Perfect Information Monte Carlo with Postponing Reasoning

Jérôme Arjonilla
Université Paris Dauphine - PSL
Paris, France
jerome.arjonilla@hotmail.fr

Abdallah Saffidine
University of New South Wales
Sydney, Australia
abdallah.saffidine@gmail.com

Tristan Cazenave
Université Paris Dauphine - PSL
Paris, France
tristan.cazenave@dauphine.psl.eu

ABSTRACT

Imperfect information games, such as Bridge and Skat, present challenges due to state-space explosion and hidden information, posing formidable obstacles for search algorithms. Determinization-based algorithms offer a resolution by sampling hidden information and solving the game in a perfect information setting, facilitating rapid and effective action estimation. However, transitioning to perfect information introduces challenges, notably one called strategy fusion. This research introduces 'Extended Perfect Information Monte Carlo' (EPIMC), an online algorithm inspired by the state-of-the-art determinization-based approach Perfect Information Monte Carlo (PIMC). EPIMC enhances the capabilities of PIMC by postponing the perfect information resolution, reducing alleviating issues related to strategy fusion. However, the decision to postpone the leaf evaluator introduces novel considerations, such as the interplay between prior levels of reasoning and the newly deferred resolution. In our empirical analysis, we investigate the performance of EPIMC across a range of games, with a particular focus on those characterized by varying degrees of strategy fusion. Our results demonstrate notable performance enhancements, particularly in games where strategy fusion significantly impacts gameplay. Furthermore, our research contributes to the theoretical foundation of determinization-based algorithms addressing challenges associated with strategy fusion, thereby enhancing our understanding of these algorithms within the context of imperfect information game scenarios.

KEYWORDS

Search algorithm, Games, Imperfect Information, Heuristic Algorithm, Determinization, Strategy Fusion

1 INTRODUCTION

Search algorithms in artificial intelligence have significantly evolved, demonstrating superhuman performance in games such as Chess, Go [25], Poker [4], Skat [10], and Contract Bridge[2]. Perfect information games, like Chess and Go, where all information is available, have been extensively studied, allowing algorithms to surpass human professionals [25–27]. In contrast, imperfect information games, including Poker, Skat, and Bridge, where some information is hidden, have been less studied, with only a few algorithms capable of outperforming professional human players [4].

In imperfect information games, two commonly used search methods are regret-based approaches, which excel in Poker and are theoretically convergent but slower [15, 19, 28], and determinization-based methods, considered state-of-the-art in various trick-taking card games, offering scalability but lacking theoretical guarantees [7, 8, 10, 17]. In recent years, both methods have incorporated neural networks to enhance performance and facilitate scalability on large games [4, 11, 18, 23, 30].

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Milec, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.

Determinization-based algorithms operate by *sampling* hidden information based on current knowledge and utilizing a *perfect information leaf evaluator* to predict game outcomes under perfect information assumptions. These algorithms achieve state-of-the-art performance, primarily due to their efficient utilization in the perfect information perspective, which is inherently simpler than solving the problem from an imperfect information standpoint. Despite their state-of-the-art performance, determinization-based algorithms face challenges, notably encountering 'strategy fusion' [9, 17]. This challenge arises from the use of the perfect information leaf evaluator, which independently solves each possible world without considering the uncertainties induced by games with imperfect information.

Within determinization-based algorithms, Perfect Information Monte Carlo (PIMC) [17] is particularly susceptible to strategy fusion due to its early usage on the perfect information leaf evaluator in decision-making. In our study, we address this challenge by postponing the leaf evaluator until a depth d, alleviating the impact of strategy fusion. The act of postponing the perfect information leaf evaluator at a depth of d introduces new considerations, specifically, it prompts the need for alternative strategies to reason from step 1 to step d. We enhance our research by formally defining the problem of strategy fusion and showing that in the worst case, increasing the depth d does not increase the strategy fusion, and in every case, there exists a depth d such that the strategy fusion is strictly reduced. More than that, for finite games, there exists a depth d such that reasoning up to it allows the removal of the problem of strategy fusion.

Section 2 delves into notation and provides a detailed explanation of determinization-based algorithms, specifically addressing PIMC and the challenge of strategy fusion. Section 3 introduces Extended PIMC, an algorithm that takes into account our idea of postponing the leaf evaluator at depth d. Our implementation operates online and without a significant initial cost, ensuring smooth deployment across diverse games or within the broader scope of General Game Playing [24]. It is noteworthy that, like other modern determinization algorithms, there exists the potential to integrate neural networks for enhanced performance. Section 4 presents the theoretical results obtained when increasing the depth d on determinisation-based algorithms. Section 5 presents experimental findings across various games, some with a minimal impact on the strategy fusion problem and others with a more pronounced effect. Our observations indicate that, particularly in cases with significant strategy fusion, deeper reasoning enhances performance, surpassing other state-of-the-art methods. Section 6 provides an overview of related research and finaly, Section 7 summarizes our contributions and outlines avenues for future research.

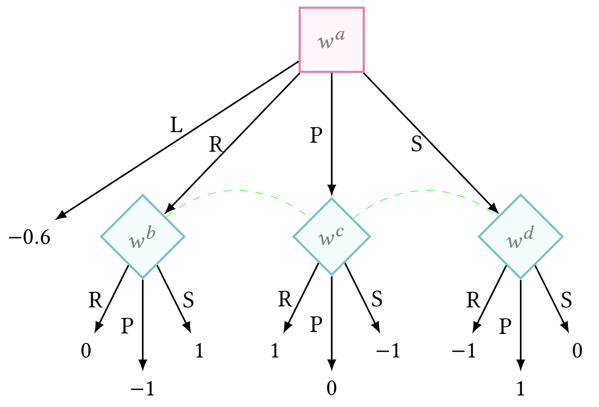


Figure 1: Variant of 'Rock-Paper-Scissors'. The red square is the first player, the green diamond is the second player and the dashed line represents worlds indistinguishable by the second player.

2 NOTATION AND BACKGROUND

2.1 Notation

We provide an overview of foundational concepts rooted in the formalism of Factored-Observation Stochastic Games (FOSG)[13]. Throughout the paper, we utilize Figure1, a variant of 'Rock-Paper-Scissors', to enhance comprehension and present notations. Our notation employs subscripts to denote players.

A game, denoted as G, involves $\mathcal{N}$ players, initiating at w^{init} and progressing through successor world states represented by $w \in \mathcal{W}$. Players make joint actions, denoted as $a = (a_1, \ldots, a_{\mathcal{N}}) \in \mathcal{A}(w)$, in each world state w, and the game continues until a terminal state is reached. Upon choosing a joint action a, players observe rewards, and the next world state w' is determined probabilistically. During this transition from w to w', each player receives an observation denoted as $o_i \in O_i(w, a, w')$, where $O_i(w, a, w')$ represents the set of possible observations for player i.

In Figure 1, the first player faces the choice between 'Leave' with a reward of -0.6 or engaging in the game. If it opts to play, the standard rules apply: Rock beats Scissors, Scissors beats Paper, and Paper beats Rock, with wins yielding 1, losses resulting in -1, and ties providing 0. The game encompasses four possible world states denoted as $\mathcal{W} = w^a, w^b, w^c, w^d$, where w^a is the initial state and w^b, w^c, and w^d follow the first action.

The first player has four possible actions in w^a and a null action in other states—specifically, $\mathcal{A}_1(w^a)$ ={Leave, Rock, Paper, Scissors} and $\mathcal{A}_1(w^b) = \mathcal{A}_1(w^c) = \mathcal{A}_1(w^d) =$ noop. The second player has a null action in w^a and three actions in other states—namely, $\mathcal{A}_2(w^a) = \emptyset$ and $\mathcal{A}_2(w^b) = \mathcal{A}_2(w^c) = \mathcal{A}_2(w^d) =$ Rock, Paper, Scissors. In this game, the second player receives the observation Play if the first player plays Paper, Scissors, or Rock. Rewards for both players are obtained at the game's conclusion.

A *history* is a finite sequence of world states and legal actions denoted as $h^t = (w^0, a^0, w^1, a^1, \ldots, w^t)$. An *infostate* s_i is a sequence of an agent's observations and actions, denoted as $s_i^t = (a_i^0, o_i^1, a_i^1, \ldots, o_i^{t-1})$. For each history, there exists a unique infostate noted as $s_i(h)$, and for each infostate, there is a set of histories that match the sequence denoted as $\mathcal{H}(s)$. $\mathcal{I}(G)$ represents the set of possible infostates in the game G, and $\Pi(G)$ is the set of possible policies in the game G, where a *policy* π is a function mapping every history h to a probability distribution over actions.

In Figure 1, if we examine the history $h^1 = (w^a, \text{Rock}, w^b)$, considering it from the second player's perspective, the infostate becomes $s_2^1 = (\text{noop}, \text{Play})$ since no action was taken by the second player, and the observation Play was noted. The set of histories that correspond to s_2^1 includes $\{(w^a, \text{Rock}, w^b); (w^a, \text{Paper}, w^c); (w^a, \text{Scissors}, w^d)\}$.

2.2 Determinization-based algorithm

Each determinization-based algorithm has its own characteristics, nevertheless, they share some common features such as (i) *samples* a history h according to a probability distribution over the current infostate s; (ii) uses a *perfect information leaf evaluator* for estimating the value of the sampled world state. In the description of the Algorithm 1 and 2 (i) is noted Sampling(s) and (ii) is noted $PerfectAlgo(h)$.

A perfect information leaf evaluator is an algorithm used in games with perfect information to estimate the value of a history h. It predicts the outcome of a game from a specific position. This evaluator can be exhaustive methods like Minimax with Alpha-Beta pruning [12], heuristics methods like Random Rollout (also called *playouts* [5]) or neural network [25–27].

Determinization-based algorithms are simple and, in practice, achieve great results. Yet, certain problems are encountered such as (i) non-locality and strategy-fusion [9, 17]; (ii) revealing private hidden information [1, 9]; (iii) no theoretical guarantees.

In Figure 1, when conducting sampling from the infostate s_2^1, the available options include sampling w^b, w^c, or w^d. Utilizing a perfect information leaf evaluator, such as Minimax, on the world state w^b would yield a value of -1, as the second player optimally plays Paper to maximize their score *i.e.*, minimises the value of the first player.

2.3 Strategy fusion

In imperfect information games, histories stemming from the same infostate must be approached with the same strategy, as players cannot distinguish between them. Formally, $\forall s \in \mathcal{I}(G), \forall h, h' \in \mathcal{H}(s), \pi(\cdot|h) = \pi(\cdot|h')$. However, determinization-based algorithms deviate from this principle. They employ a perfect information leaf evaluator algorithm to estimate the value at each sampled history. In other words, each history originating from the same infostate is solved using a strategy tailored to that specific history. Formally, $\forall s \in \mathcal{I}(G), \forall h, h' \in \mathcal{H}(s), \pi(\cdot|h) \neq \pi(\cdot|h')$.

In Figure 1, the strategies in w^b, w^c, and w^d must be identical since they originate from the same infostate s_2^1. For the second player, the optimal strategy results in an average score of 0 (playing with the same probability for all three actions). By back-propagating, the first player opts for Play, leading to an average score of 0, instead of choosing Leave, which results in -0.6.

From the perspective of a determinization-based algorithm, a policy is tailored for the sampled history. The best policy for the second player is Paper in w^b, Scissors in w^c, and Rock in w^d. In w^b, w^c, and w^d, playing the best policy yields -1. By back-propagating, the first player concludes that opting for Leave to obtain -0.6 is preferable compared to playing and receiving -1.

2.4 Perfect Information Monte Carlo

Perfect Information Monte Carlo (PIMC) is a determinization-based algorithm that is the state-of-the-art of many imperfect information games.

Algorithm 1: PIMC

```
Function PIMC($s$):
    for $a \in \mathcal{A}(s)$ do
        $score[a] \leftarrow 0$;
    end
    while budget do
        $w \leftarrow$ Sampling($s$);
        for $a \in \mathcal{A}(w)$ do
            $w' \leftarrow w.ApplyMove(a)$;
            $score[a] \leftarrow score[a] + PerfectAlgo(w')$;
        end
    end
    return Returns the best action on average;
```

PIMC is defined in Algorithm 1 and works as follows (i) samples a world state w according to the probability distribution over the current infostate s; (ii) plays an action a on the world state w, and observes the next world state w'; (iii) estimates the world state w' by using the perfect information leaf evaluator; (iv) repeats until the *budget* is over; (v) selects the action that produces the best results in average.

3 EXTENDED PIMC

As presented, employing the perfect information leaf evaluator results in strategy fusion. In the case of PIMC, this evaluator is utilized after the first action is played. However, there are no inherent constraints preventing the resolution after the first action, and it is not difficult to believe that postponing the use of the perfect information leaf evaluator could mitigate the issue of strategy fusion.

In the following, we introduce a novel algorithm that embraces this straightforward concept of postponing the leaf evaluator's utilization. The algorithm, termed 'EPIMC' (Extended Perfect Information Monte Carlo), extends the PIMC paradigm by incorporating Extended reasoning at a depth of d, where the instance of $d = 1$ corresponds to PIMC.

Algorithm 2: Extended PIMC

```
Function ExtendedPIMC($d$, $s$):
    Create the game $U$ and $u$, the initial world state;
    while budget do
        $w \leftarrow$ Sampling($s$);
        Query($U$,$u$, $w$, $d$);
    end
    return ImperfectAlgo($U$);

Function Query($U$, $u$, $w$, $d$):
    if $d == 0$ or $w$.IsTerminal() then
        $u$.value $\leftarrow$ $u$.value + PerfectAlgo($w$);
        return;
    end
    $w' \leftarrow w.ApplyMove$(RandomAction($w$));
    Get $u'$ associated with $w'$;
    Update dynamics in $U$ according to $u'$ and $u$;
    Query($U$, $u'$, $w'$, $d - 1$);
```

The pseudo-code is available in Algorithm 2 and works as follows (i) creates a subgame game U; (ii) samples a world state w according to the probability distribution over the current infostate s; (iii) plays an action a on w, observes the next world state w' continue until w' is a terminal node or the depth d is obtained, and updates the dynamic in subgame U; (iv) estimates the world state w' by using the perfect information leaf evaluator; (v) repeats phases 2 to 4 until the *budget* is over; (vi) solves the subgame U which have been created during phase 2 to 5 by using an algorithm that does not create strategy fusion (*i.e.* an algorithm that do not use a perfect leaf evaluator to resolve the subgame U). In the following, a few points are clarified.

Exploration Strategy

In the original PIMC, world state evaluations occur a maximum of $|A|$ times during each sampling phase, with $|A|$ denoting the highest feasible action count. However, applying the same approach at depth d could potentially lead to evaluating $|A|^d$ world states. Therefore, careful consideration of the number of actions per step becomes imperative. In EPIMC, our chosen strategy is to explore one action per sampling iteration, resulting in a singular world state evaluation per sampling instance, however, other choices could have been envisaged.

Depth

While theoretically, any depth d could be employed, practical considerations necessitate a prudent choice of d. Selecting a depth d that is too small can exacerbate strategy fusion issues, as observed in approaches like PIMC. On the other hand, opting for a depth d that is too large demands significant sampling efforts to accurately estimate the subgame U. Moreover, depending on the algorithm employed to solve the subgame U, a larger depth can lead to increased computational costs.

Subgame Resolution

By postponing the application of the perfect information leaf evaluator until a depth of d, alternative reasoning methods must be employed for steps 1 through d. In EPIMC, the subgame U of size d is built to approximate the real game by encapsulating various elements, including infostates, world states, post-action dynamics, and more. In particular, the leaves of the subgame U are average scores obtained from the leaf evaluator.

After the budget is finished, the subgame U is solved using an algorithm that does not create strategy fusion. In other words, one needs an algorithm that works on infostates instead of world states. This choice of algorithm, although potentially resource-intensive, carries a reduced computational burden when applied to a subgame U of size d. One can think of using information set search [20, 21] which operates on infostates according to a minimax rule or CFR/CFR+ [19, 28] that have theoretical guarantees for two players.

4 THEORETICAL FOUNDATION

In the following, we present the theoretical foundation for determinization based algorithms that suffer from strategy fusion. We formally (i) define the condition to *create* strategy fusion; (ii) define the quantity of strategy fusion; (iii) prove that, in the worst case, increasing the depth does not increase the strategy fusion, and in every case, there exists a depth d such that the strategy fusion is strictly reduced and in a finite game, there exists a depth d such as the that the strategy fusion is removed.

Definition 1. *For any game G, a policy $\pi \in \Pi(G)$ and an infostate $s \in \mathcal{I}(G)$* ***create****s strategy fusion if there are $h, h' \in \mathcal{H}(s)$ such that $\pi(h) \neq \pi(h')$. For any game G, a policy $\pi \in \Pi(G)$* ***create****s strategy fusion if there is an infostate $s \in \mathcal{I}(G)$ such that π creates strategy fusion in s.*

To evaluate the quantity of strategy fusion in $\pi \in \Pi(G)$, we propose the following measure $SF(\pi) = |\{s$ such that $\forall s \in S(G)$, π *create*s strategy fusion in s $\}|$. In other, we count the number of infostate that create strategy fusion. $SF(\pi) = 0$ implies that there is no strategy fusion.

In the subsequent discussions, for the sake of simplicity, we presume adherence to the policy provided by EPIMC, denoted as $\Pi^{E^d}(G)$ when executing EPIMC at a depth of d. Although alternative choices, could have been considered, opting for the EPIMC policy is more straightforward, as influenced by the uniform growth of the EPIMC policy across the entire depth d space.

Proposition 1. *$\forall G, \forall d \in [0, T-1]$ where $T = \{T$ if G has a finite horizon T; else $\infty\}$,$\forall \pi \in \Pi^{E^d}(G)$, then $\forall \pi' \in \Pi^{E^{d+1}}(G)$, $SF(\pi, G) \geq SF(\pi', G)$*

Proof. For any $s \in \mathcal{I}(G)$, s does not create strategy fusion in $\pi^{1:d}$. Increasing the depth by 1, extends the non-inducing region. Two possibilities arise: (i) if at least one s at $d+1$ create strategy fusion, it can no longer do so, i.e., $SF(\pi, G) > SF(\pi', G)$; (ii) if all s at $d+1$ do not create strategy fusion, increasing the depth does not reduce SF, i.e., $SF(\pi, G) \geq SF(\pi', G)$. We conclude $SF(\pi, G) \geq SF(\pi', G)$. □

Proposition 2. *$\forall G, \forall d \in [0, T]$ where $T = \{T$ if G has a finite horizon T; else $\infty\}$,$\forall \pi \in \Pi^{E^d}(G)$, if $SF(\pi, G) > 0$, then $\exists d' \in [1, T-d], \forall \pi' \in \Pi^{E^{d+d'}}(G)$ such that $SF(\pi, G) > SF(\pi', G)$.*

Proof. Leveraging the rationale from Proposition 1, increasing the depth by d' extends the non-inducing region. Given the presence of strategy fusion ($SF(\pi, G) > 0$), at least one infostate creates strategy fusion. Extending the reasoning up to this infostate, located d' away from the original depth d, effectively diminishes strategy fusion. Hence, we establish $SF(\pi, G) > SF(\pi', G)$. □

We extend Proposition 2 by showing that, in a finite game, there is a depth at which there is no longer strategy fusion.

Proposition 3. *$\forall G$ such that G has a finite horizon T, $\forall d \in [1, T-1], \forall \pi \in \Pi^{E^d} G$ if $SF(\pi, G) > 0$ then $\exists d' \in [1, T-d], \forall \pi' \in \Pi^{E^{d+d'}} G$ such that $SF(\pi', G) = 0$.*

Proof. Setting $d' = T - d$, i.e., until the end of the game, ensures that no further infostate can induce strategy fusion, as there are no infostates remaining. □

5 RESULTS

5.1 Games

Our experiment set involved testing five games: Card Game, Battleship, Dark Chess, Phantom Tic-Tac-Toe, and Dark Hex. Each of them is considered a large game, is described below and is implemented in OpenSpiel [14], a collection of environments and algorithms for research in general reinforcement learning and search/planning in games. The benchmarks were chosen to show the strengths and weaknesses of the algorithm, especially, by choosing games with public observations (Card game, Battleship) or with private observations (Phantom Tic-Tac-Toe, Dark Chess, and Dark Hex).

Public observations refer to information that is visible to all players, while private observations are exclusive to a player. This distinction significantly influences the dynamics of strategy fusion. Private observations amplify the number of potential world states within a single infostate, thereby increasing the likelihood of strategy fusion. As our method reduce strategy fusion, we expect EPIMC to be more effective in games where observations are private.

Card game. The game is played with two players, 22 cards are taken from a pack of 52, and known by all, 6 are hidden and 8 is given to each player. The playing phase is decomposed into tricks, the player starting the trick is the one who won the previous trick. The starting player of a trick can play any card in his hand, but the other player must follow the suit of the first player. If he can not, he can play any card he wants without possibly winning the trick. The winner of the trick is the one with the highest-ranking card. At the end of the game, a player wins if he has at least half of the number of tricks won.

Battleship. Battleship is a two-player strategy-type guessing game. Each player possesses a grid. In the beginning, each player secretly places a set of ships S on their grid. After placement, turn after turn, each player tries to fire at other players' ships. The game ends when all the ships of a player have been destroyed. The payoff of each player is computed as the sum of the opponent's ships that were destroyed, minus the sum of ships that the player lost. The grid is fixed to 3×3, with 2 ships, one of size 1×1, and the second of size 2×1.

Dark Chess. Dark chess is a chess variant with incomplete information. In chess, there are two players, white and black, each controlling a set of chess pieces of their respective colors. The goal of the game is to checkmate the opponent's king. In Dark chess, the incomplete information comes from the fact that each player sees his own pieces, but only sees his opponent's pieces if they are reachable by one of his pieces. Furthermore, in his variant, the purpose is to capture the king (not to checkmate it), however, a player must be wary as he is not told if their king is in check. The size of the board is fixed to 4×4.

Phantom Tic-Tac-Toe. Phantom Tic-Tac-Toe is a variant of the game of Tic-Tac-Toe with imperfect information. In Tic-Tac-Toe, the goal is to claim three cells along the same row, column, or diagonal. With imperfect information, the players do not observe the other player's pieces, only a referee knows the world state of the board. When it is a player's turn, the player selects a move and indicates it to the referee. The referee informs the player's whether the action is 'legal' or 'illegal'. If the move is 'illegal', the player must choose a new move until they find a legal one.

Dark Hex. Dark Hex is an imperfect information version of the classic game of Hex. The objective of the game is to create a connection between opposite sides of a rhombus-shaped board. In Dark Hex, players are not exposed to opposite sides pieces of information. Only a referee has the full information of the board and when a move fails due to collision/rejection the player gets some information of the cell and is allowed to make another move until success. The size of the board is fixed to 4×4.

5.2 Experimental Information

For all the experiments, the budget ranged from 0.1 seconds to 100 seconds for Card Game, Battleship and Phantom Tic-Tac-Toe, and from 0.1 seconds to 1000 seconds for Dark Hex and Dark Chess. Each experiment was conducted over 500 games, in which the games were evenly split between playing in the first and second positions. The opponent is PIMC with a fixed one-second budget.

All experiments were executed on a single CPU Intel(R) Xeon(R) Gold 5218. EPIMC and PIMC uses random rollout as the perfect information leaf evaluator, depth at 3 and Information Set Search for the subgame resolution. In practice, PIMC is often tested with minimax as the perfect information leaf evaluator, yet using it may be slow in large benchmarks or require using a handmade heuristic. As a reminder, EPIMC at depth 1 is PIMC. To reduce EPIMC/PIMC costs, multiple CPUs could be utilized, but for fairness across algorithms, this approach was not employed.

Other online algorithms. The others algorithms compared are Information Set MCTS (IS-MCTS) [8], Online Outcome Sampling (OOS) [16], Recursive PIMC (IIMC) [10], and a random agent (Random). IS-MCTS is a determinization-based algorithm that employs Monte Carlo Tree Search on infostates. OOS is a regret-based algorithm that converges to the Nash equilibrium with increasing search time. IIMC is a determinization-based algorithm rooted in PIMC, estimating action values by recursively calling PIMC until game completion.

For IS-MCTS, the exploration constant was chosen from 0.6, 1, 1.5, 2 and set at 1. In OOS, the target was selected between Information Set and Public Subgame Targeting, and it was set at Information Set. Regarding IIMC, the number of samplings at level 2 was chosen from 2, 5, 10 and set at 5.

5.3 Experimental Results

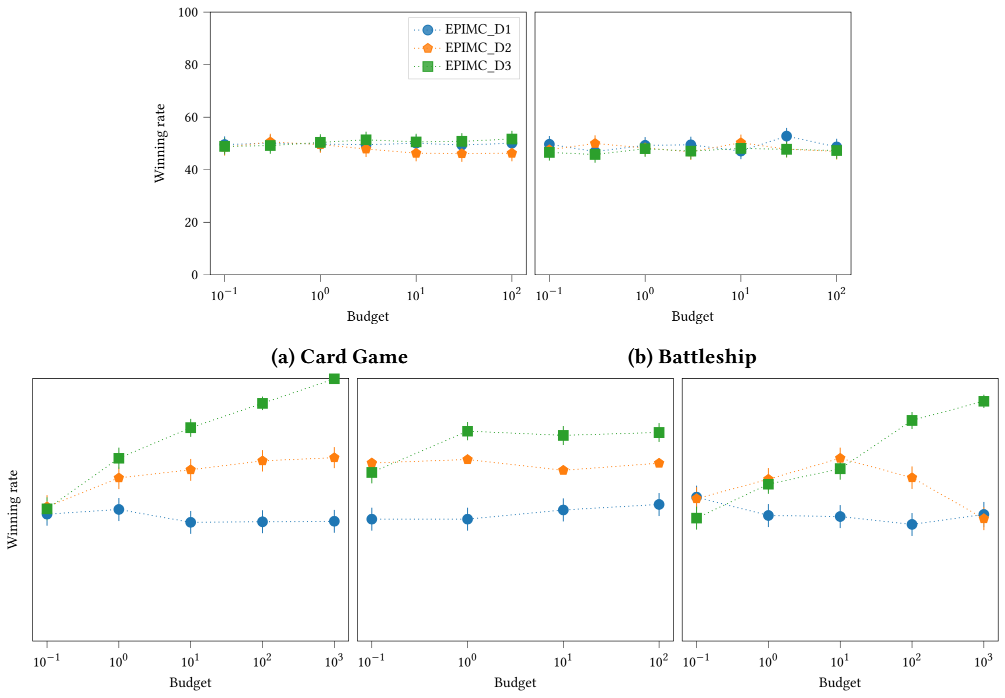
In our experiments, we aimed (i) to analyze the hyperparameters of EPIMC (depth, subgame resolution, perfect leaf evaluator) and (ii) to compare its performance against other online algorithms.

Postponing leaf evaluator. In Figure 2, we examine the performance of EPIMC according to various depths variyng from 1 to 3. As expected, for Card Game and Battleship, increasing the depth does not lead to any improvement as both games have a majority of their observation public. Conversely, for games where the observations are private, we observe an important increase in performance At 100 seconds for Dark Chess, we achieve 80%/65%/45% winning rates at depths 3/2/1. More than that, at 1000 seconds, EPIMC at depth 3 wins close to 100% for both Dark Hex and Dark Chess. Interestingly, in Dark Hex, at 1000 seconds, similar performance are observed at depth 2 and 1. Indeed reducing the strategy fusion does not necessarily mean that the strategy produced at the end will be changed.

Extended resolution. In Figure 3, we compare CFR+ agasint Information Set Search (ISS) in the Extended game resolution. CFR+ is used with 1000 iterations. Using CFR+ in Dark Hex 4×4 was too costly in computational time, and use it on Dark Hex 3×3. As can be observed, neither method is superior to the other, worse performances are obtained with CFR+ in Dark Chess and Phantom Tic-Tac-Toe but better performance is achieved in Dark Hex.

Perfect information leaf evaluator. In Figure 4, we compare Minimax with alpha-beta pruning against random rollout for the perfect information leaf evaluator. Due to the important cost of using Minimax, the test was not conducted in Dark Chess, and as before, the size Dark Hex was reduced to 3×3. As before, neither method is superior to the other, as worse performance is obtained with Minimax in Phantom Tic-Tac-Toe but stronger performance is achieved in Dark Hex. However, the differences between the two methods are important in Dark Hex, where Minimax obtains performance close to 70% while Random Rollout obtains performance close to 50%.

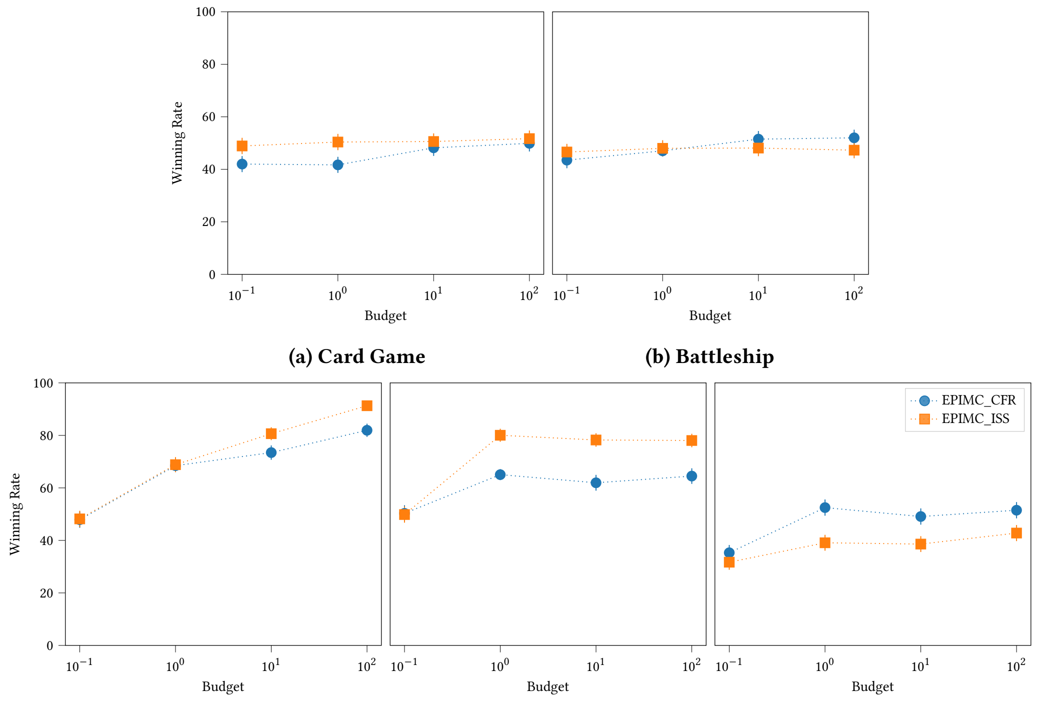
Against other online algorithms. In Figure 5, a comparative analysis between EPIMC and various online algorithms against opponents is presented. It is evident from the results that EPIMC and IS-MCTS exhibit superior performance compared to other algorithms. Notably, EPIMC achieved remarkable success, particularly at depth 3, outperforming IS-MCTS across all benchmarks considered. Furthermore, it is noteworthy that even at a depth of 2, EPIMC demonstrated comparable or even superior performance to IS-MCTS in games like Dark Chess and Phantom Tic-Tac-Toe.



(a) Card Game **(b) Battleship**

(c) Dark Chess **(d) Phantom Tic-Tac-Toe** **(e) Dark Hex**

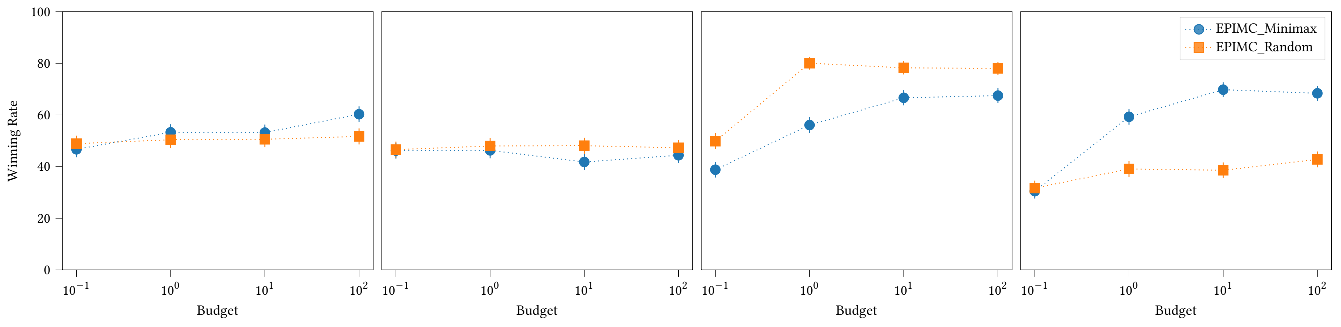
Figure 2: Winning rate of EPIMC when the depth range between one to three. The opponent is PIMC with one second of budget.



(a) Card Game **(b) Battleship**

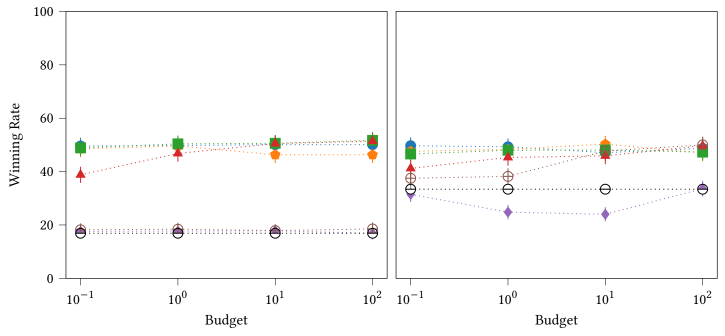
(c) Dark Chess **(d) Phantom Tic-Tac-Toe** **(e) Dark Hex**

Figure 3: Winning rate of EPIMC when the subgame is CFR+ or ISS. The opponent is PIMC with one second of budget.

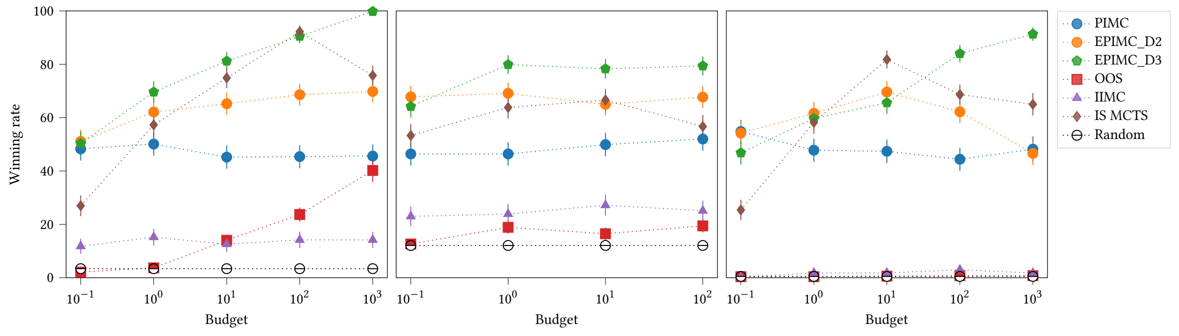


(a) Card Game (b) Battleship (c) Phantom Tic-Tac-Toe (d) Dark Hex

Figure 4: Winning rate of EPIMC when the leaf evaluator is Minimax or Random Rollout. The opponent is PIMC with one second of budget.



(a) Card Game (b) Battleship



(c) Dark Chess (d) Phantom Tic-Tac-Toe (e) Dark Hex

Figure 5: Winning rate of multiple online algorithms. The opponent is PIMC with one second of budget.

5.4 Discussion

As we have seen and as was expected, increasing the depth has a significant impact on games with a private observation (Dark Hex / Dark Chess / Phantom Tic-Tac-Toe) and much less impact on games without much private information (Card Game / Battleship). On the domains tested and given a sufficient budget, augmenting the depth consistently outperforms the basic version. In addition, a depth of 2 is sufficient, in most cases, to beat the state-of-the-art online algorithms. With regard to the other hyperparameters, we recommend using Information Set Search and Random Rollout as they require less computing time and therefore can be computed on larger games. However, customization could be beneficial in order to increase performance or the need to maintain theoretical properties in the subgame. Especially, using algorithms such as CFR+ allows to obtain the properties of CFR+ (in the subgame) such as the convergence towards the Nash equilibrium in two players.

6 RELATED WORK

In determinization-based algorithms, both IIMC [10] and IS-MCTS [8] have adopted a similar approach of extending reasoning beyond a depth of one. In IS-MCTS, the depth increases with the budget, and in IIMC, they recursively call PIMC until the end of the game. In both cases, as the depth exceeds PIMC, it is expected that strategy fusion will be reduced. However, there is no guarantee that it can be entirely eliminated, and it is challenging to quantify the extent of its reduction. This is because the two methods do not uniformly explore the state space, and worse, they use exploration/exploitation mechanisms based on estimates that are distorted by the presence of strategy fusion.

At a high-level, PIMC and our work have similarities to Unsafe/Safe Subgame Solving [6]. However, there are a number of differences, the most important of which is the fact that our algorithm does not require an expensive preparation of a value function in advance of playing a game. In fact, to function most of Unsafe/Safe algorithms need to, first hand, solve an abstraction of the game that contains the subgame constructed with the leaf estimation, and during the play, the algorithm resolves the portion of the game reached to a greater degree of accuracy than in the initial computation. What is more, in their method, there is rarely any description of how the subgame is constructed during the game (for example in Rebel [3], a function is called 'ConstructSubgame' without explaining how this function work), contrary to the determinization method that build the subgame by using sampling and fast leaf evaluator.

In a study by Zhang and Sandholm [29], a similar approach to ours was taken, utilizing a perfect leaf evaluator at an extended depth compared to PIMC. However, their research focused primarily on the concept of 'common knowledge', utilizing this version were mainly to reduce costs without the need for abstraction. Consequently, they did not delve into the intricacies of determination algorithms, nor did they explore various aspects central to our investigation, such as sub-game creation, budget considerations, influence of depth on strategy fusion, influence of private/public observations. Additionally, they employed move ordering, prioritizing the exploration of promising nodes over uniform exploration of the state space, which akin to algorithms like IS-MCTS, may lead to issues related to strategy fusion and suboptimal decision-making, which are critical aspects discuted in our study.

7 CONCLUSION AND FUTURE WORKS

In this paper, we introduce a novel online algorithm 'Extended PIMC' for games with imperfect information. Building upon the foundation of PIMC, our approach postpone the perfect leaft evaluator to a deeper depth. Thanks to that, we have been able to successfully reduce past problems of PIMC and beat other online algorithms on multiple benchmarks. Especially, when benchmarks have hidden observation, significant performance improvements are observed. Furthermore, we conducted an in-depth analysis of various hyperparameters to provide a comprehensive understanding of their impact.

We enhance our research by presenting theoretical foundations for determinization-based algorithms that suffer from strategy fusion. We demonstrate that, in the worst case, increasing the depth does not increase the strategy fusion and in every case, there exists a depth d such that the strategy fusion is strictly reduced.

As our algorithm is online, it has the advantage of being tested in a short period of time, especially in comparison to recent algorithms which need a domain-specific abstraction or a very high initialization cost due to neural networks. Even though, improving our algorithm by using deep learning could have led to superior performance. In particular, this could provide a better and faster approximation to the leaf or being able to remove the problem of non-locality by adding an inference system [22].

In future research, it would be intriguing to assess our algorithm in conjunction with other determination algorithms. Particularly, exploring a trade-off between our approach, which ensures non-strategy fusion at a certain depth thanks to uniform sampling, and other algorithms that may explore the state space more efficiently but currently encounter issues with strategy fusion during exploration, could be beneficial. Such an investigation could shed light on the strengths and limitations of different approaches and potentially lead to the development of more robust and efficient algorithms for imperfect information games.

REFERENCES

[1] Jérôme Arjonilla, Abdallah Saffidine, and Tristan Cazenave. 2023. Mixture of Public and Private Distributions in Imperfect Information Games. In *2023 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[2] Bruno Bouzy, Alexis Rimbaud, and Véronique Ventos. 2020. Recursive Monte Carlo Search for Bridge Card Play. *2020 IEEE Conference on Games (CoG)* (2020), 229–236.

[3] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. 2020. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA. event-place: Vancouver, BC, Canada.

[4] Noam Brown and Tuomas Sandholm. 2019. Superhuman AI for multiplayer poker. *Science* 365 (2019), 885 – 890.

[5] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4 (2012), 1–43.

[6] Neil Burch, Michael Bradley Johanson, and Michael Bowling. 2014. Solving Imperfect Information Games Using Decomposition. In *AAAI*.

[7] Tristan Cazenave and Véronique Ventos. 2021. The $\alpha\mu$ Search Algorithm for the Game of Bridge. In *Monte Carlo Search at IJCAI (Communications in Computer and Information Science)*.

[8] Peter I. Cowling, Edward Jack Powley, and Daniel Whitehouse. 2012. Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games* 4 (2012), 120–143.

[9] Ian Frank and David A. Basin. 1998. Search in Games with Incomplete Information: A Case Study Using Bridge Card Play. *Artif. Intell.* 100 (1998), 87–123.

[10] Timothy Furtak and Michael Buro. 2013. Recursive Monte Carlo search for imperfect information games. *2013 IEEE Conference on Computational Inteligence in Games (CIG)* (2013), 1–8.

[11] Qiqi Jiang, Kuangzheng Li, Boyao Du, Hao Chen, and Hai Fang. 2019. DeltaDou: Expert-level Doudizhu AI through Self-play. In *IJCAI*.

[12] Donald E. Knuth and Ronald W. Moore. 1975. An analysis of alpha-beta pruning. *Artificial Intelligence* 6, 4 (1975), 293–326. https://doi.org/10.1016/0004-3702(75)90019-3

[13] Vojtěch Kovařík, Martin Schmid, Neil Burch, Michael H. Bowling, and V. Lisý. 2022. Rethinking Formal Models of Partially Observable Multiagent Decision Making. *Artif. Intell.* 303 (2022), 103645.

[14] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinícius Flores Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury,

David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas W. Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. 2019. OpenSpiel: A Framework for Reinforcement Learning in Games. *ArXiv* abs/1908.09453 (2019).

[15] Marc Lanctot, K. Waugh, Martin A. Zinkevich, and Michael Bowling. 2009. Monte Carlo Sampling for Regret Minimization in Extensive Games. In *NIPS*.

[16] V. Lisý, Marc Lanctot, and Michael Bowling. 2015. Online Monte Carlo Counterfactual Regret Minimization for Search in Imperfect Information Games. In *AAMAS*.

[17] Jeffrey Richard Long, Nathan R. Sturtevant, Michael Buro, and Timothy Furtak. 2010. Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search. In *AAAI*.

[18] Matej Moravcík, Martin Schmid, Neil Burch, V. Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, K. Waugh, Michael Bradley Johanson, and Michael H. Bowling. 2017. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* 356 (2017), 508 – 513.

[19] Todd W Neller and Marc Lanctot. 2013. An introduction to counterfactual regret minimization. In *Proceedings of model AI assignments, the fourth symposium on educational advances in artificial intelligence (EAAI-2013)*, Vol. 11.

[20] Austin Parker, Dana Nau, and VS Subrahmanian. 2006. Overconfidence or paranoia? search in imperfect-information games. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, Vol. 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 1045.

[21] Austin Parker, Dana Nau, and VS Subrahmanian. 2006. Paranoia versus overconfidence in imperfect information games. In *AAAI Proceedings of the 21st National Conference on Artificial intelligence*, Vol. 2.

[22] Douglas Rebstock, Christopher Solinas, Michael Buro, and Nathan R. Sturtevant. 2019. Policy Based Inference in Trick-Taking Card Games. *2019 IEEE Conference on Games (CoG)* (2019), 1–8.

[23] Martin Schmid, Matej Moravcík, Neil Burch, Rudolf Kadlec, Joshua Davidson, K. Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, Zach Holland, Elnaz Davoodi, Alden Christianson, and Michael H. Bowling. 2021. Player of Games. *ArXiv* abs/2112.03178 (2021).

[24] M. J. Schofield and Michael Thielscher. 2019. General Game Playing with Imperfect Information. *J. Artif. Intell. Res.* 66 (2019), 901–935.

[25] D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484–489.

[26] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* 362, 6419 (2018), 1140–1144.

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, L. Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *ArXiv* abs/1712.01815 (2017).

[28] Oskari Tammelin, Neil Burch, Michael Bradley Johanson, and Michael Bowling. 2015. Solving Heads-Up Limit Texas Hold'em. In *IJCAI*.

[29] Brian Zhang and Tuomas Sandholm. 2021. Subgame solving without common knowledge. *Advances in Neural Information Processing Systems* 34 (2021), 23993–24004.

[30] Yunsheng Zhang, Dong Yan, Bei Shi, Haobo Fu, Qiang Fu, Hang Su, Jun Zhu, and Ning Chen. 2021. Combining Tree Search and Action Prediction for State-of-the-Art Performance in DouDiZhu. In *IJCAI*.