Advantage Alignment Algorithms

Juan Duque*, Milad Aghajohari*, Tim Cooijmans, Tianyu Zhang, Aaron Courville
University of Montreal & Mila
`firstname.lastname@umontreal.ca`

Abstract

The growing presence of artificially intelligent agents in everyday decision-making, from LLM assistants to autonomous vehicles, hints at a future in which conflicts may arise from each agent optimizing individual interests. In general-sum games these conflicts are apparent, where naive Reinforcement Learning agents get stuck in Pareto-suboptimal Nash equilibria. Consequently, opponent shaping has been introduced as a method with success at finding socially beneficial equilibria in social dilemmas. In this work, we introduce Advantage Alignment, a family of algorithms derived from first principles that perform opponent shaping efficiently and intuitively. This is achieved by aligning the advantages of conflicting agents in a given game by increasing the probability of mutually-benefiting actions. We prove that existing opponent shaping methods, including LOLA and LOQA, implicitly perform Advantage Alignment. Compared to these works, Advantage Alignment mathematically simplifies the formulation of opponent shaping and seamlessly works for continuous action domains. We also demonstrate the effectiveness of our algorithm in a wide range of social dilemmas, achieving state of the art results in each case, including a social dilemma version of the Negotiation Game.

1 Introduction

Recent developments in language modeling (GPT) (Radford et al., 2018), image synthesis (Diffusion) (Ho et al., 2020) and reinforcement learning (Gato) (Reed et al., 2022) hint towards a future in which artificially intelligent systems seamlessly integrate in everyday decision making. These systems often act in a decentralized manner and individually optimize for the goals of their users. However, this individual optimization can lead to conflicts, particularly in tasks characterized by having both cooperative and competitive elements. *Social dilemmas*, introduced by Rapoport and Chammah (1965) describe this kind of scenario in which each agent acting greedily often leads to an outcome with less utility than that which would have occurred if they had cooperated. A prime example of a social dilemma at a global scale is the climate change problem. Individual and national interests in economic growth often clash with the need for global cooperation to reduce carbon emissions and mitigate environmental degradation. This dichotomy highlights the complexity of aligning individual actions with the collective well-being.

The fast and opaque decision-making processes of machine learning systems make it challenging for humans to supervise every decision. Consequently, there is a pressing need to develop methods that enable agents to autonomously align their interests with each other. Despite this need, the deep reinforcement learning community has traditionally focused its attention on fully cooperative or fully competitive settings, neglecting the nuances of social dilemmas. Sandholm and Crites (1996) empirically demonstrate that naive reinforcement learning algorithms converge to Pareto suboptimal Nash equilibria in social dilemmas like the Iterated Prisoner's Dilemma (IPD). To address this gap, *opponent shaping* algorithms such as Learning with Opponent Learning Awareness (LOLA) (Foerster et al., 2018b) have been introduced.

LOLA is an opponent shaping algorithm that directs the behavior of other agents by assuming that they are naive learners and taking gradients with respect to imagined parameter updates. Since then, other opponent shaping algorithms that compute gradients w.r.t. to imaginary parameter updates have demonstrated similar success at partially competitive tasks: SOS (Letcher et al., 2021), COLA (Willi et al., 2022) and POLA (Zhao et al., 2022). More recently LOQA (Aghajohari et al., 2024b), proposes and alternative form of opponent shaping by assuming control over the value function of other agents via REINFORCE estimators (Williams, 1992). This new approach to opponent shaping has significant computational advantages compared to previous methods and establishes the foundation for the Advantage Alignment algorithms.

In this work we introduce Advantage Alignment, a family of algorithms designed with the goal of shaping rational opponents. We do so by making two assumptions about any reinforcement learning algorithm: a reinforcement learning algorithm (1) aims to maximize their own expected return and (2) takes actions proportionally to this expected return. Under these assumptions we demonstrate that opponent shaping reduces to the problem of aligning the advantages of different players and increasing the log probability of an action proportionally to their alignment. We show how this mechanism lies at the heart of some of the existing opponent shaping algorithms, including LOLA and LOQA. By distilling this objective, Advantage Alignment agents are able to shape opponents without imagined parameter updates (LOLA, SOS) or stochastic gradient estimation that relies on automatic differentiation introduced in DiCE (Foerster et al., 2018a) (POLA, COLA, LOQA).

Social dilemmas are present in all sorts of human interaction. We are particularly interested in the problem of designing climate negotiation strategies that allow agents to cooperate without centralized authorities. This problem grounds our research to a situation of great concern and applicability for reinforcement learning research, and allows us to identify and overcome key challenges that arise from scale often overlooked in more simple settings like the Iterated Prisoner's Dilemma (Rapoport and Chammah, 1965) and the Coin Game (Foerster et al., 2018b). With this goal in mind, we apply Advantage Alignment to a continuous variant of the Negotiation Game (also known as the Exchange Game) (Cao et al., 2018). In so doing, we aim to begin to address the challenges arising from scale, offering insights and solutions applicable to complex, real-world interactions.

We list our key contributions:

- We introduce Advantage Alignment and Proximal Advantage Alignment, two opponent shaping algorithms derived from first principles and reliant on policy gradient estimators.
- We prove that LOLA (and therefore its variations) and LOQA implicitly perform Advantage Alignment through different mechanisms.
- We extend REINFORCE-based opponent shaping to continuous action environments and achieve state-of-the-art results in a variant of the continuous Negotiation Game (Cao et al., 2018).

2 Background

2.1 Social Dilemmas

Social dilemmas describe situations in which selfish behavior leads to comparatively poor outcomes for everyone. Such dilemmas are often formalized as normal form games and constitute a subset of general-sum games. A classical example of a social dilemma is the Iterated Prisoner's Dilemma (IPD) (Rapoport and Chammah, 1965), in which two players can choose one of two actions: cooperate or defect. In the one-step version of the game, the dilemma occurs because defecting is a *dominating* strategy, i.e. independently of what the opponent plays the agent is better off playing defect. However, by the reward structure of the game, both the agent and the opponent would achieve a higher utility if they played cooperate simultaneously. Other social dilemmas have been studied in the literature including the Chicken Game and the Coin Game (Lerer and Peysakhovich, 2018), which has a similar reward structure to IPD but takes place in a grid world. In this paper we introduce a variation of the Negotiation Game (also known as the Exchange Game) (DeVault et al., 2015; Lewis et al., 2017), with a strong social dilemma component.

2.2 Markov Games

In this work, we consider fully observable, general sum, n-player Markov Games (Shapley, 1953) which are represented by a tuple: $\mathcal{M} = (N, \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$. Here $\mathcal{S}$ is the state space, $\mathcal{A} := \mathcal{A}^1 \times \ldots \times \mathcal{A}^n$, is the joint action space for all players, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ maps from every state and joint action to a probability distribution over states, $\mathcal{R} = \{r^1, \ldots, r^n\}$ is the set of reward functions where each $r^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ maps every state and joint action to a scalar return and $\gamma \in [0, 1]$ is the discount factor.

2.3 Reinforcement Learning

Consider two agents playing a Markov Game, 1 (agent) and 2 (opponent), with policies π^1 and π^2, parameterized by θ_1 and θ_2 respectively. We follow the notation of Agarwal et al. (2021), let τ denote a trajectory with initial state distribution μ and (unconditional) distribution given by:

$$\Pr_{\mu}^{\pi^1,\pi^2}(\tau) = \mu(s_0)\pi^1(a_0|s_0)\pi^2(b_0|s_0)P(s_1|s_0, a_0, b_0)\ldots \tag{1}$$

Where $P(\cdot|s, a, b)$, often referred as the transition dynamics, is a probability distribution over the next states conditioned on the current state being s, agent taking action a and opponent taking action b. Value-based methods like Q-learning (Watkins and Dayan, 1992) and SARSA (Rummery and Niranjan, 1994) learn an estimate of the discounted reward using the Bellman equation:

$$Q^1(s_t, a_t, b_t) = r^1(s_t, a_t, b_t) + \mathbb{E}_{s_{t+1}}\left[V^1(s_{t+1})|s_t, a_t, b_t\right] \tag{2}$$

In policy optimization, both players aim to maximize their expected discounted return by performing gradient ascent with a Reinforce estimator (Williams, 1992) of the form:

$$\nabla_{\theta_1} V^1(\mu) = \mathbb{E}_{\tau\sim\Pr_{\mu}^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t)\nabla_{\theta_1} \log \pi^1(a_t|s_t)\right] \tag{3}$$

Here $A^1(s, a, b)$ denotes the advantage of the agent taking action a in state s while the opponent takes action b.

3 Opponent Shaping

Opponent shaping, first introduced in LOLA (Foerster et al., 2018b), is a paradigm that assumes the learning dynamics of other players can be controlled via some mechanism to incentivize desired behaviors. LOLA and its variants assume that the opponent is a naive learner, i.e. an agent that performs gradient ascent on their value function, and differentiate through an imagined naive update of the opponent in order to shape it.

LOQA (Aghajohari et al., 2024b) assumes the opponent's policy is a softmax over Q-values:

$$\hat{\pi}^2(b_t|s_t) := \frac{\exp Q^2(s_t, a_t, b_t)}{\sum_b \exp Q^2(s_t, a_t, b)}. \tag{4}$$

The key idea is that these Q-values depend on π^1, and hence the opponent policy $\hat{\pi}^2$ can be differentiated w.r.t. θ_1:

$$\nabla_{\theta_1}\hat{\pi}^2(b_t|s_t) = \hat{\pi}^2(b_t|s_t)\left(\nabla_{\theta_1} Q^2(s_t, a_t, b_t) - \sum_b \hat{\pi}^2(b|s_t)\nabla_{\theta_1} Q^2(s_t, a_t, b)\right). \tag{5}$$

See Appendix A.3 for a derivation of this expression. This dependency of π^2 on θ_1 leads to the emergence of an extra term in the policy gradient:

$$\nabla_{\theta_1} V^1(\mu) = \mathbb{E}_{\tau\sim\Pr_{\mu}^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty}\gamma^t A^1(s_t, a_t, b_t)\left(\underbrace{\nabla_{\theta_1}\log\pi^1(a_t|s_t)}_{\text{policy gradient term}} + \underbrace{\nabla_{\theta_1}\log\pi^2(b_t|s_t)}_{\text{opponent shaping term}}\right)\right]. \tag{6}$$

Aghajohari et al. (2024b) demonstrate an effective way to account for this dependency using REINFORCE. The present work builds on the ideas of LOQA, but reduces opponent shaping to its bare components to derive Advantage Alignment from first principles.

Algorithm 1 Advantage Alignment

```
**Initialize:** Discount factor $\gamma$, agent Q-value parameters $\phi^1$, t Q-value parameters $\phi^1_{\text{t}}$, actor parameters $\theta^1$, opponent Q-value parameters $\phi^2$, t Q-value parameters $\phi^2_{\text{t}}$, actor parameters $\theta^2$
for iteration= $1, 2, \ldots$ do
    Run policies $\pi^1$ and $\pi^2$ for $T$ timesteps in environment and collect trajectory $\tau$
    Compute agent critic loss $L^1_C$ using the TD error with $r^1$ and $V^1$
    Compute opponent critic loss $L^2_C$ using the TD error with $r^2$ and $V^2$
    Optimize $L^1_C$ w.r.t. $\phi^1$ and $L^2_C$ w.r.t. $\phi^2$ with optimizer of choice
    Compute generalized advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$
    Compute agent actor loss, $L^1_a$, summing equations (3) and (8)
    Compute opponent actor loss, $L^2_a$, summing equations (3) and (8)
    Optimize $L^1_a$ w.r.t. $\theta^1$ and $L^2_a$ w.r.t. $\theta^2$ with optimizer of choice
end for
```

4 Advantage Alignment

Motivated by the goal of scaling opponent shaping algorithms to more diverse and complex scenarios, we derive a simple and intuitive objective for efficient opponent shaping. We begin from the assumptions that agents are learning to maximize their expected return, and will behave in a fashion that is proportional to this goal:

Assumption 1. *Each agent i learns to maximize their value function:* $\max V^i(\mu)$.

Assumption 2. *Each opponent i acts proportionally to the exponent of their action-value function:* $\pi^i(a|s) \propto \exp\big(Q^i(s,a)\big)$.

Using equation (6) and substituting $\hat{\pi}^2$ in place of π^2 (per assumption 2), we obtain

$$\nabla_{\theta_1} V^1(\mu) = \mathbb{E}_{\tau\sim \text{Pr}_\mu^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t,a_t,b_t)\left(\nabla_{\theta_1}\log\pi^1(a_t|s_t) + \nabla_{\theta_1}\log\hat{\pi}^2(b_t|s_t)\right)\right].$$

The first term is the usual policy gradient. The second term is the opponent shaping term and will be our focus. Approximating (5) by ignoring the contribution due to the partition function, this term becomes:

$$\mathbb{E}_{\tau\sim \text{Pr}_\mu^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t,a_t,b_t)\nabla_{\theta^1} Q^2(s_t,a_t,b_t)\right]. \tag{7}$$

The gradient of the Q-value can be estimated by a REINFORCE estimator, which leads to a nested expectation. Aghajohari et al. (2024b) showed that this nested expectation can be flattened, allowing efficient estimation from a single trajectory. We take the same approach (see Appendix A.1) to obtain

$$\mathbb{E}_{\tau\sim \text{Pr}_\mu^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty}\sum_{k=t+1}^{\infty} \gamma^k A^1(s_t,a_t,b_t)A^2(s_k,a_k,b_k)\nabla_{\theta^1}\log\pi^1(a_k|s_k)\right]. \tag{8}$$

This is the main result of this work. The expression above captures the essence of opponent shaping: an agent should align its advantages with those of its opponent in order to steer towards trajectories that are mutually beneficial. More precisely, an agent increases the probability of actions that have high product between the sum of its past advantages and the advantages of the opponent at the current time step. This expression for the Advantage Alignment formula is detailed in Appendix A.5. Equation (8) depends only on the log probabilities of the agent, which allows us to create a proximal surrogate objective that closely follows the PPO (Schulman et al., 2017b) formulation:

$$\mathbb{E}_{\tau\sim \text{Pr}_\mu^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty}\sum_{k=t+1}^{\infty} \gamma^k \min\left\{r_n(\theta_1)A^1_tA^2_k,\ \text{clip}\left(r_n(\theta_1);1-\epsilon,1+\epsilon\right)A^1_tA^2_k\right\}\right] \tag{9}$$

Where r_n denotes the ratio between the policy (current) after n updates and the original policy (old). We also denote $A^i_t := A^i(s_t,a_t,b_t)$, the advantage of agent i at timestep t. This surrogate objective in equation (9) is used to formulate the Proximal Advantage Alignment (Proximal AdAlign) algorithm (see appendix A.6 for implementation details).

Table 1: Update direction as a function of signs of advantages

A^1 \ A^2	+	−
+	+	−
−	−	+

Why is Assumption 1 necessary? Assumption 1 allows the agent to influence the learning dynamics of the opponent, by controlling the values for different actions. After one iteration of the algorithm the agent changes the Q-values of the opponent for different actions and, since the opponent aims to maximize their expected return (under the new agent policy), it must change its behavior accordingly.

4.1 Interpreting Advantage Alignment

Equation (8) yields four possible different cases for controlling the direction of the gradient of the log probability of the policy. As with the usual policy gradient estimator, the sign multiplying the log probability indicates whether the probability of taking an action should increase or decrease. In Table 1 we show the four different quadrants corresponding to the different signs the two advantages can take. We label and interpret the meaning of each of the four quadrants.

Cooperative: The sign is positive since the two advantages (for the agent and opponent) are positive. Therefore it is rational to increase the probability of this action, since by doing so the agent increases the probability of observing trajectories that are mutually beneficial for it and the opponent.

Empathetic: The sign is negative since the action diminishes the opponent's return. The agent reduces the probability of taking this action because it hurts the opponent, despite the action being beneficial for the agent, hence the name empathetic.

Vengeful: The sign is negative since the action diminishes the agent's return. The agent reduces the probability of taking this action because it hurts it, despite the action being beneficial for the opponent, hence the name vengeful.

Spiteful: When both advantages are negative, advantage alignment will *increase* the probability of taking this action despite hurting both agents. This behavior is counterintuitive; we believe it serves as a deterrent, much like mutually assured destruction.

We now relate existing opponent shaping algorithms to advantage alignment, and argue that these algorithms embody these four behaviors. Theorem 1 shows how opponent shaping algorithms derived from LOLA implicitly do advantage multiplication, which lies at the heart of advantage alignment. Theorem 2 proves that LOQA's opponent shaping term has the same form as that of Advantage Alignment, differing only by a scalar term. Table 1 then holds for algorithms like LOLA, POLA, COLA, SOS and LOQA.

Theorem 1 (LOLA as an advantage alignment estimator)**.** *Let the time dependent sets of gradients of log probabilities* $B_t := \{\nabla_{\theta_1} \log \pi^1(a_0|s_0), \ldots, \nabla_{\theta_1} \log \pi^1(a_t|s_t)\}$ *and* $C_t := \{\nabla_{\theta_2} \log \pi^2(b_0|s_0), \ldots, \nabla_{\theta_2} \log \pi^2(b_t|s_t)\}$ *contain all the gradients up to time t. Then the opponent shaping term in LOLA's update can be rewritten as*

$$\mathbb{E}_{\tau \sim Pr_{\mu}^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty} A_t^2 \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T\right)\right] \mathbb{E}_{\tau \sim Pr_{\mu}^{\pi^1,\pi^2}}\left[\sum_{t=0}^{\infty} A_t^1 \nabla_{\theta_2} \log \pi^2(b_t|s_t)\right]. \tag{10}$$

The proof can be found in Appendix A.2.

Theorem 2 (LOQA as an advantage aligment estimator)**.** *The opponent shaping term in LOQA is equivalent to the opponent shaping term in advantage alignment up to a scalar* $(1 - \tilde{\pi}^2(b_t|s_t))$

$$\mathbb{E}_{\tau \sim Pr_{\mu}^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \sum_{k=t+1}^{\infty} (1 - \tilde{\pi}^2(b_t|s_t)) \cdot \gamma^k A_t^1 A_k^2 \nabla_{\theta^1} \log \pi^1(a_k|s_k) \right] \tag{11}$$

where $\tilde{\pi}^2(b_t|s_t)$ *approximates the opponent policy as defined in LOQA. For a proof see appendix A.4.*

5 Experiments

5.1 Iterated Prisoner's Dilemma

We consider the *full history* version of IPD, where a gated recurrent unit (GRU) policy conditions on the full trajectory of observations before sampling an action. In this experiment we follow the architecture used in POLA (Zhao et al., 2022) (for details see appendix B.1). We also consider trajectories of length 16 with a discount factor, γ, of 0.9. As shown in figure 1, Advantage Alignment agents consistently achieve a policy that resembles *tit-for-tat* (Rapoport and Chammah, 1965) empirically. Tit-for-tat consists of cooperating on the first move and then mimicking the opponent's previous move in subsequent rounds.

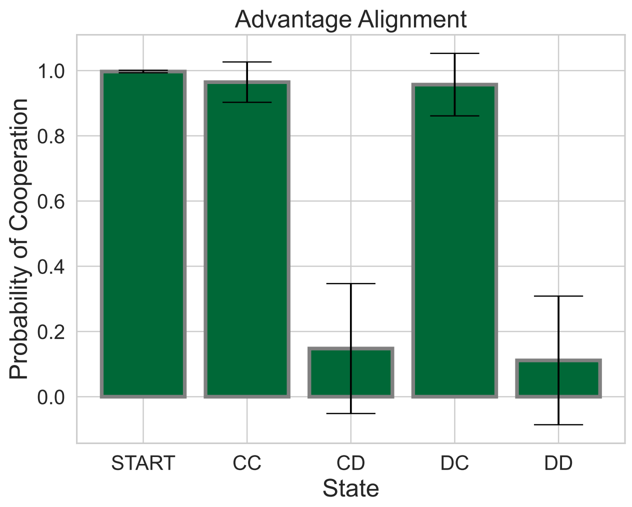

Figure 1: Empirical probability of cooperation for each previous combination of actions in the Iterated Prisoner's Dilemma. The results are averaged over 10 random seeds, the black whiskers show a one standard deviation confidence interval.

5.2 Coin Game

The Coin Game is a 3x3 grid world environment where two agents, red and blue, take turns collecting coins. During each turn, a coin of either red or blue color spawns at a random location on the grid. Agents receive a reward of +1 for collecting any coin but incur a penalty of -3 if the opponent collects a coin of their color. A Pareto-optimal strategy in the Coin Game is for each agent to collect only the coins matching their color, as this approach maximizes the total returns for both agents.

Figure 2 demonstrates that Advantage Alignment agents perform similarly to LOQA agents when evaluated against a league of different policies: Advantage Alignment agents cooperate with themselves, cooperate with always cooperate (AC) and are not exploited by always defect (AD). In contrast to LOQA, Advantage Alignment allows for different flavors of agents. Figure 3 shows the behavior of different Advantage Alignment training by masking different quadrants. As expected, agents get more defective when the Cooperative (C) and Empathetic (E) quadrants are masked, and more cooperative when the Vengeful (V) and Spiteful (S) quadrants are masked.

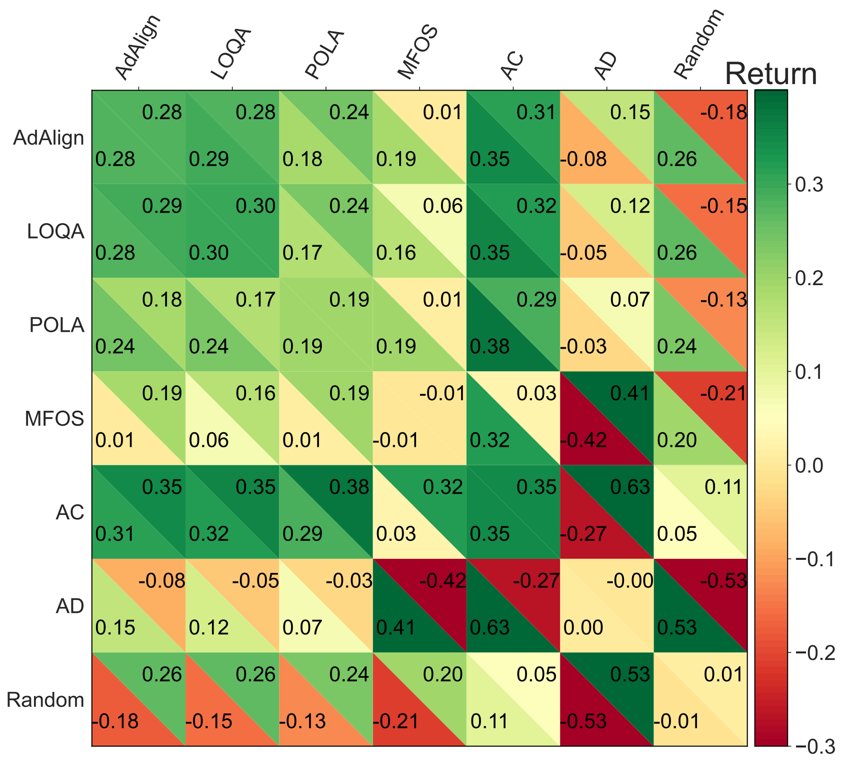

Figure 2: League Results of the Advantage Alignment agents in Coin Game: LOQA, POLA, MFOS, Always Cooperate (AC), Always Defect (AD), Random and Advantage Alignment (AdAlign). Each number in the plot is computed by running 10 random seeds of each agent head to head with 10 seeds of another for 50 episodes of length 16 and averaging the rewards.

5.3 Negotiation Game

In the original Negotiation Game, two agents bargain over n types of items over multiple rounds. In each round, both the quantity of items and the value each agent places on them are randomly set, but the agents only know their own values. They take turns proposing how to divide the items over a random number of turns. Agents can end the negotiation by agreeing to a proposal, and rewards are based on how well the agreement matches their private values. If they don't reach an agreement by the final turn, neither gets a reward.

We modify the game first by making the values public, otherwise Advantage Alignment would have an unfair edge over PPO agents by using the opponent's value function. Secondly, we do one-shot, simultaneous negotiations instead of negotiation rounds lasting multiple iterations. Third, we modify the reward function so that every negotiation yields a reward. For a given item with agent value v_a, the reward of the agent r_a depends on the proposal of the agent p_a and the proposal of the opponent p_o where $p_a, p_o \in [0, 5]$:

$$r_a = \frac{p_a \cdot v_a}{\max(5, p_a + p_o)}$$

Note that the max operation at the denominator serves to break the invariance of the game dynamics to the scale of proposals. For example, without the max operation, there would be no difference between $p_a = 1, p_o = 1$ and $p_a = 5, p_o = 5$.

The social dilemma in this version of the negotiation game arises because both agents are incentivized to take as many items as possible, but by doing so, they end up with a lower return compared to the outcome they would achieve if they split the items based on their individual utilities. A Pareto-optimal strategy entails allowing the agent to take all the items that are more valuable to them, and similarly for their opponent (this constitutes the always cooperate (AC) strategy in Figure 4a). We experiment

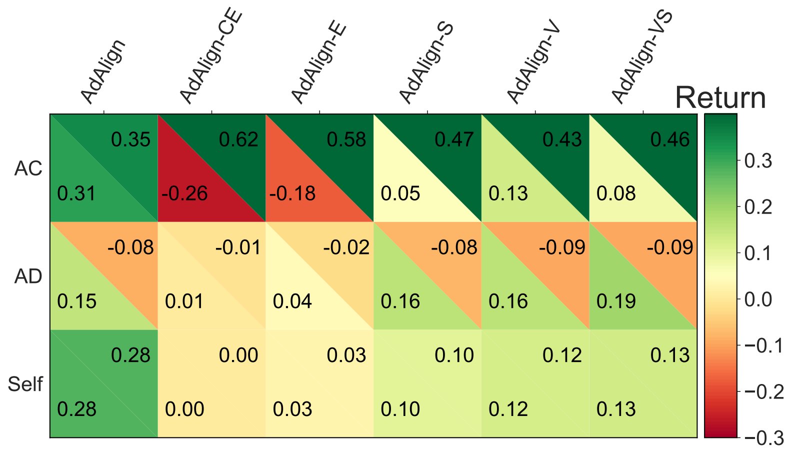
Figure 3: League Results of the different ablations of Advantage Alignment agents by masking different quadrants in the Coin Game. The letters specify the quadrants that were masked for each variation: Cooperative (C), Empathetic (E), Vengeful (V) and Spiteful (S).

with a high-contrast setting where the utilities of objects for the agents are orthogonal to each other: There are two possible combinations of values in this setup: $v_a = 5, v_b = 1$ or $v_a = 1, v_b = 5$.

As shown in Figure 4a, PPO agents do not learn to solve the social dilemma. They learn the naive policy of bidding high for every item which means they get a low return against themselves. PPO agents trained with shared rewards get a high return against themselves, only to be exploited by PPO agents. They do not learn to abandon cooperation and retaliate after they are defected against. Advantage Alignment agents solved the social dilemma. They cooperate with themselves while remaining non-exploitable against Always Defect.

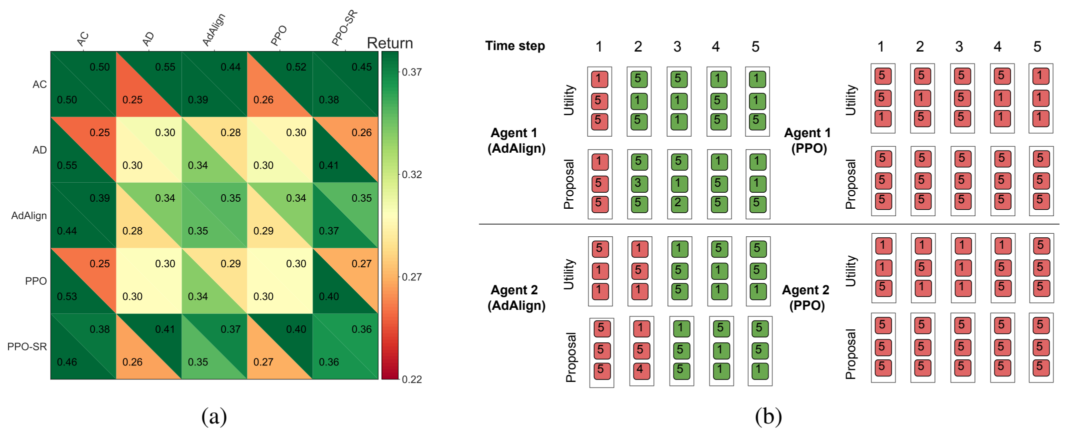
Figure 4: (a) League Results of the Advantage Alignment agents in the Negotiation Game: Always Cooperate (AC), an agent which proposes 5 for items which are more valuable to it and 1 for items that are less valuable to it, Always Defect (AD), an agent that proposes 5 regardless of the values, Advantage Alignment (AdAlign), PPO and PPO summing rewards (PPO-SR). Each number in the plot is computed by running 10 random seeds of each agent head to head with 10 seeds of another for 50 episodes of length 16 and averaging the rewards. Note that in this game, Always cooperate which is the most exploitable against the Always Defect gets an average return of 0.25 and the least Always Defect as the least exploitable agent achieves 0.30. While PPO and PPO-SR both fail to solve the game by having a cooperative reciprocation-based policy, Advantage Alignment largely solves it.(b) Sample trajectories of AdAlign vs. AdAlign and PPO vs. PPO in the negotiation game. The numbers show the utilities and proposals, which have been rounded to integer values. AdAlign agents defect first (red) and progressively cooperate whith each other (green) while PPO agents always defect.

6 Related Work

The Iterated Prisoner's Dilemma (IPD) was introduced by Rapoport and Chammah (1965). *Tit-for-tat* was discovered as a robust strategy against a population of opponents in IPD by Axelrod (1984), who organized multiple IPD tournaments. It was discovered only recently that IPD contains strategies that extort rational opponents into exploitable cooperation (Press and Dyson, 2012). Sandholm and Crites (1996) were the first to demonstrate that two Q-learning agents playing IPD converge to mutual defection, which is suboptimal. Later, Foerster et al. (2018b) demonstrated that the same is true for policy gradient methods. Bertrand et al. (2023) were able to show that with optimistic initialization and *self-play*, Q-learning agents find a *Pavlov* strategy in IPD.

Opponent shaping is first introduced in LOLA Foerster et al. (2018b), as a method for controlling the learning dynamics of opponents in a game. To do so, a LOLA agent assumes the opponents are naive learners and differentiates through a one step look-ahead optimization update of the opponent. More formally, LOLA maximizes $V^1(\theta^1, \theta^2 + \Delta\theta^2)$ where $\Delta\theta^2$ is a naive learning step in the direction that maximizes the opponent's value function $V^2(\theta^1, \theta^2)$. Variations of LOLA have been introduced to have formal stability guarantees (Letcher et al., 2021), learn consistent update functions assuming mutual opponent shaping (Willi et al., 2022) and be invariant to policy parameterization (Zhao et al., 2022). More recent work performs opponent shaping by having an agent play against a best response approximation of their policy (Aghajohari et al., 2024a). LOQA (Aghajohari et al., 2024b), on which this work is based, performs opponent shaping by controlling the Q-values of the opponent using REINFORCE (Williams, 1992) estimators.

Another approach to finding socially beneficial equilibria in general sum games relies on modeling the problem as a meta-game, where meta-rewards correspond to the returns on the inner game, meta-states correspond to joint policies of the players, and the meta-actions are updates to these policies. Al-Shedivat et al. (2018) introduce a continuous adaptation framework for multi-task learning that uses meta-learning to deal with non-stationary environments. MFOS (Lu et al., 2022) uses model-free optimization methods like PPO and genetic algorithms to optimize the meta-value of the meta-game. More recently Meta-Value Learning (Cooijmans et al., 2023) parameterizes the meta-value as a neural network and applies Q-learning to capture the future effects of changes to the inner policies.

The Negotiation Game, introduced by DeVault et al. (2015); Lewis et al. (2017) and subsequently refined by Cao et al. (2018), has proven to be a significant benchmark for studying general-sum games. it integrates elements of strategy and social dilemmas, necessitating that agents balance cooperation and competition to optimize their outcomes. Noukhovitch et al. (2021) analyze this complex benchmark, underscoring its importance in the field. Future investigations will turn towards an even more sophisticated simulation proposed by Zhang et al. (2022), which involves negotiations among countries and regions with diverse resource distributions and preferences in addressing climate change. This advanced model aims to provide deeper insights into negotiation strategies and outcomes in intricate, real-world scenarios, potentially informing policy and decision-making processes in global climate negotiations.

7 Conclusion

We introduced Advantage Alignment, a family of algorithms designed to shape the behavior of reinforcement learning opponents in social dilemmas. Advantage Alignment agents are able to find cooperative and non-exploitable policies, by aligning the advantages of different players and increasing the log probability of actions proportionally to this alignment.

We have also shown that LOLA and LOQA implicitly perform Advantage Alignment through different mechanisms, thus unifying and elucidating existing opponent shaping techniques. Our approach eliminates the need for imagined parameter updates or complex stochastic gradient estimation techniques, which are common in algorithms like LOLA, POLA, and LOQA. In the context of the Negotiation Game, a complex social dilemma scenario, we demonstrated the effectiveness of Advantage Alignment. Our algorithm achieved state-of-the-art results in the continuous action version of the Negotiation Game, showcasing the potential of Advantage Alignment agents to handle real-world applications such as formulating climate negotiation strategies.

References

Agarwal, A., Jiang, N., Kakade, S., and Sun, W. (2021). *Reinforcement Learning: Theory and Algorithms*.

Aghajohari, M., Cooijmans, T., Duque, J. A., Akatsuka, S., and Courville, A. (2024a). Best response shaping.

Aghajohari, M., Duque, J. A., Cooijmans, T., and Courville, A. (2024b). Loqa: Learning with opponent q-learning awareness.

Al-Shedivat, M., Bansal, T., Burda, Y., Sutskever, I., Mordatch, I., and Abbeel, P. (2018). Continuous adaptation via meta-learning in nonstationary and competitive environments.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic, New York.

Bertrand, Q., Duque, J., Calvano, E., and Gidel, G. (2023). Q-learners can provably collude in the iterated prisoner's dilemma.

Cao, K., Lazaridou, A., Lanctot, M., Leibo, J. Z., Tuyls, K., and Clark, S. (2018). Emergent communication through negotiation.

Cooijmans, T., Aghajohari, M., and Courville, A. (2023). Meta-value learning: a general framework for learning with learning awareness.

DeVault, D., Mell, J., and Gratch, J. (2015). Toward natural turn-taking in a virtual human negotiation agent. In *AAAI Spring Symposia*.

Foerster, J., Farquhar, G., Al-Shedivat, M., Rocktäschel, T., Xing, E. P., and Whiteson, S. (2018a). Dice: The infinitely differentiable monte-carlo estimator.

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. (2018b). Learning with opponent-learning awareness.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models.

Lerer, A. and Peysakhovich, A. (2018). Maintaining cooperation in complex social dilemmas using deep reinforcement learning.

Letcher, A., Foerster, J., Balduzzi, D., Rocktäschel, T., and Whiteson, S. (2021). Stable opponent shaping in differentiable games.

Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., and Batra, D. (2017). Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv: 1706.05125*.

Lu, C., Willi, T., de Witt, C. S., and Foerster, J. (2022). Model-free opponent shaping.

Noukhovitch, M., LaCroix, T., Lazaridou, A., and Courville, A. C. (2021). Emergent communication under competition. In Dignum, F., Lomuscio, A., Endriss, U., and Nowé, A., editors, *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pages 974–982. ACM.

Press, W. H. and Dyson, F. J. (2012). Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Rapoport, A. and Chammah, A. (1965). *Prisoner's Dilemma: A Study in Conflict and Cooperation*. University of Michigan Press.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. (2022). A generalist agent.

Rummery, G. and Niranjan, M. (1994). On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University.

Sandholm, T. and Crites, R. (1996). Multiagent reinforcement learning in the iterated prisoner's dilemma. *Bio Systems*, 37(1-2):147–166.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2017a). Trust region policy optimization.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms.

Shapley, L. (1953). Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Willi, T., Letcher, A., Treutlein, J., and Foerster, J. (2022). Cola: Consistent learning with opponent-learning awareness.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Zhang, T., Williams, A., Phade, S. R., Srinivasa, S., Zhang, Y., Gupta, P., Bengio, Y., and Zheng, S. (2022). Ai for global climate cooperation: Modeling global climate negotiations, agreements, and long-term cooperation in rice-n. *Social Science Research Network*.

Zhao, S., Lu, C., Grosse, R. B., and Foerster, J. N. (2022). Proximal learning with opponent-learning awareness.

Appendix

Table of Contents

A Mathematical Derivations

A.1 Deriving the Advantage Alignment Formula

In this section we derive the advantage alignment formula in equation (8) from the opponent shaping expression in equation (6) and assumption 2. Recall assumption 2:

$$\pi^i(a|s) \propto \exp Q^i(s,a)$$

Recall the opponent shaping policy gradient expression:

$$\nabla_{\theta_1} V^1(\mu) = \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t) \left(\underbrace{\nabla_{\theta_1} \log \pi^1(a_t|s_t)}_{\text{(A)}} + \underbrace{\nabla_{\theta_1} \log \pi^2(b_t|s_t)}_{\text{(B)}} \right) \right]$$

We expand the term (B) above splitting the expectation, by assumption 2, we can write:

$$\begin{aligned}
&\mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t) \nabla_{\theta_1} \log \pi^2(b_t|s_t) \right] \\
&\propto \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t) \nabla_{\theta_1} \log \exp Q^2(s_t, a_t, b_t) \right] \\
&= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t) \nabla_{\theta_1} Q^2(s_t, a_t, b_t) \right]
\end{aligned}$$

For convenience of notation we define:

$$r_t^i := r^i(s_t, a_t, b_t), \;\; A_t^i := A^i(s_t, a_t, b_t)$$

These are the reward and advantage of agent i at time step t after taking action a_t and opponent taking action b_t. From the Bellman equation (2) we expand as follows:

$$\mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A^1(s_t, a_t, b_t) \nabla_{\theta^1} Q^2(s_t, a_t, b_t) \right] \tag{12}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A_t^1 \nabla_{\theta^1} \left(r_t^2 + \mathbb{E}_{s'} \left[V^2(s') \middle| s_t, a_t, b_t \right] \right) \right] \tag{13}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A_t^1 \nabla_{\theta^1} \mathbb{E}_{s'} \left[V^2(s') \middle| s_t, a_t, b_t \right] \right] \tag{14}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t A_t^1 \mathbb{E}_{\tau' \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{k=0}^{\infty} \gamma^k A_k^2 \nabla_{\theta^1} \log \pi^1(a_k|s_k) \middle| s_t, a_t, b_t \right] \right] \tag{15}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \mathbb{E}_{\tau' \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{k=0}^{\infty} \gamma^{k+t} A_t^1 A_k^2 \nabla_{\theta^1} \log \pi^1(a_k|s_k) \middle| s_t, a_t, b_t \right] \right] \tag{16}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\mathbb{E}_{\tau' \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \sum_{k=t+1}^{\infty} \gamma^k A_t^1 A_k^2 \nabla_{\theta^1} \log \pi^1(a_k|s_k) \middle| s_t, a_t, b_t \right] \right] \tag{17}$$

$$= \mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \sum_{k=t+1}^{\infty} \gamma^k A_t^1 A_k^2 \nabla_{\theta^1} \log \pi^1(a_k|s_k) \right] \tag{18}$$

We use the Bellman equation in line (13). In line (16), we use the fact that $\gamma^t A^1(s_t, a_t, b_t)$ is measurable w.r.t. the natural filtration of the process up to time t, $\mathcal{F}_t$, and the independence of the two terms conditioned on $\mathcal{F}_t$ by the Markov property. In line (17) we use linearity of expectation. In line (18) we use the rule of the iterated expectations.

A.2 Proof of Theorem 1

LOLA (Foerster et al., 2018b) optimizes the return of the agent under an imagined optimization step of the opponent (assuming the opponent is a naive learning algorithm). Under their notation, a LOLA agent optimizes $V^1(\theta_1, \theta_2 + \Delta\theta_2)$ where $\Delta\theta_2$ is a gradient ascent step on the parameters of the opponent θ_2. Since computing this value function explicitly is difficult, LOLA uses the first-order Taylor expansion surrogate objective:

$$V^1(\theta^1, \theta^2 + \Delta\theta_2) \approx V^1(\theta_1, \theta_2) + (\Delta\theta_2)^T \nabla_{\theta_2} V^1(\theta_1, \theta_2) \tag{19}$$

The gradient of the expression above w.r.t. the parameters θ_1 of the agent is given by

$$\nabla_{\theta_1} V^1(\theta^1, \theta^2 + \nabla_{\theta_2} V^2(\theta_1, \theta_2)) = \nabla_{\theta_1} V^1(\theta_1, \theta_2) + \left(\nabla_{\theta_1} \nabla_{\theta_2} V^2(\theta^1, \theta^2)\right) \nabla_{\theta_2} V^1(\theta_1, \theta_2). \tag{20}$$

The first-order terms above can be computed using the Advantage form of the REINFORCE estimator, which is given by equation (3). Foerster et al. (2018b) derive the following REINFORCE estimator for the second-order term:

$$\mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t r_t^2 \left(\sum_{k=0}^{t} \nabla_{\theta_1} \log \pi^1(a_k|s_k) \right) \left(\sum_{k=0}^{t} \nabla_{\theta_2} \log \pi^2(b_k|s_k) \right)^T \right]. \tag{21}$$

For convenience of notation we construct the time dependent sets:

$$B_t := \{\nabla_{\theta_1} \log \pi^1(a_0|s_0), \ldots, \nabla_{\theta_1} \log \pi^1(a_t|s_t)\}$$
$$C_t := \{\nabla_{\theta_2} \log \pi^2(b_0|s_0), \ldots, \nabla_{\theta_2} \log \pi^2(b_t|s_t)\}$$

Using this notation, we expand the second order term beginning from equation (20), to bring out the advantage A_t^2:

$$\begin{aligned}
&\nabla_{\theta_1} \nabla_{\theta_2} V^2(\theta^1, \theta^2) && (22)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t r_t^2 \left(\sum_{k=0}^{t} \nabla_{\theta_1} \log \pi^1(a_t|s_t) \right) \left(\sum_{k=0}^{t} \nabla_{\theta_2} \log \pi^2(b_t|s_t) \right)^T \right] && (23)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) \sum_{k=t}^{\infty} \gamma^k r_k^2 \right] && (24)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) R^2(s_t) \right] && (25)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) \mathbb{E}\left[R^2(s_t)|s_t, a_t\right] \right] && (26)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) Q^2(s_t, b_t) \right] && (27)\\
&= \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) A_t^2 \right] && (28)
\end{aligned}$$

Where we reorder the terms of the summation to sum over future rewards instead of past gradient terms in line 24, we use the law of iterated expectation in line 26 and a baseline subtraction in line 28. $R^2(s_t)$ denotes the return starting at state s_t.

Per Equation (20), we multiply this Hessian with the gradient $\nabla_{\theta_2} V^1(\theta_1, \theta_2)$ to obtain

$$\mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} A_t^2 \left(\sum_{\nabla_a \in B_t} \sum_{\nabla_b \in C_t} \nabla_a \nabla_b^T \right) \right] \mathbb{E}_{\tau \sim \mathrm{Pr}_\mu^{\pi^1, \pi^2}} \left[\sum_{t=0}^{\infty} A_t^1 \nabla_{\theta_2} \log \pi^2(b_t|s_t) \right].$$

This completes the proof. □

A.3 Gradient of LOQA

Recall the opponent policy approximation used in LOQA, which takes a softmax over the Q-values of the opponent. We assume exact estimates of these Q-values:

$$\hat{\pi}^2(b_t|s_t) := \frac{\exp Q^2(s_t, a_t, b_t)}{\sum_b \exp Q^2(s_t, a_t, b)} \tag{4}$$

Note that $\hat{\pi}^2(b_t|s_t)$ is differentiable w.r.t. the parameters θ_1 of the policy of the agent because the Q-values depend on π_1. Therefore, we can use the policy gradient theorem (see Aghajohari et al. (2024b)) to differentiate the value function of the opponent w.r.t. the parameters of the agent. For convenience of notation we define:

$$Q_t^i(b) := Q^i(s_t, a_t, b)$$

Computing the gradient of the approximated opponent's policy we get:

$$\begin{aligned}
\nabla_{\theta_1} \hat{\pi}^2(b_t|s_t) &= \nabla_{\theta_1} \left(\frac{\exp Q^2(s_t, a_t, b_t)}{\sum_b \exp Q^2(s_t, a_t, b)} \right) && (29)\\
&= \frac{\nabla_{\theta_1} \exp Q_t^2(b_t)}{\sum_b \exp Q_t^2(b)} - \frac{\exp Q_t^2(b_t) \nabla_{\theta_1} \sum_b \exp Q_t^2(b)}{\left(\sum_b \exp Q_t^2(b)\right)^2} && (30)\\
&= \frac{\exp Q_t^2(b_t) \nabla_{\theta_1} Q_t^2(b_t)}{\sum_b \exp Q_t^2(b)} - \frac{\exp Q_t^2(b_t) \sum_b \exp Q_t^2(b) \nabla_{\theta_1} Q_t^2(b)}{\left(\sum_b \exp Q_t^2(b)\right)^2} && (31)\\
&= \frac{\exp Q_t^2(b_t)}{\sum_b \exp Q_t^2(b)} \left(\nabla_{\theta_1} Q_t^2(b_t) - \sum_b \frac{\exp Q_t^2(b) \nabla_{\theta_1} Q_t^2(b)}{\sum_b \exp Q_t^2(b)} \right) && (32)\\
&= \hat{\pi}^2(b_t|s_t) \left(\nabla_{\theta_1} Q_t^2(b_t) - \sum_b \hat{\pi}^2(b|s_t) \nabla_{\theta_1} Q_t^2(b) \right) && (33)
\end{aligned}$$

Where we used the quotient rule in line (30) and equation (4) in line(33). By the chain rule, the gradient of the log probability is:

$$\nabla_{\theta_1} \log \hat{\pi}^2(b_t|s_t) = \frac{\nabla_{\theta_1} \hat{\pi}^2(b_t|s_t)}{\hat{\pi}^2(b_t|s_t)} = \nabla_{\theta_1} Q_t^2(b_t) - \sum_b \hat{\pi}^2(b|s_t) \nabla_{\theta_1} Q_t^2(b).$$

This concludes the derivation.

A.4 Proof of Theorem 2

In practice, LOQA deviates from the approach discussed in Appendix A.3. Specifically, it does not differentiate through all of the Q-values, but only through that of the action b_t actually observed in the sampled trajectory:

$$\tilde{\pi}^2(b_t|s_t) := \frac{\exp Q^2(s_t, a_t, b_t)}{\exp Q^2(s_t, a_t, b_t) + \sum_{b \neq b_t} \underbrace{\exp Q^2(s_t, a_t, b)}_{\text{non-differentiable}}} \tag{34}$$

This choice is made because the trajectory provides an estimate of the Q-value of each opponent action b_t. This estimate statistically depends on the agent's actions $a_{<t}$ and therefore can be stochastically differentiated w.r.t θ_1 using REINFORCE. The other Q-values will be estimated by function approximators, which depend only implicitly on θ_1 and cannot be differentiated.

Differentiating (34) leads to a simplified gradient:

$$\nabla_{\theta_1} \tilde{\pi}^2(b_t|s_t) = \nabla_{\theta_1} \left(\frac{\exp Q^2(s_t,a_t,b_t)}{\exp Q^2(s_t,a_t,b_t) + \sum_{b \neq b_t} \exp Q^2(s_t,a_t,b)} \right) \tag{35}$$

$$= \nabla_{\theta_1} \exp Q_t^2(b_t) \frac{\left(\exp Q_t^2(b_t) + \sum_{b\neq b_t} \exp Q_t^2(b)\right) - \exp Q_t^2(b_t)}{\left(\exp Q_t^2(b_t) + \sum_{b\neq b_t} \exp Q_t^2(b)\right)^2} \tag{36}$$

$$= \nabla_{\theta_1} \exp Q_t^2(b_t) \frac{\sum_{b\neq b_t} \exp Q_t^2(b)}{\left(\exp Q_t^2(b_t) + \sum_{b\neq b_t} \exp Q_t^2(b)\right)^2} \tag{37}$$

$$= \exp Q_t^2(b_t) \nabla_{\theta_1} Q_t^2(b_t) \frac{\sum_{b\neq b_t} \exp Q_t^2(b)}{\left(\exp Q_t^2(b_t) + \sum_{b\neq b_t} \exp Q_t^2(b)\right)^2} \tag{38}$$

$$= \exp Q_t^2(b_t) \nabla_{\theta_1} Q_t^2(b_t) \frac{\sum_{b\neq b_t} \exp Q_t^2(b) + \exp Q_t^2(b_t) - \exp Q_t^2(b_t)}{\left(\exp Q_t^2(b_t) + \sum_{b\neq b_t} \exp Q_t^2(b)\right)^2} \tag{39}$$

$$= \tilde{\pi}^2(b_t|s_t)(1 - \tilde{\pi}^2(b_t|s_t)) \nabla_{\theta_1} Q_t^2(b_t). \tag{40}$$

By the chain rule, the gradient of the log probability is

$$\nabla_{\theta_1} \log \tilde{\pi}^2(b_t|s_t) = \frac{\nabla_{\theta_1} \tilde{\pi}^2(b_t|s_t)}{\tilde{\pi}^2(b_t|s_t)} = (1 - \tilde{\pi}^2(b_t|s_t)) \nabla_{\theta_1} Q_t^2(b_t). \tag{41}$$

The difference between LOQA and Advantage Alignment lies in the extra scaling factor $(1-\tilde{\pi}^2(b_t|s_t))$ which accounts for the partition function. Plugging (41) into the generalized policy gradient equation (6) proves the theorem. □

A.5 Advantage Alignment Implementation

To more clearly see the Advantage Alignment formula as an influence over each indicual log probability term recall the formulation:

$$\mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \sum_{k=t+1}^{\infty} \gamma^k A^1(s_t,a_t,b_t) A^2(s_k,a_k,b_k) \nabla_{\theta^1} \log \pi^1(a_k|s_k) \right] \tag{8}$$

Reordering the summations we get:

$$\mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \gamma^t \left(\sum_{k<t} A^1(s_k,a_k,b_k) \right) A^2(s_t,a_t,b_t) \nabla_{\theta^1} \log \pi^1(a_t|s_t) \right] \tag{42}$$

The γ^t term helps regularize the linear scaling of the sum of the advantages of the agent. Alternatively one could regularize using $1/(1+t)$ instead:

$$\mathbb{E}_{\tau \sim \text{Pr}_\mu^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \frac{1}{1+t} \left(\sum_{k<t} A^1(s_k,a_k,b_k) \right) A^2(s_t,a_t,b_t) \nabla_{\theta^1} \log \pi^1(a_t|s_t) \right] \tag{43}$$

Which accounts to increasing the probability of the actions that align the sum of the past advantages of the agent up to the current time-step $t-1$ and the advantage of the opponent at the current time-step, t. In our implementation we use equation (43), as it considers more terms in the future and works better in practice.

A.6 Proximal Advantage Alignment

Recall the Trust Region Policy Optimization (TRPO) (Schulman et al., 2017a) objective, we want to maximize the value function while maintaining the updated policy close in policy space:

$$\begin{aligned} &\max_{\theta^1} V^1(\mu) \\ &\text{s.t. } \sup_s \left\| \pi^1(\cdot|s) - \pi_n^1(\cdot|s) \right\|_{\text{tv}} \leq \delta \end{aligned} \tag{44}$$

Algorithm 2 Proximal Advantage Alignment

```
Initialize: Discount factor $\gamma$, agent Q-value parameters $\phi^1$, t Q-value parameters $\phi^1_{\text{t}}$, actor parameters $\theta^1$, opponent Q-value parameters $\phi^2$, t Q-value parameters $\phi^2_{\text{t}}$, actor parameters $\theta^2$
for iteration= $1, 2, \ldots$ do
    Run policies $\pi^1$ and $\pi^2$ for $T$ timesteps in environment and collect trajectory $\tau$
    Compute agent critic loss $L^1_C$ using the TD error with $r^1$ and $V^1$
    Compute opponent critic loss $L^2_C$ using the TD error with $r^2$ and $V^2$
    Optimize $L^1_C$ w.r.t. $\phi^1$ and $L^2_C$ w.r.t. $\phi^2$ with optimizer of choice
    Optimize $L^1_C$ w.r.t. $\phi^1$ and $L^2_C$ w.r.t. $\phi^2$ with optimizer of choice
    Compute generalized advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$
    Compute agent actor loss, $L^1_a$, summing equations (9) and (45)
    Compute opponent actor loss, $L^2_a$, summing equations (9) and (45)
    Optimize $L^1_a$ w.r.t. $\theta^1$ and $L^2_a$ w.r.t. $\theta^2$ with optimizer of choice
end for
```

We can use the PPO (Schulman et al., 2017b) surrogate objective:

$$\mathbb{E}_{\tau \sim \text{Pr}_{\mu}^{\pi^1,\pi^2}} \left[\min \left\{ r_n(\theta_1) A^1_t,\ \text{clip}\left(r_n(\theta_1); 1-\epsilon, 1+\epsilon\right) A^1_t \right\} \right] \tag{45}$$

We then apply it to the advantage alignment formula (8):

$$\mathbb{E}_{\tau \sim \text{Pr}_{\mu}^{\pi^1,\pi^2}} \left[\sum_{t=0}^{\infty} \sum_{k=t+1}^{\infty} \gamma^k \min \left\{ r_n(\theta_1) A^1_t A^2_k,\ \text{clip}\left(r_n(\theta_1); 1-\epsilon, 1+\epsilon\right) A^1_t A^2_k \right\} \right] \tag{9}$$

Where we denote $\pi^1_n(a_t|s_t)$ to be the updated policy and $r_n(\theta_1) = \pi^1_n(a_t|s_t)/\pi^1(a_t|s_t)$ is the ratio between the updated policy and the old policy. We used generalized advantage estimation (GAE) (Schulman et al., 2018) to compute the advantages in this expression.

B Experimental Details

B.1 Iterated Prisoner's Dilemma

We use an MLP layer connected to a GRU followed by another MLP head for both the actor and critic networks, similar to the architecture used in POLA (Zhao et al., 2022). We also use a replay buffer of agents collected during training, following Aghajohari et al. (2024b). All of our experiments run in 50 minutes in a nvidia A100 gpu.

Table 2: IPD Hyperparameters

Parameter	Value
Actor Training Optimizer	Adam
Actor Training Entropy Beta	0.15
Actor Training Learning Rate (Actor Loss)	0.0001
Advantage Alignment Weight	0.3
Actor Hidden Size	64
Layers Before GRU	1
Q-Value Training Optimizer	Adam
Q-Value Training Learning Rate	0.001
Q-Value Training Target EMA Gamma	0.99
Q-Value Hidden Size	64
Batch Size	2048
Self-Play	True
Reward Discount Factor	0.9
Agent Replay Buffer Capacity	10000
Agent Replay Buffer Update Frequency	1
Agent Replay Buffer Current Agent Fraction	0
Advantage Alignment Discount Factor	0.9

B.2 Coin Game

We use the same architecture used for IPD with an MLP connected to a GRU unit, followed by another MLP. We experimented with both AdAlign (Equation (8)) and Proximal AdAlign (Equation (9)), with AdAlign performing better (this is the one we used). All of our experiments run in 30 minutes in a nvidia A100 gpu.

Table 3: Coin Game Hyperparameters

Parameter	Value
Actor Training Optimizer	Adam
Actor Training Entropy Beta	0.1
Actor Training Learning Rate (Actor Loss)	0.002
Advantage Alignment Weight	0.25
Actor Hidden Size	64
Layers Before GRU	1
Q-Value Training Optimizer	Adam
Q-Value Training Learning Rate	0.005
Q-Value Training Target EMA Gamma	0.99
Q-Value Hidden Size	64
Batch Size	512
Self-Play	True
Reward Discount Factor	0.96
Agent Replay Buffer Capacity	10000
Agent Replay Buffer Update Frequency	10
Agent Replay Buffer Current Agent Fraction	0
Advantage Alignment Discount Factor	0.9

B.3 Negotiation Game

Agent's Architecture: The game observations are a concatenation of the availability of the items, agent's value for each item, opponent's value for each item, and previous round proposals. This makes up for an input vector of length 15. The previous round proposals are especially important as the agents need to examine whether the opponent defected against them by proposing high proposals for item in which the value of the item is higher for the agent compared to the value of the item to the opponent. In other words, if the opponent wanted to gain a little return in exchange of huge loss to the agent, defecting.

Encoder: The observation is then processed by an encoder. The encoder is a GRU network. The GRU network consists of first two Linear Layers with a relu non-linearity in between. Then it is passed to a GRU unit.

Critic: The output of the GRU is then fed to a two-layer MLP with relu non-linearities for the critic module of the agent. Additionally, we concatenate the output of the encoder with the time, the index of the step of the game, for the value function as otherwise it would be hard to estimate the value of the state without knowing how long the game is going to go on for.

Actor: The actor is the most complex component as it deals with continuous actions. The output of the encoder is passed to an MLP with relu non-linearities and the output of the MLP is passed to a $\tanh$ activation and scaled by 2.5, the output of this MLP is used as the mean of a normal distribution. The logarithm of the standard deviation is modeled by a single global parameter in the actor. Next, a sample of this normal distribution is passed through a $\tanh$ activation and scaled and shifted back to $(0, 5)$. Computing the log probability of this transformations requires careful implementation. Especially if the atanh operation that is used is numerically unstable. Please refer to the code released with this paper for the exact implementation.

Hyperparameters: Please refer to 4 for the hyperparameters used in our negotiation game experiments. We use a replay buffer on our gather trajectories although the rate that it is mixed with fresh trajectories is small.

Table 4: Negotiation Game Hyperparameters

Parameter	**Value**
Trajectory Length	50
Encoder Layers	2
MLP Model Layers	2
Replay Buffer Size	100000
Replay Buffer Update Size	500
Replay Buffer Off-policy Ratio	0.05
Optimizer (Actor) Learning Rate	0.001
Optimizer (Critic) Learning Rate	0.001
Entropy Beta	0.005
Advantage Alignment Weight	3.0
Self-Play	True
Batch Size	16384
Gradient Clipping Norm	1.0

Note that the optimization of the agents in the negotiation game is unstable, preventing us from taking the last checkpoint. In our experiments in Fig 4a we select the checkpoint that corresponds to the best achieved return for the agent during the optimization of the agent and the opponent. While we are not completely certain, we observe the instability happens when the policy distribution concentrates around the maximum possible proposal which is 5. We clipped the atanh operation in our implementation for more numerical stability. All of our experiments run in 1 hour on a nvidia A100 gpu.

C Additional Figures

C.1 Negotiation Game Training Curves

Figure 5 shows the training curves of advantage alignments on 10 seeds.

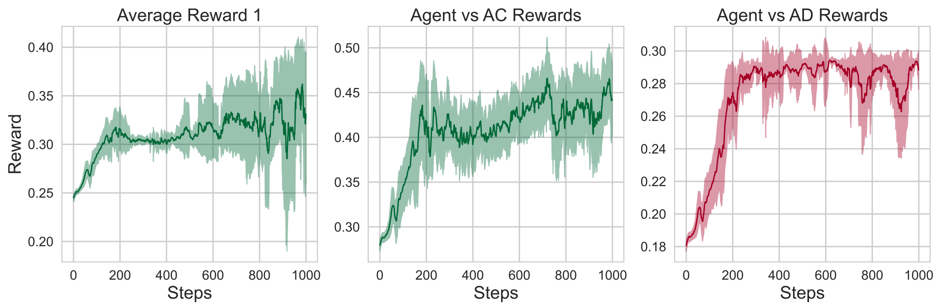

Figure 5: Training curves of Advantage Alignment averaged over 10 seeds.

C.2 Coin Game Full League Results

Figure 6 shows the head-to-head results of all agents we experimented with in a league.

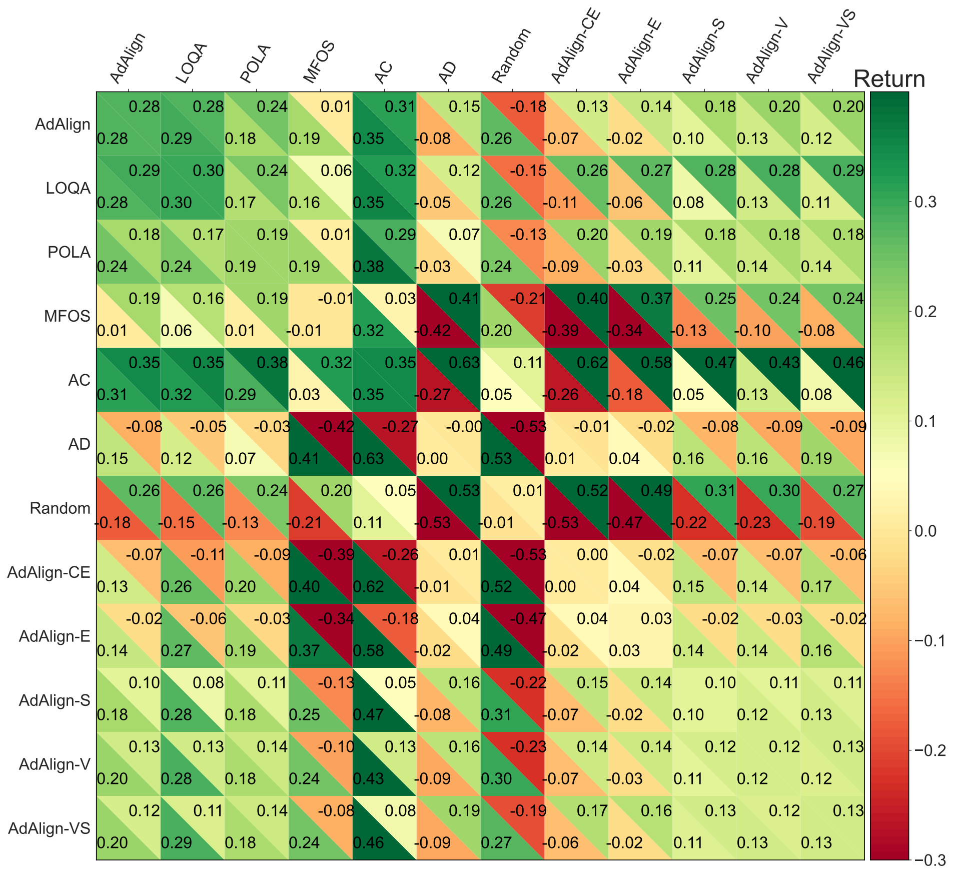

Figure 6: The head-to-head results of all variants of the cion game agents experimented with in this paper. All numbers are an average of 10 seeds of one type of agent with 10 seeds of another type of agent, where each pair play 32 games.