A Pruning Algorithm for Imperfect Information Games

Michael van Lent and David Mutchler
Department of Computer Science
University of Tennessee
Knoxville, TN 37996–1301
{vanlent,mutchler}@cs.utk.edu

Abstract

IMP-minimax is the analog to minimax for games with *imperfect information*, like card games such as bridge or poker. It computes an optimal strategy for the game if the game has a single player and a certain natural property called perfect recall. IMP-minimax is described fully in a companion paper in this proceedings. Here we introduce an algorithm IMP-alpha-beta that is to IMP-minimax as alpha-beta is to minimax. That is, IMP-alpha-beta computes the same value as IMP-minimax does, but usually faster through pruning (i.e., not examining the value of some leaves). IMP-alpha-beta includes common pruning techniques and introduces a new technique, information set pruning. We suggest a natural model in which to study the performance of search algorithms for imperfect information games and we analyze IMP-alpha-beta in the context of that model. Our analysis includes both theorems bounding the performance of IMP-alpha-beta and empirical data indicating its average-case behavior.

1 Introduction

Games with *imperfect information* are important and interesting. Two fundamental results for such games are [2, 3]:

- Solving games with imperfect information is NP-hard (in contrast to games with perfect information), even when there is only a single player.
- An algorithm called IMP-minimax (for "imperfect information 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 game has a single player and a certain natural property called perfect recall.

IMP-minimax is to imperfect information games as minimax is to perfect information games. Here we introduce and analyze 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 examing the value of some leaves).

Imperfect information games are interesting because large classes of common games, such as card games like bridge and poker, include the imperfect information property. We refer the reader to [2, 3] (the latter in this proceedings) for further motivation for studying imperfect information games. IMP-minimax is important to such games in several respects: as a first step toward understanding heuristic search in such games; as a heuristic when there is more than one player; and as a solution method for one-player applications (like blackjack and solitaire). Thus pruning algorithms for IMP-minimax, like the IMP-alpha-beta algorithm presented herein, are important too.

We refer the reader to [2, 3] for precise definitions and several examples of imperfect information games and IMP-minimax. Our treatment here is necessarily brief. Both IMP-minimax and IMP-alpha-beta can be stated in two-player versions; however, in such games they may not return an optimal strategy. Therefore, we restrict our discussion to their one-player versions.

A one-player game can be represented by a game tree with all player nodes being MAX nodes. The game may have *chance nodes*, at which some random event selects the move according to a fixed, known probability distribution. The game may also have *information sets*, which reflect the imperfect information in the game. An information set is a collection of nodes which are differentiated only by the information hidden from the

†This research was supported by NSF under grant IRI 89–10728, by AFOSR under grant 90–0135, and by the Naval Research Laboratory. The current address for Michael van Lent is the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

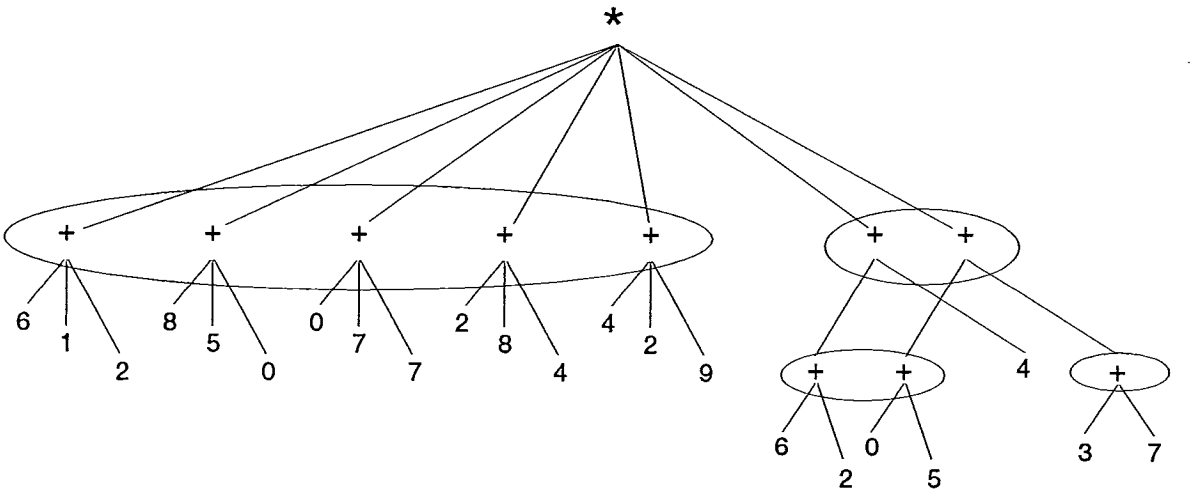

Figure 1: A game with imperfect information.

player. Because the nodes in an information set cannot be differentiated *the player is required (by the rules of the game) to select the same alternative for every node in the information set.* For example, the game in Figure 1 has a chance node at the root and four information sets, denoted by the four ellipses. The leftmost information set contains five nodes, with three alternatives from each. The player must either select the left child from each of these five nodes, or the middle child from each, or the right child from each; the player may not (for example) select the left child from some of the five nodes and the right child from others. This reflects the imperfect information: the player does not completely "know" the outcome of the chance node at the root.[1]

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.

In all our examples, we notate chance nodes by asterisks * and draw ellipses around the nodes in information sets. Further, at each chance node in our examples, the probability distribution associated with that node is the uniform distribution (i.e., each alternative is equally likely).

A *strategy* is a prescription for what alternatives to select at the player nodes. The quality of a strategy is measured by its expected payoff, which, in turn, depends on the probability of reaching leaf nodes. Given a strategy π on a game tree, 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 the game tree and $h(w)$ denotes the payoff at leaf w. For example, in Figure 1, an optimal strategy (i.e., one which maximizes the expected payoff) is to select the middle alternative from the leftmost information set and the rightmost alternative from the three other information sets; this yields an expected score of $\frac{1+5+7+8+2}{7} + \frac{4}{7} + \frac{7}{7}$.

The *value* of a one-player game with imperfect information is the value returned by `IMP-minimax`, as described below. If the imperfect information game has a certain property called perfect recall, then it can be shown that its value (as computed by `IMP-minimax`) equals the expected value of the optimal strategy [2, 3]. Informally, perfect recall means the player recalls her previous moves; the technical definition can be found in our companion paper in this proceedings [3]. Note that perfect recall is not the same thing as perfect information.

[1] This simple example is not adequate for motivating the use of information sets. See [2, 3] (the latter in this proceedings) for more elaborate examples and explanation.

IMP-minimax: call

$$V\,(\,expand\,(\,\{root\ of\ the\ game\ tree\}\,)\,)$$

where the recursive function $V(X)$ takes a set X of nodes in the game tree and is given by:

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

and where function *expand* takes a set of nodes and recursively replaces each chance node by its children; a *partial information set*, abbreviated *PI-set*, is a set of nodes all of which belong to a single information set; the j^{th} *child* of a partial information set I is the set of all immediate descendants of nodes in I reached via the j^{th} alternative; $p(x)$ is the product of the probabilities below chance nodes on the path from the root of the game to node x; $h(x)$ is the payoff at leaf x; and *partition* (X) separates the leaves from the non-leaves in X and partitions the non-leaf nodes into their respective information sets. See [2, 3] (the latter in this proceedings) for a more complete exposition of `IMP-minimax`, including examples of its use.

2 Information set 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 1 Let $\mathcal{A}$ be any algorithm that correctly solves this problem: given a 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 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 multi-player pruning [5, 6] and chance-node pruning [1]. 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 examples.

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

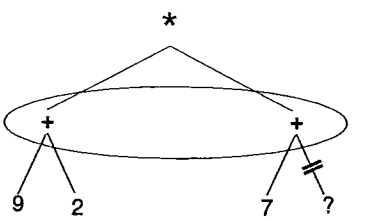

Figure 2: Information set pruning, a simple example.

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 Figure 3, 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 T via its left alternative is

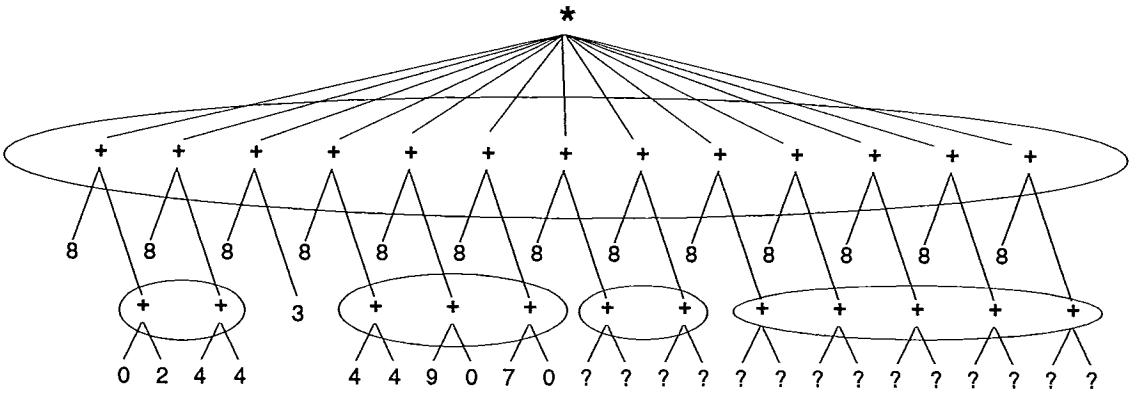

Figure 3: Information set pruning, a more elaborate example.

$\frac{8+\cdots 8}{13} = 8$. 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 $\frac{6}{13} + \frac{3}{13} + \frac{20}{13} + 2\frac{10}{13} + 5\frac{10}{13}$, which equals 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.

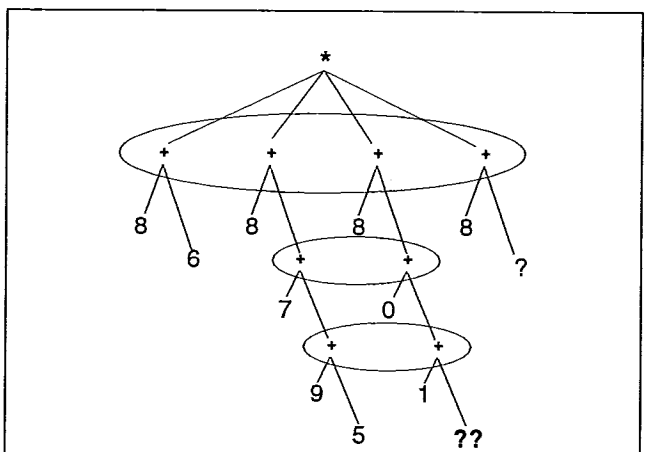

Figure 4: Information set pruning, deep pruning.

Example 3 In the game tree in Figure 4 (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 [4, 8], 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 [1], although the two are by definition not the same, since 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 the box at the top of the next page. Domain-specific functions are italicized and explained below. Scoping is indicted 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 more than one information set.

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

```
IMP-alpha-beta: call Mixed-Set (expand ({root}), −∞).

Max-Set (x, P)
        best = P
        for each alternative A from x
                temp = Mixed-Set (expand (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)
```

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 drawn from `IMP-minimax` and defined as follows:

- *prob* takes a node or set of nodes. On a node x, it returns the product of the probabilities below chance nodes on the path from the root to node x. On a set of nodes, it returns *prob* (x) summed over all nodes x in the set.
- *expand* takes a set of nodes and recursively replaces each chance node by its children, until no chance nodes remain.
- *partition* takes a set of nodes in the game tree and, after separating the leaves from the non-leaves, partitions the non-leaves into their respective information sets. It returns the set that results; each member of the set is either a leaf or a subset of an information set. For example, if the thirteen depth 3 nodes in Figure 3 are labeled $a, b \ldots m$ (left to right), then *partition* applied to the set of those thirteen nodes yields

$$\{\ \{a,b\},\ c,\ \{d,e,f\},\ \{g,h\},\ \{i,j,k,l,m\}\ \}$$

Theorem 2 (`IMP-alpha-beta` is correct)
For any real number P and set X of nodes in a one-player game with imperfect information,

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

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

Proof: by induction on the number of recursive calls to V. See [7, 9] for details. ∎

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

3 The IMP-model

Any analysis of the effectiveness of `IMP-alpha-beta` requires a general model of one-player games with imperfect information. In particular, one must extend current models to include information sets. The IMP-

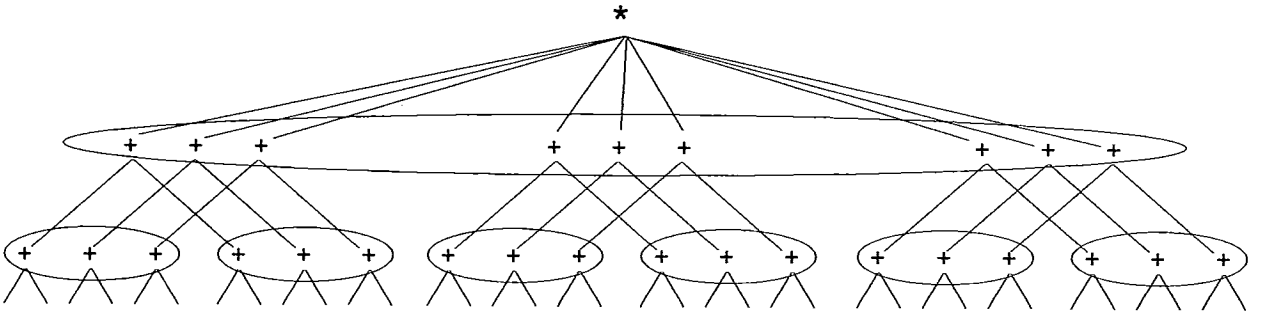

Figure 5: The IMP-model with $k = 3$, $b = 2$ and $d = 3$.

model proposed here is motivated by real games, flexible enough to generate a wide variety of trees, yet simple to generate and analyze.

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 whatever method is desired. Figure 5 shows an example of the IMP-model.

The novelty of the IMP-model is in its method for specifying the information sets, via the parameter k. Each information set fragments into k information sets, along each alternative. Thus k specifies how much "information" the player gathers at each step of the game. For example, when k is 2, each information set is half as large as its parent information set, reflecting information gained at that move; the player also gains information by remembering which alternative she selected. The $k = 1$ tree has perfect information.

The IMP-model is motivated by two very basic properties of card games: shuffling the deck and discovering additional information at each move. The chance node at the root of the IMP-model corresponds to dealing from a shuffled deck of cards, a first step in many card games. The fragmentation of each information set into k information sets along each alternative mimics the gradual process of revealing hidden information, until the end of the game when everything is revealed. For example, in bridge and many other card games a single card is revealed at each move. By observing which cards are revealed a good player not only learns what cards were held but also guesses what cards the opponent may or may not still have. (Although bridge is a two-team game, `IMP-alpha-beta` may still be useful for it, as a heuristic or subfunction [2, 3].) Another example is card counting, that is, calculating the probability that a certain card will be on top of the deck based on knowledge of which cards have already been played.

4 Effectiveness of `IMP-alpha-beta`

How much faster is `IMP-alpha-beta` than `IMP-minimax`? This section addresses that question, in the context of the IMP-model.

Following standard practice, we measure the "speed" of `IMP-minimax` and `IMP-alpha-beta` by the number of leaves examined by each. This is reasonable, since `IMP-minimax` is essentially a tree traversal, evaluating leaves when encountered, and `IMP-alpha-beta` can be implemented on top of `IMP-minimax` with little overhead.

For a given IMP-model game tree, the number of leaves examined by `IMP-alpha-beta` depends on several factors: the payoff values; the placement of these values; and the orderings of the loops in `Max-Set` and `Mixed-Set`. (That is, in what order should the b alternatives in `Max-Set` be searched, and in what order should the k leaves or partial information sets in `Mixed-Set` be searched?) This section presents both theorems and experiments to explore the space of possible values for these factors. For brevity we omit the proofs of the theorems; they can be found in [7, 9].

Theorem 3 `IMP-minimax` examines all $(kb)^{d-1}$ leaves in the IMP-model tree.

4.1 Boundaries

The following theorem provides both upper and lower bounds on the amount of pruning available in the IMP-model.

Theorem 4 For any setting of the parameters (k, b, and d) of the IMP-model with $k > 1$, and for any order-

ing of the loops in `IMP-alpha-beta`, there exist payoff values and placements such that the number of leaves examined by `IMP-alpha-beta` can be:[2]

- as many as $(kb)^{d-1}$ (that is, no pruning).
- as few as k^{d-1}. (Further, this lower bound is tight — at least this many nodes must be examined.)
- 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 [4]. 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.

4.2 Average case analysis

Theorem 4 gives upper and lower bounds on how much pruning can be obtained by using `IMP-alpha-beta`. This section provides a first attempt at determining what pruning one might expect in practice. To that end, we generated IMP-model trees with random leaf payoffs and measured the average number of leaves examined by `IMP-alpha-beta`. Using random leaf payoffs has flaws, but provides a reasonable first approximation for the average case behavior of `IMP-alpha-beta`.

Our code is written in Austin Kyoto Common Lisp.[3] The experiments ran on a collection of Sun-4's. At each leaf encountered, we obtain its payoff from the built-in function call `(random 11)`, which yields an integer between 0 and 10, inclusive. For each trial we ran, we recorded the initial state of the random number generator, so that our results can be reproduced. Our code is available upon request, as is the record of random payoffs.

We considered all combinations of k and b from 2 to 10, as well as a few additional combinations ($k = 2$ and b large). For each of these cases, we varied the depth d from 2 to 5; additionally, we ran larger depths for the smaller values of k and b. In all, we considered 427 combinations of k, b and d. For each such combination, we ran multiple trials and averaged the results of the trials. The number of trials varied from 3 to 5000 and was chosen to insure that for every combination, a 95% confidence interval for $\alpha\beta/mm$ was less than 1% (usually much less), where $\alpha\beta$ and mm are the number of leaves examined by `IMP-alpha-beta` and `IMP-minimax` respectively. The total run time used for the experiments was well over 200 hours.

The three main conclusions from the experiments are as follows, where $\alpha\beta$ and mm are as just stated. First, $\alpha\beta/mm$ decreases as the depth d of the tree increases, but its limit as $d \to \infty$ appears to be strictly positive in all cases. 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.

Figure 6 illustrates these conclusions, for $k = 2$ (left graph) and $b = 2$ (right graph). In both graphs the x-axis is the depth d and the y-axis is $100\,\alpha\beta/mm$, where $\alpha\beta$ and mm are as stated above. (Note the different scales for the y-axes.) All the curves show that $\alpha\beta/mm$ is a decreasing function of d. Less than 20% of the total nodes are examined in the more favorable cases (bottom curve of the left graph, where $b = 10$ and $k = 2$). About 96% of the total nodes are examined in the least favorable case (top curve of the right graph, where $b = 2$ and $k = 10$).

In sum, the fraction examined by `IMP-alpha-beta` of the total number of leaves is minimized when k is small and b is large; this fraction decreases (but not to zero) as the depth increases. Substantial pruning can be achieved even for modest values of b and d, if k is small. The overhead in a careful implementation of `IMP-alpha-beta` is small enough that `IMP-alpha-beta` runs faster than `IMP-minimax` when almost any pruning is achieved, for static evaluators one would encounter in practice.

4.3 Ordering

One would expect that at `Max-Set`, searching alternatives from best to worst is optimal, because high returned values create high pruning values, which in turn makes subsequent pruning more likely. Likewise, one

[2] For brevity we omit the $k = 1$ special case; it is given in [7, 9].

[3] Austin Kyoto Common Lisp is public domain software written by William Schelter at the University of Texas, and is based on Kyoto Common Lisp, written by Taiichi Yuasa and Masami Hagiya, Kyota University, 1984.

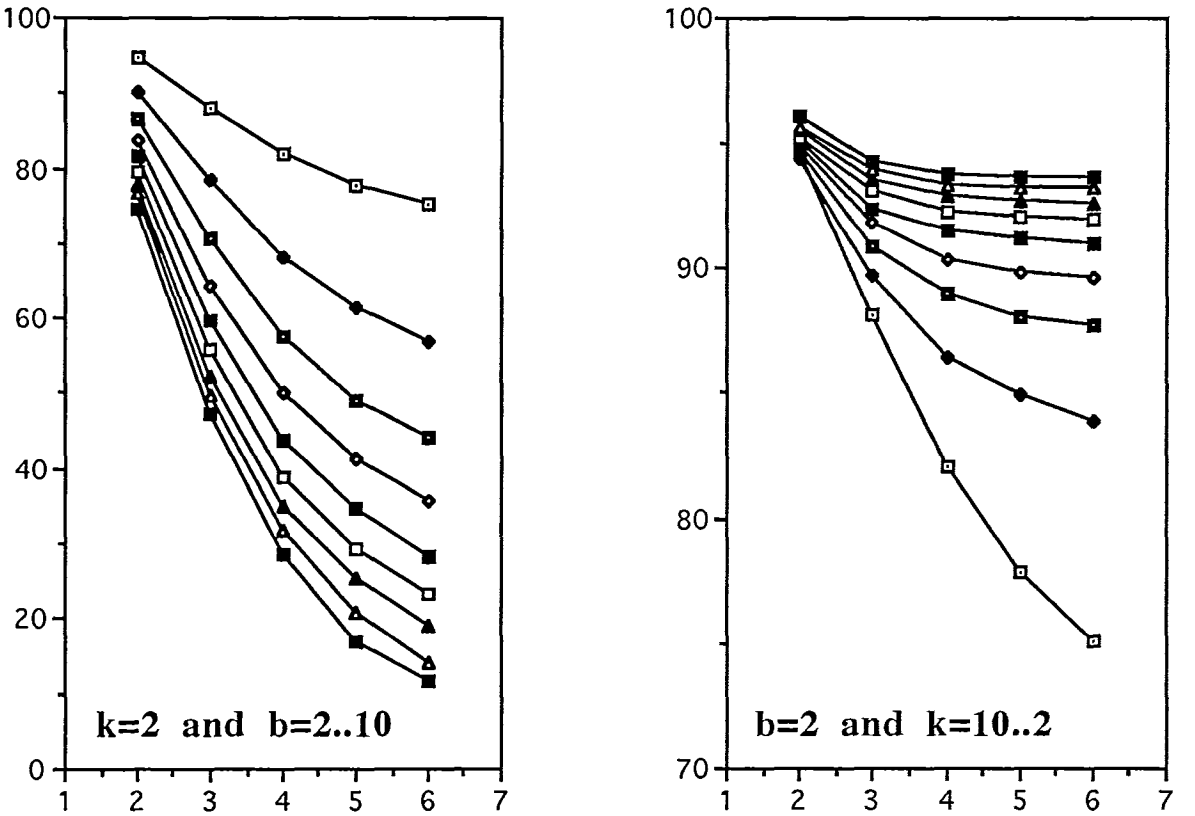

Figure 6: $100\,\alpha\beta/mm$ versus depth d, where $\alpha\beta$ and mm are the number of leaves examined by `IMP-alpha-beta` and `IMP-minimax` respectively. The curves in the left graph are all for $k = 2$, with b varying from 2 (top curve) to 10 (bottom curve). The curves in the right graph are all for $b = 2$, with k varying from 10 (top curve) down to 2 (bottom curve). All data points are averages over enough trials so that the width of its 95% confidence interval is less than 1%.

would expect that at `Mixed-Set`, searching the leaves or information sets from worst to best is optimal, since low returned values in `Mixed-Set` increase the likelihood that the pruning condition succeeds. In fact, the first of these expectations is true; the second is false in general, as shown by the following theorems.

Theorem 5 For any setting of the parameters (k, b, and d) of the IMP-model and for any payoff values and placement thereof, the fewest leaves are examined by `IMP-alpha-beta` if at `Max-Set`, the alternatives are searched from best to worst. (In general, ties cannot be broken arbitrarily.) By "best alternative," we mean the alternative for which the returned value is maximal.

Theorem 6 No ordering of the leaves or information sets in `Mixed-Set` that depends only on the returned values minimizes the number of leaves examined by `IMP-alpha-beta` for all settings of the parameters of the IMP-model and all payoff values and placement of these values.

The above contrasts with the analysis of alpha-beta, where for every set of payoffs and placements thereof, the best-first ordering yields as much pruning as is possible, in uniform trees [4].

5 Summary and open questions

We have introduced and analyzed `IMP-alpha-beta`, a pruning algorithm for `IMP-minimax`. Both algorithms find optimal strategies in imperfect information games that have a single player and perfect recall, and can be used as heuristics in more general games. Future work includes extensions of the `IMP-alpha-beta` algorithm and its analysis:

- Can more pruning be obtained through non-depth-first strategies than from `IMP-alpha-beta`? When information sets are present, not only is no fixed ordering optimal at `Mixed-Set` (Theorem 6), but the entire depth-first strategy may need to be abandoned to minimize the number of leaves examined. The same is true when only chance nodes are present, and motivates the "probing" in Ballard's pruning algorithms for perfect information games with chance nodes [1]. Ballard's technique can be used for information set pruning as well; however, the absence of MIN nodes in our one-player games appears to limit its usefulness.
- More powerful pruning may be obtained if the upper bound on payoffs at node x is required only to bound payoffs at leaves below x, instead of payoffs at all leaves. Such upper bounds are readily available in, for example, the card game bridge.
- An interesting heuristic is to prune if the required sum is "close enough" to the pruning bound. We would expect this to magnify pruning significantly in applications in which a few information sets have large probability and the rest have small probability.
- The best test for the pruning effectiveness of `IMP-alpha-beta` is its performance in a real application, possibly as a one-player heuristic for a two-player game, as described in [2, 3].

Acknowledgments

We appreciate the useful comments we received from Jean Blair and the anonymous reviewers.

References

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

[2] 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, University of Tennessee, July 1992.

[3] Jean R. S. Blair, David Mutchler, and Cheng Liu. Games with imperfect information. In *Proc. of the AAAI Fall Symposium on Games: Planning and Learning*, Raleigh, 1993.

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

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

[6] 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.

[7] David Mutchler and Michael van Lent. A pruning algorithm for imperfect information games. Technical report, Dept. of Computer Science, University of Tennessee, September 1993.

[8] 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.

[9] Michael van Lent. A pruning algorithm for one player games with hidden information. Master's thesis, University of Tennessee, May 1993.