TEST-TIME REINFORCEMENT LEARNING IN IMPERFECT INFORMATION GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time reasoning has significantly improved performance in domains ranging from games to language models. However, principled test-time policy improvement remains a challenge in two-player zero-sum imperfect-information games. Existing solutions are limited to tabular methods or single gradient step updates. In this work, we investigate policy-gradient algorithms as a method for scalable test-time reasoning. We extend the concept of gadget games, a tabular technique for test-time search, to the reinforcement learning setting. Unlike prior approaches, we represent the gadget game implicitly by training a state sampler and virtual actor policy rather than explicitly by constructing the gadget, thereby removing constraints on subgame size. Furthermore, we prove that a fixed regularization policy bounds the strategy change within a subgame, making simpler Bayesian subgame solving viable without safe gadgets. In small games, test-time training substantially improves weak blueprints, while strong blueprints may degrade slightly. In large games, it improves over the blueprint in head-to-head play and generally outperforms the update-equivalence framework.

1 INTRODUCTION

Test-time reasoning enables algorithms to allocate limited computation to enhance performance within the context actually encountered during algorithm execution. This paradigm has been crucial to successes in game-playing (Silver et al., 2018; Brown & Sandholm, 2018; Moravčík et al., 2017; Brown & Sandholm, 2019; Schmid et al., 2023; Zhang & Sandholm, 2026; Sokota et al., 2025; FAIR et al., 2022) and it has demonstrated significant gains in large language models (Yang et al., 2024; Kojima et al., 2022; Snell et al., 2025).

In imperfect-information games, test-time reasoning is challenging. Since players cannot observe the underlying game state, they must reason across a distribution of possible states. An optimal strategy at any given decision depends on the player's beliefs over these states, which are themselves a function of the strategies employed by all players acting prior to that decision. Having any fixed belief regarding an opponent's past decisions has been shown to lead to highly exploitable strategies (Gilpin & Sandholm, 2006; Burch et al., 2014; Moravčík et al., 2016).

Gadget games were developed to employ additional reasoning in tabular game solving. We will focus on *Bayesian gadget game*, which assumes both players followed some fixed strategy in the past, and *resolving gadget game* that does not require any fixed beliefs about opponents' strategies. By starting with a *blueprint strategy* and refining it at each encountered decision point, an exactly solved resolving gadget guarantees that the resulting strategy will be no more exploitable than the blueprint (Burch et al., 2014; Moravčík et al., 2016; Brown & Sandholm, 2017). We call methods with this guarantee "safe", and techniques without it, such as Bayesian gadget games, "unsafe".

To date, gadget games and test-time reasoning have been largely restricted to tabular approaches. However, policy-gradient algorithms have emerged as a strong non-tabular alternative for game-playing, and were the first to achieve expert-level and then superhuman performance in the large, imperfect-information game Stratego (Perolat et al., 2022; Sokota et al., 2025). The superhuman Stratego bot Ataraxos used the "update-equivalence framework", a test-time reasoning technique that simulates a single policy-gradient step (Sokota et al., 2024; 2025). While this ensures that the strategy's exploitability does not degrade beyond a certain bound, it also limits potential gains by restricting reasoning to a single update.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we propose two policy-gradient methods for test-time reasoning. The first represents the resolving gadget implicitly, since its explicit construction is often infeasible. A small temporary network models the opponent's new gadget decisions, using blueprint values rather than best responses at termination. Unlike some prior approaches that used policy gradient during training, and then solved explicitly constructed gadget games using tabular algorithms (Kubíček et al., 2024; Kubíček & Lisý, 2026), both methods use the same algorithm for training and test-time reasoning, reducing design complexity and improving scalability. To sample the initial states of the subgame from the distribution required by gadget games, both use a transformer belief model.

Our second method is a Bayesian gadget with a fixed regularization policy. We show that it retains a bounded-change guarantee similar to that of the update-equivalence framework beyond a single gradient step, bounding both improvement and degradation within the subgame without a safe gadget.

We evaluate the proposed approaches both in small benchmark games, where exploitability can be precisely computed, and in large-scale games where tabular methods are computationally infeasible. In small games, both methods substantially improve weaker blueprints and slightly degrade strong ones, likely due to sampling noise. In large games, our Bayesian method wins over 80% in Battleship against an opponent without any subgame solving.

2 Background

The two-player zero-sum factored-observation stochastic game (FOSG) is a tuple $\mathcal{G} = (\mathcal{N}, \mathcal{W}, w^{\text{INIT}}, p, \mathcal{A}, \mathcal{T}, \mathcal{R}_1, \mathcal{O})$. Here, $\mathcal{N} = (1, 2, c)$ is the set of players, including a chance player c, $\mathcal{W}$ represents the set of underlying world states, and $w^{\text{INIT}} \in \mathcal{W}$ is the initial state. The player function $p : \mathcal{W} \to 2^{\mathcal{N}}$ identifies which players are active in a given state. $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_c$ denotes the set of joint actions, where $\mathcal{A}_i(w)$ is the set of legal actions for player i in state w. Transitions between world states are modeled by $\mathcal{T} : \mathcal{W} \times \mathcal{A} \to \mathcal{W}$ and rewards of player 1 by $\mathcal{R}_1 : \mathcal{W} \times \mathcal{A} \to \mathbb{R}$. In zero-sum games, the rewards given to player 2 are $\mathcal{R}_2 := -\mathcal{R}_1$. The observation function $\mathcal{O} : \mathcal{W} \times \mathcal{A} \times \mathcal{W} \to \mathbb{O}$ provides observations for all players. The observation set $\mathbb{O} = \mathbb{O}_0 \times \mathbb{O}_1 \times \mathbb{O}_2$ consists of public observations $\mathbb{O}_0$ and player-specific private observations $\mathbb{O}_i$. Similarly, the observation function can be factored as $\mathcal{O} = (\mathcal{O}_0, \mathcal{O}_1, \mathcal{O}_2)$ (Kovařík et al., 2022).

A history $h = w^0 a^0 \dots a^{l-1} w^l \in (\mathcal{W}\mathcal{A})^* \mathcal{W}$ is a finite sequence starting in $w^0 = w^{\text{INIT}}$ and at each timestep $t \leq l-1$ the transitions are consistent with the game rules $w^{t+1} = \mathcal{T}(w^t, \mathcal{A}^t)$. We will use $\mathcal{H}$ to denote the set of all possible histories in the game. For any 2 histories $h, h' \in \mathcal{H}$, if h is a prefix of h', we will use following notation $h \sqsubseteq h'$. Every history ends with a uniquely identifiable state, so we will sometimes use history instead of the world state, such as $\mathcal{R}_1(h, a) := \mathcal{R}_1(w^l, a)$ and $h^{\text{INIT}} := w^{\text{INIT}}$ will be the initial history. In imperfect-information games, players cannot observe the world state directly. Instead, histories that are indistinguishable to the player i are grouped into an information set (infoset) $s_i \in \mathcal{S}_i$. Moreover, the states that are indistinguishable from the perspective of the outside observer are in the same public state $s_0 \in \mathcal{S}_0$. A different perspective is that s_i is a collection of states in which the player i has the same private information, and s_0 is a collection of states in which the public information is the same. The sets of all information sets and public states are $\mathcal{S}_i$ and $\mathcal{S}_0$, respectively. We will overload the notation and use $\mathcal{H}(s_i)$ to denote all histories that share the information set, $\mathcal{S}_i(s_0)$ to denote all information sets that share the public state, $s_i(h)$ to denote infoset or public state that corresponds to history h. Since the player cannot distinguish states within the same infoset, it must have the same legal actions in every such state. We will use $\mathcal{A}_i(s_i) := \mathcal{A}_i(w)$, such that s_i contains w.

A behavioral strategy $\pi_i : \mathcal{S}_i \to \Delta\mathcal{A}_i$ of player i maps the information set to the probability distribution over legal actions. We will also use $\pi_i(s_i, a_i)$ to denote the probability of playing action a_i under strategy π_i. The joint strategy profile is then $\pi = (\pi_1, \pi_2, \pi_c)$, where π_c is always fixed by the game rules. Consider two histories h, h', such that $h \sqsubseteq h'$. The probability that the history h' is reached under π, from h is $P^\pi(h'|h) = \prod_{h \sqsubseteq h'' a w \sqsubseteq h'} \prod_{i \in \mathcal{N}} \pi_i(s_i(h''), a_i)$. This probability is often called reach. This can be decomposed into individual player contributions $P_i^\pi(h'|h) = \prod_{h \sqsubseteq h'' a w \sqsubseteq h'} \pi_i(s_i(h''), a_i)$. We will use a standard notation where $-i$ denotes all players except player i. We also use $P^\pi(h) = P^\pi(h|h^{\text{INIT}})$.

The expected utility for player i under strategy π is $u_i^\pi(h) = \sum_{h \sqsubseteq h' a} P^\pi(h'|h) \mathcal{R}_i(h', a) \prod_{j \in \mathcal{N}} \pi_j(s_j(h'), a_j)$. The counterfactual utility of the informa-

tion set $v_i^\pi(s_j) = \frac{\sum_{h \in \mathcal{H}(s_j)} P_{-j}^\pi(h) u_i^\pi(h)}{\sum_{h \in \mathcal{H}(s_j)} P_{-j}^\pi(h)}$ is the sum of expected utilities across all histories in an infoset, weighted by the probability that those histories are reached assuming player j played to reach them. The counterfactual action utility $q_i^\pi(s_j, a_j)$ is defined analogously to the counterfactual utility, but for infoset-action pairs. A best response $\pi_i^{BR} \in BR_i(\pi_{-i})$ maximizes utility against fixed opponent strategy $\pi_i^{BR} = \arg\max_{\pi_i} u_i^{(\pi_i, \pi_{-i})}(h^{\text{INIT}})$. A strategy profile where all players play a best response is a Nash equilibrium π^*, which is a sought-after solution concept in two-player zero-sum games. Exploitability $\mathcal{E}(\pi_i) = u_{-i}^{\pi_i, \pi_{-i}^{BR}}(h^{\text{INIT}}) - u_{-i}^{\pi^*}(h^{\text{INIT}})$ measure a strategy's distance from the Nash equilibrium. It is always non-negative, and it is 0 iff the π_i is part of the Nash equilibrium.

2.1 SUBGAME SOLVING

A subgame $\mathcal{G}^S$ is a restriction of the game $\mathcal{G}$, which starts from a collection of states $\mathcal{W}^S$, which correspond to histories $\mathcal{H}^S \subseteq \mathcal{H}$ from the original game $\mathcal{G}$. Based on the choice of $\mathcal{H}^S$, there are 2 main categories of subgame solving. *Common-knowledge subgame solving* considers all histories consistent with the public state $\mathcal{H}^S = \mathcal{H}(s_0)$ (Gilpin & Sandholm, 2006; Ganzfried & Sandholm, 2015; Burch et al., 2014; Moravčík et al., 2016; Brown & Sandholm, 2017; Kubíček et al., 2026). *Knowledge-limited subgame solving* (KLSS) considers only a subset of public state histories (Zhang & Sandholm, 2021; 2026; Liu et al., 2023; Sokota et al., 2024). Original KLSS "freezes" the strategy at the boundary information sets to equal the probabilities from the blueprint. We use a *knowledge-limited unfrozen subgame solving (KLUSS)*, which allows both players to optimize their strategies across all states within the subgame (Zhang & Sandholm, 2026). Specifically, we focus on 1-KLUSS, which considers histories consistent with the information set of a player $\mathcal{H}^S = \mathcal{H}(s_i)$.

The subgame in imperfect information games is not a well-defined game, as it does not start in a single state. In order to make it a well defined game, it is necessary to connect the disjointed initial states with some new states and actions. We will call these new states and actions that were not part of the original game as *gadget* and the resulting game *gadget game*.

Bayesian gadget game creates a new initial state, where chance plays to all states in $\mathcal{W}^S$ proportionally to reach probabilities $P^{\overline{\pi}}(h)$ of some blueprint strategy $\overline{\pi}$. Although often effective (Gilpin & Sandholm, 2006; 2007; Ganzfried & Sandholm, 2015; Brown & Sandholm, 2017; Kubíček et al., 2026), this approach can lead to high exploitability if the opponent does not play the strategy assumed in the blueprint even if $\overline{\pi}$ is a Nash equilibrium (Burch et al., 2014). *Resolving gadget game* address this by providing the opponent with a hypothetical choice to modify their previous strategy, ensuring that the resulting solution is not more exploitable than the original blueprint (Burch et al., 2014; Moravčík et al., 2016; Brown & Sandholm, 2017).

2.2 POLICY-GRADIENT ALGORITHMS FOR GAMES

Policy-gradient algorithms have shown remarkable performance in very large games like Stratego (Perolat et al., 2022; Sokota et al., 2025). To ensure convergence in two-player zero-sum games, those methods employ KL regularization with respect to some *magnet* (regularization) strategy π^M. In Regularized Nash Dynamics the regularized rewards are

$$\mathcal{R}_i^R(h, a) = \mathcal{R}_i(h, a) - \eta \log\left(\frac{\pi_i(s_i(h), a_i)}{\pi_i^M(s_i(h), a_i)}\right) + \eta \log\left(\frac{\pi_{-i}(s_{-i}(h), a_{-i})}{\pi_{-i}^M(s_{-i}(h), a_{-i})}\right) \tag{1}$$

The corresponding optimization objective for behavioral strategies as defined by Perolat et al. (2022) is

$$\pi_i^t(s_i, a_i) = \arg\max_{\pi_i} \sum_{a_i} \pi_i(s_i, a_i) q_i^{\pi^{t-1}}(s_i, a_i) - \eta D_{\text{KL}}(\pi_i, \pi_i^M) \tag{2}$$

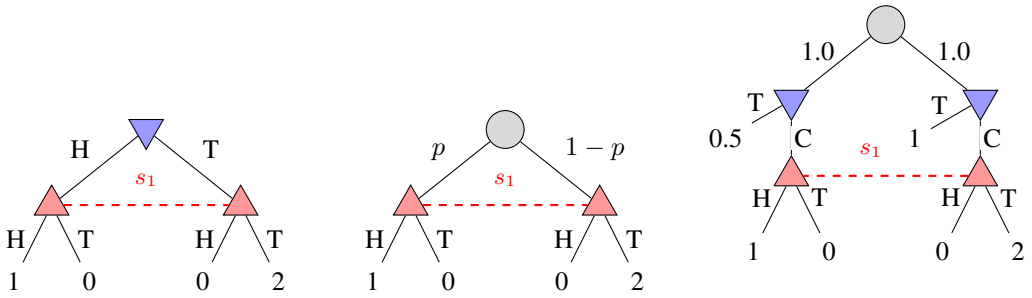
A higher regularization strength (η) forces the strategy to remain closer to the magnet. This regularization alters the Nash equilibrium of the game based on the magnet strategy. It has been shown that by either changing the magnet strategy to slowly follow the computed strategy or by annealing the η, these algorithms eventually converge to a Nash equilibrium when used on normal-form or

sequence-form strategies. Like prior theoretical results (Perolat et al., 2021; Sokota et al., 2023), our results in Theorems 2 and 3 apply to normal-form and sequence-form strategies. Still, the regularized policy-gradients exhibit surprisingly strong performance in large-scale games like Stratego even when applied to behavioral strategies (Sokota et al., 2025; Perolat et al., 2022). Recently, it has been shown that these convergence guarantees hold even in behavioral strategies settings (Kalogiannis & Farina, 2026).

3 IMPERFECT-INFORMATION SUBGAME SOLVING

In imperfect-information games, subgame solving is difficult because player strategies are interdependent across the entire game. We illustrate this challenge using a modified version of Matching Pennies, in which Player 1 receives a double reward if both players select Tails. This game has a single unique Nash equilibrium if both players play $\pi(H) = \frac{2}{3}$.

Figure 1a shows the sequential form of this game, where Player 2 acts first without revealing the choice to Player 1. Consider Bayesian gadget game for Player 1's decision with 2 nodes as illustrated in Figure 1b. The optimal strategy in this subgame depends on the belief p that the opponent played Heads. If $p > \frac{2}{3}$, Player 1 should always play Heads; If $p < \frac{2}{3}$, then Tails is optimal. Lastly, if $p = \frac{2}{3}$, then regardless of the strategy the Player 1 chooses, it will always have the same expected utility $\frac{2}{3}$, which means that any strategy of Player 1 in the Bayesian gadget is optimal.



(a) Original game (b) Bayesian gadget game (c) Resolving gadget game

Figure 1: Sequential form of Matching Pennies, where Player 2 acts first and the corresponding subgame and resolving gadget game of Player 1.

The danger is that the opponent can choose a different strategy, such as playing H, which makes the different strategies of Player 1 in the real game lead to very different payoffs. Some gadget games mitigate this risk by giving an opponent a choice to hypothetically select any strategy in the past. *Resolving gadget* inserts an additional opponent's decisions before each root state of the subgame. These new decisions respect the opponent's information sets and give the opponent a choice to either continue into the subgame or terminate the game for a fixed reward. Crucially, the probability distribution across the root nodes is proportional only to the reach of resolving and the chance player (Burch et al., 2014).

If the termination reward is the opponent's counterfactual best response value to some blueprint strategy $\overline{\pi}$, then every equilibrium of this gadget game is guaranteed to be no more exploitable in the full game than the blueprint. Moreover, solutions to gadget games have been empirically shown to often improve the blueprint (Burch et al., 2014; Brown & Sandholm, 2017; 2018; Moravčík et al., 2017; Kubíček et al., 2026).

Theorem 1. *(Moravčík et al., 2017; Burch et al., 2014; Kubíček et al., 2026) Consider a common-knowledge subgame $\mathcal{G}^{s_0}$ of game $\mathcal{G}$ starting in public state s_0. Let $\overline{\pi_i}$ be a blueprint strategy and $\mathcal{G}_i^{G,s_0}$ be a player i's resolving gadget game corresponding to the $\mathcal{G}^{s_0}$, where the terminate values correspond to the best response values to $\overline{\pi_i}$. Let π^* be a Nash equilibrium of $\mathcal{G}_i^{G,s_0}$ and π_i' strategy that plays π_i^* in a subgame $\mathcal{G}^{s_0}$ and blueprint $\overline{\pi}$ everywhere else. Then*

$$\mathcal{E}(\pi_i') \leq \mathcal{E}(\overline{\pi_i})$$

In Figure 1c we show the resolving gadget of the matching pennies subgame. The initial chance node assigns a probability of reaching the state equivalent to the reach of Player 1 and the chance. Since neither Player 1 nor the chance player has acted, the reach is 1. It is possible to normalize the chance node so that the probabilities sum to 1 without changing the optimal play in that game, but we kept them unnormalized so they reflect the reach. The reward of the terminating action corresponds to the blueprint $\overline{\pi_1}(H) = 0.5$.

3.1 Subgame solving by implicit gadget game

Prior work bridges subgame solving and policy-gradients by training auxiliary components to explicitly construct subgames, which are then solved using tabular methods, such as CFR (Kubíček et al., 2024; Kubíček & Lisý, 2026). However, this explicit construction of the complete subgame is often not computationally feasible.

Instead of explicit construction, we represent the resolving gadget game of player i implicitly by introducing a temporary actor π_Ψ with weights Ψ. This actor predicts the strategy for the initial gadget decision (terminate or continue). The training in the gadget game proceeds by first sampling the initial state proportionally to the reach of player i and the chance player. Then, it samples the trajectory until the end of the game using the current strategy $\pi_{\theta'}$. The q-value of the continue action is the estimated counterfactual value of the sampled initial state. The q-value of the terminate action is the critic's estimate using the blueprint strategy π_θ. These two q-values are then used to compute the policy-gradient update. Lastly, the loss used to train θ' is scaled by the probability of the continue action to ensure an unbiased update based on the opponent's gadget decision. We show this whole training procedure in Algorithm 1 in Appendix B.2.

Theorem 1 does not depend directly on the solver used, as long as it converges to a Nash equilibrium, and our approach only mirrors the gadget game's behavior. As a result, our technique is not limited to regularized policy-gradient algorithms and will be compatible with future policy-gradient algorithms that may offer better theoretical guarantees for behavioral strategies.

While the "terminate" reward should, in theory, be the counterfactual best response value, we use the critic's estimate of the value when both players follow the blueprint. Some prior works observed that this approach performs better in practice, because it is less conservative (Moravčík et al., 2017).

In nested subgame solving, gadget safety requires that counterfactual values and state samples remain consistent with the strategies used in previous subgames. Consequently, safety is only preserved if each subsequent resolve utilizes the weights derived from the preceding resolve.

3.2 Subgame solving with a fixed magnet

The most successful applications of policy-gradients in imperfect-information games typically employ KL regularization, which prevents the strategy from deviating too far from a regularization strategy π^M, sometimes referred to as a magnet (Perolat et al., 2021; 2022; Sokota et al., 2023; 2025). This regularization offers a different guarantee, which is not present in tabular solvers like CFR: the regularization strength η explicitly bounds how far the strategy can shift.

Theorem 2. *Given a game $\mathcal{G}$ and 1-strongly convex regularizer ψ with respect to $||\cdot||$. Let π^M be a magnet strategy profile with exploitability ϵ, $\eta > 0$ be a regularization weight. When solving a game $\mathcal{G}$ with regularized rewards, the Bregman divergence between the Nash strategy profile of the regularized game and the magnet is at most*

$$B_\psi(\pi_1^\eta, \pi_1^M) + B_\psi(\pi_2^\eta, \pi_2^M) \leq \frac{\epsilon}{\eta}$$

Theorem 3. *Given a game $\mathcal{G}$ with rewards represented by a matrix $\mathcal{R}$ and entropy as a regularizer ψ. Let π^M be a magnet strategy profile with exploitability ϵ, $\eta > 0$ be a regularization weight. When solving a game $\mathcal{G}$ with regularized rewards, the exploitability of the Nash strategy profile of the regularized game in the unregularized game $\mathcal{G}$ is at most*

$$\mathcal{E}(\pi^\eta) \leq \epsilon + ||\mathcal{R}||_\infty \sqrt{\frac{4\epsilon}{\eta}}$$

Proofs are provided in Appendix A. Theorems 2 and 3 describe the maximal possible strategy change for a fixed magnet π^M. Specifically, Theorem 3 describes the change in exploitability when the algorithm is used in the entire game to converge. When the fixed magnet is used in a Bayesian subgame, the exploitability is bounded in that subgame, but when the same strategy is used in the rest of the game, the exploitability can increase beyond this bound, which we show in Appendix C.1. However, in practice, a strong blueprint $\overline{\pi}$ often implies low exploitability within each individual Bayesian subgame, which, based on Theorem 2, bounds how much the strategy can change. So when the blueprint is near-optimal, the strategy changes in the subgame are outweighed by the KL penalty. Consequently, even when using the unsafe techniques with regularized policy-gradient algorithms, the strategy does not converge arbitrarily far from the magnet.

The update-equivalence framework offers a similar perspective by simulating a single step of the policy gradient algorithm at each decision (Sokota et al., 2024). However, the change in exploitability in the update-equivalence framework is limited by the step size. In contrast, we show that this bound is maintained even upon convergence, provided the magnet is fixed. Still, both approaches may increase exploitability in the worst case.

Another surprising implication of Theorem 2 is that it bounds the strategy change both in common-knowledge and knowledge-limited subgame solving. This is unlike the safety guaranteed by gadget games, which requires additional constraints to be extended to KLSS (Zhang & Sandholm, 2021; Liu et al., 2023).

4 GENERATING STATES

While the previous sections focus on strategy optimization, those methods require sampling states proportional to a specific probability distribution. As strategies are updated during subsequent resolves, this distribution shifts, requiring a belief model that dynamically changes along the strategy.

Prior approaches typically relied on explicit state enumeration using game rules (Gilpin & Sandholm, 2006; Ganzfried & Sandholm, 2015; Brown et al., 2018; Moravčík et al., 2017; Brown et al., 2020; Schmid et al., 2023; Zhang & Sandholm, 2026; Kubíček et al., 2024) or some form of abstraction that could be both trained or predefined (Brown & Sandholm, 2018; Kubíček & Lisý, 2026; Solinas et al., 2025). More recently, belief models have been trained to sample states using a fixed strategy and potentially domain-specific knowledge (Sokota et al., 2024; 2025). We propose a general approach: a generative model that learns to sample subgame states conditioned on available information.

Our belief model utilizes a decoder-only Transformer (Vaswani et al., 2017). Rather than predicting a state representation directly, the model auto-regressively generates sequences of actions, including chance events, that lead to a state. Each action is represented as a discrete token. To ensure samples are consistent with the current context (information), we prepend either the infoset s_i (for KLSS), or the public state s_0 (for common-knowledge) as the initial token. The model then predicts the probability distribution over subsequent actions in the sequence.

The training target for the belief model depends on the subgame solving type. When using Bayesian gadget, the model must sample states proportional to $P^{\pi_\theta}(w|s)$, the probability of reaching state w conditioned on the s, which is either an information set or public state, under strategy π_θ. We train this model by sampling trajectories using π_θ and applying standard cross-entropy loss to the generated action sequence.

For a resolving gadget of player i, the model should sample states proportionally to $P_i^{\pi_\theta}(w|s)P_c^{\pi_\theta}(w|s)$. Since standard trajectory sampling follows full reach $P^{\pi_\theta}(w|s) = P_1^{\pi_\theta}(w|s)P_2^{\pi_\theta}(w|s)P_c^{\pi_\theta}(w|s)$, which includes the opponent's j strategy, we apply the importance sampling correction $\frac{1}{P_j^{\pi_\theta}(w|s)}$ to remove bias from the samples of the opponent's specific strategy.

In a nested subgame-solving setting, the strategy is updated at every encountered decision point, while respecting past changes. This means that even the belief model needs to reflect those changes. During subgame solving, we train the belief model using the policy-gradient algorithm, ensuring it reflects changes in the strategy. Failing to do this joint training would result in a belief-strategy mismatch, where the strategy is optimized against an outdated distribution, potentially leading to significant bias in long games.

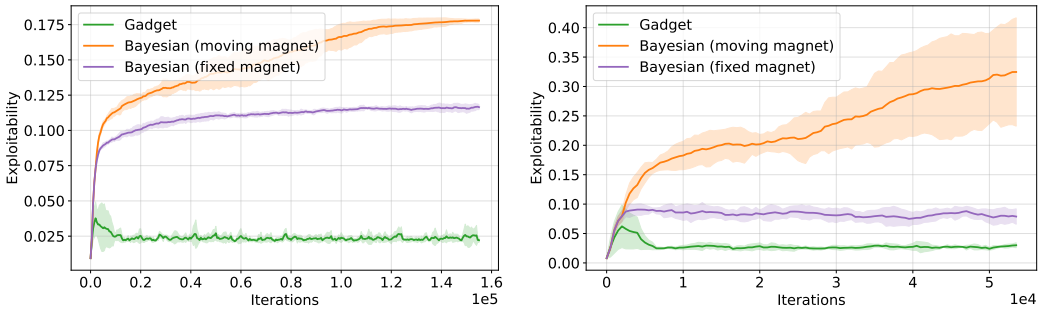
5 EXPERIMENTS

We have evaluated various subgame solving techniques using Regularized Nash Dynamics (RNaD) with Neural Replicator Dynamics as the policy-gradient algorithm. The goal was to assess how these techniques influence exploitability in small-scale games and how they scale to large environments. Detailed information is in Appendix B

5.1 ONE-SHOT GAMES

To illustrate Theorems 1 and 3, we solved subgames using either the Bayesian or the gadget approach. For Bayesian solving, we compared two configurations: either a moving magnet, where the magnet moves the same way as during full-game training, or a fixed magnet, aligning with constraints in Theorem 3. We tested these on two normal-form games, biased Matching Pennies (Figure 1a) and Rock-Paper-Scissors. We assumed that the players move sequentially in this game, as in the example in Figure 1a, and, as a result, each player has a single subgame. In both cases, the blueprint strategy was close to the Nash equilibrium $\mathcal{E}(\overline{\pi}) < 0.02$, which is the hardest setting for the subgame solving, because any small noise results in performance degradation. Consider a situation where the blueprint assigns a slightly higher probability to playing Paper than to other actions. Then the Bayesian solving would converge to a strategy that mixes between Scissors and Paper, which in turn can be exploited by playing Scissors.

Results in Figure 2 confirm that a fixed magnet prevents the strategy from diverging too far from the equilibrium. In contrast, when moving the magnet, the strategy will gradually converge to the best response, which is exploitable in the full game. When using the gadget game, exploitability first increases before stabilizing. This happens because the gadget actor is initialized randomly and must first learn the strategy for the gadget decision. However, after this strategy is learned, the exploitability remains low, though it slightly increases compared to the blueprint. This increase is due to the approximation noise in the critic's predicted utilities for the gadget's "terminate" action.



(a) Biased matching pennies

(b) Rock-Paper-Scissors

Figure 2: Exploitability with 95% confidence intervals in one-shot games, when solving a single subgame with different techniques using the same policy-gradient algorithm as the solver.

5.2 SMALL GAMES

We next evaluated exploitability across full games by applying test-time reasoning at every decision point and then merging all subgame strategies into a single strategy for the whole game. We compared solutions of resolving gadget, common-knowledge Bayesian gadget, KLSS Bayesian gadget, and the Update Equivalence Framework (UEF), the method employed by the superhuman Stratego bot Ataraxos (Sokota et al., 2025). Since UEF is time-oblivious, we slightly modified it to use the provided time to collect traces, thereby obtaining a better estimate of Q-values for its update. When solving the Bayesian subgames, we kept the magnet fixed to be in line with Theorem 2.

Since later reasoning depends on the strategy from earlier subgames, we traversed the game using breadth-first search, and at each subgame we used the weights from the corresponding decision in the previous layer. We tested three standard benchmark games: Imperfect Information Goofspiel

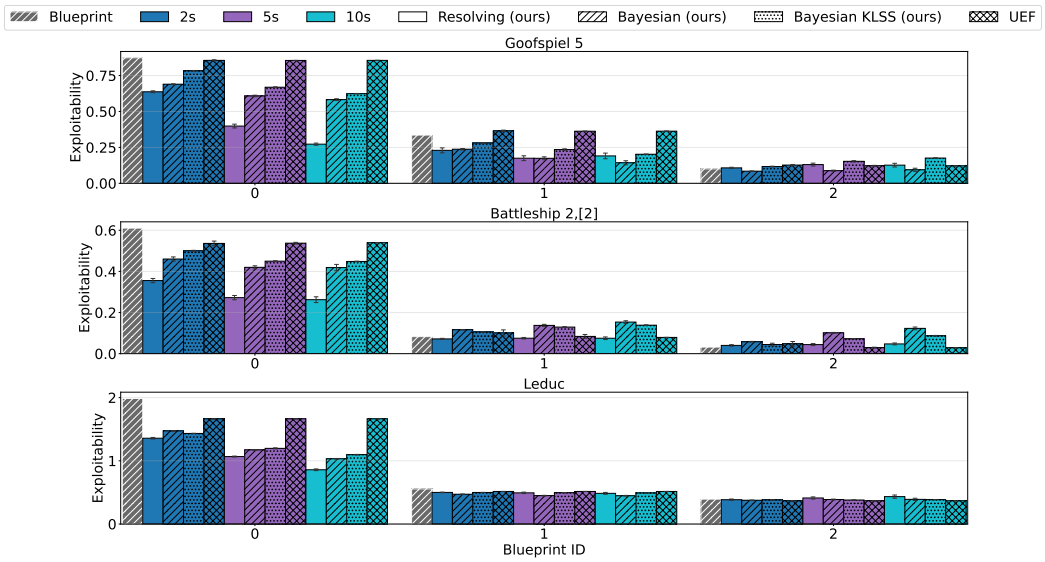


Figure 3: Exploitability with 95% confidence interval after performing different types of test-time reasoning in every subgame with the same policy-gradient algorithm for a limited time.

played with 5 cards, Leduc hold'em, and Battleship on 2x2 board with single ship of size 2, using three distinct blueprints for each. The first blueprint corresponds to the strategy near initialization of the weights, the second to the middle of the training, which will be the most common type of blueprint in large games, and the third, which is the least exploitable blueprint, to which the Regularized Nash Dynamics converged during training.

As shown in Figure 3, all techniques improve upon the blueprint when the blueprint is highly exploitable. Surprisingly, resolving gadget outperform other techniques with these blueprints, which is in contrast to the tabular settings, where Bayesian gadget is often superior (Brown & Sandholm, 2017; Kubíček et al., 2026). This advantage stems partially from the fixed-magnet constraint in Bayesian gadget and the fact that RNaD naturally encourages non-zero action probabilities, which has recently been shown to converge to stronger equilibria in gadget games (Kubíček et al., 2026).

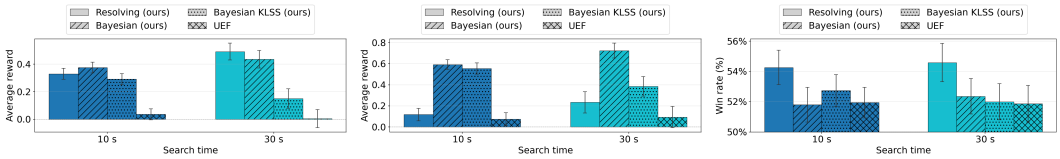
Using the strongest blueprints in Goofspiel and Leduc, the solutions of resolving gadget seem to slightly increase exploitability even with extended train time. We hypothesize that there are two main reasons for this degradation. 1. Since the safety of gadget games is only guaranteed upon convergence, the allocated computation time was insufficient to reach equilibrium in earlier subgames, leading to compounding errors in the merged strategy. 2. The additional test-time training contains noise from several sources: the critic for the "terminate" action, samples from the belief model, and sampling of the trajectories.

5.3 Large games

Finally, we evaluated the same techniques in large-scale games where computing exact exploitability is intractable. We measured performance via head-to-head play against a baseline RNaD strategy. Both the subgame solver and the blueprint were allocated equal additional computation time at each decision point. The additional blueprint time was spent on further training the blueprint from the game's initial state. To ensure that the additional test-time reasoning does not exploit the knowledge of the opponent's strategy, we have always matched different training seeds.

The goal of this experiment is to verify that additional test-time reasoning improves upon the blueprint. We do not try to argue or show that this approach outperforms all prior techniques applicable to the used games.

A key implementation detail is that we reset the updated parameters to the blueprint after every playthrough. Early experiments suggested that without resetting, updates in one subgame would "leak" into others via shared neural network parameters, degrading global performance. Making



(a) Battleship 7,[4,3,3,2] (b) Battleship 10,[5,4,3,3,2] (c) Goofspiel 10

Figure 4: Head-to-head average reward with 95% confidence intervals of different subgame solving techniques against the blueprint from different training seeds.

these updates persistent without corrupting the global strategy remains a promising direction for future work. Even in tabular settings, the resolved strategy is used only temporarily and it does not update blueprint.

The results in Figure 4 show that all methods outperform UEF. In Battleship, the UEF offered only marginal gains, likely because it is restricted to a single-step update. While Bayesian gadget is theoretically unsafe, it frequently outperformed the resolving gadget. This may be because resolving gadget weights the loss by the probability of "continue" action in the gadget decision, which effectively acts as a reduced learning rate. While common-knowledge approaches benefited from increased computation, KLSS appears to be hindered by it. Although head-to-head comparisons are not a perfect proxy for exploitability, these experiments demonstrate the potential for scalable test-time reasoning in large domains.

6 Conclusion and Discussion

In this work, we have presented a scalable framework for test-time reasoning in two-player zero-sum imperfect information games using policy-gradient algorithms. By extending the concept of gadget games to the reinforcement learning setting and introducing a generative belief model, we have removed the reliance on explicit subgame construction and tabular solvers. This allows test-time reasoning to scale to games, where subgame states may not be enumerable. Furthermore, because our approach only slightly modifies the core training algorithm, it significantly reduces design complexity compared to hybrid tabular-RL methods.

We demonstrated that resolving gadget game can be implicitly represented using a temporary actor that models the gadget's decision. We also established that KL regularization in modern policy-gradient algorithms serves as an implicit anchor, bounding how much the strategy changes, which makes simpler Bayesian solving viable.

We empirically evaluated these approaches in both small and large games, verifying that even in this setting, the additional test-time reasoning improves the performance over the blueprint. Notably, these techniques often yielded greater performance gains than the Update-equivalence framework (UEF) utilized in the superhuman Stratego bot Ataraxos (Sokota et al., 2024; 2025).

We observed that the shared network parameters necessitate resetting weights after each playthrough. This highlights a potential for future research that would enable persistent local updates, allowing agents to retain test-time insights across playthroughs without degrading global strategy.

Another promising direction is to combine our approach with depth-limited solving to allow faster gradient updates in long games. Prior work has integrated an approximate imperfect-information value function with policy-gradient algorithms (Kubíček et al., 2024; Kubíček & Lisý, 2026). We intentionally omitted such value functions to isolate the effects of test-time reasoning from the potential noise of value approximation.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to assist in constructing and writing the proofs. Specifically, Claude Sonnet 4.6 (with Claude Code) and Cursor Composer 2 were used during the implementation of the algorithms and experiments, and Claude Sonnet 4.6 and Gemini 3 were used to help construct the proofs in the paper. We have not used generative AI tools to introduce novel ideas into the paper, formulate mathematical claims, propose hypotheses, design experiments, interpret results, or generate synthetic datasets. Cleaning and reformatting datasets, translation, and qualitative data analysis do not apply to this work. Additionally, we used Claude Sonnet 4.6 and Gemini 3 to refine the writing of this paper, and Claude Opus 5.5 (with Claude Code) to port the manuscript to the ICLR template and format references. We have reviewed all AI-assisted work: all outputs from those models have been double-checked before inclusion in the final version. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018.

Noam Brown and Tuomas Sandholm. Safe and Nested Subgame Solving for Imperfect-Information Games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Noam Brown, Tuomas Sandholm, and Brandon Amos. Depth-limited solving for imperfect-information games. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

FAIR, Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022.

Sam Ganzfried and Tuomas Sandholm. Endgame Solving in Large Imperfect-Information Games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

Andrew Gilpin and Tuomas Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

Andrew Gilpin and Tuomas Sandholm. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2007.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024.

Fivos Kalogiannis and Gabriele Farina. Policy gradient methods converge globally in imperfect-information extensive-form games. *Advances in Neural Information Processing Systems*, 38: 29162–29224, 2026.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Vojtěch Kovařík, Martin Schmid, Neil Burch, Michael Bowling, and Viliam Lisý. Rethinking formal models of partially observable multiagent decision making. *Artificial Intelligence*, 303:103645, 2022.

Ondřej Kubíček and Viliam Lisý. Look-ahead Reasoning with a Learned Model in Imperfect Information Games. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Ondřej Kubíček, Neil Burch, and Viliam Lisý. Look-ahead search on top of policy networks in imperfect information games. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Ondřej Kubíček, Viliam Lisý, and Tuomas Sandholm. Equilibrium Refinements Improve Subgame Solving in Imperfect-Information Games. In *arXiv preprint*, 2026.

Weiming Liu, Haobo Fu, Qiang Fu, and Yang Wei. Opponent-limited online search for imperfect information games. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Matej Moravčík, Martin Schmid, Karel Ha, Milan Hladik, and Stephen Gaukrodger. Refining subgames in large imperfect information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. From Poincaré Recurrence to Convergence in Imperfect Information Games: Finding Equilibrium via Regularization. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623): 990–996, 2022.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G. Zacharias Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. Student of Games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Samuel Sokota, Ryan D'Orazio, J. Zico Kolter, Nicolas Loizou, Marc Lanctot, Ioannis Mitliagkas, Noam Brown, and Christian Kroer. A Unified Approach to Reinforcement Learning, Quantal Response Equilibria, and Two-Player Zero-Sum Games. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

Samuel Sokota, Gabriele Farina, David J. Wu, Hengyuan Hu, Kevin A. Wang, J. Zico Kolter, and Noam Brown. The Update-Equivalence Framework for Decision-Time Planning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Samuel Sokota, Eugene Vinitsky, Hengyuan Hu, J. Zico Kolter, and Gabriele Farina. Superhuman AI for Stratego Using Self-Play Reinforcement Learning and Test-Time Search. In *arXiv preprint*, 2025.

Christopher Solinas, Radovan Haluska, David Sychrovsky, Finbarr Timbers, Nolan Bard, Michael Buro, Martin Schmid, Nathan R. Sturtevant, and Michael Bowling. Neural Bayesian Filtering. In *arXiv preprint*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Brian Hu Zhang and Tuomas Sandholm. Subgame solving without common knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Brian Hu Zhang and Tuomas Sandholm. General search techniques without common knowledge for imperfect-information games, and application to superhuman Fog-of-War chess. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

A PROOFS

Theorem 2. *Given a game $\mathcal{G}$ and 1-strongly convex regularizer ψ with respect to $||\cdot||$. Let π^M be a magnet strategy profile with exploitability ϵ, $\eta > 0$ be a regularization weight. When solving a game $\mathcal{G}$ with regularized rewards, the Bregman divergence between the Nash strategy profile of the regularized game and the magnet is at most*

$$B_\psi(\pi_1^\eta, \pi_1^M) + B_\psi(\pi_2^\eta, \pi_2^M) \leq \frac{\epsilon}{\eta}$$

We will follow the Magnetic Mirror Descent approach, which defines the regularization with respect to the Bregman divergence B, which is more general than the KL divergence. We will denote $\mathcal{R}$ as the reward matrix corresponding to Equation (1). The strategies used throughout this proof refer to either the normal-form strategies, not the behavioral strategies. When using the fixed magnet $\pi^M = (\pi_1^M, \pi_2^M)$, the regularized objective can then be written of as

$$V(\pi_1, \pi_2) = \pi_1^T \mathcal{R} \pi_2 - \eta B_\psi(\pi_1, \pi_1^M) + \eta B_\psi(\pi_2, \pi_2^M) \tag{3}$$

For the remainder of this section, we will not denote the transposition of the strategy vector of player 1, when it is multiplied with the reward matrix, which will simplify the notation, but we will still treat it as transposed vector. and the solution of such regularized game is

$$\max_{\pi_1} \min_{\pi_2} V(\pi_1, \pi_2) \tag{4}$$

The fixed point solution π_1^η, π_2^η of this minmax problem satisfies these conditions for each π_1 and π_2

$$\langle \nabla_{\pi_1} V(\pi_1^\eta, \pi_2^\eta), \pi_1 - \pi_1^\eta \rangle \leq 0 \tag{5}$$
$$-\langle \nabla_{\pi_2} V(\pi_1^\eta, \pi_2^\eta), \pi_2 - \pi_2^\eta \rangle \leq 0 \tag{6}$$

We can compute the gradient in Equation (5)

$$\nabla_{\pi_1} V(\pi_1, \pi_2) = \mathcal{R}\pi_2 - \eta \nabla_{\pi_1} B_\psi(\pi_1, \pi_1^M) = \mathcal{R}\pi_2 - \eta \nabla \psi(\pi_1) + \eta \nabla \psi(\pi_1^M) \tag{7}$$
$$\begin{aligned} \langle \nabla_{\pi_1} V(\pi_1^\eta, \pi_2^\eta), \pi_1 - \pi_1^\eta \rangle &= \langle \mathcal{R}\pi_2^\eta - \eta \nabla \psi(\pi_1^\eta) + \eta \nabla \psi(\pi_1^M), \pi_1 - \pi_1^\eta \rangle \\ &= \langle \mathcal{R}\pi_2^\eta, \pi_1 - \pi_1^\eta \rangle - \eta \langle \nabla \psi(\pi_1^\eta) - \nabla \psi(\pi_1^M), \pi_1 - \pi_1^\eta \rangle \end{aligned} \tag{8}$$

We will use the Bregman three-point property $B_\psi(x, z) = B_\psi(x, y) + B_\psi(y, z) + \langle \nabla \psi(y) - \nabla \psi(z), x - y \rangle$

$$B_\psi(\pi_1, \pi_1^M) = B_\psi(\pi_1, \pi_1^\eta) + B_\psi(\pi_1^\eta, \pi_1^M) + \langle \nabla \psi(\pi_1^\eta) - \nabla \psi(\pi_1^M), \pi_1 - \pi_1^\eta \rangle$$
$$\langle \nabla \psi(\pi_1^\eta) - \nabla \psi(\pi_1^M), \pi_1 - \pi_1^\eta \rangle = B_\psi(\pi_1, \pi_1^M) - B_\psi(\pi_1, \pi_1^\eta) - B_\psi(\pi_1^\eta, \pi_1^M) \tag{9}$$
$$\langle \nabla_{\pi_1} V(\pi_1^\eta, \pi_2^\eta), \pi_1 - \pi_1^\eta \rangle =$$
$$= \langle \mathcal{R}\pi_2^\eta, \pi_1 - \pi_1^\eta \rangle - \eta (B_\psi(\pi_1, \pi_1^M) - B_\psi(\pi_1, \pi_1^\eta) - B_\psi(\pi_1^\eta, \pi_1^M)) \leq 0 \tag{10}$$
$$\Rightarrow \pi_1 \mathcal{R} \pi_2^\eta - \pi_1^\eta \mathcal{R} \pi_2^\eta \leq \eta (B_\psi(\pi_1, \pi_1^M) - B_\psi(\pi_1, \pi_1^\eta) - B_\psi(\pi_1^\eta, \pi_1^M)) \tag{11}$$

Similarly, we will derive for the second constraint

$$\begin{aligned} &- \langle \nabla_{\pi_2} V(\pi_1^\eta, \pi_2^\eta), \pi_2 - \pi_2^\eta \rangle \\ &= -\langle \pi_1^\eta \mathcal{R}, \pi_2 - \pi_2^\eta \rangle - \eta \langle \nabla \psi(\pi_2^\eta) - \nabla \psi(\pi_2^M), \pi_2 - \pi_2^\eta \rangle \\ &= -\langle \pi_1^\eta \mathcal{R}, \pi_2 - \pi_2^\eta \rangle - \eta (B_\psi(\pi_2, \pi_2^M) - B_\psi(\pi_2, \pi_2^\eta) - B_\psi(\pi_2^\eta, \pi_2^M)) \leq 0 \end{aligned} \tag{12}$$
$$\Rightarrow -\pi_1^\eta \mathcal{R} \pi_2 + \pi_1^\eta \mathcal{R} \pi_2^\eta \leq \eta (B_\psi(\pi_2, \pi_2^M) - B_\psi(\pi_2, \pi_2^\eta) - B_\psi(\pi_2^\eta, \pi_2^M)) \tag{13}$$

We can now sum those 2 inequalities

$$\begin{aligned} \pi_1 \mathcal{R} \pi_2^\eta - \pi_1^\eta \mathcal{R} \pi_2 \leq \eta \big(&B_\psi(\pi_1, \pi_1^M) + B_\psi(\pi_2, \pi_2^M) + \\ &- B_\psi(\pi_1, \pi_1^\eta) - B_\psi(\pi_2, \pi_2^\eta) - B_\psi(\pi_1^\eta, \pi_1^M) - B_\psi(\pi_2^\eta, \pi_2^M) \big) \end{aligned} \tag{14}$$

The Equation (14) holds for every $\pi = (\pi_1, \pi_2)$, so it has to holds even for $\pi = \pi^M$ and $B_{x,x} = 0$

$$\pi_1^M \mathcal{R} \pi_2^\eta - \pi_1^\eta \mathcal{R} \pi_2^M \leq \eta\big(- B_\psi(\pi_1^M, \pi_1^\eta) - B_\psi(\pi_2^M, \pi_2^\eta) - B_\psi(\pi_1^\eta, \pi_1^M) - B_\psi(\pi_2^\eta, \pi_2^M)\big) \quad (15)$$

$$\begin{aligned} B_\psi(\pi_1^M, \pi_1^\eta) + B_\psi(\pi_2^M, \pi_2^\eta) + B_\psi(\pi_1^\eta, \pi_1^M) + B_\psi(\pi_2^\eta, \pi_2^M) & \\ \leq \frac{1}{\eta}(\pi_1^\eta \mathcal{R} \pi_2^M - \pi_1^M \mathcal{R} \pi_2^\eta) & \\ \leq \frac{1}{\eta}(\max_{\pi_1} \pi_1 \mathcal{R} \pi_2^M - \min_{\pi_2} \pi_1^M \mathcal{R} \pi_2) = \frac{\mathcal{E}(\pi^M)}{\eta} & \quad (16) \end{aligned}$$

We know that the Bregman divergence is always non-negative, which allows us to get a bound on the resulting KL diergence for the strategy profile

$$B_\psi(\pi_1^\eta, \pi_1^M) + B_\psi(\pi_2^\eta, \pi_2^M) \leq \frac{\mathcal{E}(\pi^M)}{\eta} \quad (17)$$

Theorem 3. *Given a game $\mathcal{G}$ with rewards represented by a matrix $\mathcal{R}$ and entropy as a regularizer ψ. Let π^M be a magnet strategy profile with exploitability ϵ, $\eta > 0$ be a regularization weight. When solving a game $\mathcal{G}$ with regularized rewards, the exploitability of the Nash strategy profile of the regularized game in the unregularized game $\mathcal{G}$ is at most*

$$\mathcal{E}(\pi^\eta) \leq \epsilon + ||\mathcal{R}||_\infty \sqrt{\frac{4\epsilon}{\eta}}$$

Proof. Exploitability of the π^η is

$$\begin{aligned} \mathcal{E}(\pi^\eta) &= \pi_1^{BR} \mathcal{R} \pi_2^\eta - \pi_1^\eta \mathcal{R} \pi_2^{BR} \\ &= \pi_1^{BR} \mathcal{R} \pi_2^\eta - \pi_1^\eta \mathcal{R} \pi_2^{BR} + \pi_1^{BR} \mathcal{R} \pi_2^M - \pi_1^{BR} \mathcal{R} \pi_2^M + \pi_1^M \mathcal{R} \pi_2^{BR} - \pi_1^M \mathcal{R} \pi_2^{BR} \\ &= \pi_1^{BR} \mathcal{R} \pi_2^\eta - \pi_1^{BR} \mathcal{R} \pi_2^M + \pi_1^{BR} \mathcal{R} \pi_2^M - \pi_1^M \mathcal{R} \pi_2^{BR} + \pi_1^M \mathcal{R} \pi_2^{BR} - \pi_1^\eta \mathcal{R} \pi_2^{BR} \\ &= \pi_1^{BR} \mathcal{R} (\pi_2^\eta - \pi_2^M) + (\pi_1^{BR} \mathcal{R} \pi_2^M - \pi_1^M \mathcal{R} \pi_2^{BR}) + (\pi_1^M - \pi_1^\eta) \mathcal{R} \pi_2^{BR} \quad (18) \end{aligned}$$

We can now use following bound for middle terms

$$\pi_1^{BR} \mathcal{R} \pi_2^M - \pi_1^M \mathcal{R} \pi_2^{BR} \leq \max_{\pi_1} \pi_1 \mathcal{R} \pi_2^M - \min_{\pi_2} \pi_1^M \mathcal{R} \pi_2 = \mathcal{E}(\pi^M) \quad (19)$$

We can bound the other two terms by first using Hölder's inequality and then Pinkser's inequality

$$\begin{aligned} \pi_1^{BR} \mathcal{R} (\pi_2^\eta - \pi_2^M) &\leq |\pi_1^{BR} \mathcal{R} (\pi_2^\eta - \pi_2^M)| \\ &\leq ||\pi_1^{BR} \mathcal{R}||_\infty |(\pi_2^\eta - \pi_2^M)| \\ &\leq ||\mathcal{R}||_\infty |(\pi_2^\eta - \pi_2^M)| \\ &\leq ||\mathcal{R}||_\infty \sqrt{2 D_{\mathrm{KL}}(\pi_2^\eta, \pi_2^M)} \quad (20) \end{aligned}$$

$$(\pi_1^M - \pi_1^\eta) \mathcal{R} \pi_2^{BR} \leq ||\mathcal{R}||_\infty \sqrt{2 D_{\mathrm{KL}}(\pi_1^\eta, \pi_1^M)} \quad (21)$$

We can further use inequality $\sqrt{x} + \sqrt{y} \leq \sqrt{2}\sqrt{x+y}$

$$\begin{aligned} \mathcal{E}(\pi^\eta) &\leq \mathcal{E}(\pi^M) + ||\mathcal{R}||_\infty \sqrt{2 D_{\mathrm{KL}}(\pi_1^\eta, \pi_1^M)} + ||\mathcal{R}||_\infty \sqrt{2 D_{\mathrm{KL}}(\pi_2^\eta, \pi_2^M)} \quad (22) \\ &\leq \mathcal{E}(\pi^M) + ||\mathcal{R}||_\infty \sqrt{2 \cdot (2 D_{\mathrm{KL}}(\pi_1^\eta, \pi_1^M) + 2 D_{\mathrm{KL}}(\pi_2^\eta, \pi_2^M))} \quad (23) \end{aligned}$$

Since the used regularizer ψ is entropy, we can use Equation (17) to show

$$\mathcal{E}(\pi^\eta) \leq \mathcal{E}(\pi^M) + ||\mathcal{R}||_\infty \sqrt{\frac{4\mathcal{E}(\pi^M)}{\eta}} \quad (24)$$

□

B EXPERIMENTAL DETAILS

The codebase for this paper is written in Python 3.13, with libraries from the Jax ecosystem (Bradbury et al., 2018; Heek et al., 2024; DeepMind et al., 2020). Every experiment run used a single core of an AMD EPYC 7543 with 16 GB of RAM and a single Nvidia A100.

B.1 GAME RULES

Biased Matching Pennies Biased Matching Pennies is a one-shot game, in which players choose a side of a coin, either Heads or Tails. If the choices match, Player 1 wins; otherwise, Player 2 wins. In this version, the rewards are doubled if both choose Tails. The normal-form representation of this game is in Table 1a. In our experiments, we treat this as a sequential game in which Player 2 acts first, followed by Player 1, who did not observe Player 2's choice.

Rock-Paper-Scissors Rock-Paper-Scissors is a one-shot game where each player selects one of three actions: Rock, Paper, or Scissors. An identical choice results in a draw, rock defeats scissors, paper defeats rock, and scissors defeats paper. The normal-form representation is in Table 1b. As with Matching Pennies, we implement this sequentially, with Player 2 acting first and Player 1 acting with no knowledge of the opponent's selection.

Table 1: Normal-form representation of the games used in Section 5.1

(a) Biased Matching Pennies

		P2 H	P2 T
P1	H	1	0
P1	T	0	2

(b) Rock-Paper-Scissors

		P2 R	P2 P	P2 S
P1	R	0	-1	1
P1	P	1	0	-1
P1	S	-1	1	0

Battleship Battleship S, L is a two-player game which is played on two private $S \times S$ grids. Each player has a set of ships L of varying lengths (width is always 1). In the first phase of the game, players secretly place their ships on their respective grids, so that no ship can be placed in the 8-neighborhood of any other ship. Then, in the second phase, players take turns firing shots at the opponent's board, receiving only feedback on whether the ship was hit or not. When all segments of a ship are hit, the ship is publicly declared sunken. The game concludes when one player sinks all of the opponent's ships. An example of notation used in the paper is Battleship $7, [4, 3, 3, 2]$, which is played on 7×7 grids with ships of sizes 4, 3, 3, 2.

Goofspiel Imperfect Information Goofspiel X is a card game, where each player has cards valued from 1 to X, each only once. In each of the X rounds, a prize card from the dealer's deck is revealed. Players submit a secret bid using one card from their hand, and the player with the highest bid wins points equal to the value of the prize card. In the event of a tie, no points are awarded. Bids are not publicly revealed, only the outcome of the round. In our version, the dealer's deck is not random, but it contains prize cards in descending order from X to 1.

Leduc Hold 'em Leduc Hold 'em is a simplified poker variant played with 6 cards (2 Jacks, 2 Queens, 2 Kings). At the beginning of the game, each player is dealt a single card. The game then progresses through 2 betting rounds. After the first round, the public card is revealed. Each betting round can consist of at most 2 raises. If any player has the same card as the public one, they win. The player with the higher-ranked card wins, where the ranking order is: Jack $<$ Queen $<$ King.

B.2 SUBGAME SOLVING

To show that additional training using the resolving gadget game is a minor change to the training algorithm, we provide pseudocode for this approach in Algorithm 1, highlighting the modifications in red. This pseudocode also shows how the Bayesian unsafe approach is used, as it differs from the

blueprint training only in sampling world states using the belief model rather than starting from the initial state.

Algorithm 1 Subgame Solving

Require: Blueprint parameters θ, resolving player i, opponent j, information set s_i

All techniques, Resolving gadget specific
$\Psi \leftarrow$ Initialize gadget actor parameters
$\theta' \leftarrow \theta$
$s_0 \leftarrow s_0(s_i)$
while Available budget **do**
    Sample a w world state using the belief model — *Initial state in the training*
    Sample trajectory τ from w using $\pi_{\theta'}$
    Estimate counterfactual values $v^{\pi_{\theta'}}_{k,\text{target}}, q^{\pi_{\theta'}}_{k,\text{target}}$ for each player k in every encountered infoset in the trajectory τ
    $q_{\text{term}} \leftarrow v_{j,\theta}(s_j(w))$ — *Blueprint critic value*
    $q_{\text{cont}} \leftarrow v^{\pi_{\theta'}}_{j,\text{target}}(s_j(w))$ — *Target value*
    Update Ψ with policy-gradient loss using $(q_{\text{term}}, q_{\text{cont}})$ at $s_j(w)$
    $p_{\text{cont}} \leftarrow \pi_{j,\Psi}(s_j(w), \text{cont})$
    Update θ' with actor and critic losses scaled by p_{cont}
end while
return θ'

B.3 Belief model

We have conducted an additional experiment to verify the quality of the samples from the trained belief models used in Section 5.2. We have extracted the reaches from the trained blueprint for each possible state, and we have also extracted the probability distributions from the transformer's logits. We then compared those two distributions using Jensen-Shannon divergence, and we show the mean and maximum of those divergences from each public state in Figure 5.

Surprisingly, the belief model trained for the gadget game, which uses importance sampling correction, sometimes produces more accurate samples even with the added variance. This mostly happens with stronger blueprints in Goofspiel 5, where public states later in the game contain hundreds of states. The likely cause is that, in Bayesian solving, the reaches are closer to zero than in the gadget game, which makes them harder to predict correctly due to the variance introduced by the sampling itself.

B.4 Choice of the gadget

In this paper, we have used the resolving gadget game, but there is another type of gadget game *max-margin gadget game*, which further improves upon the resolving gadget game by maximizing the minimal improvement in the subgame (Moravčík et al., 2016). Recent work has shown that the resolving gadget game may contain stronger equilibria than the max-margin (Kubíček et al., 2026).

In this work, we did not use the max-margin because it switches the order of gadget decision. In a resolving gadget game, the chance first decides the state and only then does the opponent select its belief. In max-margin, the opponent selects its belief, and only then is the particular state selected. This means that the opponent needs to decide belief across all of its information sets at the root of the subgame $\mathcal{G}^s$. However, in large games, it is not known a priori how many information sets there are at the root of the subgame. Simulating the max-margin gadget would require this knowledge or having a way to add new actions to the network on the fly. On the other hand, the resolving gadget only adds a decision with 2 actions, so the neural network can easily simulate it.

Consider a situation in which the opponent reaches a part of the subgame only through suboptimal play. It gives you the "gift" in utility you can get in that subgame. *Reach max-margin* subgame solving redistributes this gift to allow more gains in the parts of the subgame that can be reached in optimal play (Brown & Sandholm, 2017). The gifts can be combined with any gadget game.

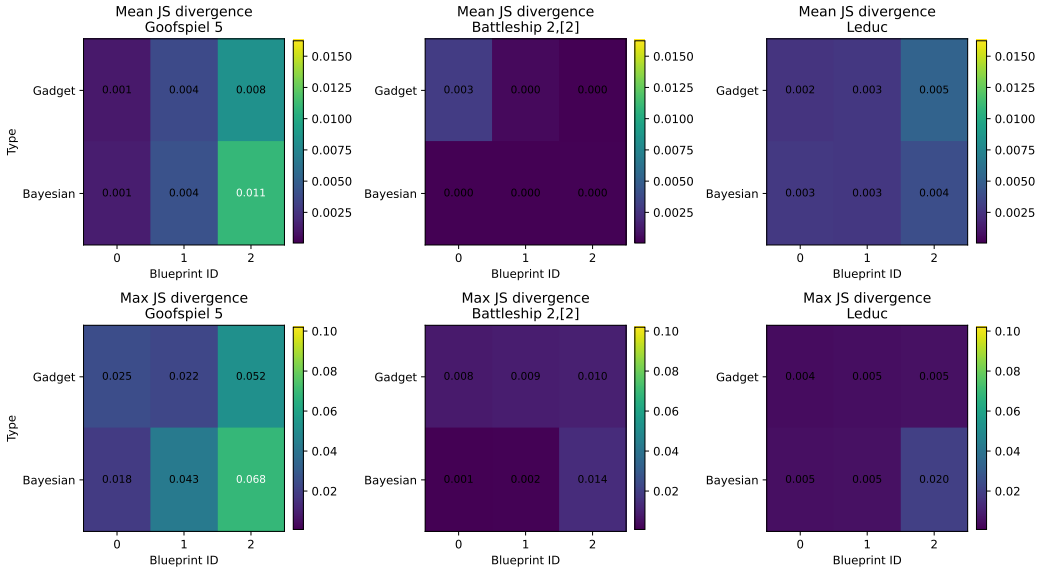


Figure 5: Mean and maximum of Jensen-Shannon divergence between the actual blueprint reaches and the samples from the belief model across each public state in the game.

However, in this work, we did not explore this combination with our simulation and leave this for future research.

B.5 HYPERPARAMETERS

We have used two sets of hyperparameters, one set for small experiments in Sections 5.1 and 5.2 and the second set for large experiments Section 5.3. Both are in Table 2

Table 2: Hyperparameters in all experiments

Parameter	**Small**	**Large**
RNaD regularization η	0.2	0.2
RNaD regularization policy change	2000	20000
NeuRD β clip	2	2
NeuRD avantage clip	5	5
MLP hidden sizes	256	1024
MLP hidden layers	2	2
Belief transformer heads	2	4
Belief transformer layers	4	8
Belief transformer dense size	128	256
Activation function	GELU	GELU
Normalization function	RMSNorm	RMSNorm
Batch size	64	256
Strategy learning rate	$3 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Belief learning rate	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Gadget learning rate	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
Optimizer	ADAM	ADAM
ADAM β_1, β_2	$0.99, 0.999$	$0.99, 0.999$
V-trace λ	0.95	0.95
Discount Factor γ	1.0	1.0

B.6 COMPUTATIONAL RESOURCES

One-shot games For each game in this category, we trained five independent blueprint strategies and their corresponding belief models. Subsequently, test-time reasoning was run for each blueprint. The total computational time for these experiments was less than 24 GPU-hours per game.

Small games For each game, we utilized a single training run and selected three checkpoints representing different stages of strategy convergence. For each checkpoint, we trained three distinct belief models: the first is used for Bayesian solving, and samples states conditioned on the public state based on the probability of reaching those states when following the blueprint. The second is used for resolving gadget games and also samples states conditioned on the public state, but it is unbiased with respect to the opponent's strategy. The third is used for KLSS and UEF, and it samples states conditioned on the information state. The initial training phase required approximately 12 GPU-hours per game. For the evaluation phase, we allocated 2, 5, or 10 seconds of training time per decision point across five different seeds. A single evaluation run was completed in under 6 hours. This makes the total time for this experiment $3 \cdot (12 + 5 \cdot 6 \cdot 3) = 306$ GPU-hours.

Large games We trained five independent blueprint strategies for each large-scale game, each requiring 72 GPU-hours. This was followed by an additional 72 GPU-hours per blueprint to train the three belief model variants. Finally, the head-to-head evaluations lasted approximately 100 hours per configuration. The total computational time is estimated to be at most $3 \cdot (5 \cdot (72 + 72 \cdot 3) + 2 \cdot 4 \cdot 100) = 6720$ GPU-hours. The games played in that time-frame for each bar in Figure 4 are in Table 3

Game	Time	Playthroughs
Battleship 7	10 s	~1500
	30 s	~500
Battleship 10	10 s	~800
	30 s	~300
Goofspiel 10	10 s	~1500
	30 s	~1300

Table 3: Amount of playthroughs used for each bar in Figure 4.

C ADDITIONAL EXPERIMENTS

We provide several additional experiments to accompany the main results from the paper

C.1 EFFECT OF MAGNET STRENGTH IN UNSAFE RESOLVING

To verify the effects of the bounded strategy change from Theorem 2, we conducted the same experiment as in Section 5.1 in Rock-paper-scissors, by using the varied strengths of magnets to solve the same subgame. We show the results of this experiment in Figure 6.

The blueprints used have a small bias towards one action, making the optimal Bayesian gadget solution a pure strategy that is fully exploitable. The magnet then prevents the strategy from changing too much, and, as expected, the lower the magnet regularization, the closer the algorithm converges to the best response.

C.2 HEAD-TO-HEAD PERFORMANCE AGAINST OTHER OPPONENTS

In addition to the experiments in Section 5.3, we also matched all the subgame-solving techniques and blueprints used against two types of players in Battleship 7,[4,3,3,2]. First, being a uniform random player, which picks an action at random, and second, being a strong heuristic player that uses strategies often employed by strong human players. We have used the same blueprints, which were trained for 3 days with RNaD. This shows how difficult it is for policy-gradient algorithms to learn unexploitable strategies in very large games like Battleship, and that subgame solving can fix many holes in such strategies.

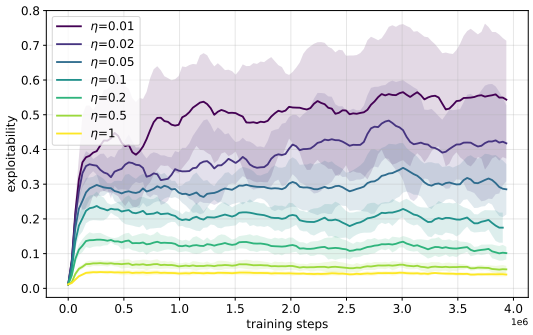
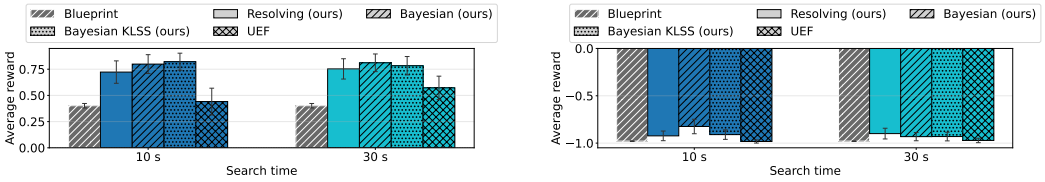


Figure 6: Exploitability in Rock-paper-scissors, when solving a single subgame using Bayesian subgames with fixed magnet, for varying strengths of the magnet.



(a) Random opponent

(b) Heuristic opponent

Figure 7: Head-to-head average reward with 95% confidence intervals of blueprint and different subgame solving techniques against the different types of opponents in Battleship 7,[4,3,3,2].

The heuristic player randomly positions its ships. Then, during the shooting phase, it always shoots a checkboard pattern based on the smallest opponent's ship. When the ship is found, the player tries to sink it as quickly as possible, then continues with the checkboard pattern shooting.

We show the results with 95% confidence intervals in Figure 7. The results are in line with those from Section 5.3, where Resolving and both Bayesian approaches significantly improve upon the blueprint, but the performance improvement among them is inconclusive. Although UEF also improves over the blueprint, the gains are smaller than those of the other methods.