic remains efficient (linear in the size of the game tree searched) in this two-player form. Since IMP-minimax finds a pure-strategy equilibrium point in one-player games with perfect recall, one might expect it to do the same for two-player games with perfect recall, when such an equilibrium point exists. However, not only does IMP-minimax not accomplish this goal, neither can any efficient algorithm (unless $\mathcal{P} = \mathcal{NP}$), as is shown by Theorem 2 in Section 4.[21]

6. INFORMATION SET PRUNING

The previous section introduces to the AI community a classic algorithm due to Wilson (1972), which we call IMP-minimax. This algorithm is correct for a special case (single

[21] Fig. 3 in Section 3 is a two-player game with perfect recall on which IMP-minimax finds a pure-strategy equilibrium point. For a simple example on which IMP-minimax fails, modify game tree G_1, which appears on the left side of Fig. 2 in Subsection 2.2, as follows: first, convert G_1 into a two-player game by inserting a player-2 node E between A and C (with sole child C) and another player-2 node F between B and D (with sole child D). Then, convert G_1 into an imperfect information game by placing A and B into one information set and C and D into another while leaving E and F in separate information sets. The resulting game has perfect recall. It has two pure-strategy equilibrium points, under both of which player 1 moves right from information set $\{A, B\}$ to obtain a score of 0. However, IMP-minimax chooses a suboptimal strategy that moves left from information set $\{A, B\}$, achieving an expected payoff of $-99/2$.

player, perfect recall) and can be used as a natural heuristic in the general case. `IMP-minimax` is to imperfect information games as minimax is to perfect information games. Here we introduce `IMP-alpha-beta`, which is to `IMP-minimax` as alpha-beta is to minimax. That is, `IMP-alpha-beta` computes the same value as does `IMP-minimax` but usually faster through *pruning* (i.e., not examining the value of some nodes). The next section analyzes the magnitude of the improvement of `IMP-alpha-beta` over `IMP-minimax`.

This section first shows that, in general, pruning is not possible in one-player games. Then, we show through examples that pruning is possible given a certain natural assumption about the payoffs at leaves. Finally, we present the `IMP-alpha-beta` algorithm that incorporates such pruning.

The following theorem shows that, in general, pruning is not possible in one-player games. We assume for the theorem that the probabilities at arcs below chance nodes are all non-zero.

Theorem 3. Let $\mathcal{A}$ be any algorithm that correctly solves this problem: given an arbitrary one-player game, return the value of that game. Then, for any one-player game G given as input to algorithm $\mathcal{A}$, every leaf in G is examined by $\mathcal{A}$. (That is, $\mathcal{A}$ determines the payoff of every leaf in G.)

Proof. By way of contradiction, suppose there were a correct algorithm $\mathcal{A}$ that determined the value of some one-player game G without examining some leaf X of G. Let M denote the maximum, over all leaves in G, of the payoffs at those leaves. Let p denote the product of the probabilities below chance nodes on the path from the root of G to node X. Construct a new game G' that is the same as G except that the payoff at leaf X in G' is $\frac{M+1}{p}$. By choice of X and construction of G', algorithm $\mathcal{A}$ computes the same value for G and G'. But this contradicts the correctness of $\mathcal{A}$—the value of game G is at most M, and the value of game G' is at least $\frac{M+1}{p}\, p = M + 1$. ■

Fortunately, a simple assumption permits pruning in one-player games: suppose there is a known upper bound on the payoffs at leaves. This assumption is quite reasonable in practice and is also used in multiplayer pruning (Luckhardt and Irani 1986; Korf 1991) and chance-node pruning (Ballard 1983). We introduce a new form of pruning, *information set pruning*, which assumes such an upper bound. Before formally stating the `IMP-alpha-beta` algorithm that implements this pruning, we show how it works through three examples. In each example, we assume that the children of the chance node at the root are equally probable.

Example 1. Consider the game tree in Fig. 7, where the upper bound on payoffs at leaves is 10. The left alternative from the information set gives an average payoff of $\frac{9+7}{2} = 8$, whereas the right alternative can give at most $\frac{2+10}{2} = 6$. Hence, an algorithm that has determined the three values specified in the figure need not examine the subtree labeled ``?.''

Information set pruning is available because the value of an alternative A from an information set X is the sum of the values of the leaves and/or information sets into which X fragments via alternative A. (Confer the second case in the V function for `IMP-minimax`.) Knowing the values of some elements of this sum and bounding the values of the other elements of the sum by using the upper bound on payoffs at leaves provides an upper bound u on the value of X via alternative A. If information set X is known to have (via another alternative B) a value higher than u, then the remaining terms of the sum can be pruned.

Example 2. Consider the game tree in Fig. 8, where again the upper bound on payoffs at leaves is 10. Let T denote the top-level information set; it contains 13 nodes. The value of

FIGURE 7. Information set pruning, a simple example.

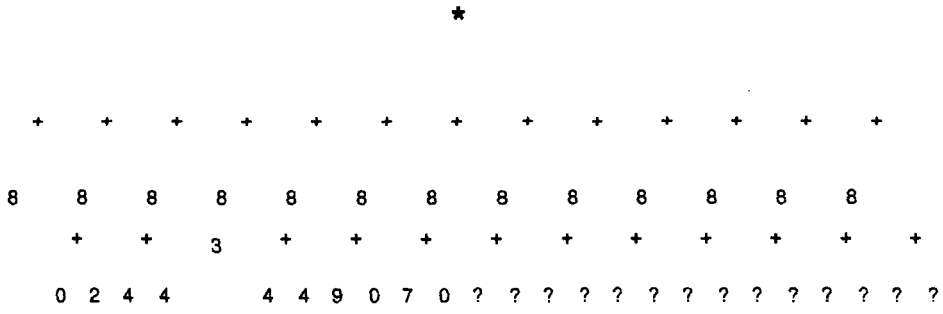

FIGURE 8. Information set pruning, a more elaborate example.

T via its left alternative is $\frac{8+\cdots 8}{13} = 8$. The value of T via its right alternative is the weighted sum of the four information sets below T and the leaf whose payoff is 3. The first two of those information sets have values $\max\{\frac{0+4}{13}, \frac{2+4}{13}\} = 6/13$ and $\max\{\frac{4+9+7}{13}, \frac{4+0+0}{13}\} = 20/13$, respectively. The last two of those information sets have values at most $2\frac{10}{13}$ and $5\frac{10}{13}$, respectively; hence, the value of T via its right alternative is at most $\frac{6}{13}+\frac{3}{13}+\frac{20}{13}+2\frac{10}{13}+5\frac{10}{13} = 99/13$, which is less than the value (8) of T via its left alternative. Both of the forests below the rightmost two information sets at depth 2 can be pruned.

As in alpha-beta pruning, the source of the pruning bound for information set pruning can be either one level above the pruned nodes (shallow pruning) or many levels above the pruned nodes (deep pruning). Here is an example of the latter.

Example 3. In the game tree in Fig. 9 (again with 10 as the upper bound on payoffs at leaves), the node marked `??` can be pruned. This happens not because the left child of its parent information set has value $\frac{9+1}{4} = 5/2$, nor because the left child of its grandparent information set has value $\frac{7+0}{4} = 7/4$, but rather because the left child of its great-grandparent information set has value $\frac{8+8+8+8}{4} = 8$. As we will see, `IMP-alpha-beta` propagates this value (combined with other pruning information) down to the bottommost information set, where the pruning of node `??` occurs.

Information set pruning is different from alpha-beta pruning (Slagle and Dixon 1969; Knuth and Moore 1975), which requires both MAX and MIN nodes. Because our games have only a single player, alpha-beta pruning is available to us only in the weak form of *immediate pruning*: if the value of one alternative from an information set equals the highest value possible, then the remaining alternatives can be pruned. Information set pruning is more closely related to chance-node pruning (Ballard 1983), although the two are by definition not

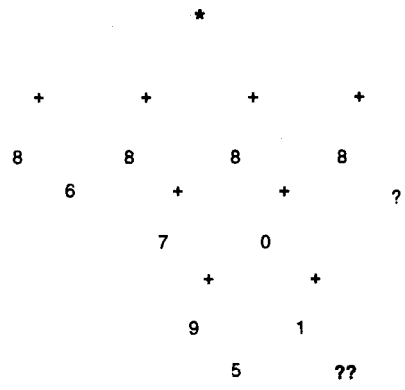

FIGURE 9. Information set pruning, deep pruning.

the same because chance-node pruning operates at chance nodes while information set pruning operates at collections of information sets that may be far removed from any chance nodes.

Algorithm `IMP-alpha-beta` appears in Fig. 10.[22] Scoping is indicated by indentation. Note that the two subfunctions call each other recursively, with the recursion ending in `Mixed-Set`. Function `Max-Set` takes a set of game-tree nodes, all of which are contained in a single information set. Function `Mixed-Set` also takes a set of nodes, but the set may contain leaves and/or nodes from multiple information sets.

Theorem 4. (`IMP-alpha-beta` *is correct.*) For any real number P and set X of nodes in a one-player game with imperfect information,

$$\texttt{Mixed-Set}\ (\textit{extend}\,(X),\ P) = \begin{cases} V\,(\textit{extend}\,(X)) & \text{if } V\,(\textit{extend}\,(X)) \geq P \\ P & \text{otherwise.} \end{cases}$$

In particular, `IMP-alpha-beta` computes the same value as `IMP-minimax`.

Proof. The proof is by induction on the number of recursive calls to V (van Lent 1993; Mutchler and van Lent 1995). ■

To obtain the strategy associated with the value `IMP-alpha-beta` returns, simply record in `Max-Set` the alternative with which *best* is associated. We note that an efficient implementation of `IMP-alpha-beta` must make various constant-time optimizations.

7. EFFECTIVENESS OF `IMP-alpha-beta`

The previous two sections present a classic algorithm `IMP-minimax` and a new algorithm `IMP-alpha-beta` that computes the same value as does `IMP-minimax` but usually does so faster through *pruning* (i.e., not examining the value of some nodes). In Mutchler and

[22]The behavior of most of the domain-specific functions is clear from their names: *alternative* finds the alternatives from an information set; *move* returns the nodes obtained from following a given alternative from a given information set; *payoff* returns the payoff of the given leaf; *root* refers to the root of the given game tree; and U is an upper bound on payoffs at leaves. The other three domain-specific functions are the functions *prob*, *extend*, and *partition*, as defined for `IMP-minimax` in Subsection 5.2.

IMP-alpha-beta: call `Mixed-Set` (*extend* ({*root*}), $-\infty$).

```
Max-Set (x, P)
        best = P
        for each alternative A from x
                temp = Mixed-Set (extend (move (A, x)), best)
                if temp > best
                        if temp == U * prob (x)
                                return (temp)
                        best = temp
        return (best)

Mixed-Set (X, P)
          sum = 0
          prob-left = prob (X)
          X := partition (X)
          for each member x of X
                  prob-left = prob-left − prob (x)
                  if x is a leaf
                          sum = sum + prob (x) * payoff (x)
                  else
                          sum = sum + Max-Set (x, P − sum − U * prob-left)
                  if sum + U * prob-left ≤ P
                          return (P)
          return (sum)
```

FIGURE 10. `IMP-alpha-beta`.

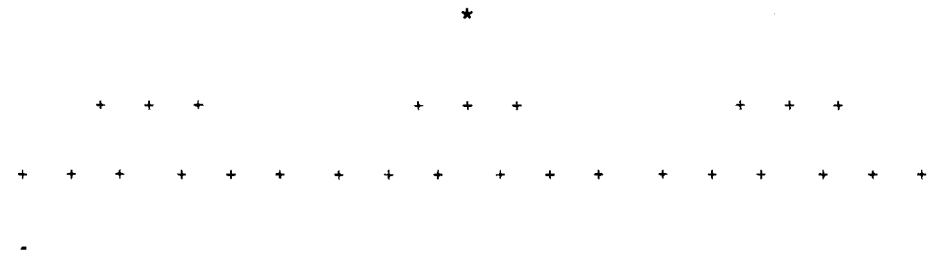

FIGURE 11. The IMP-model with $k = 3$, $b = 2$, and $d = 3$.

van Lent (1995) we present an analysis of the performance of `IMP-alpha-beta`. Here, we have space only to sketch the main conclusions.

In Mutchler and van Lent (1995) we motivate the use of the following IMP-model for the analysis. The IMP-model has three positive integers as parameters: k, b, and d. Each game tree within the model has a single chance node at the root of the tree. This chance node has k^{d-1} children all within a single information set. Each interior node of the game tree, except for the root, has b children. For each information set X in the game tree, the jth child of X (for j from 1 to b) is partitioned into k information sets of equal size; however, depth d nodes are leaves. The IMP-model describes only the structure of the game tree. Payoffs can be assigned by any method desired. Figure 11 shows an example of the IMP-model.

Theorem 5. `IMP-minimax` examines all $(kb)^{d-1}$ leaves in the IMP-model tree.

Theorem 6. For any setting of the parameters (k, b, and d) of the IMP-model with $k > 1$, and for any ordering of the loops in `IMP-alpha-beta`, there exist payoff values and a placement of those payoff values such that the number of leaves examined by `IMP-alpha-beta` can be[23]

- as many as $(kb)^{d-1}$ (that is, no pruning).
- as few as k^{d-1} (further, this lower bound is tight).
- as few as

$$\begin{cases} \dfrac{b^{d-1}(b-1) - k^{d-1}(k-1)}{b-k} & \text{if } k \neq b \\ b^{(d-2)}(bd - d + 1) & \text{if } k = b, \end{cases}$$

with none of the pruning due to immediate pruning.

The second result represents the extreme case when all the examined leaves give the upper bound and the rest of the tree is pruned through immediate pruning. The second and third results in the above theorem show that it is possible to prune vast portions of the game tree if the payoffs and placements are favorable, even if the effects of immediate pruning are ignored. The third result bears strong resemblance to the best-case behavior of alpha-beta, in which approximately $2b^{d/2}$ leaves out of b^d total leaves are examined (Knuth and Moore 1975). When k equals b, the above theorem shows that in the best case, fewer than db^{d-1} leaves will be explored out of $b^{2(d-1)}$ total leaves.

We also performed an extensive average-case analysis. The three main conclusions from the experiments are as follows, where $\alpha\beta$ and mm are the number of leaves examined by `IMP-alpha-beta` and `IMP-minimax`, respectively. First, $\alpha\beta/mm$ decreases as the depth d of the tree increases. Second, for fixed k and d, function $\alpha\beta/mm$ decreases as b increases. Third, for fixed b and d, function $\alpha\beta/mm$ increases as k increases.

In our average case study, less than 20% of the total nodes are examined in the more favorable cases (where $b = 10$ and $k = 2$). About 96% of the total nodes are examined in the least favorable case (where $b = 2$ and $k = 10$).

8. SUMMARY

Imperfect information games are an important and interesting class of games. This paper provides two sets of results fundamental to an understanding of algorithms for solving such games:

1. We show that finding a pure-strategy equilibrium point in an imperfect information, perfect recall, n-player game, given that such an equilibrium point exists, is $\mathcal{NP}$-hard, for any $n \geq 2$. This provides an interesting complement and contrast with Koller and Meggido's result that for *one-player* games, the perfect-recall property forms a natural dividing line. That is, in one-player games, finding pure strategy equilibria is $\mathcal{NP}$-hard in general but can be done in linear time in games with perfect recall.
2. Motivated by these $\mathcal{NP}$-hardness results, we introduce to the AI community a classic algorithm due to Wilson (1972) that solves a special case: one-player, imperfect information games with perfect recall. Wilson's algorithm, which we call `IMP-minimax`, is

[23]For brevity, we omit the $k = 1$ special case; it is given in Mutchler and van Lent (1995); van Lent (1993).

akin to minimax in that it does a single traversal of the game tree but is designed to handle the existence of information sets. We introduce a new algorithm, `IMP-alpha-beta`, that computes the same value as does `IMP-minimax` but usually does so faster through *pruning* (i.e., not examining the value of some nodes). Thus, `IMP-alpha-beta` is to `IMP-minimax` as alpha-beta is to minimax. Our analysis of `IMP-alpha-beta`, that includes both theorems bounding its performance and empirical data indicating its average-case behavior, suggests that `IMP-alpha-beta` will be a useful substitute for `IMP-minimax`.

AI researchers have had great success in developing programs for playing some *perfect information* games, notably, Othello (Lee and Mahajan 1990), backgammon (Berliner 1980), checkers (Schaeffer *et al.* 1993), and chess (Kopec *et al.* 1992). Much of this success comes from two ideas: the use of a static evaluator (Shannon 1950), and the existence of efficient algorithms (e.g., minimax, alpha-beta, conspiracy numbers (McAllester 1988; Schaeffer 1990) and singular extensions (Anantharaman *et al.* 1990) for solving perfect information games. It remains to be seen whether similar ideas will be effective for *imperfect information* games like bridge and poker. First, it is not clear whether the use of a static evaluator in such games will be as successful as it has been in games like chess. This issue can be determined only empirically, by building game-playing programs. Second, whereas perfect information games have efficient solution methods (e.g., alpha-beta), imperfect information games do not have efficient methods for finding pure strategy equilibrium points, unless $\mathcal{P} = \mathcal{NP}$.

ACKNOWLEDGMENTS

The authors are grateful to Robert Levinson, Barney Pell, and the anonymous reviewers for their detailed comments. Jim Plank provided software (*jgraph*) that was helpful in our viewing of the average case analysis results.

This research was supported by the National Science Foundation under grants IRI 89–10728 and CCR–9209803, by the Air Force Office of Scientific Research under grant 90–0135, and by the Naval Research Laboratory.

REFERENCES

AHO, A. V., J. E. HOPCROFT, and J. D. ULLMAN. 1983. Data structures and algorithms. Addison-Wesley, Reading, MA.

ANANTHARAMAN, T., M. S. CAMPBELL, and F. HSU. 1990. Singular extensions: adding selectivity to brute-force searching. Artificial Intelligence, **43**(1):99–109.

BALLARD, B. W. 1983. The $*$-minimax search procedure for trees containing chance nodes. Artificial Intelligence, **21**(1,2):327–350.

BAMPTON, H. 1994. Solving imperfect information games using the Monte Carlo heuristic. Master's thesis, University of Tennessee.

BARR, D. 1992. Experiments in heuristic search utilizing sampling and precomputation. Master's thesis, University of Tennessee.

BERLINER, H. J. 1980. Backgammon computer program beats world champion. Artificial Intelligence, **14**(2):205–220. Reprinted in Computer Games I. *Edited by* D. N. L. Levy. Springer-Verlag, New York, 1988.

BLAIR, J. R. S. and D. MUTCHLER. 1995. Pure-strategy equilibria in the presence of imperfect information—NP-hard problems. Technical Report 95-02, Department of Computer Science, Rose-Hulman Institute of Technology.

BLAIR, J. R. S., D. MUTCHLER, and C. LIU. 1992. Heuristic search in one-player games with hidden information. Technical Report CS-92-162, Department of Computer Science, University of Tennessee.

FINDLER, N. V. 1977. Studies in machine cognition using the game of poker. Communications of the ACM, **20**(4):230–245.

GAMBÄCK, B., M. RAYNER, and B. PELL. 1991. An architecture for a sophisticated mechanical bridge player. *In* Heuristic programming in artificial intelligence—the second computer olympiad. *Edited by* D. N. L. Levy and D. F. Beal. Ellis Horwood Limited, Chichester, England, pp. 87–107.

GAMBÄCK, B., M. RAYNER, and B. PELL. 1993. Pragmatic reasoning in bridge. Technical Report 299, Computer Laboratory, University of Cambridge.

GAREY, M. R. and D. S. Johnson. 1979. Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman, San Francisco.

GORDON, S. 1993. A comparison between probabilistic search and weighted heuristics in a game with incomplete information. *In* Games: planning and learning, papers from the 1993 Fall Symposium, Raleigh, NC, pp. 77–83. AAAI Press Technical Report FS-93-02, Menlo Park, CA.

KNUTH, D. E. and R. W. MOORE. 1975. An analysis of alpha-beta pruning. Artificial Intelligence, **6**(4):293–326.

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

KOLLER, D., N. MEGIDDO, and B. VON STENGEL. 1994. Fast algorithms for finding randomized strategies in game trees. *In* Proceedings 26th Annual ACM Symposium on Theory of Computing, pp. 750–759.

KOPEC, D., M. NEWBORN, and M. VALVO. 1992. The 22nd annual ACM international chess championship. Communications of the ACM, **35**(11):100–110.

KORF, R. E. 1991. Multi-player alpha-beta pruning. Artificial Intelligence, **48**(1):99–111.

KUHN, H. W. 1953. Extensive games and the problem of information. *In* Contributions to the theory of games. *Edited by* H. W. Kuhn, and A. W. Tucker. Princeton University Press, Princeton, NJ, vol. II, pp. 193–216.

LEE, K.-F., and S. MAHAJAN. 1990. The development of a world class Othello program. Artificial Intelligence, **43**(1):21–36.

LEVY, D. 1989. The million pound bridge program. *In* Heuristic programming in artificial intelligence. *Edited by* D. N. L. Levy and D. F. Beal. Ellis Horwood Limited, Chichester, England, pp. 95–103.

LUCE, R. D., and H. RAIFFA. 1957. Games and decisions. John Wiley and Sons, New York.

LUCKHARDT, C. A., and K. B. IRANI. 1986. An algorithmic solution of n-person games. *In* Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86). Morgan Kaufmann Publishers, Los Altos, CA, pp. 158–162.

MCALLESTER, D. A. 1988. Conspiracy numbers for min-max search. Artificial Intelligence, **35**(3):287–310.

MUTCHLER, D., and M. VAN LENT. 1995. A pruning algorithm for imperfect information games. Technical Report 95-01, Department of Computer Science, Rose-Hulman Institute of Technology.

NASH, J. F. 1951. Non-cooperative games. Annals of Mathematics, **54**(2):286–295.

QUINLAN, J. R. 1979. A knowledge-based system for locating missing high cards in bridge. *In* Proceedings of the 6th International Joint Conference on Artificial Intelligence (IJCAI-79), pp. 705–707.

RASMUSEN, E. 1989. Games and information—an introduction to game theory. Blackwell Publishers, Oxford.

SCHAEFFER, J. 1990. Conspiracy numbers. Artificial Intelligence, **43**(1):67–84.

SCHAEFFER, J., N. TRELOAR, P. LU, and R. LAKE. 1993. Man versus machine for the world checkers championship. AI Magazine, **14**(2):28–35.

SHANNON, C. E. 1950. Programming a computer for playing chess. Philosophical Magazine, seventh series, **41**(314):256–275. Reprinted in Compendium of computer chess. D. N. L. Levy, Batsford, London, 1988.

SHUBIK, M. 1982. Game theory in the social sciences. MIT Press, Cambridge, MA.

SLAGLE, J. R., and J. K. DIXON. 1969. Experiments with some programs that search game trees. Journal of the ACM, **16**(2):189–207.

SMITH, S. J. J., and D. S. NAU. 1993. Strategic planning for imperfect information games. *In* Games: planning and learning, papers from the 1993 Fall Symposium, AAAI Press Technical Report FS-93-02, Menlo Park, CA, pp. 84–91.

THOMPSON, G. L. 1953. Signaling strategies in **n**-person games. *In* Contributions to the theory of games. *Edited by* H. W. KUHN and A. W. TUCKER. Princeton University Press, Princeton, N. J., vol. II, pp. 267–277.

VAN LENT, M. 1993. A pruning algorithm for one player games with hidden information. Master's thesis, University of Tennessee.

VON NEUMANN, J. 1928. Zur theorie der Gesellschaftespiele. Mathematische Annalen, **100**:295–320.

WILSON, R. 1972. Computing equilibria of two-person games from the extensive form. Management Science, **18**(7):448–460.

ZERMELO, E. 1913. Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. *In* Proceedings of the Fifth International Congress of Mathematicians, Cambridge, vol. 2, pp. 501–504.