Evaluation of Perfect-Information Monte Carlo Tree Search in Imperfect-Information Games

Varun Adithiyan Ravichandran, James Goodman, Simon Lucas
Queen Mary University of London
vadithiyan@gmail.com, james.goodman@qmul.ac.uk, simon.lucas@qmul.ac.uk

***Abstract*—Perfect Information Monte Carlo (PIMC) methods address imperfect-information by sampling fully determinized states and aggregating search results. This is theoretically flawed, but often effective. We evaluate Perfect Information Monte Carlo Tree Search (PI-MCTS) across twelve tabletop games using three aggregation policies in comparison with Open Loop MCTS (OL-MCTS) and Information Set MCTS (IS-MCTS). Results show that PI-MCTS matches or exceeds OL-MCTS but rarely outperforms IS-MCTS. Performance strongly depends on the determinization count and varies by game.**

I. Introduction

Monte Carlo Tree Search (MCTS) works well in perfect-information games [1], but imperfect-information settings introduce uncertainty about hidden states. Perfect Information Monte Carlo (PIMC) mitigates this by sampling *determinizations*, fully revealed game states consistent with known information, and aggregating search outcomes. PIMC has proven effective in games like Skat [2] and Bridge [3], despite theoretical issues such as strategy fusion and non-locality [4], [5].

Alternatively, Information Set MCTS (IS-MCTS) addresses strategy fusion by constructing a tree of information sets rather than ground states [6], This prevents an agent from thinking that it can make different decisions from two ground states (e.g. that differ in the cards held by the opponent) that are part of the same information set (i.e. that are indistinguishable given the information known by the agent). However, the resulting tree can have a very high branching factor, limiting search depth and potentially degrading decisions in practice.

PIMC is an umbrella algorithm where any appropriate algorithm can serve as its per-determinization solver. MiniMax search has been most common [2], [3], though MCTS is a compelling alternative given its *anytime* properties [7].

Direct comparisons between these fully anytime algorithms are scarce. The one prior example, Cribbage, found IS-MCTS to be significantly better [7]. To address this gap, we evaluate PI-MCTS (PIMC using MCTS as the solution algorithm) across twelve games in the Tabletop Games (TAG) framework [8], testing three aggregation strategies against Open Loop MCTS (OL-MCTS) and IS-MCTS.

We hypothesize PI-MCTS would outperform OL-MCTS across all imperfect-information games and potentially match IS-MCTS in some. Results show, however, that PI-MCTS does beat OL-MCTS in many such games, but IS-MCTS generally outperforms both. PI-MCTS performance is highly sensitive to determinization count but aggregation strategy has comparatively little impact.

* 979-8-3315-9476-3/26/$31.00 ©2026 IEEE

II. Background

A. Monte Carlo Tree Search (MCTS)

MCTS is an anytime tree search algorithm that uses random sampling to guide decision making in games [1]. It incrementally builds a search tree through four phases: selection, expansion, simulation, and backpropagation. During selection, actions are chosen using a tree policy such as UCT,

$$UCT(a) = Q(a) + C\sqrt{\frac{\ln N}{n(a)}} \tag{1}$$

where $Q(a)$ is the mean value of action a, $n(a)$ is its visit count, N is the total visits to the node, and C controls exploration [9]. A new node is then expanded, a simulation is run to the end of the game, and the result is backpropagated to update node statistics.

B. Open Loop Monte Carlo Tree Search

Open Loop Monte Carlo Tree Search (OL-MCTS) [10] is a variation of MCTS in which nodes are defined by the sequence of actions taken from the root node. In a perfect information deterministic game OL-MCTS is identical to state-based MCTS. However, in imperfect information environments different visits to the node may have different underlying states depending on stochastic events in the game.

C. Perfect Information Monte Carlo

Perfect Information Monte Carlo (PIMC) uses Monte Carlo sampling techniques to tackle decision making problems in an imperfect information setting. In PIMC each of the determinizations is treated as a game where the state is fully observable and consistent with the player's current partially observed game state [1]. A tree search is then performed on each of these determinized states, usually Minimax. Actions are chosen based on results aggregated from all such determinized trees. PIMC has been shown to suffer from theoretical weaknesses, notably Strategy Fusion, Non-Locality [4], and Information Leakage [11]. PIMC with postponed reasoning was recently introduced to at least partially overcome strategy fusion [12], and deserves further investigation in future work.

D. Information Set Monte Carlo Tree Search

Information Set Monte Carlo Tree Search (IS-MCTS) addresses imperfect information by building a single tree over information sets rather than determinized states [6]. Each iteration samples a compatible determinization, performs a simulation, and updates shared node statistics. This avoids strategy fusion and has demonstrated strong empirical performance in a range of games [6], [13].

III. Previous Work

A. Perfect Information Monte Carlo in Card Games

Previous work has shown that PIMC performs strongly in several imperfect information games, particularly in trick-taking domains where hidden information is progressively revealed through a process of disambiguation [5], [14].

GIB [3] was an early example of PIMC, combining Monte Carlo sampling with double-dummy analysis by treating all hands as visible to enable perfect-information search. Similarly, Kermit achieved expert level Skat play [2] using human game data to evaluate states and biased sampling with domain knowledge from bidding outcomes.

In Cribbage, several MCTS variants were compared by Kelly et al., 2017 [7], including Determinized UCT (here termed PI-MCTS) and IS-MCTS. Results showed that PI-MCTS underperformed a scripted agent, while IS-MCTS outperformed all other approaches without game specific tuning. The present work generalises these findings across a broader set of games.

B. Aggregation Strategies

While an array of choices are presented in the literature, there is limited consensus with studies reporting different winners depending on domain and budget. Root Parallelization [15] and Ensemble MCTS [16] variants use different types of aggregation rules for combining the results from different trees, in both perfect and imperfect information games. In Ensemble UCT [16] multiple UCT instances are run independently and their root information aggregated at decision time using a weighted average for aggregation (as the best performer of several methods tried). In experiments conducted in Cribbage [7], visit counts for child nodes across all determinizations were summed and the move with the highest total visit count was chosen as the best move.

In computer Go majority voting outperformed average aggregation at higher core counts[15], suggesting shallow, diverse trees can help UCT make better choices.

IV. Methods

A. PI-MCTS

PI-MCTS samples D determinizations $\{s_1, \ldots, s_D\}$ of the current game state s by filling in all information hidden from the agent in s at random while remaining consistent with the agent's information set. For each determinization s_d, an independent OL-MCTS search is performed with a simulation budget of B/D, where B is the total budget allocated to the agent for an action.

OL-MCTS is executed for each determinization $d \in \mathcal{D}$, the visit count $N_d(a)$ and estimated value $V_d(a)$ are recorded for every available action a. These statistics are then aggregated across all determinized states to inform the selection of the final move. Three aggregation policies are evaluated (Section IV-B).

B. Aggregation Policies

The following notation is used throughout this section.

- $\mathcal{D}$ is the set of determinizations with $|\mathcal{D}| = D$
- d is a single determinization, where $d \in \mathcal{D}$
- $\mathcal{A}$ is the set of possible actions.
- $a \in \mathcal{A}$ is a single possible action in the game state
- $V_d(a)$: the accumulated value of a in determinization d
- $N_d(a)$: the visit count of a in determinization d

1) **Total Visits**: The action with the highest number of visits summed across all determinizations is chosen:

$$a' = \arg\max_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} N_d(a) \tag{2}$$

2) **Average Value**: The action with the highest mean estimated value, calculated as the ratio of the total accumulated value to the total number of visits in all determinized MCTS trees. The average value of action $a \in \mathcal{A}$ is defined as:

$$\bar{Q}(a) = \frac{\sum\limits_{d \in \mathcal{D}} V_d(a)}{\sum\limits_{d \in \mathcal{D}} N_d(a)}, \tag{3}$$

The final action is then selected as:

$$a' = \arg\max_{a \in \mathcal{A}} \bar{Q}(a). \tag{4}$$

3) **Single Vote**: Based on [15], this policy selects the final action by a majority vote over the best actions chosen independently from each determinized MCTS tree. Each tree selects the action a_d^* with the highest mean value:

$$a_d^* = \arg\max_{a \in \mathcal{A}} \frac{V_d(a)}{N_d(a)}$$

The final action a' is selected as the one that receives the most votes:

$$a' = \arg\max_{a \in \mathcal{A}} \sum_{d \in \mathcal{D}} \mathbb{I}[a = a_d^*].$$

All games are deterministic except for deck shuffling. When stochasticity is limited to the initial deal, each determinization evolves deterministically, making open-loop PI-MCTS effectively equivalent to a state-based approach. This equivalence does not hold for IS-MCTS, where node statistics aggregate across multiple determinizations, so values reflect a mixture of underlying states rather than a single one. If stochastic events occur during play (e.g., reshuffling in Poker, Dominion, or Colt Express), open-loop PI-MCTS diverges from a state-based representation deeper in the tree, which can affect performance.

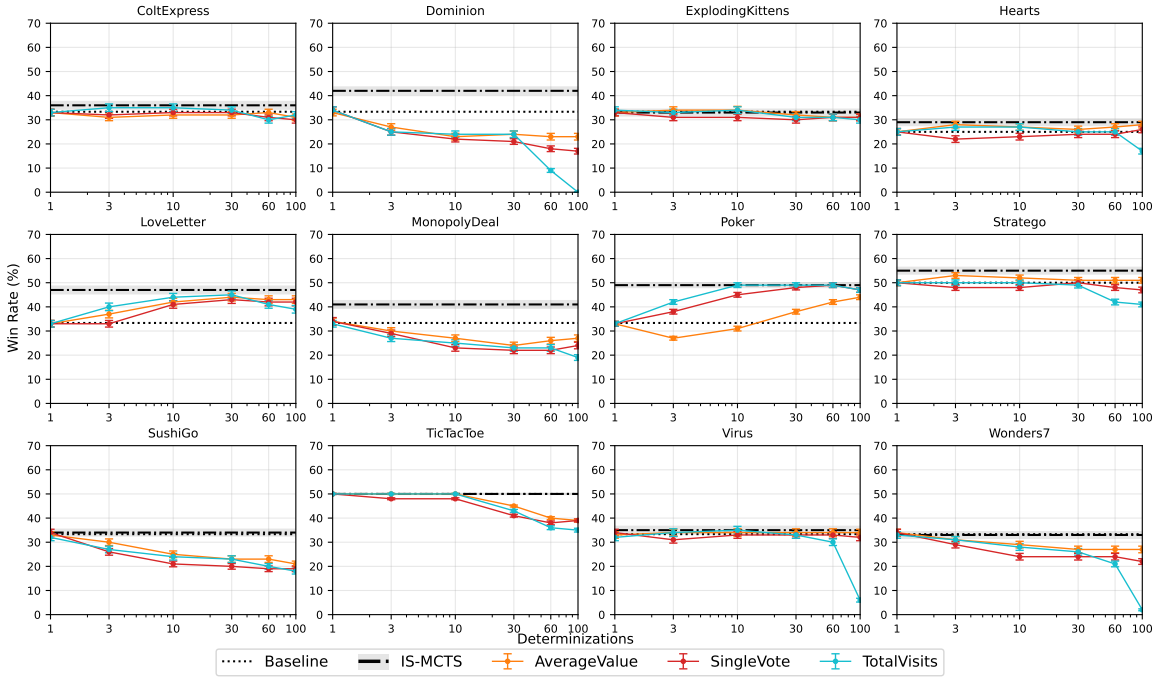

Fig. 1: Each solid line is PI-MCTS with a different aggregation policy. The shaded Dot-Dash Line is the win rate of IS-MCTS (with error margin). The baseline is the expected win rate for equal skill agents (i.e. 1/players). Error bars show 95% Confidence Intervals over win rates. PI-MCTS variants performs better than OL-MCTS in most games, but IS-MCTS consistently shows the best performance.

V. Experiments

PI-MCTS was evaluated on twelve games against IS-MCTS and OL-MCTS (baseline). MCTS iterations per decision were used as the computational budget to ensure fair comparison between agents based on algorithmic structure rather than implementation complexity or computational overheads.

Tournaments used 4000 MCTS iterations and 3 players unless fixed by the game (Tic-Tac-Toe and Stratego: 2 players; Hearts: 4 players). Parameters were tuned independently for each game with equal optimisation budgets, or taken from prior work [13]. Each tournament assigned either PI-MCTS or IS-MCTS as the focus player, with all other players using OL-MCTS. Only information-handling differed between agents, and all other settings were held constant. Additional results for tournaments conducted with 10,000 iterations are reported in the discussion.

VI. Results & Discussion

The baseline OL-MCTS agent is equivalent to PI-MCTS with determinization $D = 1$. In the following discussion, PI-MCTS refers to runs with $D > 1$. Overall, PI-MCTS outperforms the MCTS baseline in games with persistent hidden information, but its performance is sensitive to the number of determinizations. In most games, performance improves as D increases, before declining at higher values (relative to the baseline), as shown in Fig. 1. optimal D is game dependent. Contrary to expectations, IS-MCTS consistently matches or outperforms PI-MCTS across all evaluated games.

Performance differences between agents are primarily driven by the temporal dynamics of hidden information, specifically the "disambiguation rate", which is the rate at which hidden information is revealed [5]. PI-MCTS and IS-MCTS yield the greatest benefits when hidden information persists over many turns.

This pattern is evident in Hearts and Stratego, where PI-MCTS outperformed OL-MCTS by 5% under standard budgets, increasing to 10% with higher budgets. In Poker, this pattern is stronger, where PI-MCTS consistently exceeds the baseline and matches IS-MCTS at lower budgets and surpasses IS-MCTS performance under higher budget settings at higher determinizations (100–500), particularly under total visits aggregation (Fig 2). Poker is the only environment where the average value strategy performs significantly worse than other strategies, for reasons currently unclear. All three game exhibit low disambiguation.

Conversely, when uncertainty is transient or rapidly resolved, the benefits of determinization are offset by budget fragmentation. Splitting a search budget across multiple determinizations reduces per-tree depth; if the game state quickly transitions to perfect information, this shallow search becomes a liability. In Sushi Go!, PI-MCTS performs poorly with three players because the game becomes one of perfect information by turn three, though its performance improves with five players as information remains hidden longer. Similarly, Colt Express and Virus show only marginal gains (2–5%) due to their rapidly resolving or cyclical hidden information. In

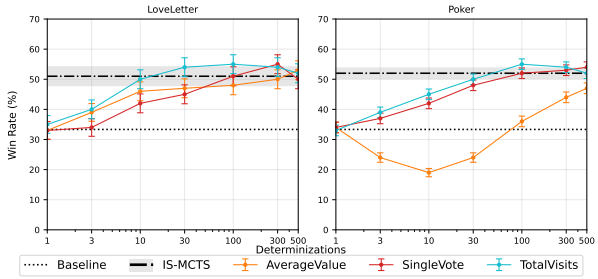

Fig. 2: Comparison of algorithm performance under a higher computational budget of 10,000 iterations. Key and other details are as in Fig. 1.

Exploding Kittens, high stochasticity and low skill expression resulted in no significant performance gaps. Finally, in four other games (Tic-Tac-Toe, Monopoly Deal, Dominion & Wonders7), PI-MCTS performed worse than the baseline.

Effect of Determinization Count on Search Quality

At low D, each tree has enough budget for UCT to meaningfully distinguish between actions, so visits concentrate on higher-value moves; aggregating visits across trees (Total Visits) then reliably favours better actions. As D increases, per-tree budgets shrink, search becomes shallow, and visit counts across trees converge toward one per action, making the policy approach behaviour similar to a player making random decisions. No aggregation strategy shows a consistent advantage and performance is mainly driven by the choice of D, which varies by game.

VII. Conclusions and Future Work

We evaluated PI-MCTS across a range of tabletop games. It outperforms OL-MCTS when hidden information persists, but is typically matched or exceeded by IS-MCTS. While some PI-MCTS configurations approach or surpass IS-MCTS at higher budgets, this requires careful tuning due to sensitivity to determinization count, making it more computationally demanding. These results support the prior findings in Cribbage [7], but across a wider range of games. Overall, the findings suggest that PI-MCTS is most effective when hidden information persists long enough to justify the reduced per-tree search depth induced by maintaining multiple determinizations, and when sufficient computational resources are available for additional parameter tuning.

References

[1] C. B. Browne, E. Powley, D. Whitehouse, *et al.*, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, no. 1, 2012. DOI: 10.1109/TCIAIG.2012.2186810.

[2] M. Buro, J. R. Long, T. Furtak, and N. Sturtevant, "Improving State Evaluation, Inference, and Search in Trick-Based Card Games," 2009.

[3] M. L. Ginsberg, "G I B : Steps Toward an Expert-Level Bridge-Playing Program," 1999.

[4] I. Frank and D. Basin, "Search in games with incomplete information: A case study using Bridge card play," *Artificial Intelligence*, no. 1, 1998. DOI: 10.1016/S0004-3702(97)00082-9.

[5] J. Long, N. Sturtevant, M. Buro, and T. Furtak, "Understanding the Success of Perfect Information Monte Carlo Sampling in Game Tree Search," *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 1, 2010. DOI: 10.1609/aaai.v24i1.7562.

[6] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information Set Monte Carlo Tree Search," *IEEE Transactions on Computational Intelligence and AI in Games*, no. 2, 2012. DOI: 10.1109/TCIAIG.2012.2200894.

[7] R. Kelly and D. Churchill, "Comparison of Monte Carlo Tree Search Methods in the Imperfect Information Card Game Cribbage," 2017.

[8] R. D. Gaina, M. Balla, and A. Dockhorn, "TAG: A Tabletop Games Framework," 2020.

[9] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Machine Learning: ECML 2006*, Berlin, Heidelberg, 2006. DOI: 10.1007/11871842_29.

[10] D. Perez Liebana, J. Dieskau, M. Hunermund, S. Mostaghim, and S. Lucas, "Open Loop Search for General Video Game Playing," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015. DOI: 10.1145/2739480.2754811.

[11] T. Furtak and M. Buro, "Recursive Monte Carlo search for imperfect information games," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 2013. DOI: 10.1109/CIG.2013.6633646.

[12] J. Arjonilla, A. Saffidine, and T. Cazenave, "Perfect information monte carlo with postponing reasoning," in *2024 IEEE Conference on Games (CoG)*, 2024.

[13] J. Goodman, D. Perez-Liebana, and S. Lucas, "Following the Leader in Multiplayer Tabletop Games," in *Proceedings of the 18th International Conference on the Foundations of Digital Games*, Lisbon Portugal, 2023. DOI: 10.1145/3582437.3582449.

[14] D. Rebstock, C. Solinas, M. Buro, and N. R. Sturtevant, *Policy Based Inference in Trick-Taking Card Games*, 2019. DOI: 10.48550/arXiv.1905.10911.

[15] Y. Soejima, A. Kishimoto, and O. Watanabe, "Evaluating Root Parallelization in Go," *IEEE Transactions on Computational Intelligence and AI in Games*, no. 4, 2010. DOI: 10.1109/TCIAIG.2010.2096427.

[16] A. Fern and P. Lewis, "Ensemble Monte-Carlo Planning: An Empirical Study," *Proceedings of the International Conference on Automated Planning and Scheduling*, 2011. DOI: 10.1609/icaps.v21i1.13458.