MODELING STRONG AND HUMAN-LIKE GAMEPLAY WITH KL-REGULARIZED SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the task of building strong but human-like policies in multi-agent decision-making problems, given examples of human behavior. Imitation learning is effective at predicting human actions but may not match the strength of expert humans, while self-play learning and search techniques (e.g. AlphaZero) lead to strong performance but may produce policies that are difficult for humans to understand and coordinate with. We show in chess and Go that regularizing search based on the KL divergence from an imitation-learned policy results in higher human prediction accuracy and stronger performance than imitation learning alone. We then introduce a novel regret minimization algorithm that is regularized based on the KL divergence from an imitation-learned policy, and show that using this algorithm for search in no-press Diplomacy yields a policy that matches the human prediction accuracy of imitation learning while being substantially stronger.

1 INTRODUCTION

Self-play AI algorithms have matched or exceeded expert human performance in many games, such as chess (Campbell et al., 2002; Silver et al., 2018), Go (Silver et al., 2016; 2017), and poker (Moravčík et al., 2017; Brown & Sandholm, 2017; 2019). However, the resulting policies often differ markedly from how humans play (McIlroy-Young et al., 2020a). This is a serious problem for human-computer interactions that involve cooperation rather than purely competition. In such settings, modeling the other participants accurately is important for success. For example, it is important for a self-driving car at a four-way stop sign to conform to existing human conventions rather than its own self-play solution to the problem (Lerer & Peysakhovich, 2019). Even in purely adversarial games, the alien nature of AI policies makes it difficult for humans to understand and learn from superhuman bots.

The classic approach toward modeling human behavior is imitation learning (IL) on human data. Yet, evidence in multiple games indicates that IL on expert human data produces policies that are much weaker than actual expert human play in domains with complex strategic planning. In this paper, we study the problem of producing policies that are both *strong* and *human-like* in games with complex strategic planning like chess, Go, Hanabi, and Diplomacy. In all four, we find that conducting search with KL-regularization towards an IL policy matches or exceeds the prior state of the art for prediction accuracy of expert humans while also better matching expert human performance.

In Section 3, we show that Monte Carlo tree search (MCTS) with a human imitation-learned policy prior and value function surpasses prior state-of-the-art results for human prediction accuracy in chess and Go. As explained by Grill et al. (2020), standard MCTS with a policy prior can be viewed as a form of KL-regularized search, optimizing a value function subject to a KL-divergence term with that prior. Although MCTS has been extensively studied for developing strong agents, it has been explored much less in the context of developing human-like agents.

Section 4 builds on these findings and shows how to generalize them to a class of imperfect-information games (in which ordinary MCTS is unsound and cannot be applied) via a new algorithm for KL-regularized regret minimization. We show that existing regret minimization algorithms achieve low accuracy in predicting expert human actions in no-press Diplomacy. We then introduce the first regret minimization algorithm to incorporate a cost term proportional to the KL divergence between the search policy and a human-imitation learned **anchor policy**. We call this algorithm **policy-regularized hedge**, or **piKL-hedge**. We prove that piKL-hedge converges to an equilibrium in which all players' policies are optimal given the joint policies of the players and the cost of deviating

from the anchor policy. We then present results in no-press Diplomacy showing that piKL-hedge produces policies that predict human actions as accurately as imitation learning while also improving head-to-head performance in a population of prior agents.

Appendix M additionally shows that applying KL-regularization toward a human IL policy in the search algorithm SPARTA Lerer et al. (2020) produces similar or better human prediction accuracy while greatly improving performance in the benchmark domain of Hanabi Bard et al. (2020).

Our experiments demonstrate the benefits of KL-regularized search in all four of chess, Go, no-press Diplomacy, and Hanabi to producing agents that are simultaneously more human-like and closer in strength to actual human experts than purely imitation-learned models.

2 PRELIMINARIES

We study the problem of learning policies for multiplayer games. Here we briefly introduce the key ingredients of both classes of games we study; Section 3 and Section 4 give a more formal presentation tailored to individual game types and learning algorithms.

An (N-player) game is defined by a **state space** S, an **action space** A, a (deterministic) **transition function** $T : S \times A^N \to S$, and a collection of **reward functions** u_i. We model the behavior of each player in a game as a **policy** $\pi_i : S \to \Delta(A)$ (a distribution over actions given states). In every round of a game, each player observes a (possibly incomplete) view s_i^t of the current state. One or more players select actions $a_i^t \sim \pi_i(\cdot \mid s_i^t)$, then each player receives a reward $u_i^t(s^t, \mathbf{a}^t = a_1^t, \ldots, a_n^t)$, and the game transitions into a new state $s^{t+1} = T(s^t, \mathbf{a}^t)$. Each player i aims to maximizes its expected reward, and the optimal policies for doing so may depend on the policies $\pi_{-i} = \{\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_N\}$ of the other players.

The sequential decision-making problem described above is extremely general, and in this paper we focus on two special cases. In **perfect-information** games, players make moves sequentially (e.g., u^1 and s^2 depend only on a_1, u^2 and s^3 depend only on a^2, etc.). Many important games, including chess and Go, fall into this category. Next, we study a more general class of **imperfect-information, simultaneous-action** games that make no assumptions about the dependence of different u_i and T on $\mathbf{a}$; here we focus on games with only a single round, also called matrix games. Owing to the large differences between these two settings, the tools for computing strong policies are quite different. The remainder of this paper accordingly offers a deeper exploration of each class of games: perfect-information games in Section 3 and imperfect-information games in Section 4.

3 PERFECT-INFORMATION GAMES: POLICY REGULARIZATION IN MONTE CARLO TREE SEARCH

In this section, we focus on developing strong human-like agents for perfect-information games. Monte Carlo tree search (MCTS) has been highly successful for developing strong, but not necessarily human-like agents in this setting, and is a key component of general learning algorithms such as AlphaZero and MuZero capable of achieving superhuman performance in chess, Go, and similar games (Silver et al., 2018; Schrittwieser et al., 2020). By contrast, for developing human-like agents, the best prior human move prediction accuracies for chess and Go were all achieved with pure imitation learning on human data McIlroy-Young et al. (2020a); Cazenave (2017); Silver et al. (2017).

The state of the art for predicting human moves in chess is the Maia engine created by McIlroy-Young et al. (2020a) via pure imitation learning without any search. However, this approach appears to be of limited effectiveness for modeling sufficiently strong humans. Although the weakest Maia models at low temperatures appear to outperform the players they imitate (due to "averaging away" of the imitated players' individual idiosyncratic mistakes (Anderson et al., 2021)) each successive model on data from stronger players improves by much less than the players improve.[1] The strongest model, trained to predict human 1900-1999 rated players, even with low temperature appears to be clearly below a 1950-average level of performance in all but the minority of bullet-speed games (in which very little time is available for planning and players are forced to rely more heavily on intuition). Similarly, in Go, pure imitation-learning agents have not exceeded mid-expert level on various online servers despite being trained on top-expert and master-level games (Cazenave, 2017).

[1] See ratings data at https://lichess.org/@/maia1, https://lichess.org/@/maia5, https://lichess.org/@/maia9

In contrast, search-based reinforcement learning (RL) agents such as AlphaZero that do not use a human policy prior play at a superhuman level, but often in non-human ways that humans find difficult to understand even when given access to interactively query and inspect the agent's analysis (Silver et al., 2017; Egri-Nagy & Törmänen, 2020).

However, we show in both chess and Go that if the human-learned model is used in MCTS with appropriate parameters, MCTS outperforms those models' human prediction accuracy while simultaneously reducing the shortcomings in those models' strength.

3.1 Background

We consider sequential games where each player i alternatively chooses action a from a policy π_i where, $a \sim \pi_i(\cdot \mid s)$. Each action deterministically transitions the game into a new state $s' = T(s, a)$ and gives rewards $u_i(s, a)$. Notationally, we may elide the player i in some places when it is clear that i is the next player to move in the state being considered.

For this work, following Silver et al. (2016), we implement one of the most common modern forms of MCTS, which uses a value function V predicting the expected total future reward $V_i(s) = \mathbb{E}[\sum_t u_i(s_t, a_t) \mid \pi_1, \pi_2, s_0 = s]$ and a policy prior τ (typically both the outputs of a trained deep neural net) and attempts to produce an improved policy π.

Each turn, MCTS builds a game tree over multiple iterations rooted at the current state. Each iteration t, MCTS explores a single path down the tree by simulating at each successive state s with player i to move the action:

$$\arg\max_a \; Q(s, a) + c_{\text{puct}} \tau(s, a) \frac{\sqrt{\sum_b N(s, b)}}{N(s, a) + 1} \tag{1}$$

where $Q(s, a)$ is the estimated expected future reward for i from playing action a in state s, the visit count $N(s, a)$ is the number of times a has been explored from s, $\tau(s, a)$ is the prior policy probability, and c_{puct} is a tunable parameter trading off exploration versus exploitation.

Upon reaching a state s_t not yet seen, MCTS queries the value function $V_i(s_t)$ for each player i, and based on $V_i(s_t)$ and any intermediate rewards received, updates all $Q(s, a)$ estimates on the path traversed. The final agent policy is $\pi(s, a) = N(s, a) / \sum_b N(s, b)$ where s is the root state, or optionally we may also have $\pi(s, a) \sim N(s, a)^{1/T}$ where T is a temperature parameter. See also Appendix G for a fuller description of MCTS.

Grill et al. (2020) show that the agent policy π computed by this form of MCTS is an approximate solution to the optimization problem:

$$\arg\max_\pi \sum_a Q(s, a)\pi(s, a) + \lambda D_{\text{KL}}(\tau \,\|\, \pi) \tag{2}$$

where $\lambda \sim c_{\text{puct}}\sqrt{N}$ and N is the total number of iterations.

In other words, at every node of the tree recursively, MCTS implicitly optimizes its expected future reward subject to KL regularization of its policy towards the prior policy τ with strength controlled by λ. For any fixed computational budget N, we can therefore tune c_{puct} to vary the strength of that prior, with $c_{\text{puct}} = \infty$ approximating the prior policy before search, and $c_{\text{puct}} \to 0$ approaching a greedy argmax of the Q value estimates.

If our goal is a strong *human-like* agent rather than solely a strong agent, and the KL-regularizing policy is learned from human data, then that policy serves not just as a prior, but also as an **anchor policy** that regularizing towards is desirable in and of itself. With good choice of c_{puct}, MCTS can improve that policy while remaining close to human. Our experiments confirm that MCTS improves on the strength and human prediction accuracy of the best existing models in both chess and Go.

3.2 Experiments in Chess and Go

In chess and Go, we ran two main experiments each. First, in chess using the prior state-of-the-art Maia models from McIlroy-Young et al. (2020a) and in Go using a model trained on professional games from the GoGoD dataset, we demonstrate that MCTS with that model outperforms the raw model in human prediction accuracy. Second, we also sanity-check that MCTS with the same parameters greatly improves the strength of the same models in chess and Go.

Game	Model	Predicting	Raw Model	Model + MCTS				
				$c_{\text{puct}} = 10$	5	2	1	0.5
Chess	Maia1100	Rating 1100	51.1	51.2	51.3	50.8	49.5	47.4
Chess	Maia1500	Rating 1500	52.4	52.7	52.9	52.8	51.9	50.1
Chess	Maia1900	Rating 1900	53.2	53.6	54.0	54.3	53.8	52.4
Go	Cazenave	GoGoD	54.7					
Go	Wu (2018)	GoGoD	55.3					
Go	Wang et al.	GoGoD	57.7					
Go	Ours	GoGoD	57.8	58.1	58.3	58.5	58.1	57.1

Table 1: % top-1 test accuracy predicting human moves in chess and Go using MCTS with 50 playouts and various c_{puct}, or raw model without MCTS, equivalent to infinite c_{puct}. In chess, first 10 ply per game and moves with < 30s time left excluded, similar to McIlroy-Young et al. (2020a). Standard error is approx 0.1 or less on all values. MCTS on top of current state-of-the-art models improves human prediction accuracy significantly.

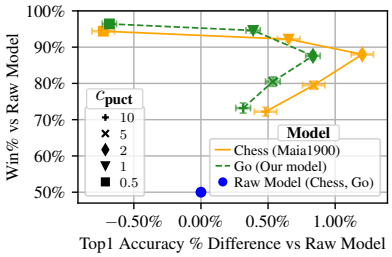

Figure 1: Improvement in top-1 accuracy of Maia1900 in Chess or our GoGoD model in Go using MCTS, plotted versus winrate of MCTS against the raw model (temperature 1). Error bars indicate 1 standard error. Many c_{puct} values increase both human prediction accuracy and winrate over the raw model in both Chess and Go.

3.2.1 Data and Model Architecture In chess, for the human-learned anchor policy we use the pre-trained Maia1100, Maia1500, and Maia1900 models from McIlroy-Young et al. (2020a), achieving state-of-the-art performance on rating-conditional human move prediction. These models follow a standard AlphaZero-like residual block architecture, including both a policy and a value head, and were trained to imitate players in ratings "buckets" 1100, 1500, and 1900 respectively, based on roughly 10 million games each from the popular Lichess server (each bucket contains games between players from rating N to N+99).

For Go, we trained a deep neural net on the GoGoD professional game dataset[2]. We match Cazenave (2017) in using games from 1900 through 2014 for training and 2015-2016 as the test set, with roughly 73000 and 6500 games, respectively. Our architecture matches the 20-block residual net structure used by some versions of AlphaZero (Silver et al., 2017), except adds squeeze-and-excitation layers which have been successful in image processing tasks (Hu et al., 2018) and self-play learning in chess and Go (LC0, 2020; Troisi, 2019). See Appendix F for additional details.

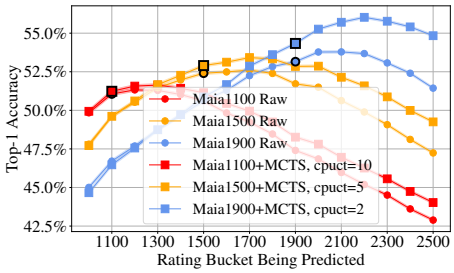

Figure 2: Top-1 % test accuracy in chess for Maia models trained to predict moves by players in rating buckets 1100, 1500, 1900, applied to predict *all* rating buckets, with and without MCTS. The black bolded outline indicates where the model is predicting the same rating as it was trained on. Standard error is very small, the slight thin shade around each line. MCTS most improves prediction of a model on players of its target rating and higher.

3.2.2 Improved Human Prediction and Strength In Table 1 we show the top-1 accuracy of MCTS at predicting players in chess and Go. MCTS on top of each model tested outperforms that model at predicting human moves.

In chess, the benefit provided increases greatly as the rating of players predicted increases, while the optimal choice for c_{puct} appears to decrease (allowing increasingly small value differences to affect the search). This is consistent with the intuitive hypothesis that stronger players plan more deeply, increasing the value of explicitly modeling planning, and that they are more sensitive to small future value differences. In Go, despite our baseline model being equal or better than all prior imitation-learning models on the GoGoD human pro games dataset, MCTS improves it yet further.

In Figure 2, for chess we see that while KL-regularized search improves each model's accuracy on players of its target rating, surprisingly, the improvement grows yet larger when each model predicts

[2]https://gogodonline.co.uk/

players of *higher* rating than it was trained on. This suggests that as human players improve, the incremental average change in their behavior resembles or is correlated with the way that highly-regularized search improves the strength of a baseline policy.

Additionally, in Appendix D, we show that if we apply post-processing to the MCTS policy based on Grill et al. (2020), MCTS improves cross entropy with human moves in both chess and Go, not just top-1 accuracy. In other words, not only does policy-regularized search improve the prediction of the top move, but it also better models the overall *distribution* of moves that humans may likely play.

We measure the strength impact of regularized search with 1000 games[3] per c_{puct} setting between the raw model policy and the MCTS policy, sampling each at temperature 1. Figure 1 shows the change in human prediction accuracy of MCTS in both chess and Go plotted jointly versus winrate of MCTS against the raw model. Rather than solely a tradeoff between strength and accuracy, most c_{puct} values in the range we tested increase *both*, some achieving more than 90% winrate while still improving human prediction. See Appendix E for results at lower temperature and evidence that accuracy improves further at longer time controls.

Although we did not test against humans directly to calibrate, this gives clear evidence that a well-tuned human-regularized MCTS agent would be better able to match the 1900-1999-rated chess players that Maia1900 currently falls hundreds of Elo short of imitating, while simultaneously being more accurate to their move-by-move behavior, and similarly for human-imitation agents in Go.

4 Imperfect-Information Games: Policy-regularized Regret Minimization

While MCTS is a popular search algorithm for perfect-information deterministic games, it is not able to compute optimal policies in imperfect-information games. Instead, iterative algorithms based on regret minimization are the leading approach to search in imperfect-information games.

Hedge (Littlestone & Warmuth, 1994; Freund & Schapire, 1997) is an iterative regret minimization algorithm that in general converges to a coarse correlated equilibrium (CCE) (Hannan, 1957). In the special case of two-player zero-sum games, it also converges to a Nash equilibrium (NE) (Nash, 1951).

Regret Matching (RM) (Blackwell et al., 1956; Hart & Mas-Colell, 2000) is another equilibrium-finding algorithm similar to Hedge that has historically been more popular and that we compare our algorithm to in this paper.

The effectiveness of the implicit KL-regularization in MCTS that we study in Section 3 motivates us to develop an equilibrium-finding algorithm called piKL-Hedge that similarly biases the search towards an anchor policy. In Section 4.3, we show that piKL-Hedge achieves better empirical performance against baseline human-imitation models than Hedge and RM in a large imperfect-information game, as well as much higher human prediction accuracy.

4.1 Background

We consider a game with $\mathcal{N}$ players where each player i chooses an action a from a set of actions $\mathcal{A}_i$. We denote the actions of all players other than i as $\boldsymbol{a}_{-i}$. After all players simultaneously choose an action, player i receives a reward of $u_i(a, \boldsymbol{a}_{-i})$. Players may also choose a probability distribution over actions, where the probability of action a is denoted $\boldsymbol{\pi}_i(a)$ and the vector of probabilities is denoted $\boldsymbol{\pi}_i$. We also define the fixed policy that we are interested in biasing player i towards as the anchor policy $\boldsymbol{\tau}_i \in \Delta(A_i)$.

Each player i maintains a **regret** for each action. The regret on iteration t is denoted $R_i^t(a)$. Initially, all regrets are zero. On each iteration t of Hedge, $\boldsymbol{\pi}_i^t(a)$ is set according to

$$\boldsymbol{\pi}_i^t(a) \propto \exp\left(R_i^t(a)\right) \tag{3}$$

[3]In Go, we also use the open-source KataGo (Wu, 2020) to determine when the game is over and to score the result. Unlike RL agents, humans which our models imitate universally pass and score the game well before it becomes mechanically scorable, so we use KataGo as a neutral judge.

Next, each player samples an action a^* from $\mathcal{A}_i$ according to π_i^t and all regrets are updated such that

$$R_i^{t+1}(a) = R_i^t(a) + u_i(a, \boldsymbol{a}_{-i}^*) - \sum_{a' \in \mathcal{A}_i} \boldsymbol{\pi}_i^t(a') u_i(a'_i, \boldsymbol{a}_{-i}^*) \tag{4}$$

It is proven that the *average* policy of Hedge over all iterations converges to a NE in two-player zero-sum games and, more broadly, the players' joint policy distribution converges to a CCE as $t \to \infty$.

We wish to model agents that seek to maximize their expected reward, while at the same time playing "close" to the anchor policy. The two goals can be reconciled by defining a composite utility function that adds a penalty based on the "distance" between the player policy and their anchor policy, with coefficient $\lambda_i \in [0, \infty)$ scaling the penalty.

For each player i, we define i's utility as a function of the the agent policy $\boldsymbol{\pi}_i \in \Delta(A_i)$ given policies $\boldsymbol{\pi}_{-i}$ of all other agents:

$$\mathcal{U}_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}) \coloneqq u_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i}) - \lambda_i \, D_{\mathrm{KL}}(\boldsymbol{\pi}_i \,\|\, \boldsymbol{\tau}_i). \tag{5}$$

When λ is large, the utility function is dominated by the KL-divergence term $\lambda_i \, D_{\mathrm{KL}}(\boldsymbol{\pi}_i \,\|\, \boldsymbol{\tau}_i)$, and so the agent will naturally tend to play a policy $\boldsymbol{\pi}_i$ close to the anchor policy $\boldsymbol{\tau}_i$ [4]. When λ_i is small, the dominating term is the rewards $u_i(\boldsymbol{\pi}_i, \boldsymbol{a}_{-i}^t)$ and so the agent will tend to maximize reward without as closely matching the anchor policy $\boldsymbol{\tau}_i$. These statements are made precise in Theorem 1 and Theorem 2.

4.2 No-Regret Learning for Policy-Regularized Utilities

In this section, we present Algorithm 1, a no-regret algorithm based on Hedge for any player i to learn strong policies relative to the regularized utilities defined in equation 5. As we show in Proposition 1 in Appendix B, it guarantees that each player i accumulates sublinear regret (of order $\log T$) with respect to the regularized utility functions:

$$\mathcal{U}_i^t(\boldsymbol{\pi}_i) \coloneqq \mathcal{U}_i(\boldsymbol{\pi}_i, \boldsymbol{a}_{-i}^t) = u_i(\boldsymbol{\pi}_i, \boldsymbol{a}_{-i}^t) - \lambda_i \, D_{\mathrm{KL}}(\boldsymbol{\pi}_i \,\|\, \boldsymbol{\tau}_i),$$

no matter the opponents' actions $\boldsymbol{a}_{-i}^t$ at each time t.

As with many other regret-minimization methods, we consider the *average* policy of each player i over T iterations:

$$\bar{\boldsymbol{\pi}}_i^T \coloneqq \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{\pi}_i^t \tag{7}$$

where $\boldsymbol{\pi}_i^t$ is defined in equation 6. We take $\bar{\boldsymbol{\pi}}_i^T$ to be the final agent policy produced by piKL-HedgeBot in no-press Diplomacy (as described in more detail in Section 4.3). As shown in Appendix B, the KL-divergence of $\bar{\boldsymbol{\pi}}_i^T$ from the anchor policy $\boldsymbol{\tau}_i$ converges to be inversely proportional to λ_i:

Theorem 1. *(piKL stays close to the anchor policy) Upon running Algorithm 1 for T iterations in a multiplayer general-sum game, the policy $\bar{\boldsymbol{\pi}}_i^T$ is at a distance $D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \leq \frac{1}{\lambda_i}\left(\frac{R_i^T}{T} + D_i\right)$, where D_i is any upper bound on possible rewards for Player i. In particular, if $\eta > 0$ is set so that $R_i^T = o(T)$, then $D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \to D_i/\lambda_i$ as $T \to +\infty$.*

Algorithm 1: piKL-Hedge (for Player i)

Data:
- A_i set of actions for Player i;
- u_i reward function for Player i;
- $\eta > 0$ learning rate hyperparameter.

function Initialize()
- $t \leftarrow 0$
- **for each** action $a \in A_i$ **do**
  - $\mathsf{CV}_i^0(a) \leftarrow 0$

function Play()
- $t \leftarrow t + 1$
- let $\boldsymbol{\pi}_i^t$ be the policy such that

$$\boldsymbol{\pi}_i^t(a) \propto \exp\left\{\frac{\eta \, \mathsf{CV}_i^{t-1}(a) + t\lambda_i\eta \log \tau_i(a)}{1 + t\lambda_i\eta}\right\}. \tag{6}$$

- sample an action $a^t \sim \boldsymbol{\pi}_i^t$
- play a^t and observe actions $\boldsymbol{a}_{-i}^t$ played by the opponents
- **for each** $a \in A_i$ **do**
  - $\mathsf{CV}_i^t(a) \leftarrow \mathsf{CV}_i^{t-1}(a) + u_i(a, \boldsymbol{a}_{-i}^t)$

We can also show (Appendix B) that in the case of a *two-player zero-sum* game, $\bar{\boldsymbol{\pi}}_i^T$ approximates a Nash equilibrium of the original utility functions, with the approximation guarantee controlled by λ:

[4] The careful reader may observe that the direction of the KL-divergence term, $D_{\mathrm{KL}}(\pi \,\|\, \tau)$ is the opposite of the direction implicit in MCTS, $D_{\mathrm{KL}}(\tau \,\|\, \pi)$. We choose this direction for greater ease of theoretical analysis and implementation in the context of regret minimization; for our use cases we have not found the exact form of the loss to be critical so much as simply doing any reasonable regularized search.

Theorem 2. *Let $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ be any limit point of the average policies $(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)$ of the players. Almost surely, $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ is a $(\max_{i=1,2}\{\lambda_i \beta_i\})$-approximate Nash equilibrium policy with respect to the original utility functions u_i, where β_i is as defined in equation 21.*

Lastly, we remark that in the special case that $\boldsymbol{\tau}_i$ is the uniform policy for all players i, the above results imply that Algorithm 1 converges towards a **quantal response equilibrium** (McKelvey & Palfrey, 1995a), in which an imperfect agent is modeled as choosing actions with probability exponentially decaying in the amount that each action is worse than the best action(s), given that all other agents behave the same way. Our method can be seen as a generalization that takes into account a human-learned prior for what actions may be more likely.

4.3 Diplomacy Experiments

Diplomacy is a benchmark 7-player simultaneous-action game featuring both cooperation and competition. In Appendix H, we summarize the rules of the game. Using piKL-Hedge, we develop an agent piKL-HedgeBot and show that it improves upon prior approaches.In Appendix C, we also illustrate the key features of piKL-Hedge in Blotto, a famous 2-player simultaneous action game.

4.3.1 Algorithms and Models In no-press Diplomacy, we compare the different equilibrium search algorithms (RM, Hedge and piKL-Hedge) using the procedure introduced by Gray et al. (2020). We perform 1-ply lookahead where on each turn, we sample up to 30 of the most likely actions for each player from a policy network trained via imitation learning on human data (IL policy). We then consider the 1-ply subgame consisting of those possible actions where the rewards for a given joint action are given by querying a value network trained on human game data as in Gray et al. (2020). We play according to the approximate equilibrium computed for that subgame by that algorithm. For piKL-Hedge, the anchor policy is simply the same human-trained policy network. Our baseline policy and value models also contain a few improvements over prior models for no-press Diplomacy, described in Appendices J and K. In our experiments, we label our RM, Hedge, and piKL-Hedge agents as RMBot, HedgeBot, and piKL-HedgeBot respectively. We compare also against SearchBot Gray et al. (2020) (similar to RMBot but using the models from Gray et al. (2020) rather than our models). See Appendix I for more details about the hyperparameters used.

4.3.2 Strong, human-like play with piKL-Hedge Similar to Chess and Go, we compare the human prediction accuracy of RMBot, HedgeBot, piKL-HedgeBot (with different λs) to the IL anchor policy, as well as testing their head-to-head strength. In particular, we test their ability to predict human moves in 226 no-press Diplomacy games from a validation set, and measure their score against the IL policy across 700 games each.

In Figure 3 (Left), we present the average top-1 accuracy of unit orders in each action predicted by these methods as well as their average scores against 6 IL anchor policies. The raw IL model ($\lambda = \infty$) predicts human moves with high accuracy but is weak and achieves low average score. Unregularized Hedge and RM ($\lambda = 0$) achieve high score but low human prediction accuracy. By contrast, piKL-HedgeBot with different λ achieves a variety of highly favorable combinations of the two. $\lambda = 10^{-1}$ gives about the same top-1 accuracy as the IL policy but improves score by a factor of 1.4x over the IL anchor policy. $\lambda = 10^{-3}$ outperforms unregularized search methods in both score (by ~5%) and human prediction accuracy (by ~6%). Mild regularization improves average score, rather than harming it. We also tested pure RL agents and found they perform poorly in predicting human moves. In particular, the recently proposed DORA and HumanDNVI-NPU algorithms Bakhtin et al. (2021) achieve top-1 accuracy of only 29.1% and 37.8% respectively.

Next, in Figure 3 (Right), we compare the top-1 accuracy of these methods across players of different pseudo-Elo ratings. The pseudo-Elo (e_i) for player i is constructed based on the logit rating s_i introduced in Gray et al. (2020), where, $e_i = \frac{s_i \cdot 400}{\log(10)} + 1000$. The top-1 accuracy for all the search-based policies increases with pseudo-Elo, indicating that they are better at modeling stronger players than weaker players. piKL-Hedge ($\lambda = 10^{-1}$) performs just as well as the anchor policy across pseudo-Elos while being significantly stronger than the anchor, while $\lambda = 10^{-3}$ is as strong or stronger than Hedge and RM but matches human play far better.

4.3.3 piKL-HedgeBot performs well against a varied pool of agents We also develop a new head-to-head evaluation setting, where rather than testing one agent vs 6x of another agent, all 7 agents per game are sampled uniformly from a pool. The 1v6 head-to-head scores used in prior work Gray et al. (2020); Bakhtin et al. (2021); Anthony et al. (2020); Paquette et al. (2019) indicate whether

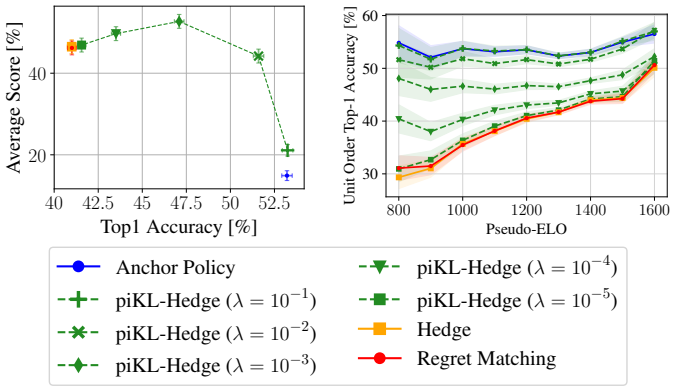

Figure 3: (Left) Average top-1 accuracy of unit orders in each action predicted by the human IL anchor policy, RM, Hedge and piKL-Hedge, versus head to head score against 6 IL anchor policies. piKL-HedgeBot ($\lambda = 10^{-1}$) predicts human moves as accurately as the anchor policy while achieving a much higher score. At the same time, piKL-HedgeBot ($\lambda = 10^{-3}$) allows for a stronger and more human-like policy than unregularized search methods (hedge, RM). Note that equal performance would imply an average score of 1/7 $\approx 14.3\%$. Error bars indicate 1 standard error. (Right) Average top-1 accuracy of unit orders in each action predicted by the human policy, RM, and piKL, as a function of pseudo-Elo player rating.

Agent	Average Score
DipNet†	3.7% ± 0.3%
DipNet RL†	4.7% ± 0.3%
Blueprint‡	4.9% ± 0.3%
BRBot‡	16.1% ± 0.6%
SearchBot‡	13.4% ± 0.5%
IL Policy	7.9% ± 0.4%
RMBot	31.3% ± 0.7%
piKL-HedgeBot ($\lambda = 10^{-3}$)	32.9% ± 0.7%

Table 2: Average score achieved by agents in a uniformly sampled pool of other agents. piKL-HedgeBot ($\lambda = 10^{-3}$) outperforms all other agents in this setting. DipNet agents from (Paquette et al., 2019) use a temperature of 0.1, while IL Policy and blueprint (Gray et al., 2020) use a temperature of 0.5. The ± shows one standard error. †(Paquette et al., 2019); ‡(Gray et al., 2020).

a population of 6x agents can be invaded by a 1x agent, and hence whether the 6 agents constitute an Evolutionarily Stable Strategy (ESS) (Taylor & Jonker, 1978; Smith, 1982). By contrast, assigning the 7 agents per game randomly from a pool studies the robustness of an agent to a variety of other agents.We experiment with a pool of 8 agents. Five are previously published agents: DipNet, DipNet RL Paquette et al. (2019), Blueprint, BRBot, SearchBot Gray et al. (2020) and three are our agents: IL policy, RMBot and piKL-HedgeBot. Doing well in this population requires playing well with both human-like policies (DipNet, Blueprint) and equilibrium policies (SearchBot, RMBot). Each experiment only compares one lambda value of piKL-HedgeBot for fairness. The results of these experiment with piKL-HedgeBot ($\lambda = 10^{-3}$) is presented in Table 2. piKL-HedgeBot ($\lambda = 10^{-3}$) outperforms all other agents. Overall, unlike Chess and Go, we do not find that piKL-Hedge clearly improves human prediction accuracy over the IL model. However, unlike Chess and Go, we observe that piKL-Hedge does improve playing strength over prior search methods against various past agents. In general-sum games like Diplomacy, it appears that piKL-hedge with a human anchor policy allows for slightly better play in a population containing human-like agents while still doing well against equilibrium-searchers, or alternatively can improve strength over IL to a lesser degree with no cost at all to prediction accuracy.

5 Conclusion

In this paper, we showed across several domains that regularizing search policies according to a KL-divergence loss with an imitation learned (IL) policy produces policies that maintain high human prediction accuracy while being far stronger than the original learned policy. In chess and Go, applying standard MCTS regularized toward a human-learned policy achieves state-of-the-art prediction accuracy, surpassing imitation learning, while also winning more than 85% of games against an IL model. We then introduced a novel regret minimization algorithm that is regularized based on the KL divergence from an IL policy. In no-press Diplomacy, this algorithm yields both a policy that predicts human play with the same accuracy as imitation learning alone while increasing win rate against state-of-the-art baselines by a factor of 1.4, or alternately a policy that outperforms unregularized search while achieving much higher human prediction accuracy. We presented in Appendix M similar successful results for KL-regularized search in Hanabi.

There are several directions for future work, such as extending piKL-Hedge to handle extensive-form games. Additionally, there may be better ways to regularize search than KL-divergence. Finally, it remains to be seen how KL-regularized search performs when combined with RL.

REFERENCES

Jacob Abernethy and Alexander Rakhlin. Beating the adaptive bandit with high probability. Technical Report UCB/EECS-2009-10, EECS Department, University of California, Berkeley, Jan 2009. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-10.html.

Ashton Anderson, Reid McIlroy-Young, Siddhartha Sen, and Jon Kleinberg. Introducing maia, a human-like neural network chess engine, 2021. URL https://lichess.org/blog/X9PUixUAANCqFRSh/introducing-maia-a-human-like-neural-network-chess-engine.

Thomas Anthony, Tom Eccles, Andrea Tacchetti, János Kramár, Ian Gemp, Thomas Hudson, Nicolas Porcel, Marc Lanctot, Julien Perolat, Richard Everett, Satinder Singh, Thore Graepel, and Yoram Bachrach. Learning to play no-press diplomacy with best response policy iteration. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17987–18003. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d1419302db9c022ab1d48681b13d5f8b-Paper.pdf.

Hendrik Baier, Adam Sattaur, Edward Powley, Sam Devlin, Jeff Rollason, and Peter Cowling. Emulating human play in a leading mobile card game. *IEEE Transactions on Games*, PP:1–1, 05 2018. doi: 10.1109/TG.2018.2835764.

Anton Bakhtin, David Wu, Adam Lerer, and Noam Brown. No-press diplomacy from scratch. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.

David Blackwell et al. An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8, 1956.

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189. JMLR Workshop and Conference Proceedings, 2011.

Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, pp. eaao1733, 2017.

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

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep Blue. *Artificial intelligence*, 134 (1-2):57–83, 2002.

Tristan Cazenave. Residual networks for computer go. *IEEE Transactions on Computational Intelligence and AI in Games*, PP:1–1, 03 2017. doi: 10.1109/TCIAIG.2017.2681042.

Shicong Cen, Yuting Wei, and Yuejie Chi. Fast policy extragradient methods for competitive games with entropy regularization. In *Neural Information Processing Systems (NeurIPS)*, 2021.

Brandon Cui, Hengyuan Hu, Luis Pineda, and Jakob Foerster. K-level resoning for zero-shot coordination in hanabi. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Attila Egri-Nagy and Antti Törmänen. The game is not over yet—go in the post-alphago era. *Philosophies*, 5(4):37–0, 2020. ISSN 2409-9287. doi: 10.3390/philosophies5040037. URL https://www.mdpi.com/2409-9287/5/4/37.

Gabriele Farina, Christian Kroer, and Tuomas Sandholm. Online convex optimization for sequential decision processes and extensive-form games. In *AAAI Conference on Artificial Intelligence*, 2019.

Arnaud Fickinger, Hengyuan Hu, Brandon Amos, Stuart Russell, and Noam Brown. Scalable online planning via reinforcement learning fine-tuning. *CoRR*, abs/2109.15316, 2021. URL https://arxiv.org/abs/2109.15316.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jonathan Gray, Adam Lerer, Anton Bakhtin, and Noam Brown. Human-level performance in no-press diplomacy via equilibrium search. In *International Conference on Learning Representations*, 2020.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-carlo tree search as regularized policy optimization. In *International Conference on Machine Learning*, pp. 3769–3778. PMLR, 2020.

James Hannan. Approximation to bayes risk in repeated play. *Contributions to the Theory of Games*, 3:97–139, 1957.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pp. 4399–4410. PMLR, 2020.

Hengyuan Hu, Adam Lerer, Noam Brown, and Jakob N. Foerster. Learned belief search: Efficiently improving policies in partially observable settings. *CoRR*, abs/2106.09086, 2021a. URL https://arxiv.org/abs/2106.09086.

Hengyuan Hu, Adam Lerer, Brandon Cui, Luis Pineda, David Wu, Noam Brown, and Jakob Foerster. Off-belief learning. In *International Conference on Machine Learning*. PMLR, 2021b.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018. doi: 10.1109/CVPR.2018.00745.

Matthew Lai. Forum post on alphazero news (post by user matthewlai). http://talkchess.com/forum3/viewtopic.php?f=2&t=69175&sid=06ca6a966c29743d765c11b13402be8d&start=70#p781765, 2018.

LC0. Leela chess zero information page on "neural network topology". https://lczero.org/dev/backend/nn/, 2020.

Adam Lerer and Alexander Peysakhovich. Learning existing social conventions via observationally augmented self-play. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 107–114. ACM, 2019.

Adam Lerer, Hengyuan Hu, Jakob Foerster, and Noam Brown. Improving policies via search in cooperative partially observable games. In *AAAI Conference on Artificial Intelligence*, 2020.

Chun Kai Ling, Fei Fang, and J Zico Kolter. What game are we playing? end-to-end learning in normal and extensive form games. *International Joint Conferences on Artificial Intelligence Organization*, 2018.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1677–1687, 2020a.

Reid McIlroy-Young, Russell Wang, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Learning personalized models of human behavior in chess. Technical report, Microsoft, Inc., August 2020b. URL https://www.microsoft.com/en-us/research/publication/learning-personalized-models-of-human-behavior-in-chess/.

Richard D. McKelvey and Thomas R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38, 1995a. ISSN 0899-8256. doi: https://doi.org/10.1006/game.1995.1023. URL https://www.sciencedirect.com/science/article/pii/S0899825685710238.

Richard D McKelvey and Thomas R Palfrey. Quantal response equilibria for normal form games. *Games and economic behavior*, 10(1):6–38, 1995b.

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

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

John Nash. Non-cooperative games. *Annals of mathematics*, pp. 286–295, 1951.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Philip Paquette, Yuchen Lu, Seton Steven Bocco, Max Smith, O-G Satya, Jonathan K Kummerfeld, Joelle Pineau, Satinder Singh, and Aaron C Courville. No-press diplomacy: Modeling multi-agent gameplay. In *Advances in Neural Information Processing Systems*, pp. 4474–4485, 2019.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Alexander Rakhlin. Lecture notes on online learning, 2009.

Tim GJ Rudner, Cong Lu, Michael Osborne, Yarin Gal, and Yee Whye Teh. On pathologies in kl-regularized reinforcement learning from expert demonstrations. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *International Conference on Learning Representations*, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

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

Ho Chit Siu, Jaime Peña, Kimberlee C. Chang, Edenna Chen, Yutai Zhou, Victor J. Lopez, Kyle Palko, and Ross E. Allen. Evaluation of human-ai teams for learned and rule-based agents in hanabi. *CoRR*, abs/2107.07630, 2021. URL https://arxiv.org/abs/2107.07630.

John Maynard Smith. *Evolution and the Theory of Games*. Cambridge university press, 1982.

Peter D Taylor and Leo B Jonker. Evolutionary stable strategies and game dynamics. *Mathematical biosciences*, 40(1-2):145–156, 1978.

Yuandong Tian. Github thread for elf opengo, "[suggestion] clarify fpu in paper". https://github.com/pytorch/ELF/issues/140, 2019.

Yuandong Tian and Yan Zhu. Better computer go player with neural network and long-term prediction, 2016.

Seth Troisi. Github thread for minigo "[experiment] squeeze and excitation". https://github.com/tensorflow/minigo/issues/683, 2019.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

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

Jinzhuo Wang, Wenmin Wang, Ronggang Wang, and Wen Gao. Beyond monte carlo tree search: Playing go with deep alternative neural network and long-term evaluation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. URL https://ojs.aaai.org/index.php/AAAI/article/view/10749.

David Wu. Go neural net sandbox. https://github.com/lightvector/GoNN#raw-neural-net-results, 2018.

David Wu. Accelerating self-play learning in go. In *AAAI-20 Workshop on Reinforcement Learning in Games*, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. 2019.

A Related Work

A.1 Regularized Learning and Planning

Several prior works have explored augmenting reinforcement learning with supervised data from expert demonstrations (Vecerik et al., 2017; Nair et al., 2018). For example, Hester et al. (2018) augment Deep Q Learning with a margin loss on demonstration data that aims to make $Q(a)$ for each demonstration action higher than that of other actions. KL-regularization has also been used successfully to incorporate expert demonstrations into RL training (Rudner et al., 2021; Ng et al., 2000; Boularias et al., 2011; Wu et al., 2019; Peng et al., 2019; Siegel et al., 2019). In these settings, the standard RL objective is augmented by a KL divergence penalty that expresses the dissimilarity between the online policy and a reference policy derived from demonstrations. This helps guide exploration in RL or ameliorate inaccurate modeling of the environment in domains such as robotics. AlphaStar Vinyals et al. (2019), which achieved expert human performance in StarCraft 2, uses self-play RL initialized from supervised policies with a KL penalty term for deviating from the supervised policy during training to "aid in exploration and to preserve strategic diversity". In our work, we use the KL term to better approximate human play during inference-time search.

Prior work has explored entropy-regularized utilities in games. Ling et al. (2018) show that a particular type of entropic regularization in extensive-form games leads to quantal response equilibria. Cen et al. (2021) give fast online optimization algorithms for entropy-regularized utilities in the context of quantal response. Farina et al. (2019) regularize utilities with a KL divergence term from a precomputed Nash equilibrium strategy to design agents that trade off game-theoretic safety and exploitation. In our work, we leverage the KL divergence term towards a human-imitation learned policy instead, to regularize the utilities. And unlike previous works, we empirically study our approach in a much larger imperfect information game.

A.2 Strong Human-Compatible Policies

Prior work in multi-agent reinforcement learning has emphasized the importance of human-compatible policies in cooperative multi-agent environments. Lerer & Peysakhovich (2019) demonstrate that self-play policies may perform poorly with other agents if they do not conform to the population equilibrium (social conventions), and propose a combination of policy gradient and imitation loss directly on samples of population data.

Human-compatible policies have also been studied on the benchmark game of Hanabi, where ad hoc play with humans is regarded as an open challenge problem Bard et al. (2020). Most work on this challenge has focused on *zero-shot* coordination with humans, in which an agent must adapt to human play with no prior experience with human partners Hu et al. (2020; 2021b); Cui et al. (2021). Learning human-compatible policies from a combination of human data and planning are less well-studied in this setting.

A.2.1 MCTS, Chess and Go Prior work in tree search methods, especially in chess and Go, has typically focused on developing strong agents without concern for accurately modeling human behavior. For example in Go, a significant body of older work investigates imitation learning (IL) to obtain a baseline policy prior to use with MCTS, but tunes and evaluates the final agent via playing strength alone Tian & Zhu (2016); Cazenave (2017).

Regarding the use of search for human modeling, recent work in chess by McIlroy-Young et al. (2020b) found that pure IL outperformed all other approaches and that adding MCTS with the parameters of a standard engine significantly harmed human prediction accuracy. However in our work, using a different range of parameters, we show clear results to the contrary. In Go, Wang et al. (2017) report promising results on human prediction and playing strength using search, albeit with a specialized architecture and rollout method. Baier et al. (2018) report in the card game Spades both excellent human accuracy and playing strength by adding a human policy bias to a variant of MCTS. To our knowledge, our work is the first to demonstrate a clear gain in human prediction accuracy over deep learning models in the highly-studied domains of chess and Go via simple well-established methods of policy-regularized planning.

A.2.2 Diplomacy Diplomacy is a benchmark 7-player game that involves communication, negotiation, cooperation, and competition in a strategic multi-agent setting. While chess and Go are two-player zero-sum games for which optimal play is well-defined and can be computed through self-play Nash (1951), Diplomacy has no such guarantees and strong play likely requires modeling other agents, even in the no-press variant where natural-language communication is not allowed Bakhtin et al. (2021).

Paquette et al. (2019) showed that neural network IL on human data in no-press Diplomacy can reasonably approximate human play, but that bootstrapping RL from this agent leads to a breakdown in cooperation. Anthony et al. (2020) developed new RL methods based on fictitious play that improve the performance of agents in no-press Diplomacy, and Gray et al. (2020) showed that an equilibrium-finding regret minimization search procedure on top of human IL models achieves human-level performance in no-press Diplomacy. However, although both methods rely on the human IL model to generate a restricted action set for RL or search, neither contains any explicit regularization when choosing among those actions, and we show that the equilibrium search in Gray et al. (2020) greatly decreases the accuracy of modeling human players. Similarly, Bakhtin et al. (2021) achieved strong results in no-press Diplomacy via self-play RL both from scratch and initialized with a human-learned policy, but we show that the resulting final agents do not ultimately model humans well.

B PROOFS

In this Appendix, we present detailed proofs of Proposition 1 and Theorem 2.

B.1 KNOWN RESULTS

We start by recalling a few standard results. First, we recall the follow-the-regularized-leader (FTRL) algorithm, one of the most well-studied algorithms in online optimization. At every time t, the FTRL algorithm instantiated with domain $\mathcal{X}$, 1-strongly-convex regularizer $\phi : \mathcal{X} \to \mathbb{R}$, and learning rate $\eta > 0$, produces iterates according to

$$\boldsymbol{x}^{t+1} = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \left\{ -\frac{\phi(\boldsymbol{x})}{\eta} + \sum_{\tau=1}^{t} \ell^{\tau}(\boldsymbol{x}) \right\}, \tag{FTRL}$$

where $\boldsymbol{\ell}^1, \dots, \boldsymbol{\ell}^t$ are the convex utility functions gave as feedback by the environment. The FTRL algorithm guarantees the following regret bound.

Lemma 1 (Rakhlin (2009), Corollary 7)**.** *The iterates $\boldsymbol{x}^t \in \mathcal{X}$ produced by the FTRL algorithm set up with constant step size $\eta > 0$ and 1-strongly convex regularizer ϕ satisfy the regret bound*

$$\sum_{t=1}^{T} \ell^t(\boldsymbol{u}) - \ell^t(\boldsymbol{x}^t) \le \frac{\phi(\boldsymbol{u})}{\eta} + \sum_{t=1}^{T} \ell^t(\boldsymbol{x}^{t+1}) - \ell^t(\boldsymbol{x}^t) \qquad \forall\, \boldsymbol{u} \in \mathcal{X}. \tag{8}$$

In the analysis of Algorithm 1 we will also make use of the following technical lemma, a proof of which can be obtained starting using the same techniques as Abernethy & Rakhlin (2009, Lemma A.4)

Lemma 2. *Let $\boldsymbol{p} \in \Delta(A)$ be a distribution over a discrete set A, $\boldsymbol{q} \in \mathbb{R}^{|A|}$ be a vector, and $D > 0$ be any constant such that $\max_{a,a' \in A}\{\boldsymbol{q}(a) - \boldsymbol{q}(a')\} \le M$. Then,*

$$\frac{\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}^2}{\left(\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}\right)^2} - 1 \le \frac{\exp\{2M\}}{M^2} \sum_{a \in A} p(a) \boldsymbol{q}(a)^2.$$

Proof. Let $q_{\min} \coloneqq \min_{a \in A} \boldsymbol{q}(a)$ and $\tilde{\boldsymbol{q}}(a) \coloneqq \boldsymbol{q}(a) - q_{\min}$ be a shifted version of $\boldsymbol{q}(a)$ so that $0 \le \tilde{\boldsymbol{q}}(a) \le M$ for all $a \in A$. Let now X denote a random variable with value $\tilde{\boldsymbol{q}}(a)$ with probability $\boldsymbol{p}(a)$ for all $a \in A$. Then,

$$\begin{aligned}
\frac{\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}^2}{\left(\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}\right)^2} - 1 &= \frac{\exp\{q_{\min}\}^2 \sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{\boldsymbol{q}(a)\}^2}{\exp\{q_{\min}\}^2 \left(\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}\right)^2} - 1 \\
&= \frac{\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\tilde{\boldsymbol{q}}(a)\}^2}{\left(\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\tilde{\boldsymbol{q}}(a)\}\right)^2} - 1 \\
&= \frac{\mathbb{E}[\exp(-X)]}{[\mathbb{E}\exp(-X)]^2} - 1.
\end{aligned}$$

Applying Lemma A.4 from Abernethy & Rakhlin (2009) we obtain

$$\frac{\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}^2}{\left(\sum_{a \in A} \boldsymbol{p}(a) \cdot \exp\{-\boldsymbol{q}(a)\}\right)^2} - 1 \le \frac{\exp\{2M\} - 2M - 1}{M^2} (\mathbb{E}[X^2] - \mathbb{E}[X]^2) \le \frac{\exp\{2M\}}{M^2} \mathbb{E}[X^2],$$

which is exactly the statement. □

B.2 BOUNDING THE DISTANCE BETWEEN THE ITERATES OF ALGORITHM 1

Lemma 3. *At all times t and for all players i, the policies $\boldsymbol{\pi}_i^t$ produced by the FTRL algorithm set up with constant step size η and negative entropy regularizer $\varphi(\boldsymbol{x}) \coloneqq \sum_{a \in A_i} \boldsymbol{x}(a) \log \boldsymbol{x}(a)$, when observing the utilities $\mathcal{U}_i^t$, match the policies $\boldsymbol{\pi}_i^t$ produced by Algorithm 1.*

Proof. Plugging the particular choices of utilities and regularizer into equation FTRL, we obtain

$$\begin{aligned}
\boldsymbol{\pi}_i^{t+1} &= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \left(\sum_{t'=1}^{t} \mathcal{U}_i^{t'}(\boldsymbol{\pi}) \right) - \frac{1}{\eta} \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\} \\
&= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \eta \left(\sum_{t'=1}^{t} \sum_{a\in A_i} \boldsymbol{\pi}(a)\, u_i(a, \boldsymbol{a}_{-i}^{t'}) - \lambda_i\, \boldsymbol{\pi}(a) \log\left(\frac{\boldsymbol{\pi}(a)}{\boldsymbol{\tau}_i(a)} \right) \right) - \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\} \\
&= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \eta \sum_{a\in A_i} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} u_i(a, \boldsymbol{a}_{-i}^{t'}) \right) \boldsymbol{\pi}(a) - (1 + t\lambda_i\eta) \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\} \\
&= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \frac{\eta}{1+t\lambda_i\eta} \sum_{a\in A_i} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} u_i(a, \boldsymbol{a}_{-i}^{t'}) \right) \boldsymbol{\pi}(a) - \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\}. \qquad (9)
\end{aligned}$$

A well-known closed form solution to the above entropy-regularized problem is given by the softmax function. In particular, let

$$\boldsymbol{w}_i^{t+1}(a) \coloneqq \frac{\eta}{1+t\lambda_i\eta} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} \tilde{u}_i(a, \boldsymbol{a}_{-i}^{t'}) \right) \qquad \forall a \in A_i. \qquad (10)$$

Then,

$$\boldsymbol{\pi}_i^{t+1}(a) = \frac{\exp\{\boldsymbol{w}_i^{t+1}(a)\}}{\sum_{a'\in A_i} \exp\{\boldsymbol{w}_i^{t+1}(a')\}} \qquad \forall a \in A_i,$$

which coincides with the iterate produced by Algorithm 1. □

The next observation shows that the iterates $\boldsymbol{\pi}_i$ do not change if the utility function u_i is first shifted to be in the range $[0, D_i]$.

Remark 1. *Consider the shifted utilities*

$$\tilde{u}_i(a, \boldsymbol{a}_{-i}^{t}) \coloneqq u_i(a, \boldsymbol{a}_{-i}^{t'}) - \min_{\boldsymbol{a}\in A_1\times\cdots\times A_n} u_i(\boldsymbol{a}) \in [0, D_i] \qquad (11)$$

and let $\boldsymbol{v}$ be defined as equation 10 using $\tilde{u}_i$ in place of u_i, that is,

$$\boldsymbol{v}_i^{t+1}(a) \coloneqq \frac{\eta}{1+t\lambda_i\eta} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} \tilde{u}_i(a, \boldsymbol{a}_{-i}^{t'}) \right) \qquad \forall a \in A_i. \qquad (12)$$

Then, the iterates $\boldsymbol{\pi}_i$ can be equivalently expressed as

$$\boldsymbol{\pi}_i^{t+1}(a) = \frac{\exp\{\boldsymbol{v}_i^{t+1}(a)\}}{\sum_{a'\in A_i} \exp\{\boldsymbol{v}_i^{t+1}(a')\}} \qquad \forall a \in A_i.$$

Proof. Let $\gamma \coloneqq$
$min_{\boldsymbol{a}\in A_1\times\cdots\times A_n} u_i(\boldsymbol{a})$ denote the minimum utility that Player i can get against the actions of the opponents. Since the argmax of a function does not change if a constant is added to the objective, from equation 9 we can write

$$\begin{aligned}
\boldsymbol{\pi}_i^{t+1} &= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ -\frac{\eta}{1+t\lambda_i\eta}\gamma + \frac{\eta}{1+t\lambda_i\eta} \sum_{a\in A_i} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} u_i(a, \boldsymbol{a}_{-i}^{t'}) \right) \boldsymbol{\pi}(a) - \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\} \\
&= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \frac{\eta}{1+t\lambda_i\eta} \sum_{a\in A_i} \left(t\lambda_i \log \boldsymbol{\tau}_i(a) + \sum_{t'=1}^{t} (u_i(a, \boldsymbol{a}_{-i}^{t'}) - \gamma) \right) \boldsymbol{\pi}(a) - \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\} \\
&= \arg\max_{\boldsymbol{\pi}\in\Delta(A_i)} \left\{ \sum_{a\in A_i} \boldsymbol{v}_i^{t+1}(a)\boldsymbol{\pi}(a) - \sum_{a\in A_i} \boldsymbol{\pi}(a) \log \boldsymbol{\pi}(a) \right\},
\end{aligned}$$

where the second equality follows from the fact that $\boldsymbol{\pi} \in \Delta(A_i)$.

A solution is again given by softmax function

$$\boldsymbol{\pi}_i^{t+1}(a) = \frac{\exp\{\boldsymbol{v}_i^{t+1}(a)\}}{\sum_{a'\in A_i} \exp\{\boldsymbol{v}_i^{t+1}(a')\}} \qquad \forall a \in A_i. \qquad (13)$$

□

In the rest of the proof we will use equation 13 to analyze the iterates $\boldsymbol{\pi}_i$ produced by the algorithm.

Lemma 4. *Let $\eta \leq 1/(\lambda_i\beta_i + 2D_i)$. Then, at all times t,*

$$\|\boldsymbol{\pi}_i^{t+1} - \boldsymbol{\pi}_i^t\|_{\nabla^2\varphi(\boldsymbol{\pi}_i^t)} \leq \frac{\sqrt{3}\,e}{t\lambda_i\eta}.$$

Proof. At all times t, introduce the vector $\boldsymbol{\xi}_i^t \in \mathbb{R}^{|A_i|}$ defined as

$$\boldsymbol{\xi}_i^t(a) \coloneqq \frac{\eta}{1+t\lambda_i\eta}\big(-\lambda_i\boldsymbol{v}^t(a) + \lambda_i \log \boldsymbol{\tau}_i(a) + \tilde{u}_i(a, \boldsymbol{a}_{-i}^t)\big) \qquad \forall a \in A_i. \tag{14}$$

At all times t and for all a, it holds that

$$\begin{aligned}
\boldsymbol{v}_i^{t+1}(a) &= \frac{\eta}{1+t\lambda_i\eta}\left(\frac{1+(t-1)\lambda_i\eta}{\eta}\boldsymbol{v}^t(a) + \lambda_i \log \boldsymbol{\tau}_i(a) + \tilde{u}_i(a, \boldsymbol{a}_{-i}^t)\right) \\
&= \frac{\eta}{1+t\lambda_i\eta}\left(\frac{1+t\lambda_i\eta}{\eta}\boldsymbol{v}^t(a) - \lambda_i\boldsymbol{v}^t(a) + \lambda_i \log \boldsymbol{\tau}_i(a) + \tilde{u}_i(a, \boldsymbol{a}_{-i}^t)\right) \\
&= \boldsymbol{v}^t(a) + \boldsymbol{\xi}_i^t(a). \qquad (15)
\end{aligned}$$

Substituting equation 15 we can write

$$\boldsymbol{\pi}_i^{t+1}(a) = \frac{\exp\{\boldsymbol{v}_i^t(a)\}\cdot\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\exp\{\boldsymbol{v}_i^t(a')\}\cdot\exp\{\boldsymbol{\xi}_i^t(a')\}} = \frac{\boldsymbol{\pi}_i^t(a)\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}}. \tag{16}$$

Expanding the definition of the local norm induced by $\nabla^2\varphi(\boldsymbol{\pi}_i^t)$ we find

$$\begin{aligned}
\|\boldsymbol{\pi}_i^{t+1} - \boldsymbol{\pi}_i^t\|^2_{\nabla^2\varphi(\boldsymbol{\pi}_i^t)} &= \sum_{a\in A_i}\frac{1}{\boldsymbol{\pi}_i^t(a)}\big(\boldsymbol{\pi}_i^{t+1}(a) - \boldsymbol{\pi}_i^t(a)\big)^2 \\
&= \sum_{a\in A_i}\boldsymbol{\pi}_i^t(a)\left(\frac{\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}} - 1\right)^2 \qquad (17) \\
&= \sum_{a\in A_i}\boldsymbol{\pi}_i^t(a)\left[\left(\frac{\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}}\right)^2 - 2\left(\frac{\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}}\right) + 1\right] \\
&= \frac{\sum_{a\in A_i}\boldsymbol{\pi}_i^t(a)\exp\{\boldsymbol{\xi}_i^t(a)\}^2}{\big(\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}\big)^2} - 2\frac{\sum_{a\in A_i}\boldsymbol{\pi}_i^t(a)\exp\{\boldsymbol{\xi}_i^t(a)\}}{\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}} + \sum_{a\in A_i}\boldsymbol{\pi}_i^t(a) \\
&= \frac{\sum_{a\in A_i}\boldsymbol{\pi}_i^t(a)\exp\{\boldsymbol{\xi}_i^t(a)\}^2}{\big(\sum_{a'\in A_i}\boldsymbol{\pi}_i^t(a')\exp\{\boldsymbol{\xi}_i^t(a')\}\big)^2} - 1, \qquad (18)
\end{aligned}$$

where equation 17 follows from substituting equation 16. We now apply Lemma 2, applied with $\boldsymbol{q} = \boldsymbol{\xi}_i^t$, $\boldsymbol{p} = \boldsymbol{\pi}_i^t$, and $A = A_i$. First, we study the range $D_i = \max_{a,a'\in A_i}\{\boldsymbol{\xi}_i^t(a) - \boldsymbol{\xi}_i^t(a')\}$ used in the statement of the Lemma. In particular, using equation 15 we have

$$\begin{aligned}
\max_{a,a'\in A_i}\{\boldsymbol{\xi}_i^t(a) - \boldsymbol{\xi}_i^t(a')\} &= \max_{a,a'\in A_i}\{\boldsymbol{v}_i^{t+1}(a) - \boldsymbol{v}_i^t(a) - \boldsymbol{v}_i^{t+1}(a') + \boldsymbol{v}_i^t(a')\} \\
&= \max_{a,a'\in A_i}\Bigg\{(\log\boldsymbol{\tau}_i(a) - \log\boldsymbol{\tau}_i(a'))\cdot\left(\frac{t\lambda_i\eta}{1+t\lambda_i\eta} - \frac{(t-1)\lambda_i\eta}{1+(t-1)\lambda_i\eta}\right) \\
&\qquad + \frac{\eta}{1+t\lambda_i\eta}\left(\sum_{t'=1}^{t}\tilde{u}_i(a, \boldsymbol{a}_{-i}^{t'}) - \tilde{u}_i(a', \boldsymbol{a}_{-i}^{t'})\right) \\
&\qquad - \frac{\eta}{1+(t-1)\lambda_i\eta}\left(\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}_{-i}^{t'}) - \tilde{u}_i(a', \boldsymbol{a}_{-i}^{t'})\right)\Bigg\} \\
&= \max_{a,a'\in A_i}\Bigg\{\frac{\lambda_i\eta}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}(\log\boldsymbol{\tau}_i(a) - \log\boldsymbol{\tau}_i(a')) \\
&\qquad + \frac{\lambda_i\eta^2}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\left(\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}_{-i}^{t'}) - \tilde{u}_i(a', \boldsymbol{a}_{-i}^{t'})\right)
\end{aligned}$$

$$
\begin{aligned}
&\qquad\qquad + \frac{\eta}{1+t\lambda_i\eta}(\tilde{u}_i(a, \boldsymbol{a}^t_{-i}) - \tilde{u}_i(a', \boldsymbol{a}^t_{-i}))\Bigg\} \\
&\leq \max_{a,a'\in A_i}\left\{\frac{\lambda_i\eta}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}(\log \boldsymbol{\tau}_i(a) - \log \boldsymbol{\tau}_i(a'))\right\} \\
&\qquad + \max_{a,a'\in A_i}\left\{\frac{\lambda_i\eta^2}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\left(\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i}) - \tilde{u}_i(a', \boldsymbol{a}^{t'}_{-i})\right)\right\} \\
&\qquad\qquad + \max_{a,a'\in A_i}\left\{\frac{\eta}{1+t\lambda_i\eta}(\tilde{u}_i(a, \boldsymbol{a}^t_{-i}) - \tilde{u}_i(a', \boldsymbol{a}^t_{-i}))\right\} \\
&\leq \eta(\lambda_i\beta_i + 2D_i), \qquad (19)
\end{aligned}
$$

where the first inequality follows from upper bounding the max of a sum with the sum of max of each term, and the second inequality follows from noting that

$$
\frac{\lambda_i\eta}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)} \leq \frac{\eta}{1+t\lambda_i\eta} \leq \eta,
$$

and

$$
\frac{\lambda_i\eta^2}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)} \leq \frac{\eta}{t}\left(\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i}) - \tilde{u}_i(a', \boldsymbol{a}^{t'}_{-i})\right) \leq \frac{\lambda_i\eta^2}{t\lambda_i\eta}\cdot tD_i = \eta D_i.
$$

Applying Lemma 2 to the right-hand side of equation 18 using the bound on the range of $\boldsymbol{\xi}^t_i$ shown in equation 19 yields

$$
\|\boldsymbol{\pi}^{t+1}_i - \boldsymbol{\pi}^t_i\|^2_{\nabla^2\varphi(\boldsymbol{\pi}^t_i)} \leq \frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\sum_{a\in A_i}\boldsymbol{\pi}^t_i(a)\big(\boldsymbol{\xi}^t_i(a)\big)^2. \qquad (20)
$$

Using the fact that any convex combination of values is upper bounded by the maximum value, we can further bound the right-hand side of equation 20 as

$$
\begin{aligned}
\|\boldsymbol{\pi}^{t+1}_i - \boldsymbol{\pi}^t_i\|^2_{\nabla^2\varphi(\boldsymbol{\pi}^t_i)} &\leq \frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}(\boldsymbol{\xi}^t_i(a))^2 \\
&= \frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}\big(\boldsymbol{v}^{t+1}_i(a) - \boldsymbol{v}^t_i(a)\big)^2,
\end{aligned}
$$

where the equality follows from equation 15. Hence, expanding the definition of $\boldsymbol{v}^t_i$ and $\boldsymbol{v}^{t+1}_i$,

$$
\begin{aligned}
\|\boldsymbol{\pi}^{t+1}_i - \boldsymbol{\pi}^t_i\|^2_{\nabla^2\varphi(\boldsymbol{\pi}^t_i)} &\leq \frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}\Bigg\{\left(\frac{t\lambda_i\eta}{1+t\lambda_i\eta} - \frac{(t-1)\lambda_i\eta}{1+(t-1)\lambda_i\eta}\right)\log \boldsymbol{\tau}_i(a) \\
&\qquad + \frac{\eta}{1+t\lambda_i\eta}\sum_{t'=1}^{t}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i}) - \frac{\eta}{1+(t-1)\lambda_i\eta}\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i})\Bigg\}^2 \\
&= \frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}\Bigg\{\frac{\lambda_i\eta}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\log \boldsymbol{\tau}_i(a) + \frac{\eta}{1+t\lambda_i\eta}\tilde{u}_i(a, \boldsymbol{a}^t_{-i}) \\
&\qquad - \frac{\lambda_i\eta^2}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i})\Bigg\}^2 \\
&\leq 3\frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}\Bigg\{\left(\frac{\lambda_i\eta}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\log \boldsymbol{\tau}_i(a)\right)^2 + \left(\frac{\eta}{1+t\lambda_i\eta}\tilde{u}_i(a, \boldsymbol{a}^t_{-i})\right)^2 \\
&\qquad + \left(\frac{\lambda_i\eta^2}{(1+t\lambda_i\eta)(1+(t-1)\lambda_i\eta)}\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i})\right)^2\Bigg\} \\
&= \frac{3}{(1+t\lambda_i\eta)^2}\frac{\exp\{2\eta(\lambda_i\beta_i + 2D_i)\}}{\eta^2(\lambda_i\beta_i + 2D_i)^2}\max_{a\in A_i}\Bigg\{\left(\frac{\lambda_i\eta}{1+(t-1)\lambda_i\eta}\log \boldsymbol{\tau}_i(a)\right)^2 + \big(\eta\tilde{u}_i(a, \boldsymbol{a}^t_{-i})\big)^2 \\
&\qquad + \left(\frac{\lambda_i\eta^2}{(1+(t-1)\lambda_i\eta)}\sum_{t'=1}^{t-1}\tilde{u}_i(a, \boldsymbol{a}^{t'}_{-i})\right)^2\Bigg\}
\end{aligned}
$$

$$
\begin{aligned}
&\le \frac{3}{(1+t\lambda_i\eta)^2}\frac{\exp\{2\eta(\lambda_i\beta_i+2D_i)\}}{\eta^2(\lambda_i\beta_i+2D_i)^2}(\lambda^2\eta^2\beta_i^2+2\eta^2D_i^2)\\
&\le 3\frac{\exp\{2\eta(\lambda_i\beta_i+2D_i)\}}{(1+t\lambda_i\eta)^2}\\
&\le 3\frac{\exp\{2\eta(\lambda_i\beta_i+2D_i)\}}{(t\lambda_i\eta)^2} = \left(\sqrt{3}\cdot\frac{\exp\{\eta(\lambda_i\beta_i+2D_i)\}}{t\lambda_i\eta}\right)^2.
\end{aligned}
$$

Using the hypothesis that $\eta \le 1/(\lambda_i\beta_i+2D_i)$ and taking square roots yields the statement. □

B.3 Completing the analysis

Proposition 1. *Fix a player $i \in \{1,\dots,n\}$. The regret*

$$
R_i^T \coloneqq \max_{\boldsymbol{\pi}^*\in\Delta(A_i)}\sum_{t=1}^T \mathcal{U}_i^t(\boldsymbol{\pi}^*) - \sum_{t=1}^T \mathcal{U}_i^t(\boldsymbol{\pi}_i^t)
$$

incurred up to any time T by policies $\boldsymbol{\pi}_i^t$ defined in equation 6 where the learning rate is set to any value $0<\eta\le 1/(\lambda_i\beta_i+2D_i)$, satisfies

$$
R_i^T \le \frac{\log|A_i|}{\eta} + \frac{3\,e(1+\log T)}{\lambda_i\eta}(D_i+\lambda_i\beta_i+\lambda_i\sqrt{|A_i|}),
$$

where D_i is any upper bound on the range of the possible rewards of Player i, and

$$
\beta_i \coloneqq \max_{a\in A_i}\log(1/\boldsymbol{\tau}(a)). \tag{21}
$$

Proof. Let

$$
\boldsymbol{q}_i^t \coloneqq \Big(\tilde{u}_i(a,\boldsymbol{a}_{-i}^t)\Big)_{a\in A_i},
$$

and note that, by definition of the regularized utilities $\mathcal{U}_i$,

$$
\begin{aligned}
\mathcal{U}_i^t(\boldsymbol{\pi}_i^{t+1}) - \mathcal{U}_i^t(\boldsymbol{\pi}_i^t) &= \boldsymbol{q}_i^\top\big(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big) - \lambda_i D_{\mathrm{KL}}(\boldsymbol{\pi}_i^{t+1}\,\|\,\boldsymbol{\tau}) + \lambda_i D_{\mathrm{KL}}(\boldsymbol{\pi}_i^t\,\|\,\boldsymbol{\tau})\\
&= \boldsymbol{q}_i^\top\big(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big) - \lambda_i\varphi(\boldsymbol{\pi}_i^{t+1}) + \lambda_i\varphi(\boldsymbol{\pi}_i^t) - \lambda_i\nabla\varphi(\boldsymbol{\tau}_i)^\top(\boldsymbol{\pi}_i^t-\boldsymbol{\pi}_i^{t+1})\\
&= (\boldsymbol{q}_i-\nabla\varphi(\boldsymbol{\tau}_i))^\top\big(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big) - \lambda_i\varphi(\boldsymbol{\pi}_i^{t+1}) + \lambda_i\varphi(\boldsymbol{\pi}_i^t)\\
&\le (\boldsymbol{q}_i+\nabla\varphi(\boldsymbol{\tau}_i))^\top\big(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big) - \lambda_i\varphi(\boldsymbol{\pi}_i^t) - \lambda_i\nabla\varphi(\boldsymbol{\pi}_i^t)^\top(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t) + \lambda_i\varphi(\boldsymbol{\pi}_i^t)\\
&= \big(\boldsymbol{q}_i+\lambda_i\nabla\varphi(\boldsymbol{\tau}_i)-\lambda_i\nabla\varphi(\boldsymbol{\pi}_i^t)\big)^\top\big(\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big)\\
&\le \big\|\boldsymbol{q}_i+\lambda_i\nabla\varphi(\boldsymbol{\tau}_i)-\lambda_i\nabla\varphi(\boldsymbol{\pi}_i^t)\big\|_{\nabla^{-2}\varphi(\boldsymbol{\pi}_i^t)}\cdot\big\|\boldsymbol{\pi}_i^{t+1}-\boldsymbol{\pi}_i^t\big\|_{\nabla^2\varphi(\boldsymbol{\pi}_i^t)},
\end{aligned}
$$

where the first inequality follows by convexity and the second inequality by the generalized Cauchy-Schwarz inequality with the primal-dual norm pair $\|\cdot\|_{\nabla^2\varphi(\boldsymbol{\pi}_i^t)}$ and $\|\cdot\|_{\nabla^{-2}\varphi(\boldsymbol{\pi}_i^t)}$. A bound for the second term in the product if given by Lemma 4. We now bound the first norm. First,

$$
\begin{aligned}
\big\|\boldsymbol{q}_i+\lambda_i\nabla\varphi(\boldsymbol{\tau}_i)-\lambda_i\nabla\varphi(\boldsymbol{\pi}_i^t)\big\|^2_{\nabla^{-2}\varphi(\boldsymbol{\pi}_i^t)} &= \sum_{a\in A_i}\boldsymbol{\pi}^t(a)\cdot\big(\tilde{u}_i(a,\boldsymbol{a}_{-i}^t)+\lambda_i\log\boldsymbol{\tau}_i(a)-\lambda_i\log\boldsymbol{\pi}_i^t(a)\big)^2\\
&\le 3\sum_{a\in A_i}\boldsymbol{\pi}^t(a)\cdot\big(\tilde{u}_i(a,\boldsymbol{a}_{-i}^t)^2+\lambda_i^2(\log\boldsymbol{\tau}_i(a))^2+\lambda_i^2(\log\boldsymbol{\pi}_i^t(a))^2\big)\\
&\le 3\big(D_i^2+\lambda_i^2\beta_i^2+\lambda_i^2|A_i|\big)\\
&\le 3\Big(D_i+\lambda_i\beta_i+\lambda_i\sqrt{|A_i|}\Big)^2,
\end{aligned}
$$

where the second inequality follows from the fact that $x\log^2 x\le 1$ for all $x\in[0,1]$. Taking square roots, we find

$$
\big\|\boldsymbol{q}_i+\lambda_i\nabla\varphi(\boldsymbol{\tau}_i)-\lambda_i\nabla\varphi(\boldsymbol{\pi}_i^t)\big\|_{\nabla^{-2}\varphi(\boldsymbol{\pi}_i^t)} \le \sqrt{3}\Big(D_i+\lambda_i\beta_i+\lambda_i\sqrt{|A_i|}\Big).
$$

So, using Lemma 4, we can write

$$
\mathcal{U}_i^t(\boldsymbol{\pi}_i^{t+1}) - \mathcal{U}_i^t(\boldsymbol{\pi}_i^t) \le \frac{3\,e}{t\lambda_i\eta}(D_i+\lambda_i\beta_i+\lambda_i\sqrt{|A_i|}).
$$

Plugging in the above expression into Lemma 1 yields

$$
\begin{aligned}
R_i^T &\le \frac{\log |A_i|}{\eta} + \sum_{t=1}^{T} \frac{3\,e}{t\lambda_i \eta}(D_i + \lambda_i \beta_i + \lambda_i \sqrt{|A_i|}) \\
&\le \frac{\log |A_i|}{\eta} + \frac{3\,e(1+\log T)}{\lambda_i \eta}(D_i + \lambda_i \beta_i + \lambda_i \sqrt{|A_i|}),
\end{aligned}
$$

which is the statement. □

B.4 Proof of Theorem 1

Theorem 1. *(piKL stays close to the anchor policy) Upon running Algorithm 1 for T iterations in a multiplayer general-sum game, the policy $\bar{\boldsymbol{\pi}}_i^T$ is at a distance $D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \le \frac{1}{\lambda_i}\left(\frac{R_i^T}{T} + D_i\right)$, where D_i is any upper bound on possible rewards for Player i. In particular, if $\eta > 0$ is set so that $R_i^T = o(T)$, then $D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \to D_i/\lambda_i$ as $T \to +\infty$.*

Proof. By definition of regret,

$$
\begin{aligned}
\frac{1}{T} R_i^T &= \frac{1}{T} \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \sum_{t=1}^{T} \mathcal{U}_i^t(\boldsymbol{\pi}_i^*) - \mathcal{U}_i^t(\boldsymbol{\pi}_i^t) \right\} \\
&= \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^t) \right) - \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \right) - \frac{\lambda_i}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(\boldsymbol{\pi}_i^* \,\|\, \boldsymbol{\tau}_i) + \frac{\lambda_i}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(\boldsymbol{\pi}_i^t \,\|\, \boldsymbol{\tau}_i) \right\} \\
&= \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^t) \right) - \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \right) - \lambda_i D_{\mathrm{KL}}(\boldsymbol{\pi}_i^* \,\|\, \boldsymbol{\tau}_i) + \frac{\lambda_i}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(\boldsymbol{\pi}_i^t \,\|\, \boldsymbol{\tau}_i) \right\} \\
&\ge \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^t) \right) - \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \right) + \frac{\lambda_i}{T} \sum_{t=1}^{T} D_{\mathrm{KL}}(\boldsymbol{\pi}_i^t \,\|\, \boldsymbol{\tau}_i) \right\} \\
&\ge \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^t) \right) - \left(\frac{1}{T} \sum_{t=1}^{T} u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \right) + \lambda_i D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \right\} \\
&= \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} (u_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}^t) - u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \right) + \lambda_i D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \right\} \\
&\ge \max_{\boldsymbol{\pi}_i^* \in \Delta(A_i)} \left\{ \left(\frac{1}{T} \sum_{t=1}^{T} -D_i \right) + \lambda_i D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i) \right\} = -D_i + \lambda_i D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_i^T \,\|\, \boldsymbol{\tau}_i),
\end{aligned}
$$

where the first inequality holds since the KL divergence is nonnegative, the second inequality by convexity of the KL divergence, and the third inequality by definition of D_i. Rearranging yields the inequality in the statement. □

B.5 Relationship with Nash Equilibrium

In this subsection, we show that when all players play according to Algorithm 1 in a *two-player zero-sum* game, then the average policies $\bar{\boldsymbol{\pi}}_i^T$ converge to a Nash equilibrium *of the regularized game whose utilities are $\mathcal{U}_i$.*

Proposition 2. *For any $T \in \mathbb{N}$, $\eta > 0$, and $\delta \in (0,1)$, define the quantity*

$$
\xi^T(\delta) \coloneqq \frac{R_1^T + R_2^T}{T} + (\max_i D_i) \sqrt{\frac{32}{T} \log \frac{2 \max_i |A_i|}{\delta}}.
$$

Upon running Algorithm 1 for any T iterations with learning rate $\eta > 0$, the average policies $\bar{\boldsymbol{\pi}}_i^T$ of each player form a $\xi^T(\delta)$-approximate Nash equilibrium with respect to the regularized utility functions $\mathcal{U}_i$ with probability at least $1-\delta$, for any $\delta \in (0,1)$.

Proof. Fix a player $i \in \{1,2\}$, and any policy $\boldsymbol{\pi}^* \in \Delta(A_i)$, and introduce the discrete-time stochastic process

$$
w^t \coloneqq \Big(\mathcal{U}_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - \mathcal{U}_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \Big) - \Big(\mathcal{U}_i(\boldsymbol{\pi}^*, a_{-i}^t) - \mathcal{U}_i(\boldsymbol{\pi}_i^t, a_{-i}^t) \Big).
$$

Since the opponent player $-i$ plays according to Algorithm 1, its action a_{-i}^t at all times t is selected by sampling (unbiasedly) an action from the policy $\boldsymbol{\pi}_{-i}^t$. Therefore, w^t is a martingale difference sequence. Furthermore, by expanding the definition of $\mathcal{U}_i$, the absolute value of w^t satisfies

$$\begin{aligned}|w^t| &= \left|\Big(u_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t)\Big) - \Big(u_i(\boldsymbol{\pi}^*, a_{-i}^t) - u_i(\boldsymbol{\pi}_i^t, a_{-i}^t)\Big)\right| \\ &\le \left|u_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - u_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t)\right| - \left|u_i(\boldsymbol{\pi}^*, a_{-i}^t) - u_i(\boldsymbol{\pi}_i^t, a_{-i}^t)\right| \le 2D_i.\end{aligned}$$

Hence, using Azuma-Hoeffding's inequality, for any $\delta \in (0, 1)$,

$$\begin{aligned}1 - \delta &\le \mathbb{P}\left[\sum_{t=1}^T w^t \le D_i\sqrt{8T\log\frac{1}{\delta}}\right] \\ &= \mathbb{P}\left[\left(\sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - \sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t)\right) - \left(\sum_{t=1}^T u_i(\boldsymbol{\pi}^*, a_{-i}^t) - \sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}_i^t, a_{-i}^t)\right) \le \sqrt{8T\log\frac{1}{\delta}}\right] \\ &= \mathbb{P}\left[\sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - \sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \le R_i^T + D_i\sqrt{8T\log\frac{1}{\delta}}\right],\end{aligned}$$

where R_i^T is as defined in Proposition 1. Since the above expression holds for any $\boldsymbol{\pi}^* \in \Delta(A_i)$, in particular, using the union bound on each $a \in A_i$,

$$\mathbb{P}\left[\max_{\boldsymbol{\pi}^* \in \Delta(A_i)} \sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) - \sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}_i^t, \boldsymbol{\pi}_{-i}^t) \le R_i^T + D_i\sqrt{8T\log\frac{|A_i|}{\delta}}\right] \ge 1 - \delta \tag{22}$$

for any player $i \in \{1, 2\}$ and any $\delta \in (0, 1)$.

Summing Inequality equation 22 for $i \in \{1, 2\}$ and using the union bound, we can further write

$$\begin{aligned}\mathbb{P}\Bigg[&\max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \left\{\sum_{t=1}^T \mathcal{U}_1(\boldsymbol{\pi}_1^*, \boldsymbol{\pi}_2^t)\right\} + \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \left\{\sum_{t=1}^T \mathcal{U}_2(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^*)\right\} - \left(\sum_{t=1}^T \mathcal{U}_1(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t) + \mathcal{U}_2(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t)\right) \\ &\le R_1^T + R_2^T + (\max_i D_i)\sqrt{32T\log\frac{\max_i |A_i|}{\delta}}\Bigg] \ge 1 - 2\delta.\end{aligned}$$

Dividing by T and noting that for any player $i \in \{1, 2\}$

$$\frac{1}{T}\sum_{t=1}^T \mathcal{U}_i(\boldsymbol{\pi}^*, \boldsymbol{\pi}_{-i}^t) = \mathcal{U}_i\left(\boldsymbol{\pi}^*, \frac{1}{T}\sum_{t=1}^T \boldsymbol{\pi}_{-i}^t\right) = \mathcal{U}_i(\boldsymbol{\pi}^*, \bar{\boldsymbol{\pi}}_{-i}^T)$$

further yields

$$\begin{aligned}\mathbb{P}\Bigg[&\max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \left\{\mathcal{U}_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2^T)\right\} + \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \left\{\mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \boldsymbol{\pi}_2^*)\right\} - \frac{1}{T}\left(\sum_{t=1}^T \mathcal{U}_1(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t) + \mathcal{U}_2(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t)\right) \\ &\le \frac{R_1^T + R_2^T}{T} + D_i\sqrt{\frac{32}{T}\log\frac{\max_i |A_i|}{\delta}}\Bigg] \ge 1 - 2\delta.\end{aligned} \tag{23}$$

We now analyze the term in parenthesis, that is,

$$(\clubsuit) \coloneqq -\frac{1}{T}\left(\sum_{t=1}^T \mathcal{U}_1(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t) + \mathcal{U}_2(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t)\right)$$

Plugging in the definition of $\mathcal{U}_1$ and $\mathcal{U}_2$, that is,

$$\begin{aligned}\mathcal{U}_1(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) &\coloneqq u_1(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) - \lambda_1 D_{\mathrm{KL}}(\boldsymbol{\pi}_1 \,\|\, \boldsymbol{\tau}_1) \\ \mathcal{U}_2(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) &\coloneqq u_2(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) - \lambda_2 D_{\mathrm{KL}}(\boldsymbol{\pi}_2 \,\|\, \boldsymbol{\tau}_2) = -u_1(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) + \lambda_2 D_{\mathrm{KL}}(\boldsymbol{\pi}_2 \,\|\, \boldsymbol{\tau}_2)\end{aligned}$$

into (♣) yields

$$(\clubsuit) = -\frac{1}{T}\left(\sum_{t=1}^{T} \mathcal{U}_1(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t) + \mathcal{U}_2(\boldsymbol{\pi}_1^t, \boldsymbol{\pi}_2^t)\right) = \frac{1}{T}\left(\sum_{t=1}^{T} \lambda_1 D_{\mathrm{KL}}(\boldsymbol{\pi}_1^t \,\|\, \boldsymbol{\tau}_1) + \lambda_2 D_{\mathrm{KL}}(\boldsymbol{\pi}_2^t \,\|\, \boldsymbol{\tau}_2)\right)$$

$$\geq \lambda_1 D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_1^T \,\|\, \boldsymbol{\tau}_1) + \lambda_2 D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_2^T \,\|\, \boldsymbol{\tau}_2) \tag{24}$$

$$= -\big(\mathcal{U}_1(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T) + \mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\big), \tag{25}$$

where equation 24 follows from convexity of the KL divergence function, and equation 25 follows again from the definition of $\mathcal{U}_1$ and $\mathcal{U}_2$. Substituting equation 25 back into equation 23, we find

$$\mathbb{P}\Bigg[\max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{\mathcal{U}_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2^T) - \mathcal{U}_1(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\} + \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \{\mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \boldsymbol{\pi}_2^*) - \mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\}$$
$$\leq \frac{R_1^T + R_2^T}{T} + (\max_i D_i)\sqrt{\frac{32}{T}\log\frac{\max_i |A_i|}{\delta}}\Bigg] \geq 1 - 2\delta. \tag{26}$$

Since

$$\max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{\mathcal{U}_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2^T) - \mathcal{U}_1(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\} \geq 0, \quad \text{and} \quad \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \{\mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \boldsymbol{\pi}_2^*) - \mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\} \geq 0,$$

the inequality above in particular implies that

$$\mathbb{P}\Bigg[\max\Bigg\{\max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{\mathcal{U}_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2^T) - \mathcal{U}_1(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\}, \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \{\mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \boldsymbol{\pi}_2^*) - \mathcal{U}_2(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)\}\Bigg\}$$
$$\leq \frac{R_1^T + R_2^T}{T} + (\max_i D_i)\sqrt{\frac{32}{T}\log\frac{\max_i |A_i|}{\delta}}\Bigg] \geq 1 - 2\delta, \tag{27}$$

which is equivalent to the statement after making the variable substitution $\delta \coloneqq \delta'/2$. □

In particular, when $\eta \leq 1/(\lambda_i\beta_i + 2D_i)$ for both players $i \in \{1, 2\}$, Proposition 2 implies that the average strategy profile is a $O(1/\sqrt{T})$-Nash equilibrium with respect to the regularized utility functions $\mathcal{U}_i$.

A standard application of the Borel-Cantelli lemma enables to convert from the high-proability guarantees of Proposition 2 at finite time to almost-sure convergence in the limit. Specifically,

Corollary 1. *Let $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ be any limit point of the average policies $(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)$ of the players. Almost surely, $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ is a Nash equilibrium with respect to the regularized utility functions $\mathcal{U}_1, \mathcal{U}_2$, respectively.*

From there, it is immediate to give guarantees with respect to the original (*i.e.*, unregularized) game, and Theorem 2 follows.

Theorem 2. *Let $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ be any limit point of the average policies $(\bar{\boldsymbol{\pi}}_1^T, \bar{\boldsymbol{\pi}}_2^T)$ of the players. Almost surely, $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ is a $(\max_{i=1,2}\{\lambda_i\beta_i\})$-approximate Nash equilibrium policy with respect to the original utility functions u_i, where β_i is as defined in equation 21.*

Proof. From Corollary 1, almost surely $(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)$ is a Nash equilibrium of the regularized game whose players' utilities are $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. Expanding the definition of Nash equilibrium relative to Player 1, we have that

$$\begin{aligned}
0 &= \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{\mathcal{U}_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - \mathcal{U}_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)\} \\
&= \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{u_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - \lambda_1 D_{\mathrm{KL}}(\boldsymbol{\pi}_1^* \,\|\, \boldsymbol{\tau}_1) - u_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2) + \lambda_1 D_{\mathrm{KL}}(\bar{\boldsymbol{\pi}}_1 \,\|\, \boldsymbol{\tau}_1)\} \\
&\geq \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{u_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - u_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)\} - \lambda_1 D_{\mathrm{KL}}(\boldsymbol{\pi}_1^* \,\|\, \boldsymbol{\tau}_1) \\
&= \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \left\{(u_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - u_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)) - \lambda_1 \sum_{a \in A_1} \boldsymbol{\pi}_i^*(a)\log(\boldsymbol{\pi}_i^*(a)) - \lambda_1 \sum_{a \in A_1} \boldsymbol{\pi}_1^*(a)\log(1/\boldsymbol{\tau}_1(a))\right\} \\
&\geq \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \left\{(u_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - u_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)) - \lambda_1 \sum_{a \in A_1} \boldsymbol{\pi}_1^*(a)\log(1/\boldsymbol{\tau}_1(a))\right\} \\
&\geq \max_{\boldsymbol{\pi}_1^* \in \Delta(A_1)} \{u_1(\boldsymbol{\pi}_1^*, \bar{\boldsymbol{\pi}}_2) - u_1(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)\} - \lambda_1\beta_1,
\end{aligned}$$

where the first inequality follows since the KL divergence is alwasy nonnegative, the second inequality since the negative entropy function is nonpositive on the simplex, and the third inequality follows from the definition of β_1. Symmetrically, for Player 2 we find that

$$0 \geq \max_{\boldsymbol{\pi}_2^* \in \Delta(A_2)} \{u_2(\bar{\boldsymbol{\pi}}_1, \boldsymbol{\pi}_2^*) - u_2(\bar{\boldsymbol{\pi}}_1, \bar{\boldsymbol{\pi}}_2)\} - \lambda_2 \beta_2.$$

Hence, the exploitability of $\bar{\boldsymbol{\pi}}_1$ is at most $\lambda_1 \beta_1$, while the exploitability of $\bar{\boldsymbol{\pi}}_2$ is at most $\lambda_2 \beta_2$, which immediately implies the statement. □

C ILLUSTRATIONS OF PIKL-HEDGE IN BLOTTO

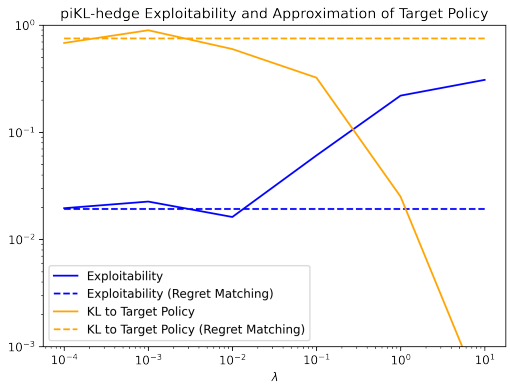

Figure 4: Comparison of piKL and regret matching as a function of λ in Colonel Blotto(10, 3). As λ increases in piKL-hedge, piKL moves closer to the anchor policy at the cost of increased exploitability. The scale of λ is related to the scale of the payoffs in the game, which are [0, 1] in Blotto.

Colonel Blotto is a famous 2-player simultaneous action game that has a large action space but has rules that are short and simple. In Blotto, each player has c coins to be distributed across f fields. The aim is to win the most fields by allocating the player's coins across the fields. A field is won by contributing the most coins to that field (and drawn if there is a tie). The winner receives a reward of +1 and the loser receives -1. Both receive 0 in the case of a tie.

In Figure 4 we illustrate the key features of piKL-Hedge in Blotto, using a uniform anchor policy for convenience. Incidentally, piKL-Hedge with a uniform anchor policy converges to a quantal response equilibrium McKelvey & Palfrey (1995b). piKL-Hedge finds policies that play close to the anchor policy while having low regret, with λ controlling the relative optimality of these two desiderata.

D HUMAN POLICY KL-REGULARIZED SEARCH ALSO IMPROVES CROSS-ENTROPY

Model	Dataset	**(raw model)**	$c_{\text{puct}} = 10$	$c_{\text{puct}} = 5$	$c_{\text{puct}} = 2$	$c_{\text{puct}} = 1$	$c_{\text{puct}} = 0.5$
Maia1500 (Chess)	Lichess 1500 Rating Bucket	1.476	1.469	1.465	1.470	1.504	1.598
Maia1900 (Chess)	Lichess 1900 Rating Bucket	1.440	1.429	1.422	1.418	1.443	1.529
Our Model (Go)	GoGoD	1.388	1.362	1.359	1.362	1.391	1.478

Table 3: Cross-entropy predicting human moves in chess and Go using smooth KL optimization post-processing of MCTS with various c_{puct}. Largest standard error of any value is around 0.0033.

Here we show that not only does policy-regularized search improve top-1 accuracy, it also improves cross entropy for a reasonable range of parameter values in both chess and Go, as well as performing well in Diplomacy.

For Chess and Go, to obtain the policy distribution with which to compute its cross entropy with the human data, unlike for top-1 accuracy or for play we cannot directly use the MCTS visit distribution because occasionally simply due to discretization MCTS may give zero visits to the actual move that a human played, resulting in an undefined (i.e. infinite) cross-entropy. Instead, we leverage the result of Grill et al. (2020) that PUCT-style MCTS with a policy prior can be seen as a discrete approximation to solving a smooth optimization:

$$\arg\max_{\pi} \sum_{a} Q(s,a)\pi(s,a) + \lambda D_{\text{KL}}(\tau \,\|\, \pi)$$

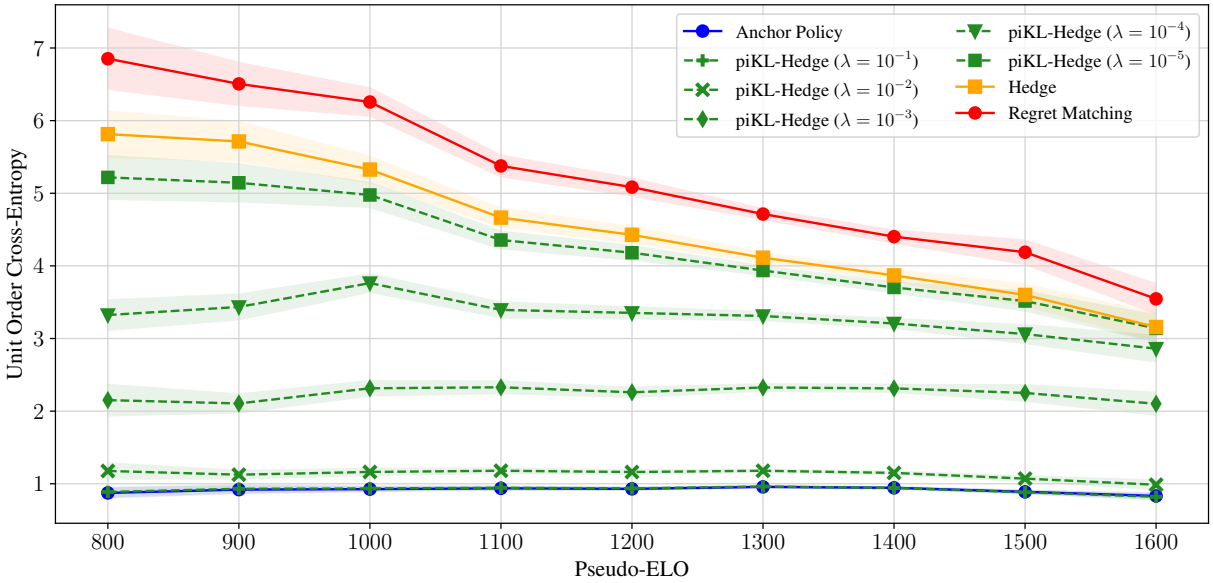

Figure 5: Average cross-entropy of per-unit order prediction in Diplomacy, comparing the human-imitation-learned anchor policy, regret matching, hedge, and piKL-Hedge, as a function of pseudo-Elo player rating.

where

$$\lambda = c_{\text{puct}} \frac{\sqrt{\sum_a n_a}}{(k + \sum_a n_a)}$$

where Q is the current value estimate from search for each action, τ is the anchor or prior policy, λ controls the strength of the regularization towards that prior as a function of the number of visits, c_{puct} is the MCTS exploration coefficient, n_a is the number of times a was explored and k is an arbitrary constant not affecting the asymptotic results.

We perform MCTS exactly the same as normal, the only difference is that at the very end, rather than using visit counts, we compute π optimizing the above objective using the human imitation-learned anchor policy for τ and the MCTS-estimated Q-values for Q (and using the same unweighted average child value for moves that lack a Q-value estimate due to having zero visits as described in Appendix G). We use this resulting smooth π as the final policy prediction and compute its cross entropy with the actual human moves.

Note that the authors choose $k = |A|$, the number of legal actions, for mathematical convenience, corresponding to adding one extra visit per every possible action (Grill et al., 2020), but since in our experiments we use only use 50 visits including the root visit (i.e. $\sum_a n_a = 49$), and the branching factor can be as large as 362 in Go, adding one extra visit per legal action in our case greatly overestimates the total number of visits, which in practice gives a less accurate correspondence between MCTS and this smoother regularized solution, so we instead choose $k = 0$.

In Table 3 we show the results. Across roughly the same parameter ranges, the regularized search policy using this smoothed MCTS postprocessing achieves lower cross-entropy with human moves than the raw imitation-learned policy without search. This suggests that not only does search improve on the raw imitation-learned policy at pinpointing the top action, it also gives a more accurate model of the overall distribution of likely human actions.

Similarly, in Figure 5, we show that piKL-Hedge achieves better average cross entropy of unit orders in no-press Diplomacy compared to unregularized search methods, and matches that of imitation learning at $\lambda = 0.1$. This corroborates the results of Section 4.3 and shows that piKL-Hedge provides the same benefits in modeling the overall distribution of human actions as it does on predicting the top move - outpredicting unregularized search (while playing as well or slightly better against human-like opponents), or equaling the prediction quality of imitation learning (while playing much more strongly than IL).

E MORE EXPERIMENTS IN CHESS AND GO

Game	c_{puct}	**MCTS Win% vs raw model** temp = 1	temp = 0.3
Chess	10.0	72.2% ± 1.2%	62.4% ± 1.3%
Chess	5.0	79.5% ± 1.1%	72.3% ± 1.2%
Chess	2.0	88.0% ± 0.8%	86.1% ± 0.9%
Chess	1.0	92.2% ± 0.7%	92.9% ± 0.6%
Chess	0.5	94.4% ± 0.6%	94.7% ± 0.5%
Go	10.0	73.2% ± 1.4%	63.3% ± 1.5%
Go	5.0	80.5% ± 1.3%	74.1% ± 1.4%
Go	2.0	87.6% ± 1.0%	85.3% ± 1.1%
Go	1.0	94.6% ± 0.7%	94.4% ± 0.7%
Go	0.5	96.4% ± 0.6%	97.0% ± 0.5%

Table 4: Winrate of base model + MCTS vs base model at temperature 1 and 0.3. Base model is Maia1900 in chess, and our GoGoD model in Go. 1000 games per figure, draws count as half a win, ± indicates one standard error. Go uses Japanese rules with 6.5 komi. MCTS greatly improves strength in Chess and Go, the smallest c_{puct} values improve it most.

Model	Predicting	Main Time	Increment	Raw Model Acc %	MCTS Acc %	**Acc Gain from MCTS**	Approx Stderr
Maia1500	1500	3m	0s	51.9	52.1	**0.2**	0.17
Maia1500	1500	5m	0s	52.7	53.2	**0.5**	0.16
Maia1500	1500	10m	0s	52.4	53.1	**0.7**	0.20
Maia1500	1500	3m	2s	53.1	53.7	**0.6**	0.31
Maia1500	1500	5m	3s	52.5	53.4	**0.9**	0.35
Maia1500	1500	15m	15s	51.9	52.6	**0.7**	0.37
Maia1900	1900	3m	0s	53.0	53.9	**0.8**	0.12
Maia1900	1900	5m	0s	53.2	54.5	**1.3**	0.17
Maia1900	1900	10m	0s	52.9	54.6	**1.7**	0.24
Maia1900	1900	3m	2s	54.1	55.5	**1.4**	0.27
Maia1900	1900	5m	3s	53.1	55.1	**2.0**	0.51
Maia1900	1900	15m	15s	53.7	55.9	**2.2**	0.56

Table 5: Difference in top-1 accuracy between raw model and MCTS using the best c_{puct} in predicting human moves for chess players in rating buckets 1500 and 1900 using Maia1500 and Maia1900, split by time control of the games, excluding all time controls with fewer than 100 games. Approx Stderr indicates the rough standard error of raw accuracy values on that row given the number of games of that time control. Despite the statistical uncertainty on some individual values, overall the improvement of MCTS vs the raw model does clearly tend to be larger on the games with longer time controls.

This section summarizes the results of a small number of additional experiments in chess and Go. Whereas Figure 1 used a temperature of 1.0 when sampling from the agent policy, we show in Table 4 that MCTS also achieves similar winrates versus the raw model when both are sampled using a much lower temperature of 0.3.

Additionally, we confirm in Go the same result that McIlroy-Young et al. (2020a) reported in chess, that current top RL agents are far worse at matching human moves than even just imitation learning. We tested the current top open-source Go program KataGo (Wu, 2020) using its final 6, 10, and 15 block models[5] which range from upper-amateur to superhuman level, and found that they achieve accuracies of about 35%, 43%, and 43%, all more than 10% lower than the results in Table 1.

Lastly, in Table 5 we show that in chess the amount of improvement given by MCTS over the raw model in predicting human players on the Lichess test set tends to be larger for games with longer time controls than for shorter time controls, going from about 0.2% to about 0.7% between the shortest and longer time controls for 1500-1599-rated players, and going from about 0.8% to around 2.0% for 1900-1999-rated players. This is consistent with the intuition that humans rely more heavily on planning when they have more time to think, increasing the gain from modeling that planning.

[5] from https://katagotraining.org

F BASELINE MODEL ARCHITECTURE AND TRAINING FOR GO

As summarized in Section 3.2, for training baseline imitation-learning models to play on the 19x19 board in Go, our architecture follows the same 20-block 256-channel residual net described in Silver et al. (2017), except with the addition of squeeze-and-excitation layers at the end of each residual block (Hu et al., 2018). In particular, the following additional operations are inserted just prior to each skip connection that adds the output R of a residual block to the trunk X:

- Channelwise global average pooling of R from $19 \times 19 \times 256$ channels to 256 channels.
- A fully connected layer including bias from 256 channels to 64 channels.
- A ReLU nonlinearity.
- A fully connected layer including bias from 64 channels to 512 channels, which are split two vectors S and B of 256 channels each.
- Output $R \times \text{sigmoid}(S) + B$ to be added back to the trunk X, instead of R as in a normal residual net.

In other words, the final result of the residual block as a whole is $\text{ReLU}(X + R \times \text{sigmoid}(S) + B)$ instead of $\text{ReLU}(X + R)$.

Additionally, some games are played with a *komi* (compensation given to White for playing second) that is not equal to the value of 7.5 used by Silver et al. (2017). Therefore, for the final feature of the input encoding, rather than a binary-valued feature equalling 1 if the player to move is White and 0 if the player to move is Black, we instead use the real-valued feature of komi/10 if the player to move is White or -komi/10 if the player to move is Black. We additionally exclude a very tiny number of games with extreme komi values, outside of the range [-60,60].

We train using a mini-batch size of 2048 distributed as 8 batches of 256 across 8 GPUs, and train for a total of 64 epochs - roughly 475000 minibatches for the GoGoD dataset. We use SGD with momentum 0.9, weight decay coefficient of 1e-4, and a learning rate schedule of of 1e-1, 1e-2, 1e-3, 1e-4 for the first 16, next 16, next 16, and last 16 epochs respectively.

We train both the policy and values heads jointly, minimizing the cross entropy of the policy head with respect to the one-hot move made in the actual game, and the MSE of the value head with respect to the game result of -1 or 1, except similarly to Silver et al. (2017) we weight the MSE value loss by 0.01 to avoid overfitting of the value head.

G MCTS ALGORITHMIC DETAILS

In this appendix, we summarize the details of the version of MCTS used in our experiments, including one often-overlooked detail. We follow a standard MCTS implementation very similar to that of Silver et al. (2017).

Each turn, the algorithm builds and expands a game tree over multiple iterations rooted at the current state for that turn. On each iteration t MCTS starts at the root and descends the tree by exploring at each state s an action a according to some exploration method. Upon reaching a state s_t not yet explored, it adds s_t to the tree, queries the value function $V_i(s_t)$ for each i to estimate the total expected future reward, and updates the statistics of all nodes traversed based on $V_i(s_t)$ and any intermediate rewards received. Subsequent iterations begin again from the root. For our work in chess and Go, we follow the convention where win, loss, and draw have reward 1,-1, and 0.

The statistics tracked at the node for each state s where player i is to move include the visit counts $N(s, a)$ which are the number of iterations that reached state s and tried action a, and $Q(s, a)$ the average value of those iterations from the perspective of player i, i.e. $Q(s, a) = (1/N(s, a)) \sum_t V_i(s_t) + U_i(s, s_t)$ where the sum ranges only over those iterations t that reached state s and tried action a and $U_i(s, s_t)$ is the total intermediate reward on the path from s to s_t. When descending the tree, the exploration method is to always select the action:

$$\arg\max_a \; Q(s, a) + c_{\text{puct}} \tau(s, a) \frac{\sqrt{\sum_b N(s, b)}}{N(s, a) + 1} \tag{28}$$

where $\tau(s, a)$ is the prior policy probability for action a in state s, and c_{puct} is a tunable parameter controlling the tradeoff between exploration and exploitation. The final agent policy π is simply proportional to the visit counts for the root, i.e. $\pi(s, a) = N(s, a)/\sum_b N(s, b)$ where s is the root state, or optionally we may also have $\pi(s, a) \sim N(s, a)^{1/T}$ where T is a temperature parameter.

One final often-overlooked detail concerns how to evaluate states for which there is no $Q(s, a)$ estimate. Since the tree policy depends on the $Q(s, a)$ estimates of the possible actions, there is a nontrivial choice of what Q value to use for an action a that has been tried *zero* times and therefore never estimated. Since the tree branches exponentially, deeper in the MCTS tree there will always be many actions with zero visits, and so this choice can affect the behavior of MCTS even in the limit of large amounts of

search. Unfortunately, the details of this choice have sometimes been left undiscussed and undocumented in major past work, and as a result major MCTS implementations have not standardized on it, variously choosing game-loss (Lai, 2018), the current running average parent Q or a Q-value minus a heuristic offset parameter (Tian, 2019), or many other options.

In our work, we use the equal-weighted average value of all actions at the parent node that have been visited at least once, i.e. $\sum_a Q(s,a)I(N(s,a)>0)/\sum_a I(N(s,a)>0)$. This can be viewed as corresponding to a naive prior that the values of actions are i.i.d draws from an unknown distribution. While not perfect, our choice is simple, parameter-free, behaves in a way that is invariant to any global translation or scaling of the game's rewards, and works reasonably in practice for our purposes.

H BRIEF DESCRIPTION OF DIPLOMACY

We briefly summarizing the rules of Diplomacy. See Paquette et al. (2019) for a more detailed description. The board is a map of Europe partitioned into 75 regions, 34 of which are *supply centers* (SCs) that players compete to control. Players command multiple units and each turn privately issue orders for each unit they own (to hold, move, support another unit, or convoy). These orders are revealed at the same time, thereby making Diplomacy a simultaneous-action game. A player wins the game by controlling a majority (18) of the SCs. A game may also end in a draw if all remaining players agree. In this case, we use the **Sum-of-Squares (SoS)** scoring system as used in prior works Paquette et al. (2019); Gray et al. (2020); Bakhtin et al. (2021). If no player wins, SoS defines the score of player i as $C_i^2/\sum_{i'} C_{i'}^2$, where C_i is the SC count for player i.

Diplomacy is specifically designed so that a player is unlikely to achieve victory without help from other players. The full game allows unrestricted private natural-language communication between players each turn prior to choosing orders, but we focus on the simpler *no-press* variant, in which no such communication is allowed, but modeling how opponents will behave continues to be important.

I DIPLOMACY HYPER-PARAMETERS

We describe the describe the search parameters used in this work and compare it to those in previous works. As compared to Gray et al. (2020), we use a much less expensive set of search parameters for all results in the main section of this paper (See, Table 7). In Table 8, we show that these tuned-down set of parameters, slightly reduces piKL-HedgeBot's performance but allows us to make similar conclusions when comparing against SearchBot Gray et al. (2020) and supervised learning-based bots from prior works. Additionally, in Table 8, we also show that when our agent (piKL-HedgeBot ($\lambda = 10^{-3}$)) uses the same search parameters as Gray et al. (2020), it outperforms SearchBot by a big margin.

In our experiments, to compare against DipNet (Paquette et al., 2019) we use the original model checkpoint[6] and we sample from the policy with temperature 0.1 (as used in prior works). Similarly, to compare against SearchBot (Gray et al., 2020) agent we use the released checkpoint[7] and agent configuration[8].

The only other search parameter unique and new to our piKL-HedgeBot algorithm is η in Algorithm 1, which we heuristically set on each Hedge iteration t to $c/(\sigma\sqrt{t})$ where σ is the standard deviation across iterations of the average utility experienced by the agent i being updated. We find $c = 10/3$ works well for no-press Diplomacy.

Model	**Temperature**
DipNet (Paquette et al., 2019)	0.1
DipNet RL (Paquette et al., 2019)	0.1
Blueprint (Gray et al., 2020)	0.1
IL Policy (Ours)	0.5

Table 6: Sampling temperatures used in the models across prior works. For our imitation learning model (IL Policy), we use a temperature of 0.5 in all experiments to encourage stochasticity.

J DIPLOMACY MODEL ARCHITECTURE AND INPUT FEATURES

Our imitation learning policy model for Diplomacy uses the same transformer-encoder LSTM-decoder architecture as Bakhtin et al. (2021) for reinforcement learning in Diplomacy, but applied to imitation learning over human games. This architecture also

[6]DipNet SL from `https://github.com/diplomacy/research`.

[7]blueprint from `https://github.com/facebookresearch/diplomacy_searchbot/releases/tag/1.0`.

[8]`https://github.com/facebookresearch/diplomacy_searchbot/blob/master/conf/common/agents/searchbot_02_fastbot.prototxt`

Parameter	Gray et al. (2020)	Ours
Number candidate actions (N_c)	50	30
Max candidate actions per unit	3.5	3.5
Number search iterations	256	512
Policy sampling temperature for rollouts	0.75	N/A
Policy sampling top-p	0.95	0.95
Rollout length, move phases	2	0

Table 7: Search parameters used in Gray et al. (2020) compared to the search parameters used in our work. All experiments in the main body of this work uses search settings that are much cheaper to run. We use a rollout length of 0 and $N_c = 30$ while increasing the number of search iterations to 512.

1x ↓ 6x →	DipNet	DipNet RL	Blueprint	BRBot	SearchBot
DipNet (Paquette et al., 2019)	-	6.7% ± 0.9%	11.6% ± 0.1%	0.1% ± 0.1%	0.7% ± 0.2%
DipNet RL (Paquette et al., 2019)	18.9% ± 1.4%	-	10.5% ± 1.1%	0.1% ± 0.1%	0.6% ± 0.2%
Blueprint (Gray et al., 2020)	20.2% ± 1.3%	7.5% ± 1.0%	-	0.3% ± 0.1%	0.9% ± 0.2%
BRBot (Gray et al., 2020)	67.3% ± 1.0%	43.7% ± 1.0%	69.3% ± 1.7%	-	11.1% ± 1.1%
SearchBot (Gray et al., 2020)	51.1% ± 1.9%	35.2% ± 1.8%	52.7% ± 1.3%	17.2% ± 1.3%	-
piKL-HedgeBot ($\lambda = 0.001$)	54.8% ± 1.8%	31.4% ± 1.8%	50.3% ± 1.8%	19.2% ± 1.4%	16.6% ± 1.3%
piKL-HedgeBot ($\lambda = 0.001$) (Gray et al. (2020) parameters)	60.1% ± 1.8%	33.3% ± 1.8%	58.1% ± 1.8%	23.6% ± 1.6%	20.3% ± 1.4%

Table 8: Average SoS scores achieved by the 1x agent against the 6x agents. This table compares the performance of SearchBot Gray et al. (2020) and other agents from prior work with piKL-HedgeBot ($\lambda = 10^{-3}$) that uses a much cheaper search setting. Using the much cheaper search setting, comes at relatively small cost in its performance as we use improved value and policy models (See, Appendix J and Appendix I for more details). When using the same parameters as Gray et al. (2020), piKL-HedgeBot ($\lambda = 10^{-3}$) significantly outperforms SearchBot under most settings. Note that equal performance would be 1/7 $\approx$ 14.3%. The $\pm$ shows one standard error.

resembles the architecture used by a significant amount of past work Gray et al. (2020); Anthony et al. (2020); Paquette et al. (2019) but replaces the graph-convolution encoder with a transformer, which we find to produce good results.

Additionally, we slightly modify the input feature encoding relative to Gray et al. (2020), removing a small number of redundant channels and adding channels to indicate the "home centers" of each of the 7 powers (Austria, England, France,... etc), which are the locations where that power is allowed to build new armies or fleets. See Table 9 for the new list of input features. By adding the home centers to the input encoding instead of leaving them implicit, the game becomes entirely equivariant to permutations of those powers - e.g. if one swaps all the units of England and France, and all their centers, and which centers are their home centers, the resulting game is isomorphic to the original except with the two powers renamed.

This allows us to then augment the training data via equivariant permutations of the seven possible powers in the encoding. Every time we sample a position from the dataset for training, we also choose among all 7-factorial permutations of the powers uniformly at random, and correspondingly permute both the input and output, to reduce overfit and improve the model's generalization given the limited human data available.

Feature	Type	Number of Channels
Location has unit?	One-hot (army/fleet), or all zero	2
Owner of unit	One-hot (7 powers), or all zero	7
Buildable, Removable?	Binary	2
Location has dislodged unit?	One-hot (army/fleet), or all zero	2
Owner of dislodged unit	One-hot (7 powers), or all zero	7
Area type	One-hot (land,coast,water)	3
Supply center owner	One-hot (7 powers or neutral), or all zero	8
Home center	One-hot (7 powers), or all zero	7

Table 9: Per-location input features used

K IMPROVED VALUE MODEL IN DIPLOMACY

For no-press Diplomacy, we note that Gray et al. (2020) observed that their search agent benefits from short rollouts using the trained human policy before applying the human-learned value model to evaluate the position. Doing so appears to result in more accurate evaluations reflecting the likely outcomes from a given game state, which the raw value model may failed to learn sufficiently accurately on the limited human dataset. Since expectation of the learned value model after a short rollout appears to be better than the learned value model itself, this motivates training a model to directly approximate the former.

In a fashion broadly similar to Silver et al. (2016) generating rollout games to train a more accurate value head for Go, we therefore generated a large stream of data by uniformly sampling positions from the human game dataset for Diplomacy, rolling them forward between 4-8 phases of game play via the same rollout settings as Gray et al. (2020), i.e. policy sampling temperature 0.75, top-p 0.95, and training a new value model to predict the resulting post-rollout value estimate of the old value model. Samples were continuously and asynchronously added to a replay buffer of 10000 batches, and the buffer was continuously sampled to train the same transformer-based architecture as the human-trained model from Appendix J initialized with the weights of that model. Training was constrained to never exceed the rate of data generation by more than a factor of 2 (i.e. using each sample twice in expectation) and proceeded for 128000 mini-batches of 1024 samples each using the ADAM optimizer with a fixed learning rate of 1e-5.

L MORE EXPERIMENTS IN DIPLOMACY

This section compiles additional performance results from evaluation games.

L.1 HEAD-TO-HEAD PERFORMANCE

In Table 10, we compare the performance of piKL-HedgeBotin 1v6 head-to-head games against the underlying imitation anchor policy, following prior work Gray et al. (2020); Bakhtin et al. (2021); Paquette et al. (2019); Anthony et al. (2020). We find that the $\lambda = 10^{-1}$ policy is substantially stronger than the imitation policy while matching the accuracy in predicting human moves, while the $\lambda = 10^{-3}$ policy outperforms unregularized search methods while playing much closer to the human policy.

1x	6x	**Average SoS Score**
IL Policy	piKL-HedgeBot ($\lambda = 10^{-1}$)	8.3±0.9%
	piKL-HedgeBot ($\lambda = 10^{-2}$)	2.5±0.4%
	piKL-HedgeBot ($\lambda = 10^{-3}$)	1.8±0.3%
	piKL-HedgeBot ($\lambda = 10^{-4}$)	2.1±0.3%
	piKL-HedgeBot ($\lambda = 10^{-5}$)	1.6±0.2%
	HedgeBot	1.5±0.2%
	RMBot	1.4±0.2%

1x	**6x**	**Average SoS Score**
IL Policy	piKL-HedgeBot ($\lambda = 10^{-1}$)	21.1±1.4%
	piKL-HedgeBot ($\lambda = 10^{-2}$)	44.2±1.7%
	piKL-HedgeBot ($\lambda = 10^{-3}$)	52.7±1.7%
	piKL-HedgeBot ($\lambda = 10^{-4}$)	49.7±1.7%
	piKL-HedgeBot ($\lambda = 10^{-5}$)	46.9±1.7%
	HedgeBot	46.5±1.7%
	RMBot	46.2±1.7%

Table 10: Average SoS score attained by the 1x agent against the 6x agent. piKL-HedgeBot($\lambda = 10^{-1}$) policy is substantially stronger than IL Policy, while the ($\lambda = 10^{-2}$) policy is almost as strong as RMBot. The $\pm$ shows one standard error. Note that equal performance would be 1/7 $\approx$ 14.3%.

L.2 piKL-HEDGEBOT'S PERFORMANCE IN POPULATION-BASED EXPERIMENTS

In this section, we provide all the results from the population experiments across various piKL-HedgeBot's lambda values. Figure 6 and Table 11 show the results. piKL-HedgeBot with $\lambda = 10^{-3}$ performs best across individual population experiments with an SoS score of 32.9%. The performance drops as we continue to increase λ past $1e-3$. Error bars indicate 1 standard error.

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	4.9% ± 0.3%
DipNet RL (Paquette et al., 2019)	5.6% ± 0.4%
Blueprint (Gray et al., 2020)	7.1% ± 0.4%
BRBot (Gray et al., 2020)	18.2% ± 0.6%
SearchBot (Gray et al., 2020)	36.1% ± 0.8%
IL Policy	10.2% ± 0.6%
RMBot	36.8% ± 1.1%
piKL-HedgeBot ($\lambda = 10^{-1}$)	15.6% ± 0.6%

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	3.8% ± 0.3%
DipNet RL (Paquette et al., 2019)	4.2% ± 0.3%
Blueprint (Gray et al., 2020)	5.8% ± 0.4%
BRBot (Gray et al., 2020)	16.3% ± 0.6%
SearchBot (Gray et al., 2020)	14.1% ± 0.6%
IL Policy	8.5% ± 0.4%
RMBot	31.7% ± 0.8%
piKL-HedgeBot ($\lambda = 10^{-2}$)	29.9% ± 0.7%

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	3.7% ± 0.3%
DipNet RL (Paquette et al., 2019)	4.7% ± 0.3%
Blueprint (Gray et al., 2020)	4.9% ± 0.3%
BRBot (Gray et al., 2020)	16.1% ± 0.6%
SearchBot (Gray et al., 2020)	13.4% ± 0.5%
IL Policy	7.9% ± 0.4%
RMBot	31.3% ± 0.7%
piKL-HedgeBot ($\lambda = 10^{-3}$)	32.9% ± 0.7%

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	3.5% ± 0.3%
DipNet RL (Paquette et al., 2019)	4.6% ± 0.3%
Blueprint (Gray et al., 2020)	5.7% ± 0.3%
BRBot (Gray et al., 2020)	14.3% ± 0.6%
SearchBot (Gray et al., 2020)	13.6% ± 0.5%
IL Policy	8.8% ± 0.4%
RMBot	31.8% ± 0.7%
piKL-HedgeBot ($\lambda = 10^{-4}$)	31.8% ± 0.7%

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	3.6% ± 0.3%
DipNet RL (Paquette et al., 2019)	4.4% ± 0.3%
Blueprint (Gray et al., 2020)	5.3% ± 0.3%
BRBot (Gray et al., 2020)	15.0% ± 0.6%
SearchBot (Gray et al., 2020)	13.0% ± 0.5%
IL Policy	8.9% ± 0.4%
RMBot	32.2% ± 0.7%
piKL-HedgeBot ($\lambda = 10^{-5}$)	31.9% ± 0.7%

Agent	Average SoS Score
DipNet (Paquette et al., 2019)	3.6% ± 0.3%
DipNet RL (Paquette et al., 2019)	3.9% ± 0.3%
Blueprint (Gray et al., 2020)	5.5% ± 0.3%
BRBot (Gray et al., 2020)	14.7% ± 0.6%
SearchBot (Gray et al., 2020)	14.1% ± 0.6%
IL Policy	8.7% ± 0.4%
RMBot	32.2% ± 0.7%
HedgeBot	31.7% ± 0.7%

Table 11: Average SoS score achieved by agents in uniformly sampled pools of other agents. piKL-HedgeBot with $\lambda = 10^{-3}$ performs best across individual sweeps with an SoS score of 32.9%. The ± shows one standard error.

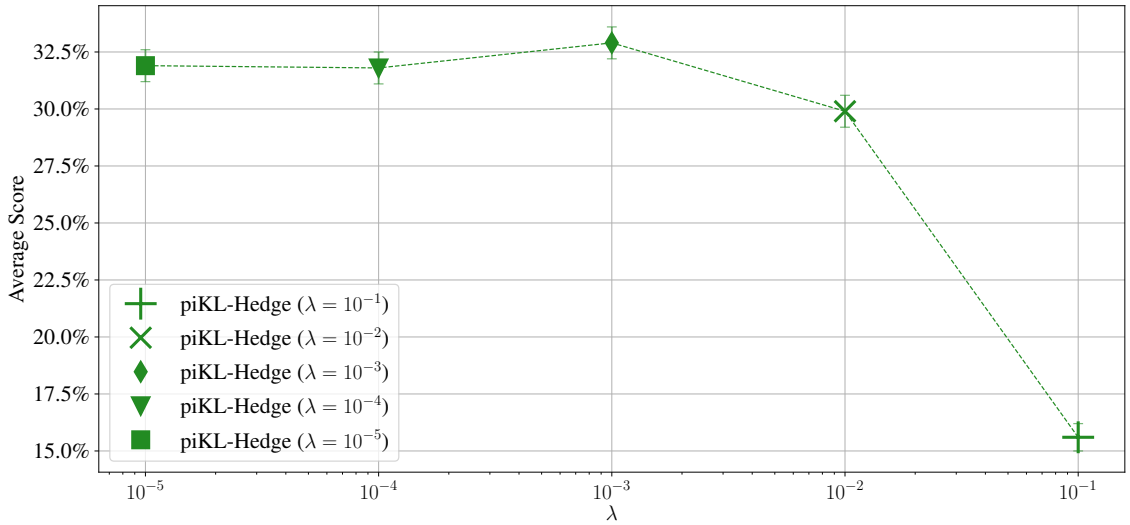

Figure 6: Average SoS score achieved by piKL-HedgeBots in uniformly sampled pools of other agents as a function of λ. piKL-HedgeBot ($\lambda = 10^{-3}$) performs best across individual sweeps with an SoS score of 32.9%.

M DEC-POMDP GAMES: POLICY-REGULARIZED SPARTA ON HANABI

In this section, we extend KL-regularized search to decentralized partially observable Markov decision processes (Dec-POMDP) and test it on the Hanabi benchmark Bard et al. (2020). We first train an imitation learning policy (IL policy) from human data. We then use the IL policy as the blueprint policy in SPARTA Lerer et al. (2020), a search technique for Dec-POMDPs, and apply KL-regularization toward the IL policy in SPARTA. We call this new algorithm piKL-SPARTA. We show that piKL-SPARTA matches or even slightly improves the original IL policy in human move prediction accuracy while greatly improving self-play performance.

M.1 BACKGROUND

A Dec-POMDP is an N-player fully cooperative game with state space S that is partially observed by each player i through their individual observation function $o_i = \Omega_i(s)$ for $s \in S$, with joint action space $A = A_1 \times A_2 \times \cdots \times A_N$ and transition function $\mathcal{T}\colon S \times A \to P(S)$ that returns the distribution of next state given current state and joint action. The reward function $R\colon S \times A \to \mathbb{R}$ assigns a scalar reward for the *entire* team at each time step. A trajectory is denoted as $\tau^t = (s^0, \mathbf{a}^0, r^0, \ldots, s^t)$ while the action-observation history (AOH) of each player is defined as $\tau_i^t = (o^0, \mathbf{a}^0, r^0, \ldots, o^t)$. A full trajectory or full AOH that reaches the terminal state may be denoted more simply as τ or τ_i respectively. The policy for each individual player $\pi_i(a_i^t|\tau_i^t)$ takes as input the AOH and returns a distribution over valid actions. The joint policy $\pi = (\pi_1, \ldots, \pi_N)$ is a tuple containing all players' policies. The goal is to find a policy to maximize the expected total return $\pi^* = \arg\max_\pi J(\pi) = \mathbb{E}_{\tau\sim P(\tau|\pi)} R^0(\tau)$ where $R^t(\tau) = \sum_{t'\geq t} \gamma^{(t'-t)} r'_t$ is the forward looking return with optional discount factor $\gamma \leq 1$.

Hanabi is a well-established large-scale Dec-POMDP benchmark Bard et al. (2020). It is a 2 to 5 player card game with a deck of 50 cards equally divided into 5 color suits. Each color consists of five ranks with three 1s, two 2s, two 3s, two 4s, and one 5. Each player draws five cards from a randomly shuffled deck to start the game. The goal of the team is to play cards in order of increasing rank from 1 to 5 for every color suit. Players take turns to either play a card, discard a card, or give a hint to another player about their cards. When giving a hint, the acting player picks a receiver of the hint and a rank or color of any card in the receiver's hand. The recipient will then learn exactly which cards in their hand match the given rank or color. Hinting costs one information token. The team starts with eight information tokens and they can recoup one information token after discarding a card or successfully playing a 5 of any color. If a player makes an invalid play, e.g. playing a red 3 when a red 2 is not played yet, the team loses one life token. After each play or a discard, a player draws a new card if possible. The game ends when three life tokens are lost, in which case the team receives 0 points, or one round after the entire deck is exhausted, in which case the score equals the number of cards successfully played.

The majority of existing works in the Hanabi domain focus on learning human compatible policies without using human data Bard et al. (2020); Siu et al. (2021); Hu et al. (2021b). Fewer works have explored better modeling human policies using human game data, partly due to the lack of publicly available datasets. To the best of our knowledge, the strongest supervised learning agent trained from human data was done by Hu et al. (2021b) where the authors use it as an unseen test-time partner agent in evaluation to estimate how well their agents might collaborate with humans. No prior work has been done to better predict human moves in Hanabi.

A few search techniques such as SPARTA Lerer et al. (2020) and RL-search Fickinger et al. (2021) have been proposed for large scale Dec-POMDPs and have specifically been applied in Hanabi. In this work we choose SPARTA as our backbone for simplicity, while noting that our KL-regularization methods may also be generalized to other search techniques. SPARTA is a test-time policy improvement algorithm that can be applied on top of any policy. SPARTA assumes that a **blueprint** policy (BP) π is common knowledge in the Dec-POMDP and all players play their part of the blueprint unless they should deviate according to the SPARTA rule. Here we briefly discuss the single-agent variant where the search agent assumes that other agents will always play the blueprint. Given blueprint π, SPARTA first defines a belief function that tracks the distribution of the real trajectory given the search agent's own AOH $\mathcal{B}_i(\tau^t) = P(\tau^t|\tau_i^t, \pi)$. Then the search agent i computes the expected value for each action a using Monte Carlo rollouts:

$$Q_\pi(\tau_i^t, a) = \mathbb{E}_{\tau^t\sim\mathcal{B}_i(\tau^t)} Q_\pi(\tau^t, a), \tag{29}$$

where:

$$Q_\pi(\tau^t, a) = \mathbb{E}_{\tau\sim P(\tau|\mathcal{T},\tau^t,a_i^t=a,a_{j\neq i}^t\sim\pi,\mathbf{a}^{t'>t}\sim\pi)} R^t(\tau)$$

is the expected forward looking return on τ^t assuming that the search agent will perform the action a for the current step and follow the blueprint afterwards while other agents always follow the blueprint. SPARTA computes the belief $\mathcal{B}_i(\tau_t)$ analytically. It maintains a distribution of all possible trajectories and adjusts it at every step by removing the trajectories that contradict public knowledge or would have led to different joint actions according to the known joint policy π.

M.2 METHOD

We start with an imitation learning policy (IL policy) trained from human gameplay data collected from an online Hanabi game platform. The IL policy is used as both the baseline for predicting human moves as well as the blueprint policy for our piKL-SPARTA. The analytical belief update procedure in SPARTA requires full knowledge of the partners' policy. This is challenging when predicting human moves and playing with humans because our IL policy models the average behavior of the entire population of players in the training dataset and a player at test time may perform actions that the IL policy will never do, leading to a null belief space that will terminate the search. Therefore, we follow the practice in Hu et al. (2021a) and train an approximate neural network belief model on self-play data generated by the IL policy.

Given the IL policy π as blueprint and the approximate belief model $\hat{\mathcal{B}}_i$ that replaces the $\mathcal{B}_i$ in Eq. 29, piKL-SPARTA selects actions for the search player i following

$$P(a) \propto \pi(a|\tau_i^t) \cdot \exp\left[\frac{Q_\pi(\tau_i^t, a)}{\lambda}\right]. \tag{30}$$

M.3 EXPERIMENTAL SETUP

We use a similar dataset acquired from `en.boardgamearena.com` as in Hu et al. (2021b). The dataset consists of 240,954 2-player Hanabi games. We randomly sample 1,000 games to create a validation set and another 4,000 games for the test set. The training set contains the remaining 235,954 games with an average score of 15.88. Each game records the AOH $\tau_i, i \in \{1, 2\}$ for both players. The IL policy π_θ is parameterized by a neural network θ and is trained to minimize the cross-entropy loss

$$\mathcal{L}(\theta) = -\mathbb{E}_{\tau_i \sim D} \sum_{t=0}^{T} \pi_\theta(a_i^t|\tau_i^t)$$

with stochastic gradient descent. The training set D is dynamically augmented with color shuffling Hu et al. (2020) where a random color permutation that changes both observation and action space is applied to each trajectory τ_i sampled from the dataset before feeding it to the network. Hu et al. (2021b) shows that this data augmentation method greatly reduces overfitting and leads to better policies. The network θ uses the Public-LSTM structure in Hu et al. (2021a), which eliminates the need to re-unroll LSTM on sampled trajectories from the beginning of the game as they share the same public observations as the real trajectory.

To construct the approximate belief model $\hat{\mathcal{B}}$, we train a neural network ϕ to predict each player's own hand, which is the only hidden information in Hanabi. The network predicts each card in hand from oldest to newest auto-regressively. We denote the hidden cards as $\{h_i^j\}$ with h_i^1 being the oldest and h_i^m being the newest, $m \leq 5$. Then the cross-entropy loss for ϕ in an N-player setting becomes

$$\mathcal{L}(\phi) = -\mathbb{E}_{\tau \sim \pi_\theta}\left[\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{j=1}^{m} \log P_\phi(h_i^j|\tau_i^t, h_i^1, \ldots, h_i^{j-1})\right].$$

In Hanabi, it is sufficient to reconstruct τ^t given τ_i^t and $\{h_j^i\}$. The model is trained on infinite stream of data generated by π_θ through self-play to avoid overfitting. We train two variants of the belief model, one with sampled action $\mathbf{a} \sim \pi_\theta$ while the other with greedy action $\mathbf{a} = \arg\max \pi_\theta$. They are used by piKL-SPARTA and piKL-SPARTA-G(reedy) respectively.

We first run piKL-SPARTA on the test set to compare its ability to predict human moves against the IL policy. For each τ_i^t in the test set, we sample $\frac{K}{|A_i|}$ hands from the belief model P_ϕ where K is the total number of searches for this step and $|A_i|$ is the number of legal actions of the search player. In all our experiments we set $K = 10,000$. In practice, we sample $\frac{2K}{|A_i|}$ hands and take the top $\frac{K}{|A_i|}$ samples not contradicting the public knowledge. If the belief model fails to produce any samples that comply with the public knowledge, we revert back to the blueprint. We then compute the expected value for each search action by unrolling the IL policy until the end of the game for each sampled trajectory and compare $a_{\text{pred}} = \arg\max_a \pi_\theta(a|\tau_i^t) \cdot \exp[\frac{Q_\pi(\tau_i^t, a)}{\lambda}]$ against the human move a_i^t from the data. We experiment with different λ to study its effect on prediction accuracy.

We also evaluate piKL-SPARTA in self-play to compare its performance under different λ against the IL policy. We run piKL-SPARTA and the IL policy on 4,000 games with different seeds for the deck. At each step, the IL policy acts following $a_t \sim \pi_\theta(a|\tau_i^t)$ while piKL-SPARTA acts following Eq. (30) again using $K = 10,000$ rollouts.

The λ in these experiments are significantly higher than those in Diplomacy or implicitly in MCTS in Chess and Go[9] because λ represents the scale of utility difference that offsets a particular KL penalty, and the range of the utilities $Q_\pi(\tau_i^t, a)$ in Hanabi is $[-25, 25]$ whereas the range in other games are either $[0, 1]$ or $[-1, 1]$.

[9] Via the relationship $\lambda \approx c_{\text{puct}}\sqrt{N}$ where N is the number of MCTS iterations

Subset of Test Set	λ =0	λ =0.5	λ =1	λ =2	IL Policy
All games	25.30% $\pm$ 0.16%	56.04% $\pm$ 0.24%	60.22% $\pm$ 0.21%	62.44% $\pm$ 0.19%	63.63% $\pm$ 0.18%
Games w/score $\geq$ 10	27.01% $\pm$ 0.18%	58.86% $\pm$ 0.25%	62.62% $\pm$ 0.22%	64.51% $\pm$ 0.20%	65.29% $\pm$ 0.19%
Games w/score $\geq$ 20	28.87% $\pm$ 0.19%	61.86% $\pm$ 0.25%	65.21% $\pm$ 0.23%	66.81% $\pm$ 0.21%	67.39% $\pm$ 0.19%
Subset of Test Set	**λ =5**	**λ =10**	**λ =20**	**λ =50**	**IL Policy**
All games	63.50% $\pm$ 0.18%	63.68% $\pm$ 0.18%	**63.71% $\pm$ 0.18%**	63.68% $\pm$ 0.18%	63.63% $\pm$ 0.18%
Games w/score $\geq$ 10	65.33% $\pm$ 0.19%	**65.43% $\pm$ 0.19%**	65.42% $\pm$ 0.19%	65.35% $\pm$ 0.19%	65.29% $\pm$ 0.19%
Games w/score $\geq$ 20	67.48% $\pm$ 0.19%	**67.54% $\pm$ 0.19%**	67.53% $\pm$ 0.19%	67.45% $\pm$ 0.19%	67.39% $\pm$ 0.19%

Table 12: Human prediction accuracy of unregularized SPARTA ($\lambda = 0$) and of piKL-SPARTA with different λ and the IL policy on test set. Each row represents their accuracy on a subset filtered by the final score of the games. pikl-SPARTA achieves similar prediction accuracy as IL for most λ and is far more accurate than unregularized SPARTA.

We experiment with both 1p piKL-SPARTA, where one player uses piKL-SPARTA and the other follows the blueprint, and 2p piKL-SPARTA, where both players run the same single-agent version of piKL-SPARTA independently. The latter is not theoretically sound and 2p SPARTA has in past papers produced worse performance than 1p SPARTA Lerer et al. (2020) because 1p SPARTA assumes that partners are playing according to the blueprint policy while in actuality the partners are playing according to a SPARTA policy. Despite the lack of theoretical soundness, we are interested in whether 2p piKL-SPARTA can obtain empirical improvement over the 1p version, since piKL-SPARTA regularizes towards the blueprint and therefore the mismatch between assuming the partner follows the blueprint and the policy they actually play would likely be less severe.

Lastly, in Dec-POMDPs, it is common to select actions greedily in self-play instead of sampling from a mixed policy, since in fully cooperative settings there is no need to avoid being deterministic or predictable to an adversary. Therefore, we also experimented with piKL-SPARTA-G(reedy) where every agent, including the baseline IL policy, play according to the $\arg\max$ of their action distributions at every step, and the belief model for piKL-SPARTA-G is trained on trajectories produced by a greedy IL policy.

M.4 Results

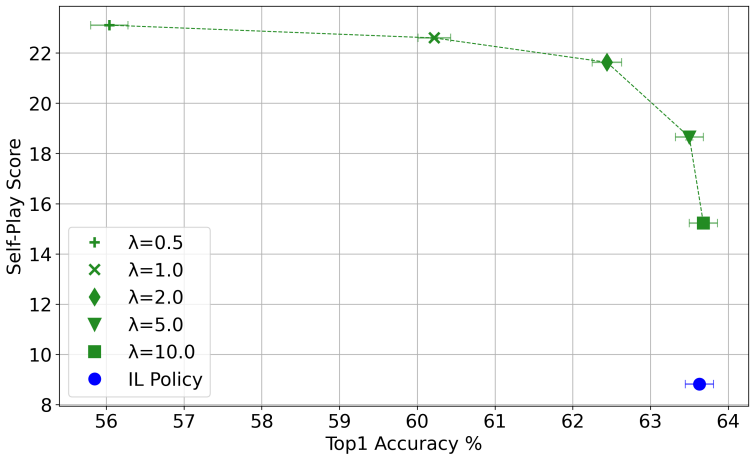

Figure 7: Top-1 test accuracy and self-play score of IL policy and piKL-SPARTA in Hanabi. Blue dot is the IL policy and green dots are piKL-SPARTA with different λ. The self-play score is evaluated with sampling based 2p piKL-SPARTA and sampling based SL policy. Additional evaluations with more λ and more algorithm variants are presented in Table 12 and Table 13. Error bar is 1 standard error.

Table 12 summarizes the results for human move prediction. On the full test set, piKL-SPARTA with $\lambda = 10$ and $\lambda = 20$ outperform the IL policy slightly, although not statistically confidently, and vastly outperform unregularized SPARTA. We also investigate the prediction accuracy on games with score ≥ 10 or ≥ 20 that more predominantly come from more experienced human players. The improvement of piKL-SPARTA over the IL policy may be larger on these games, although this result is noisy and at best should be considered only mildly suggestive since filtering on final score also introduces other major confounding factors as well.

	λ =0	λ =0.5	λ =1	λ =2	λ =5	λ =10	IL Policy
1p piKL-SPARTA	20.56 ± 0.05	21.16 ± 0.06	20.02 ± 0.09	17.71 ± 0.12	13.53 ± 0.16	11.04 ± 0.17	8.81 ± 0.15
2p piKL-SPARTA	20.23 ± 0.04	**23.11** $\pm$ **0.03**	22.60 ± 0.03	21.63 ± 0.05	18.65 ± 0.11	15.23 ± 0.15	
1p piKL-SPARTA-G	22.78 ± 0.03	23.07 ± 0.03	22.76 ± 0.03	22.41 ± 0.04	21.91 ± 0.05	21.45 ± 0.06	19.72 ± 0.10
2p piKL-SPARTA-G	19.98 ± 0.04	**23.72** $\pm$ **0.02**	23.39 ± 0.03	22.93 ± 0.03	22.36 ± 0.03	21.98 ± 0.04	

Table 13: Performance of unregularized SPARTA ($\lambda = 0$) and piKL-SPARTA under different λ and IL policy evaluated on 4,000 self-play games, reported $\pm$ one standard error. In the top two rows, both piKL-SPARTA and IL policy sample actions according to their action distribution respectively. In the bottom two rows, both algorithms take greedy actions; "-G" is short for "-Greedy". All numbers shown in each table section use the same learned belief model for fair comparison. At the high lambdas that maintain or improve human accuracy, piKL-SPARTA significantly improves playing strength over IL, while at lower lambda values piKL-SPARTA outscores unregularized SPARTA. For reference, unregularized greedy 1p SPARTA with exact beliefs rather than learned beliefs gets 23.49 ± 0.02.

In Table 13, we show the self-play performance piKL-SPARTA and IL policy. The top two rows show the results of the sampling version while the bottom two rows show those of the greedy version. The conclusions are consistent across both cases. Both 1p and 2p variants of piKL-SPARTA outscore IL for all λ tested. Together with Table 12 we find with $\lambda = 10$, piKL-SPARTA maintains IL prediction accuracy and outscores IL greatly in self-play, an overall improvement without any tradeoff. For smaller $\lambda = 2$ or $\lambda = 5$, piKL-SPARTA further outscores IL while losing some prediction accuracy on human moves, but remains vastly more accurate than unregularized SPARTA ($\lambda = 0$).

Through qualitative analysis of the games played by both methods, we find that IL makes many mistakes due to both sampling low probability actions and the fact that the training set contains bad moves. piKL-SPARTA, on the other hand, avoids many of these mistakes while still otherwise following the same strategies and humanlike signaling conventions. It is particularly good at preventing catastrophic failures where the agents lose all life tokens and points since the Q values for good and bad actions differ much more widely in those cases.

The 2p piKL-SPARTA variants, despite being theoretically unsound, result in a further score improvement. Meanwhile, we notice that 2p versions of the plain SPARTA ($\lambda = 0$) underperform their 1p variants, which is consistent with the observations from Lerer et al. (2020) despite us using an approximate learned belief model instead of exact belief. This suggests that regularization towards the blueprint IL policy is successful in keeping the trajectories similar enough to those from the blueprint that the unsound assumption that the partner plays the blueprint does not cause major problems, while still allowing both players to deviate enough to correct major blunders that they would otherwise make.

Lastly, we notice that 1p piKL-SPARTA with $\lambda = 0.5$ outperforms the unregularized version in self-play. The reason could be that the quality of the samples from the learned belief model worsen as the trajectories it sees at test time becomes more off-distribution for smaller λ. For reference, we also run the original SPARTA which does not have this problem as it computes beliefs analytically. The original SPARTA gets 23.49 ± 0.02, which is better than 1p piKL-SPARTA with $\lambda = 0.5$ but worse than 2p piKL-SPARTA with the same λ. The computational cost of 1p SPARTA with exact beliefs and 2p piKL-SPARTA with approximate learned beliefs is roughly the same because the exact beliefs computation accounts for 50% of the entire computation. Therefore, 2p piKL-SPARTA could be a preferable option to improve performance in Dec-POMDPs via self play without incurring the huge cost of joint SPARTA or RL-search Fickinger et al. (2021).