Look-ahead Search on Top of Policy Networks in Imperfect Information Games

Ondřej Kubíček, Neil Burch and Viliam Lisý

1 Artificial Intelligence Center, Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague
2 Sony AI
3 Alberta Machine Intelligence Institute, University of Alberta
{kubicon3, viliam.lisy} @fel.cvut.cz, nburch@ualberta.ca

Abstract

Search in test time is often used to improve the performance of reinforcement learning algorithms. Performing theoretically sound search in fully adversarial two-player games with imperfect information is notoriously difficult and requires a complicated training process. We present a method for adding test-time search to an arbitrary policy-gradient algorithm that learns from sampled trajectories. Besides the policy network, the algorithm trains an additional critic network, which estimates the expected values of players following various transformations of the policies given by the policy network. These values are then used for depth-limited search. We show how the values from this critic can create a value function for imperfect information games. Moreover, they can be used to compute the summary statistics necessary to start the search from an arbitrary decision point in the game. The presented algorithm is scalable to very large games since it does not require any search during train time. We evaluate the algorithm’s performance when trained along Regularized Nash Dynamics, and we evaluate the benefit of using the search in the standard benchmark game of Leduc hold’em, multiple variants of imperfect information Goofspiel, and Battleships.

1 Introduction

The field of multi-agent deep reinforcement learning has achieved remarkable success in training strong agents for two-player adversarial games through selfplay. Many of these agents develop strategies represented by neural networks, which subsequently provide probability distributions for actions based on current observations during gameplay.

Despite their success, the neural networks can occasionally lead the agent to make suboptimal decisions that an adversarial opponent may exploit. AlphaZero-style algorithms in Go are susceptible to exploitation, as evidenced by winning in the trained policy network 97% of the time [Wang et al., 2023]. Similarly, policy-gradient algorithms display evident gameplay errors, necessitating the application of multiple heuristics to rectify these errors [Perolat et al., 2022]. However, there are principled ways of using search to avoid these mistakes, as evidenced by AlphaZero [Silver et al., 2018], DeepStack [Moravčík et al., 2017], and Student of Games [Schmid et al., 2023].

Algorithms that use policy-gradient methods in imperfect information games like Regularized Nash Dynamics (RNaD) [Perolat et al., 2022; Perolat et al., 2021] rely on directly applying the neural network’s policy during gameplay. Adding search to these methods could improve the strategy and mitigate some mistakes. However, this enhancement comes at the expense of turning model-free algorithms into model-dependent ones during gameplay.

Adding search to policy-gradient methods in imperfect information games presents multiple challenges that are not present in a perfect information case, which makes algorithms like AlphaZero unusable in this setting. First, the imperfect information search must be performed not only from the current unobserved state of the game, but from all possible states that share the same public information amongst all players. Second, the state values depend on beliefs players have about the current information of their opponents. Thirdly, the opponent’s strategy is unknown, and the optimal strategies are not unique. Hence, we need to optimize over all possible strategies the opponent may deploy both in the remainder of the game as well as unobserved past actions.

In this paper, we introduce the algorithm Multi-Agent Search With Policy Transformations (SePoT), which builds upon the concept of multi-valued states [Brown et al., 2018]. SePoT trains a policy transformation critic network that estimates the expected value of policies derived from the policy network. This critic is trained alongside a policy network with a policy-gradient technique without requiring any additional trajectories beyond those sampled by the original policy-gradient algorithm. The critic network is trained with an off-policy V-trace that estimates expected values against a multitude of potential opponent strategies derived from the policy represented by the policy network. We show that this critic provides sufficient information to perform a safe search [Šustr et al., 2021] in imperfect information games.

Furthermore, our approach enables safe search at any point within a game, utilizing the information provided by the introduced critic network. This eliminates the need for conducting search in all decision points from the root until the current decision point, which is often intractable in games with
little public knowledge, but at the cost of requiring a domain-specific function that reconstructs current game segment. It is in strict contrast to prior methods [Brown et al., 2018; Moravčík et al., 2017; Schmid et al., 2023]. This flexibility allows us to perform search only in game segments sufficiently simple for available compute power, while employing the neural network policy in other parts of the game.

Finally, we establish a connection between value functions and multi-valued states within imperfect information games that enables training previously used imperfect information value functions [Moravčík et al., 2017; Schmid et al., 2023] without the expensive search in the training time.

In our experiments, we evaluate SePoT with RNaD [Pero- lat et al., 2022]. We show that using search in head-to-head play in imperfect-information Goofspiel and Battleships improves the performance over the network trained with RNaD. Furthermore, we demonstrate that the exploitability of policy derived from the search is the same or lower than that of the policy network from RNaD in a small version of imperfect information Goofspiel and the standard benchmark game Leduc hold’em.

Our main contributions are: (1) Adding safe search on top of the policy-gradient method without requiring training time search or additional training trajectories. (2) Introducing search technique that may be performed anywhere in the game without performing search in previous parts of the game. (3) Establishing a practical connection between the multi-valued states and the value functions. (4) Showing that the additional search improves the performance of RNaD.

2 Issues With Imperfect Information Search

![Figure 1: Current state in a smaller version of Battleships and all possible states sharing the same public information](image)

Consider a smaller version of Battleships, in which two players have three ships of sizes 3, 2, and 2 on a square grid of size 5. Players take turns shooting on a tile of the opponent’s board after all ships are secretly placed. The player that first hits all tiles occupied by the opponent’s ships wins. Suppose we are in the middle of the game, and the current boards are shown in Figure 1. Each player knows their remaining ship’s position, but does not know the opponent’s. Based on the public shooting actions up until this point, we know that there are only two possible positions for the player and the opponent. The player may have either placed the ship at A1-A2 or A2-B2. Similarly, the opponent may have placed the ship at A1-A3 or A3-C3. Possible positions are shown on each board with the dashed rectangle. There are four possible states, that is, all possible combinations between the positions across both players, as shown in Figure 1c. The first player cannot distinguish between states $w_1$ and $w_2$, neither between $w_3, w_4$. Similarly, the opponent cannot distinguish between $w_1, w_3$ and between $w_2, w_4$. Shooting anywhere else than A1, A2, A3, B3, and C3 does not make sense for a rational player. However, training RNaD for three days results in a policy that shoots to these tiles with a probability less than 45%.

Employing additional search should improve this policy, but doing so naïvely in imperfect information games introduces multiple issues. The first issue is that starting search only from states that are possible from the player’s point of view, which are $w_1$ and $w_2$, would search only in those parts of the game where the player placed the ship at A1-A2. This means that the player incorrectly assumes the opponent knows the ship’s position during this search. The outcome of the search would be that the game always ends in 2 turns because the opponent would shoot at the ship. Since the player requires at least three turns to win, it could never win against this informed opponent, and its strategy would be to play randomly. However, in the real gameplay scenario, the opponent lacks knowledge of the player’s ship’s location, providing the player, who is moving first, with a genuine opportunity to win. Nevertheless, this issue can be fixed by starting the search from all four possible states.

The second issue is more profound. In practice, for many different reasons, the player may incorrectly assume that the opponent is slightly more likely to place his ship at position A1 instead of C3. In such a case, the player would always shoot at either tile A1, A2, or A3 since it provides more reward in expectation from its point of view. This leads to a strategy that could be exploited by an adversarial opponent who would never put the ship there in the first place. This is a well-studied problem in imperfect information games. One possible solution during search is to assume that the opponent can play any strategy in earlier parts of the game. This assumption then allows computing a more robust strategy against possible play from the game’s previous and subsequent parts [Burch et al., 2014; Moravčík et al., 2016; Brown and Sandholm, 2017].

The third issue for applying search in the situation in Figure 1 are sizes of required game structures. The full end-game cannot be solved online in reasonable time since it has over $10^{10}$ terminal histories. Also, it is impossible to safely search in earlier decision points, since after the first move there are more than $10^8$ possible ship configurations and branching factor of 50, which means that even a depth-limited game with look-ahead of one move has more than $10^{10}$ histories. Using a depth-limited search with a value function at the depth limit only later in the game enables search even with much less computational resources [Kovářík et al., 2023].

3 Background

Factored-observation stochastic game (FOSG) [Kovářík et al., 2022] is a tuple $G = (\mathcal{N}, \mathcal{W}, p, u^0, \mathcal{A}, \mathcal{T}, R, \mathcal{O})$, where $\mathcal{N} = \{1, \ldots, N\}$ is a player set, $\mathcal{W}$ is the set of states and $u^0$ is an initial state, $p : \mathcal{W} \rightarrow 2^{\mathcal{N}}$ is a player function, which for given state $w \in \mathcal{W}$ assigns currently acting players from $\mathcal{N}$, $\mathcal{A} = \prod_{i \in \mathcal{N}} A_i$, is the space of joint actions. We denote $\mathcal{A}_i(w)$ legal actions for player $i$ in the world state $w$. $\mathcal{T}$:
$W \times A \rightarrow W$ is the transition function, $R : W \times A \rightarrow \mathbb{R}$ is the reward function and $O : W \times A \times W \rightarrow \mathbb{O}$ is the observation function. $O$ is factored as $O = (O_0, O_1, \ldots, O_N)$, where $O_0$ are public observations given to all players and $O_i$ are private observations of player $i$. In this work, we focus on two-player zero-sum games, where $N = \{1, 2, c\}$ and $R_i(w, a) = -R_j(w, a) \forall (w, a) \in W \times A; c$ denotes a chance player who has publicly announced a non-changeable policy throughout the game. Moreover, we study only games with perfect recall, where neither player forgets any information.

A trajectory $\tau = w^0a^0w^1a^1 \ldots w^t \in (W, A)^*W$ for which $a^i \in A(w^i)$ and $w^{i+1} = T(w^i, a^i)$ is a finite sequence of states and actions in the game. Trajectory utility for player $i$ is a cumulative reward $u_i(\tau) = \sum_{t=0}^{\infty} R_i(w^t, a^t)$. We use $H$ to denote histories, which are trajectories that start in the initial state $w^0$. $Z = \{z \in H | z$ is terminal$\}$ are histories in which neither player has an action to play, effectively ending the game. $h \subseteq h'$ means that $h'$ extends $h$ and $h^0$ is an initial history, that corresponds to the initial state $w^0$, before any player played any action. Each history ends with some world state $w^t$. Therefore, we denote $R_i(h, a) = R_i(w^t, a)$ as a reward given to player $i$, if in history $h \in H$ the players play joint action $a$. Since player $i$ does not observe the state of the game directly, but only observes its actions $A_i$, private observations $O_i$ and public observations $O_0$, it cannot distinguish between multiple different histories. We say that these indistinguishable histories are in the same infoset $s_i \in S_i$. If two histories $h, h' \in H$ belong to the same infoset $s_i$, then player $i$ has to have the same actions in these histories and $s_i(h)$ denotes an infoset of player $i$ in history $h$. We will use $A_i(s_i)$ to denote these available actions in infoset $s_i$. Furthermore, we will use a notion of public-states $s_0 \in S_0$, which may be viewed as an information set for an external player which does not act in the game. Two histories belong to the same public state if and only if they have the same sequence of public observations from the initial state. Each public state is made from multiple information sets for a given player $i$. We will use $S_i(s_0)$ to denote all infosets that are in public state $s_0$ and $H(s)$ to denote all histories in an infoset $s$.

Policy of player $i$ is a mapping $\pi_i : S_i \rightarrow \Delta A_i$ s.t. $\pi_i(s_i) \in \Delta A_i(s_i)$. $\Pi_i$ is a set of all policies of player $i$. We will use $\pi_i(s_i, a_i)$ to denote the probability of playing action $a_i$ in $s_i$, while following the policy $\pi_i$. A strategy profile is defined as $\pi = (\pi_1, \ldots, \pi_N)$, and $\pi_{-i}$ is used to denote a strategy profile without the policy of player $i$.

Expected utility, if all players play according to strategy profile $\pi$ from history $h$ onward is defined as $u^\pi(h) = E_{\tau \sim \pi \mid h}(\tau)$. $P^\pi(h) = \prod_{h \in \tau \subseteq h} \prod_{j \in N} \pi_j(s_j(h), a_j)$ is a probability of reaching history $h$ if all players play according to strategy profile $\pi$. This may be separated into $P^\pi(h) = P^\pi_i(h)P^\pi_{-i}(h)$, where $P^\pi_i(h)$ is only a players $i$ contribution to reaching history $h$, similarly $P^\pi_{-i}(h)$ is a contribution of all the other players except $i$, which is called counterfactual reach probability. Counterfactual values for infoset $s_i$ is a sum of expected utilities of each history $h$ within the same infoset, weighted by the counterfactual reach probability of this history $\tilde{v}^\pi(s_i) = \frac{\sum_{h \in H(s_i)} P^\pi_i(h)\tilde{u}^\pi(h)}{\sum_{h \in H(s_i)} P^\pi_i(h)}$.

Best response against policy $\pi_i$ of a player $i$ is a policy $\pi_{BR}^i \in BR_i(\pi_i)$, where $u_i(\pi_{BR}^i, \pi_{-i}) \geq u_i(\pi_i, \pi_{-i})$ for all $\pi_{-i}$. If all players play a best response to each other, then the resulting policy profile $\pi^{NE}$ is called Nash equilibrium. The usual metric to compare strategies in two-player zero-sum games is an exploitability $E(\pi_i) = u_i(\pi_i, \pi_{BR}^i) - u_i^{\pi_{BR}^i}(h^0)$, that is how much utility can opponent get if it plays best response to the player’s strategy, compared to the utility of Nash equilibrium. From definition, if player $i$ plays Nash equilibrium, then the exploitability is 0.

### 3.1 Depth-Limited Solving

Range $r_i \in \Delta S_i(s_0)$ for player $i$ in public state $s_0$ is a probability distribution over infosets $s_i \in S_i(s_0)$ in public state $s_0$, which corresponds to player’s contribution to the probability of reaching this infoset. Furthermore range for public state $s_0$ is defined as $r = (r_1, \ldots, r_N) \in \Delta S_1(s_0) \times \cdots \times \Delta S_N(s_0)$. A public belief state is a public state with associated range $\beta = (s_0, r)$. Optimal value function is a function $v^*(\beta) = \mathbb{R}[S_1(s_0) \times [\Delta s_0]]$, that for given public belief state $\beta$ returns counterfactual values $v^*_\beta(s_i)$ for each player $i$ and infoset $s_i \in S_i(s_0)$ associated with public belief state $\beta$ if both players played optimally from that point based on associated ranges $r$. Value function $v(\beta)$ is then defined similarly, but it’s output could be noisy. For any public belief state, we can construct subgame $\mathcal{G}(\beta)$, which is the same FOSG as $\mathcal{G}$, but it starts in $s_0$ with initial ranges $r$ instead of the initial state. Depth-limited subgame is defined as $\mathcal{G}^{BR}(\beta)$. However, after some depth limit, the value function $v$ is called and returned counterfactual values are used as a terminal values.

Resolving some subgame to improve blueprint policy $\pi$ naively would involve fixing policy $\pi$ for all players in previous parts of the game and using single $w^A$, where players act with corresponding ranges. However, this may result in a much more exploitable strategy [Burch et al., 2014] than the original blueprint $\pi$. Multiple techniques were developed to deal with this issue in two-player zero-sum games [Burch et al., 2014; Moravcik et al., 2016; Brown and Sandholm, 2017; Brown et al., 2020]. In this work we will focus on the one proposed in [Burch et al., 2014]. When resolving from the perspective of player $i$, it constructs a slightly modified subgame called a gadget game $G_i^G(\beta, v^*_\beta)$, where $v^*_\beta$ are counterfactual values for each infoset $s_i \in S_i(s_0)$ of opponent $o$ in a current public state $s_0$, when players play according to policy $\pi$. This subgame has the artificial state $w^A$ as an initial state, but instead of using full ranges $r$, it uses only ranges $r_i$ and $r_c$. Directly after this artificial state, $[H(s_0)]$ opponent’s decision nodes are separated into multiple infosets based on the information player has in a given history. The opponent $o$ has a choice to either terminate the game and receive the counterfactual value $v^*_o(s_0)$ or continue the game as in the original subgame.

Suppose that after the depth limit, the optimal value function is used and that the counterfactual values $v^*_o^{NE}(s_0)$ used in the gadget game are computed for Nash equilibrium policy $\pi^{NE}$. Then the solution of gadget game $G_i^G(\beta, v^*_\beta)$ is part of a Nash equilibrium in the entire game [Burch et al., 2014;
Kovafik et al., 2023).

In this work, we focus on an alternative to the value function called multi-valued states [Brown et al., 2018], which, after the depth limit, gives the opponent the choice to fix its strategy until the end of the game. Then, the game terminates with the reward corresponding to the reward if player $i$ follows some blueprint policy $π_i$, while the opponent follows the fixed strategy. Authors of [Kovafik et al., 2023] pointed out that multi-valued states could be viewed as a form of a value function. We further discuss this in Section 4.2.

### 3.2 Regularized Nash Dynamics

Using usual reinforcement learning techniques to find an optimal policy with self-play does not guarantee convergence to Nash equilibrium and can be exploited [Lancot et al., 2017]. However, regularizing the reward leads to establishing this guarantee [Perolat et al., 2020; Sokota et al., 2023].

Reward regularization with KL-Divergence is

$$R_i^R(h, a, π) := R_i(h, a) − \eta \log \frac{π(s_i, a_i)}{π^R(s_i, a_i)} + \eta \log \frac{π^R(s_i, a_i)}{π^R(s_i, a_{−i})}$$

(1)

where $π^R$ is a regularization policy and $\eta$ is a hyperparameter for adjusting the regularization effect. Games with such a regularized reward have a single unique fixed point $π_{Nash}$ for any regularization policy $π^R$ [Perolat et al., 2021]. Follow the Regularized Leader [McMahan, 2011] may be used to converge to $π_{Nash}$. Setting $π^R = π_{Nash}$ will change the unique fixed point. If this is done repetitively, the fixed point will gradually converge to the Nash equilibrium.

In order to converge to a Nash equilibrium in large games with reward regularization, the authors in [Perolat et al., 2022] propose using two-player V-trace with Neural Replicator Dynamics [Hennes et al., 2020] as a loss function for policy head and usual L2 loss for value head. With this method, it is possible to achieve human-level performance in Stratego.

### 4 Multi-Agent Search With Policy Transformations

In this section, we present an algorithm Multi-Agent Search With Policy Transformations (SePoT) that trains an additional critic alongside a policy with some policy-gradient algorithm for imperfect information games, like RNaD [Perolat et al., 2022]. During gameplay, this critic is used to construct a depth-limited gadget game that is then solved with some search algorithm, in our case CFR+ [Tammelin et al., 2015].

#### 4.1 Network Training

Performing a depth-limited search in imperfect information games requires a value function, that after the depth limit, assigns value to each infoset based on ranges [Kovafik et al., 2023]. To use a value function based on multi-valued states, it is necessary for each history $h$ to learn multiple values that correspond to the expected values if the opponent plays some fixed policies against the player’s blueprint policy. Training such value function presents two difficulties to overcome. The first is the selection of the opponent policies to be used as a basis to compute the expected values. In large games, simply using some predefined policy is not viable due to the sheer size of full policies. The second is estimating the expected value of these policies.

To find different opponent policies, we decided to use a similar method to the "bias approach" in [Brown et al., 2018]. The main idea is to use a single policy, which in our case is the policy from the policy network, and then apply some mapping that maps the network policy to a different policy. We call this mapping a policy transformation.

**Definition 1.** Policy transformation $f^T : Π_i \rightarrow Π_i$ for player $i$ is a function that maps policy to a different one.

In poker, an example of a transformation is folding more frequently [Bowling et al., 2009; Bard et al., 2014]. Similarly, in Goofspiel, certain transformations increase the probability of playing cards with lower value if the player is winning or decrease the probability of playing even-valued cards. These transformations do not have to be game-domain specific. However, the effectiveness of different transformations varies across different games. These transformations aim to cover the strategy space as much as possible. Constant transformations that transform any policy into each pure strategy cover this space fully [Brown et al., 2018].

In our experiments, we use transformations that move the actor policy $π$ in some direction with a fixed step size $k \in \mathbb{R}$. These transformations are encoded within a neural network, which undergoes concurrent training with the actor network. At each training step, for each trajectory, we compute the direction in which the actor policy for that trajectory has shifted. Subsequently, we update only the transformation associated with the minimum Euclidean distance across the trajectory. Since each update should improve the actor policy against the current opponent’s policy, this direction should aim into the important regions of strategy space, but the quality of these transformations is not guaranteed.

To estimate the expected value of these transformed policies against the player’s blueprint, we propose to train an additional critic network $u_i(h)[f^T]$, that takes history $h$ as an input and outputs the player’s $i$ expected value for each policy transformation $f^T$. To avoid sampling additional trajectories, which may be impossible in continuing tasks, we would like to use already sampled trajectories by the original policy-gradient algorithm more effectively. This may be achieved by using some off-policy training algorithm. In this work, we have used a V-trace estimator.

We define the V-trace operator $V$ for histories, while following some sampling policy $μ$ as

$$V_{μ} (h) = v_μ(h) +$$

$$\mathbb{E}_μ \left[ \sum_{t=0}^{T} \rho^t \left( \prod_{0 \leq q < t} c^q \right) \left( R^T_t + v_{μ}(h^{t+1}) - v_μ(h^t) \right) h^0 = h, μ \right]$$

(2)

where $h^t$ is history at timestep $t$ in trajectory $τ$ with initial history being $h^0$. $\rho^t = \min(\mathbb{E}_{μ} π(s_i(h^t), a^t))$ and $c^q = \min(τ_i π(s_i(h^t), a^t))$ are truncated importance sampling.
weights. If player does not act in timestep $t$, or if the $t \geq l$, we say that $v_\rho(h) = 0$.

**Theorem 1. (Inspired by [Espeholt et al., 2018]):** Let $\rho^1 = \min(\rho, \pi(s_i(h^s), a^s_i)), c^1 = \min(c, \pi(s_i(h^c), a^c_i), \rho \geq \varepsilon \geq 1$ and $\mu(s, a) > 0 \forall s, a$. Let us assume there exists $\kappa \in [0, 1]$, such that $\mathbb{E}_{\mu}^{\rho}[(\prod_{0 \leq g < l} c^1)] \geq \kappa$, where $l$ is a length of the trajectory. Then the V-trace operator $V$ has a unique fixed point $v^\pi$, which is the expected value of following policy

$$\pi_P(s_i(h^t), a^i_t) = \min(\pi(u(s_i(h^u), a^u_i), \pi(s_i(h^t), a^t_i))$$

(3)

Using $\rho \to \infty$ will result in convergence to the $v^\pi$, that is, the expected value of the policy $\pi$ in given history. The proof of Theorem 1 is in Appendix A. For training the history network, we use standard L2 loss.

### 4.2 Constructing Public Belief State

After the training phase, the original policy-gradient algorithm ends up with a trained policy network, which is then used at each decision point to retrieve policy. Based on this policy, the action is sampled, and the game proceeds. SePoT finishes the training with the additional history value network.

As discussed in the Section 2, naively employing search in the game could lead to highly exploitable strategies. Using resolving gadget [Burch et al., 2014], which limits the opponent’s counterfactual value in root infosets, bounds the error after performing search in a subgame. Constructing a gadget game requires the opponent’s counterfactual values $v_o^\pi$ in each root infoset and public belief state $\beta = (s_0, r)$. Most previous methods stored counterfactual values $v_o^\pi$, public state $s_0$ and ranges $r$ from previous searches [Moravčík et al., 2017; Schmid et al., 2023]. SePoT could leverage the same idea. Still, we propose a novel approach of using the trained history network to compute counterfactual values, the trained policy network to retrieve ranges for searching player $i$, and a function that reconstructs the public state. This allows running the search from an arbitrary public state without the need to search in all parent public states.

Safe search in imperfect information games has to be performed from all possible histories based on public information. This requires all histories $H(s_0)$ from current public state $s_0$ to construct the gadget game $G_o^2$. Let us assume a function that generates all histories in the public state $s_0$ from the publicly available information to all players. While the details of reconstructing the public state $s_0$ are not explored in detail here, for games like Goofspiel or Battleships, the public state can be reconstructed by Constraint Satisfaction Problem (CSP) [Russell and Norvig, 2009; Seitz et al., 2021].

Gadget game $G_o^2$ requires ranges for the player $i$ and the chance player. Chance player ranges $r_c$ can easily be stored during gameplay like older methods [Moravčík et al., 2017; Schmid et al., 2023; Brown and Sandholm, 2018]. Suppose the player $i$ played only by using the policy from the neural network during current gameplay and decides to perform a search at some public state $s_0$. In that case, it has to recompute the ranges $r_i$ for each infoset $s_i \in S_i(s_0)$. This requires backtracking from all of these infosets to all previous decision points and acquiring the policy from the policy network for each of them. Because we are in a perfect recall setting, each infoset cannot have more parenting infosets than the maximal trajectory length $l_{max}$ so this infoset from the initial state $w_0$, so this backtracking involves at most $|S_i(s_0)|l_{max}$ queries to the policy network.

In order to compute the counterfactual values in root infosets for the opponent, let us assume that one of the transformations we used in training is identity transformation, meaning that the critic network output corresponds to the expected value if both players follow the policy from the network, which serves as an approximation of a Nash equilibrium. The opponent’s counterfactual values are the expected values multiplied by the reaches of both chance and resolving players. As discussed, these ranges are always available, so the counterfactual values can always be computed from the definition. Thanks to this, we have both counterfactual values $v^\pi_o$ and $\beta$ available in each decision point to construct a gadget game and perform a safe search.

We compare previously used continual resolving for constructing public belief state with our approach in Appendix C.1

### 4.3 Value Function

Figure 2: Subgame with 3 leaf public states. Right part shows the detail of public state $s_0^1$ with 4 histories and 2 information sets for each player. The multi-valued states technique modifies this public state by giving player 2 a choice between two strategies in each history against blueprint policy of player 1.

Multi-valued states [Brown et al., 2018] are a modification of the original game, where after the depth limit, the opponent has one more decision point, in which it chooses its strategy for the remainder of the game against some fixed blueprint policy. This modified game is a standard game and can be solved by any game-solving algorithm unaware of value functions. Authors of [Moravčík et al., 2017] trained value function for Poker, which served as a generalization of the value function from perfect information games. This value function returns counterfactual values based on ranges at the depth limit. These were then specially adapted for Counterfactual Regret Minimization (CFR) [Zinkevich et al., 2007].

These two approaches to the depth-limited search may not seem connected, but the authors of [Kovarík et al., 2023] show this connection by explaining that the multi-valued states could be viewed as a value function, which is not trained for all possible strategies in the game, but only for some subset. However, their explanation does not explicitly clarify how this connection could be leveraged.
Let us assume a depth-limited subgame, as in Figure 2, with three leaf public states \( s^1_1, s^2_2, s^3_3 \). Further examining public state \( s^2_2 \) in Figure 2b, there are four possible states in this public state, which are separated into different infosets. First player has infoset \( s^1_1 \), that contains histories \( h_1, h_2 \) and \( s^2_1 \) containing histories \( h_3, h_4 \). Similarly second player has infosets \( s^2_2, s^3_3 \) that contains histories \( h_1, h_2 \) and \( h_3, h_4 \) respectively. The value function would return counterfactual values for each infoset, namely \( v^1_i(s^1_1), v^2_i(s^2_1), v^2_i(s^2_2), v^3_i(s^3_3) \).

In the same example, using multi-valued states as proposed in [Brown et al., 2018] gives the opponent, player 2, another choice to pick between two strategies in each infoset, as shown in Figure 2b.

Using CFR requires storing multiple statistics in each infoset, like regret and average strategy. These statistics must be stored even for these artificial infosets created by the multi-valued states. In such a scenario, the multi-valued states could be viewed as a stateful value function where these statistics determine the value function state. Each CFR iteration, this value function updates its state and returns counterfactual values based on the current state and input ranges \( r \). Let us assume that instead of performing regret matching and storing these necessary statistics, the opponent always picks a best response based on input ranges. In such a case, we can use multi-valued states as a usual value function with just additional computation step, without keeping the state inside the value function. Let us rephrase the value function in terms of using a history network with policy transformations for player \( i \).

\[
v(\beta)[s_i] = \min_{f^i \in F^i} \sum_{h \in H(s_i)} p_{\pi_i}^\beta(h) u_i(h)[f^T] / \sum_{h \in H(s_i)} p_{\pi_i}^\beta(h)\]  

(4)

where \( F^T \) is a set of all used policy transformations and \( P_{\pi_i}^\beta \) are directly taken from ranges \( r_{-i} \) in \( \beta \). Since we are in a two-player zero-sum setting and the network returns an expected value for the player \( i \), choosing a best response by the opponent is taking such transformation that minimizes the counterfactual value of the player \( i \).

This presents a new way to train value functions without using search in the training time at all by first training multi-valued states and then using them with different possible ranges to generate counterfactual values for training the value function.

With the constructed gadget game and a corresponding value function, any previously developed search algorithm for imperfect information games may conduct the subsequent search. During our experiments, we used CFR+ [Tammelin et al., 2015] exclusively. When using CFR+, a subclass of Growing tree-CFR (GT-CFR) introduced in [Schmid et al., 2023], the bounds already established by the GT-CFR authors in Theorem 3 also apply to our approach. However, the quality of this value function \( \xi \), which is the distance between values returned by the optimal value function and the trained value function, is affected by additional errors, even if the value function is trained perfectly. The first of these errors is that the blueprint strategy of resolving player \( i \) is not Nash equilibrium but just some approximation. Second is that the opponent’s strategies used for multi-valued states may not be best responses to any strategy.

The multi-valued states could be expanded by giving both players a choice to pick from multiple strategies for the rest of the game [Brown and Sandholm, 2019; Kovářík et al., 2023]. This would increase the number of values that must be trained quadratically because each pair of strategies between players yields different values. SePoT is compatible with this approach, and the only required change is to compute a Nash equilibrium of the underlying normal-form game at the depth limit during search instead of a best response.

We have already shown how the history critic trained for multi-valued states can be used as a value function. However, we could go one step further and train the usual value function out of this history critic. The value function may be trained before the gameplay phase by randomly sampling possible ranges, computing the counterfactual values from multi-valued states, and then training the value function on these counterfactual values. Such trained value function can then be used in already developed algorithms [Moravčík et al., 2017; Schmid et al., 2023] without any changes.

We provide pseudocode for training and gameplay in Appendix E.

5 Experiments

We have conducted two experiments to validate SePoT’s performance. Our experiments have used the framework OpenSpiel [Lanctot et al., 2019]. The depth-limited search was conducted with CPU Intel Xeon Scalable Gold 6146, operating at a frequency of 3.2 GHz, while the network training additionally used a single GPU Tesla V100. The hyperparameter setting is described in Appendix D. Appendix B provides a detailed description of the game rules.

5.1 Head-To-Head Play in Large Games

The first experiment shows the effectiveness of SePoT in large games. We trained RNA-D and RNA-D with SePoT separately, each three times, with the same hyperparameter setting. These were then used in head-to-head play against each other, where RNA-D directly followed the policy network and SePoT applied search in those subgames that contained less than 25000 or 100000 unique histories. The evaluation encompassed these games: Goofspiel with eight cards and 2500 unique histories, SePoT with eight cards and 1000 unique histories, and Battleships on square grids of size 3x3, 5x5, and 7x7 with ships of size (2, 3), (3, 2, 2), (4, 3, 3, 2) respectively.

For each game, we have trained 3 RNA-D policy networks and 3 different RNA-D policy networks along with SePoT’s history critic. Each network was trained for 3 days. We have run over one hundred thousand game simulations of RNA-D against SePoT for each game. In half of them, the SePoT was a first player; in half, it was a second player. The average rewards multiplied by 100 from these simulations are presented in Table 1 along with a 95% confidence interval. Employing search outperformed the RNA-Ds policy network with statistical significance in all the games tested.

The usage of search improves the performance in Battleships more than in Goofspiel. This is because, in Goofspiel, most of the subgames are quite large, so the search is used
In the second experiment, we aim to evaluate the effect of depth-limited search with multi-valued states on the quality of the policy compared to the policy network. In the experiments, we compare the exploitability of SePoT and the policy network from RNaD. We use Leduc hold’em and Goofspiel with 5 cards and randomized order of point cards for evaluation. We trained ten networks with different random seeds while computing exploitability every 500 iterations up to iteration 75,000. The results in Figure 3 show the mean and 95% confidence interval of the exploitability throughout these runs. In Leduc hold’em, we separately solve part before dealing a public card. In Goofspiel, we use different depth limits for search, where DL=n means that both players play n actions before the value function is called.

The plots in Figure 3b show how the exploitability changes with additional iterations. Using the depth limit 2 or 3 yields similar results, showing that both outperform depth limit 1 and the policy network. This experiment shows that with additional training, the search still improves over the network policy. Furthermore, it confirms the intuition that having a longer look-ahead increases the quality of the search solution.

Table 1: Average rewards from head-to-head play of RNaD against RNaD iterations ×10

<table>
<thead>
<tr>
<th>Max subgame size</th>
<th>Goofspiel 5 randomized</th>
<th>Goofspiel 13 randomized</th>
<th>Battleships 3x3 randomized</th>
<th>Battleships 5x5 randomized</th>
<th>Battleships 7x7 randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>4.10 ± 0.19</td>
<td>3.51 ± 0.11</td>
<td>1.43 ± 0.31</td>
<td>6.07 ± 0.09</td>
<td>6.5 ± 0.5</td>
</tr>
<tr>
<td>4350</td>
<td>4.0 ± 0.1</td>
<td>3.51 ± 0.11</td>
<td>1.43 ± 0.31</td>
<td>6.07 ± 0.09</td>
<td>6.5 ± 0.5</td>
</tr>
</tbody>
</table>

![Figure 3: Exploitability of policy network RNaD and the search with SePoT based on RNaD training iterations.](image)

Figure 3: Exploitability of policy network RNaD and the search with SePoT based on RNaD training iterations.

6 Related Work

Previous search algorithms require using safe search both in train and test time [Brown et al., 2020; Brown and Sandholm, 2018; Moravčík et al., 2017; Schmid et al., 2023]. Using those algorithms in larger games requires some subtree reduction techniques like abstracting the game [Brown and Sandholm, 2018] or revealing part of the private information [Zhang and Sandholm, 2021]. SePoT uses a policy-gradient training algorithm that trains in arbitrarily sized games. Furthermore, SePoT chooses whether the search should be used at each point independently during gameplay. This enables safe search in larger games than previously possible without using any subtree reduction techniques. However, SePoT is flexible enough to be used with these subtree reduction techniques to increase the number of subtrees that the search may be applied in even further.

Imagine a game of Battleships as an example. After placing the ships, there are more than 1020 possible states in a public state, which makes training of search algorithms infeasible. After several more rounds, the information is steadily being revealed to both players, and eventually, the size of the public state becomes small enough. SePoT can perform search in such a game, which was previously not possible.

Our method is heavily based on the [Brown et al., 2018], with the notable differences being a more principled way to train the blueprint policy with policy-gradient algorithms and choosing the policy transformations. Moreover, we show how to use multi-valued states as a value function practically. We present a way to leverage an already trained critic network to construct a gadget game anywhere in the game without the need to perform a search in previous parts.

Model-free deep reinforcement learning methods based on regret minimization [Steinberger et al., 2020; Gruslys et al., 2020; McAleer et al., 2023] use search in the gameplay and still can be scaled to very large games. The main limitation of these methods is the need to compute the average strategy, which either requires storing all the trajectories used in learning or approximating this average, which is prone to error. In games like Battleships, the training may consist of billions of trajectories to train properly. Storing all of these trajectories and computing averages is both memory and computationally demanding. SePoT uses a policy-gradient algorithm that approximates the optimal strategy in immediate strategies, so it does not compute the average strategy.

7 Conclusion

Policy-gradient algorithms proved to be powerful for solving huge problems in various settings. Recently, these algorithms were successfully applied to two-player zero-sum imperfect information games and outperformed human professionals in Stratego. However, these algorithms could not incorporate search to improve mistakes caused by network inaccuracies. We have presented an algorithm SePoT that can be used alongside any trajectory sampling policy-gradient algorithm in two-player zero-sum imperfect information games to train additional critic network without any search during training. We have shown that this network is sufficient to enable safe search during test time. The search we use differs from previous approaches since it is independent of the application of search in previous parts of the game, with the only limitation being the local size of the subgame and reconstruction of the public states. Thanks to this, SePoT can employ search even in larger games while still having a choice to follow policy from the policy network. Our experimental results demonstrate that SePoT is able to outperform RNaD in most of the tested scenarios.
Acknowledgments

This research is supported by Czech Science Foundation (GA22-26655S) and Grant Agency of the CTU in Prague (SGS22/168/OHK3/3T/13), and the Canadian Institute for Advanced Research. Computational resources were supplied by (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic and also (CZ.02.1.01/0.0/0.0/16 019/0000765).

References


[McAleer et al., 2023] Stephen Marcus McAleer, Gabriele Farina, Marc Lanctot, and Tuomas Sandholm. ESCHER:


