The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Computer Poker and Imperfect Information Games: Technical Report WS-16-06

Evaluating the Performance of Presumed Payoff Perfect Information Monte Carlo Sampling Against Optimal Strategies

Florian Wisser
Vienna University of Technology
Vienna, Austria
wisser@dbai.tuwien.ac.at

Abstract

Despite some success of Perfect Information Monte Carlo Sampling (PIMC) in imperfect information games in the past, it has been eclipsed by other approaches in recent years. Standard PIMC has well-known shortcomings in the accuracy of its decisions, but has the advantage of being simple, fast, and scalable, making it well-suited for imperfect information games with large state-spaces. Presumed Payoff PIMC is a variant of PIMC lessening the effect of implicit overestimation of opponent's knowledge of hidden information in future game states, while adding only very little complexity. We give a detailed description of Presumed Payoff PIMC and analyze its performance against Nash-equilibrium approximation algorithms and other PIMC variants in the game of Phantom Tic-Tac-Toe.

Introduction

Perfect Information Monte Carlo Sampling (PIMC) in tree search of games of imperfect information has been around for many years. The approach is appealing, for a number of reasons: it allows the usage of well-known methods from perfect information games, its complexity is magnitudes lower than the problem of weakly solving a game in the sense of game theory, it can be used in a just-in-time manner (no precalculation phase needed) even for games with large state-space, and it has proven to produce competitive AI agents in some games. Let us mention Bridge (Ginsberg 2001), Skat (Buro et al. 2009) and Schnapsen (Wisser 2010).

In recent years research in AI in games of imperfect information was heavily centered around equilibrium approximation algorithms (EAA). Clearly in games small enough for EAAs to work within reasonable time and space limits, EAAs are the method of choice. However, games more complex than e.g. heads-up limit Texas Hold'em with much larger state-spaces will probably never be manageable with EAAs. Maybe only because of the lack of an effective storage for the immense sizes of the resulting strategies. To date, we are not even able so solve large perfect information games like chess or Go. Using state-space abstraction (Johanson et al. 2013) EAAs may still be able to find good strategies for larger games, but they depend on finding an appropriate simplification of manageable size. On the other hand, just-in-time heuristics like PIMC do not need a precalculation phase and hence also do not need to store a strategy. So, we think it is still a worthwhile task to investigate just-in-time heuristics like PIMC that are able to tackle larger problems. In games where EAAs are available the quality of such heuristics can be measured by comparing them to near–Nash strategies, which we will do in the following.

On the other hand, in the 2nd edition (and only there) of their textbook, Russell and Norvig (Russell and Norvig 2003, p179) quite accurately use the term "averaging over clairvoyancy" for PIMC. A more formal critique of PIMC was given in a series of publications by Frank, Basin, et al. (Frank and Basin 1998b; Frank, Basin, and Matsubara 1998; Frank and Basin 1998a; 2001), where the authors show that the heuristic of PIMC suffers from strategy-fusion and non-locality producing erroneous move selection due to an overestimation of MAX's knowledge of hidden information in future game states. A very recent algorithm shows how both theoretical problems can be fixed (Lisý, Lanctot, and Bowling 2015), but has yet to be applied to large games typically used for search. More recently overestimation of MAX's knowledge is also dealt with in the field of general game play (Schofield, Cerexhe, and Thielscher 2013). To the best of our knowledge, all literature on the deficiencies of PIMC concentrates on the overestimation of MAX's knowledge. Frank et al. (Frank and Basin 1998a) explicitly formalize the "best defense model", which basically assumes a clairvoyant opponent, and state that this would be the typical assumption in game analysis in expert texts. This may be true for some games, but clearly not for all.

Think, for example, of a game of heads-up no-limit Texas Hold'em poker playing an opponent with perfect information, knowing your hand as well as all community cards before they even appear on the table. The only reasonable strategy left against such an opponent would be to immediately concede the game, since one will not achieve much more than stealing a few blinds. And in fact expert texts in poker do never assume playing a clairvoyant opponent when analyzing the correctness of the actions of a player.

In the following — and in contrast to the references above — we start off with an investigation of the problem of overestimation of MIN's knowledge, from which PIMC and its known variants suffer. We set this in context to the

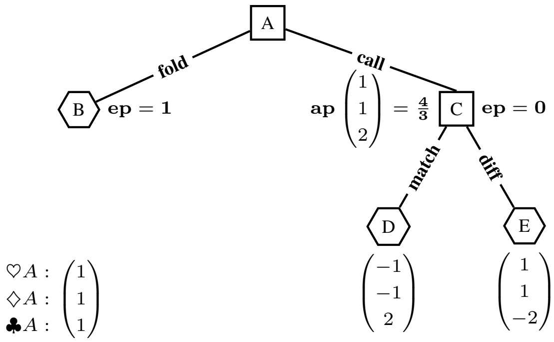

Figure 1: PIMC Tree for XX

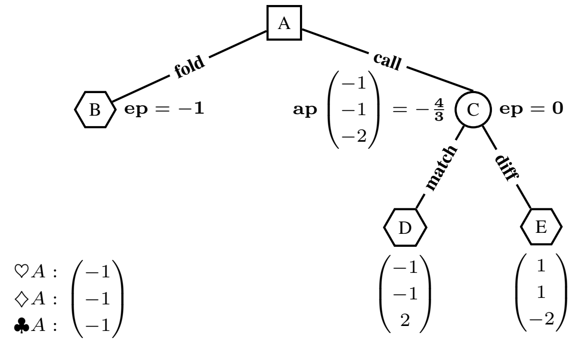

Figure 2: PIMC Tree for XI

best defense model and show why the very assumption of it is doomed to produce sub-optimal play in many situations. For the purpose of demonstration we use the simplest possible synthetic games we could think of. We go on describing a heuristic algorithm termed Presumed Payoff PIMC (Wisser 2015), targeting imperfect information games decided within a single round of play, dealing with the problem of MIN overestimation.

Finally we do an experimental analysis in the game of Phantom Tic-Tac-Toe, where we compare the traditional PIMC algorithm and its enhancement Presumed Payoff PIMC with Counterfactual Regret Minimization in game play as well as with respect to the exploitability of the resulting strategies.

Background Considerations

In a 2-player game of imperfect information there are generally 4 types of information: information publicly available (ι^P), information private to MAX (ι^X), information private to MIN (ι^I) and information hidden to both (ι^H). To exemplify the effect of "averaging over clairvoyancy" we introduce two very simple 2-player games: XX with only two successive decisions by MAX, and XI with two decisions, first one by MAX followed by one of MIN. The reader is free to omit the rules we give and view the game trees as abstract ones. Both games are played with a deck of four aces, $\spadesuit A$, $\heartsuit A$, $\diamondsuit A$ and $\clubsuit A$. The deck is shuffled and each player is dealt 1 card, with the remaining 2 cards lying face down on the table. ι^P consists of the actions taken by the players, ι^H are the 2 cards face down on the table, ι^X the card held by MAX and ι^I the card held by MIN.

In XX, MAX has to decide whether to **fold** or **call** first. In case MAX calls, the second decision to make is to bet, whether the card MIN holds matches color with its own card (**match**, both red or both black) or differs in color (**diff**). Fig. 1 shows the game tree of XX with payoffs, assuming without loss of generality that MAX holds $\spadesuit A$. Modeling MAX's decision in two steps is entirely artificial in this example, but it helps to keep it as simple as possible. The reader may insert a single branched MIN node between A and C to get an identically rated, non-degenerate example. Node C is in fact a collapsed information set containing 3 nodes, which is represented by vectors of payoffs in terminal nodes, representing worlds possible from MAX's point of view. To the right of the child nodes B and C of the root node the expected payoff (**ep**) is given. It is easy to see that the only Nash equilibrium strategy (i.e. the optimal strategy) is to simply fold and cash 1.

To the left of node C, the evaluation of straight PIMC (for better distinction abbreviated by SP in the following) of this node is given, averaging over the payoffs in different worlds after building the point-wise maximum of the payoff vectors in D and E. By construction SP assumes perfect information for *both* players right after the decision it reasons about. This is, both MAX and MIN are assumed to know all of ι^P, ι^X, ι^I and ι^H after the decision currently rated, i.e. the root node. We see that SP is willing to turn down an ensured payoff of 1 by folding, to go for an expected payoff of 0, by calling and going for either bet then. The reason is the well-known overestimation of hidden knowledge, i.e.: it assumes to know ι^I when deciding whether to bet on matching or differing colors in node C, and thereby evaluates it to an average payoff (**ap**) of $\frac{4}{3}$. Frank et al. (Frank and Basin 1998b) analyzed this behavior in detail and termed it strategy-fusion. We will call it MAX-strategy-fusion in the following, since it is strategy-fusion happening in MAX nodes.

The basic solution given for this problem is vector minimax. Instead of evaluating each world independently (as SP does), vector minimax operates on the payoff vectors of each node. In MIN nodes vector minimax resembles SP, building the point-wise minimum of the payoff vectors of all child nodes. However, a MAX nodes evaluates to the payoff vector with the highest mean over all components. The payoff vectors of nodes D and E, $v_D = (-1, -1, 2)$ and $v_E = (1, 1, -2)$ both have a mean of 0. So vector minimax evaluates node C to either of v_D and v_E, and *not* to $(1, 1, 2)$ as SP does. This leads to the correct decision to fold. Note, that by choosing the vector with the best unweighted mean, vector minimax implicitly assumes, that MAX has no knowledge at all about ι^I at any stage of the game. In our toy example XX, this assumption is entirely correct, but it does not hold in many situations in real-world games.

We list the average payoffs for various agents playing XX on the left-hand side of Table 1. Looking at the results we see that a uniformly random agent (RAND) plays worse than a Nash equilibrium strategy (NASH), and SP

XX	—	XI	ANY
PPP	0	VM	-1
SP	0	SP	-1
RAND	$\frac{1}{2}$	RAND	$-\frac{1}{2}$
NASH	1	NASH	0
VM	1	PPP	0

Table 1: Average Payoffs for XX and XI

plays even worse than RAND. VM stands for vector minimax, but includes all variants proposed by Frank et al., most notably payoff-reduction-minimax, vector-minimax-β and payoff-reduction-minimax-β. Any of these algorithms solves the deficiency of MAX-strategy-fusion in this example and plays optimally.

Let us now turn to the game XI, its game tree given in Fig. 2. The two differences to XX are that MAX's payoff at node B is -1, not 1, and C is a MIN (not a MAX) node. The only Nash equilibrium strategy of the game is MAX calling, followed by an arbitrary action of MIN. This leaves MAX with an expected payoff of 0. Conversely, an SP player evaluates node C to the mean of the point-wise minimum of the payoff vectors in D and E, leading to an evaluation of $-\frac{4}{3}$. So SP always folds, since it assumes perfect knowledge of MIN over ι^X, which is just as wrong as the assumption on the distribution of information in XX. Put in another way, in such a situation SP suffers from MIN-strategy-fusion. VM acts identically to SP in this game and, looking at the right-hand side of Table 1, we see that both score an average of -1, playing worse than NASH and even worse than a random player.

The best defense model (Frank and Basin 1998a) is defined by 3 assumptions: MIN has perfect information (A1) (it knows ι^X as well as ι^H), MIN chooses its strategy after MAX (A2) and MAX plays a pure strategy (A3). Both, SP and VM (including all subsumed algorithms), implicitly assume at least (A1). And it is this very assumption, that makes them fail to pick the correct strategy for XI, in fact they pick the worst possible strategy. So it is the model itself that is not applicable here. While XI itself is a highly artificial environment, similar situations do occur in basically every reasonable game of imperfect information. Unlike single-suit problems in Bridge, which were used as the real-world case study by Frank et al., even in the full game of Bridge there are situations where the opponent can be forced to make an uninformed guess. This is exactly the situation created in XI, and in doing so, one will get a better average payoff than the best defense assumption suggests.

Going back to the game XI itself, let us for the moment assume that MIN picks its moves uniformly at random (i.e. C is in fact a random node). An algorithm evaluating this situation should join the vectors in D and E with a probability of $\frac{1}{2}$ each, leading to a correct evaluation of the overall situation. And since no knowledge is revealed until MIN has to take its decision, this is a reasonable assumption in this particular case. The idea behind Presumed Payoff PIMC, described in the following, is to drop assumption (A1) of facing a perfectly informed MIN, and model MIN instead somewhere between a random agent and a perfectly informed agent.

World	ι^P	ι^X	ι^I_j	ι^H_j
w_1	—	$\spadesuit A$	$\heartsuit A$	$\diamondsuit A, \clubsuit A$
w_2	—	$\spadesuit A$	$\diamondsuit A$	$\heartsuit A, \clubsuit A$
w_3	—	$\spadesuit A$	$\clubsuit A$	$\heartsuit A, \diamondsuit A$

Table 2: Worlds Evaluated by PIMC in XX and XI

Presumed Payoff PIMC (PPP)

The approach of PIMC is to create possible perfect information sub-games in accordance with the information available to MAX, this is given a situation $S = (\iota^P, \iota^X)$ possible states of (ι^I_j, ι^H_j) give a world state $w_j(S) = (\iota^P, \iota^X, \iota^I_j, \iota^H_j)$. So, there is no hope for a method analogous to vector minimax in MIN nodes, since the perfect information sub-games we construct are in accordance with ι^P and ι^X and one can hardly assume that MIN forgets about its own private information, while perfectly knowing ι^X when reasoning over its choices.

Recently, Error Allowing Minimax (EAM), an extension of the classic minimax algorithm, was introduced (Wisser 2013). While EAM is an algorithm suited for perfect information games it was created for the use in imperfect information games following a PIMC approach. It defines a custom operator for MIN node evaluation to provide a generic tie-breaker for equally rated actions in games of perfect information. The basic idea of EAM is to give MIN the biggest possible opportunity to make a decisive error. By a decisive error we mean an error leading to a game-theoretically unexpected increase in the games payoff for MAX. To be more specific, the EAM value for MAX in a node H is a triple (m_H, p_H, a_H). m_H is the standard minimax value. p_H is the probability for an error by MIN, if MIN was picking its actions uniformly at random. Finally a_H is the guaranteed advancement in payoff (leading to a payoff of $m_H + a_H$ with $a_H \geq 0$ by definition of EAM) in case of *any* decisive error by MIN. The value p_H is only meant as a generic estimate to compare different branches of the game tree. p_H — seen as an absolute value — does *not* reflect the true probability for an error of a realistic MIN player in a perfect information situation. What it does reflect is the probability for a decisive error by a random player. One of the nice features of EAM is that it calculates its values entirely out of information encoded in the game tree. Therefore, it is applicable to any N-ary tree with MAX and MIN nodes and does *not* need any specific knowledge of the game or problem it is applied to. We will use EAAB (Wisser 2013), an EAM variant with very effective pruning capabilities. The respective function **eaab** returns an EAM value given a node of a perfect information game tree.

To define Presumed Payoff Perfect Information Monte Carlo Sampling (PPP) we start off with a 2-player, zero-sum game G of imperfect information between MAX and MIN. As usual we take the position of MAX and want to evaluate the possible actions in an imperfect information situation S

observed by MAX. We take the standard approach to PIMC. We create perfect information sub-games

$$w_j(S) = (\iota^P, \iota^X, \iota_j^I, \iota_j^H), \quad j \in \{1, \dots, n\}$$

in accordance with S. In our implementation w_j are not chosen beforehand, but created on-the-fly using a Sims table based algorithm (Wisser 2010), allowing seamless transition from random sampling to full explorations of all perfect information sub-games, if time to think permits. Let $\mathcal{N}$ be the set of nodes of all perfect information sub-games of G. For all legal actions A_i, $i \in \{1, \dots, l\}$ of MAX in S let $S(A_i)$ be the situation derived by taking action A_i. For all w_j we get nodes of perfect information sub-games $w_j(S(A_i)) \in \mathcal{N}$ and applying EAAB we get EAM values

$$e_{ij} := \mathbf{eaab}(w_j(S(A_i))).$$

The last ingredient we need is a function $k : \mathcal{N} \to [0, 1]$, which is meant to represent an estimate of MIN's knowledge over ι^X, the information private to MAX. For all $w_j(S(A_i))$ we get a value $k_{ij} := k(w_j(S(A_i)))$. While all definitions we make throughout this article would remain well-defined, we still demand $0 \le k_{ij} \le 1$, with 0 meaning no knowledge at all and 1 meaning perfect information of MIN. Contrary to the EAM values e_{ij}, which are generically calculated out of the game tree itself, k_{ij} have to be chosen ad hoc in a game-specific manner, estimating the distribution of information. At first glance this seems a difficult task to do, but a coarse estimate suffices. We allow different k_{ij} for different EAM values, since different actions of MAX may leak different amounts of information. This implicitly leads to a preference for actions leaking less information to MIN. For any pair $e_{ij} = (m_{ij}, p_{ij}, a_{ij})$ and k_{ij} we define the extended EAM value $x_{ij} := (m_{ij}, p_{ij}, a_{ij}, k_{ij})$. After this evaluation step we get a vector $x_i := (x_{i1}, \dots, x_{in})$ of extended EAM values for each action A_i.

To pick the best action we need a total order on the set of vectors of extended EAM values. So, let x be a vector of n extended EAM values:

$$x = \begin{pmatrix} x_1 \\ \cdots \\ x_n \end{pmatrix} = \begin{pmatrix} (m_1, p_1, a_1, k_1) \\ \cdots \\ (m_n, p_n, a_n, k_n) \end{pmatrix} \tag{1}$$

We define three operators $\mathbf{pp}$, $\mathbf{ap}$ and $\mathbf{tp}$ on x as follows:

$$\begin{aligned} \mathrm{pp}(x_j) &:= k_j \cdot m_j + \\ &\quad +(1 - k_j) \cdot \big((1 - p_j) \cdot m_j + p_j \cdot (m_j + a_j)\big) \\ &\quad = m_j + (1 - k_j) \cdot p_j \cdot a_j \\ \mathbf{pp}(x) &:= \frac{\sum_{j=1}^{n} \mathrm{pp}(x_j)}{n} \\ \mathbf{ap}(x) &:= \frac{\sum_{j=1}^{n} m_j}{n} \\ \mathbf{tp}(x) &:= \frac{\sum_{j=1}^{n} \big((1 - p_j) \cdot m_j + p_j \cdot (m_j + a_j)\big)}{n} \end{aligned} \tag{2}$$

The presumed payoff ($\mathbf{pp}$), the average payoff ($\mathbf{ap}$) and the tie-breaking payoff ($\mathbf{tp}$) are real numbers estimating the terminal payoff of MAX after taking the respective action. For any action A with associated vector x of extended EAM values the following propositions hold:

$$\begin{gathered} \mathbf{ap}(x) \le \mathbf{pp}(x) \le \mathbf{tp}(x) \\ k_j = 1,\ \forall j \in \{1, \dots, n\} \Rightarrow \mathbf{ap}(x) = \mathbf{pp}(x) \\ k_j = 0,\ \forall j \in \{1, \dots, n\} \Rightarrow \mathbf{pp}(x) = \mathbf{tp}(x) \end{gathered} \tag{3}$$

The average payoff is derived from the payoffs as they come from standard minimax, assuming to play a clairvoyant MIN, while the tie-breaking payoff implicitly assumes no knowledge of MIN over MAX's private information. The presumed payoff lies somewhere in between depending on the choice of all k_j, $j \in \{1, \dots, n\}$.

By the heuristic nature of the algorithm, none of these values is meant to be an exact predictor for the expected payoff ($\mathbf{ep}$) playing a Nash equilibrium strategy, which is reflected by their names. Nonetheless, what one can hope to get is that $\mathbf{pp}$ is a better *relative* predictor than $\mathbf{ap}$ in game trees where MIN-strategy-fusion happens. To be more specific, $\mathbf{pp}$ is a better relative predictor if for two actions A_1 and A_2 with $\mathbf{ep}(A_1) > \mathbf{ep}(A_2)$ and associated vectors x_1 and x_2, $\mathbf{pp}(x_1) > \mathbf{pp}(x_2)$ holds in more situations than $\mathbf{ap}(x_1) > \mathbf{ap}(x_2)$ does.

Finally, for two actions A_1, A_2 with associated vectors x_1, x_2 of extended EAM values we define

$$\begin{aligned} A_1 \le A_2 :\Leftrightarrow &\big(\mathbf{pp}(x_1) < \mathbf{pp}(x_2)\big) \vee \\ &\big(\mathbf{pp}(x_1) = \mathbf{pp}(x_2) \wedge \mathbf{ap}(x_1) < \mathbf{ap}(x_2)\big) \vee \\ &\big(\mathbf{pp}(x_1) = \mathbf{pp}(x_2) \wedge \mathbf{ap}(x_1) = \mathbf{ap}(x_2) \wedge \\ &\quad \mathbf{tp}(x_1) \le \mathbf{tp}(x_2)\big) \end{aligned} \tag{4}$$

to get a total order on all actions, the lexicographical order by $\mathbf{pp}$, $\mathbf{ap}$ and $\mathbf{tp}$. $\mathbf{tp}$ only breaks ties of $\mathbf{pp}$ and $\mathbf{ap}$ if there are different values of k for different actions. In our case study we will not have this situation, but we still want to mention the possibility for completeness.

Going back to XI (Fig. 2), the extended EAM-vectors of child nodes B and C of the root node are

$$x_B = \begin{pmatrix} (-1, 0, 0, 0) \\ (-1, 0, 0, 0) \\ (-1, 0, 0, 0) \end{pmatrix}, \quad x_C = \begin{pmatrix} (-1, 0.5, 2, 0) \\ (-1, 0.5, 2, 0) \\ (-2, 0.5, 4, 0) \end{pmatrix}$$

The values in the second and third slot of the EAM entries in x_C are picked by the $\min$ operator of EAM, combining the payoff vectors in D and E. E.g. $(-1, 0.5, 2, 0)$: If MIN picks D MAX loses by -1, but with probability 0.5 it picks E leading to a score of $-1 + 2$ for MAX. Since the game does not allow MIN to gather any information on the card MAX holds, we set all knowledge values to 0. For both nodes, B and C we calculate their $\mathbf{pp}$ value and get $\mathbf{pp}(x_B) = -1 < 0 = \mathbf{pp}(x_C)$. So contrary to SP and VM, PPP correctly picks to call instead of folding and plays the NASH strategy (see Table 1). Note that in this case $\mathbf{pp}(x_C) = 0$ even reproduces the correct expected payoff, which is not a coincidence, since all parameter in the heuristic are exact. In more complex game situations with other knowledge estimates this will generally not hold. But PPP picks the correct strategy in this example as long as $k_{C1} = k_{C2} = k_{C3}$ and $0 \le k_{C1} < \frac{3}{4}$

holds, since $\mathbf{pp}(x_B) = -1$ for any choices of k_{Bj} and $\mathbf{pp}(x_C) = -\frac{4}{3}k_{C1}$. As stated before, the estimate on MIN's knowledge can be quite coarse in many situations. To close the discussion of the games XX and XI, we once more look at the table of average payoffs (Table 1). While SP suffers from both, MAX-strategy-fusion and MIN-strategy-fusion, VM resolves MAX-strategy-fusion, while PPP resolves the errors from MIN-strategy-fusion.

In the influential article "Overconfidence or Paranoia? Search in Imperfect-Information Games" (Parker, Nau, and Subrahmanian 2006) the authors discuss search techniques assuming either a random opponent or a perfect one. It is fair to say, that PPP is a PIMC approach on the "overconfident" side, deriving its decisions partly from the assumption of a random opponent, while VM represents the "paranoia" side of the approach. However, the algorithms discussed by Parker et al. choose an opponent model in the absence of the knowledge of the actual opponent's strategy, and calculate a best response to this modeled strategy. The resulting algorithms need to traverse the entire game tree, without any option for pruning. This renders their computation intractable for most non-trivial games. On the other hand, PPP retains the scalability of SP keeping the algorithm open for games with larger state spaces.

We close this section with a few remarks. First, while VM (meaning all subsumed algorithms, including the β-variants) increases the computational costs in relation to SP by roughly one magnitude, PPP only does so by a fraction. Second, we checked all operators needed for VM as well as for EAM and it is perfectly possible to redefine these operators in a way that both methods can be blended in one algorithm. Third, while reasoning over the history of a game may expose definite or probabilistic knowledge of parts of ι^I, we still assume all worlds w_j to be equally likely, i.e. we do not model the opponent. If one decides to use such modeling by associating a probability to each world, the operators defined in equation (2) can be modified easily to reflect these probabilities. Forth, there is a variant of PPP termed Presumed Value PIMC (PVP) aiming at games of imperfect information that are not decided within a single round of play (Wisser 2015).

Phantom Tic-Tac-Toe — a Case Study

PPP was designed with the Central–European tricktaking card game Schnapsen in mind. In this particular game it has produced an AI agent playing above human expert level (Wisser 2015). Since the game has an estimated 10^{20} information sets this game is too large to be open to an EAA approach, at least without abstraction. Searching for a game in which we could comparatively evaluate PPP against EAAs we ran into Phantom Tic-Tac-Toe.

Phantom Tic-Tac-Toe (PTTT) is similar to the classic game of Tic-Tac-Toe played on a 3x3 board where the goal of one player is to make a straight horizontal, vertical, or diagonal line. However there is one critical difference: each players does not get to see the moves made by the opponent, so they do not know the true state of the board. Each player submits a move to a referee who knows the true state of the board, and only that player is told whether it succeeded (there was no opponent piece on that square) or failed (there was an opponent piece on the square). The turn only alternates when a player plays a successful move, so it is possible to try several actions that fail in a row before one succeeds. We will in the following stick with the usual convention, that the player taking the first action uses symbol X to mark its actions (player X) and the other player uses O to mark its actions (player O).

$(0.124, -0.625, 0.3, 0.19)$	$(-0.136, -0.875, 0.038, 0.19)$	$(0.124, -0.625, 0.3, 0.19)$
$(-0.136, -0.875, 0.038, 0.19)$	$(0.759, 0.0, 0.938, 0.19$	$(-0.136, -0.875, 0.038, 0.19)$
$(0.124, -0.625, 0.3, 0.19)$	$(-0.136, -0.875, 0.038, 0.19)$	$(0.124, -0.625, 0.3, 0.19)$

Figure 3: PPP Evaluation of First O Action in PTTT

We choose Phantom Tic-Tac-Toe for several reasons. First, phantom games have been a classical interest for application of search algorithms to imperfect information, especially Monte Carlo techniques (Ciancarini and Favini 2010; Cazenave 2006). Simply hiding players' actions makes the game significantly harder, larger, and amenable to new search techniques (Auger 2011; Lisy 2014). PTTT has approximately 10^{10} unique full game histories and $5.6 \cdot 10^6$ information sets (Lanctot 2013b, Section 3.1.7). Second, it has been used as a benchmark domain in several analyses in computational game theory (Teytaud and Teytaud 2011; Lanctot et al. 2012; Bosansky et al. 2014).

The AI agents we compare are CFR, PPP, SP, VM, PRM, and RAND. Counterfactual Regret Minimization (Zinkevich et al. 2008) (CFR) is a popular algorithm that has been widely successful in computer poker, culminating in the solving of heads-up limit Texas Hold'em poker (Bowling et al. 2015). We base our implementation on the pseudo-code provided in (Lanctot 2013b, Algorithm 1) and Marc Lanctot's public implementation (Lanctot 2013a). In this domain, since the same information set can be reached by many different combinations of opponent actions, to avoid strategy update interactions during an iteration we store regret changes and then only apply them (and rebuild the strategies) on the following iteration. The CFR strategy we use was trained in 64k iterations.

Straight Perfect Information Monte Carlo sampling (SP) goes through all possible configurations of hidden information (unknown opponents moves) leading to a set of perfect information sub-games. These sub-games are evaluated with standard minimax, so each game state reached by a legal action is associated with a vector of minimax values. To decide on which action to take, the mean values over the components of these vectors are compared, and one vector with the highest mean value is picked (breaking ties randomly). This is the standard approach to PIMC and this agent is meant as a reference to compare PPP. Comparing the numbers SP and PPP produce in the evaluation, one can check the improvement the small change in the otherwise unaltered method achieves.

X \ O	CFR	PPP	SP	VMM	PRM	RAND
CFR	0.6654	0.6653	0.6655	0.8124	0.7333	0.9348
PPP	0.6666	0.7503	0.7501	0.8593	0.7555	0.9532
SP	0.6321	0.4538	0.4533	0.7712	0.5574	0.9343
VMM	0.1333	-0.0207	-0.0195	0.1095	0.1718	0.7040
PRM	0.2109	-0.0876	-0.0873	0.1461	0.1006	0.7107
RAND	-0.3789	-0.6192	-0.6172	-0.4849	-0.4542	0.2970

Table 3: Payoff Table for Player X in PTTT (2.5M games each match)

X \ O	CFR	PPP	SP	VMM	PRM	RAND
CFR	71.82% / 5.28%	74.69% / 8.16%	74.72% / 8.17%	85.32% / 4.09%	78.85% / 5.53%	94.54% / 1.06%
PPP	66.66% / 0.00%	75.03% / 0.00%	75.01% / 0.00%	85.93% / 0.00%	75.55% / 0.00%	95.32% / 0.00%
SP	65.69% / 2.48%	59.00% / 13.62%	58.98% / 13.65%	81.38% / 4.27%	66.93% / 11.19%	93.98% / 0.55%
VMM	42.88% / 29.55%	36.30% / 38.36%	36.45% / 38.40%	44.21% / 33.26%	48.16% / 30.97%	79.67% / 9.27%
PRM	51.61% / 30.53%	37.94% / 46.70%	37.98% / 46.71%	49.37% / 34.76%	47.94% / 37.88%	81.21% / 10.14%
RAND	20.69% / 58.58%	12.43% / 74.35%	12.57% / 74.30%	18.58% / 67.07%	20.89% / 66.31%	58.50% / 28.80%

Table 4: Wins / Losses of X in PTTT

Presumed Payoff Perfect Information Monte Carlo sampling (PPP) works similar to SP, but evaluates perfect information sub-games using EAAB, resulting in a vector of EAM–values associated to each legal action. Comparison of these vectors is done following definition 4. Note that two actions are only evaluated identically if each value $\mathbf{pp}$, $\mathbf{ap}$, and $\mathbf{tp}$ of both actions are identical (we use a small indifference threshold below which values are considered identical to prevent an influence of rounding errors). As knowledge value k we use the proportion of all our previous actions the opponent will know on average if our opponent played random actions. The calculation of this proportion is a bit involved and it should not make a difference using a coarse estimate instead, so we omit the equations used. Fig. 3 shows the evaluation of the first move of PPP playing O placed in the respective field of a Tic-Tac-Toe board. The values are tuples $(\mathbf{pp}, \mathbf{ap}, \mathbf{tp}, k)$ and since the value $\mathbf{pp}$ of the central field is greater than in all other fields, it always picks to play this field.

We also implemented two algorithms of the family of vector-based algorithms to compare this more defensive approach, based on the idea of best defense against clairvoyant play. Prior to applying these algorithms, the game tree has to be evaluated with standard minimax in perfect information sub-games. After this step each leaf node carries a vector of minimax values. Vector-minimax (VMM) is the most basic variant. To propagate the values at the leaf nodes up the game tree, the entire tree is traversed. In MIN nodes the pointwise minimum of the vectors of all child nodes is attached to the node, in MAX nodes one of the vectors with the highest mean value is attached. To decide on which action to take, a child node of the root node is picked, with the highest mean. Payoff Reduction Minimax (PRM) has an additional step between minimaxing over sub-games and the vector minimax step, where payoffs of leaf nodes get reduced depending on the minimax evaluation of their ancestors. This step is introduced to tackle the problem non-locality. For a more detailed description, see (Frank and Basin 2001). Finally, the random agent RAND picks actions uniformly at random.

Table 3 shows the average payoffs of the agents in 2.5 million games for each match. Payoffs are given with respect to the player taking the first action (X, players in the column) with 1 for a win, 0 for a draw, and -1 for a loss. For each value we calculated an approximation of the confidence interval for the proportional value (Agresti–Coull interval) to a confidence level of 98%. We do not give the intervals for each value to prevent confusion. Let us just say that all values are accurate up to ± 0.0008 (or better), with a confidence level of 98%.

With optimal play of both players, player X scores an average of $\frac{2}{3}$ per game. The precalculated CFR strategies are near equilibrium strategies, so not surprisingly CFR playing both, X and O gives an average score near 0.6666 for X. What is more surprising is that CFR playing X does not score any better against neither, PPP nor SP playing O. PPP playing X also scores 0.6666 against CFR, so in game play PPP is absolutely on par with CFR. Looking at PPP vs. PPP and PPP vs. SP we see that both show a weakness in being unable to defend the optimal average playing O. Interestingly PPP playing X is better at exploiting itself than CFR is, and SP playing X is even playing below optimal against PPP. Finally, while the values scored against RAND playing O are nearly equal, RAND playing X gets exploited more heavily by PPP as well as SP than by CFR. Neither VMM, nor PRM are anywhere near a reasonable performance in PTTT match play. PPP, playing X as well as O, is better at exploiting weaknesses of VMM and PRM than CFR.

To get a better insight we compiled a table showing the percent of wins/losses for player X in these matches. Table 4 shows the results. What is immediately obvious is that PPP does not lose a single of its 15M games playing X. While both CFR and PPP playing X score an average of $\frac{2}{3}$ against CFR (O), the strategies leading to this result are clearly different. While CFR wins more of these games (71.82%) it occasionally also loses a match (5.28%), while PPP only

	(σ_1, σ_2)	$u_1(\sigma_1^{\text{BR}}, \sigma_2)$	$u_2(\sigma_1, \sigma_2^{\text{BR}})$
CFR	0.0047	0.6676	-0.6629
PPP	0.3750	1.0000	-0.6250
SP	0.5648	1.0000	-0.4352

Table 5: Exploitability of Agents in PTTT

wins 66.66% of the games but gets a draw out of the remaining games. However, the overall performance of PPP playing X is equal to that of CFR. This is particularly interesting since PPP's X strategy is nearly as unexploitable as CFR's, as we shall see in the following. Let us look in contrary to the results of PPP playing O, against CFR respectively PPP playing X. While it allows both to win around 75%, it only manages to win a few games against CFR (8.16%) while it fails to win a single game against itself. It is the O–part of the strategy produced by PPP that seems vulnerable.

Looking at the results of VMM and PRM playing X it is obvious that both lose a lot of games. Even against a random player they lose around 10% of their games playing X. This is due to the assumption of an opponent that is perfectly informed about the board situation, which makes them concede a lot of games too early.

In summary, PPP is either on par or significantly better than SP in each constellation and it is far ahead of VMM and PRM. PPP is on par with CFR, being even better in exploiting weaknesses with one exception: it gets exploited by itself (PPP playing O) losing 0.7503, while CFR playing O only loses 0.6666 against PPP. These results were very surprising, since we did not expect PPP to work that well compared to CFR. Clearly, the "overconfident" version of a PIMC approach (PPP) outperforms the "paranoid" versions (VMM, PRM).

Another measure for the quality of a strategy is its exploitability. As we have already seen in game play, PPP's O strategy is exploitable. We use *exploitability* to determine how close each strategy is from a Nash equilibrium. If player 1 uses strategy σ_1 and player 2 uses strategy σ_2, then a *best response strategy* for player 1 is $\sigma_1^{\text{BR}} \in \text{BR}(\sigma_2)$, where $\text{BR}(\cdot)$ denotes the set of strategies for which player 1's utility is maximized against the fixed σ_2. Similarly for player 2 versus player 1's strategy. Then, exploitability is defined by

$$(\sigma_1, \sigma_2) = u_1(\sigma_1^{\text{BR}}, \sigma_2) + u_2(\sigma_1, \sigma_2^{\text{BR}}), \tag{5}$$

and when this value is 0 it means neither player has incentive to deviate and $\sigma = (\sigma_1, \sigma_2)$ is a Nash equilibrium. Finally, unlike Poker, in PTTT information sets contain histories of different length; to compute these best response values our implementation uses generalized expectimax best response from (Lanctot 2013b, Appendix B). For the computation we extracted the entire strategies of PPP and SP. Each information set was evaluated with the respective algorithm and the resulting strategy was recorded. The resulting strategy completely resembles the behavior of the respective agent.

Table 5 shows the exploitability, the average payoff of a best response strategy against the O-part of the strategy, and the average payoff of a best response against the X-part. While CFR is quite close to a Nash equilibrium, PPP as well as SP show exploitability. While SP is exploitable in its X as well as its O strategy, PPP is mainly exploitable in its O strategy, resolving the exploitability of the X strategy.

The strategies the PIMC based agents produce are mainly pure strategies. Only in cases of equally rated best actions the probabilities are equally distributed. Looking back at Fig. 3 we see, that the first action of PPP playing O is always the central field. So the strategies are "pseudo–mixed", which contributes to their exploitability. While it is possible to compile truly mixed strategies out of the evaluation of PPP, we did not succeed until now in finding a method that leads to a mixed strategy with a performance comparable to the pseudo–mixed strategies.

This is clearly a downside of a PIMC approach. We still think it is worth considering the method in larger games of imperfect information mainly for two reasons. First, compared to CFR it is extremely fast allowing just-in-time evaluation of actions. Calculating the CFR strategy for PTTT took several days of precalculation. SP as well as PPP evaluate all actions within less than 0.05 seconds on a comparable machine. Second, especially in larger games in the absence of knowledge over the entire strategy of the opponent, finding a best response strategy (i.e. maximally exploiting an opponent) may not be possible within a reasonable amount of games played. We run an online platform[1] for the tricktaking card game Schnapsen backed by an AI agent using PVP (a variant of PPP). No human player, even those who played a few thousand games, is able to play superior to PVP in the long run, with most humans playing significantly inferior. This is clearly not a proof against theoretical exploitability, still it shows that at least human experts in the field fail to exploit it.

Conclusion and Future Work

Despite its known downsides PIMC still is an interesting search techniques for imperfect information games. With modifications of the standard approach it is able to produce reasonable to very good AI agent in many games of imperfect information, without being restricted to games with small state-spaces.

We implemented PPP for heads-up limit Texas Hold'em total bankroll, to get a case study in another field. Unfortunately this game will not be played in the annual computer poker competition (ACPC) in 2016 and we were unable to organize a match up with one of the top agents of ACPC 2014 so far. We are about to implement a PVP backed agent for the no-limit Texas Hold'em competition.

We still are searching for a method to compile a truly mixed strategy out of the evaluations of PPP, resolving or lessening the exploitability of the resulting strategy.

Acknowledgments

We are very thankful for the support of Marc Lanctot, answering questions, pointing us to the references on PTTT, and sharing his implementation of CFR in PTTT. Without his help this work would not have been possible.

[1] http://www.doktorschnaps.at/

References

Auger, D. 2011. Multiple tree for Monte Carlo tree search. In *Applications of Evolutionary Computation*, volume 6624 of *LNCS*. 53–62.

Bosansky, B.; Kiekintveld, C.; Lisy, V.; and Pechoucek, M. 2014. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence* 51:829–866.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit holdem poker is solved. *Science* 347(6218):145–149.

Buro, M.; Long, J. R.; Furtak, T.; and Sturtevant, N. R. 2009. Improving state evaluation, inference, and search in trick-based card games. In Boutilier, C., ed., *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 1407–1413.

Cazenave, T. 2006. A phantom-Go program. In *Advances in Computer Games*, volume 4250 of *LNCS*. 120–125.

Ciancarini, P., and Favini, G. 2010. Monte Carlo tree search in Kriegspiel. *Artificial Intelligence* 174(11):670–684.

Frank, I., and Basin, D. A. 1998a. Optimal play against best defence: Complexity and heuristics. In van den Herik, H. J., and Iida, H., eds., *Computers and Games*, volume 1558 of *Lecture Notes in Computer Science*, 50–73. Springer.

Frank, I., and Basin, D. A. 1998b. Search in games with incomplete information: A case study using bridge card play. *Artif. Intell.* 100(1-2):87–123.

Frank, I., and Basin, D. A. 2001. A theoretical and empirical investigation of search in imperfect information games. *Theor. Comput. Sci.* 252(1-2):217–256.

Frank, I.; Basin, D. A.; and Matsubara, H. 1998. Finding optimal strategies for imperfect information games. In *AAAI/IAAI*, 500–507.

Ginsberg, M. L. 2001. GIB: imperfect information in a computationally challenging game. *J. Artif. Intell. Res. (JAIR)* 14:303–358.

Johanson, M.; Burch, N.; Valenzano, R.; and Bowling, M. 2013. Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 271–278. International Foundation for Autonomous Agents and Multiagent Systems.

Lanctot, M.; Gibson, R.; Burch, N.; and Bowling, M. 2012. No-regret learning in extensive-form games with imperfect recall. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning (ICML 2012)*.

Lanctot, M. 2013a. Counterfactual regret minimization code for Liar's Dice. `http://mlanctot.info/`.

Lanctot, M. 2013b. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games*. Ph.D. Dissertation, University of Alberta, University of Alberta, Computing Science, 116 St. and 85 Ave., Edmonton, Alberta T6G 2R3.

Lisý, V.; Lanctot, M.; and Bowling, M. 2015. Online Monte Carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 27–36.

Lisy, V. 2014. Alternative selection functions for information set Monte Carlo tree search. *Acta Polytechnica: Journal of Advanced Engineering* 54(5):333–340.

Parker, A.; Nau, D. S.; and Subrahmanian, V. S. 2006. Overconfidence or paranoia? search in imperfect-information games. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, 1045–1050. AAAI Press.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach, Second Edition*. Upper Saddle River, NJ: Prentice Hall.

Schofield, M.; Cerexhe, T.; and Thielscher, M. 2013. Lifting hyperplay for general game playing to incomplete-information models. In *Proc. GIGA 2013 Workshop*, 39–45.

Teytaud, F., and Teytaud, O. 2011. Lemmas on partial observation, with application to phantom games. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 243–249.

Wisser, F. 2010. Creating possible worlds using sims tables for the imperfect information card game schnapsen. In *ICTAI (2)*, 7–10. IEEE Computer Society.

Wisser, F. 2013. Error allowing minimax: Getting over indifference. In *ICTAI*, 79–86. IEEE Computer Society.

Wisser, F. 2015. An expert-level card playing agent based on a variant of perfect information monte carlo sampling. In *Proceedings of the 24th International Conference on Artificial Intelligence*, 125–131. AAAI Press.

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 (NIPS 2007)*.