Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search

Jeffrey Long and **Nathan R. Sturtevant** and **Michael Buro** and **Timothy Furtak**
Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada T6G 2E8
{jlong1|nathanst|mburo|furtak}@cs.ualberta.ca

Abstract

Perfect Information Monte Carlo (PIMC) search is a practical technique for playing imperfect information games that are too large to be optimally solved. Although PIMC search has been criticized in the past for its theoretical deficiencies, in practice it has often produced strong results in a variety of domains. In this paper, we set out to resolve this discrepancy. The contributions of the paper are twofold. First, we use synthetic game trees to identify game properties that result in strong or weak performance for PIMC search as compared to an optimal player. Second, we show how these properties can be detected in real games, and demonstrate that they do indeed appear to be good predictors of the strength of PIMC search. Thus, using the tools established in this paper, it should be possible to decide a priori whether PIMC search will be an effective approach to new and unexplored games.

Introduction

Imperfect information is a common element of the world that all humans or agents must deal with in one way or another. The ideal solution, at least for two-player zero-sum games, is to use a solution technique that can produce a Nash equilibrium, guaranteeing perfect play against perfect opponents. This is, however, computationally infeasible in all but the most simple of games.

One popular way of dealing with imperfect information has been to avoid the issue. Instead of solving a full game, perfect information worlds from the game are sampled and solved either exactly or heuristically. This approach, also called Perfect Information Monte Carlo (PIMC), has produced expert-caliber players in games like Bridge (Ginsberg 2001) and Skat (Buro et al. 2009), and has produced strong play in games like Hearts (Sturtevant 2008). Yet this work has often been criticized for avoiding the issue of imperfect information. For instance, in the 2nd edition of their AI textbook, Russell and Norvig (Russell and Norvig 2002) write that PIMC search (which they call "averaging over clairvoyancy") suggests a course of action that "no sane person would follow" in a simple example that they present.

While these criticisms are technically correct, they do not explain the true mystery of why PIMC has been successful. If PIMC was fundamentally the wrong approach, one would expect that no program of reasonable quality could use it to play at any level approaching human strength. This paper represents the first attempt to answer why PIMC has been successful in practice.

We hypothesize that there should be a small number of concise properties which can describe the general properties of a game tree and measure, from these properties, whether PIMC as an approach is likely to be successful or not. Several properties are proposed and used to build small synthetic trees which can be solved and also played with PIMC. We show that these properties directly influence the performance of PIMC, and that they can be measured in real games as well. Thus, we are able to show that in many classes of games PIMC will not suffer large losses in comparison to a game-theoretic solution.

Background and Related Work

The first big success for PIMC search emerged from the work of Ginsberg (Ginsberg 2001), creator of the GIB computer player for contract bridge. Prior to 1994, Ginsberg describes computer players in bridge as "hopelessly weak". Ginsberg's program GIB, introduced in 1998, was the first full-featured computer bridge player to make use of PIMC search, and by 2001, GIB was claimed to be the strongest computer bridge player in the world and of roughly equivalent playing strength to human experts. Most of the strength came from PIMC search, however a number of other techniques were used to correct for errors introduced by PIMC.

Ginsberg improves GIB's card play further by introducing the concept of alpha-beta search over lattices. This allows the program to search over *sets* of card configurations in which the declarer makes the contract, instead of over the numeric interval normally used in evaluation functions. With this enhancement, GIB is able to capture the imperfect information of bridge to a greater extent; however, Ginsberg only uses this technique for GIB's declarer card play, and it still assumes that the defenders have perfect information of the game. Furthermore, this enhancement only improves GIB's performance by 0.1 IMPs per deal (the standard performance measure for bridge), which Ginsberg states is only significant because GIB's declarer card play was already its strongest component and on par with human experts.

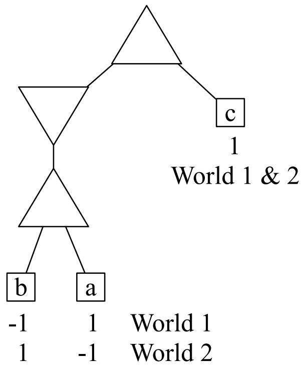
In 1998, Frank and Basin published an extensive critique of the PIMC approach to imperfect information games (Frank and Basin 1998). They showed that the nature of PIMC search makes it prone to two distinct types of errors, irrespective of the number of hypothetical worlds examined. The first of these errors is termed *strategy fusion*. Strategy fusion arises because PIMC search (incorrectly) believes it can use a different strategy in each world, whereas in reality there are situations (or information sets) which consist of multiple perfect information scenarios. In the full imperfect information game, a player cannot distinguish between these situations, and must choose the same strategy in each one; but PIMC search erroneously assumes that it can choose a strategy tailored to each individual scenario.

We illustrate strategy fusion in Figure 1(a). The maximizing player is represented as an upward pointing triangle, and the minimizing player by a downward triangle. Terminal nodes are squares with payoffs for the max player below them. There are two worlds which would be created by a chance node higher in the tree. We assume neither player knows whether they are in world 1 or 2, so we do not show this information in the tree.

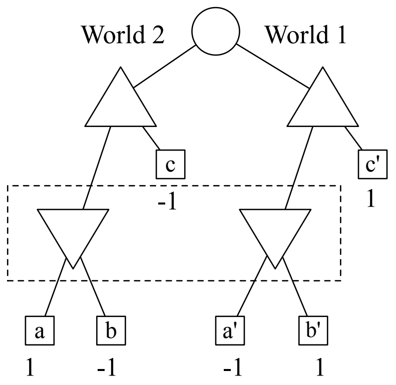
At the root, the maximizing player has the choice of moving to the right to node (c) where a payoff of 1 is guaranteed, no matter the world. The maximizing player can also get a payoff of 1 from the nodes marked (a) in World 1 and the nodes marked (b) in World 2. PIMC search will think that it can always make the right decision above nodes (a) and (b), and so both moves at the root look like wins. However, in reality the max player is confused between worlds 1 and 2 and may actually make a mistake in disambiguation on the left side of the tree. We note that there are two conditions required for strategy fusion to actually cause an error in the play of PIMC search. First, there must be moves which are anti-correlated values (nodes (a) and (b)) on one portion of the tree, and second, there must be a move which is guaranteed to be better on the other side of the tree. If node (c) had the value -1, PIMC search would make the correct decision, although it would overestimate the value of the tree.

The second error identified by Frank and Basin is termed *non-locality*. Non-locality is a result of the fact that in a perfect information game, the value of a game tree node is a function only of its subtree, and therefore the value of a node is completely determined by a search starting with its children. In an imperfect information game, a node's value may depend on other regions of the game tree not contained within its subtree, primarily due to the opponent's ability to direct the play towards regions of the tree that he knows (or at least guesses) are favorable for him, using private information that he possesses but we do not. This phenomenon creates non-local dependencies between potentially distant nodes in the tree.

We illustrate non-locality in Figure 1(b). In this figure there is a chance node at the top of the tree. The maximizing player knows the chance action, but the minimizing player cannot distinguish between the states within the dotted rectangle. In this tree PIMC search would make a random move for the minimizing player. But, in fact, the minimizing player can always know the correct move. Because the maximizing player will take the win in world 1 if possible, the minimizing player will only have an opportunity to play if he is in world 2, when the maximizing player moves to the left to avoid the immediate loss. Thus, the minimizing player can infer the correct world and the correct action.



(a) Strategy Fusion



(b) Non-locality

Figure 1: Examples of strategy fusion and non-locality.

While we will not create these structures explicitly in our game model, we will be able to tune the probability that they occur and that PIMC search will be confused. We can also measure how often this occurs in actual game trees.

Domains

We use two illustrative domains in this paper. The first is a class of trick-based card games. The precise rules for the domain are not important for our purposes, but the actions in the domain are. In a trick-based card game an action is to play a card from one's hand onto the table face up. This has two implications. First, information is revealed and information sets are split when actions take place. Second, there are many possible legal actions. Most western games use a 52 card deck, allowing up to 52 possible actions at each node in the game tree. Some European card games use a short deck of 32 cards, resulting in at most 32 actions in each state.

The second domain we examine is Poker. Again, there are many variants of poker which we will not discuss here. What is important is that there are a limited number of actions (*bet, raise, call, fold*), and actions do not directly reveal any hidden information. Therefore, the number of true game states in each information set in poker does not change with the action of a player in the game.

Between 2003 and 2010 the size of Poker game trees that can be solved has grown by 5 orders of magnitude, from 10^7 states (Billings et al. 2003) to 10^{12} states (Zinkevich et al. 2008). The most recent gains are a result of the Counter-factual Regret algorithm (CFR) (Zinkevich et al. 2008), which is able to approximately solve games in space proportional to the number of information sets and in time $O(I^2N)$, where I is the number of information sets and N is the number of games states.

Further Motivation

Ultimately, the decision about what technique used to solve or approximate a solution to a game depends primarily on the cost of generating that solution and the quality of the resulting solution. CFR requires building strategies for all players and iterating over all information sets. There are effective abstractions which have been applied to poker which are able to significantly reduce the size of the game trees that must be solved without significantly reducing the quality of play. This works particularly well in poker because player actions do not directly reveal information about the state of the game. Consider, however, a trick-based card game.

First, we analyze a 4-player card game with 52 cards, such as Bridge. There are $\binom{52}{13} \approx 6.35 \times 10^{11}$ possible hands that can be dealt to each player. Thus, the number of hands for a single player already approaches the limits of the number of states that can be solved with CFR, and this doesn't include the hands for the opponents and possible lines of play. So, completely solving such games is out of the question.

What about smaller trick-based card games? Skat is a 3-player card game with 32 cards, from which 10 are dealt to each player. There are $\binom{32}{10} = 364,512,240$ hands each player can have and $H := \binom{22}{10}\binom{12}{10} = 42,678,636$ hands for the other players which is what constitutes an information set at the start of a game. At the beginning of each trick, the trick leader can choose to play any of his remaining cards. Therefore, there are at least $10!H \approx 1.54 \cdot 10^{14}$ information sets. But, from an opponents perspective there are actually 20 or 22 unknown cards that can be lead, so this is only a loose lower bound on the size of the tree. This should clearly establish that even when using short decks it is infeasible to solve even a single game instance of a trick-based card game.

We have worked on solving sub-portions of a game. For instance, if a game is still undecided by the last few plays it is possible to build and solve the game tree using CFR and balance the possible cards that the opponents hold using basic inference. In the game of Skat approximately 15% of games are still unresolved when there are three tricks left in the game, based on analyzing thousands of games played on a Skat server. Over a randomly selected set of 3,000 unresolved games PIMC makes mistakes that cost a player 0.42 tournament points (TPs) per deal on average against the solution computed by CFR. We must also note the caveat that CFR is not guaranteed to produce optimal solutions to multi-player games such as Skat; however, in practice, it often seems to do so, especially for small games of the type considered here. If we assume the value of 0.42 to be close to the true loss against a Nash-optimal player, then as only 15% of games are unresolved at this point, PIMC's average loss is only 0.063 TP per deal. In a series of 180 deals for each player in typical Skat tournaments the expected loss amounts to 11.3 TPs, which is dwarfed by the empirical TP standard deviation of 778. Thus, the advantage over PIMC in the endgame hardly matters for winning tournaments.

Finally, we have also looked into methods for abstracting trick-based game trees to make solutions more feasible. While this approach has shown limited success, it has not, on average, shown to be a better approach than existing PIMC methods. While we may eventually make progress in this area, we have pursued the work described here in order to better understand why PIMC has been so strong in the domains we are interested in.

Methodology

As outlined in the background above, work by Frank and Basin has already formalized the kinds of errors made by PIMC search through the concepts of strategy fusion and non-locality. However, not only does the mere presence of these properties seem difficult to detect in real game trees where computing a full game-theoretic solution is infeasible, but as we have previously argued, their presence alone is not enough to necessarily cause PIMC search to make mistakes in its move selection.

Therefore, instead of focusing on the concepts of strategy fusion and non-locality directly, our approach is to measure elementary game tree properties that probabilistically give rise to strategy fusion and non-locality in a way that causes problems for PIMC search. In particular, we consider three basic properties.

- *Leaf Correlation*, lc, gives the probability all sibling, terminal nodes have the same payoff value. Low leaf node correlation indicates a game where it is nearly always possible for a player to affect their payoff even very late in a game.
- *Bias*, b, determines the probability that the game will favor a particular player over the other. With very high or very low bias, we expect there to be large, homogeneous sections of the game, and as long as a game-playing algorithm can find these large regions, it should perform well in such games.
- *Disambiguation factor*, df, determines how quickly the number of nodes in a player's information set shrinks with regard to the depth of the tree. For instance, in trick-taking card games, each play reveals a card, which means the number of states in each information set shrinks drastically as the game goes on. Conversely, in a game like poker, no private information is directly revealed until the game is over. We can determine this factor by considering how much a player's information set shrinks each time the player is to move.

All of these properties can easily be measured in real game trees, as we describe below.

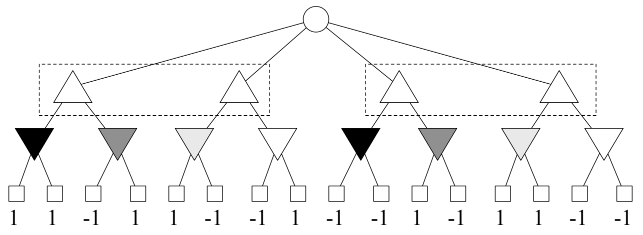


Figure 2: A sample of a depth 2 synthetic tree, with 2 worlds per player. Max nodes in boxes and min nodes with the same shading are in the same information set respectively.

Measuring Properties in Real Games

To measure leaf correlation, bias, and the disambiguation factor in real games (i.e. *large* games) we suggest using random playouts to sample the terminal nodes (unless there is some a priori reason to discard portions of the game tree as uninteresting or irrelevant). Once a terminal node is reached, the bias and correlation may be estimated from the local neighbourhood of nodes. Along these random playout paths the size of the information set to which each node belongs may be computed in a straightforward manner, and compared to the subsequent size when next that player is to move. It is then straightforward to convert the average reduction ratio into a *df* value for one's desired model.

Experiments

In this section, we first perform experiments on synthetic trees measuring the performance of PIMC search in the face of various tree properties, and then show the measurement of these properties in the real games that make up our domains of interest.

Synthetic Trees

We construct the simple, synthetic trees used in our experiments as follows. We assume, for simplicity's sake, that our trees represent a two-player, zero-sum, stochastic imperfect information game. We may also assume, without loss of generality, that the game has alternating moves between the two players, and that all chance events that occur during the game are encapsulated by a single large chance node at the root of the game tree. Each player node is of degree 2, while the degree of the beginning chance node is defined in terms of worlds per player, W. Furthermore, the information concerning these worlds is assumed to be strictly disjoint; for each world of player $p1$, there are initially W worlds for player $p2$ that $p1$ cannot distinguish. We restrict ourselves to this disjoint case because in cases where the players' information overlaps, the game collapses to a perfect information stochastic game (i.e. there may be information unknown to both players, but at least they are in the same boat). Therefore, the total degree of the chance node is W^2. Game tree nodes are initially partitioned into information sets based on these worlds. We assume that all player moves are observed by both players, in the sense that both players know whether the player to move chose the 'left' or 'right' branch at each of their information sets. Finally, terminal payoffs are restricted to be either 1 (a win for $p1$) or -1 (a win for $p2$). A small sample of such a tree is presented in Figure 2.

Now, under the above assumptions, we define our three properties in the synthetic tree context, each one of which is continuous valued in the range $[0, 1]$. We describe below the effect of these parameters on the construction of the synthetic trees.

- *Leaf Correlation*, lc: With probability lc, each sibling pair of terminal nodes will have the same payoff value (whether it be 1 or -1). With probability $(1 - lc)$, each sibling pair will be *anti-correlated*, with one randomly determined leaf having value 1 and its sibling being assigned value -1.
- *Bias*, b: At each *correlated* pair of leaf nodes, the nodes' values will be set to 1 with probability b and -1 otherwise. Thus, with bias of 1, all correlated pairs will have a value of 1, and with bias of 0.5, all correlated pairs will be either 1 or -1 at uniform random (and thus biased towards neither player). Note that *anti-correlated* leaf node pairs are unaffected by bias.
- *Disambiguation factor*, *df*: Initially, each information set for each player will contain W game nodes. Each time p is to move, we recursively break each of his information sets in half with probability *df* (thus, each set is broken in two with probability *df*; and if a break occurs, each resulting set is also broken with probability *df* and so on). If *df* is 0, then p never gains any direct knowledge of his opponent's private information. If *df* is 1, the game collapses to a perfect information game, because all information sets are broken into sets of size one immediately. Note that this generative model for *df* is slightly different than when measuring disambiguation in real games trees.

Note that we do not specifically examine correlation within an information set; rather, we hypothesize that these properties represent the lowest level causes of tree structures that result in problems for PIMC search, and that if they are present with sufficient frequency, then higher-level confusion at the information set level will occur.

Experiments on Synthetic Game Trees

Using the synthetic game tree model outlined above, we performed a series of experiments comparing the playing strength of both PIMC search and a uniform random player against an optimal Nash-equilibrium player created using the CFR algorithm. In each experiment, synthetic trees were created by varying the parameters for leaf correlation, bias and disambiguation. Tree depth was held constant at depth 8, and we used 8 worlds per player at the opening chance node, for a total chance node size of 64. Playing strength is measured in terms of average score per game, assuming 1 point for a win and -1 for a loss. For each triple of parameter values, we generated 10000 synthetic trees and played 2 games per tree, with the competing players swapping sides in the second game.

The results of these tests are presented in figures 3 through 5. For ease of visualization, each figure plots two parameters

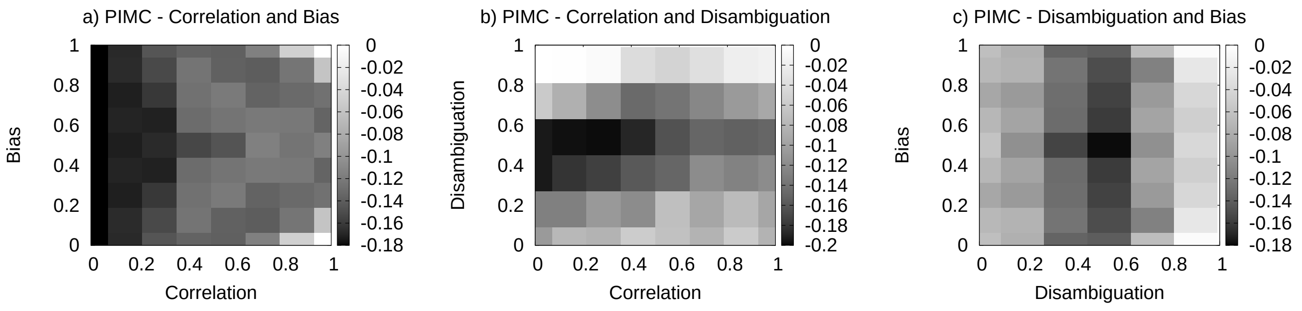


Figure 3: Performance of PIMC search against a Nash equilibrium. Darker regions indicate a greater average loss for PIMC. Disambiguation is fixed at 0.3, bias at 0.75 and correlation at 0.5 in figures a, b and c respectively.

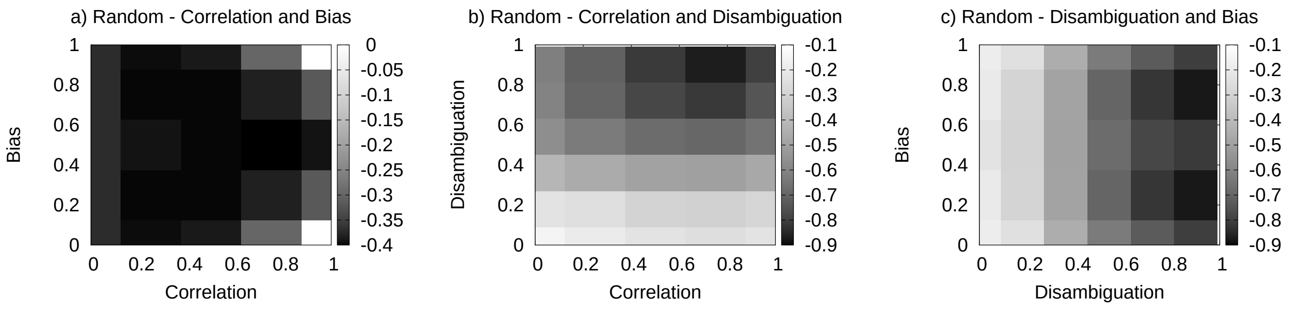


Figure 4: Performance of random play against a Nash equilibrium. Darker regions indicate a greater average loss for random play. Disambiguation is fixed at 0.3, bias at 0.75 and correlation at 0.5 in figures a, b and c respectively.

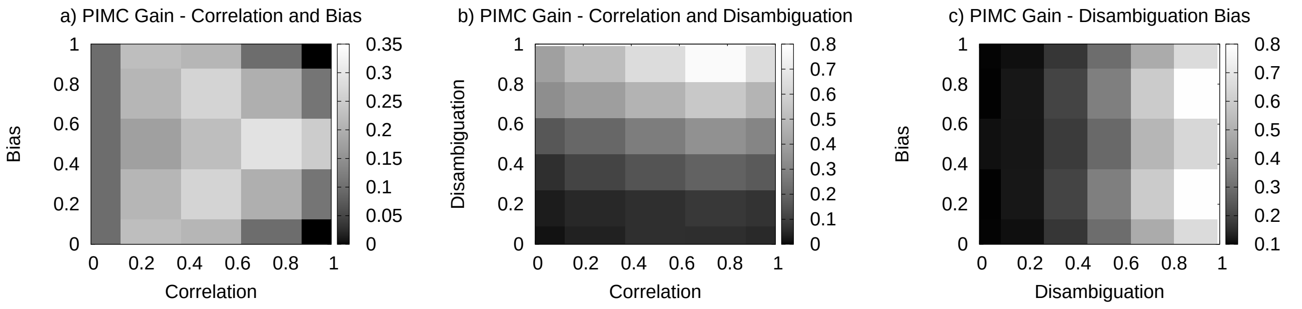


Figure 5: Performance gain of PIMC search over random against a Nash equilibrium. Darker regions indicate minimal performance gain for using PIMC search over random play. Disambiguation is fixed at 0.3, bias at 0.5 and correlation at 0.75 in figures a, b and c respectively.

against each other on the x and y axes, while the third parameter is held constant. Figures 3 and 4 shows the playing performance of the challenging player (either PIMC search or uniform random) against the equilibrium player. White shaded regions are areas of the parameter space where the challenger breaks even with equilibrium. The darker the shading, the greater the challenger's loss against the equilibrium. Figure 5 is similar, except that the shading represents the *gain* of PIMC search over the random player when playing against equilibrium. Dark regions of these plots represent areas where PIMC search is performing almost no better than the random player, whereas lighter regions indicate a substantial performance gain for PIMC search over random.

These plots show that PIMC search is at its worst when leaf node correlation is low. This is true both in absolute performance, and in PIMC's relative improvement over random play. The most likely explanation for this behavior is that when anti-correlation occurs deep in the game tree – particularly at the leaves – then PIMC search always believes that the critical decisions are going to come 'later' and that what it does higher up the tree does not actually matter. Of course, when an information set structure (which PIMC ignores at every node except the root of its own search) is imposed on the tree, early moves frequently *do* matter, and thus the superior play of the equilibrium player. When correlation is medium to low, bias also seems to play a role here, with more extreme bias resulting in better performance for PIMC, although the effect of bias is generally small. The

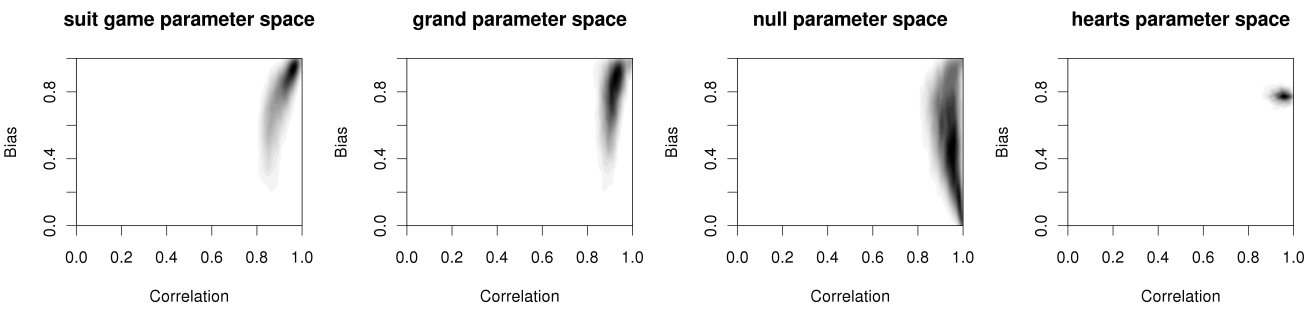


Figure 6: Parameter space estimation for Skat game types and Hearts. Dark regions correspond to a high density of games with those measured parameters. Values were sampled using 10000 games for each skat type and 3000 games for hearts. Bias is given in terms of score w.r.t. a fixed player.

performance gain due to bias for PIMC is likely because a more extreme bias reduces the probability of interior nodes that are effectively anti-correlated occuring perhaps one or two levels of depth up from the leaves of the tree. Note that, of course, in the case of maximum bias and correlation, even the random player will play perfectly, since the same player is guaranteed to win no matter what the line of play (we can only suppose these would be very boring games in real life).

The situation with the disambiguation factor initially appears counter-intuitive; it appears that a low disambiguation factor is good for the absolute performance of PIMC search, while the worst case is a mid-range disambiguation factor. However, in the relative case of PIMC's gain over random, the trend is very clearly reversed. The explanation for this lies in the fact that the random player performs relatively well in games with a low disambiguation factor. In some sense, because there is so much uncertainty in these games, there is a lot of 'luck,' and there is only so much an optimal player can do to improve his position. As we increase the disambiguation factor, the performance of the random player deteriorates rapidly, while PIMC search is much more successful at holding its own against the optimal player. As disambiguation approaches 1, the performance of PIMC improves drastically, since the game is approaching a perfect information game. Finally, we note that with a high disambiguation in the 0.7-0.9 range, low correlation is actually good for PIMC's performance. This is a result of the fact that these games become perfect information games very quickly, and low correlation increases the probability that a win is still available by the time the PIMC player begins playing optimally in the perfect information section of the tree.

Real Games

To test the predictive powers of our three properties, we estimated the distribution of those parameters for actual games. The first game so measured is Skat. Although the exact rules are unimportant, the specific type of Skat game varies depending on an initial auction phase. The winner of the auction (the *soloist*) chooses the game type and competes against the two other players (who now form a temporary coalition). The two most common game types are *suit* games and *grand* games; both have a trump suit and are concerned with taking high-valued tricks. The third type of game is *null*, in which the soloist tries not to win any tricks (and loses if even one trick is won).

For each game type, 10000 human-bid games were explored using random actions from the start of the cardplay phase. In each game correlation and bias were measured 1000 times near the leaves. To do this, we walk down the tree, avoiding moves which lead to terminal positions (after collapsing chains of only one legal move). When all moves lead directly to terminal positions we take their value to be the game value with respect to the soloist (to emulate the values of the fixed depth synthetic trees). We say these "pre-terminal" nodes are correlated if all move values are the same, and compute bias as the fraction of correlated nodes which are soloist wins.

Disambiguation was measured by comparing the change in the number of possible (consistent) worlds since the current player was last to move. Only 10 disambiguation rollouts were performed per world, since the resulting ratios were tightly clustered around $df = 0.6$. The observed distributions are shown in Fig. 6. In this figure, we also display results for Hearts, which, like Skat, is a trick-taking card game, but played with a larger deck and different scoring rules. 3000 Hearts games using 500 sample points per game were used to generate this data.

For both the skat and hearts games, the resulting graphs show a very high level of correlation (from 0.8 to nearly 1.0), with bias varying more widely and disambiguation very close to 0.6, as mentioned above. Examining Figures 3(b) and 5(b) puts skat in a parameter space where the PIMC player loses only 0.1 points per game against equilibrium and gains 0.4 points over random play (recalling that our synthetic trees use payoffs from -1 to 1), with perhaps plus or minus 0.05 points depending on the bias of the individual hand. This seems like relatively good performance for PIMC search, which coincides with our motivating evidence that PIMC search seems to perform well in these games in practice.

The second game we measured is Kuhn poker, a highly

	Opponent	
Player:	Nash	Best-Response
Random (p1)	-0.161	-0.417
Random (p2)	-0.130	-0.500
PIMC (p1)	-0.056	-0.083
PIMC (p2)	0.056	-0.166

Table 1: Average payoff achieved by random and PIMC against Nash and best-response players in Kuhn poker.

simplified poker variant for which Nash-optimal solutions are known. In this game two players are each dealt one card out of a deck of three. The game proceeds as: both players ante; player 1 may check or raise; player 2 may fold, check, call, or raise as appropriate; if the game is still proceeding, player 1 may fold or call. The player with the high card then wins the pot. With Nash-optimal strategies player 1 is expected to lose 1/18 = $0.0\bar{5}$ bets per game and player 2 to win 1/18 bets per game.

This game has a disambiguation factor of 0, since no cards are revealed (if at all) until the end, and the size of an information set is never decreased. By inspecting the game tree and using our notion of pre-terminal nodes the correlation and bias can be seen to be 0.5 and 0.5 respectively. These parameters are very different from skat and hearts and lie in the portion of parameter space where we would predict that PIMC search performs more poorly and offers little improvement over random play. This is then, perhaps, at least one explanation of why research in the full game of poker has taken the direction of finding game-theoretic solutions to abstract versions of the game rather than tackling the game directly with PIMC search.

We present results comparing play between a random player, PIMC player and Nash player in Table 1. Kuhn poker is not symmetric, so we distinguish the payoffs both as player 1 and player 2. Because the PIMC player does not take dominated actions, when playing against a Nash equilibrium this player achieves the equilibrium payoff, while a random player loses significantly against even an equilibrium player. If an opponent is able to build a best-response against a PIMC player, then the PIMC player is vulnerable to significant exploitation as player 2, while the random player loses -0.5 as the second player, where 0.056 could have been won. Thus, these results present the experimental evidence showing that PIMC is not a good approach in practice for a game like poker. Although it plays better and is less exploitable than a random player, PIMC may lose significantly to a opponent that can model its play.

Conclusion and Future Work

In this paper, we performed experiments on simple, synthetic game trees in order to gain some insight into the mystery of why Perfect Information Monte Carlo search has been so successful in a variety of practical domains in spite of its theoretical deficiencies. We defined three properties of these synthetic trees that seem to be good predictors of PIMC search's performance, and demonstrate how these properties can be measured in real games.

There are still several open issues related to this problem. One major issue that we have not addressed is the potential exploitability of PIMC search. While we compared PIMC's performance against an optimal Nash-equilibrium in the synthetic tree domain, the performance of PIMC search could be substantially worse against a player that attempts to exploit its mistakes. Another issue is that the real games we consider in this paper represent the 'extremes' of the parameter space established by our synthetic trees. It would be informative if we could examine a game that is in between the extremes in terms of these parameters. Such a game could provide further evidence of whether PIMC's performance scales well according to our properties, or whether there are yet more elements of the problem to consider.

Finally, we have seen that in games like skat that there isn't a single measurement point for a game, but a cloud of parameters depending on the strength of each hand. If we can quickly analyze a particular hand when we first see it, we may be able to use this analysis to determine what the best techniques for playing are on a hand-by-hand basis and improve performance further.

Acknowledgements

The authors would like to acknowledge NSERC, Alberta Ingenuity, and iCORE for their financial support.

References

Billings, D.; Burch, N.; Davidson, A.; Holte, R. C.; Schaeffer, J.; Schauenberg, T.; and Szafron, D. 2003. Approximating game-theoretic optimal strategies for full-scale poker. In *IJCAI*, 661–668.

Buro, M.; Long, J. R.; Furtak, T.; and Sturtevant, N. R. 2009. Improving state evaluation, inference, and search in trick-based card games. In *IJCAI*, 1407–1413.

Frank, I., and Basin, D. 1998. Search in games with incomplete information: A case study using bridge card play. *Artificial Intelligence* 87–123.

Ginsberg, M. 2001. GIB: Imperfect Information in a Computationally Challenging Game. *Journal of Artificial Intelligence Research* 303–358.

Russell, S., and Norvig, P. 2002. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 2nd edition.

Sturtevant, N. R. 2008. An analysis of UCT in multi-player games. In *Computers and Games*, 37–49.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2008. Regret Minimization in Games with Incomplete Information. In *Advances in Neural Information Processing Systems 20*, 1729–1736.