

Adaptive Mechanism Design: Learning to Promote Cooperation

Tobias Baumann and Thore Graepel and John Shawe-Taylor¹

Abstract. In the future, artificial learning agents are likely to become increasingly widespread in our society. They will interact with both other learning agents and humans in a variety of complex settings including social dilemmas. We consider the problem of how an external agent can promote cooperation between artificial learners by distributing additional rewards and punishments based on observing the learners' actions. We propose a rule for automatically learning how to create the right incentives by considering the players' anticipated parameter updates. Using this learning rule leads to cooperation with high social welfare in matrix games in which the agents would otherwise learn to defect with high probability. We show that the resulting cooperative outcome is stable in certain games even if the planning agent is turned off after a given number of episodes, while other games require ongoing intervention to maintain mutual cooperation. However, even in the latter case, the amount of necessary additional incentives decreases over time.

1 INTRODUCTION

Social dilemmas highlight conflicts between individual and collective interests. Cooperation allows for better outcomes for all participants, but individual participants are tempted to increase their own payoff at the expense of others. Selfish incentives can therefore destabilize the socially desirable outcome of mutual cooperation and often lead to outcomes that make everyone worse off [25].

Cooperation often emerges due to direct reciprocity [23] or indirect reciprocity [15]. However, even if these mechanisms are not sufficient on their own, humans are often able to establish cooperation by changing the structure of the social dilemma. This is often referred to as *mechanism design*. For instance, institutions such as the police and the judicial system incentivize humans to cooperate in the social dilemma of peaceful coexistence, and have succeeded in dramatically reducing rates of violence [17].

Studies of social dilemmas have traditionally focused on the context of human agents. However, in the future, artificial learning agents will likely be increasingly widespread in our society, and be employed in a variety of economically relevant tasks. In that case, they will interact both with other artificial agents and humans in complex and partially competitive settings.

This raises the question of how we can ensure that artificial agents will learn to navigate the resulting social dilemmas productively and safely. Failing to learn cooperative policies would lead to socially inefficient or even disastrous outcomes. In particular, the escalation of conflicts between artificial agents (or between artificial agents and humans) may pose a serious security risk in safety-critical systems.

The behaviour of artificial agents in cooperation problems is thus of both theoretical and practical importance.

In this work, we will examine how mechanism design can promote beneficial outcomes in social dilemmas among artificial learners. We consider a setting with N agents in a social dilemma and an additional *planning agent* that can distribute (positive or negative) rewards to the players after observing their actions, and aims to guide the learners to a socially desirable outcome (as measured by the sum of rewards).

We derive a learning rule that allows the planning agent to learn how to set the additional incentives by looking ahead at how the agents will update their policy parameter in the next learning step. We also extend the method to settings in which the planning agent does not know what internal parameters the other agents use and does not have direct access to the opponents' policy.

We evaluate the learning rule on several different matrix game social dilemmas. The planning agent learns to successfully guide the learners to cooperation with high social welfare in all games, while they learn to defect in the absence of a planning agent. We show that the resulting cooperative outcome is stable in certain games even if the planning agent is turned off after a given number of episodes. In other games, cooperation is unstable without continued intervention. However, even in the latter case, we show that the amount of necessary additional rewards decreases over time.

2 RELATED WORK

The study of social dilemmas has a long tradition in game theory, theoretical social science, and biology. In particular, there is a substantial body of literature that fruitfully employs matrix games to study how stable mutual cooperation can emerge [2]. Key mechanisms that can serve to stabilize the socially preferred outcome of mutual cooperation include direct reciprocity [23], indirect reciprocity [15], and norm enforcement [1]. [3] examine how cooperation can be stabilized via supplemental payments from an external party.

Our work is inspired by the field of *mechanism design*, pioneered by [27], which aims to design economic mechanisms and institutions to achieve certain goals, most notably social welfare or revenue maximization. [19] studies how informal and formal incentives for cooperative behaviour can prevent a tragedy of the commons. [13] considers a setting in which an interested party can commit to non-negative monetary transfers, and studies the conditions under which desirable outcomes can be implemented with a given amount of payment. Mechanism design has also been studied in the context of computerized agents [26] and combined with machine learning techniques [14].

We also draw on the rich literature on multi-agent reinforcement learning. It is beyond the scope of this work to review all relevant methods in multi-agent reinforcement learning, so we refer the reader

¹ University College London, UK, email: tobias.baumann.17@ucl.ac.uk

to existing surveys on the subject [4, 24]. However, we note that most work in multi-agent reinforcement learning considers coordination or communication problems in the fully cooperative setting, where the agents share a common goal [16, 6].

As an exception, [9] study the learned behaviour of deep Q-networks in a fruit-gathering game and a Wolfpack hunting game that represent sequential social dilemmas. [22] successfully train agents to play Pong with either a fully cooperative, a fully competitive, or a mixed cooperative-competitive objective. [5] introduce a learning algorithm that uses novel mechanisms for generating and acting on signals to learn to cooperate with humans and with other machines in iterated matrix games. Finally, [11] propose a centralized actor-critic architecture that is applicable to both the fully cooperative as well as the mixed cooperative-competitive setting.

However, these methods assume a given set of opponent policies as given in that they do not take into account how one’s actions affect the parameter updates on other agents. In contrast, [7] introduce Learning with Opponent-Learning Awareness (LOLA), an algorithm that explicitly attempts to shape the opponent’s anticipated learning. The LOLA learning rule includes an additional term that reflects the effect of the agent’s policy on the parameter update of the other agents and inspired the learning rule in this work. However, while LOLA leads to emergent cooperation in an iterated Prisoner’s dilemma, the aim of LOLA agents is to shape the opponent’s learning to their own advantage, which does not always promote cooperation.

3 BACKGROUND

3.1 Markov games

We consider partially observable Markov games [10] as a multi-agent extension of Markov decision processes (MDPs). An N -player Markov game \mathcal{M} is defined by a set of states \mathcal{S} , an observation function $O : \mathcal{S} \times \{1, \dots, N\} \rightarrow \mathbb{R}^d$ specifying each player’s d -dimensional view, a set of actions $\mathcal{A}_1, \dots, \mathcal{A}_N$ for each player, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \rightarrow \mathcal{P}(\mathcal{S})$, where $\mathcal{P}(\mathcal{S})$ denotes the set of probability distributions over \mathcal{S} , and a reward function $r_i : \mathcal{S} \times \mathcal{A}_1 \times \dots \times \mathcal{A}_N \rightarrow \mathbb{R}$ for each player. To choose actions, each player uses a policy $\pi_i : \mathcal{O}_i \rightarrow \mathcal{P}(\mathcal{A}_i)$, where $\mathcal{O}_i = \{o_i \mid s \in \mathcal{S}, o_i = O(s, i)\}$ is the observation space of player i . Each player in a Markov game aims to maximize its discounted expected return $R_i = \sum_{t=0}^T \gamma^t r_i^t$, where γ is a discount factor and T is the time horizon.

3.2 Policy gradient methods

Policy gradient methods [21] are a popular choice for a variety of reinforcement learning tasks. Suppose the policy π_θ of an agent is parametrized by θ . Policy gradient methods aim to maximize the objective $J(\theta) = \mathbb{E}_{s \sim p^{\pi_\theta}, a \sim \pi_\theta} [R]$ by updating the agent’s policy steps in the direction of $\nabla_\theta J(\theta)$.

Using the policy gradient theorem [20], we can write the gradient as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p^{\pi_\theta}, a \sim \pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a)] \quad (1)$$

where p^{π_θ} is the state distribution and $Q^{\pi_\theta}(s, a) = \mathbb{E}[R|s_t = s, a_t = a]$.

3.3 Matrix game social dilemmas

A matrix game is the special case of two-player perfectly observable Markov games with $|\mathcal{S}| = 1$, $T = 1$ and $\mathcal{A}_1 = \mathcal{A}_2 = \{C, D\}$. That

is, two actions are available to each player, which we will interpret as cooperation and defection.

Table 1. Payoff matrix of a symmetric 2-player matrix game. A cell of X, Y represents a utility of X to the row player and Y to the column player.

	C	D
C	R, R	S, T
D	T, S	P, P

Table 1 shows the generic payoff structure of a (symmetric) matrix game. Players can receive four possible rewards: R (reward for mutual cooperation), P (punishment for mutual defection), T (temptation of defecting against a cooperator), and S (sucker outcome of cooperating against a defector).

A matrix game is considered a social dilemma if the following conditions hold [12]:

1. Mutual cooperation is preferable to mutual defection: $R > P$
2. Mutual cooperation is preferable to being exploited: $R > S$
3. Mutual cooperation is preferable to an equal probability of unilateral defection by either player: $R > \frac{T+S}{2}$
4. The players have some reason to defect because exploiting a cooperator is preferable to mutual cooperation ($T > R$) or because mutual defection is preferable to being exploited ($P > S$).

The last condition reflects the mixed incentive structure of matrix game social dilemmas. We will refer to the motivation to exploit a cooperator (quantified by $T - R$) as *greed* and to the motivation to avoid being exploited by a defector ($P - S$) as *fear*. As shown in Table 2, we can use the presence or absence of greed and fear to categorize matrix game social dilemmas.

Table 2. The three canonical examples of matrix game social dilemmas with different reasons to defect. In Chicken, agents may defect out of greed, but not out of fear. In Stag Hunt, agents can never get more than the reward of mutual cooperation by defecting, but they may still defect out of fear of a non-cooperative partner. In Prisoner’s Dilemma (PD), agents are motivated by both greed and fear simultaneously.

Chicken	C	D	Stag Hunt	C	D
C	3, 3	1, 4	C	4, 4	0, 3
D	4, 1	0, 0	D	3, 0	1, 1
PD	C	D			
C	3, 3	0, 4			
D	4, 0	1, 1			

4 METHODS

4.1 Amended Markov game including the planning agent

Suppose N agents play a Markov game described by $\mathcal{S}, \mathcal{A}_1 \dots \mathcal{A}_N, r_1, \dots, r_n, \mathcal{O}$ and \mathcal{T} . We introduce a *planning agent* that can hand out additional rewards and punishments to the players and aims to use this to ensure the socially preferred outcome of mutual cooperation.

To do this, the Markov game can be amended as follows. We add another action set $\mathcal{A}_p \subset \mathbb{R}^N$ that represents which additional rewards and punishments are available to the planning agent. Based on its observation $\mathcal{O}_p : \mathcal{S} \times \{1, \dots, N\} \rightarrow \mathbb{R}^d$ and the

other player's actions a_1, \dots, a_n , the planning agent takes an action $a_p = (r_1^p, \dots, r_N^p) \in \mathcal{A}_p \subset \mathbb{R}^{N \cdot 2}$. The new reward function of player i is $r_i^{\text{tot}} = r_i + r_i^p$, i.e. the sum of the original reward and the additional reward, and we denote the corresponding value functions as $V_i^{\text{tot}}(\theta_1, \dots, \theta_N) = V_i(\theta_1, \dots, \theta_N) + V_i^p(\theta_1, \dots, \theta_N)$. Finally, the transition function \mathcal{T} formally receives a_p as an additional argument, but does not depend on it ($\mathcal{T}(s, a_1, \dots, a_N, a_p) = \mathcal{T}(s, a_1, \dots, a_N)$).

4.2 The learning problem

Let $\theta_1, \dots, \theta_N$ and θ_p be parametrizations of the player's policies π_1, \dots, π_N and the planning agent's policy π_p .

The planning agent aims to maximize the total social welfare $V(\theta_1, \dots, \theta_N) := \sum_{i=1}^N V_i(\theta_1, \dots, \theta_N)$, which is a natural metric of how socially desirable an outcome is. Note that without restrictions on the set of possible additional rewards and punishments, i.e. $\mathcal{A}_p = \mathbb{R}^N$, the planning agent can always transform the game into a fully cooperative game by choosing $r_i^p = \sum_{j=1, j \neq i}^N r_j$.

However, it is difficult to learn how to set the right incentives using traditional reinforcement learning techniques. This is because $V(\theta_1, \dots, \theta_N)$ does not depend *directly* on θ_p . The planning agent's actions only affect $V(\theta_1, \dots, \theta_N)$ indirectly by changing the parameter updates of the learners. For this reason, it is vital to explicitly take into account how the other agents' learning changes in response to additional incentives.

This can be achieved by considering the next learning step of each player (cf. [7]). We assume that the learners update their parameters by simple gradient ascent:

$$\begin{aligned} \Delta\theta_i &= \eta_i \nabla_i V_i^{\text{tot}}(\theta_1, \dots, \theta_N) \\ &= \eta_i (\nabla_i V_i(\theta_1, \dots, \theta_N) + \nabla_i V_i^p(\theta_1, \dots, \theta_N)) \end{aligned} \quad (2)$$

where η_i is step size of player i and $\nabla_i := \nabla_{\theta_i}$ is the gradient with respect to parameters θ_i .

Instead of optimizing $V(\theta_1, \dots, \theta_N)$, the planning agent looks ahead one step and maximizes $V(\theta_1 + \Delta\theta_1, \dots, \theta_N + \Delta\theta_N)$. Assuming that the parameter updates $\Delta\theta_i$ are small, a first-order Taylor expansion yields

$$\begin{aligned} V(\theta_1 + \Delta\theta_1, \dots, \theta_N + \Delta\theta_N) &\approx \\ &\approx V(\theta_1, \dots, \theta_N) + \sum_{i=1}^N (\Delta\theta_i)^T \nabla_i V(\theta_1, \dots, \theta_N) \end{aligned} \quad (3)$$

We use a simple rule of the form $\Delta\theta_p = \eta_p \nabla_p V(\theta_1 + \Delta\theta_1, \dots, \theta_N + \Delta\theta_N)$ to update the planning agent's policy, where η_p is the learning step size of the planning agent. Exploiting the fact that $V(\theta_1, \dots, \theta_N)$ does not depend directly on θ_p , i.e. $\nabla_p V(\theta_1, \dots, \theta_N) = 0$, we can

calculate the gradient:

$$\begin{aligned} \nabla_p V(\theta_1 + \Delta\theta_1, \dots, \theta_N + \Delta\theta_N) &\approx \\ &\approx \sum_{i=1}^N \nabla_p (\Delta\theta_i)^T \nabla_i V(\theta_1, \dots, \theta_N) \\ &= \sum_{i=1}^N \eta_i (\nabla_p \nabla_i V_i^{\text{tot}}(\theta_1, \dots, \theta_N))^T \nabla_i V(\theta_1, \dots, \theta_N) \\ &= \sum_{i=1}^N \eta_i (\nabla_p \nabla_i V_i^p(\theta_1, \dots, \theta_N))^T \nabla_i V(\theta_1, \dots, \theta_N) \end{aligned} \quad (4)$$

since $\nabla_i V_i(\theta_1, \dots, \theta_N)$ does not depend on θ_p either.

4.3 Policy gradient approximation

If the planning agent does not have access to the exact gradients of $V_i^p(\theta_1, \dots, \theta_N)$ and $V(\theta_1, \dots, \theta_N)$, we use policy gradients as an approximation. Let $\tau = (s_0, \mathbf{a}^0, a_p^0, \mathbf{r}^0, \dots, s_T, \mathbf{a}^T, a_p^T, \mathbf{r}^T)$ be a state-action trajectory of horizon $T + 1$, where $\mathbf{a}^t = (a_1^t, \dots, a_N^t)$, $\mathbf{r}^t = (r_1^t, \dots, r_N^t)$, and $a_p^t = (r_{1,p}^t, \dots, r_{N,p}^t)$ are the actions taken and rewards received in time step t . Then, the episodic return $R_i^0(\tau) = \sum_{t=0}^T \gamma^t r_i^t$ and $R_{i,p}^0(\tau) = \sum_{t=0}^T \gamma^t r_{i,p}^t$ approximate $V_i(\theta_1, \dots, \theta_N)$ and $V_i^p(\theta_1, \dots, \theta_N)$, respectively. Similarly, $R^0(\tau) = \sum_{i=0}^N R_i^0(\tau)$ approximates the social welfare $V(\theta_1, \dots, \theta_N)$.

We can now calculate the gradients using the policy gradient theorem:

$$\begin{aligned} \nabla_i V_i(\theta_1, \dots, \theta_N) &\approx \nabla_i \mathbb{E}[R_i^0(\tau)] \\ &= \mathbb{E}[\nabla_i \log \pi_i(\tau) R_i^0(\tau)] \end{aligned} \quad (5)$$

The other gradients $\nabla_i V(\theta_1, \dots, \theta_N)$ and $\nabla_p \nabla_i V_i^p(\theta_1, \dots, \theta_N)$ can be approximated in the same way. This yields the following rule for the parameter update of the planning agent:

$$\begin{aligned} \Delta\theta_p &= \eta_p \sum_{i=1}^N \eta_i \left(\mathbb{E}[\nabla_p \log \pi_p(\tau) \nabla_i \log \pi_i(\tau) R_{i,p}^0(\tau)] \right)^T \\ &\quad \cdot \mathbb{E}[\nabla_i \log \pi_i(\tau) R^0(\tau)] \end{aligned} \quad (6)$$

4.4 Opponent modeling

Equations 4 and 6 assume that the planning agent has access to each agent's internal policy parameters and gradients. This is a restrictive assumption. In particular, agents may have an incentive to conceal their inner workings in adversarial settings. However, if the assumption is not fulfilled, we can instead model the opponents' policies using parameter vectors $\hat{\theta}_1, \dots, \hat{\theta}_N$ and infer the value of these parameters from the player's actions [18]. A simple approach is to use a maximum likelihood estimate based on the observed trajectory:

$$\hat{\theta}_i = \arg \max_{\theta_i'} \sum_{t=0}^T \log \pi_{\theta_i'}(a_i^t | s_t). \quad (7)$$

Given this, we can substitute $\hat{\theta}_i$ for θ_i in equation 4.

4.5 Cost of additional rewards

In real-world examples, it may be costly to distribute additional rewards or punishment. We can model this cost by changing the planning agent's objective to $V(\theta_1 + \Delta\theta_1, \dots, \theta_N + \Delta\theta_N) -$

² Technically, we could represent the dependence on the other player's actions by introducing an extra step after the regular step in which the planning agent chooses additional rewards and punishments. However, for simplicity, we will discard this and treat the player's actions and the planning action as a single step. Formally, we can justify this by letting the planning agent specify its action for every possible combination of player actions.

$\alpha \|V^p(\theta_1, \dots, \theta_N; \theta_p)\|_2$, where α is a cost parameter and $V^p = (V_1^p, \dots, V_N^p)$. The modified update rule is (using equation 4)

$$\Delta\theta_p = \eta_p \left(\sum_{i=1}^N \eta_i (\nabla_p \nabla_i V_i^p(\theta_1, \dots, \theta_N))^T \nabla_i V(\theta_1, \dots, \theta_N) - \alpha \nabla_p \|V^p(\theta_1, \dots, \theta_N; \theta_p)\|_2 \right) \quad (8)$$

5 EXPERIMENTAL SETUP

In our experiments, we consider $N = 2$ learning agents playing a matrix game social dilemma (MGSD) as outlined in section 3.3. The learners are simple agents with a single policy parameter θ that controls the probability of cooperation and defection: $P(C) = \frac{\exp(\theta)}{1+\exp(\theta)}$, $P(D) = \frac{1}{1+\exp(\theta)}$. The agents use a centralized critic [11] to learn their value function.

The agents play 4000 episodes of a matrix game social dilemma. We fix the payoffs $R = 3$ and $P = 1$, which allows us to describe the game using the level of greed and fear. We will consider three canonical matrix game social dilemmas as shown in Table 3.

Table 3. Levels of fear and greed and resulting temptation (T) and sucker (S) payoffs in three matrix games. Note that the level of greed in Chicken has to be smaller than 1 because it is otherwise not a social dilemma ($R > \frac{T+S}{2}$ is not fulfilled).

Game	Greed	Fear	T	S
Prisoner’s Dilemma	1	1	4	0
Chicken	0.5	-1	3.5	2
Stag Hunt	-1	1	2	0

The planning agent’s policy is parametrized by a single layer neural network. We limit the maximum amount of additional rewards or punishments (i.e. we restrict \mathcal{A}_p to vectors that satisfy $\max_{i=1}^N |r_i^p| \leq c$ for a given constant c). Unless specified otherwise, we use a step size of 0.01 for both the planning agent and the learners, use cost regularization (Equation 8) with a cost parameter of 0.0002, set the maximum reward to 3, and use the exact value function. In some experiments, we also require that the planning agent can only redistribute rewards, but cannot change the total sum of rewards (i.e. \mathcal{A}_p is restricted to vectors that satisfy $\sum_{i=1}^N r_i^p = 0$). We refer to this as the *revenue-neutral* setting.

6 RESULTS

In this section, we summarize the experimental results.³ We aim to answer the following questions:

- Does the introduction of the planning agent succeed in promoting significantly higher levels of cooperation?
- What qualitative conclusions can be drawn about the amount of additional incentives needed to learn and maintain cooperation?
- In which cases is it possible to achieve cooperation even when the planning agent is only active for a limited timespan?
- How does a restriction to revenue-neutrality affect the effectiveness of mechanism design?

Figure 1a illustrates that the players learn to cooperate with high probability if the planning agent is present, resulting in the socially

³ Source code available at <https://github.com/tobiasbaumann1/Adaptive-Mechanism-Design>

preferred outcome of stable mutual cooperation. Thus the planning agent successfully learns how to distribute additional rewards to guide the players to a better outcome.

Figure 1b shows how the planning agent rewards or punishes the player conditional on each of the four possible outcomes. At first, the planning agent learns to reward cooperation, which creates a sufficient incentive to cause the players to learn to cooperate. In Figure 1c we show how this changes the level of fear and greed in the modified game. The levels of greed and fear soon drop below zero, which means that the modified game is no longer a social dilemma.

Note that rewarding cooperation is less costly than punishing defection if (and only if) cooperation is the less common action. After the player learns to cooperate with high probability, the planning agent learns that it is now less costly to punish defection and consequently stops handing out additional rewards in the case of mutual cooperation outcome. As shown in Figure 1d, the amount of necessary additional rewards converges to 0 over time as defection becomes increasingly rare.

Table 4 summarizes the results of all three canonical social dilemmas. Without adaptive mechanism design, the learners fail to achieve mutual cooperation in all cases. By contrast, if the planning agent is turned on, the learners learn to cooperate with high probability, resulting in a significantly higher level of social welfare.

Table 4. Comparison of the resulting levels of cooperation after 4000 episodes, a) without mechanism design, b) with mechanism design, and c) when turning off the planning agent after 4000 episodes and running another 4000 episodes. Each cell shows the mean and standard deviation of ten training runs. $P(C, C)$ is the probability of mutual cooperation at the end of training and V is the expected social welfare that results from the players’ final action probabilities. The initial probability of cooperation is 0.25 for each player.

		Prisoner’s Dilemma	Chicken	Stag Hunt
Greed		1	0.5	-1
Fear		1	-1	1
No mech. design	$P(C, C)$	0.004% ±0.001%	3.7% ±1.3%	0.004% ±0.002%
	V	2.024 ±0.003	5.44 ±0.01	2.00 ±0.00
With mech. design	$P(C, C)$	98.7% ±0.1%	99.0% ±0.1%	99.1% ±0.1%
	V	5.975 ±0.002	5.995 ±0.001	5.964 ±0.005
Turning off	$P(C, C)$	0.48% ±0.4%	53.8% ±29.4%	99.6% ±0.0%
	V	2.60 ±0.69	5.728 ±0.174	5.986 ±0.002

The three games differ, however, in whether the cooperative outcome obtained through mechanism design is stable even when the planning agent is turned off. Without additional incentives, mutual cooperation is not a Nash equilibrium in the Prisoner’s Dilemma and in Chicken [8], which is why one or both players learn to defect again after the planning agent is turned off. These games thus require continued (but only occasional) intervention to maintain cooperation. By contrast, mutual cooperation is a stable equilibrium in Stag Hunt [8]. As shown in Table 4, this means that long-term cooperation in Stag Hunt can be achieved even if the planning agent is only active over a limited timespan (and thus at limited cost).

Table 5 compares the performance of different variants of the learning rule. Interestingly, restricting the possible planning actions to redistribution leads to lower probabilities of cooperation in Prisoner’s

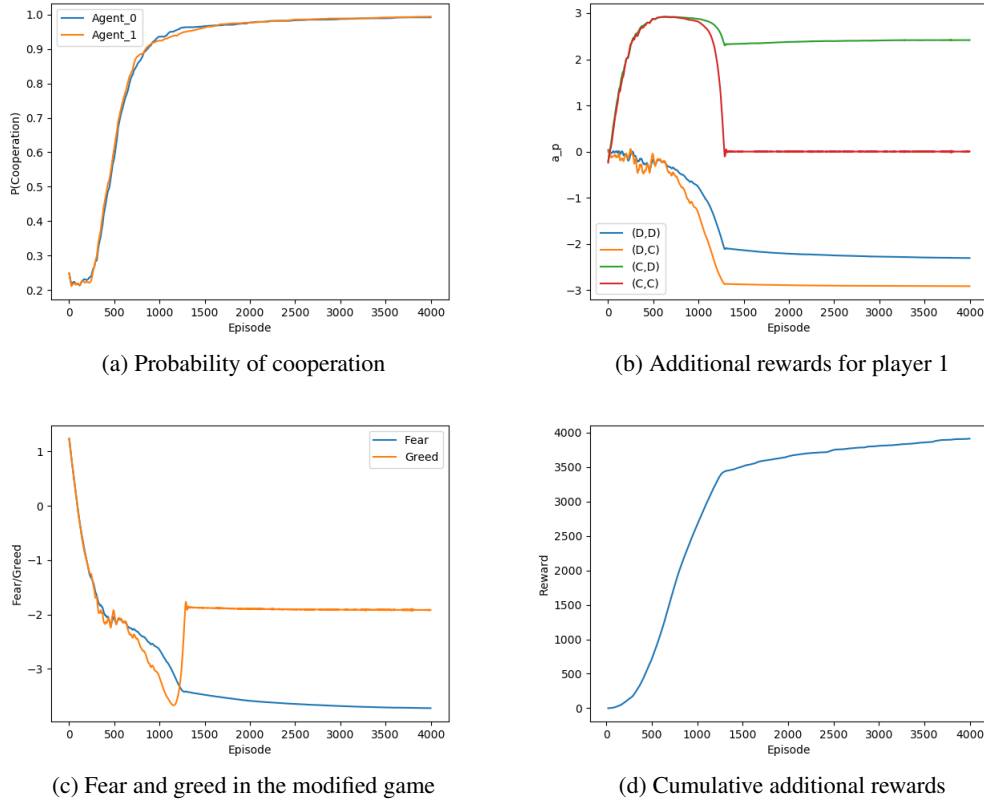


Figure 1. Mechanism design over 4000 episodes of a Prisoner's Dilemma. The initial probability of cooperation is 0.25 for each player. Shown is (a) the probability of cooperation over time, (b) the additional reward for the first player in each of the four possible outcomes, (c) the resulting levels of fear and greed including additional rewards, and (d) the cumulative amount of distributed rewards.

Table 5. Resulting levels of cooperation and average additional rewards (AAR) per round for different variants of the learning rule. The variants differ in whether they use the exact value function (Equation 4) or an estimate (Equation 6) and in whether the setting is revenue-neutral or unrestricted.

	Prisoner's Dilemma	Chicken	Stag Hunt
Greed	1	0.5	-1
Fear	1	-1	1
Exact V			
$P(C, C)$	98.7% $\pm 0.1\%$	99.0% $\pm 0.1\%$	99.1% $\pm 0.1\%$
AAR	0.77 ± 0.21	0.41 ± 0.02	0.45 ± 0.02
Exact V Revenue-neutral			
$P(C, C)$	91.4% $\pm 1.0\%$	98.9% $\pm 0.1\%$	69.2% $\pm 45.3\%$
AAR	0.61 ± 0.04	0.31 ± 0.02	0.19 ± 0.11
Estimated V			
$P(C, C)$	61.3% $\pm 20.0\%$	52.2% $\pm 18.6\%$	96.0% $\pm 1.2\%$
AAR	3.31 ± 0.63	2.65 ± 0.31	4.89 ± 0.39

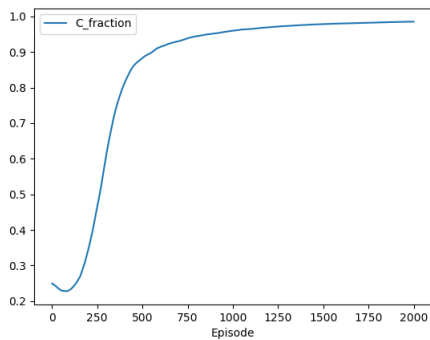
Dilemma and Stag Hunt, but not in Chicken. We hypothesize that this is because in Chicken, mutual defection is not in the individual interest of the players anyway. This means that the main task for the planning agent is to prevent (C,D) or (D,C) outcomes, which can be easily achieved by redistribution. By contrast, these outcomes are fairly unattractive (in terms of individual interests) in Stag Hunt, so the most effective intervention is to make (D,D) less attractive and (C,C) more attractive, which is not feasible by pure redistribution. Consequently, mechanism design by redistribution works best in Chicken and worst in Stag Hunt.

Using an estimate of the value function leads to inferior performance on all three games, both in terms of the resulting probability of mutual cooperation and with respect to the amount of distributed additional results. However, the effect is by far least pronounced in Stag Hunt. This may be because mutual cooperation is an equilibrium in Stag Hunt, which means that a beneficial outcome can more easily arise even if the incentive structure created by the planning agent is imperfect.

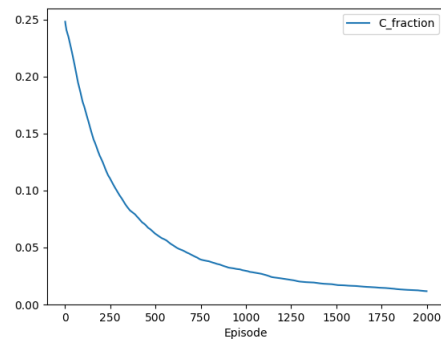
Finally, we note that the presented approach is also applicable to settings with more than two players.⁴ We consider a multi-player Prisoner's Dilemma with $N = 10$ agents.⁵ Figure 2a illustrates that,

⁴ Source code available in a separate repository at https://github.com/tobiasbaumann1/Mechanism_Design_Multi-Player

⁵ The payoffs are as follows: 3 if all players cooperate, 1 if all players defect, 4 if you are the only to defect, 0 if you are the only to cooperate. Payoffs



(a) Average probability of cooperation with mechanism design



(b) Average probability of cooperation without mechanism design

Figure 2. Mechanism design in a multi-player Prisoner’s Dilemma. The initial probability of cooperation is 0.25 for each player. Shown is the average probability of cooperation over time (a) in the presence of a planning agent, (b) without mechanism design.

just as in the case of $N = 2$, the players learn to cooperate with high probability if the planning agent is present. By contrast, without mechanism design, the players (unsurprisingly) converge to the socially undesirable outcome of mutual defection. This shows that the presented approach for learning how to distribute additional rewards scales easily to multi-agent social dilemmas.

7 CONCLUSIONS AND FUTURE WORK

We have presented a method for learning how to create the right incentives to ensure cooperation between artificial learners. Empirically, we have shown that a planning agent that uses the proposed learning rule is able to successfully guide the learners to the socially preferred outcome of mutual cooperation in several different matrix game social dilemmas, while they learn to defect with high probability in the absence of a planning agent. The resulting cooperative outcome is stable in certain games even if the planning agent is turned off after a given number of episodes, while other games require continued (but increasingly rare) intervention to maintain cooperation. We also showed that restricting the planning agent to redistribution leads to worse performance in Stag Hunt, but not in Chicken.

In the future, we would like to explore the limitations of adaptive mechanism design in more complex environments, particularly in games with more than two players, without full observability of the players’ actions, and using opponent modeling (cf. Equation 7). Future work could also consider settings in which the planning agent aims to ensure cooperation by altering the dynamics of the environment or the players’ action set (e.g. by introducing mechanisms that allow players to better punish defectors or reward cooperators).

Finally, under the assumption that artificial learners will play vital roles in future society, it is worthwhile to develop policy recommendations that would facilitate mechanism design for these agents (and the humans they interact with), thus contributing to a cooperative outcome in potential social dilemmas. For instance, it would be helpful if the agents were set up in a way that makes their intentions as transparent as possible and allows for simple ways to distribute additional rewards and punishments without incurring large costs.

of intermediate outcomes, where some fraction of players cooperate, are obtained by linear interpolation.

REFERENCES

- [1] Robert Axelrod, ‘An Evolutionary Approach to Norms’, *American Political Science Review*, (1986).
- [2] Robert Axelrod and William D. Hamilton, ‘The Evolution of Cooperation’, *Evolution*, (1981).
- [3] Yoram Bachrach, Edith Elkind, Reshef Meir, Dmitrii Pasechnik, Michael Zuckerman, Jörg Rothe, and Jeffrey S Rosenschein, ‘The cost of stability in coalitional games’, in *International Symposium on Algorithmic Game Theory*, pp. 122–134. Springer, (2009).
- [4] Lucian Busoni, Robert Babuska, and Bart De Schutter, ‘A Comprehensive Survey of Multiagent Reinforcement Learning’, *Systems, Man, and Cybernetics, Part C: Applications and Reviews*, (2008).
- [5] Jacob W. Crandall, Mayada Oudah, Fatimah Ishowo-Oloko, Sherief Abdallah, Jean-François Bonnefon, et al., ‘Cooperating with machines’, *Nature communications*, **9**(1), 233, (2018).
- [6] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson, ‘Learning to Communicate with Deep Multi-Agent Reinforcement Learning’, 2137–2145, (2016).
- [7] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch, ‘Learning with Opponent-Learning Awareness’, (2017).
- [8] Drew Fudenberg and Jean Tirole, *Game Theory*, MIT Press, Cambridge, MA, 1991.
- [9] Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel, ‘Multi-agent Reinforcement Learning in Sequential Social Dilemmas’, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, (2017).
- [10] Michael L. Littman, ‘Markov games as a framework for multi-agent reinforcement learning’, in *Machine Learning Proceedings 1994*, (1994).
- [11] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch, ‘Multi-agent actor-critic for mixed cooperative-competitive environments’, in *Advances in Neural Information Processing Systems*, pp. 6382–6393, (2017).
- [12] Michael W. Macy and Andreas Flache, ‘Learning Dynamics in Social Dilemmas’, *Proceedings of the National Academy of Sciences of the United States of America*, (2002).
- [13] Dov Monderer and Moshe Tennenholtz, ‘k-implementation’, *Journal of Artificial Intelligence Research*, **21**, 37–62, (2004).
- [14] Harikrishna Narasimhan, Shivani Brinda Agarwal, and David C Parkes, ‘Automated mechanism design without money via machine learning’, (2016).
- [15] Martin A. Nowak and Karl Sigmund. Evolution of Indirect Reciprocity, 2005.
- [16] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian, ‘Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability’, (2017).
- [17] Steven Pinker, ‘The Better Angels of Our Nature’, (2011).
- [18] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell, ‘A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning’, (2010).

- [19] Paul Seabright, 'Managing Local Commons: Theoretical Issues in Incentive Design', *Journal of Economic Perspectives*, **7**(4), 113–134, (1993).
- [20] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour, 'Policy gradient methods for reinforcement learning with function approximation', 1057–1063, (2000).
- [21] RS Sutton and AG Barto, *Reinforcement learning: An introduction*, 1998.
- [22] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente, 'Multiagent cooperation and competition with deep reinforcement learning', *PLoS ONE*, (2017).
- [23] Robert L. Trivers, 'The Evolution of Reciprocal Altruism', *The Quarterly Review of Biology*, (1971).
- [24] Karl Tuyls and Gerhard Weiss, 'Multiagent Learning: Basics, Challenges, and Prospects', *AI Magazine*, (2012).
- [25] Paul A. M. Van Lange, Jeff Joireman, Craig D. Parks, and Eric Van Dijk, 'The Psychology of Social Dilemmas: A Review', *Organizational Behavior and Human Decision Processes*, **120**(2), 125–141, (2013).
- [26] Hal R. Varian, 'Economic mechanism design for computerized agents.', in *USENIX workshop on Electronic Commerce*, pp. 13–21, (1995).
- [27] William Vickrey, 'Counterspeculation, Auctions, and Competitive Sealed Tenders', *The Journal of Finance*, **16**(1), 8–37, (1961).