Proximal algorithms for self-play in IIGs

In these notes we are interested in learning by self-play how to act optimally in an imperfect information games (IIG). Such games, e.g. poker or bridge, differ fundamentally from perfect information games (PIGs) like chess or backgammon. Indeed, since all players share in PIGs the same state of the game, e.g. the board configuration in chess, one can see PIGs as basically a one player game with objective varying upon the current state. However, in IIGs, each player has only access to partial information about the current state of the game. For instance, it would be easy to play against oneself in chess but almost impossible in poker since at each round one have to forget the hand of the other players we just played!

Our specific interest lies in learning through self-play near-optimal policies, only utilizing sampled trajectories of the game without exhaustively traversing the entire game tree. To accomplish this, we delve into adapting the proximal point algorithm, a technique initially designed for small matrix games, to large IIGs. Specifically, we are dedicated to finding the most straightforward adaptation that still delivers competitive results.

1 Self-Play in Imperfect Information Games

Imperfect information game Consider an episodic, finite, two-player IIG $(\mathcal{S}, \mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}, H, p, r)$, which consists of the following components:

- A state space $\mathcal{S}$ and an information set space (partitions of $\mathcal{S}$) $\mathcal{X}$ for the max-player and the $\mathcal{Y}$ min-player.
- Action space $\mathcal{A}$ for max-player and action space $\mathcal{B}$ for the min-player.
- The length H of the game.
- Initial state distribution $p_0 \in \Delta(\mathcal{S})$ and a state-transition probability kernel $(p_h)_{h\in[H-1]}$ with $p_h : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \Delta(\mathcal{S})$ for each $h \in [H-1]$.
- A reward function for the max-player $(r_h^M)_{h\in[H]}$ with $r_h^M : \mathcal{S}\times\mathcal{A}\times\mathcal{B} \to [0,1]$ and a reward function for the min-player $(r_h^m)_{h\in[H]}$.

Zero-sum We assume that the game is zero-sum, that is the reward of the min-player is the opposite of the reward of the max-player $r_h(s,a,b) := r_h^M(s,a,b) = -r_h^m(s,a,b)$.

Perfect-recall We assume perfect-recall [Kuhn, 1950], that is a player does not forget the action taken nor the information set observed during an episode. Formally, this means that for the max-player and for each information set $x \in \mathcal{X}$ there exists a unique $h \in [H]$ and history $(x_1, a_1, ..., x_h)$ such that $x_h = x$ and a similar property holds for the min-player.

Policy A policy of the max-player is a sequence $\mu = (\mu_h)_{h\in[H]}$ such that for all information set $x \in \mathcal{X}$, the function $\mu_h(.|x) \in \Delta(\mathcal{A})$ is a probability distribution over the set of action. A policy ν of the min-player can be defined similarly. Note that because of the perfect-recall we do not need to specify the step in the policy which is implicitly encoded in the information-set, and we can just write $\mu(x|a)$.

Profile A profile $\pi = (\mu, \nu)$ is the combination of a policy for the max-player μ and a policy for the min-player ν.

Remark (Turn-based formulation) Note that it is always possible to transform the game in a turn-based game where only one player acts at each step by hiding some information to the other player until the next step. In this case, we can use a shared information set space $\mathcal{X}$, a shared action space $\mathcal{A}$ and rewards $r_h^M(s,a) = -r_h^m(s,a)$ for $(s,a) \in \mathcal{S} \times \mathcal{A}$. We need a player index function $i : \mathcal{X} \to \{\max, \min\}$ such that $i(x)$ tells us which player should act at information set x, provided that we augment the information set with the information about the current player. In this case, the profile $\pi = (\mu, \nu)$ could be seen as a function of the information state that returns the policy of the current player

$$\pi(a|x) = \begin{cases} \mu(a|x) \text{ if } i(x) = \max \\ \nu(a|x) \text{ if } i(x) = \min \end{cases} .$$

2 Self-play

We now describe the learning framework.

Game realization Given the two players' policies μ and ν, an episode of the game proceeds as follows: an initial state $s_1 \sim p_0$ is sampled. At step h, the max- and min-player observe their information sets x_h and y_h. Given the information, the max- and min-player choose and execute actions $a_h \sim \mu_h(\cdot|x_h)$ and $b_h \sim \nu_h(\cdot|y_h)$. As a result, the current state transitions to a next state $s_{h+1} \sim p_h(\cdot|s_h, a_h, b_h)$, and the max- and min-player receive rewards $r_h(s_h, a_h, b_h)$ and $-r_h(s_h, a_h, b_h)$ respectively. This is repeated until the end of the game at time step H, after which the episode finishes.

Value and best response The expected reward of the max-player for profile $\pi = (\mu, \nu)$ is denoted by $V^{\mu,\nu} := \mathbb{E}_\pi \left[\sum_{h=1}^H r_h(s_h, a_b, b_h)\right]$. A best response $\nu^\dagger(\mu)$ to the max-player μ, is a policy among the worst policies the max-player playing according to μ could face. It is obtained by minimizing the average cumulative reward of the max-player $\nu^\dagger(\mu) \in \arg\min_\nu V^{\mu,\nu}$. Note that a best response is not unique in general.

Nash equilibrium A Nash equilibrium is a profile $\pi^\star = (\mu^\star, \nu^\star)$ such that each policy is a best response to the opponent policy

$$\nu^\star \in \arg\min_\nu V^{\mu^\star,\nu} \text{ and } \mu^\star \in \arg\max_\mu V^{\mu,\nu^\star} .$$

Our *goal is to learn a Nash equilibrium* that provides us with policies that plays optimally in the game. Indeed, we observe that $\mu^\star \in \arg\max_\mu \min_\nu V^{\mu,\nu}$ is the best choice of policy if we make no assumption on the policy played by the min-player, i.e. the opponent could pick the best response against our policy. Note that the Nash equilibrium is the solution of the following saddle point where we recognize in the second inequality the von Neumann's minimax theorem

$$V^{\mu^\star,\nu^\star} = \max_\pi \min_\nu V^{\mu,\nu} = \min_\nu \max_\pi V^{\mu,\nu} .$$

Exploitability gap The exploitability gap of profile $\pi = (\mu, \nu)$ is the sum of the improvement gap of each player which is what a player could gain by switching unilaterally to the best response,

$$\Delta(\pi) = \max_{\tilde{\mu}} V^{\tilde{\mu},\nu} - V^{\mu,\nu} + V^{\mu,\nu} - \min_{\tilde{\nu}} V^{\mu,\tilde{\nu}} .$$

This quantity is always positive as the sum of positive terms and of particular interest because it measures how close the profile π is close to a Nash equilibrium:

$$\Delta(\pi) \geq 0 \text{ and } \Delta(\pi) = 0 \text{ if and only if } \pi \text{ is a Nash equilibrium.}$$

Learning via self-play In self-play we assume that we can simulate the game and control both players. Precisely, we pick a policy for each player and observe the sampled trajectory $(x_1, y_1, a_1, b_1, r_1, \ldots, x_H, y_H, a_H, b_H, r_H)$. In particular, we do not assume that we observe the underlying state of the game s_h. Even if in some simple games, i.e. board game, this state could be observed, in some setting where the game simulator is some black-box, observing the state is impossible. After simulating as many trajectories as we want we output a profile that should be close to a Nash equilibrium measured by the exploitability gap.

3 Proximal algorithm

Before diving into proximal algorithm for general IIGs we present the simpler case of matrix games.

3.1 Matrix game

Matrix games represent a specific category of IIGs characterized by having only one step $H = 1$, and a single initial state. We describe the game using a reward matrix $(r(a,b))_{a\in\mathcal{A},b\in\mathcal{B}}$ for the maximizing player. Player policies are represented as probability vectors. Note that one can always represent an IIG as a matrix indexed over the set of all profile made of deterministic policies. This is the normal form of the game which is far less compact and practical than the extensive form.

The proximal point algorithm [Martinet, 1970, Nemirovski, 2004] sequentially performs the following updates starting from some profile (μ^1, ν^1),

$$\mu^{t+1} \in \arg\max_{\mu} \sum_{a,b} \mu(a) r(a,b) \nu^{t+1}(b) - \beta\,\mathrm{KL}(\mu, \mu^t)\,,$$
$$\nu^{t+1} \in \arg\min_{\nu} \sum_{a,b} \mu^{t+1}(a) r(a,b) \nu(b) - \beta\,\mathrm{KL}(\nu, \nu^t).$$

Intuitively, at each step, we compute the Nash-equilibrium of a (non-zero-sum) game, consisting of the original game regularized toward the previous profile (μ^t, ν^t). Note that when the regularization parameter β is sufficiently large, the Nash equilibrium of this regularized game should be close to the previous profile. A priori, it might seem challenging to perform these updates due to their recursive nature. However, it's worth noting that the profile (μ^{t+1}, ν^{t+1}) is the fixed point of a specific operator, which is a contraction for a well-chosen regularization parameter β. Consequently, we can approximate the proximal update by conducting a few inner fixed-point iterations. Starting with $(\widetilde{\mu}^{t,1}, \widetilde{\nu}^{t,1}) = (\mu^t, \nu^t)$, we perform the following iterative steps

$$\tilde{\mu}^{t,k+1} \in \arg\max_{\mu} \sum_{a,b} \mu(a) r(a,b) \tilde{\nu}^{t,k}(b) - \beta\,\mathrm{KL}(\mu, \mu^t)\,,$$
$$\tilde{\nu}^{t,k+1} \in \arg\min_{\nu} \sum_{a,b} \tilde{\mu}^{t,k}(a) r(a,b) \nu(b) - \beta\,\mathrm{KL}(\nu, \nu^t),$$

then return $(\mu^t, \nu^t) = (\tilde{\mu}^{t,K}, \tilde{\nu}^{t,K})$ for some K. In the sequel we refer to the fixed-point iterations as the inner iteration loop and to the updates of the proximal profile as the outer loop. Notably, when $K = 2$, the algorithm corresponds to the `Mirror-Prox` algorithm introduced by Nemirovski [2004]. In particular, they prove that only two inner iterations $K = 2$ are sufficient for the approximated proximal algorithm to converge (in average) toward a Nash equilibrium.

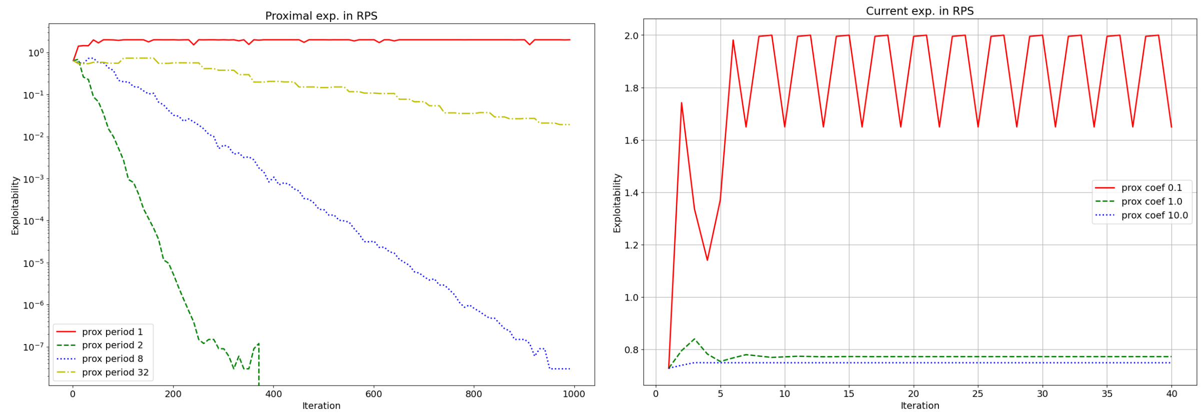

Figure 1: Exploitability gap of the proximal profile as a function of the number of inner iterations for various proximal periods K in the RPS game (left). Exploitibility gap of the current profile in one inner loop, in the RPS game (right).

In Figure 1 (left), the exploitability gap of the proximal profile is plotted against the number of inner iterations in the Rock Paper Scissors (RPS) game, considering different values of iterations in the inner loop $K \in \{1, 2, 8, 32\}$ and a fixed regularization parameter $\beta = 1.0$. Notably, we observe convergence towards the Nash equilibrium even with $K = 2$.

In Figure 1 (right), we maintain the proximal profile constant and plot the exploitability gap of the current profile $(\tilde{\mu}^{t,k}, \tilde{\nu}^{t,k})$ for a single inner loop, exploring varied regularization parameters $\beta \in \{0.1, 1.0, 10.0\}$. It is evident that $\beta = 0.1$ does not provide enough regularization for the fixed point iteration scheme to converge, while larger $\beta \geq 1.0$ values result in faster convergence of the current profile. Interestingly, even though the proximal algorithm converges at $K = 2$ (refer to Figure 1, left), a regularization parameter of $\beta = 1.0$ requires more than $K = 2$ iterations for the current profile to completely reach the fixed point (see to Figure 1, right). This indicates that the proximal algorithm performs well with a sufficiently accurate approximation of the outer loop solution [Nemirovski, 2004].

It's worth mentioning that our approach slightly deviates from the self-play learning framework described earlier. In our setting, given a profile (μ, ν), we can only observe a game outcome by randomly sampling $a \sim \mu$, $b \sim \nu$ and observing the reward $r(a, b)$. In contrast, the proximal algorithm assumes a more powerful oracle, which provides access to $\sum_b \nu(b) r(a, b)$ for all a and $\sum_a \mu(a) r(a, b)$ for all b. While this assumption is reasonable for small matrix games, it becomes impractical for larger IIGs ($H > 1$), as obtaining such a strong oracle would require testing all possible action sequences against the opponent, along with computing the associated average rewards. Additionally, this process would need to be repeated at each inner iteration, making it prohibitive for large IIGs. Furthermore, it's important to acknowledge that in our setting, the observed game outcomes inherently carry some level of noise due to the randomness of player strategies and potential randomness within the game itself.

3.2 A proximal algorithm for IIGs

The challenges outlined in the preceding section mirror those encountered in reinforcement learning (RL). Consequently, a viable approach involves adapting an RL algorithm to address these obstacles. Specifically, we propose to use the vanilla policy gradient algorithm to solve each inner loop within the proximal algorithm. This choice is logical because the rewards are non-stationary, evolving with each iteration of the outer loop.

Let's delve into the details of this algorithm, which we've dubbed `Prox`. For the sake of clarity, we adopt a turn-based formulation in this section.

Parametrization We use two profiles: the current profile for sampling new trajectories and the proximal profile for regularization. The current profile is a simple multilayer perceptron (MLP) network $\pi_\theta : x \to \Delta_{\mathcal{A}}$ parametrized by some parameter θ. The proximal profile $\pi_{\theta_{\text{prox}}}$ uses the same architecture but parametrized by the parameter θ_{prox}. For variance reduction, we also need a value network $V_\varphi : x \to \mathbb{R}$, a simple MLP parametrized by φ.

Trajectories collection Before each update we sample a batch $\mathcal{D} = \{\tau^k = (x_1^k, a_1^k, r_1^k, \ldots, x_H^k, a_h^k, r_H^k) : k \in [K]\}$ of independent trajectories with the current policy $a_h^k \sim \pi_\theta(x_h^k)$.

Regularization As for matrix game we need to introduce regularization toward the proximal profile. To this aim, one can simply modify the reward to obtain in expectation the Kullback-Leibler regularization term. Furthermore, for stability reasons, we also regularize toward the uniform distribution by adding an entropic regularization. Precisely we transform the reward r_h^k observed at information state x_h^k for action a_h^k to

$$\tilde{r}_h^k = r_h^k - \beta_{\text{prox}} \log \frac{\pi_\theta(x_h^k, a_h^k)}{\pi_{\theta_{\text{prox}}}(x_h^k, a_h^k)} - \beta_{\text{ent}} \log \frac{\pi_\theta(x_h^k, a_h^k)}{\pi_{\text{unif}}(x_h^k, a_h^k)},$$

where β_{prox} is the proximal regularization coefficient, β_{prox} is the entropic regularization coefficient and π_{unif} the uniform profile. In practice, we clip the regularization to avoid numerical instability

$$\tilde{r}_h^k = r_h^k - \text{Clip}\left(\beta_{\text{prox}} \log \frac{\pi_\theta(x_h^k, a_h^k)}{\pi_{\theta_{\text{prox}}}(x_h^k, a_h^k)}, -C_{\text{prox}}, C_{\text{prox}}\right) - \text{Clip}\left(\beta_{\text{ent}} \log \frac{\pi_\theta(x_h^k, a_h^k)}{\pi_{\text{unif}}(x_h^k, a_h^k)}, -C_{\text{ent}}, C_{\text{ent}}\right).$$

Advantage estimation Since `Prox` is a policy gradient type algorithm we need to estimate the advantages. Precisely we rely on the generalized advantage estimation (GAE, Schulman et al., 2016) method.

Given a trajectory $\tau^k = (x_1^k, a_1^k, r_1^k, \dots, x_H^k, a_H^k, r_H^k)$ we first compute the current value $V_h = V_\varphi(x_h)$ and the temporal difference error with the regularized rewards

$$\delta_h^k = \tilde{r}_h^k + V_h^k - V_{h+1}^k$$

where we use the convention $V_{H+1}^k = 0$. Then the advantage at step h is estimated with

$$A_h^k = \delta_h^k + \lambda A_{h+1}^k \,,$$

still with the convention $A_{H+1}^k = 0$ and where λ is the GAE parameter. Note that for $\lambda = 1$, the GAE methodreduces to the Monte Carlo estimate $A_h^k = \sum_{h'=h}^H \tilde{r}_{h'}^k$ whereas for $\lambda = 0$ the GAE technique reduces to the one step TD estimate $A_h^k = \tilde{r}_h^k + V_{h+1}^k - V_h^k$. One can see GAE as a particular reward shaping procedure that aim at reducing the variance of the advantage's estimate.

Current policy update The current policy is udpated by minimizing the proximal loss which can be estimated as in vanilla policy gradient but with the regularized advantages. Precisely we compute the loss

$$L_\pi(\theta) = \frac{1}{K}\sum_{k=1}^K \sum_{h=1}^H \log(\pi_\theta(x_h^k, a_h^k)) A_h^k \,.$$

In practice, for stability reasons, we clip the gradient to some value A_{clip} and perform no update if the profile probability is below some ε or above $1 - \varepsilon$,

$$L_\pi(\theta) = \frac{1}{K}\sum_{k=1}^K \sum_{h=1}^H \log(\pi_\theta(x_h^k, a_h^k)) \mathbb{1}\{\varepsilon \le \pi_\theta(x_h^k, a_h^k) \le 1 - \varepsilon\} \mathrm{Clip}(A_h^k, -A_{\text{clip}}, A_{\text{clip}}).$$

The current parameter θ^t is then updated by performing one step of gradient descent on the loss $L_\pi(\theta)$, i.e. by feeding some optimizer, Adam in our case, with the gradient $\nabla_\theta L_\pi(\theta^t)$. Note that since we perform only one step of gradient we do not need to introduce explicitly importance weights as for example in the PPO algorithm. Nevertheless, implicit importance weights still appear by differentiating the log-probabilties. Performing only one step of gradient with a batch of trajectories is not sample efficient. Nevertheless, in our setting we usually have access to an efficient simulator of the game and in this case collecting fresh trajectories is not very costly.

Proximal policy update One cannot just update the proximal parameter θ_{prox} after each gradient update of the current parameter θ since we need few iterations to solve the fixed point as in the proximal algorithm for matrix game and to perform function approximation. On the other hand we do not want the proximal regularization to slow down the training. Therefor we update the proximal parameter via an exponential moving average of the current parameter

$$\theta_{\text{prox}}^{t+1} = (1 - \eta_{\text{prox}})\theta_{\text{prox}}^t + \eta_{\text{prox}}\theta^{t+1}$$

for some learning rate $\eta_{\text{prox}} \in (0, 1)$.

Value update The value parameter φ is updated by gradient descent on the mean square error with the TD-λ targets

$$L_V(\varphi) = \frac{1}{K}\sum_{k=1}^K \sum_{h=1}^H \Big(V_\varphi(x_h^k) - (A_h^k + V_h^k)\Big)^2 .$$

The complete procedure for Prox is detailed in Algorithm 1.

Related work The first line of work to propose algorithms that only use trajectory feedback are build upon the well-known CFR algorithm, which necessitates full traversal of the game tree. To alleviate this requirement, algorithms such that Deep Counterfactual Regret Minimization (Deep CFR, Brown et al., 2019) or Deep Regret minimization with Advantage baselines and Model-free learning (DREAM, Steinberger et al., 2020) approximate full traversal by Monte Carlo sampling. Another approach, the NFSP algorithm [Heinrich and Silver, 2016] is based on fictitious play [Brown, 1951]. It computes best responses to the averages profile with the DQN [Mnih et al., 2015] algorithm and then employs these best responses to update the average profile. However, a drawback of these methods is that convergence

Algorithm 1 Prox

```
1: Input: Parameter of the current profile network $\theta$, parameter of the proximal profile network $\theta_{\text{prox}}$,
   parameter of the value network $\phi$.
2: for $t \geq 1$ do
3:     Sample a batch of trajectories $\mathcal{D} = \{\tau^k : k \in [K]\}$ with current profile $\pi_{\theta^t}$.
4:     Compute regularized rewards
           $$\tilde{r}_h^k = r_h^k - \beta_{\text{prox}} \log \frac{\pi_{\theta^t}(x_h^k, a_h^k)}{\pi_{\theta^t_{\text{prox}}}(x_h^k, a_h^k)} - \beta_{\text{ent}} \log \frac{\pi_{\theta^t}(x_h^k, a_h^k)}{\pi_{\text{unif}}(x_h^k, a_h^k)}\,.$$
5:     Compute values and GAEs
           $$V_h^k = V_{\phi^t}(x_h^k)\,, \quad A_h^k = (\tilde{r}_h^k + V_{h+1}^k - V_h^k) + \lambda A_{h+1}^k\,.$$
6:     Update current profile network by one step of gradient ascent using
           $$\nabla_\theta \frac{1}{K} \sum_{k=1}^{K} \sum_{h=1}^{H} \log\left(\pi_{\theta^t}(x_h^k, a_h^k)\right) A_h^k\,.$$
7:     Update proximal profile network
           $$\theta_{\text{prox}}^{t+1} = (1 - \eta_{\text{prox}})\theta_{\text{prox}}^t + \eta_{\text{prox}}\theta^{t+1}\,.$$
8:     Update value network by one step of gradient descent using
           $$\nabla_\varphi \frac{1}{K} \sum_{k=1}^{K} \sum_{h=1}^{H} \left(V_{\varphi_t}(x_h) - (A_h^k + V_h^k)\right)^2\,.$$
9: end for
```

toward a Nash-equilibrium only happens for the average profile which is hard to estimate [Brown et al., 2019, Heinrich and Silver, 2016]. In contrast, proximal point methods offer a distinctive advantage: the proximal profile directly converges toward a Nash equilibrium, bypassing the need to compute an average profile.

The closest approach to our method is the R-NaD algorithm proposed by Perolat et al. [2022]. The differences between Prox and R-NaD are the following: R-NaD has an off-policy flavor whereas Prox is an on-policy algorithm. Prox used the GAE for the advantage estimation instead of the V-trace procedure [Espeholt et al., 2018]. The update of the proximal parameter θ_{prox} is slightly more involved in R-NaD, with different phases. The regularization in R-NaD is shaped such that the regularized game solved in a proximal step is zero-sum.

4 Experiments

In this section, we conduct an in-depth ablation study on the Prox algorithm. Our focus is to investigate the significance of regularization as a fundamental mechanism for achieving convergence.

Games To evaluate Prox we consider the following usual benchmark games:

- *Kuhn Poker* [Kuhn, 1950]; a minimal simplified form of Poker. In Kuhn poker, the deck includes only three playing cards, for example a king, queen, and jack. One card is dealt to each player, which may place bets similarly to a standard poker. If both players bet or both players pass, the player with the higher card wins, otherwise, the betting player wins.

- *Leduc Poker* [Southey et al., 2005]; a less simplified form of Poker. This version employs a deck consisting of six distinct cards: two Kings, two Queens, and two Jacks, drawn from two possible suits, such as Diamonds and Hearts. To commence a round of Leduc Poker, each player receives a concealed private card and initiates the betting by posting an ante of one chip. The game progresses through two betting rounds, and in each of them, players are allowed a maximum of two raises. The initial raise size is set at two chips during the first round, which increases to four chips in the second round. Following the first betting round, an additional card is unveiled face-up on the table, known as the community or board card. As the game nears its conclusion, players unveil their private cards. The winner is determined by a simple rule: if one player's private card matches the rank of the board card, they claim victory; otherwise, the player whose private card holds the higher rank wins the game.

- *Liar's Dice*; a bluffing dice game also called Dudo. Each player starts with one six-sided die. Both players shake their die in the cup and then secretly looks at the result without revealing it to the other player. The active player then makes a bid by stating a number and a face value. For example, they might say "one three," which means they are claiming that there are at least one die showing the number three when both dice are revealed. Sixes are wild and count as the face of the current bid [1]. The other player has two options: they can either challenge the bid made by the active player or raise the bid by increasing either the number or the face value (or both). For instance, they could say "two fours" if they believe there are at least two dice showing the number four. When a player challenges the bid, both players reveal their dice. If the bid was accurate (i.e., there are at least as many of the announced number and face value as stated), the challenging player loses. If the bid was not accurate (i.e., there are fewer of the announced number and face value than stated), the active player loses.

Table 1: Description of the games.

Game	**Kuhn Poker**	**Leduc Poker**	**Liar's Dice**
# info. states	12	288	24576
# actions	2	3	13
traj. length	3	8	13
info. state dim.	11	24	21

Table 2: Hyperparameters of Prox.

Parameter	**Value**
Profile network	$(256, 256)$
Value network	$(256, 256)$
Batch size K	256
Regularization β_{prox}	0.2
Clipping C_{prox}	1.0
EMA η_{prox}	10^{-5}
Regularization β_{ent}	0.005
Clipping C_{ent}	1.0
GAE λ	0.75
Clipping A_{clip}	2.0
Probability ε	0.01
Adam α	0.0001
Adam β_1	0.9
Adam β_2	0.999

In Table 1 we give an order for the number of information sets, number of actions and maximal length of the game. All the games are implemented in JAX (strongly inspired from the library Pgx) and closely follows the implementation proposed in the OpenSpiel library.

Unless stated otherwise the default hyperparameters used for `Prox` are provided in Table 2. The profile network and the value network are both two layers MLP of size 256. We use the `Adam` optimizer.

[1] Usually it is the ones that are wild, but in the OpenSpiel library they use the sixes, which may change a bit the dynamic of the game.

Approximating the proximal step We initiate our analysis by examining if `Prox` can effectively solve the regularized game, which constitutes the inner loop of the proximal algorithm. To assess this, we experiment with halting the update of the parameter θ_{prox} at different stages, namely, after `prox-stop` iterations (where `prox-stop` takes values 0, 1000000, or infinity).

Upon freezing the proximal parameter, the current profile swiftly converges to a specific profile, purportedly the Nash equilibrium of the regularized game. Notably, this profile demonstrates lower exploitability, indicating proximity to the actual Nash equilibrium, compared to the frozen proximal profile. In the scenario where the proximal profile undergoes continuous updates (blue curve), both the current and proximal profiles exhibit a consistent reduction in exploitability.

The choice of the parameter $\eta_{\text{prox}} = 10^{-5}$ results in the complete update of θ_{prox} after 100000 iterations. However, Figure 2 demonstrates that the convergence of the current profile, evident in both the red and green curves, occurs at a significantly faster rate. This observation suggests that for this specific game, a more aggressive learning rate, such as choosing $\eta_{\text{prox}} \sim 10^{-3}$, could have been beneficial in accelerating the learning process.

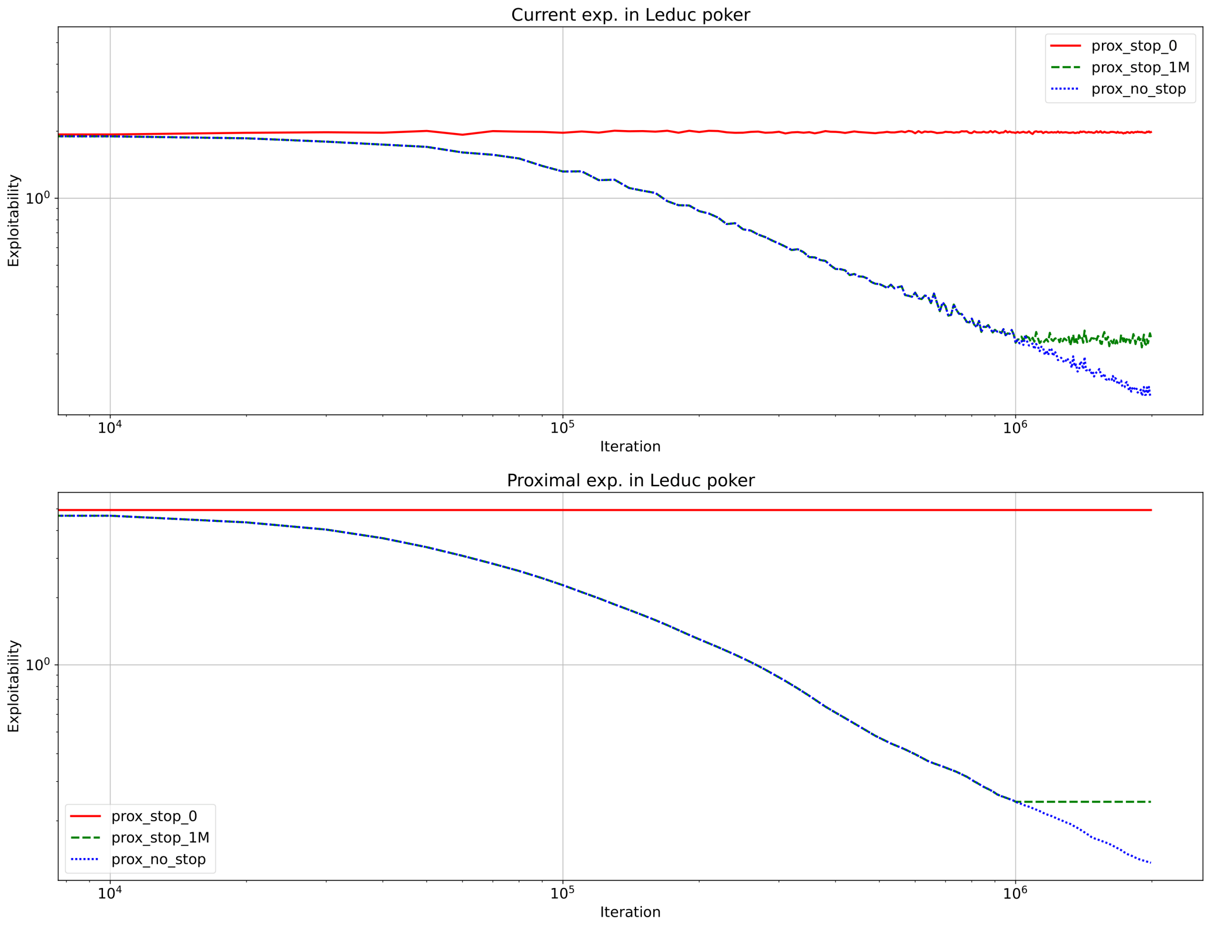

Figure 2: Stopping the update of the proximal parameters.

Exploitability In this experiment, we assess the performance of the Prox algorithm across various games, namely Kuhn poker, Leduc poker, and Liar's dice, as introduced earlier in this section. As illustrated in Figure 3, both the current and proximal profiles consistently achieve low exploitability scores across all games. In Figure 3, it's worth noting that 256 trajectories are collected for each iteration with a batchsize of 256.

The proximal profile demonstrates remarkable results in Leduc poker, achieving an exploitability of approximately 0.09. This places it in direct competition with the R-Nad algorithm, an exploitability of ≈ 0.2 is reported there. Similarly, the NFSP algorithm, as outlined by Heinrich and Silver [2016], manages at best an exploitability[2] of approximately 0.12. However, it's crucial to acknowledge that a comprehensive and rigorous comparison of these algorithms would necessitate further analysis.

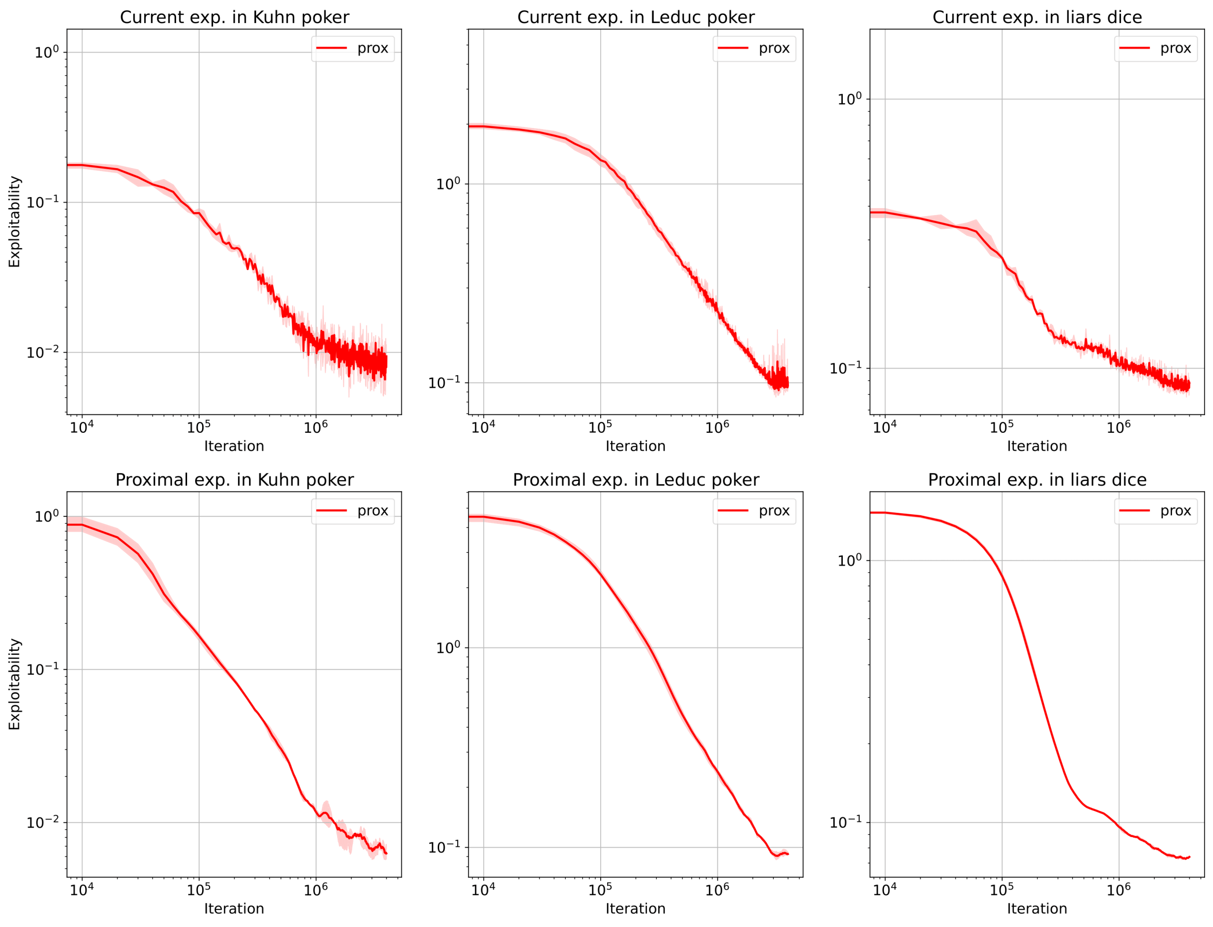

Figure 3: Exploitability as a function of the iteration number. Results are averaged over 3 seeds.

[2]They use an additional 1/2 factor in their exploitability definition compared to ours.

Proximal regularization coefficient In this experiment, we explore the impact of proximal regularization by testing the Prox algorithm with various proximal regularization coefficients $\beta_{\text{prox}} \in \{0, 0.02, 0.2, 2.0\}$ on Leduc poker game.

Figure 4 presents our findings. When β_{prox} is set to 0 (indicating the absence of proximal regularization), the Prox algorithm fails to converge. This absence of convergence underscores the pivotal role proximal regularization plays in stabilizing the algorithm. A stronger proximal regularization, such as $\beta_{\text{prox}} = 2.0$, slows down the convergence process. Intriguingly, for a modest β_{prox} value of 0.02, while the current profile struggles to converge, the proximal profile exhibits convergence, albeit with slightly more variance compared to the scenario with a larger β_{prox}.

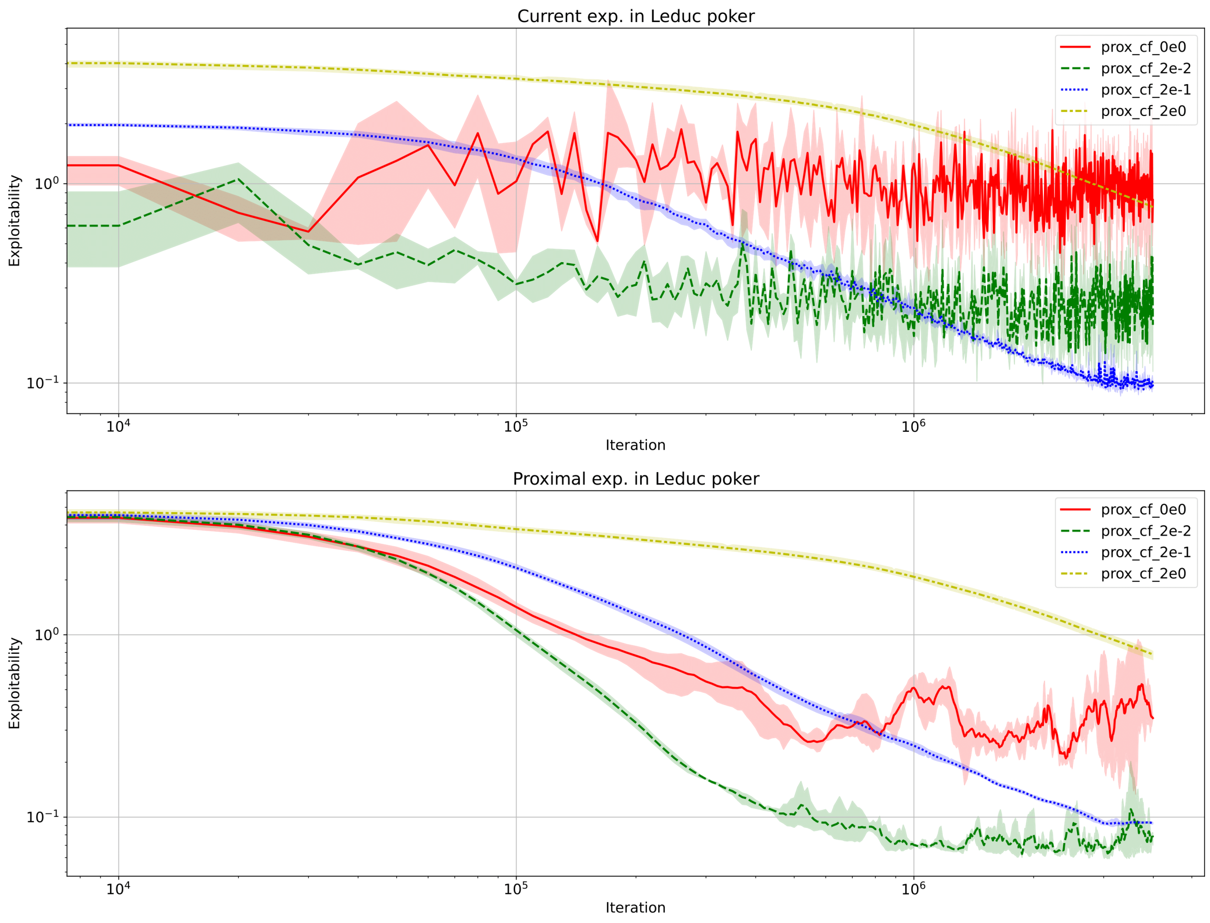

Figure 4: Varying proximal regularization coefficient. Results are averaged over 3 seeds.

Entropic regularization coefficient In this experiment, we explore the impact of entropic regularization by testing different parameters, specifically $\beta_{\text{ent}} \in \{0, 0.005, 0.5\}$. The results are provided in Figure 5. When the regularization parameter is set too high, both the current and proximal profiles tend to hover close to the uniform distribution, leading to a large exploitability gap. Intriguingly, when no entropic regularization is applied, the current and proximal profiles diverge significantly or display considerable variance. This divergence emphasizes the vital role of entropic regularization in stabilizing the `Prox` algorithm, by preventing the profiles from approaching the extremes of the probability simplex.

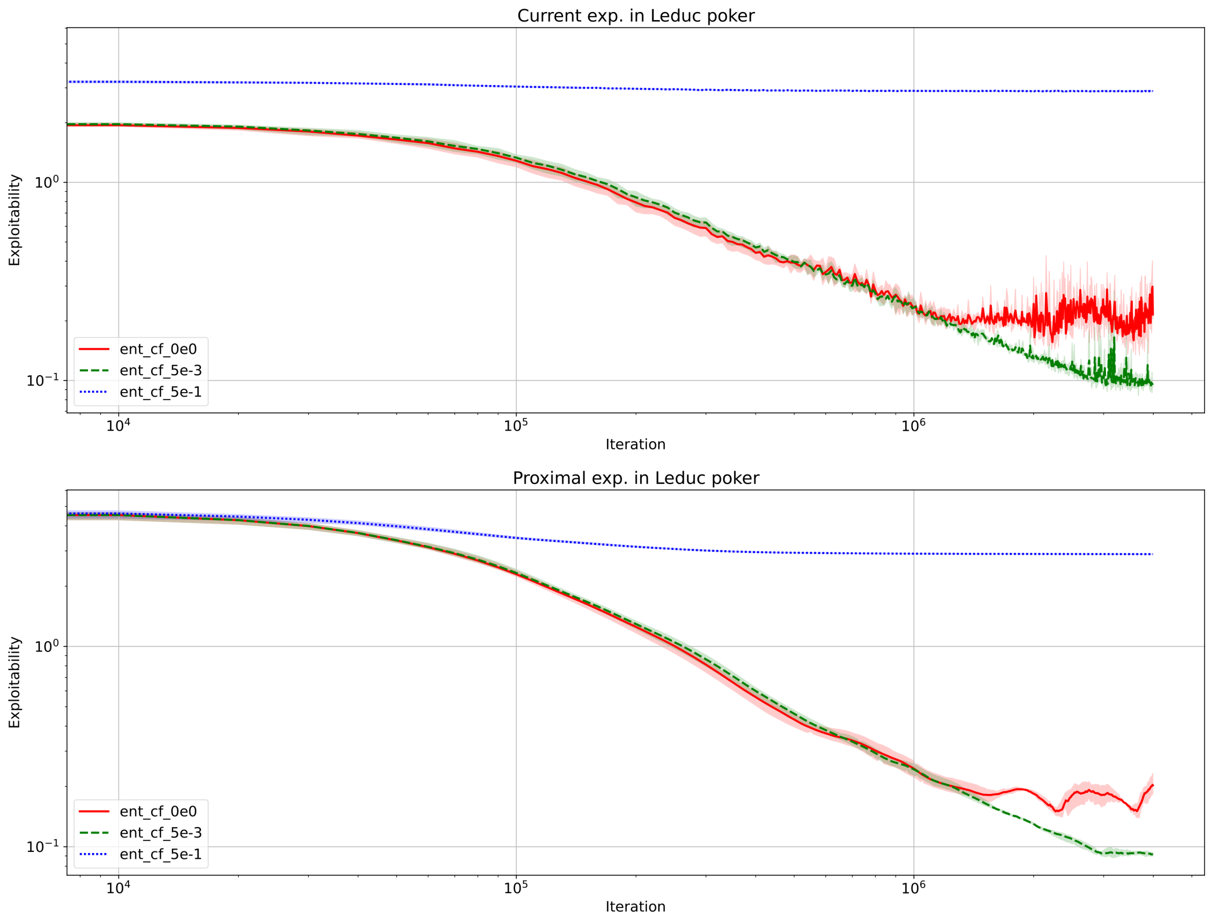

Figure 5: Varying entropic regularization coefficient. Results are averaged over 3 seeds.

GAE parameter In Figure 6, we present the exploitability gap of the `Prox` algorithm across different values of the Generalized Advantage Estimation (GAE) parameter, $\lambda \in \{0.0, , 0.75, , 0.95, , 1.0\}$ for the Leduc poker game. The selection of this parameter exhibits minimal influence on the learning process. This observation is rather expected given that in Leduc poker, the trajectories are relatively short.

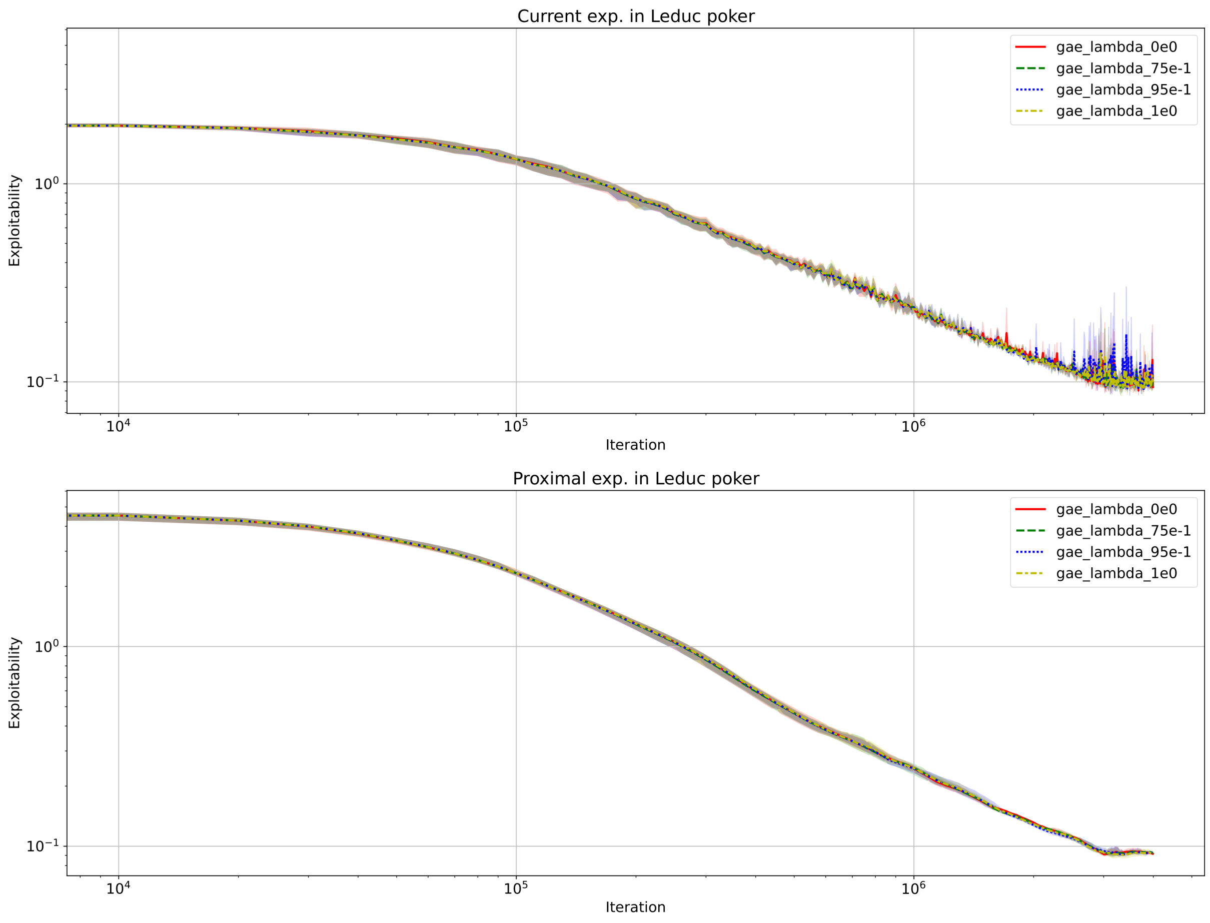

Figure 6: Varying GAE parameter. Results are averaged over 3 seeds.

Epsilon threshold In Figure 7, we present the exploitability gap of the `Prox` algorithm across different values of the gradient clipping regularization parameter, $\varepsilon \in 0.0, 0.01, 0.1$. When the regularization is completely absent ($\epsilon = 0$), the `Prox` algorithm fails to converge. Conversely, if the regularization is excessively large, the convergence of the algorithm slows down significantly. It's noteworthy that entropic regularization and gradient clipping, though both preventing the current profile to drift to far away from the uniform policy, serve as complementary mechanisms.

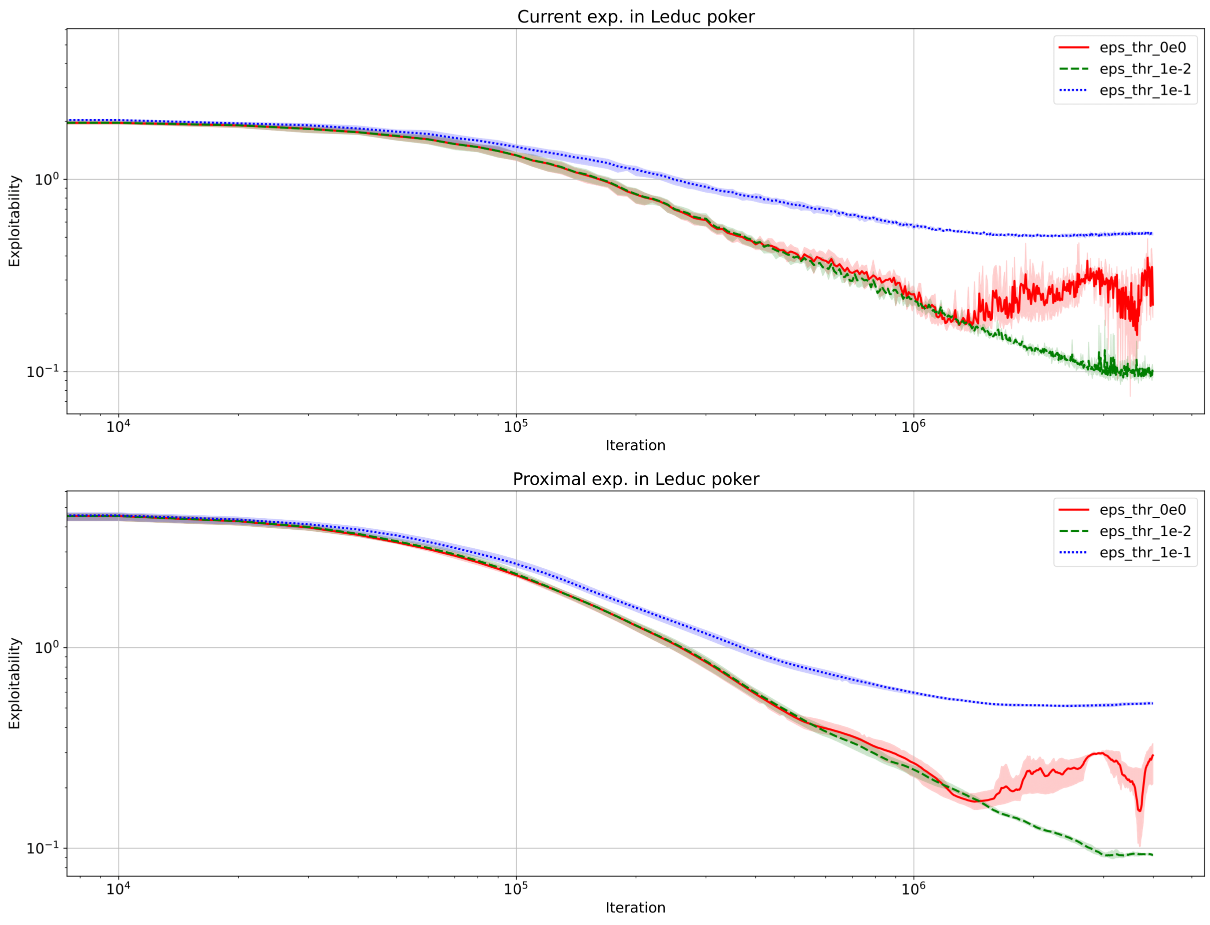

Figure 7: Varying epsilon threshold. Results are averaged over 3 seeds.

Proximal period In Figure 8, we depict the exploitability gap of the Prox algorithm across varying proximal periods, characterized as the characteristic time of the exponential moving average (prox period = $1/\eta_{\text{prox}}$) with values $1.0, 10000, 100000$. Interestingly, with a proximal period of one, the Prox algorithm fails to converge, a behavior consistent with observations from the previous section, especially in the context of proximal algorithms for matrix games. Remarkably, even with a relatively small proximal period, Prox manages to achieve a low exploitability gap, albeit with increased variance. Remark that if we push the training further to ≈ 10 million iterations the blue curve eventually passes below the green one.

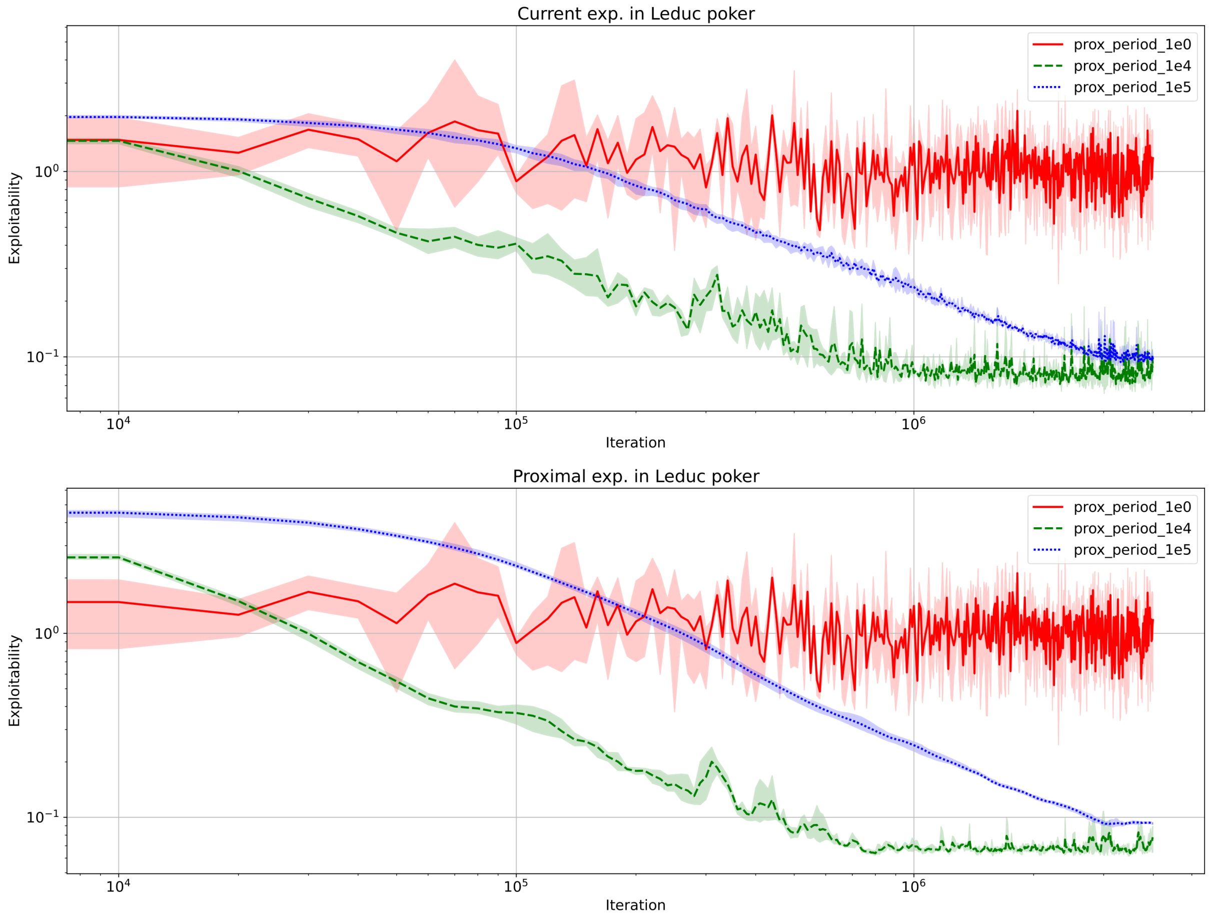

Figure 8: Varying the proximal period. Results are averaged over 3 seeds.

Batch size In Figure 9, we illustrate the exploitability gap of the `Prox` algorithm across different batch sizes, ranging from 32 to 256. Note that, even with small batch sizes, both the current and proximal profiles of `Prox` achieve convergence to a low exploitability gap.

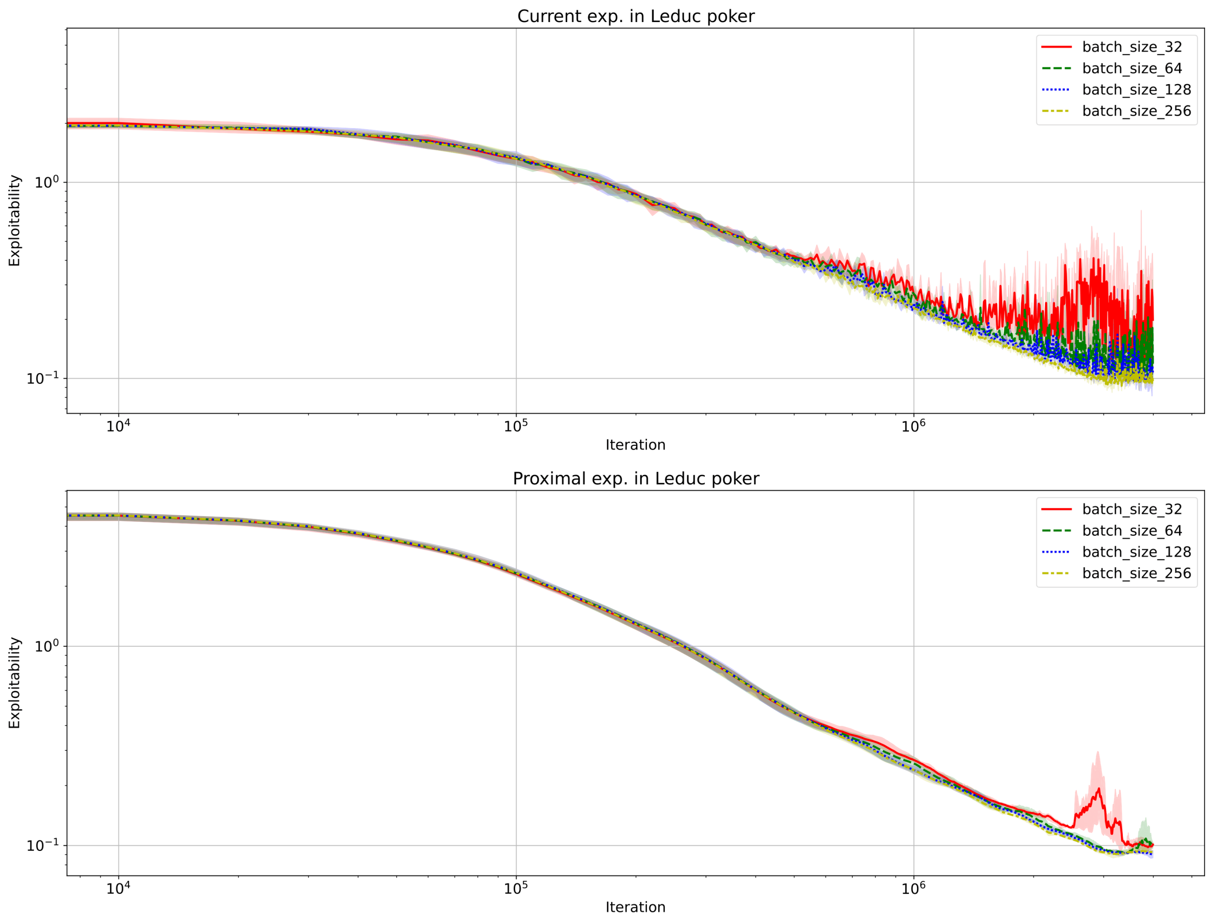

Figure 9: Varying the batch size. Results are averaged over 3 seeds.

5 Conclusion

We introduced Prox, a straightforward extension of the proximal algorithm tailored for extensive-form IIGs, which has demonstrated competitive performances across various standard benchmark games. Our work paves the way for several intriguing avenues of future research:

- *Scaling and alternative metrics*: An interesting prospect lies in assessing the scalability of Prox by testing it on larger games. Additionally, exploring alternative performance metrics beyond the computationally intensive exploitability gap in large IIGs would be necessary. Metrics such as the Elo rating system [Balduzzi et al., 2018] or the more sophisticated *alpha*-rank [Omidshafiei et al., 2019] could provide valuable candidates.
- *Improving updates*: The current Prox update assumes fixed policies for both max and min players after each step. However, one could update the profile in a backward fashion using as policy for the next steps the updated one akin to a tree-search with only one trajectory. We could also perform several steps of gradient update given one batch of trajectories in order to improve the sample complexity.
- *Comparison with algorithm for PIGs*: Conducting a comparative analysis between model-free policy gradient methods such as Prox, which entirely eschew tree-search, and more sophisticated model-based approaches like AlphaGo [Silver et al., 2016] or MuZero[Schrittwieser et al., 2019], which heavily rely on tree-search, in the context of perfect information games, would be interesting.

References

D. Balduzzi, K. Tuyls, J. Perolat, and T. Graepel. Re-evaluating evaluation. In *Advances in Neural Information Processing Systems*, pages 3268–3279, 2018.

G. W. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 1951.

N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 793–802. PMLR, 2019. http://proceedings.mlr.press/v97/brown19b/brown19b.pdf.

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1407–1416. PMLR, 2018.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. 2016.

Harold W Kuhn. Extensive games. *Proceedings of the National Academy of Sciences*, 36(10):570–576, 1950.

B. Martinet. Brève communication: régularisation d'inéquations variationnelles par approximations successives. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 4(R3):154–158, 1970.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

A. Nemirovski. Prox-method with rate of convergence o(1/t) for variational inequalities with lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

S. Omidshafiei, C. Papadimitriou, G. Piliouras, K. Tuyls, M. Rowland, J. B. Lespiau, W. M. Czarnecki, M. Lanctot, J. Perolat, and R. Munos. alpha-rank: Multi-agent evaluation by evolution. *Scientific Reports*, 9(1):9937, 2019.

J. Perolat, B. De Vylder, D. Hennes, E. Tarassov, F. Strub, V. de Boer, and K. Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022. URL https://www.science.org/doi/abs/10.1126/science.add4679.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model, 2019. URL http://arxiv.org/abs/1911.08265. cite arxiv:1911.08265.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, May 2-4 2016. http://arxiv.org/abs/1506.02438.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. ISSN 0028-0836. doi: 10.1038/nature16961.

Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertaintyin Artificial Intelligence (UAI)*, pages 550–558, 2005.

E. Steinberger, A. Lerer, and N. Brown. Dream: Deep regret minimization with advantage baselines and model-free learning. *CoRR*, 2020. https://arxiv.org/abs/2006.10410.