A Meta-Game Evaluation Framework for Deep Multiagent Reinforcement Learning

Zun Li
Google DeepMind
New York, United States
lizun@google.com

Michael P. Wellman
University of Michigan
Ann Arbor, United States
wellman@umich.edu

ABSTRACT

Evaluating deep multiagent reinforcement learning (MARL) algorithms is complicated by stochasticity in training and sensitivity of agent performance to the behavior of other agents. We propose a meta-game evaluation framework for deep MARL, by framing each MARL algorithm as a *meta-strategy*, and repeatedly sampling normal-form empirical games over combinations of meta-strategies resulting from different random seeds. Each empirical game captures both *self-play* and *cross-play* factors across seeds. These empirical games provide the basis for constructing a sampling distribution, using bootstrapping, over a variety of game analysis statistics. We use this approach to evaluate state-of-the-art deep MARL algorithms on a class of negotiation games. From statistics on individual payoffs, social welfare, and empirical best-response graphs, we uncover strategic relationships among self-play, population-based, model-free, and model-based MARL methods. We also investigate the effect of run-time search as a *meta-strategy operator*, and find via meta-game analysis that the search version of a meta-strategy generally leads to improved performance.

KEYWORDS

Deep Multi-Agent Reinforcement Learning, Evaluation, Meta-Game, Empirical Game-Theoretic Analysis, AlphaZero, Negotiation

1 INTRODUCTION

Evaluating complex AI algorithms requires careful attention to stochastic factors. Deep reinforcement learning (RL) algorithms in particular are subject to randomness within the algorithm and the operational environment, and variations with choices of hyperparameters and initial conditions. It is conventional to address these uncertainties in part by aggregating results across multiple runs [Bellemare *et al.* 2013; Machado *et al.* 2018]. Deep multiagent RL (MARL) algorithms present these issues, and additional challenges due to agent interactions. Evaluating performance against humans has been one source of compelling demonstrations [Perolat *et al.* 2022; Silver *et al.* 2016; Vinyals *et al.* 2019; Wurman *et al.* 2022], but this approach is limited by the range of tasks for which human expertise exists, and the cost of engaging it when it is available. Generally speaking, we lack evaluation protocols for comparing different MARL methods in a statistically principled way.

In purely adversarial (*i.e.*, two-player zero-sum) environments, distance to Nash equilibrium may be a sufficient evaluation metric [Brown *et al.* 2020; Schmid *et al.* 2023], as all equilibria are interchangeably optimal. More generally, where there are multiple

equilibria or where we do not necessarily expect equilibrium behavior, the metrics for MARL performance may be less clear. In collaborative domains, global team return is the common objective [Foerster *et al.* 2018; Rashid *et al.* 2020], however complex learning dynamics may lead agents using the same MARL algorithm to equilibria of distinct machine conventions in different runs [Bakhtin *et al.* 2021; Hu *et al.* 2020].

We seek an approach to evaluating deep MARL in general-sum domains. We propose a *meta-game* evaluation framework (§5), which frames MARL algorithms as *meta-strategies*: mapping games and random seeds to joint policies. We sample seed combinations to generate meta-game instances, from which we compute evaluation metrics of interest based on game-theoretic solution concepts. Through resampling and bootstrap techniques, we generate a statistical characterization of algorithm performance in these games. Our contributions:

- The meta-game evaluation framework.
- A new search algorithm for imperfect information games, *Gumbel Information-Set Monte-Carlo Tree Search* (§6), based on the recent development of Gumbel AlphaZero [Danihelka *et al.* 2022].
- Extensive experiments on state-of-the-art MARL algorithms (§7.2) on a class of negotiation games (§7.1), illustrating the framework and providing new evidence regarding the algorithms studied.

2 RELATED WORK

While evaluating single-agent deep RL algorithms is well-studied [Agarwal *et al.* 2021; Henderson *et al.* 2018; Jordan *et al.* 2020], there are relatively few works that consider evaluation principles for deep MARL [Gronauer and Diepold 2022]. Some domain-specific evaluation protocols have been proposed [Leibo *et al.* 2021; Li *et al.* 2019; Lowe *et al.* 2017; Song *et al.* 2020], in which agents are evaluated based on their performance against a selected set of background opponents or emergent behaviors in certain contexts. Other work has defined evaluation protocols for cooperative settings [Gorsane *et al.* 2022; Papoudakis *et al.* 2020], where a global team reward is well-defined. Our scenario falls into the *agent-vs-agent* category of Balduzzi *et al.* [2018], who argue for the necessity of comprehensively considering the possible joint interactions.

Kiekintveld and Wellman [2008] employed a concept of meta-games for evaluating general game-playing methods. In their setting, meta-strategies map normal-form game specifications to strategy choices. Treutlein *et al.* [2021] construct *label-free coordination* (LFC) games among instances of a cooperative MARL algorithm, in order to study zero-shot coordination [Hu *et al.* 2020]. LFC games

are a kind of meta-game in our sense, as they take dynamic game descriptions (Dec-POMDPs) as input.

Bootstrapping is a non-parametric statistical approach that constructs sampling distributions for any specified statistic by resampling from the original dataset [Davison and Hinkley 1997]. Bootstrapping techniques have been applied in machine learning methods such as aggregating decision-tree models [Breiman 1996]. Wiedenbeck *et al.* [2014] applied the bootstrap for statistical analysis of game-theoretic models estimated from simulations.

We test our methods on a class of sequential bargaining games, specifically a bilateral alternating-offer negotiation version developed for recent studies in multi-agent emergent communication [Cao *et al.* 2018; Lewis *et al.* 2017]. These games also resemble the multi-issue negotiation settings explored in multiagent systems research, exemplified in the Automated Negotiation Agent Competition [Baarslag *et al.* 2012]. Much research on agent-based negotiation focuses on heuristic strategies [Fatima *et al.* 2014; Jennings *et al.* 2001], incorporating knowledge of the negotiation domain. A few recent works apply RL methods for bargaining games. Bakker *et al.* [2019] proposed Q-learning-based negotiation strategies with domain knowledge incorporated. Higa *et al.* [2023] employed policy gradient approaches and Chen *et al.* [2023] trained deep RL strategies using off-line interaction data with humans.

3 GAME THEORY PRELIMINARIES

A *normal-form representation* of a *game* $\mathcal{G}$ consists of a *player set* $\mathcal{N} = \{1, \ldots, N\}$, and for each player $i \in \mathcal{N}$ a *pure strategy space* Π_i and a *utility function* $u_i : \Pi \to \mathbb{R}$. $\Pi = \Pi_1 \times \cdots \times \Pi_N$ is the *joint pure strategy space*. A *mixed strategy* $\sigma_i \in \Delta(\Pi_i)$ for player i defines a probability distribution over that player's pure strategy space. Player i's *payoff* for choosing σ_i while others play $\sigma_{-i} = (\sigma_j)_{j \neq i}$ is given by expectation over the respective mixtures: $u_i(\sigma) = \mathbb{E}_{\pi_i \sim \sigma_i, \pi_{-i} \sim \sigma_{-i}}[u(\pi_i, \pi_{-i})]$.

The *regret* of player i who plays σ_i when others are playing σ_{-i} is $\text{REGRET}_i(\sigma_i, \sigma_{-i}) = \max_{\pi'_i \in \Pi_i} u_i(\pi'_i, \sigma_{-i}) - u_i(\sigma_i, \sigma_{-i})$. A *Nash equilibrium* (NE) is a mixed strategy profile σ such that nobody has positive regret: $\forall i. \text{REGRET}_i(\sigma_i, \sigma_{-i}) = 0$. As a measure of approximation to NE, we define $\text{SUMREGRET}(\sigma) = \sum_i \text{REGRET}_i(\sigma_i, \sigma_{-i})$.

If a game is known to be symmetric, we can reduce its normal-form description complexity. Formally, in a *symmetric game*, players share the same strategy space $\Pi_1 = \cdots = \Pi_N = \Pi$, and utility function $u_1 = \cdots = u_N = u$. Furthermore, each player's utility is permutation-invariant to other players' strategies. For symmetric games we overload Π to refer to the common individual strategy space, rather than the joint space. In a *symmetric profile*, every player adopts the same (generally mixed) strategy. Solutions in symmetric profiles are guaranteed to exist in relevant settings [Cheng *et al.* 2004; Hefti 2017; Nash 1951], and are generally preferred absent any basis for breaking symmetry [Kreps 1990].

Also given symmetry, let $u(\sigma', \sigma)$ be the expected payoff of a player when it chooses a strategy $\sigma' \in \Delta(\Pi)$ while the other $N - 1$ play the same mixed strategy $\sigma \in \Delta(\Pi)$. We thus have $\text{REGRET}(\sigma', \sigma) = \max_{\pi' \in \Pi} u(\pi', \sigma) - u(\sigma', \sigma)$. A symmetric NE strategy σ satisfies $\text{REGRET}(\sigma, \sigma) = 0$.

An *extensive-form game* (EFG) representation goes beyond normal form to include temporal and information structure. Effectively,

an EFG defines a dynamical system (usually visualized by a tree) with a world state or history h that is not necessarily fully observable by every player. At each state h, starting with the initial state h_0, there is a single player $\tau(h) \in \mathcal{N} \cup \{c\}$, where c is the *chance player*, designated to select an action $a \in \mathcal{A}(h)$. The chance player chooses according to a fixed random policy. Player $i \in \mathcal{N}$ chooses according to its *information state/set*, $s_i(h)$, which comprises the information that i has observed at h. Often $s_i(h)$ is represented by a concatenation of public and private action-observation histories. Following action a, the world transits to $h' = ha$ until a terminal state $z \in \mathcal{Z}$ is reached. Then each player $i \in \mathcal{N}$ receives a utility $u_i(z)$ as a function of z. A *behavioral strategy* or *policy*, π_i, of player i is a mapping from i's infostates to distributions over actions. For EFGs we overload Π_i to refer to the set of behavioral strategies of player i. A joint behavioral strategy profile π thus induces a probability distribution over the terminal states and we define $u_i(\pi)$ as the expected payoff for player i under this distribution. In this paper, we consider EFGs with *perfect recall*, that is, information sets $s_i(h)$ distinguish all actions i had taken to reach h. A consequence is that any mixed strategy $\sigma_i \in \Delta(\Pi_i)$ is payoff-equivalent to some behavioral strategy $\pi_i \in \Pi_i$ [Aumann 1964].

4 MULTIAGENT TRAINING ALGORITHMS

We define a *multiagent training algorithm* (MATA) $\mathcal{M}$ as a stochastic procedure that produces a policy profile $\pi = \mathcal{M}(\mathcal{G}, \Theta, \omega)$ for an EFG. In general, the input EFG $\mathcal{G}$ cannot be tractably represented as an explicit game tree. Instead, we assume the game is given in the form of a black-box simulator that the algorithm can exercise by submitting actions and receiving observations and rewards. Θ is the set of hyperparameters of the algorithm, and ω is a random seed. If $\mathcal{G}$ is symmetric, then it is often natural to constrain the output π to be likewise symmetric (*i.e.*, single policy to be played by all). More generally, π is a policy profile. For the AlphaZero MATA, for example, $\mathcal{G}$ could be represented by a Go simulator, and Θ would include the learning rate schedule and neural architecture. The output profile π specifies Go-playing policies for white and black, respectively.

A MATA is effectively a form of *meta-strategy*: a procedure that given a game $\mathcal{G}$, generates a strategy profile for $\mathcal{G}$. We can employ the MATA to play from the perspective of any particular player i, simply by selecting the ith element $\pi(i)$ of the output profile.

A key issue for analysis of MATAs is uncertainty in strategy generation. It has been well observed (*e.g.*, by Hu *et al.* [2020] and Bakhtin *et al.* [2021]) that a MATA with the same $\mathcal{G}$ and Θ but different ω may generate policies with vastly different strategic behaviors. For example, in a negotiation game, different runs of a MATA may lead to strategies that adopt distinct offering conventions. In the present work, we assume the hyperparameters Θ^m for each MATA $\mathcal{M}^m$ have been fixed, so the uncertainty in behavior of a training algorithm is fully captured by the random seeds. Note that there is always discretion about what one considers a distinct MATA $\mathcal{M}$ versus a parametric variation, so it is possible to bring choice among hyperparameter settings within the scope of our analysis framework.

5 META-GAME EVALUATION FRAMEWORK

Given M different MATAs $\{(\mathcal{M}^1, \Theta^1), \ldots, (\mathcal{M}^M, \Theta^M)\}$ with associated hyperparameters, how can we evaluate their relative performance with respect to a given game $\mathcal{G}$? Viewing the MATAs simply as game-solvers, we could focus on measures of their effectiveness in deriving a solution—for example, time and accuracy of convergence to Nash equilibrium. Viewing the MATAs' role as generating strategies to play a game, however, requires a different focus that considers the interaction with other strategy generators. This is particularly salient for games that have a multiplicity of solutions (the general case), or for which the operable solution concept may be open to question. Consequently, we propose to analyze competing MATAs by framing their interaction as itself a game. As MATAs are *meta-strategies*, we refer to this approach as a ***meta-game*** evaluation framework.

As noted above, we are particularly concerned with uncertainty in the results of multiagent training. In analysis of the MATA meta-game, therefore, we aim to characterize the implications of this uncertainty in probabilistic form.

5.1 Empirical Game-Theoretic Analysis

Our approach employs ***empirical game-theoretic analysis*** (EGTA), a methodology for reasoning about games through agent-based simulation [Tuyls *et al.* 2020; Wellman 2016]. EGTA aligns with our assumption that the game of interest $\mathcal{G}$ is defined by a black-box simulator. In the typical framing, EGTA operates by estimating an ***empirical game model*** in normal form over an enumerated strategy set. The enumerated strategies are a small selection from the full strategy space of the extensive game represented by the simulator.

We employ EGTA with respect to an N-player game of interest $\mathcal{G}$, and strategies defined by the output of MATAs. The meta-game is likewise over N players, and is symmetric regardless of whether $\mathcal{G}$ is symmetric or not. Let $\hat{\pi}^m$ denote the output from MATA m, for instance $\hat{\pi}^m = \mathcal{M}^m(\mathcal{G}, \Theta^m, \omega)$: the result from running the MATA on $\mathcal{G}$ for a particular random seed. We also allow that $\hat{\pi}^m$ be an aggregate of policy profiles from multiple random seeds. From these MATA-generated policy profiles $\hat{\Pi} = \{\hat{\pi}^1, \ldots, \hat{\pi}^M\}$, we construct an empirical meta-game $\mathcal{MG}(\hat{\Pi})$ over policy space $\hat{\Pi}$ as follows.

If the base game $\mathcal{G}$ is symmetric, then the $\hat{\pi}^m$ are single-player policies, and we estimate the meta-game utility function by the standard EGTA approach of simulating profiles over $\hat{\Pi}$. If $\mathcal{G}$ is not symmetric, then each $\hat{\pi} \in \hat{\Pi}$ is an N-player profile for $\mathcal{G}$. Simulating a meta-game profile $(\hat{\pi}_1, \ldots, \hat{\pi}_N)$ of these base-game profiles entails first assigning the $\hat{\pi}$ to players, according to a random permutation *perm* drawn uniformly from the $N!$ possibilities. Then it simulates a play where player i of $\mathcal{MG}$ plays $\hat{\pi}_i(perm(i))$ as if it is player $perm(i)$ in $\mathcal{G}$. This construction also corresponds to an EFG beginning with a root chance node that uniformly chooses among $N!$ different outcomes, each followed by a copy of the original EFG with player indices permuted.

Symmetry of the meta-game reflects a view that, for multiagent training, developing effective strategies for each of the player positions is equally important. If this were not the case, or if one wished to perform an analysis of the differential effectiveness of various MATAs from the perspectives of different players or roles,

a non-symmetric (or role-symmetric) meta-game model could be constructed instead.

5.2 Meta-Game Evaluation Procedure

Just as single-agent RL algorithms are statistically evaluated by return performance across different random seeds, we can analyze strategic properties among MATAs across different *combinations* of seeds. Let X denote statistics characterizing the strategic properties of interest (discussed below). Given an N-player game $\mathcal{G}$ and parametrized MATAs $\{(\mathcal{M}^1, \Theta^1), \ldots, (\mathcal{M}^M, \Theta^M)\}$, our evaluation procedure comprises of the following steps:

(1) Select a finite set of seeds Ω^m for each MATA m. Generate $\hat{\pi}^m(\omega) = \mathcal{M}^m(\mathcal{G}, \Theta^m, \omega)$ for each $\omega \in \Omega^m$.

(2) For each m, uniformly sample $|\Omega^m|$ seeds from Ω^m with replacement, yielding the sequence $(\omega_1, \ldots, \omega_{|\Omega^m|})$. Let $\hat{\pi}^m$ be a profile that that is payoff-equivalent to a uniform mixture over the multiset $\{\hat{\pi}^m(\omega_1), \ldots, \hat{\pi}^m(\omega_{|\Omega^m|})\}$.

(3) Given $\hat{\Pi} = \{\hat{\pi}^1, \ldots, \hat{\pi}^M\}$, estimate the symmetric empirical meta-game $\mathcal{MG}(\hat{\Pi})$ as described in §5.1.

(4) Compute the statistics-of-interest X from $\mathcal{MG}(\hat{\Pi})$

(5) Repeat Steps 2 through 4. Estimated profile payoffs should be memoized for reuse across iterations. Obtain an empirical distribution of X and report statistical properties of X.

One way to understand this evaluation procedure is to view each MATA as a *mixed strategy*, selecting policies uniformly over the possible random seeds. The "ground-truth" meta-game represents an expectation over the results of this randomization. The empirical meta-game estimates this from finite samples Ω^m. By resampling from these seeds at hand, Step 2 constructs multiple empirical games among MATAs, from which we construct sampling distributions over X using bootstrapping.

There are a variety of choices for statistics X to gather. The only requirement is that X can be computed from the information in a normal-form game model. For example, one possible metric on MATA performance is ***uniform-score***: average payoff against a uniform distribution over other MATAs. Such scores have been employed in a variety of contexts, however putting equal weight on the possible counterparts is questionable, as they are not equally relevant [Balduzzi *et al.* 2018].

An alternative proposed by Jordan *et al.* [2007] is ***NE-regret***: REGRET(π, σ^*), where σ^* is a symmetric mixed equilibrium of $\mathcal{MG}(\hat{\Pi})$. The motivation for this measure is that it focuses on behavior against rational opponents. Performance against obviously flawed opponents should carry much less weight, as they are less likely to be encountered in practice, all else equal.[1] For games with multiple equilibria, however, NE-regret is sensitive to the choice of solutions σ^*. This sensitivity is inherent to situations with multiple equilibria, but it can still be helpful to adopt a focal equilibrium to reduce ambiguity in analysis. Balduzzi *et al.* [2018] proposed ***Nash averaging***, which is essentially NE-regret with respect to the max-entropy equilibrium. The intuition for preferring to evaluate with respect to higher-entropy solutions is that they reflect diversity, and thus reward robustness to a wide range of rational opponents.

---

[1]If however there is some basis to expect opponents who are flawed or boundedly rational in some particular way, then by all means it would make sense to measure regret with respect to a solution concept capturing that basis.

5.3 Max-Entropy Nash Equilibrium

Computing max-entropy NE is hard in general, but practically feasible to approximate for bi-matrix games of modest size. We adapt the mixed-integer programming formulation of Sandholm *et al.* [2005]:

$$\min_{\sigma^*} \quad \sum_{\pi \in \hat{\Pi}} \sigma^*(\pi) \log \sigma^*(\pi)$$

$$\text{s.t.} \quad \forall \pi \in \hat{\Pi}. \quad u_\pi = \sum_{\pi' \in \hat{\Pi}} \sigma^*(\pi') u(\pi, \pi'), \quad (1)$$

$$u^* \geq u_\pi, \quad u^* - u_\pi \leq U b_\pi, \quad \sigma^*(\pi) \leq 1 - b_\pi,$$

$$\sigma^*(\pi) \geq 0, \quad \sum_\pi \sigma^*(\pi) = 1, \quad b_\pi \in \{0, 1\},$$

with real variables $\sigma^*(\pi)$, u_π, binary variables b_π for each strategy π, and an equilibrium payoff real variable u^*. U is the maximum difference across payoffs. The variables b_π indicate whether strategy π is outside the equilibrium support. That is, $b_\pi = 1$ iff $\sigma^*(\pi) = 0$. Otherwise $u_\pi = u^*$: strategies in the support have the same payoff. The details of our implementation are in App. A.

Although max-entropy NE is not generally unique, it narrows the possibilities considerably compared to unconstrained (approximate or exact) NE.

6 SEARCH AS A META-STRATEGY OPERATOR

Many MATAs produce a policy network p that maps directly from an infostate to a distribution over actions in a forward pass for every player. Recent work has found that leveraging computation at run-time and adding search to p can improve performance in large EFG domains [Brown *et al.* 2020; Schmid *et al.* 2023; Silver *et al.* 2018]. As a case study for our meta-game evaluation framework, we apply it to investigate the effect of search as a general policy improver.

Toward that end, we propose a heuristic search method for large EFGs based on information-set MCTS (IS-MCTS) [Cowling *et al.* 2012] and Gumbel AlphaZero [Danihelka *et al.* 2022]. Details of this procedure, ***Gumbel IS-MCTS***, are specified in Alg. 1. Parameterized by a policy net p and a value net v, the procedure conducts multiple passes over the game-tree guided by v and p at an input infostate s, and outputs an action a for decision-making. We can apply this procedure to a variety of underlying MATAs, as a ***meta-strategy operator***: transforming $\mathcal{M}$ to $\mathcal{M}'$ (with additional hyperparameters like simulation budget). The meta-strategy $\mathcal{M}'$ in effect adds run-time search to the output policy p of $\mathcal{M}$. Notice that unlike AlphaZero—which uses the same MCTS method for training and run-time, with meta-strategy operators we can explore a variety of MATAs as training-time methods which produce v and p for search at test-time [Sokota *et al.* 2023] (details in §7.2.4).

We next explain the key design of Gumbel IS-MCTS and how it differs from previous search methods. We defer further technical details in App. B. Just like MCTS, IS-MCTS incrementally builds and traverses a search tree and aggregates statistics such as visit counts $C(s, a)$ and aggregated values $R(s, a)$ for visited (s, a) pairs. During each simulation of the search (line 5), a world state is sampled from a posterior belief $\Pr(h \mid s, p)$. In our test domains, $\Pr(h \mid s, p)$ can be computed exactly via Bayes's rule, whereas in larger domains, using particle filtering [Silver and Veness 2010] or deep generative

---

Algorithm 1 Gumbel IS-MCTS

1: **function** Gumbel-Search(s, v, p)
2: $\quad \forall(s, a).\ R(s, a) \leftarrow 0, C(s, a) \leftarrow 0$
3: $\quad \forall a \in \mathcal{A}(s).$ sample $g(a) \sim Gumbel(0), \hat{\mathcal{A}} \leftarrow \mathcal{A}(s)$
4: $\quad$ **repeat**
5: $\quad\quad$ Sample a world state: $h \sim \Pr(h \mid s, p)$
6: $\quad\quad$ **while do**
7: $\quad\quad\quad$ **if** h is terminal **then**
8: $\quad\quad\quad\quad r \leftarrow$ payoffs of players **Break**
9: $\quad\quad\quad$ **else if** $i \triangleq \tau(h)$ is chance **then**
10: $\quad\quad\quad\quad a \leftarrow$ sample according to chance
11: $\quad\quad\quad$ **else if** $s_i(h)$ not in search tree **then**
12: $\quad\quad\quad\quad$ Add $s_i(h)$ to search tree
13: $\quad\quad\quad\quad r \leftarrow v(s_i(h)).$ **Break**
14: $\quad\quad\quad$ **else if** $s_i(h)$ is root node s **then**
15: $\quad\quad\quad\quad a, \hat{\mathcal{A}} \leftarrow$ one step of sequential halving (Alg. 3) based on $GS(s, a)$ and remaining actions in $\hat{\mathcal{A}}$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad$ Select a according to Eq.(3) in App. B.2.
18: $\quad\quad\quad$ **end if**
19: $\quad\quad\quad h \leftarrow ha$
20: $\quad\quad$ **end while**
21: $\quad\quad$ **for** (s_i, a) in this trajectory **do**
22: $\quad\quad\quad$ Increment $R(s_i, a)$ by r_i, $C(s_i, a)$ by 1.
23: $\quad\quad$ **end for**
24: $\quad$ **until** *num_sim* simulations done
25: $\quad$ **return** Action a that remains in $\hat{\mathcal{A}}$
26: **end function**

---

models [Hu *et al.* 2021; Li *et al.* 2023] to approximate $\Pr(h \mid s, p)$ are possible.

The key feature of Gumbel IS-MCTS is how it selects actions at the search nodes. At the beginning of the search (line 3), a Gumbel random variable, $g(a)$, is sampled i.i.d. for each legal action a of the root, for later use in action selection. At the root (line 15), the algorithm treats each legal action as an arm of a stochastic bandit, and uses a sequential-halving algorithm [Pepels *et al.* 2014] (Alg. 3) to distribute the simulation budget. Sequential-halving algorithms usually are designed for minimizing the *simple regret* [Bubeck *et al.* 2009], which is the regret at the last-iteration action recommendation. By contrast, UCB-styled algorithms are usually designed for minimizing the *accumulated regret* during an online learning process. For a game-playing search algorithm, minimizing simple regret makes more sense in terms of producing a single optimal action at a decision point.

We assign to each arm a a ***Gumbel score*** $GS(s, a) = g(a) + $ logit $p(s, a) + G(\hat{q}(s, a))$. The second term is the logit of a produced by p, and the third term is a monotone transformation of the action value $\hat{q}(s, a)$, which is estimated by $R(s, a)$, $C(s, a)$, and v. The intuition is that a high $\hat{q}(s, a)$ value indicates a direction for policy improvement. Indeed, the improved policy $Imp(p)(s, a) \triangleq$ SoftMax(logit $p(s, a) + G(\hat{q}(s, a))$) provably achieves higher expected values Danihelka *et al.* [2022, App. B]. The forms of G and $\hat{q}$ is detailed in App. B.1.

Adding Gumbel noise $g(a)$ implements the "Gumbel top-K-trick": deterministically selecting the top K actions according to $GS(s, a)$

Figure 1: Example start of sequential bargaining game instance.

is equivalent to sampling K actions from $Imp(\boldsymbol{p})(s, a)$ without replacement [Huijben *et al.* 2022]. The Gumbel score induces a low-variance non-deterministic action selection of the root node during the sequential halving process, which encourages exploration while distributing the simulation budget toward actions likely to yield higher expected values.

At a non-root node (line 17), an action is selected to minimize the discrepancy between $Imp(\boldsymbol{p})$ and the produced visited frequency (details in App. B.2). At the end of the search, Gumbel IS-MCTS outputs the action a that survives the sequential halving procedure.

7 EVALUATION STUDY

7.1 Domain: Alternating Negotiation

We test our MATAs and evaluation framework in the context of a two-player negotiation game. The particular version we study, called "Deal-or-No-Deal" (DoND) by Lewis *et al.* [2017], is a family of two-player negotiation games where the players alternate proposals to divide a pool of resources: three different goods available in various amounts. At the start of the game, quantities of goods in the pool and players' private valuations for units of each good are sampled from a known distribution. The valuation distribution is constrained to fix a constant value for both players of the aggregate pool: $\mathbf{c} \cdot \mathbf{w}_1 = \mathbf{c} \cdot \mathbf{w}_2 = 10$. For example, in Fig. 1, the pool of goods can be represented as $\mathbf{c} = [1, 2, 3]$. The negotiation proceeds through alternating proposals for dividing the resources. In this example, player 1 first proposes $\boldsymbol{o}_1 = [1, 2, 0]$ for itself which entails $\boldsymbol{o}_2 = \mathbf{c} - \boldsymbol{o}_1 = [0, 0, 3]$ for player 2. Player 2 rejects and proposes a counter-offer, to which 1 may either agree or continue bargaining. For a particular game parametrization $\texttt{Barg}(T, \varepsilon, \gamma)$, the game ends if either (1) a deal is made, (2) a maximum number of rounds T is reached, or (3) chance decides to terminate the game, which happens at every round with probability ε. If an agreement $(\boldsymbol{o}_1, \boldsymbol{o}_2)$ was reached at the tth round, player i receives payoff $\gamma^t \boldsymbol{w}_i \cdot \boldsymbol{o}_i$. Otherwise both players receive zero payoff.

DoND is a family of challenging general-sum environments with imperfect information. Players have a common interest in reaching a deal (and quickly, for $\gamma < 1$), and the different ways of dividing the pool have different total value. In our studies, we sample configurations uniformly from a database of 6796 published by Lewis *et al.* [2017]. This leads to prohibitive complexity: a game with $T = 10$ has $O(10^{11})$ infosets for every player, which is intractable for enumeration (detailed in App. G), and grows exponentially with T.

7.2 Benchmark Algorithms

Our meta-game evaluation considers the following $M = 17$ MATAs. These algorithms represent a comprehensive set of state-of-the-art

MARL algorithms, including methods using self-play based training, population based training, and model-free and model-based approaches. Details are below and in App. D.

7.2.1 Independent/Multiagent PPO (IDPPO/MAPPO). Both IDPPO and MAPPO train policy and value nets using self-play trajectories by minimizing the trust-region clipped loss [Schulman *et al.* 2017] using the generalized advantage estimator (GAE) [Mnih *et al.* 2016]. Value nets are trained by minimizing L2 loss from the targets produced by GAEs. In IDPPO each player maintains its own policy and value nets whereas in MAPPO [Yu *et al.* 2022] players share the same neural nets.

7.2.2 Regularized Nash Dynamics (R-NaD) and NFSP. Both R-NaD [Perolat *et al.* 2021] and Neural Fictitious Self-Play (NFSP) [Heinrich and Silver 2016] are self-play model-free MARL algorithms originally designed for purely adversarial settings. R-NaD has recently shown success in producing human-level agents in Stratego [Perolat *et al.* 2022], where it iteratively trains policy nets by minimizing NeuRD loss [Hennes *et al.* 2020] and value nets using the V-Trace estimator [Espeholt *et al.* 2018] on a sequence of regularized games. NFSP mimics the classic fictitious play algorithm [Heinrich and Silver 2016] by alternating between training a supervised learning net that summarizes historical plays and training a DQN policy that serves as a best response.

7.2.3 Policy Space Response Oracles (PSRO) and FCP. PSRO [Lanctot *et al.* 2017] and Fictitious Co-Play (FCP) [Strouse *et al.* 2021] are two population-based MARL algorithms. PSRO is an EGTA method that iteratively adds policies that are best responses to the current solution. We use max-entropy Nash as the solution concept (solving an asymmetric version of (1)), and PPO as the best-response method. We consider two ways of extracting the final agents: (i) PSRO: using the final-iteration solution and (ii) PSRO-LAST: using the final-iteration best-response policies. FCP has notably demonstrated success in collaborating with humans from scratch [Strouse *et al.* 2021]. FCP builds a population by picking policies of different skill levels on multiple self-play runs, and trains final best responses against such populations. Our implementation of FCP uses IDPPO to generate self-play runs, picks the agents based on social welfare, and uses PPO as the best-response method.

7.2.4 Gumbel Search and Vanilla AlphaZero-style Search. We include the following variants of Gumbel IS-MCTS: (i) G-Search: Using Gumbel IS-MCTS for both training the networks (Alg. 4) at training time and conducting search at test time. (ii) G-Search-PN: Using search at training time, but only policy net without search at test time. (iii) G-Search-IDPPO: Using IDPPO to train the policy and value nets, based on which Gumbel IS-MCTS executes test-time search. (iv) G-Search-MAPPO and (v) G-Search-R-NaD are similarly defined. In addition, we implement an extension of the original AlphaZero-style MCTS to IS-MCTS. The differences between this search method (Alg. 2 in App. C) and Gumbel IS-MCTS are in the mechanisms for exploration (*e.g.*, Dirichlet noise) and action selection (*e.g.*, PUCT) within the search tree. We include (vi) VA-Search: Using this search for both training time and test time, and (vii) VA-Search-PN: similarly defined as (ii).

7.2.5 Heuristic Strategies. We further include the following baseline bargaining strategies: (i) **Uniform**, which uniformly samples among all legal actions at each decision point. (ii) **Tough**, which never agrees, and always proposes uniformly among offers that maximize its own payoff. (iii) **Soft**, which always agrees, or uniformly proposes among all offers if it is the first-mover.

7.3 Experimental Setup

All methods are implemented within the programming model of OpenSpiel [Lanctot *et al.* 2019]. Each MATA that involves neural training uses models of approximately equal size (*e.g.*, number of layers, hidden nodes). The neural net inputs are likewise the same across all MATAs, and include complete history information (*i.e.*, losslessly represent infosets) for the DoND game. For procedures that involve self-play or RL training we employ roughly the same number of training trajectories. For all search methods we use 200 simulations per call. We fine-tuned the learning rates based on SUMREGRET performance. Most of the MATAs reliably produce approximate equilibria. Details are in App. E.

We set $|\Omega^m| = 10$, $m \in \{1, \ldots, M\}$. To simplify the evaluation procedure, we pre-compute an empirical payoff matrix covering every policy pair: $(\hat{\pi}_1^m(\omega), \hat{\pi}_2^{m'}(\omega'))$, for $m, m' \in \{1, \ldots, M\}$, ω and ω' among the seeds sampled for $\mathcal{M}^m$ and $\mathcal{M}^{m'}$, respectively. For each payoff entry we run 2×10^4 simulations to compute the expected payoff. We then sample and analyze 10^6 $M \times M$ empirical meta-games from this matrix, per Steps 2–4 of §5.2, to obtain distributions over the statistics of interest.

7.4 Results

We test all 17 MATAs on two DoND instances: `Barg(10, 0, 1)` and `Barg(30, 0.125, 0.935)`. The two settings are qualitatively different. The latter includes discounting and per-round ending probability, incentivizing the players to find and agree on a good deal in early rounds. Without these factors, the first game reduces to an ultimatum-game-like environment. Meta-game statistics are reported in Tables 1 and 2.

In addition to NE-Regret and uniform scores, we also report a statistic we term the ***NE-Nash-Bargaining-Score*** (NE-NBS). The NE-NBS for MATA m in an empirical game is defined as the utility-product between π^m and an opponent strategy σ^*: $u(\pi^m, \sigma^*) \cdot u(\sigma^*, \pi^m)$. Here σ^* is a max-entropy Nash computed by (1). This statistic is intended to measure the effectiveness of an agent in achieving high social welfare and fairness with a rational opponent.

From the tables we can see that while there is positive correlation between NE-Regret Scores and Uniform Scores, NE-Regret Scores are better at identifying the most robust MATAs. The rankings of the top four MATAs in NE-regret scores are the same for both environments, and from the empirical distribution plots in App. F we can see G-Search-R-NaD and G-Search consistently produce strategies that are selected in a max-entropy equilibrium support with high probability. By contrast uniform-score comparisons can be distorted by non-salient success against weak strategies. It is also interesting to note positive correlation between NE-Regret and NE-NBS. This reflects the non-zero-sum nature of these environments.

The performance of search-based agents is especially noteworthy. Three of the top four MATAs employ search at test time. All search methods are stronger than their policy net counterparts, confirming that IS-MCTS is a policy-improver in imperfect information games. We hypothesize that this is because a search-based algorithm is usually a better responder to its policy network counterpart, while both behave strategically similarly against other methods. The relative strength among search-based algorithms mirrors that of their policy-net counterparts. In both environments, G-Search and VA-Search are among the top four. This suggests that using search at training time is compatible with the same search at test time, and produces better policy and value nets compared with other methods.

We notice some algorithms behave qualitatively differently in these two environments. We found FCP consistently produces collaborative strategies with rather high agreement probability. This may explain its better performance in the second environment, which encourages settling a deal in early rounds. Likewise, Soft achieves higher individual scores and NE-NBS in the second environment. By contrast, while Tough achieves better individual performance than Soft in the ultimatum game, it is still worse than all learning methods, as shown by its terrible NE-NBS values for both environments.

Another interesting view is provided by ***best-response graphs***. In a best-response graph for a game instance, strategies are vertices, and there is a directed edge $(m_1 \rightarrow m_2)$ iff $m_2 = \arg\max_{m'} u(\pi^{m'}, \pi^{m_1})$. We generate an aggregate graph for a MATA meta-game by recording the frequency of such edges in the sampled empirical games. These are shown for the two game settings in Fig. 2 and Fig. 3.

The empirical best-response graphs support several observations. First, many MATA vertices such as IDPPO and MAPPO have self-edges with non-negligible frequencies. This is consistent with previous observations that self-play MARL algorithms are likely to produce agents that overfit to themselves [Hu *et al.* 2020]. However, self-edges do not necessarily suggest poor generalization performances. For example, G-Search-NaD has a self-edge with a high probability in both environments, but still performs the best. Self-edges persist even after applying search as policy-improver at test time. For example, there are self-edges for G-Search-IDPPO and G-Search-MAPPO. This may suggest that there are certain strategic correlations between a search agent and its policy net counterpart. Interestingly, we also observe certain self-edge probabilities for PSRO and PSRO-LAST, which are normally regarded as population-based training methods. It could be that the equilibria selected at each iteration introduce correlations between different players. By contrast, NFSP and FCP are two MATAs without strong self-edges; both use a uniform distribution for opponent modeling, which may reduce the correlation between player positions.

Edges across MATAs illuminate strategic interaction structure. Tough is unsurprisingly the best response to Soft. G-Search-NaD is a strong attractor in both graphs, as G-Search is in the first. Sometimes a search-based MATA is a best response to its policy-net counterpart, as illustrated by R-NaD to G-Search-R-NaD, and VA-Search-PN to VA-Search. This could also explain why MATAs such as G-Search-PN and VA-Search-PN do not have apparent self-edges.

MATA Name	NE-Regret-Score	Uniform-Score	NE-NBS	MATA Name	NE-Regret-Score	Uniform-Score	NE-NBS
G-Search-R-NaD	**0.002±0.020**	6.158±0.040	44.010±0.794	MAPPO	0.721±0.059	5.602±0.035	32.832±0.818
G-Search	0.010±0.046	**6.215±0.015**	**44.345±0.496**	IDPPO	0.766±0.133	5.645±0.050	30.240±2.051
R-NaD	0.045±0.043	5.977±0.027	42.008±0.738	NFSP	0.806±0.050	5.766±0.014	36.534±0.721
VA-Search	0.092±0.023	6.074±0.029	40.836±0.551	VA-Search-PN	0.843±0.071	5.570±0.026	36.540±0.717
G-Search-MAPPO	0.354±0.047	6.038±0.027	37.808±0.962	FCP	1.116±0.096	5.804±0.013	38.665±1.092
PSRO-LAST	0.414±0.051	5.890±0.026	39.187±0.762	Tough	1.801±0.093	4.017±0.031	6.581±0.137
PSRO	0.417±0.055	5.904±0.022	38.288±0.873	Soft	3.189±0.098	3.773±0.013	27.023±0.855
G-Search-IDPPO	0.480±0.062	6.088±0.031	36.385±1.317	Uniform	4.274±0.062	2.391±0.004	11.980±0.494
G-Search-PN	0.575±0.048	5.811±0.013	42.174±0.493				

Table 1: Results for `Barg`$(10, 0, 1)$, with 95% confidence intervals. MATAs are listed in increasing order of NE-Regret.

MATA Name	NE-Regret-Score	Uniform-Score	NE-NBS	MATA Name	NE-Regret-Score	Uniform-Score	NE-NBS
G-Search-R-NaD	**0.000±0.001**	**6.147±0.013**	43.681±0.497	PSRO	0.488±0.056	5.688±0.052	39.851±0.728
G-Search	0.005±0.033	6.078±0.019	43.198±0.545	G-Search-MAPPO	0.729±0.095	5.656±0.056	32.983±0.994
R-NaD	0.045±0.010	6.121±0.012	**44.072±0.417**	IDPPO	0.749±0.059	5.504±0.042	36.350±0.685
VA-Search	0.075±0.023	6.058±0.014	42.605±0.610	PSRO-LAST	0.906±0.213	5.391±0.161	35.723±2.119
G-Search-PN	0.294±0.019	5.833±0.016	42.285±0.448	MAPPO	1.135±0.144	5.306±0.086	31.878±1.378
FCP	0.360±0.026	5.864±0.013	43.321±0.265	Soft	1.918±0.018	4.352±0.011	35.990±0.267
VA-Search-PN	0.412±0.033	5.668±0.014	40.314±0.325	Uniform	4.161±0.031	2.370±0.007	12.562±0.211
NFSP	0.421±0.011	5.786±0.014	39.151±0.481	Tough	4.542±0.082	2.685±0.025	2.270±0.097
G-Search-IDPPO	0.450±0.038	5.820±0.019	36.997±0.432				

Table 2: Results for `Barg`$(30, 0.125, 0.935)$, with 95% confidence intervals. MATAs are listed in increasing order of NE-Regret.
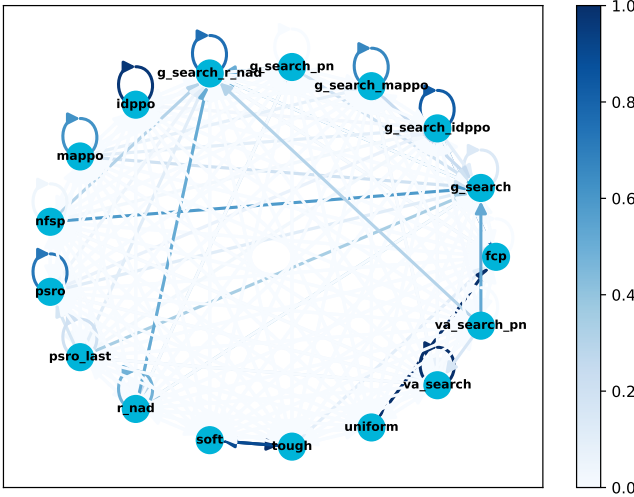

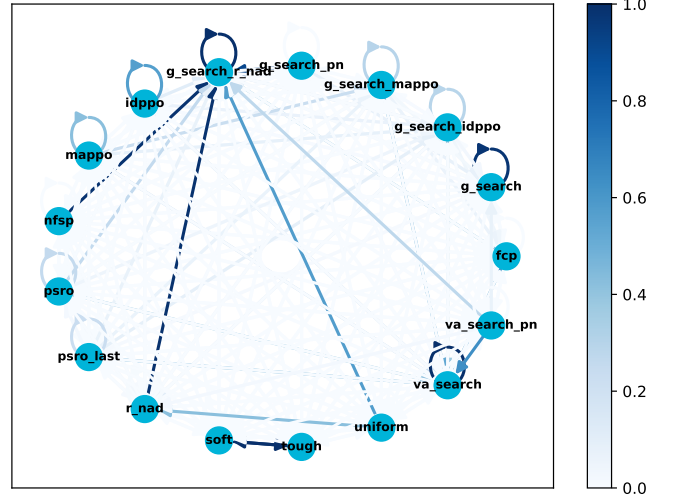Figure 2: Empirical best-response graph for `Barg`(10, 0, 1).

Figure 3: Empirical best-response graph for `Barg`(30, 0.125, 0.935).

8 CONCLUSION

We propose a meta-game evaluation framework for MARL in general-sum environments. Our approach is analogous to the evaluation process for single-agent RL, effectively aggregating the strategic analysis procedures across possible worlds defined by different seed combinations. We illustrated the method by constructing a meta-game model over a comprehensive set of multiagent training algorithms using deep RL on a class of negotiation games. The meta-game analysis evaluated the algorithms in multiple ways, most prominently through max entropy NE-regret ranking and the structure of best-response graphs. Bootstrap statistics provide a basis for assessing uncertainty in evaluation results.

Experimental results support several interesting observations about Gumbel IS-MCTS as a meta-strategy operator, especially

regarding the value of search at both training time and test time, and on patterns of strategic interactions among the algorithms.

A key feature of our evaluation framework is the flexibility to investigate a variety of statistics of interest through a carefully structured process. Future work will focus on understanding the robustness of alternative measures that can be employed for algorithm assessment through meta-games, and developing effective tools for statistical analysis.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *35th Int'l Conf. on Neural Information Processing Systems*, pages 29304–29320, 2021.

Robert J. Aumann. Mixed and behavior strategies in infinite extensive game. In M. Dresher, L. S. Shapley, and A. W. Tucker, editors, *Advances in Game Theory*, Annals of Mathematics Studies, pages 627–650. Princeton University Press, 1964.

Tim Baarslag, Koen Hindriks, Catholijn Jonker, Sarit Kraus, and Raz Lin. The first automated negotiating agents competition (ANAC 2010). *New Trends in agent-based complex automated negotiations*, pages 113–135, 2012.

Anton Bakhtin, David Wu, Adam Lerer, and Noam Brown. No-press diplomacy from scratch. In *35th Int'l Conf. on Neural Information Processing Systems*, pages 18063–18074, 2021.

Jasper Bakker, Aron Hammond, Daan Bloembergen, and Tim Baarslag. RLBOA: A modular reinforcement learning framework for autonomous negotiating agents. In *18th Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 260–268, 2019.

David Balduzzi, Karl Tuyls, Julien Perolat, and Thore Graepel. Re-evaluating evaluation. In *32nd Int'l Conf. on Neural Information Processing Systems*, 2018.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *34th Int'l Conf. on Neural Information Processing Systems*, pages 17057–17069, 2020.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *20th Int'l Conf. on Algorithmic Learning Theory*, pages 23–37, 2009.

Kris Cao, Angeliki Lazaridou, Marc Lanctot, Joel Z. Leibo, Karl Tuyls, and Stephen Clark. Emergent communication through negotiation. In *6th Int'l Conf. on Learning Representations*, 2018.

Siqi Chen, Jianing Zhao, Gerhard Weiss, Ran Su, and Kaiyou Lei. An effective negotiating agent framework based on deep offline reinforcement learning. In *Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 324–335, 2023.

Shih-Fen Cheng, Daniel M. Reeves, Yevgeniy Vorobeychik, and Michael P. Wellman. Notes on equilibria in symmetric games. In *AAMAS-04 Workshop on Game-Theoretic and Decision-Theoretic Agents*, 2004.

Peter I. Cowling, Edward J. Powley, and Daniel Whitehouse. Information set Monte Carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:120–143, 2012.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *10th Int'l Conf. on Learning Representations*, 2022.

Anthony Christopher Davison and David Victor Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, 1997.

Lasse Espeholt, Hubert Soyer, Remi Munos, et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *35th Int'l Conf. on Machine Learning*, pages 1407–1416, 2018.

Shaheen Fatima, Sarit Kraus, and Michael Wooldridge. *Principles of Automated Negotiation*. Cambridge University Press, 2014.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *32nd AAAI Conf. on Artificial Intelligence*, 2018.

Rihab Gorsane, Omayma Mahjoub, Ruan John de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. In *36th Int'l Conf. on Neural Information Processing Systems*, pages 5510–5521, 2022.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, 55:895–943, 2022.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

Andreas Hefti. Equilibria in symmetric games: Theory and applications. *Theoretical Economics*, 12(3):979–1002, 2017.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *32nd AAAI Conf. on Artificial Intelligence*, 2018.

Daniel Hennes, Dustin Morrill, Shayegan Omidshafiei, et al. Neural replicator dynamics: Multiagent learning via hedging policy gradients. In *19th Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 492–501, 2020.

Ryota Higa, Katsuhide Fujita, Toki Takahashi, Takumu Shimizu, and Shinji Nakadai. Reward-based negotiating agent strategies. In *37th AAAI Conf. on Artificial Intelligence*, pages 11569–11577, 2023.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "Other-play" for zero-shot coordination. In *37th Int'l Conf. on Machine Learning*, pages 4399–4410, 2020.

Hengyuan Hu, Adam Lerer, Noam Brown, and Jakob Foerster. Learned belief search: Efficiently improving policies in partially observable settings. *arXiv preprint arXiv:2106.09086*, 2021.

Iris A. M. Huijben, Wouter Kool, Max B. Paulus, and Ruud J. G. Van Sloun. A review of the Gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1353–1371, 2022.

Nicholas R. Jennings, Peyman Faratin, Alessio R. Lomuscio, Simon Parsons, Carles Sierra, and Michael Wooldridge. Automated negotiation: Prospects, methods and challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

Patrick R. Jordan, Christopher Kiekintveld, and Michael P. Wellman. Empirical game-theoretic analysis of the TAC supply chain game. In *6th Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1188–1195, 2007.

Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *37th Int'l Conf. on Machine Learning*, pages 4962–4973, 2020.

Christopher Kiekintveld and Michael P. Wellman. Selecting strategies using empirical game models: An experimental analysis of meta-strategies. In *7th Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1095–1101, 2008.

David M. Kreps. *Game Theory and Economic Modelling*. Oxford University Press, 1990.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, et al. A unified game-theoretic approach to multiagent reinforcement learning. In *31st Int'l Conf. on Neural Information Processing Systems*, pages 4190–4203, 2017.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

Joel Z. Leibo, Edgar A. Dueñez-Guzman, Alexander Vezhnevets, et al. Scalable evaluation of multi-agent reinforcement learning with Melting Pot. In *38th Int'l Conf. on Machine Learning*, pages 6187–6199, 2021.

Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? End-to-end learning for negotiation dialogues. In *Conference on Empirical Methods in Natural Language Processing*, 2017.

Shihui Li, Yi Wu, Xinyue Cui, et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *33rd AAAI Conf. on Artificial Intelligence*, pages 4213–4220, 2019.

Zun Li, Marc Lanctot, Kevin R. McKee, et al. Combining tree-search, generative models, and Nash bargaining concepts in game-theoretic reinforcement learning. *arXiv preprint arXiv:2302.00797*, 2023.

Ryan Lowe, Yi I. Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *31st Int'l Conf. on Neural Information Processing Systems*, 2017.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, et al. Asynchronous methods for deep reinforcement learning. In *33rd Int'l Conf. on Machine Learning*, pages 1928–1937, 2016.

John Nash. Non-cooperative games. *Annals of Mathematics*, pages 286–295, 1951.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Comparative evaluation of cooperative multi-agent deep reinforcement learning algorithms. *arXiv preprint arXiv:2006.07869*, 2020.

Tom Pepels, Tristan Cazenave, Mark H. M. Winands, and Marc Lanctot. Minimizing simple and cumulative regret in monte-carlo tree search. In *ECAI-14 Workshop on Computer Games*, pages 1–15, 2014.

Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, et al. From Poincaré recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In *38th Int'l Conf. on Machine Learning*, pages 8525–8535, 2021.

Julien Perolat, Bart De Vylder, Daniel Hennes, et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):7234–7284, 2020.

Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *19th AAAI Conf. on Artificial Intelligence*, pages 495–501, 2005.

Martin Schmid, Matej Moravcik, Neil Burch, and Rudolf Kadlec others. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 2023.

John Schulman, Philipp Moritz, et al. High-dimensional continuous control using generalized advantage estimation. In *Fourth International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *24th Int'l Conf. on Neural Information Processing Systems*, volume 23, 2010.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Thomas Hubert, Julian Schrittwieser, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

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. *arXiv preprint arXiv:2304.13138*, 2023.

Yuhang Song, Andrzej Wojcicki, Thomas Lukasiewicz, et al. Arena: A general evaluation platform and building toolkit for multi-agent intelligence. In *34th AAAI Conf. on Artificial Intelligence*, pages 7253–7260, 2020.

D. J. Strouse, Kevin McKee, Matt Botvinick, Edward Hughes, and Richard Everett. Collaborating with humans without human data. In *35th Int'l Conf. on Neural Information Processing Systems*, pages 14502–14515, 2021.

Johannes Treutlein, Michael Dennis, Caspar Oesterheld, and Jakob Foerster. A new formalism, method and open issues for zero-shot coordination. In *38th Int'l Conf. on Machine Learning*, pages 10413–10423, 2021.

Karl Tuyls, Julien Perolat, Marc Lanctot, Edward Hughes, et al. Bounds and dynamics for empirical game-theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(7), 2020.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Michael P. Wellman. Putting the agent in agent-based modeling. *Autonomous Agents and Multi-Agent Systems*, 30:1175–1189, 2016.

Bryce Wiedenbeck, Ben-Alexander Cassell, and Michael P. Wellman. Bootstrap statistics for empirical games. In *13th Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 597–604, 2014.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *36th Int'l Conf. on Neural Information Processing Systems*, pages 24611–24624, 2022.

A   MAX-ENTROPY NASH

In practice, it might be expensive to directly optimize the objective, due to the nonlinearity of the entropy function. We instead optimize a piecewise linear approximation of the objective. We have the following result.

THEOREM 1. *Given $\epsilon > 0$, an ϵ-maximum-entropy symmetric Nash can be solved by a mixed-integer linear program based on (1) with an additional $O(|\hat{\Pi}|^2/\epsilon)$ linear constraints.*

A.1   Proof of Theorem 1

PROOF. The idea is to transform (1) into the following mixed-integer linear program:

$$
\begin{aligned}
\min_{\sigma^*} \quad & \sum_{\pi \in \hat{\Pi}} \gamma_\pi \\
\text{s.t.} \quad & \forall \pi \in \hat{\Pi}, \\
& \forall k \in [K], \gamma_\pi \geq l_k(\sigma^*(\pi)) \\
& u_\pi = \sum_{\pi' \in \hat{\Pi}} \sigma^*(\pi) u(\pi, \pi') \\
& u^* \geq u_\pi \\
& u^* - u_\pi \leq U b_\pi \\
& \sigma(\pi) \leq 1 - b_\pi \\
& \sigma^*(\pi) \geq 0 \\
& \sum_\pi \sigma^*(\pi) = 1 \\
& b_\pi \in \{0, 1\}
\end{aligned}
\tag{2}
$$

Where $l_k(x) = f(\frac{k}{K}) + \frac{f(\frac{k+1}{K}) - f(\frac{k}{K})}{\frac{1}{K}}(x - f(\frac{k}{K}))$ is the k-th piecewise linear segment of the function $f(x) = x \log x$, for a total of K segments. Denote the convex envelope of all these K segments as $l(x)$. The additional constraints are $\gamma_\pi \geq l_k(\sigma^*(\pi))$. When we solve (2) exactly, one of these K inequalities will be satisfied at equality.

Let $f(x^*)$ be the minimum of f and $l(x')$ be the minimum of the piecewise approximation. Then if $|f - l|_\infty < \epsilon$, we can see that $f(x^*) \geq l(x^*) - \epsilon \geq l(x') - \epsilon$. Then x' is an ϵ-optimal solution of f.

Then to achieve an ϵ-optimal solution of (1) we need to choose some K, such that $|f - l|_\infty < \frac{\epsilon}{|\hat{\Pi}|}$.

For the segment $I_k = [\frac{k}{m}, \frac{k+1}{m}]$, by elementary calculus we can find $g(k) = \max_{x \in I_k} |x \log x - l_k(x)| = \frac{k+1}{eK}(1 + \frac{1}{k})^k - \frac{k(k+1)}{K} \log(1 + \frac{1}{k})$. This happens when $x = \frac{k+1}{eK}\left(1 + \frac{1}{k}\right)^k$.

We will now prove

$$
g'(k) = \frac{e + \left(\left(1 + \frac{1}{k}\right)^k (1 + k) - e(1 + 2k)\right) \log\left(1 + \frac{1}{k}\right)}{eK} < 0
$$

To show this, by arranging the numerator, we need to show that

$$
\left(k + (1 + k)\left(1 - \frac{1}{e}\left(1 + \frac{1}{k}\right)^k\right)\right) \log\left(1 + \frac{1}{k}\right) > 1
$$

First notice that

$$
\begin{aligned}
1 - \frac{1}{e}\left(1 + \frac{1}{k}\right)^k &= 1 - e^{k \log(1 + \frac{1}{k}) - 1} \\
&= 1 - e^{-\frac{1}{2k} + \frac{1}{3k^2} - \frac{1}{4k^3} + \frac{1}{5k^4} + \dots} \\
&\geq 1 - e^{-\frac{1}{2k} + \frac{1}{3k^2}} \\
&\geq (-\frac{1}{2k} + \frac{1}{3k^2}) - \frac{1}{2}\left(-\frac{1}{2k} + \frac{1}{3k^2}\right)^2 \\
&= \frac{1}{2k} - \frac{11}{24k^2} + \frac{1}{6k^3} - \frac{1}{18k^4}
\end{aligned}
$$

Then when $k \geq 7$

$$(1 + k) \left(1 - \frac{1}{e} \left(1 + \frac{1}{k} \right)^k \right)$$

$$\geq (1 + k) \left(\frac{1}{2k} - \frac{11}{24k^2} + \frac{1}{6k^3} - \frac{1}{18k^4} \right)$$

$$= \frac{1}{2} + \frac{1}{24k} - \frac{7}{24k^2} + \frac{1}{9k^3} - \frac{1}{18k^4} > \frac{1}{2}$$

Therefore

$$\left(k + (1 + k) \left(1 - \frac{1}{e} \left(1 + \frac{1}{k} \right)^k \right) \right) \log \left(1 + \frac{1}{k} \right) > \left(k + \frac{1}{2} \right) \log \left(1 + \frac{1}{k} \right)$$

Let $h(k) = \left(k + \frac{1}{2} \right) \log \left(1 + \frac{1}{k} \right)$. Then $h'(k) = \log \left(1 + \frac{1}{k} \right) - \frac{1}{2k} - \frac{1}{2(k+1)}$ $h''(k) = -\frac{1}{k(k+1)} + \frac{1}{2k^2} + \frac{1}{2(k+1)^2} > 0$. Therefore $h'(k)$ increases. And since $\lim_{k \to \infty} h'(k) = 0$ we can deduce $h'(k) < 0$. Therefore $h(k)$ is decreasing. And since $\lim_{k \to \infty} h(k) = 1$ we can see that $h(k) > 1$. Therefore the we have proved $g'(k) < 0$ for $k \geq 7$ is decreasing. It is easy to verify it also holds for $k < 7$. So $g(k)$ achieves maximum $\frac{1}{eK}$ when $k = 0$.

Then we have when $K = \left\lfloor \frac{|\hat{\Pi}|}{e\epsilon} \right\rfloor + 1$, we can obtain an ϵ-optimal maximum entropy Nash by solving (2) with additional $K \cdot (|\Pi|) = O \left(\frac{|\hat{\Pi}|^2}{\epsilon} \right)$ linear constraints.

$\square$

A.2 Setup

We use GUROBI [Gurobi Optimization, LLC 2023] as the solver. In our experiments we always solve for a 0.05-optimal max-entropy Nash.

B DETAILS OF GUMBEL IS-MCTS

B.1 Value Estimation

When $C(s, a) > 0$, we simply let $\hat{q}(s, a) = \frac{R(s,a)}{C(s,a)}$. When $C(s, a) = 0$, following [Danihelka *et al.* 2022], we let

$$\hat{q}(s, a) = \frac{1}{1 + \sum_b C(s, b)} \left(v(s, a) + \frac{\sum_b C(s, b)}{\sum_{b:C(s,b)>0} p(s, b)} \sum_{b:C(s,b)>0} \frac{R(s, b)}{C(s, b)} p(s, b) \right)$$

as an estimator. And we use $G(\hat{q}(s, a)) = c_2(c_1 + \max_b C(s, b)) \hat{q}(s, a)$, for some $c_1, c_2 > 0$.

B.2 Action Selection at Non-Root Nodes

At a non-root node, an action is selected to minimize the discrepancy between $Imp(p)$ and the produced visited frequency:

$$\arg\min_a \sum_b \left(Imp(p)(s_i, b) - \frac{C(s_i, b) + 1\{a = b\}}{1 + \sum_c C(s_i, c)} \right)^2 \tag{3}$$

C ALGORITHMS PSEUDOCODE

C.1 Vanilla AlphaZero Search

Pseudocode presented as Alg. 2.

C.2 Sequential Halving

Pseudocode presented as Alg. 3.

C.3 Self-play based Training

Pseudocode presented as Alg. 4.

D HYPERPARAMETERS

D.1 Input Representation

We use the infostate_tensor in OpenSpiel [Lanctot *et al.* 2019] as the representation of infostate. In Bargaining games, this includes information (1) which player the current agent is playing (2) pool configuration (3) the player's own private valuation (4) the current round number.

Algorithm 2 Vanilla AlphaZero-styled IS-MCTS

1: **function** VA-Search($s, \boldsymbol{v}, \boldsymbol{p}$)
2:   $\forall(s, a), R(s, a) = 0, C(s, a) = 0$.
3:   $\forall a \in \mathcal{A}(s)$, sample $d(a) \sim Dirichlet(\alpha)$
4:   **for** $iter = 1, \ldots, num\_sim$ **do**
5:     Sample a world state: $h \sim \Pr(h \mid s, \boldsymbol{p})$
6:     **while** **do**
7:       **if** h is terminal **then**
8:         $\boldsymbol{r} \leftarrow$ payoffs of players. **Break**
9:       **else if** $i \triangleq \tau(h)$ is chance **then**
10:         $a \leftarrow$ sample according to chance
11:       **else if** $s_i(h)$ not in search tree **then**
12:         Add $s_i(h)$ to search tree.
13:         $\boldsymbol{r} \leftarrow \boldsymbol{v}(s_i(h))$. **Break**
14:       **else if** $s_i(h)$ is root node s **then**
15:         $a \leftarrow \arg\max_a \frac{R(s_i,a)}{C(s_i,a)} + c_{puct}((1-\epsilon)\boldsymbol{p}(s,a) + \epsilon d(a))\frac{\sqrt{C(s_i,a)}}{1+\sum_b C(s_i,b)}$.
16:       **else**
17:         $a \leftarrow \arg\max_a \frac{R(s_i,a)}{C(s_i,a)} + c_{puct}\boldsymbol{p}(s,a)\frac{\sqrt{C(s_i,a)}}{1+\sum_b C(s_i,b)}$
18:       **end if**
19:       $h \leftarrow ha$
20:     **end while**
21:     **for** (s_i, a) in this trajectory **do**
22:       $R(s_i, a) \mathrel{+}= \boldsymbol{r}_i, C(s_i, a) \mathrel{+}= 1$
23:     **end for**
24:   **end for**
25:   **return** Action $a = \arg\max C(s, a)$, (During training) a policy target that is the normalized visited count of s.
26: **end function**

Algorithm 3 Sequential Halving

1: **function** Seq-Hal($\hat{\mathcal{A}}, K$)
2:   Maintain a static variable $epoch$ across different calls of this function, initialized as 0
3:   **if** $epoch == 0$ **then**
4:     $\hat{\mathcal{A}} \leftarrow K$ actions with largest $g(a) + \text{logit } \boldsymbol{p}(s, a)$
5:     $epoch \mathrel{+}= 1$
6:   **end if**
7:   $a \leftarrow$ an action that has not been visited $\lfloor \frac{sim\_num}{\frac{K}{2^{epoch-1}}\log_2 K} \rfloor$ times at current epoch in $\hat{\mathcal{A}}$
8:   **if** All actions in $\hat{\mathcal{A}}$ have been visited $\lfloor \frac{sim\_num}{\frac{K}{2^{epoch-1}}\log_2 K} \rfloor$ times in current epoch **then**
9:     $\hat{\mathcal{A}} \leftarrow$ Top $\lfloor \frac{K}{2^{epoch}} \rfloor$ actions in $\hat{\mathcal{A}}$ based on $g(a) + \text{logit } \boldsymbol{p}(s, a) + G(\hat{q}(s, a))$
10:     $epoch \leftarrow epoch + 1$
11:   **end if**
12:   **return** $a, \hat{A}$.
13: **end function**

D.2 PPO Algorithms

We use the same PPO hyperparameters for every place when it is used. See Table 3. For single-agent PPO and IDPPO, the PPO algorithm trains policy nets by minimizing the clipped loss $\mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} \hat{A}_t, clip \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}, 1+\epsilon, 1-\epsilon \right) \hat{A}_t \right) \right]$, where A_t is estimated by *generalized advantage estimation* (GAE) [Schulman *et al.* 2016]. Value nets are updated by minimizing L2 loss from the value targets produced by GAE. For MAPPO, the loss is simply the summation of individual PPO losses of every player.

D.3 R-NaD

We fine-tuned the learning rate to be 1e-3, the total number of game trajectories to be around 1e6. Others are the same as the default hyperparameters in OpenSpiel [Lanctot *et al.* 2019].

Algorithm 4 Self-Play Training for Search Methods
_____
 1: **function** Self-Play-Train($\mathcal{G}$)
 2:    Initialize v, v', p, p'
 3:    $D_v = \{\}, D_p = \{\}$
 4:    **for** $iter = 1, \ldots, num\_epoch$ **do**
 5:       $h \leftarrow$ Initial state h_0 of $\mathcal{G}$
 6:       **while do**
 7:          **if** h is terminal **then**
 8:             $r \leftarrow$ payoffs of players **Break**
 9:          **else if** $i \triangleq \tau(h)$ is chance **then**
10:             $a \leftarrow$ sample according to chance
11:          **else**
12:             $a, \pi \leftarrow$ SEARCH$(s_i(h), v', p')$
13:             $D_p \leftarrow D_p \cup \{(s_i(h), \pi)\}$, π is the visiting frequency during search
14:          **end if**
15:          $h \leftarrow ha$
16:       **end while**
17:       **for** s_i in this trajectory **do**
18:          $D_v \leftarrow D_v \cup \{s_i, r_i\}$
19:       **end for**
20:       $v, p \leftarrow$ UPDATE(D_v, D_p)
21:       Replace parameters of v', p' with the latest parameters of v, p periodically.
22:    **end for**
23:    **return** v, p
24: **end function**
_____

Hyperparameter Name	Values
Learning rate	$2e - 4$
Optimizer	Adam
Batch size	16
Number of minibatch	4
Number of updates per epoch	10
Number of steps per PPO trajectory	64
Number of game trajectories in total	1e6
Entropy weight	0.01
Clipped parameter ϵ	0.2
GAE lambda	0.95
RL discount factor	1
Torso for policy nets	[256, 256]
Torso for value nets	[256, 256]

Table 3: Hyper-parameters for PPO.

D.4 NFSP

See Table 4.

D.5 PSRO

See Table 5.

D.6 FCP

See Table 6

D.7 Gumbel Search

See Table 7.

Hyperparameter Name	Values
RL network Learning rate	0.01
DQN buffer size	2^{17}
SL network learning rate	0.01
Optimizer	Adam
Batch size	256
Number of game trajectories in total	1e6
Torso for RL net	[256, 256]
Torso for SL net	[256, 256]

Table 4: Hyper-parameters for NFSP.

Hyperparameter Name	Values
Number of PSRO iterations	32
best-response method	PPO
Number of game trajectories to train each BR	1e6
Meta-strategy solver	Max-entropy Nash

Table 5: Hyper-parameters for PSRO.

Hyperparameter Name	Values
Number of self-play runs	32
Number of checkpoint strategies considered per run	100
Strategies picked each run	The first one, the last one, and the one that just achieves half of the social welfare of the last one
Self-play method	IDPPO
best-response method	PPO

Table 6: Hyper-parameters for FCP.

Hyperparameter Name	Values
Learning rate	$2e-4$
Optimizer	SGD
Batch size	256
Max buffer size	2^{17}
c_1	50
c_2	0.1
sim_num	200
K	16
Network delayed period	1000
Number of game trajectories in total	1e6
Torso for PVN	[256, 256]

Table 7: Hyper-parameters for Gumbel Search.

D.8 VA Search

See Table 8.

E SUMREGRET RESULTS

All SUMREGRETs are estimated using PPO as a best response method. The results are reported across three random seeds.

E.1 `Barg`$(10, 0, 1)$

See Figures 4.

E.2 `Barg`$(30, 0.125, 0.935)$

See Figures 5.

Hyperparameter Name	Values
Learning rate	$1e-3$
Optimizer	SGD
Batch size	256
Max buffer size	2^{17}
c_{puct}	20
sim_num	200
Dirichlet α	$\frac{1}{|\mathcal{A}|} \cdot \mathbf{1}$
ϵ	0.25
Network delayed period	1000
Number of game trajectories in total	1e6
Torso for PVN	[256, 256]

Table 8: Hyper-parameters for VA Search.
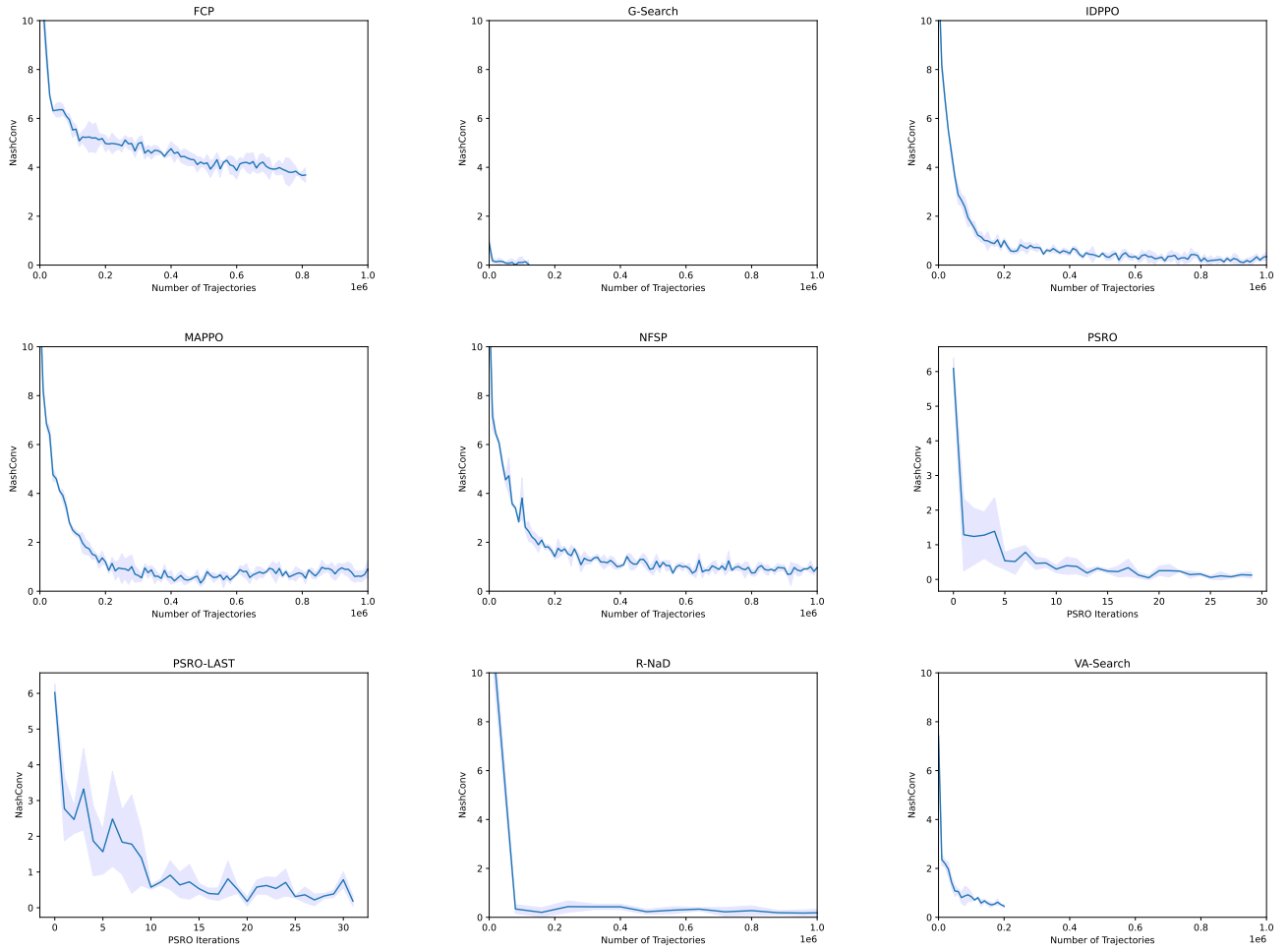
Figure 4: SUMREGRET of **Barg**$(10, 0, 1)$

F EMPIRICAL DISTRIBUTION OF REGRET

F.1 **Barg**$(10, 0, 1)$

See Figures 6.

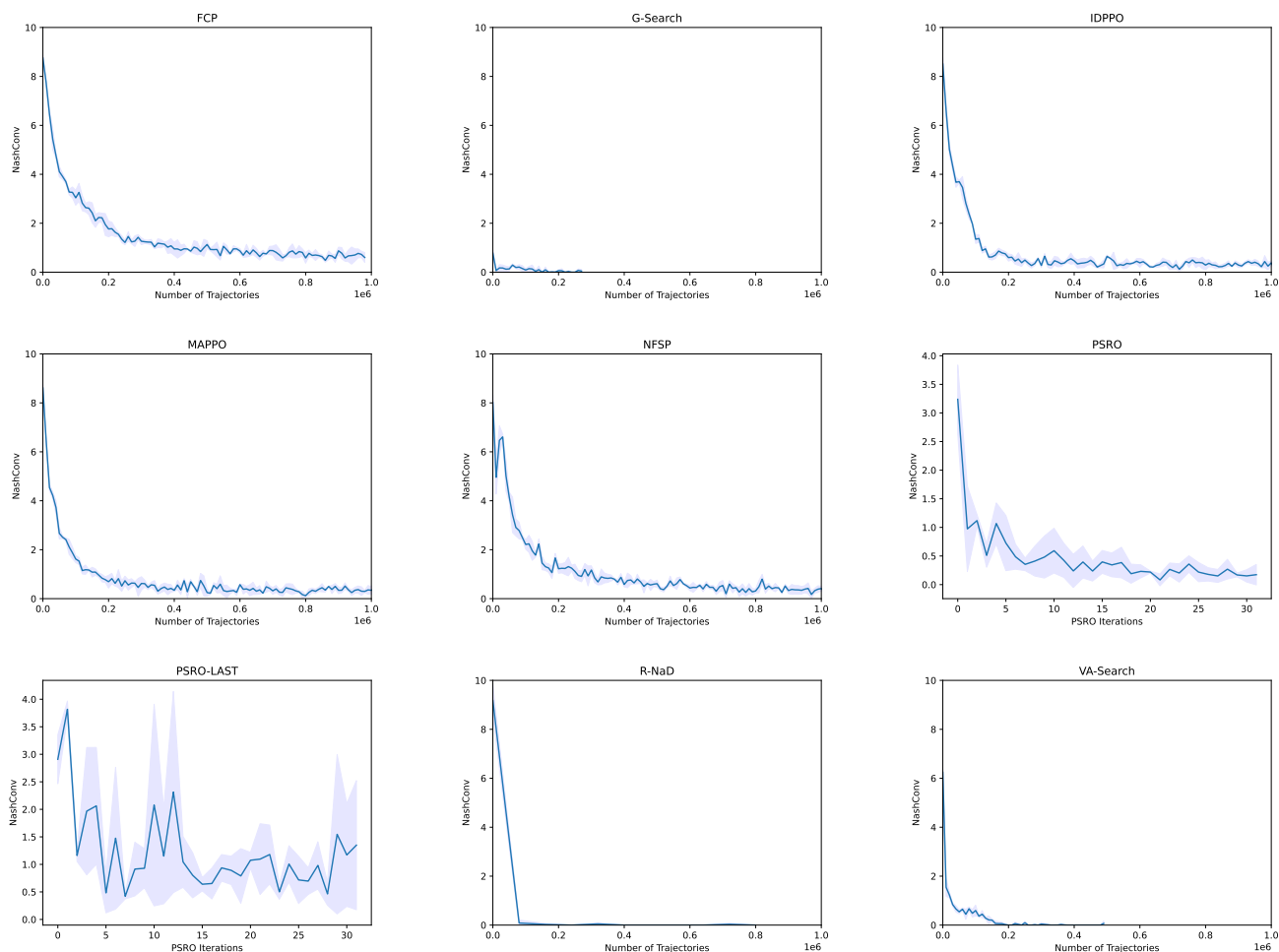**Figure 5:** SUMREGRET of `Barg`$(30, 0.125, 0.935)$

F.2 `Barg`$(30, 0.125, 0.935)$

See Figures 7

G APPROXIMATE SIZE OF DEAL OR NO DEAL

To estimate the size of Deal or No Deal, we first verified that there are 142 unique preference vectors per player. Then, we generated 10,000 simulations of trajectories using uniform random policy, computing the average branching factor (number of legal actions per player at each state) as $b \approx 23.5$.

Since there are 142 different information states for player 1's first decision, about $142bb$ player 1's second decision, etc. leading to $142(1 + b^2 + b^4 + b^8) \approx 13.2 \times 10^{12}$ information states. Similarly, player 2 has roughly $142(1 + b^1 + b^3 + b^5 + b^7) = 5.63 \times 10^{11}$ information states.
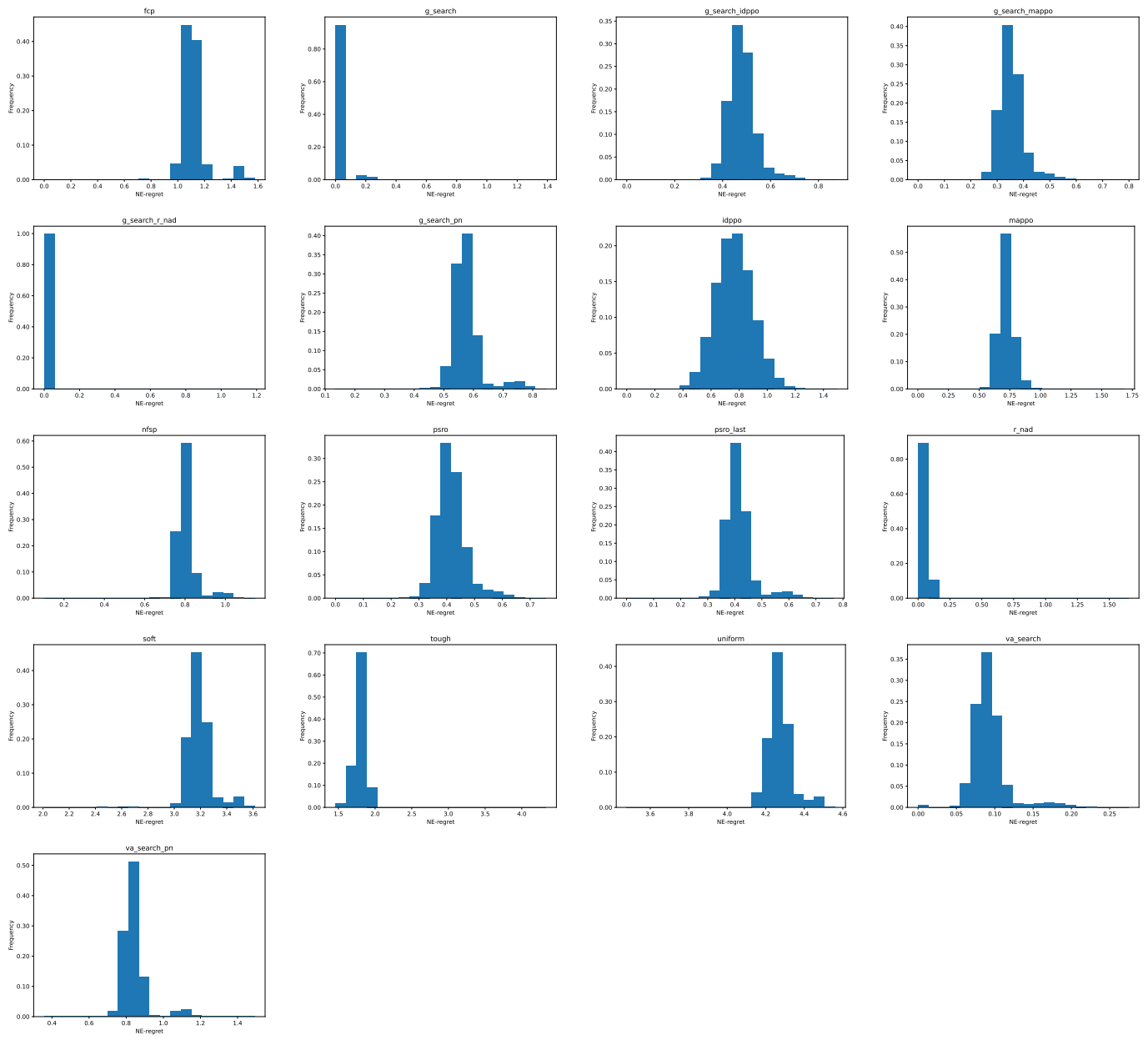
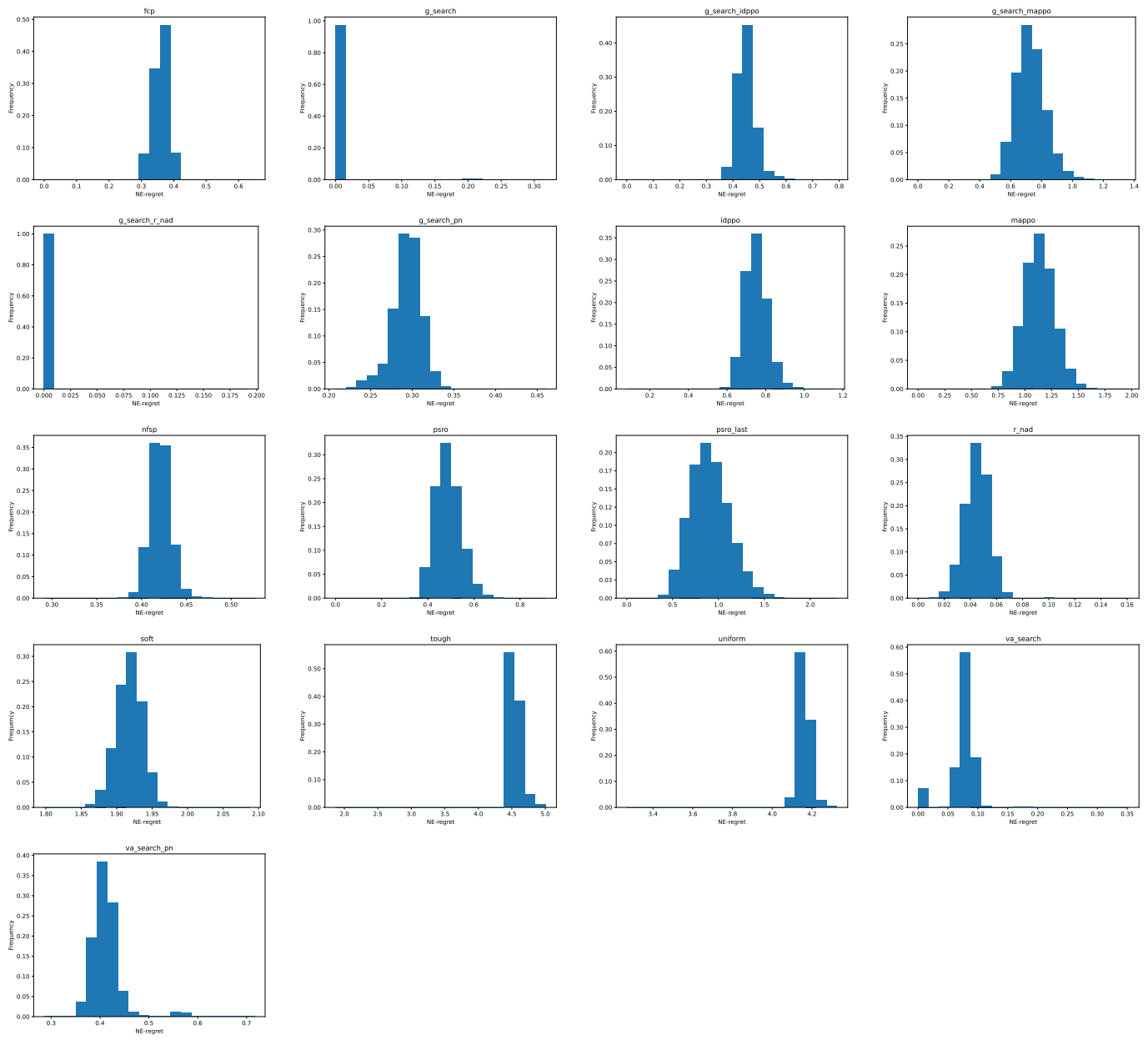Figure 6: Empirical Distribution of NE-Regret of **Barg**$(10, 0, 1)$

Figure 7: Empirical Distribution of NE-Regret of **Barg**(30, 0.125, 0.935)