Games with Imperfect Information

Jean R. S. Blair, David Mutchler, and Cheng Liu
Department of Computer Science
University of Tennessee
Knoxville, TN 37996–1301
{blair,mutchler,cliu}@cs.utk.edu

Abstract

An *information set* in a game tree is a set of nodes from which the rules of the game require that the same alternative (i.e., move) be selected. Thus the nodes in an information set are indistinguishable to the player moving from that set, thereby reflecting *imperfect information*, that is, information hidden from that player. Information sets arise naturally in (for example) card games like poker and bridge.

Here we focus not on the solution concept for imperfect information games (which has been studied at length), but rather on the computational aspects of such games: how hard is it to compute solutions? We present two fundamental results for imperfect information games. The first result shows that even if there is only a single player, we must seek special cases or heuristics. The second result complements the first, providing an efficient algorithm for just such a special case. Additionally, we show how our special case algorithm can be used as a heuristic in the general case.

1 Introduction

The development and analysis of game-tree search algorithms dates back to the inception of AI and remains an active area of research today. Games with *imperfect information* are an interesting and important class of games.[1] They have been studied at length in the game theory literature. They include many important applications, for example:

- Parlor games like bridge, poker and Clue.
- Economic models of (for example) 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).

There exist many game-tree search algorithms for *perfect information* games. Such algorithms include minimax [17], alpha-beta [8, 19], conspiracy numbers [13, 16], and singular extensions [2]. Ballard [3] provides game-tree search algorithms for perfect information games that include *chance nodes*, at which "nature" determines the move, according to some fixed, known probability distribution. 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).

For games with perfect information, even with chance nodes, a simple modification of minimax computes the value of the game tree [3]:

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

where function F is a *max* operator at nodes where Player 1 moves, *min* at nodes where Player 2 moves, and *average* at chance nodes. (See [9, 10, 12] for the extension to more than two players.) 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* and *min* selections).

In this paper we show that games with imperfect information are quite different from games with perfect

†This research was supported by NSF under grants IRI 89–10728 and CCR–9209803, and by AFOSR under grant 90–0135. Some of the results reported here form part of a paper that has been submitted to the journal *Artificial Intelligence*.

[1] A technical note for the game theorist: this paper discusses games with imperfect information [10], as opposed to games with incomplete information [7]. (Sometimes the former is called decision-making under risk, the latter decision-making under uncertainty.) In imperfect information games, the players lack information only about the game's history. In incomplete information games, the players also lack information about the game's structure. See [14, page 207] or [15, Chapter 2] for a more complete discussion of this distinction.

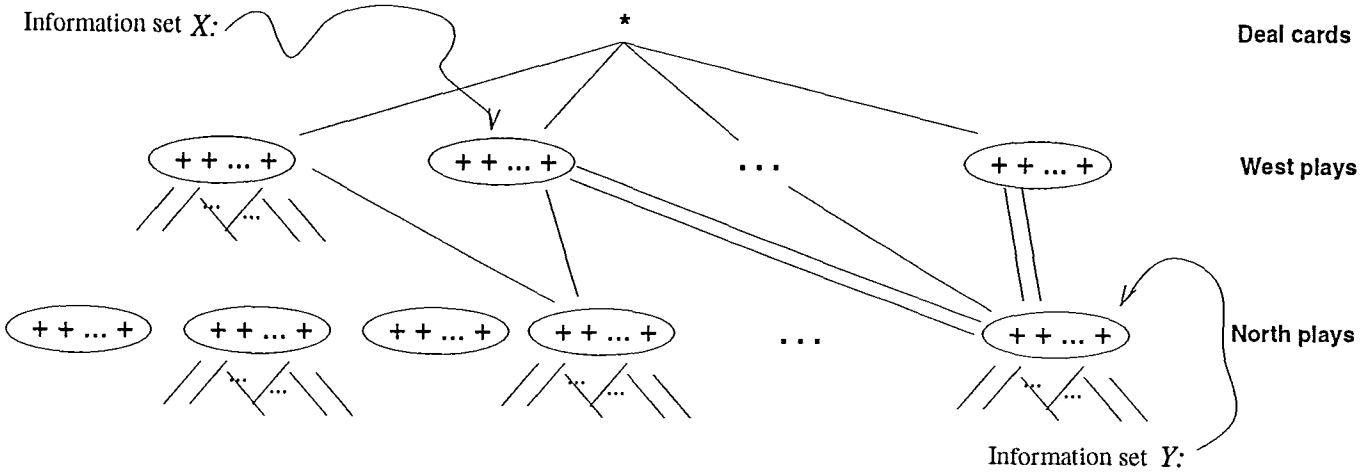

Figure 1: The top portion of the game tree for bridge. In information set X, West holds ♠AQ8743 ♡9 ◇KQ96 ♣32. In information set Y, North holds ..., South holds ..., and West has led ...

information (even with chance nodes). In particular, we answer the following questions:

1. How hard is it to solve imperfect information games, via game-tree search?
2. Is there an analog to minimax for games with imperfect information?

1.1 What is imperfect information?

The card game bridge[2] 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 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 ♠AQ8743 ♡9 ◇KQ96 ♣32. 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 left alternative from each node in X, corresponding to playing ♠A. Section 2 defines imperfect information and other terms more precisely.

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 ♠KJ65 ♡876 ◇J72 ♣AK5, South holds ♠92 ♡QJ54 ◇853 ♣J874, and West has led the ♠A. For each node in Y there are 13 alternatives, corresponding to the 13 cards the North/South team could play from the North (dummy) hand. 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 ◇J. The game tree continues in like fashion.

The following example illustrates the computational difference between perfect information games and imperfect information games. First consider game tree G_1 on the left side of Figure 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. The

[2] We assume the reader is familiar with the game bridge. Our example ignores the so-called "bidding" phase that precedes the card-play.

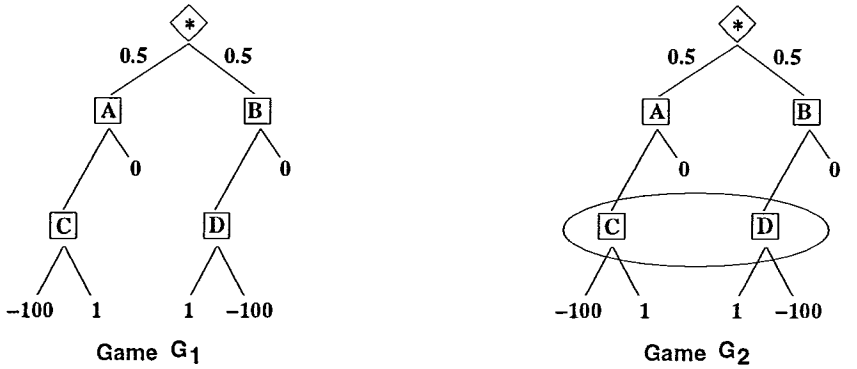

Figure 2: Two game trees, with perfect and imperfect information, respectively.

`chance-minimax` algorithm correctly computes that the value of this game 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 Figure 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, 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 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 only by comparing it to all other strategies on the entire search space.

1.2 Summary of results

The above examples show that imperfect information games are both interesting and different from perfect information games (even with chance nodes). We present the following results in this report:

1. Solving games with imperfect information is NP-hard (in contrast to games with perfect information), *even when there is only a single player.*
2. We give an algorithm called `IMP-minimax` (for "imperfect information minimax") that is the natural analog to `chance-minimax`, but designed to reflect the presence of information sets. `IMP-minimax` computes a strategy for games with imperfect information in time linear in the size of the search tree. The strategy produced by this algorithm is guaranteed to be an optimal strategy, if the imperfect information game has a single player and a certain natural property called perfect recall.

The above results are fundamental to an understanding of game-tree search in imperfect information games. The first result shows that even if there is only a single player, we must seek special cases or heuristics. The second result complements the first, providing an efficient algorithm for just such a special case. Additionally,

3. Section 5 provides extensions to both of the above results. In particular, we show how `IMP-minimax` can be used as a heuristic when the game does not have perfect recall, and when the game has more than one player.

2 Definitions

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, [1]) for definitions of these terms. We follow with minor variation the standard game-theory terminology and notation [11, 15, 18] introduced in [10].

An n-player game Γ consists of:

- A finite tree $\mathcal{K}$ called the game tree. 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.

Throughout we reserve the symbols Γ, $\mathcal{K}$ and h to refer to the game, game tree and payoff function, respectively, under current consideration.

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 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.

We have yet to comment on the interpretation of information sets in the above definition. First we need to define what we mean by a "strategy."[3] A 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).

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

"What a player knows" is reflected in the 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 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.

Chess is a game of perfect information: the state of the game is described by the positions of the pieces and whose turn it is, and this information is available to both players. Backgammon is also a game of perfect information, but includes chance nodes: at certain positions, the next position in the game is selected by rolling the dice. The card game bridge is a game of imperfect information. The first move of the game is to deal the cards at random. Each player knows the contents of the player's own hand, but the contents of the other players' hands are revealed only gradually, as cards are played one by one.

The quality of a strategy is measured by its expected payoff, which, in turn, depends on the probability of reaching leaf nodes. Given 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 non-chance node granted probability 1 or 0 depending on whether or not π selects that arc. The expected payoff under strategy π, denoted $H(\pi)$, is defined to be $\sum p_\pi(w)\ h(w)$, where the sum is over all leaves w in $\mathcal{K}$. For example, the expected payoff for the strategy indicated by thick lines in Figure 4 in Section 4 is $\frac{34}{6} + \frac{44}{6} + \frac{2}{12} + \frac{9}{12} + \frac{75}{2}$.

A player-k strategy π_k is optimal for strategy π if for every player-k strategy α_k, we have $H(\pi) \geq H(\alpha)$, 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 strategy π is an equilibrium point[4] 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. A solution to an n-player game Γ is an equilibrium point for $\mathcal{K}$. Clearly, for one-player games a solution is a strategy that maximizes (over all strategies) the expected payoff to the single player.

3 Solving games with imperfect information is hard

In this section, the task of finding a solution to n-player imperfect information games is shown to be NP-hard for any value of n. It will follow that no efficient algorithm exists for solving imperfect information games in general, unless P=NP. Thus, the results in this sec-

[3] Game theorists will recognize our definition of "strategy" as what they would call a "pure strategy," which we focus on here.

[4] Game theorists will recognize our definition of "equilibrium point" as what they would call a "pure strategy equilibrium point."

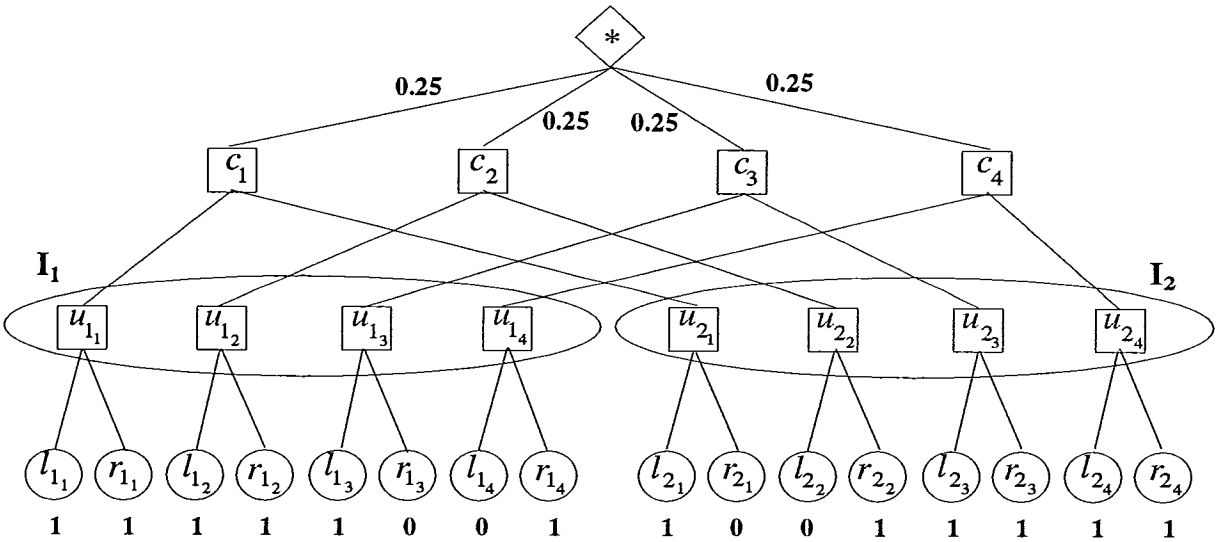

Figure 3: The constructed tree $\mathcal{K}$ for 3SAT instance $C = \{\{u_1, \bar{u_1}, u_2\}, \{u_1, \bar{u_1}, \bar{u_2}\}, \{u_1, u_2, \bar{u_2}\}, \{\bar{u_1}, u_2, \bar{u_2}\}\}$. Payoff values are shown below each leaf node.

tion motivate the linear time algorithm presented in the next section, which solves imperfect information games for a special class of games.

We begin by considering one-player games. The one-player game decision problem cast in the format of [6] is as follows.

ONE-PLAYER GAME (OPG):
Instance: A one-player imperfect information game Γ and a real number K.
Question: Is there a strategy π for Γ with expected payoff $H(\pi) \geq K$?

Theorem 1 *OPG is NP-complete.*

Proof: Here we only sketch the proof; complete details can be found in [4]. We use an example to describe the main ideas of the polynomial time transformation from 3-Satisfiability to OPG. Consider an instance of 3-Satisfiability that contains two variables (u_1 and u_2) and the following clauses: $C_1 = \{u_1, \bar{u_1}, u_2\}$, $C_2 = \{u_1, \bar{u_1}, \bar{u_2}\}$, $C_3 = \{u_1, u_2, \bar{u_2}\}$, $C_4 = \{\bar{u_1}, u_2, \bar{u_2}\}$. The corresponding one-player game tree is shown in Figure 3.

The tree contains one clause node below the root node for each clause. On the level below the clause nodes the tree contains an information set for each variable, each such information set containing a distinct variable node for each clause. Below each variable node there are two leaves, the left leaf representing an uncomplemented literal and the right leaf representing a complemented literal. The function h is given by

$$h(l_{j_i}) = \begin{cases} 1 & \text{if literal } u_j \text{ appears in clause } c_i \\ 0 & \text{otherwise,} \end{cases}$$

$$h(r_{j_i}) = \begin{cases} 1 & \text{if literal } \bar{u_j} \text{ appears in clause } c_i \\ 0 & \text{otherwise.} \end{cases}$$

Given an instance C of 3-Satisfiability, there is a strategy π for the corresponding one-player game with $H(\pi) \geq 1$ if and only if C is satisfiable. In our example, a satisfying truth assignment is one where $t(u_1) = T$ and $t(u_2) = F$. We can derive a strategy π by choosing the left edge out of every node in I_1 (indicating a true value for the variable u_1), the right edge out of every node in I_2 (indicating a false value for the variable u_2), and the set of edges $\{(c_1, u_{1_1}), (c_2, u_{1_2}), (c_3, u_{1_3}), (c_4, u_{2_4})\}$ – each edge "choosing" a true literal in the clause. Notice that for π we have $H(\pi) = p_\pi(l_{1_1}) + p_\pi(l_{1_2}) + p_\pi(l_{1_3}) + p_\pi(r_{2_4}) = 0.25 + 0.25 + 0.25 + 0.25 = 1$.

Conversely, suppose π is a strategy with $H(\pi) \geq 1$. We can then define a truth assignment for the instance of 3-Satisfiability as follows: for each variable u_j, $1 \leq j \leq n$, assign $t(u_j) = T$ if π chooses the left edge for each node in I_j, otherwise assign $t(u_j) = F$. ∎

It is easy to generalize the result in the above theorem to games played by *two or more* players:

GAME TREE SEARCH (GTS):
Instance: An n-player imperfect information game Γ where Γ has one or more equilibrium points.
Question: Find a strategy π that is an equilibrium point in Γ.

Corollary 1 *GTS is NP-hard.*

4 A linear-time algorithm

The results in the previous section show that even if there is only a single player, we must seek heuristics or special cases. Here we take the latter approach. We present an algorithm, `IMP-minimax`, that is to imperfect information games what minimax is to perfect information games. In this section, we describe `IMP-minimax` for one-player games; the next section shows how to extend `IMP-minimax` to games with more than a single player.

4.1 Properties of the algorithm

`IMP-minimax` has two important properties. (Space restrictions prohibit our giving the proofs herein; see [4] for details.)

Theorem 2 *The run-time of* `IMP-minimax` *is linear in the number of nodes in the game tree.*

Theorem 3 *If a game has only a single player and a certain natural property called perfect recall (defined below), then the strategy computed by* `IMP-minimax` *is optimal. Otherwise, the strategy might or might not be optimal.*

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 one-player game Γ has perfect recall[5] if for every pair of 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. Note that perfect recall is not the same thing 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. When later 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.

[5] It is easy to show that this definition of perfect recall, which is given by Thompson in [20], is equivalent to the definition Kuhn gives in [10].

4.2 The algorithm

Before describing `IMP-minimax`, we need to define the functions it uses that are specific to the game tree $\mathcal{K}$ under consideration:

- Function $h(x)$ specifies the payoff at leaf x of $\mathcal{K}$.
- 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. For example, in Figure 4, *extend*($\{$*root of the tree*$\}$) yields the five nodes labeled *Level 1* in the figure; *extend* applied to the Level 1 nodes yields the Level 2 nodes; and so on.
- For any node x in $\mathcal{K}$, its reachable probability $p(x)$ is the product of the probabilities below chance nodes on the path from the root to x. For example, since the probability distributions below chance nodes in Figure 4 are all the uniform distribution, p(leaf whose value is -100) is $1/6$.
- Any subset X of an information set is a partial information set, abbreviated PI-set. A child of PI-set X is the set of all nodes directly below X via a fixed 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.
- Function *partition* takes a set of nodes in $\mathcal{K}$ and partitions it into individual leaves and maximal PI-sets. For the example of Figure 4, letting -100 and 2 denote the leaves whose values are -100 and 2 respectively, we have that *partition* ($\{$ E, -100, 2, F, H, I $\}$) returns $\{$ $\{E\}$, -100, 2, $\{F\}$, $\{H, I\}$ $\}$.

Given a set X of nodes in $\mathcal{K}$, the recursive function V is defined by:

$$V(X) = \begin{cases} \max \{\, V(\mathit{extend}\,(Y)) \mid Y \text{ is a child of } X \,\} \\ \qquad \text{if } X \text{ is a PI-set} \\ \displaystyle\sum_{\substack{x \in \mathit{partition}\,(X) \\ x \text{ a leaf}}} p(x)\, h(x) \;+\; \sum_{\substack{x \in \mathit{partition}\,(X) \\ x \text{ a PI-set}}} V(x) \\ \qquad \text{otherwise} \end{cases}$$

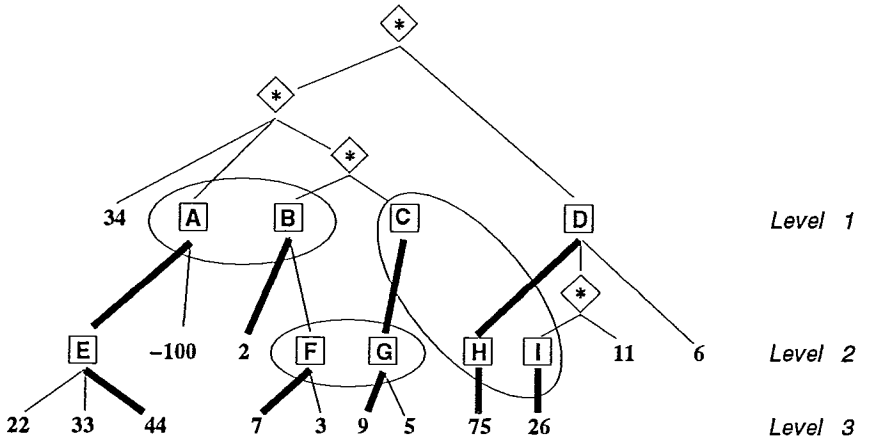

Figure 4: A one-player game tree. For any chance node (labeled *), the arcs directly below it are equally likely; player nodes are labeled with letters; and leaves are labeled with their respective payoff values. Ellipses are used to show information sets that contain more than one node. The thick lines indicate the strategy selected by `IMP-minimax`.

IMP-minimax: call $V(\,extend\,(\{root\ of\ \mathcal{K}\}))$. 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 $V(X)$.

Thus `IMP-minimax` both returns a number (per the V function) and computes a strategy π^* for $\mathcal{K}$.

Note that `IMP-minimax` processes the game tree in a post-order (depth-first) traversal fashion, where at each major step a *partial information set* rather than a single node is processed. Intuitively, the algorithm works as follows. If X is a partial information set then any strategy can pick only one child of X. The algorithm examines each child, recursively. Then the algorithm greedily picks the child of X that maximizes the expected payoff in the forest rooted at X. This recursive search followed by a greedy choice yields the post-order traversal. If, on the other hand, X is not a single partial information set then the algorithm considers each subtree ("rooted" at a partial information set or a leaf node) separately and sums the expected payoff of each subtree to reflect the fact that the expected payoff of the entire tree will have a component from each of the subtrees that have non-zero probability. The reader may find it instructive to trace the operation of `IMP-minimax` on the example in Figure 4, where $V(\,extend\,(\{root\ of\ \mathcal{K}\})) = 617/12$.

It should be clear that `IMP-minimax` assigns an alternative to each information set in the tree and, thus, computes a strategy for $\mathcal{K}$. However, in general the computed strategy is not an optimal strategy (first example below). By Theorem 3, if the game has perfect recall, then π^* is an optimal strategy (second example below).

4.3 Examples using the algorithm

Example (not perfect recall): Consider game tree G_2 in Figure 2. To compute π^*, one computes $V(\,extend\,(\{root\ of\ G_2\})) = V(A, B)$.[6] This causes the computation of

$V(A) = \max\{\, V(C),\ 0\}$,
$V(B) = \max\{\, V(D),\ 0\}$, and
$V(C) = V(D) = 1$,

so that $V(\,extend\,(\{root\ of\ G_2\})) = 1$.

Here the information set containing C and D was 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 π^* is defined by:

$\pi^*(A) = C$,
$\pi^*(B) = D$, and
$\pi^*(C)$, $\pi^*(D)$ = the left child of C, D.

As such, strategy π^* is not optimal. It has expected payoff $-99/2$, while an optimal strategy (same as π^* except select the right child at A) 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. Theorem 3 shows that such an event is impossible when there is perfect recall.

Example (perfect recall): Again consider the game tree G_2 in Figure 2, but with A and B in a single

[6] For ease in reading the examples we have omitted the set brackets around the input to the function V throughout.

information set. (Thus, the game has imperfect information but perfect recall.) Here the computation of $V(\textit{extend}\ (\{\textit{root of modified}\ G_2\})) = V(A,B)$ causes the computation of

$$V(A,B) = \max\{\ V(C,D),\ \ 0.5 \times 0\ +\ 0.5 \times 0\}$$
$$V(C,D) = \max\{\ 0.5 \times (-100)\ +\ 0.5 \times 1,\ \ 0.5 \times 1\ +\ 0.5 \times (-100)\}$$

so that $V(\textit{extend}\ (\{\textit{root of modified}\ G_2\})) = 0$ and π^* is the optimal strategy (go right at $\{A,B\}$).

Another example (perfect recall): In [4] we describe a class of games that illustrates the magnitude of the improvement `IMP-minimax` can achieve over the naive, examine-all-strategies algorithm. 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 for solving one-player games 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). The following table contrasts these quantities, for several game tree depths.

Depth d	nodes in game tree	strategies
2	4	2
3	15	32
4	61	2,097,152
5	249	3.9×10^{25}
6	1009	4.5×10^{102}
7	4065	8.1×10^{410}
8	16321	8.4×10^{1643}

5 Extensions and open questions

The results of the previous sections provide a foundation for an understanding of game-tree search in imperfect information games. In this section we discuss several extensions to those results.

As mentioned earlier, the NP-completeness results in Section 3 motivate us to consider special cases for which one can efficiently find optimal strategies for n-player games. The linear time `IMP-minimax` algorithm (as well as its pruning variant `IMP-alpha-beta` [21], described elsewhere in this proceedings) is a solution to such a special case: the class of games with perfect recall and a single player. This suggests the following important question: is there an efficient algorithm for solving games with perfect recall and two or more players, when a solution exists?[7]

[7] In some two-player games a solution (pure strategy equilibrium point) does not exist. Even if the two-player game is zero-sum (i.e., payoffs to the players sum to zero), mixed strategies may be required for equilibria.

A natural extension to `IMP-minimax` for two-player zero-sum games with imperfect information is to use the following V operator.

$$V(X) = \begin{cases} \max\{\ V(\textit{extend}\ (Y))\ \mid\ Y \text{ is a child of } X\ \} \\ \qquad\qquad \text{if } X \text{ is a player } A \text{ PI-set} \\ \min\{\ V(\textit{extend}\ (Y))\ \mid\ Y \text{ is a child of } X\ \} \\ \qquad\qquad \text{if } X \text{ is a player } B \text{ PI-set} \\ \sum_{\substack{x \in \textit{partition}\ (X) \\ x \text{ a leaf}}} p(x)\, h(x) \ +\ \sum_{\substack{x \in \textit{partition}\ (X) \\ x \text{ a PI-set}}} V(x) \\ \qquad\qquad \text{otherwise} \end{cases}$$

`IMP-minimax` remains efficient (linear in the size of the game tree) in this two-player form. Since `IMP-minimax` finds a solution for one-player games with perfect recall (Theorem 3), one might expect it to do the same for two-player games with perfect recall, when a solution exists. However, not only does `IMP-minimax` not accomplish this goal, neither can any efficient algorithm (unless $P = NP$), as the following result shows.[8]

Theorem 4 *For any $n > 1$, it is NP-complete to determine whether or not an n-player imperfect information game with perfect recall has a solution (i.e. has a pure strategy equilibrium point).*

Theorem 4 forces us to turn to special cases or heuristics for more-than-one-player games, even for games with perfect recall. `IMP-minimax` (in the above two-player form) is one such heuristic.

`IMP-minimax` can also be used in its one-player form as a heuristic for two-player games in an iterative technique like the one proposed in [5] as follows. Start with a reasonable guess α_1 for player A's strategy. Fixing that strategy, use `IMP-minimax` to find a strategy β_1 for player B. Next fix player B's strategy at β_1 and use `IMP-minimax` to find a strategy α_2 for player A. Repeat this process some number of times, generating a sequence of strategy pairs: $(\alpha_1,\beta_1),(\alpha_2,\beta_2),\cdots$. If after i iterations we have $(\alpha_{i-1},\beta_{i-1}) = (\alpha_i,\beta_i)$, and if the game has perfect recall, then the properties of `IMP-minimax` guarantee that (α_i,β_i) is an equilibrium point. This iterative technique suggests several important questions. First, how do we come up with a reasonable first guess for player A? Second, for what classes of games with equilibrium points is the technique guaranteed to converge? Third, how many iterations and

[8] For brevity we omit the proof. The reduction is from 3-partition.

what conditions are necessary before one can assume the strategy is reasonably close to an equilibrium point?

In summary, our results show that: first, a heuristic approach will be necessary to solve practical applications of imperfect information games with more than one player; second, `IMP-minimax` is a natural starting point for that heuristic approach, because of its theoretical properties in the one-player realm.

Acknowledgments

We appreciate the useful comments we received from Mike van Lent and the anonymous reviewers.

References

[1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[2] Thomas Anantharaman, Murray S. Campbell, and Feng-hsiung Hsu. Singular extensions: adding selectivity to brute-force searching. *Artificial Intelligence*, 43(1):99–109, April 1990.

[3] Bruce W. Ballard. The $*$-minimax search procedure for trees containing chance nodes. *Artificial Intelligence*, 21(1,2):327–350, March 1983.

[4] Jean R. S. Blair, David Mutchler, and Cheng Liu. Heuristic search in one-player games with hidden information. Technical Report CS-92-162, Dept. of Computer Science, The University of Tennessee, July 1992.

[5] G. W. Brown. Iterative solutions of games by fictitious play. In T. C. Koopmans, editor, *Activity analysis of production and allocation*, Cowles Commission Monograph 13, pages 374–376. John Wiley & Sons, New York, 1951.

[6] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[7] John C. Harsanyi. Games with incomplete information played by "Bayesian" players, I–III. *Management Science*, 14(3):159–182, November 1967. Parts II and III appear in the same journal, in 14(5):320–334, January 1968, and 14(7):486–502, March 1968, respectively.

[8] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, Winter 1975.

[9] Richard E. Korf. Multi-player alpha-beta pruning. *Artificial Intelligence*, 48(1):99–111, February 1991.

[10] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, pages 193–216. Princeton University Press, Princeton, 1953.

[11] R. Duncan Luce and Howard Raiffa. *Games and Decisions.* John Wiley & Sons, New York, 1957.

[12] Carol A. Luckhardt and Keki B. Irani. An algorithmic solution of n-person games. In *Proc. of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 158–162, Los Altos, CA, 1986. Morgan Kaufmann Publishers.

[13] David Allen McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35(3):287–310, July 1988.

[14] Michael Mesterton-Gibbons. *An Introduction to Game-Theoretic Modelling.* Addison-Wesley Publishing Company, Redwood City, California, 1992.

[15] Eric Rasmusen. *Games and Information — An Introduction to Game Theory.* Blackwell Publishers, Oxford, 1989.

[16] Jonathan Schaeffer. Conspiracy numbers. *Artificial Intelligence*, 43(1):67–84, April 1990.

[17] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, Seventh Series, 41(314):256–275, March 1950. Reprinted in *Compendium of Computer Chess*, by D. N. L. Levy, Batsford, London, 1988.

[18] Martin Shubik. *Game Theory in the Social Sciences.* The MIT Press, Cambridge, MA, 1982.

[19] James R. Slagle and John K. Dixon. Experiments with some programs that search game trees. *Journal of the ACM*, 16(2):189–207, April 1969.

[20] G. L. Thompson. Signaling strategies in n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Vol. II*, pages 267–277. Princeton University Press, Princeton, 1953.

[21] Michael van Lent and David Mutchler. A pruning algorithm for imperfect information games. In *Proc. of the AAAI Fall Symposium on Games: Planning and Learning*, Raleigh, 1993.