Learning to Incentivize Other Learning Agents

Jiachen Yang*
Georgia Institute of Technology
jiachen.yang@gatech.edu

Ang Li
DeepMind
anglili@google.com

Mehrdad Farajtabar
DeepMind
farajtabar@google.com

Peter Sunehag
DeepMind
sunehag@google.com

Edward Hughes
DeepMind
edwardhughes@google.com

Hongyuan Zha
Georgia Institute of Technology
zha@cc.gatech.edu

ABSTRACT

The challenge of developing powerful and general Reinforcement Learning (RL) agents has received increasing attention in recent years. Much of this effort has focused on the single-agent setting, in which an agent maximizes a predefined extrinsic reward function. However, a long-term question inevitably arises: how will such independent agents cooperate when they are continually learning and acting in a shared multi-agent environment? Observing that humans often provide incentives to influence others' behavior, we propose to equip each RL agent in a multi-agent environment with the ability to give rewards directly to other agents, via a learned incentive function. Each agent learns the incentive function by considering how the given incentives affect its own extrinsic objective, through the learning of agents who receive incentives. We demonstrate in experiments that such agents significantly outperform policy gradient agents and opponent-shaping agents in a small yet challenging general-sum Markov game. Our work points toward a more general research program of endowing agents with expanded capabilities for incentivizing others to ensure the common good in a multi-agent future.

KEYWORDS

multi-agent; reinforcement learning; incentives; cooperation

1 INTRODUCTION

Reinforcement Learning (RL) [31] agents are achieving increasing success on an expanding set of tasks [3, 15, 21, 26, 34]. While much effort is devoted to single-agent environments and fully-cooperative games, there is a possible future in which large numbers of RL agents with imperfectly-aligned objectives must interact and continually learn in a shared multi-agent environment. Excluding the option of training a fully centralized policy using a global reward, which does not scale to large populations, there is no guarantee that groups of agents can achieve high individual and collective return [23]. Moreover, agents in many real world situations with mixed motives may face a social dilemma, wherein mutual selfish behavior leads to low individual and total utility, due to fear of being exploited or greed to exploit others [17, 18, 24]. Whether, and how, independent learning and acting agents can achieve cooperation while optimizing their own objectives remains an open question.

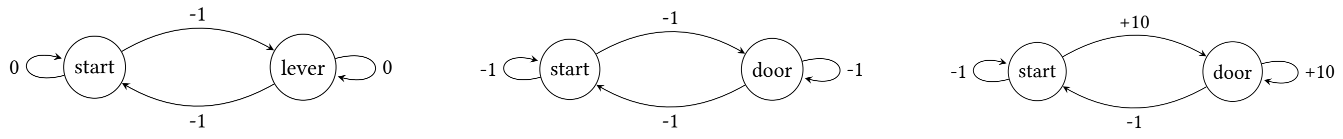
The conundrum of attaining multi-agent cooperation with decentralized[1] training requires us to go beyond the restrictive mindset that the collection of predefined individual objectives cannot be changed by the agents themselves. We draw inspiration from the observation that this fundamental multi-agent problem arises at multiple scales of human activity and, crucially, that it can be successfully resolved when agents give the right incentives to *alter* the objective of other agents, in such a way that the recipients' behavior changes for everyone's advantage. Indeed, a significant amount of individual, group, and international effort is expended on creating effective incentives or sanctions to shape the behavior of other individuals, social groups, and nations [4, 5, 33]. The rich body of work on understanding the game-theoretic aspects of side payments [10, 12, 14] attests to the importance of inter-agent incentivization in the real world.

Translated to the framework of Markov games for multi-agent reinforcement learning (MARL) [20], the key insight is to remove the constraints of an immutable reward function. Instead, we allow agents to *learn* an incentive function that gives rewards to other learning agents, who now themselves learn using the combination of received incentives and predefined extrinsic rewards. The learning problem for a generic agent, who both gives and receives incentives, becomes two-fold: learn a behavioral policy that optimizes the total extrinsic rewards and incentives it receives, and learn a reward function to give the correct incentives to optimize its original extrinsic objective. While the emergence of incentives in nature may have an evolutionary explanation [11], human societies contain ubiquitous examples of learnt incentivization and we focus on the learning viewpoint in this work.

Learning to incentivize other learning agents expands the capabilities of agents and poses significant new research challenges for multi-agent learning. The effect of incentives manifests in the recipient's behavior only after a sufficient number of learning updates by the recipient. Hence, in an episodic Markov game setting, a reward-giver may not receive any feedback within an episode, much less an immediate feedback, on whether a given reward is beneficial to its own extrinsic objective. This implies that merely augmenting an agent's action space with a "give-reward" action and falling back to conventional reinforcement learning is not the best approach. Furthermore, mistakes in giving rewards has long-term consequences on other learning agents' behavior, and it may be hard to steer the recipient back to the right behavior.

As a first step toward addressing these challenges, we make the following technical and experimental contributions. (1) We create an agent that learns an incentive function to reward other learning agents by explicitly accounting for the impact of given incentives on its own performance, through the learning of recipients. Each agent learns two components: a standard policy that is optimized

*Work done at DeepMind.
[1]We take decentralized to mean that no agent is designed with an objective to maximize collective performance, and that agents optimize separate sets of policy parameters.



(a) Agent A2 is penalized for any change of state, if not receiving reward from A1.

(b) Agent A1 is penalized at every step if A2 does not pull the lever.

(c) A1 get +10 and terminates the episode by going to the door if A2 pulls the lever.

Figure 1: An "escape room" game involving two agents, A1 and A2. (a) In the absence of incentives, A2's optimal policy is to stay at the start state and not pull the lever. (b) Hence A1 cannot exit the door and is penalized at every step. (c) A1 can receive positive reward if it learns to incentivize A2 to pull the lever. Giving incentives is not an action depicted here.

via RL, and a new vector-valued incentive function that is optimized directly by gradient ascent on its extrinsic objective. (2) Working with the concrete case where agents update their policies via policy gradient, we derive the gradient of an agent's extrinsic objective with respect to the parameters of its learned incentive function. (3) We create a simple synthetic Markov game, which standard policy-gradient agents, even if augmented with reward-giving actions, were unable to solve, and on which existing opponent-shaping agents do not consistently achieve optimal performance. In contrast, our agent approaches the global optimum in both the asymmetric case where only one agent gives rewards, and the symmetric case where all agents give and receive rewards.

2 RELATED WORK

Learning to incentivize other learning agents is motivated by the problem of cooperation among independent learning agents in intertemporal social dilemmas (ISDs) [17], in which defection is preferable to individuals in the short term but mutual defection leads to low collective performance in the long term. While algorithms for fully-cooperative MARL have shown promise in complex games [9, 25, 29, 37], ISDs have mixed motives, and cannot canonically be reduced to fully cooperative problems. Previous work showed that independent agents who employ *intrinsic* rewards are able to improve collective performance in ISDs [6, 13, 35]. These intrinsic rewards are either hand-crafted or slowly evolved and are used by each agent to modulate its own extrinsic reward. In contrast, the incentive function in our work is *learned* by a reward-giver on the same timescale as policy learning and is given to, and maximized by, *other* agents.

Learning to incentivize is complementary to existing work on opponent shaping in MARL, in which an agent learns to influence the learning update of other agents for its own benefit. While Learning with Opponent Learning Awareness (LOLA) [8] and Stable Opponent Shaping (SOS) [19] only influences other agents via actions taken by its policy, whose effects manifest through the Markov game state transition, our proposed agent exerts direct influence via an incentive function, which is distinct from its policy and which explicitly affects the recipient agent's learning update. Hence the need to influence other agents does not restrict a reward-giver's policy, potentially allowing for more flexible and stable shaping. We describe the mathematical differences between our method and LOLA in Section 4.1, and experimentally compare with LOLA agents augmented with reward-giving actions.

Our work is related to a growing collection of work on modifying or learning a reward function that is in turn maximized by another learning algorithm [2, 28, 38]. Previous work employed a centralized operator on utilities for 2-player games with side payments [28], and employed an additional centralized agent who directly optimizes collective reward by giving extra rewards to players in a 2-player matrix game [2]. In contrast, we work in the general N-player setting where the collective performance cannot be optimized directly and agents themselves must learn to incentivize other agents. The technical approach in our work is inspired by online cross validation [30], which is employed to optimize hyperparameters in meta-gradient RL [36], and by the optimal reward framework [27], in which a single agent learns an intrinsic reward by ascending the gradient of its own extrinsic objective [38].

3 THE ESCAPE ROOM GAME

We may illustrate the benefits of incentivization with a simple example. The 2-player *Escape Room* game is a finite state finite action Markov game with individual extrinsic rewards between two learning agents (A1 and A2) as described in Figure 1. A1 gets +10 extrinsic reward for exiting a door and ending the game (Figure 1c), but the door can only be opened when A2 pulls a lever; otherwise, A1 is penalized at every time step (Figure 1b). However, the extrinsic reward for A2 discourages it from taking the cooperative action (Figure 1a). If both A1 and A2 are standard independent RL agents who only optimize their individual rewards, A2 converges to a policy of not moving, which traps A1 at the global minimum, as we show in Section 6.

Suppose we augment A1's action space with an additional "give reward" action—we allow it to give +2 reward to A2, at a cost -2 to itself—and let it observe A2's chosen action prior to taking its own action. In principle, assuming that A2 conducts sufficient exploration, an intelligent reward-giver can learn to use the +2 reward to incentivize A2 to pull the lever. However, we hypothesize and find evidence in experiments that standard RL faces significant difficulty in learning to incentivize correctly. RL optimizes the expected cumulative reward within an episode, but the value of a reward-giving action can only be measured after *many* episodes of the recipient's learning. Instead, we need an agent that explicitly accounts for the impact of given rewards on the recipient's learning, and the effect of such learning on its own future performance.

4 LEARNING TO INCENTIVIZE OTHERS

We design Learning to Incentivize Others (LIO), an agent that learns an incentive function by explicitly accounting for the impact of given rewards on its own extrinsic objective, through the learning of reward recipients. We build on the idea of online cross-validation [32], to reflect the fact that an incentive has measurable effect only after a recipient's learning update step. For clarity of exposition, we work with the ideal limit where agents have a perfect model of other agents' parameters and gradients; this could be weakened in practice by having each agent estimate a model of other agent's behavior. We formally present the general case where all N agents are LIO agents; see Algorithm 1.

Consider a population of N agents, each indexed by $i \in [N] := \{1, \dots, N\}$, who can give and receive rewards from one another. For clarity, we use index i when referring to the reward-giving part of an agent, and we use j for the part of an agent that learns from received rewards. Let $-i$ denote a collection of all indices except i. Let $\mathbf{a}$ and $\boldsymbol{\pi}$ denote the joint action and the joint policy over all agents, respectively. Let $o^i := O^i(s) \in O$ denote the local observation of agent i at global state s.

A reward-giver agent i optimizes a vector-valued incentive function $r_{\eta^i} : O \times \mathcal{A}^{-i} \mapsto \mathbb{R}^{N-1}$, parameterized by $\eta^i \in \mathbb{R}^n$, that maps its own observation o^i and all other agents' actions a^{-i} to a vector of rewards for the other $N-1$ agents. We do not allow an agent to reward itself. Let $r^j_{\eta^i}$ denote the reward that agent i gives to agent j, i.e., the j-th component of r_{η^i}. Note that r_{η^i} is neither part of the agent's policy nor a separate policy. It is a deterministic function that is learned via direct gradient ascent on the agent's own extrinsic objective, involving its effect on all other agents' policies (as we elaborate below), instead of via RL. These incentive function updates are separate from the agent's policy updates, which we chose to be policy gradient ascent. Hence we are not merely augmenting the agent's action space with a "give-reward" action.

At each time step t, each recipient j receives a total reward

$$r^j(s_t, \mathbf{a}_t, \eta^{-j}) := r^{j,\text{env}}(s_t, \mathbf{a}_t) + \sum_{i \neq j} r^j_{\eta^i}(o^i_t, a^{-i}_t), \tag{1}$$

where $r^{j,\text{env}}$ denotes agent j's extrinsic reward. Each agent j optimizes its policy π^j, parameterized by $\theta^j \in \mathbb{R}^m$, to maximize

$$J^{\text{policy}}(\theta^j, \eta^{-j}) := \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{T} \gamma^t r^j(s_t, \mathbf{a}_t, \eta^{-j})\right]. \tag{2}$$

Upon experiencing a trajectory $\tau^j := (s_0, \mathbf{a}_0, r^j_0, \dots, s_T)$, the recipient carries out an update

$$\hat{\theta}^j \leftarrow \theta^j + \beta f(\tau, \theta^j, \eta^{-j}) \tag{3}$$

that adjusts parameters $\theta^j \in \mathbb{R}^m$ of its policy π^j (Algorithm 1, lines 4-5). Assuming policy gradient learners, the update function is

$$f(\tau^j, \theta^j, \eta^{-j}) = \sum_{t=0}^{T} \nabla_{\theta^j} \log \pi^j(a^j_t | o^j_t) G^j_t(\tau^j; \eta^{-j}), \tag{4}$$

where the return is $G^j_t(\tau^j, \eta^{-j}) = \sum_{l=t}^{T} \gamma^{l-t} r^j(s_l, \mathbf{a}_l, \eta^{-j})$.

After all agents have updated their policies to $\hat{\pi}^j$, each of which is parameterized by new $\hat{\theta}^j$, they generate new trajectories $\hat{\tau}^j$. These

Algorithm 1 Learning to Incentivize Others

```
1:  procedure TRAIN LIO AGENTS
2:      Initialize all agents' policy parameters $\theta^i$, incentive function parameters $\eta^i$, exploration rate $\epsilon$
3:      for each iteration do
4:          Generate an episode trajectory $\tau$ using $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$
5:          For reward-recipients $j$, update $\hat{\theta}^j$ using (3)
6:          Generate new episode trajectory $\hat{\tau}$ using new $\hat{\boldsymbol{\theta}}$
7:          For reward-givers $i$, update $\hat{\eta}^i$ by gradient ascent on (5)
8:          $\epsilon \leftarrow \epsilon - \epsilon_{\text{step}}$ if $\epsilon > \epsilon_{\text{end}}$
9:          $\theta^i \leftarrow \hat{\theta}^i$, $\eta^i \leftarrow \hat{\eta}^i$
10:     end for
11: end procedure
```

trajectories are used by the reward-givers i to update the individual incentive function parameters η^i to maximize their individual expected extrinsic return (Algorithm 1, lines 6-7). We define the objective function for the incentive function of each agent i as

$$J^i(\hat{\tau}^i, \tau^i, \hat{\boldsymbol{\theta}}, \eta^i) := \mathbb{E}_{\hat{\boldsymbol{\pi}}}\left[\sum_{t=0}^{T} \gamma^t \hat{r}^{i,\text{env}}_t\right] - \alpha L(\eta^i, \tau^i), \tag{5}$$

where the first term is the expected return for the reward-giver in the second trajectory $\hat{\tau}^i$, and the second term is a regularizer based on the rewards given in the first trajectory τ^i:[2]

$$L(\eta^i, \tau^i) := \sum_{(o^i_t, a^{-i}_t) \in \tau^i} \gamma^t \| r_{\eta^i}(o^i_t, a^{-i}_t) \|^2_2 . \tag{6}$$

Letting $J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}})$ denote the first term in (5), the gradient w.r.t. η^i is:

$$\nabla_{\eta^i} J^i(\hat{\tau}, \hat{\boldsymbol{\theta}}) = \sum_{j \neq i} (\nabla_{\eta^i} \hat{\theta}^j)^T \nabla_{\hat{\theta}^j} J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}}) . \tag{7}$$

The first factor of each summation follows directly from (3) and (4):

$$\nabla_{\eta^i} \hat{\theta}^j = \alpha \sum_{t=0}^{T} \nabla_{\theta^j} \log \pi^j(a^j_t | o^j_t) \left(\nabla_{\eta^i} G^j_t(\tau^j; \eta^{-j})\right)^T \tag{8}$$

Note that in contrast to Xu et al. [36], (3) does not contain recursive dependence of θ^j on η^i because θ^j is a function of incentives received during *previous* episodes, not on those received during the trajectory τ^i. The second factor in (7) is

$$\nabla_{\hat{\theta}^j} J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}}) = \mathbb{E}_{\hat{\boldsymbol{\pi}}}\left[\nabla_{\hat{\theta}^j} \log \hat{\pi}^j(\hat{a}^j | \hat{o}^j) Q^{i,\hat{\boldsymbol{\pi}}}(\hat{s}, \hat{\mathbf{a}})\right] \tag{9}$$

The derivation (omitted) is similar to that for policy gradient [32]. In practice, to avoid manually computing the matrix-vector product in (7), one can define the loss

$$\text{Loss}(\eta^i, \hat{\tau}^i) := -\sum_{j \neq i} \sum_{t=0}^{T} \log \pi_{\hat{\theta}^j}(\hat{a}^j_t | \hat{o}^j_t) \hat{G}^i_t \tag{10}$$

and directly minimize it via automatic differentiation [1]. Crucially, $\hat{\theta}^j$ must preserve the functional dependence of the policy update step (4) on η^i within the same computation graph. The derivation (omitted) is again similar to policy gradient, except that the gradient is w.r.t. η^i, and all the $\hat{\theta}^j$ for $j \neq i$ have dependence on η^i.

[2] (6) accounts for the fact that incentives should carry a cost to the reward-giver. Another option is to include the cost in $r^{i,\text{env}}$, but this is difficult to optimize via (7).

Our method can be seen as part of the paradigm of centralized training for decentralized execution [7, 22]. The agents and their objectives are fully decentralized, for they share no parameters and do not have knowledge of others' rewards; however, we do assume that each agent has access to parameter gradients of the others to learn the incentive function. In this sense, the centralization of our training may be described as the common knowledge that all agents are RL agents who learn from rewards. Note that by learning models of other agents we could implement our algorithm in a fully decentralized way, an extension we leave to future work.

4.1 Relation to opponent shaping via actions

LIO conducts a particular form of opponent shaping via the incentive function. This resembles LOLA [8], but there are key algorithmic differences. Firstly, LIO's incentive function can be viewed as a pseudo-action channel that exerts influence on other agents, but its parameters are trained separately from policy parameters, unlike LOLA where opponent shaping is done solely via the policy. Secondly, the LOLA gradient correction for agent i is derived from $\nabla_{\theta^i} J^i(\theta^i, \theta^j + \Delta\theta^j)$ under Taylor expansion, but LOLA disregards a term with $\nabla_{\theta^i}\nabla_{\theta^j} J^i(\theta^i, \theta^j)$ even though it is non-zero in general. In contrast, LIO is constructed from the principle of online cross-validation [30], rather than Taylor expansion, and hence this particular mixed derivative is absent—the analogue for LIO would be $\nabla_{\eta^i}\nabla_{\theta^j} J^i(\theta^i, \theta^j)$, which is zero because incentive parameters η^i affect all agents *except* agent i. Thirdly, LOLA optimizes its objective assuming one step of opponent learning, *before* the opponent actually does so. In contrast, LIO updates the incentive function *after* recipients carry out policy updates using received incentives. This gives LIO a more accurate measurement of the impact of given incentives, which reduces variance and increases performance, as we discuss in Section 6.1. On the other hand, LIO can be viewed as more restrictive than LOLA, as it assumes the possibility of adding a differentiable reward channel to the environment. Nevertheless, this modification can readily be made in many practical applications.

5 EXPERIMENTAL SETUP

Our experiments demonstrate that LIO agents are able to reach near-optimal individual performance by incentivizing other agents in cooperation problems with conflicting individual and group utilities. We define the environment in Section 5.1 and describe the implementation of our method and baselines in Section 5.2.

5.1 The N-Player Escape Room game

We work with an N-player episodic *Escape Room* game in which some agent(s) must incur extrinsic penalties in order for other agent(s) to receive positive extrinsic reward. Each episode has a maximum of 5 time steps. For a fair comparison of all methods that allow inter-agent incentivization, the game accounts for all given rewards as a cost to the reward-giver when measuring performance.

The first variant is the asymmetric 2-player Markov game shown in Figure 1 and described in Section 3. The global optimum combined reward is +9, and it is impossible for A2 to get positive extrinsic reward. Due to the asymmetry, A1 is the reward-giver and A2 is the reward recipient for methods that allow incentivization. Each agent observes both agents' positions, and can move between

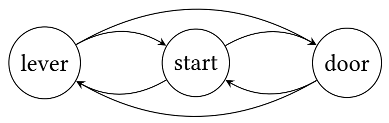


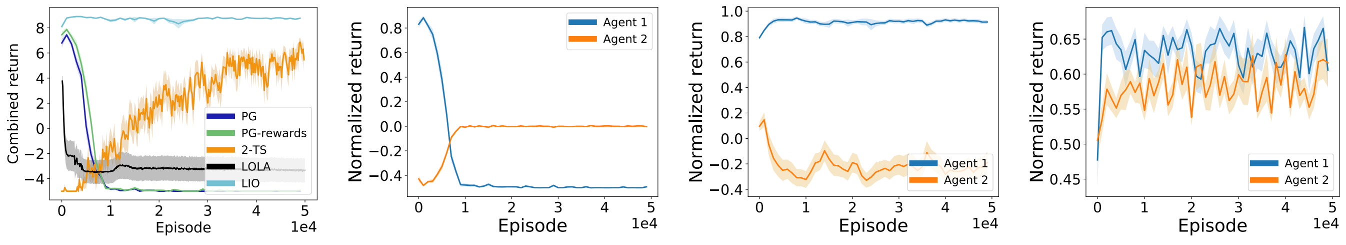
Figure 2: If fewer than M agents pull the lever, which incurs a cost of -1 if they are not already at the lever, then all agents receive -1 for changing their current position. Otherwise, the agent(s) who is not pulling the lever can get $+10$ by going to the door and end the episode.

the two states available to itself. We allow A1 to observe A2's current action before choosing its own action, which is necessary for methods that learn to reward A2's cooperative actions. We use a standard policy gradient for A2 unless otherwise specified.

The second variant is a symmetric N-player game shown in Figure 2. For a given parameter $M < N$, if fewer than M agents pull the lever, then all agents receive -1 extrinsic penalty for moving. Otherwise, the remaining $N - M$ agents can move to the door to get +10 extrinsic reward and terminate the episode. In principle, any agent can receive positive reward, as long as some other agent(s) cooperate. Hence, for methods that allow incentivization, every agent is both a reward giver and recipient. Each agent observes all agents' positions and can move among the three available states. At every time step, all agents commit to and disclose their chosen actions, compute the incentives based on their observations of state and others' actions (only for methods that allow incentivization), and receive the sum of extrinsic rewards and incentives (if any). We experiment with the case $N = 2, M = 1$ and $N = 3, M = 2$. For easier symmetry-breaking in the 3-agent case, we use uniformly random initialization for agents' starting location. Within each method (Section 5.2), all agents have the same implementation without sharing parameters.

5.2 Agent implementation and baselines

We use multi-layer perceptrons for all function approximation. The policy network has two ReLU layers with 64 and 32 units each, and a softmax output layer with size equal to the number of actions. The learned reward function in LIO has outputs bounded in $[0, R_{\max}]$: it has two ReLU layers with 64 and 16 units each; the input is the concatenation of the agent's observation and all other agents' chosen action; the output layer has sigmoid activation and size equal to the total number of agents; each output node i is interpreted as the real-valued reward given to agent with index i in the game (we zero-out the value it gives to itself); the output is scaled element-wise by multiplier $R_{\max} = 2$, since intuitively the incentive must overcome the -1 extrinsic penalty incurred for the cooperative action. We use on-policy learning with policy gradient for each agent, using stochastic gradient descent with learning rate 10^{-3} for policy parameters and the Adam optimizer [16] with initial learning rate 10^{-4} for the incentive function parameters in LIO, which is regularized by (6) with coefficient $\alpha = 10^{-3}$. To ensure the reward recipient performs sufficient exploration for the reward-giver to learn the effect of given rewards, we include an exploration lower bound ϵ such that $\tilde{\pi}(a|s) = (1 - \epsilon)\pi(a|s) + \epsilon/|\mathcal{A}|$, with ϵ decaying linearly from $\epsilon_{\text{start}} = 1.0$ to $\epsilon_{\text{end}} = 0.1$ by 1000 episodes. We include



(a) Sum of agent rewards (b) Two PG agents (c) LIO (A1) and PG agent (A2) (d) 1-episode LIO and PG agent

Figure 3: Results in asymmetric 2-player escape room game. (a) LIO (paired with PG) converges rapidly to global optimum, 2-TS (paired with tabular Q-learner) converges slower, while policy gradient baselines could not learn to cooperate. (b) Two PG agents cannot cooperate, as A2 converges to "do-nothing". (c) A LIO agent (A1) attains near-optimum reward by incentivizing a PG agent (A2). (d) 1-episode LIO has larger variance and lower performance. Normalization factors are 1/10 (A1) and 1/2 (A2).

an entropy regularization with coefficient 0.01, and use discount factor $\gamma = 0.99$.

Baselines. The first baseline is independent policy gradient for all agents, labeled **PG**, using the same architecture and hyperparameters as the policy part of LIO as described earlier. Second, we augment policy gradient with reward-giving actions, labeled **PG-rewards**, whereby the augmented action space is now $\mathcal{A} \times \{\text{no-op, give-reward}\}$. As reward-giving is a discrete action in this baseline, we try reward values in the set $\{2, 1.5, 1.1\}$. Giving reward incurs an extrinsic cost equal to the value given.[3] Next, we run **LOLA** with the same augmented action space as PG-rewards. Finally, we create a two-timescale method, labeled **2-TS**. A 2-TS agent has the same augmented action space as the PG-rewards baseline, except that it learns over a longer time horizon than the reward recipient. Each "epoch" for the 2-TS agent spans multiple regular episodes of the recipient, during which the 2-TS agent executes a fixed policy. The 2-TS agent only caries out a learning update using a final terminal reward, which is the average extrinsic rewards it gets during *test* episodes that are conducted at the end of the epoch. Performance on test episodes serve as a measure of whether correct reward-giving actions were taken to influence the recipient's learning during the epoch. To the best of our knowledge, 2-TS is a novel baseline but has key limitations: the use of two timescales only applies to the asymmetric 2-player game, and requires fast learning by the reward-recipient, chosen to be a tabular Q-learner, to avoid intractably long epochs.

6 RESULTS

We find that LIO agents are able to reach near-optimal *collective* performance in both the asymmetric and symmetric escape room game, in contrast to the policy gradient baselines that were unable to show any sign of cooperation. LOLA was able to solve the task sometimes, but this behavior was not robust across random seeds. For each experiment result, we report the mean and standard error across 20 independent runs with different random seeds. Section 6.1 analyzes the result in the asymmetric 2-player case, and Section 6.2 shows the result in the symmetric case for $N = 2$ and $N = 3$.

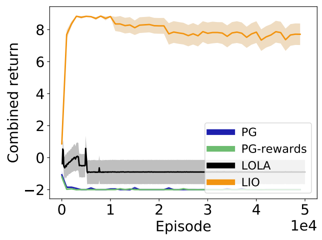
[3]Note that LIO uses direct regularization instead of accounting for cost of giving reward in the extrinsic reward. The comparison to baselines is fair as we enforce that given rewards is a cost during evaluation for all methods.

6.1 Asymmetric game

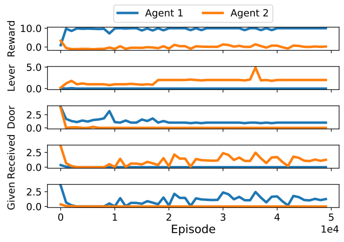
Figure 3 shows the sum of both agents' rewards for all methods on the asymmetric 2-player game, as well as agent-specific performance for policy gradient and LIO, across training episodes. A LIO reward-giver agent paired with a policy gradient recipient converges rapidly to a combined return near 9.0 (Figure 3a), which is the global maximum, while both PG and PG-rewards could not escape the global minimum for A1. LOLA paired with a PG recipient found the cooperative solution in two out of 20 runs; this suggests the difficulty of using a fixed incentive value to conduct opponent shaping via discrete actions. The 2-TS method is able to improve combined return but does so much more gradually than LIO, because an epoch consists of many base episodes and it depends on a highly delayed terminal reward. Figure 3b for two PG agents shows that A2 converges to the policy of not moving (reward of 0), which results in A1 incurring penalties at every time step. In contrast, Figure 3c verifies that A1 (LIO) receives the large extrinsic reward (scaled by 1/10) for exiting the door, while A2 (PG) has average normalized reward above -0.5 (scaled by 1/2), indicating that it is receiving incentives from A1. Average reward of A2 (PG) is below 0 because incentives given by A1 need not exceed 1 continually during training—once A2's policy is biased toward the cooperative action in early episodes, its decaying exploration rate means that it may not revert to staying put even when incentives do not overcome the penalty for moving. Figure 3d shows results on a one-episode version of LIO where the same episode is used for both policy update and incentive function updates, with importance sampling corrections. This version performs significantly lower for A1 and gives more incentives than is necessary to encourage A2 to move. It demonstrates the benefit of learning the reward function using a separate episode from that in which it is applied.

6.2 Symmetric game

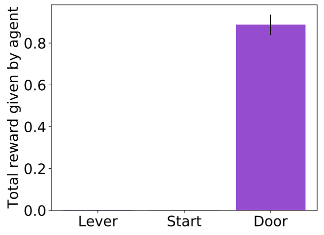
Figures 4 and 5 show that groups of LIO agents are able to discover a division of labor in both the 2-player and 3-player cases, whereby some agent(s) cooperate by pulling the lever to allow another agent to exit the door, such that the collective return approaches the optimal value (9 for the 2-player case, 8 for the 3-player case). We found that uniform randomization of agents' initial position was helpful in breaking the symmetry among three LIO agents. As expected,



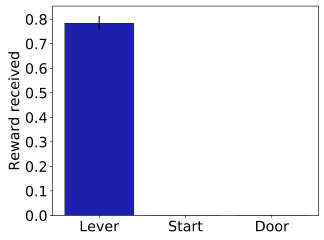
(a) Sum of agent rewards



(b) Actions and incentives (LIO)

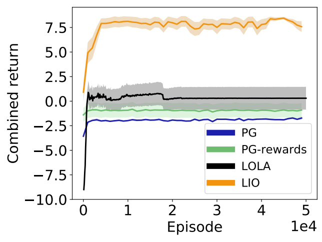


(c) Rewards given at each state

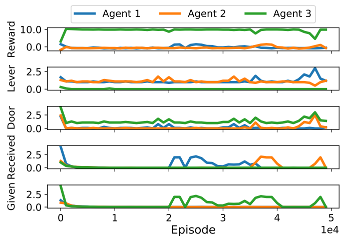


(d) Rewards received at each state

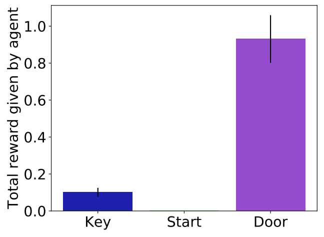
Figure 4: Results in symmetric 2-player escape room game. (a) Two LIO agents converge near the global optimum, while policy gradient agents (with and without reward-giving actions) do not. (b) Training dynamics of two LIO agents. (c-d) LIO agents cooperate by giving and receiving rewards at the correct states (mean and standard error of 100 test episodes).



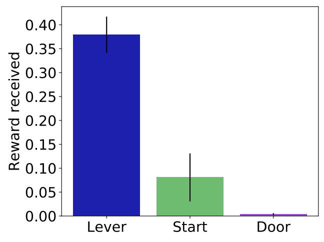
(a) Sum of agent rewards



(b) Actions and incentives (LIO)



(c) Rewards given at each state



(d) Rewards received at each state

Figure 5: Results in symmetric 3-player escape room game. Two agents must pull the lever for the door to open. (a) Three LIO agents converge near the global optimum (8); PG agents do not. (b) Agent 3 exits the door by giving incentives to Agents 1 and 2 (one run). (c-d) LIO agents give and receive rewards at mostly the correct states (mean and standard error of 100 test episodes).

both PG and PG-rewards were unable to find a cooperative solution. LOLA sometimes successfully influence the learning of another agent to solve the game, but exhibits high variance across independent runs. For the two-player case, Figure 4c verifies that only agents who go to the door give non-zero rewards, which shows the effect of the regularization term in (6). Moreover, Figure 4d shows that rewards are given correctly—only agents who pull the lever receive non-zero rewards. Figure 4b shows total reward, counts of "pull-lever" and "go to door" actions, total received rewards, and total given rewards across one particular training run. Here, A1 becomes the reward-giver and A2 the reward recipient. Somewhat surprisingly, it is not always necessary for Agent 1 to give rewards. The fact that LIO models the learning updates of recipients may allow it to find that reward-giving is unnecessary during some episodes when the recipient's policy is sufficiently biased toward cooperation. However, for the 3-player case, Figures 5c and 5d show that LIO makes mistakes in some runs, where agents who stay at the start state received rewards from others, while agents who pull the lever gave rewards to others. This suggests room for further improvement. Figure 5b shows an example run in which Agents 1 and 2 learn to cooperate to pull the lever and receive rewards from Agent 3, who exits the door.

Overall, the consistent failure of PG-rewards even in this seemingly simple game supports the intuition that the standard RL framework is not well suited to the task of learning to incentivize, as the effect of given rewards manifests only in future episodes. LOLA succeeds sometimes but with high variance, as it does not benefit from the stabilizing effects of online cross-validation and separation of the incentivization channel from regular actions. Results of 2-TS shows that learning over a longer horizon is a feasible albeit slow and indirect approach, while the significantly better performance of LIO shows the advantage of directly accounting for the impact of giving incentives on other learning agents.

7 CONCLUSION

We created Learning to Incentivize Others (LIO), an agent who learns to give rewards directly to other reinforcement learning agents. A LIO agent learns an incentive function by explicitly accounting for the impact of given rewards on its own extrinsic objective, through the learning updates of reward recipients. In multiple variants of a deceptively simple synthetic escape room game, where policy gradient agents cannot achieve cooperation even when augmented with reward-giving actions, we show that LIO agents paired with either policy gradient or other LIO agents are able discover a division of labor, whereby a LIO agent correctly incentivizes other agents to overcome extrinsic penalties to achieve optimum collective performance. In future work, we will scale up experiments to more complex domains such as intertemporal social dilemmas [13, 35] and take further steps toward a fully-decentralized implementation of LIO.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Tobias Baumann, Thore Graepel, and John Shawe-Taylor. 2018. Adaptive Mechanism Design: Learning to Promote Cooperation. *arXiv preprint arXiv:1806.04067* (2018).

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* (2019).

[4] Josse Delfgaauw and Robert Dur. 2008. Incentives and workers' motivation in the public sector. *The Economic Journal* 118, 525 (2008), 171–191.

[5] Margaret P Doxey. 1980. *Economic sanctions and international enforcement.* Springer.

[6] Tom Eccles, Edward Hughes, János Kramár, Steven Wheelwright, and Joel Z Leibo. 2019. Learning reciprocity in complex sequential social dilemmas. *arXiv preprint arXiv:1903.08082* (2019).

[7] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems.* 2137–2145.

[8] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 122–130.

[9] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence.*

[10] Yuk-fai Fong and Jay Surti. 2009. The optimal degree of cooperation in the repeated Prisoners' Dilemma with side payments. *Games and Economic Behavior* 67, 1 (2009), 277–291.

[11] Werner Güth. 1995. An evolutionary approach to explaining cooperative behavior by reciprocal incentives. *International Journal of Game Theory* 24, 4 (1995), 323–344.

[12] Bård Harstad. 2008. Do side payments help? Collective decisions and strategic delegation. *Journal of the European Economic Association* 6, 2-3 (2008), 468–477.

[13] Edward Hughes, Joel Z Leibo, Matthew Phillips, Karl Tuyls, Edgar Dueñez-Guzman, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin McKee, Raphael Koster, et al. 2018. Inequity aversion improves cooperation in intertemporal social dilemmas. In *Advances in neural information processing systems.* 3326–3336.

[14] Matthew O Jackson and Simon Wilkie. 2005. Endogenous games and mechanisms: Side payments among players. *The Review of Economic Studies* 72, 2 (2005), 543–566.

[15] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. 2019. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 364, 6443 (2019), 859–865.

[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations.*

[17] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 464–473.

[18] Adam Lerer and Alexander Peysakhovich. 2017. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068* (2017).

[19] Alistair Letcher, Jakob Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. 2019. Stable opponent shaping in differentiable games. In *International Conference on Learning Representations.*

[20] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994.* Elsevier, 157–163.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[22] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[23] Mancur Olson. 1965. *The Logic of Collective Action.* Harvard University Press.

[24] Anatol Rapoport. 1974. Prisoner's Dilemma—Recollections and observations. In *Game Theory as a Theory of a Conflict Resolution.* Springer, 17–34.

[25] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning.* 4295–4304.

[26] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354–359.

[27] Satinder Singh, Richard L Lewis, and Andrew G Barto. 2009. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society.* Cognitive Science Society, 2601–2606.

[28] Eric Sodomka, Elizabeth Hilliard, Michael Littman, and Amy Greenwald. 2013. Coco-Q: Learning in stochastic games with side payments. In *International Conference on Machine Learning.* 1471–1479.

[29] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 2085–2087.

[30] Richard S Sutton. 1992. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *AAAI.* 171–176.

[31] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[32] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems.* 1057–1063.

[33] Joseph Veroff and Joanne B Veroff. 2016. *Social incentives: A life-span developmental approach.* Elsevier.

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

[35] Jane X Wang, Edward Hughes, Chrisantha Fernando, Wojciech M Czarnecki, Edgar A Duéñez-Guzmán, and Joel Z Leibo. 2019. Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems.* International Foundation for Autonomous Agents and Multiagent Systems, 683–692.

[36] Zhongwen Xu, Hado P van Hasselt, and David Silver. 2018. Meta-gradient reinforcement learning. In *Advances in neural information processing systems.* 2396–2407.

[37] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. 2020. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In *International Conference on Learning Representations.*

[38] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. 2018. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems.* 4644–4654.