IQ-Flow: Mechanism Design for Inducing Cooperative Behavior to Self-Interested Agents in Sequential Social Dilemmas

Bengisu Guresti
Istanbul Technical University
ITU AI Center
Istanbul, Turkey
guresti15@itu.edu.tr

Abdullah Vanlioglu
Istanbul Technical University
ITU AI Center
Istanbul, Turkey
vanlioglu16@itu.edu.tr

Nazim Kemal Ure
Istanbul Technical University
ITU AI Center
Istanbul, Turkey
ure@itu.edu.tr

ABSTRACT

Achieving and maintaining cooperation between agents to accomplish a common objective is one of the central goals of Multi-Agent Reinforcement Learning (MARL). Nevertheless in many real-world scenarios, separately trained and specialized agents are deployed into a shared environment, or the environment requires multiple objectives to be achieved by different coexisting parties. These variations among specialties and objectives are likely to cause mixed motives that eventually result in a social dilemma where all the parties are at a loss. In order to resolve this issue, we propose the Incentive Q-Flow (IQ-Flow) algorithm, which modifies the system's reward setup with an incentive regulator agent such that the cooperative policy also corresponds to the self-interested policy for the agents. Unlike the existing methods that learn to incentivize self-interested agents, IQ-Flow does not make any assumptions about agents' policies or learning algorithms, which enables the generalization of the developed framework to a wider array of applications. IQ-Flow performs an offline evaluation of the optimality of the learned policies using the data provided by other agents to determine cooperative and self-interested policies. Next, IQ-Flow uses meta-gradient learning to estimate how policy evaluation changes according to given incentives and modifies the incentive such that the greedy policy for cooperative objective and self-interested objective yield the same actions. We present the operational characteristics of IQ-Flow in Iterated Matrix Games. We demonstrate that IQ-Flow outperforms the state-of-the-art incentive design algorithm in Escape Room and 2-Player Cleanup environments. We further demonstrate that the pretrained IQ-Flow mechanism significantly outperforms the performance of the shared reward setup in the 2-Player Cleanup environment.

KEYWORDS

Sequential Social Dilemmas; Adaptive Mechanism Design; Multi-agent Reinforcement Learning; Meta-gradient Learning

ACM Reference Format:
Bengisu Guresti, Abdullah Vanlioglu, and Nazim Kemal Ure. 2023. IQ-Flow: Mechanism Design for Inducing Cooperative Behavior to Self-Interested Agents in Sequential Social Dilemmas. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 17 pages.

1 INTRODUCTION

Social Dilemmas [16] emerge when self-interested parties have conflicting objectives. Greed or fear of being exploited drives the agents towards defecting, which results in worse outcomes for the whole group in comparison to outcomes that would come out of cooperation [11, 12]. This problem has many applications in computer science, economics and social sciences; hence, it is well-studied under Game Theory using Matrix Game Social Dilemmas (MGSD) and their iterated extension Repeated Matrix Games [12]. Although MGSDs are useful for modelling social dilemmas in real world scenarios, they omit significant characteristics of real world social dilemmas, which are addressed by Sequential Social Dilemmas (SSD) due to their temporally extended structure [12]. Since cooperation and defection are defined for policies in SSD rather than elementary actions [12], how to induce cooperative behavior to agents in an SSD while the agents are concurrently learning is an open research question.

Centralized training methods [5, 17, 18] are popular approaches in Multi Agent Reinforcement Learning (MARL) when cooperation is necessary. However, the centralized approaches involve a shared objective to optimize agents' policies and assume full control over agents' internal parameters and learning. Nevertheless, as the use of artificial intelligence becomes common and agents that are separately trained for different objectives are deployed in a shared environment [22], it will not be realistic to either expect no conflicting objectives or assume full control over agents' internal parameters and learning. Since it is not possible to guarantee the type, tasks and number of the deployed agents in an unrestricted environment, agents need to be able to continually learn and adapt to the environment while cooperating with each other. Therefore, in this work, we focus on independently learning self-interested agents in an SSD where the agents receive adaptive incentives in order to promote cooperation.

There are different configurations for how agents can be incentivized during learning. Agents can give each other adaptive incentives to shape each other's behavior for their own benefit [4, 13], or there can be a central institution to provide the incentives to shape the agents' behavior for the welfare of the whole community [1, 22]. In this work, we adopt the latter approach and provide a mechanism that provides incentives to all of the agents in the system in order to prevent any undesirable outcome, such as tragedy of the commons due to defecting. While it might seem like a trivial problem to learn incentives for the mechanism, since providing the average reward to all agents would certainly remove

the existing dilemma, it is shown to yield suboptimal results [15, 22]. Therefore, it is important to design a mechanism that promotes cooperation without incurring performance losses. Furthermore, promoting cooperation to artificial self-interested agents is not the only direction for mechanism design research. Mechanism design can also be used to model human incentives and solve human dilemmas such as determining tax rate for a higher social welfare [22].

In this work, we propose Incentive Q-Flow (IQ-Flow) algorithm to design incentive mechanisms for increasing social welfare and promoting cooperation. IQ-Flow aims to make the cooperative policy correspond to the self-interested policy of the agents by changing system's reward setup. IQ-Flow collects the experience obtained from agents into a replay buffer and trains critic networks to learn state-action values (Q-Values) for agents' self-interests and the group's collective interest. IQ-Flow parameterizes incentive function using meta-parameters and performs meta-gradient learning as in [3, 20–22] to update the incentive network. In order to learn incentive meta-parameters, IQ-Flow trains the critic using Offline Implicit Q-Learning [10] with the train set for multiple steps and obtains updated parameters, performs policy evaluation with the validation set, and updates the meta-parameters in the direction that makes the actions of the collaborative policy the greedy choice for self-interested agents' Q-Values.

Our algorithm is distinguished from the existing incentive design methods by grounding itself on reward system shaping rather than opponent shaping. Using Offline Reinforcement Learning (Offline RL) with Implicit Q-Learning makes it possible to get a proximate estimate of Q-Values for as greedy as possible policies with self-interested and collective interest objectives only using experience collected by external agents. This approach enables IQ-Flow to modify the reward system with the incentive function by getting a close enough estimate of how changing the incentive affects the expected future return brought about by the reward system. Using an offline method such as Implicit Q-Learning instead of standard Deep Q-Learning is justified by the fact that incentivizer critic has an indirect effect on recipient agents' policies and can only affect collecting experience indirectly. Furthermore, using Implicit Q-Learning also makes extending IQ-Flow to fully offline training simpler for future work. As opposed to opponent shaping based algorithms, IQ-Flow does not possess or make assumptions on any of the agents' internal parameters, learning algorithms, or hyperparameters which makes it independent from the agents in the environment except for the collected experience. Another key difference of IQ-Flow from existing work is that it does not require cost regularization to train an incentive mechanism in SSDs. Nevertheless, we find that including cost regularization improves IQ-Flow's performance as well. Finally, it should be noted that IQ-Flow does not learn a multi-agent policy or perform value factorization to determine the actions of a cooperative policy. This is due to the fact that the algorithm only needs to know the cooperative or selfish action of a specific agent when the actions of other agents are provided. Our contributions can be summarized as below:

- Proposing reward system shaping instead of opponent shaping for incentive design; thus, instead of pushing agents towards a Nash-Equilibrium with cooperative outcomes, modifying the reward system such that rational agents are stuck in Nash-Equilibrium with cooperative outcomes
- Extending incentive design framework to learn mechanisms off-policy using offline RL and replay buffer; thus, applying offline RL and replay buffer with meta-gradient learning for MARL for the first time to the best of our knowledge
- Removing the requirement of accessing or making assumptions on agents' internal learning state for incentive design
- Removing the requirement of cost regularization for incentive design in SSDs

We illustrate how IQ-Flow operates for Iterated Matrix Games in Iterated Prisoner's Dilemma, Iterated Chicken Game and Iterated Stag Hunt. We further evaluate the performance of our algorithm in the common benchmarks Escape Room [21] and SSD-Cleanup [7, 8, 19] with 2 Players. We demonstrate that it outperforms the state-of-the-art incentive design algorithm ID and perform ablation studies for IQ-Flow. We further demonstrate that the pretrained mechanism, learned by IQ-Flow, leads to significantly better learning performance than using a shared reward setup. We provide the code for our implementation and experiments at https://github.com/data-and-decision-lab/IQ-Flow.git.

2 RELATED WORK

Centralized training methods in MARL such as COMA [5], VDN [18], and QMIX [17] are successful at optimizing all agents' policies or factorize value functions to achieve a common objective. However, SSD problems can not be approached as fully cooperative problems due to the nature of the problem emerging from coexisting mixed motives and diverse objectives. Hence, decentralized training methods have been developed along with opponent shaping and incentivization practices [4, 8, 13, 21] in order to model and resolve social dilemma problems.

Opponent shaping was proposed by [4] to provide independent learners with the ability to shape each other's behavior in the face of a mixed motive. LOLA [4] agents can access the policy parameters of their opponents and actively learn in the direction that improves their own returns by considering how their opponent's future policy is expected to change. The disadvantage of the LOLA is that it can adopt arrogant behavior, as claimed by [13] and fixed with a new algorithm named SOS. SOS [13] algorithm is similar to LOLA in adopting opponent shaping, but offers a more robust algorithm by removing the arrogant behavior and inheriting the guarantees of LookAhead [23] on avoiding strict saddles in all differentiable games.

Incentivization practices can be exemplified by Social Influence [8], AMD [1], LIO [21] and ID [22]. Social Influence [8] rewards the agent action that has the most impact on others' behavior as an intrinsic reward. In LIO [21] an agent learns to use incentive reward that affects the learning update of opponents' policies and changes the objectives of the recipient agents in the direction that improves incentivizer agents' objectives by using meta-gradient learning. AMD [1] uses a central planner agent that learns how to set an incentive reward according to agents expected policy

update in the next step. [24] presents a two-dimensional grid world dynamic economic environment Gather-Trade-Build game, where agents collect resources, earn coins by building houses with these materials, and trade resources; moreover, there is a central tax-planner agent who learns to improve the trade-off between income equality and productivity by setting taxes that correspond to a payoff from the agent's income. [22] use same environment and propose meta-gradient approach to train Incentive Designer (ID), the central planning analogue of LIO, as an incentive mechanism. Mechanism design can also be used to model human incentives and solve human dilemmas such as determining tax rate for a higher social welfare [22] using the simulation environment AI Economist, proposed in [24]. This is a good illustration of how it can be used as a recommendation system for solving social problems in the future. Because we adopt the approach of directly incentivizing the agents using an extra additive reward and the economic simulation environment from Zheng et al. [24] requires an indirect approach such as determining the tax policy, we do not address the taxation problem in AI Economist in this work and leave it to future work.

Incentivization practices that use meta-gradient learning to shape opponent behavior, such as LIO and ID are the approaches closest to our learning algorithm. However, while in LIO and ID, the meta-gradient based incentive mechanism performs on-policy learning [21, 22], IQ-Flow's incentive mechanism learns in an off-policy manner with a replay buffer. A prior work that uses off-policy learning with a replay buffer for the first time in meta-gradient learning is MetaL [3]. Unlike the opponent shaping based methods, IQ-Flow does not need access or modelling of other agents' parameters. Instead of focusing on how the behavior of agents change, IQ-Flow focuses on rendering cooperative actions in the Nash-Equilibrium for the possible states. Since non-cooperation would incur a loss for all agents, IQ-Flow tasks the agents' to optimize their returns and choose cooperation; IQ-Flow does not keep track of how agents' behavior policies change. This is due to training the incentivizer critic by offline Implicit Q-Learning, which is the key difference from LIO and ID that use online incentivizer training.

3 BACKGROUND

In this work, we assume a Partially Observable MDP (POMDP) where N agents learn independently. $\mathcal{S}$ denotes the global state of the environment, $a^i \in a$ denote action of i'th agent in joint action a, and i^- denotes all agent indices except i with index set denoted as $\mathcal{I} = \{0, 1, ..., N-1\}$. Observation space of agent i is $O_i = \{o_i | s \in \mathcal{S}, o_i = O(s, i)\}$ with the observation function $O : \mathcal{S} \times \mathcal{I} \rightarrow \mathbb{R}^d$ that maps the observations to the d-dimensional space. State, observation, action and reward at time step k are denoted as s_k, o_k, a_k, r_k respectively along with time horizon T and discount factor γ. We have the transition function of the environment $\mathcal{T} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathcal{P}(S)$ with $\mathcal{P}$ denoting the probability distribution over S and batch length l_B. Joint reward provided by the environment is $R_{env} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathbb{R}^N$ where each agent receives a specific reward $R^i_{env} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathbb{R}$. The incentive reward that can be given to an agent is constrained according to the environment as $\mathcal{U} \subset \mathbb{R}$. Thus, the joint incentives provided by the mechanism and parametrized by η is $R_{inc,\eta} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathcal{U}^N \subset \mathbb{R}^N$ where each agent receives a specific incentive $R^i_{inc,\eta} : \mathcal{S} \times \mathcal{A}^N \rightarrow \mathcal{U} \subset \mathbb{R}$. We define the total reward an agent receives which directs that agent's behavior policy as $R^i_{ind} = R^i_{env} + R^i_{inc,\eta}$. We further define the sum of the rewards that environment provides to all agents as $R^i_{coop} = \sum_{id=0}^{N-1} R^{id}_{env}$. It should be noted that R^i_{coop} is defined for all agents with the same value.

We define three different policies that are necessary for our problem case and solution method.

- $\boldsymbol{\pi}^i_b \in \boldsymbol{\pi}_b$: i'th agent's behavior policy which is optimized to maximise
  $V^i_{\boldsymbol{\pi}_b,ind}(s) := \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{ind,t} | s_k = s\right]$
- $\boldsymbol{\pi}^i_{coop} \in \boldsymbol{\pi}_{coop}$: i'th agent's cooperative policy which is optimized to maximise
  $V^i_{\boldsymbol{\pi}_{coop}}(s) := \mathbb{E}_{\boldsymbol{\pi}_{coop}}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{coop,t} | s_k = s\right]$
- $\boldsymbol{\pi}^i_{env} \in \boldsymbol{\pi}_{env}$: i'th agent's environment policy which is optimized to maximise
  $V^i_{\boldsymbol{\pi}_{env},env}(s) := \mathbb{E}_{\boldsymbol{\pi}_{env}}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{env,t} | s_k = s\right]$

We further denote the different objectives that are necessary for our problem case and solution method as follows:

- Action-values of i'th agent under $\boldsymbol{\pi}_b$ accounting for the individual total reward R^i_{ind}
  $Q^i_{\boldsymbol{\pi}_b,ind}(s, a) = \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{ind,t} | s_k = s, a_k = a\right]$
- Action-values of i'th agent under $\boldsymbol{\pi}_{coop}$ accounting for the cooperative reward R^i_{coop}
  $Q^i_{\boldsymbol{\pi}_{coop}}(s, a) = \mathbb{E}_{\boldsymbol{\pi}_{coop}}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{coop,t} | s_k = s, a_k = a\right]$
- Action-values of i'th agent under $\boldsymbol{\pi}_{env}$ accounting for the individual environment reward R^i_{env}
  $Q^i_{\boldsymbol{\pi}_{env},env}(s, a) = \mathbb{E}_{\boldsymbol{\pi}_{env}}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{env,t} | s_k = s, a_k = a\right]$
- Values of i'th agent under $\boldsymbol{\pi}_b$ accounting for the individual environment reward R^i_{env}
  $V^i_{\boldsymbol{\pi}_b,env}(s) = \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{env,t} | s_k = s\right]$
- Action-values of i'th agent under $\boldsymbol{\pi}_b$ accounting for the individual environment reward R^i_{env}
  $Q^i_{\boldsymbol{\pi}_b,env}(s, a) = \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{env,t} | s_k = s, a_k = a\right]$
- Values of i'th agent under $\boldsymbol{\pi}_b$ accounting for the individual incentive reward R^i_{inc}
  $V^i_{\boldsymbol{\pi}_b,inc}(s) = \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{inc,t} | s_k = s\right]$
- Action-values of i'th agent under $\boldsymbol{\pi}_b$ accounting for the individual incentive reward R^i_{inc}
  $Q^i_{\boldsymbol{\pi}_b,inc}(s, a) = \mathbb{E}_{\boldsymbol{\pi}_b}\left[\sum_{t=k}^{T-1} \gamma^{t-k} R^i_{inc,t} | s_k = s, a_k = a\right]$

Table 1: Matrix Game payoff table

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

Social Dilemma conditions. According to preliminary work in social dilemmas [12, 14], a Matrix Game such as Table 1 is a Social Dilemma if it satisfies the following conditions:

(1) $R > P$

(2) $R > S$
(3) $2R > T + S$
(4) $T > R$ or $P > S$

In this canonical Matrix Game in Table 1 actions C and D represent cooperate and defect actions as convention dictates [14]. We adopt the definitions proposed by [14] and we denote R, P, T and S respectively as reward from mutual cooperation, punishment from mutual defection, temptation reward from defecting while the other player cooperates and sucker reward from cooperating while the other player defects.

Offline Implicit Q-Learning. Offline Implicit Q-Learning is performed to learn critics as proposed by [10] for dataset $\mathcal{D}$, value parameters ψ, critic parameters θ, target critic parameters $\bar{\theta}$, state s, action a, next state $s^{'}$, discount γ, expectile $\tau_{exp} \in (0, 1)$ with the following loss equations:

$$
\begin{aligned}
&L_2^{\tau_{exp}}(u) = \left|\tau_{exp} - \mathbb{1}(u < 0)\right| u^2 \\
&L_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^{\tau_{exp}}(Q_{\bar{\theta}}(s,a) - V_\psi(s))\right] \\
&L_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[(r(s,a) + \gamma V_\psi(s^{'}) - Q_\theta(s,a))^2\right] \text{[10]}
\end{aligned}
\tag{1}
$$

We extend offline Implicit Q-Learning to our multi-agent case in order to approximate $Q^i_{\boldsymbol{\pi}_{b,ind}}$, $Q^i_{\boldsymbol{\pi}_{coop}}$, and $Q^i_{\boldsymbol{\pi}_{env,env}}$. We give the corresponding losses in Appendix A.1. We denote the training batch with $\mathcal{B}_T$ and validation batch $\mathcal{B}_V$.

4 INCENTIVE Q-FLOW

IQ-Flow bases itself on reversing the fourth social dilemma condition and make $T < R$ and $P < S$ in Table 1. When $R > T$ and $S > P$, choosing C over D becomes the greedy policy automatically without regard to the opponents' policy. Thus, IQ-Flow aims to make the action of the cooperative policy the greedy choice for the incentivized behavior policy using meta-gradients as we defined in background in section 3.

The necessity of using meta-gradients for estimating how Q-Values change according to η comes from the fact that it is not possible to directly estimate the long term value change as a result of a change of incentives. Let the optimal actions of the cooperative policy and incentivized behavior policy be defined respectively as:

$$
\begin{aligned}
a^i_{coop} &= \underset{a^i}{\operatorname{argmax}}\, Q^i_{\boldsymbol{\pi}_{coop}}(s, a^{i^-}, .) \\
a^i_b &= \underset{a^i}{\operatorname{argmax}}\, Q^i_{\boldsymbol{\pi}_{b,ind}}(s, a^{i^-}, .)
\end{aligned}
\tag{2}
$$

Let the optimal actions for the self-interested policy of agents under standard environment conditions with no extra incentives be defined as:

$$
a^i_{env} = \underset{a^i}{\operatorname{argmax}}\, Q^i_{\boldsymbol{\pi}_{env,env}}(s, a^{i^-}, .) \tag{3}
$$

In order to determine a^i_{coop}, a^i_b, and a^i_{env}, IQ-Flow needs to estimate $Q^i_{\boldsymbol{\pi}_{coop}}$, $Q^i_{\boldsymbol{\pi}_{b,ind}}$, and $Q^i_{\boldsymbol{\pi}_{env,env}}$. IQ-Flow approximates $Q^i_{\boldsymbol{\pi}_{coop}}$, $Q^i_{\boldsymbol{\pi}_{b,ind}}$, and $Q^i_{\boldsymbol{\pi}_{env,env}}$ respectively by $Q^i_{\boldsymbol{\pi}_{coop}}(\theta_{coop})$, $Q^i_{\boldsymbol{\pi}_{b,ind}}(\theta_{ind})$, and $Q^i_{\boldsymbol{\pi}_{env,env}}(\theta_{env})$. An important point is that since incentive function is dynamic, $Q^i_{\boldsymbol{\pi}_{b,ind}}(\theta_{ind})$ and $V^i_{\boldsymbol{\pi}_{b,ind}}(\psi_{ind})$ need to be updated with the r^i_{inc} inferred from the last η. IQ-Flow updates the critic parameters ψ_{ind} and θ_{ind}, respectively for $V^i_{\boldsymbol{\pi}_{b,ind}}(s, \psi_{ind})$ and $Q^i_{\boldsymbol{\pi}_{b,ind}}(s, a, \theta_{ind})$, with Implicit Q-Learning extended to MARL with the equations in Appendix A.1.

In order to update η, we first update our predefined critics with learning rate β_{ind} for K steps. This update can be given as following for the Stochastic Gradient Descent (SGD) optimizer:

$$
\begin{aligned}
&\hat{\theta}_{ind} \leftarrow \theta_{ind} + \beta_{ind}\nabla_{\theta_{ind}} \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1} \\
&\left(r^i_{env}(s_k, a_k) + r^i_{inc}(s_k, a_k, \eta) + \gamma V^i_{\psi_{ind}}(s^{'}_k) - Q^i_{\theta_{ind}}(s_k, a_k)\right)^2
\end{aligned}
\tag{4}
$$

Since we want to update η in the direction that flows Q-Values from actions of defective policies to actions of cooperative policies, we regard the a_{coop} as target labels in a classification problem and use a modified version of cross-entropy loss. The necessity of the modification in the cross-entropy loss is because we only want the gradient flow as long as there is a dilemma in the system so that there is no unnecessary and excessive incentivization. We identify an action that causes a dilemma as $a^i_b \neq a^i_{coop}$. Therefore we further mask our meta-loss for the case when there is no estimated dilemma. In order to get a probabilistic view of Q-Values and use them in the cross-entropy loss, we pass them through a softmax layer.

Finally our meta-loss can be defined as follows:

$$
\begin{aligned}
&L^m_\eta(\hat{\theta}_{ind}) := -\frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1}\sum_{\tilde{a}=0}^{|A|-1} \mathbb{1}\left(\tilde{a} = a^i_{coop,k}\right) \\
&\times\left(1 - \mathbb{1}(a^i_{b,k} = a^i_{coop,k})\right)\log\left(\sigma\left(Q^i_{\boldsymbol{\pi}_{b,ind}}\left(s_k, a^i, a^{i^-}_k, \hat{\theta}_{ind}\right)\right)\right)\Big|_{a^i=\tilde{a}} \\
&\sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}
\end{aligned}
\tag{5}
$$

Since we do not want to give an unnecessary incentive if there is no dilemma in the original case without extra incentives, we use another mask which determines if $a^i_{env} = a^i_{coop}$. Therefore we add a cost regularization term to the meta loss with cost coefficient c_1.

$$
\begin{aligned}
&L^{cost_1}_\eta(\hat{\theta}_{ind}) := \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1}\sum_{act=0}^{|A|-1} \mathbb{1}(a^i_{coop,k} = a^i_{env,k}) \\
&\times\left|Q^i_{\boldsymbol{\pi}_{b,ind}}(s_k, a^i, a^{i^-}_k \hat{\theta}_{ind}))\right|\Big|_{a^i=act}
\end{aligned}
\tag{6}
$$

If the incentives become too high prematurely, they can have a destructive effect, especially if they are the wrong incentives. Therefore we add another cost regularization term to the meta loss with cost coefficient c_2. Although our experiments show that these cost regularization terms are not required to get a successful performance, especially in simple problems, we find that including them leads to higher performance.

$$L_{\eta}^{cost_2}(\hat{\theta}_{ind}) := \frac{1}{l_B N} \sum_{k=0}^{l_B-1} \sum_{i=0}^{N-1} \sum_{act=0}^{|A|-1} \left(1 - \mathbb{1}(a^i_{coop,k} = a^i_{env,k})\right) \times \left| Q^i_{\boldsymbol{\pi}_b,ind}(s_k, a^i, a^{i-}_k \hat{\theta}_{ind}))) \right| \Big|_{a^i=act} \tag{7}$$

Our final incentive loss for η is given below as $L_{\eta}^{R_{inc}}(\hat{\theta}_{ind})$:

$$L_{\eta}^{R_{inc}}(\hat{\theta}_{ind}) = L_{\eta}^{m}(\hat{\theta}_{ind}) + c_1 L_{\eta}^{cost_1}(\hat{\theta}_{ind}) + c_2 L_{\eta}^{cost_2}(\hat{\theta}_{ind}) \tag{8}$$

If we use α as learning rate for η, set number of critic update steps K as 1, and assume SGD for optimizer, the update becomes:

$$\hat{\eta} \leftarrow \eta + \alpha \nabla_{\eta} L_{\eta}^{R_{inc}}(\hat{\theta}_{ind})$$

$$\nabla_{\eta} L_{\eta}^{R_{inc}}(\hat{\theta}_{ind}) = \frac{\partial L_{\eta}^{m}(\hat{\theta}_{ind}) + c_1 L_{\eta}^{cost_1}(\hat{\theta}_{ind}) + c_2 L_{\eta}^{cost_2}(\hat{\theta}_{ind})}{\partial \hat{\theta}_{ind}} \frac{\partial \hat{\theta}_{ind}}{\partial \eta} \tag{9}$$

The diagram for how η meta-parameter is updated is given below in Figure 1:

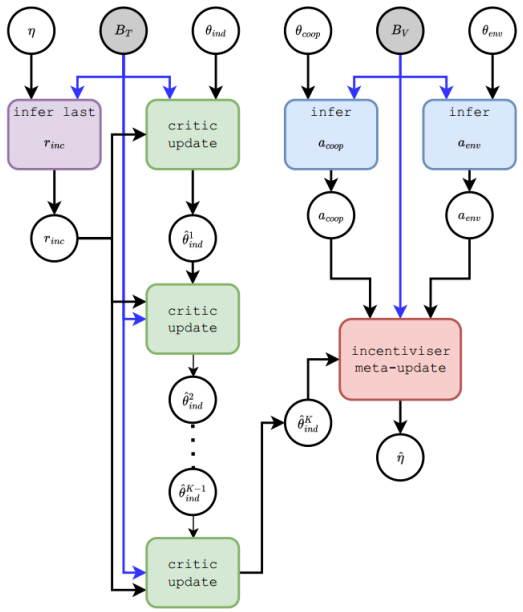

Figure 1: Meta-update diagram for incentive parameter η

The pseudocode of the algorithm is given below in Algorithm 1.

Algorithm 1 Incentive Q-Flow

procedure TRAIN IQ-FLOW MECHANISM(ϕ^0, ϕ^1,...,ϕ^{N-1}, args) ▷
Input: policy of all agents, hyperparameters
  Initialize η, θ_{coop}, θ_{env}, θ_{ind}, ψ_{coop}, ψ_{env}, ψ_{ind}
  $num\_episode \leftarrow 0$
  for number of episodes to train **do**
    Run agents with policies ϕ^0, ϕ^1,...,ϕ^{N-1} for an episode with incentives given by η
    $num\_episode \leftarrow num\_episode + 1$
    Add the transitions from episode to replay buffer of IQ-Flow
    Update agent policies ϕ^0, ϕ^1,...,ϕ^{N-1} according to their private learning rules
    Update θ_{coop}, θ_{env}, θ_{ind}, ψ_{coop}, ψ_{env}, ψ_{inc} using equations in 10
    sample $\mathcal{B}_T$ and $\mathcal{B}_V$ for meta-update
    simulate mechanism critic update for K times using $\mathcal{B}_T$, θ_{ind}
    Update η using $\mathcal{B}_V$ (with equations 5 or 9)
  end for
end procedure

5 EXPERIMENTS

5.1 Iterated Matrix Games

We demonstrate how IQ-Flow operates on the iterated extension of the three canonical Matrix Games, which are Prisoner's Dilemma, Chicken Game, and Stag Hunt. The payoff matrices for these games are given in Table 2, Table 3, and Table 4. We extend the implementation used by LOLA [4] and use the policy gradient agents for the independent learners as used by LIO [21] and ID [22]. The incentive reward is set as $R^i_{inc} \in (0, 2)$ to provide only sufficient incentivization and number of iterations is set as 20 for all experiments. Since the experimentation purpose here is for illustration rather than comparison, hyperparameter tuning was not performed to optimize learning performance and cost regularization was not added to the meta-objective. We demonstrate how IQ-Flow changes the payoff matrix of the games in Figure 2 and Appendix C. The first column in Figure 2 represents the original payoffs. The other column represents the modified total payoffs by IQ-Flow where each row represents the mechanism state trained for 30, 210, 390, 570, 750 episodes respectively. The first rows in the figures in Appendix C represent the original payoffs, while the other rows represent the state (initial state, previous action taken CC, previous action taken CD, previous action taken DC, and previous action taken DD). The columns represent the total payoff output of the mechanism state trained for 30, 210, 390, 570, 750 episodes respectively.

We depict how IQ-Flow changes the estimated Q-Values of the games in Figure 3 and Appendix D. The first columns in Figure 3 and figures in Appendix D represent the Q-Values without the mechanism incentives. The other columns represent the Q-Values with the mechanism incentives where each row represents the mechanism state trained for 30, 210, 390, 570, 750 episodes respectively. These outputs are given for the initial state.

Table 2: Prisoner's Dilemma

PD	C_2	D_2
C_1	(3, 3)	(0, 4)
D_1	(4, 0)	(1, 1)

Table 3: Chicken Game

Chicken	C_2	D_2
C_1	(3, 3)	(1, 4)
D_1	(4, 1)	(0, 0)

Table 4: Stag Hunt

Stag Hunt	C_2	D_2
C_1	(4, 4)	(0, 3)
D_1	(3, 0)	(1, 1)

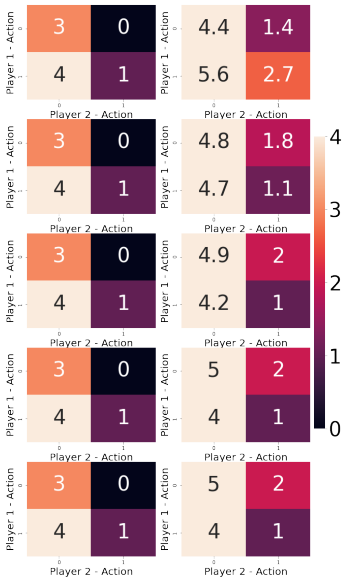

Figure 2: IPD player 1 payoff matrices

In addition to the detailed payoff and Q-Value charts, we provide a plot for Iterated Matrix Games to show how the inequalities turn from $T > R$ and/or $P > S$ to $T < R$ and $P < S$ in Figure 4 and Appendix E as training progresses. We highlight that the R, T, P, and S denotes the corresponding estimated Q-Values for all states and not the single step payoffs.

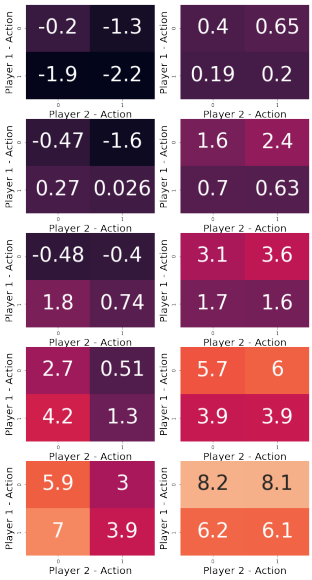

Figure 3: IPD player 1 estimated Q-Value matrices (left: without incentives, right: with incentives)

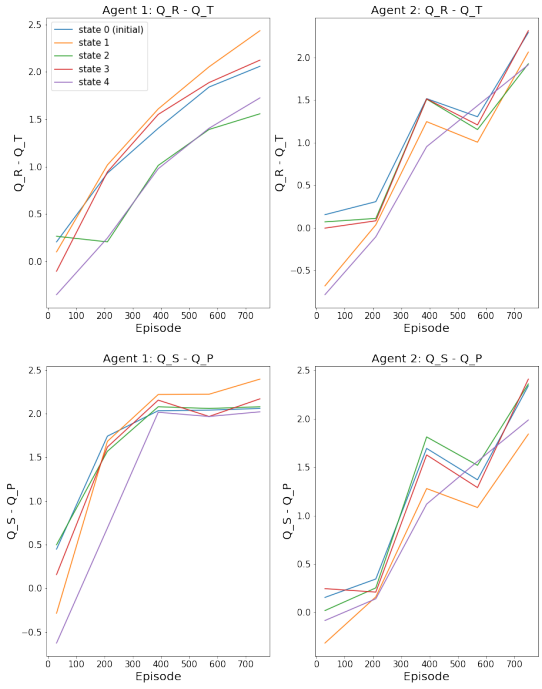

Figure 4: IPD $R - T$ and $S - P$ plot for Q-Values

Consequently, our results demonstrate clearly that IQ-Flow is capable of removing the social dilemma for Iterated Prisoner's Dilemma, Chicken Game and Stag Hunt; since we obtain $T < R$ and $P < S$ in the end for all of the cases in both single step payoffs and estimated future returns.

5.2 Escape Room

Escape room is a small, N-player Markov game proposed by [21]. The game contains 3 different state: initial, lever and door state where agents spawn in the initial state and aim to reach the door which is the terminal state [21]. But M number of agents must cooperate by pulling the lever at the same time to get others out of the door so that the agent who goes out of the door gets +10 reward individually while the cost of the pulling lever is -1 [21]. Therefore, in order to increase total return, some of the agents should give up their own interest and act cooperatively. We extend the implementation used by LIO [21], benefit from [9], and use the policy gradient agents for the independent learners as used by LIO [21] and ID [22]. We use the same experiment setup used by ID [22] and evaluate IQ-Flow's performance along with an ablation study given in Appendix B.

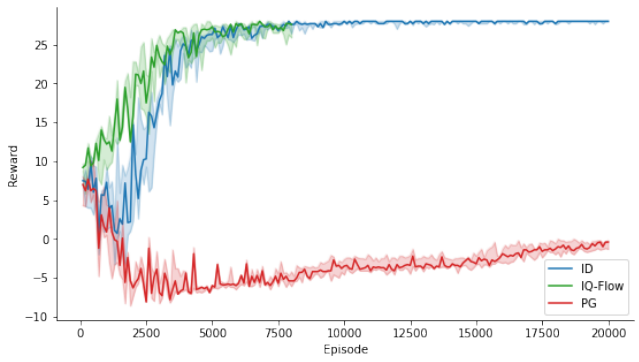

Figure 5: ER(5,2)

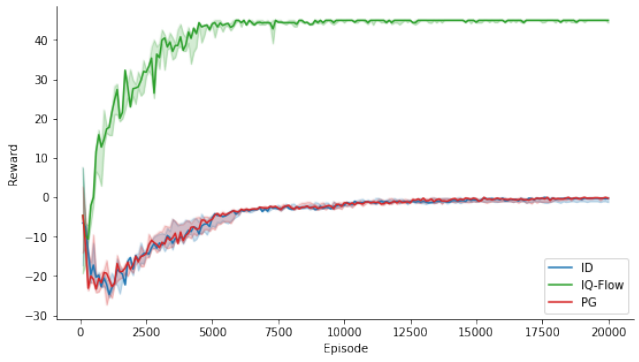

Figure 6: ER(10,5)

We give the results of Escape Room (5, 2) experiment, as in [22], in Figure 5. The basic case of no incentivization, denoted as PG, performs poorly as expected. ID reaches the optimal total return of the environment, which is 28. IQ-Flow performs the best by reaching 28 faster and with better initial training performance. The results of the experiment Escape Room (10, 5) is given in Figure 6. The basic case of no incentivization, denoted as PG, performs poorly again as expected. Although Yang et al. show [22] that ID reaches the optimal return of 45, we could not replicate those results with our implementation and obtained the performance of ID similar to PG. IQ-Flow reaches the optimal return of 45 faster than ID in both our implementation and the resuls given in [22]. Since the results of ablation study, given in Appendix B, does not provide distinctive results, we focus on the ablation of experiments in the 2-Player Cleanup environment.

5.3 SSD Environment - 2-Player Cleanup

Cleanup [7] is a grid-world social dilemma environment where the objective is to collect apples from field that give +1 reward. Since the respawn time of the apples depends on the amount of waste, which increases over time, if the amount of waste exceeds a threshold no apples can spawn [7]; therefore, agents need to clean the waste by using clean beam skills for apples to continue to spawn even though staying in the apple field returns more individual rewards. We use decentralized independent actor critic learners and the same environment setup with 2 agents, which we call the 2-Player Cleanup environment, as used by LIO [21] and ID [22] for the 7×7 map.

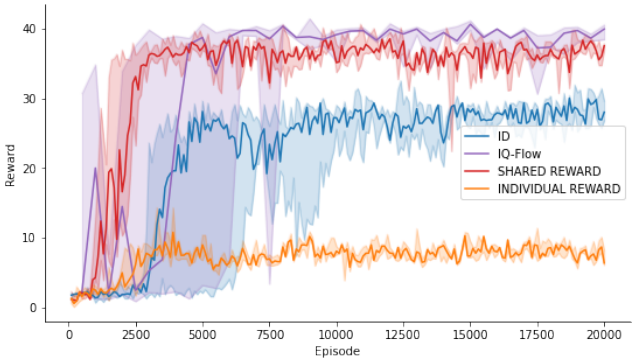

Figure 7: 7×7 experiment result

It can be seen from Figure 7 that IQ-Flow performs the best while reaching the return upper bound, identified by shared reward agent's performance as in LIO [21]. Decentralized actor critic agents perform poor as expected while the decentralized actor critic agents with the shared centralized reward set the return upper bound. Although ID performs close to the return upper bound in both our implementation and the results provided by Yang et al. in [22], it fails to reach it. It should also be noted that while IQ-Flow performs best and reaches the upper bound for good runs, it has high variance close to the end of training for naive training. This variance occurs due to some loss in performance when the actor critic agents' policies get too disconnected from the mechanism. Therefore, in order to obtain a stable training, we reset the actor-critic agents in the environment each 1000 episodes. Since after each reset operation

the actor-critic agents start learning from scratch, we sample evaluation results each 500 episodes in order to filter the pseudo-loss in performance caused by learning from scratch and have a fair comparison with other algorithms.

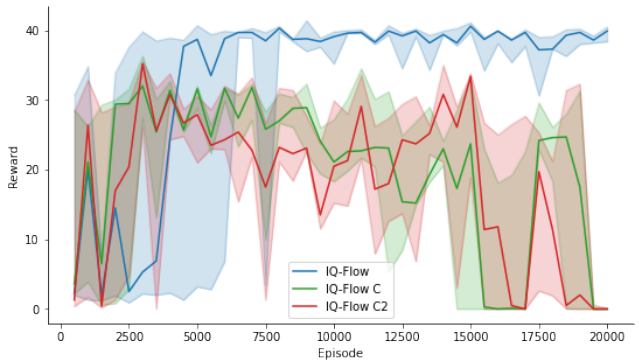

Figure 8: Ablation results

The ablation results for 2-Player Cleanup is given in Figure 8. IQ-Flow denotes the standard algorithm with cost regularization cost 1 and cost 2. IQ-Flow C denotes the case when cost coefficient 1 is 0 and IQ-Flow C2 denotes the case where there is no cost regularization. It is demonstrated that having cost regularization with both coefficients greater than 0 indeed increases learning performance.

The incentive rewards provided by ID and IQ-Flow in 2-Player Cleanup Environment is given in Figure 9.

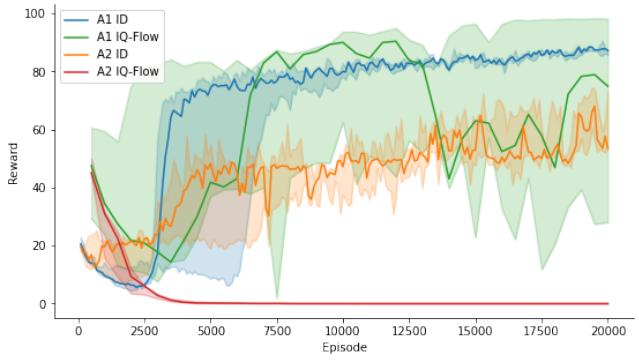

Figure 9: Incentive Rewards given by IQ-Flow and ID

The incentive rewards given by IQ-Flow and ID to agent 1 and agent 2 are presented in Figure 9. Incentive rewards given to agent 1 (A1, cleaner) are in close range with each other for IQ-Flow and ID, but incentive rewards given to agent 2 (A2, harvester) are dissimilar. While ID learns to give an unnecessary incentive to the harvester agent, IQ-Flow learns not to give any unnecessary incentive to this harvester agent. This is attributed to IQ-Flow's capacity to infer when there is a dilemma and when there is no dilemma.

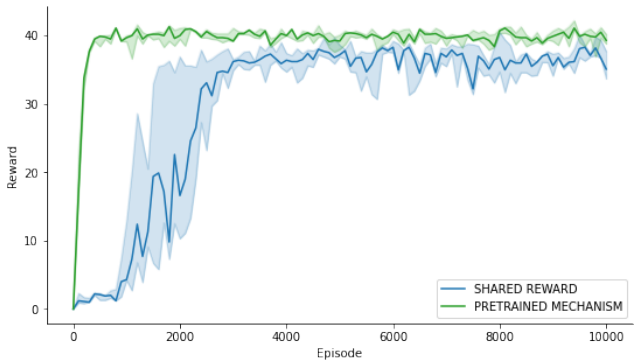

Figure 10: Comparison between pretrained IQ-Flow mechanism and shared reward setup

Finally, we demonstrate in Figure 10 how a reward system supported by a pretrained incentive mechanism by IQ-Flow performs in comparison to a shared reward system. Although the shared reward case with actor-critic agents gives the return upper-bound for 2-Player Cleanup environment, incentivized case with pretrained and frozen IQ-Flow mechanism and actor-critic agents yields much faster learning with higher performance.

6 CONCLUSION

In conclusion, we presented a new algorithm named IQ-Flow to design incentivizers to remove a social dilemma from an environment without any need to perform opponent modelling or access to internal agent parameters. IQ-Flow is fully decentralized and uses the offline RL method Implicit Q-Learning to evaluate policies which are not available in the experienced data. We demonstrated how IQ-Flow modifies the payoff matrix and estimated Q-Values of Iterated Matrix Games for both players, and that it outperforms ID in the existing sequential social dilemma benchmarks. We also demonstrated how much more efficient the reward setup that IQ-Flow produces is than the shared reward case. We consider a promising direction for future work in this area to learn incentive designers with IQ-Flow from offline data with fully offline training so that we can have a method to remove dilemmas from real world that we can not simulate.

ACKNOWLEDGMENTS

We thank Tolga Ok for the valuable discussions throughout this research. Bengisu Guresti thanks the DeepMind scholarship program for the support during her studies. This work is supported by the Scientific Research Project Unit (BAP) of Istanbul Technical University, Project Number: MOA-2019-42321.

REFERENCES

[1] Tobias Baumann, Thore Graepel, and John Shawe-Taylor. 2020. Adaptive Mechanism Design: Learning to Promote Cooperation. *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), 1–7.

[2] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/google/jax

[3] Dan A. Calian, Daniel J Mankowitz, Tom Zahavy, Zhongwen Xu, Junhyuk Oh, Nir Levine, and Timothy Mann. 2021. Balancing Constraints and Rewards with

Meta-Gradient D4{PG}. In *International Conference on Learning Representations*. https://openreview.net/forum?id=TQt98Ya7UMP

[4] 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 Multiagent Systems*. 122–130.

[5] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[6] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2020. *Flax: A neural network library and ecosystem for JAX*. http://github.com/google/flax

[7] 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. *Advances in Neural Information Processing Systems* 31 (2018).

[8] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3040–3049.

[9] Ilya Kostrikov. [n.d.]. JAXRL: Implementations of Reinforcement Learning algorithms in JAX., 10 2021. *URL https://github. com/ikostrikov/jaxrl* ([n. d.]).

[10] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=68n2s9ZJWF8

[11] Paul Lange, Jeff Joireman, Craig Parks, and Eric Dijk. 2013. The Psychology of Social Dilemmas: A Review. *Organizational Behavior and Human Decision Processes* 120 (03 2013), 125–141. https://doi.org/10.1016/j.obhdp.2012.11.003

[12] 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 International Conference on Autonomous Agents and Multiagent Systems*. 464–473.

[13] Alistair Letcher, Jakob N. Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. 2019. Stable Opponent Shaping in Differentiable Games. In *7th International Conference on Learning Representations*.

[14] Michael W. Macy and Andreas Flache. 2002. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences* 99, suppl_3 (2002), 7229–7236. https://doi.org/10.1073/pnas.092080099 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.092080099

[15] Dario Paccagnan, Rahul Chandan, and Jason R. Marden. 2020. Utility Design for Distributed Resource Allocation—Part I: Characterizing and Optimizing the Exact Price of Anarchy. *IEEE Trans. Automat. Control* 65, 11 (2020), 4616–4631. https://doi.org/10.1109/TAC.2019.2961995

[16] Anatol Rapoport. 1974. Game theory as a theory of conflict resolution.

[17] 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 *International Conference on Machine Learning*. PMLR, 4295–4304.

[18] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *CoRR* abs/1706.05296 (2017). arXiv:1706.05296 http://arxiv.org/abs/1706.05296

[19] Eugene [Vinitsky, Natasha Jaques, Joel Leibo, Antonio Castenada, and Edward] Hughes. 2019. An Open Source Implementation of Sequential Social Dilemma Games. https://github.com/eugenevinitsky/sequential_social_dilemma_games/issues/182. GitHub repository.

[20] Zhongwen Xu, Hado P van Hasselt, and David Silver. 2018. Meta-Gradient Reinforcement Learning. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/2715518c875999308842e3455eda2fe3-Paper.pdf

[21] Jiachen Yang, Ang Li, Mehrdad Farajtabar, Peter Sunehag, Edward Hughes, and Hongyuan Zha. 2020. Learning to incentivize other learning agents. *Advances in Neural Information Processing Systems* 33 (2020), 15208–15219.

[22] Jiachen Yang, Ethan Wang, Rakshit Trivedi, Tuo Zhao, and Hongyuan Zha. 2022. Adaptive Incentive Design with Multi-Agent Meta-Gradient Reinforcement Learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1436–1445.

[23] Chongjie Zhang and Victor Lesser. 2010. Multi-Agent Learning with Policy Prediction. https://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1885

[24] Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C Parkes, and Richard Socher. 2020. The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332* (2020).

A APPENDIX

A.1 MARL Critic Losses by Extending Implicit Q-Learning

$$
\begin{aligned}
&Loss_{V^i_{\pi_{coop}}}(\psi_{coop}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1} L_2^{\tau_{exp}}\Big(Q^i_{\bar{\theta}_{coop}}(s_k,a_k) \\
&- V^i_{\psi_{coop}}(s_k)\Big) \\
&Loss_{Q^i_{\pi_{coop}}}(\theta_{coop}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1}\Big(\sum_{j=0}^{N-1} r^j_{env}(s_k,a_k) \\
&+ \gamma V^i_{\psi_{coop}}(s_k^{'}) - Q^i_{\theta_{coop}}(s_k,a_k)\Big)^2 \\
&Loss_{V^i_{\pi_{b,ind}}}(\psi_{ind}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1} L_2^{\tau_{exp}}\Big(Q^i_{\bar{\theta}_{ind}}(s_k,a_k) \\
&- V^i_{\psi_{ind}}(s_k)\Big) \\
&Loss_{Q^i_{\pi_{b,ind}}}(\theta_{ind}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1}\Big(r^i_{ind}(s_k,a_k) \\
&+ \gamma V^i_{\psi_{ind}}(s_k^{'}) - Q^i_{\theta_{ind}}(s_k,a_k)\Big)^2 \\
&Loss_{V^i_{\pi_{env,env}}}(\psi_{env}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1} L_2^{\tau_{exp}}\Big(Q^i_{\bar{\theta}_{env}}(s_k,a_k) \\
&- V^i_{\psi_{env}}(s_k)\Big) \\
&Loss_{Q^i_{\pi_{env,env}}}(\theta_{env}) = \frac{1}{l_B N}\sum_{k=0}^{l_B-1}\sum_{i=0}^{N-1}\Big(r^i_{env}(s_k,a_k) \\
&+ \gamma V^i_{\psi_{env}}(s_k^{'}) - Q^i_{\theta_{env}}(s_k,a_k)\Big)^2
\end{aligned}
\tag{10}
$$

A.2 Implementation Details

In IQ-Flow experiments, the agents are equipped with Actor-Critic networks that have a similar structure with LIO [21] and ID [22]. The input of the agents' actor network is an image, which passes through a single convolution layer with six filters of size [3,3]. The output of the convolution operation is flattened and fed to three consecutive fully connected layers with sizes 64,64,6 respectively. Rectified Linear Unit (ReLU) activation function is applied in all layers except the last layer, where the softmax activation function is used to produce the activation probabilities.

The critic network shares the same structure as the actor network up until the last layer. However, in the critic network, the last layer has a size of 1 and does not apply any activation function.

The IQ-Flow algorithm employs seven neural networks, excluding target critics, each designed with a specific objective. These networks include Q-value (state-action value), V-value (state value), and Incentive (Rewarder) networks of different types. The objectives of these networks are cooperative reward Q-value, cooperative reward V-value, environment reward Q-value, environment reward V-value, incentive reward Q-value, incentive reward V-value, and incentive reward, respectively. To ensure an efficient training, target critic networks are employed for each Q-value network in accordance with Implicit Q-Learning [10]. During training, after updating the agent policies, we perform $K = 20$ critic updates for the meta-update process.

To ensure consistency in state input across environments that do not provide a global state, we implemented a concatenation operation to assume the state as the union of agent observations. When the observations are in the form of 1-D vectors, the concatenation operation is performed first and the resulting concatenated vector is then fed to the networks. For 2-D image observations, all image inputs are first passed through a shared convolution layer, and the resulting embedding vectors are then concatenated to obtain the state more efficiently.

In the case of Q-Value networks and Incentive networks, the assumed state embeddings are concatenated with the union of the agents' actions in 1-hot form for the full action space, where each agent's own actions are masked. We use the full action space for this operation in all experiments of ID and IQ-Flow, as opposed to Yang et al. [22], where the ID algorithm uses binary input indicating whether a cleaning beam was used or not in the Cleanup environment for their Incentive network.

The hyperparameters used in both the implementation and experiments are listed in A.4.

A.3 Experimental Setup

We implement IQ-Flow and ID [21] algorithm with JAX [2, 6, 9] which automatically differentiate through functions, loops, and other operations.

All experiments were run on the following hardware: 2x28 cores Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz, with 4 Nvidia Geforce 2080 Ti graphic cards.

A.4 Hyperparameters

Random exploration rate epsilon is denoted by ϵ, reward discount factor is denoted by γ, policy entropy coefficient is denoted by $c_{entropy}$, learning rate of the actor network is denoted by lr_{actor}, incentive network learning rate is denoted by lr_{reward}, cost regularization coefficients are denoted by $c_{costreg}$ and $c_{costreg2}$, fully connected layer size is denoted by h, and rewarder network fully connected layer size is denoted by hr.

Table 5: PG hyperparameters in Escape Room

Parameter	ER(5,2)	ER(10,5)
γ	0.99	0.99
lr_{actor}	$9.56e^{-5}$	$9.56e^{-5}$
$c_{entropy}$	$1.66e^{-2}$	$1.66e^{-2}$
h_1	64	64
h_2	64	64

Table 6: ID hyperparameters in Escape Room

Parameter	ER(5,2)	ER(10,5)
ϵ_{end}	0.05	0.5
ϵ_{start}	1.0	1.0
γ	0.99	0.99
lr_{actor}	$9.56e^{-5}$	$9.56e^{-5}$
lr_{cost}	$6.03e^{-5}$	$6.03e^{-5}$
$c_{entropy}$	$1.66e^{-2}$	$1.66e^{-2}$
lr_{reward}	$7.93e^{-4}$	$7.93e^{-4}$
c_{reg}	1.0	1.0
h_1	64	64
h_2	64	64
hr_1	64	64
hr_2	32	32

Table 7: ID hyperparameters in 2-Player Cleanup

Parameter	
ϵ_{end}	0.05
ϵ_{start}	1.0
γ	0.99
$c_{entropy}$	0.43554
lr_{actor}	$1.7841e^{-5}$
lr_{reward}	$4.1990e^{-5}$
lr_v	$3.05892e^{-5}$
lr_{vmodel}	$2.1070e^{-5}$
c_{reg}	1e-4
τ	0.1
h_1	64
h_2	64
$filters$	6
$kernel$	[3, 3]

Table 8: IQ-Flow hyperparameters in Prisoner's Dilemma

Parameter	
ϵ_{end}	0.05
ϵ_{start}	1.0
γ	0.99
lr_{cost}	$6.03e^{-5}$
$c_{entropy}$	$1.66e^{-2}$
lr_{actor}	$1e^{-3}$
lr_{reward}	$3e^{-3}$
lr_v	$1e^{-3}$
lr_{vmodel}	$1e^{-3}$
$lr_{vrewarder}$	$1e^{-3}$
c_{reg}	1.0
$c_{costreg}$	$1e^{-2}$
$c_{costreg2}$	$1e^{-4}$

Table 9: IQ-Flow hyperparameters in Escape Room

Parameter	ER(5,2)	ER(10,5)
ϵ_{end}	0.05	0.5
ϵ_{start}	1.0	1.0
γ	0.99	0.99
$c_{entropy}$	$1.66e^{-2}$	$1.66e^{-2}$
lr_{actor}	$9.56e^{-5}$	$9.56e^{-5}$
lr_{reward}	$1e^{-3}$	$1e^{-3}$
lr_{vmodel}	$1e^{-3}$	$1e^{-3}$
$lr_{vrewarder}$	$1e^{-3}$	$1e^{-3}$
$c_{costreg}$	0.5	0.5
$c_{costreg2}$	0.5	0.5
lr_{cost}	$6.03e^{-5}$	$6.03e^{-5}$
h_1	64	64
h_2	64	64
hr_1	64	64
hr_2	32	32

Table 10: IQ-Flow hyperparameters in 2-Player Cleanup

Parameter	
ϵ_{end}	0.05
ϵ_{start}	1.0
γ	0.99
$c_{entropy}$	$8.1439e^{-2}$
lr_{actor}	$1.959e^{-4}$
lr_{reward}	$4.323e^{-6}$
lr_v	$7.445e^{-5}$
lr_{vmodel}	$4.0089e^{-5}$
$lr_{vrewarder}$	$1e^{-3}$
$c_{costreg}$	0.25
$c_{costreg2}$	$1.4924e^{-4}$
τ	0.01
h_1	64
h_2	64
$filters$	6
$kernel$	[3, 3]

B ADDITIONAL ABLATION EXPERIMENTS

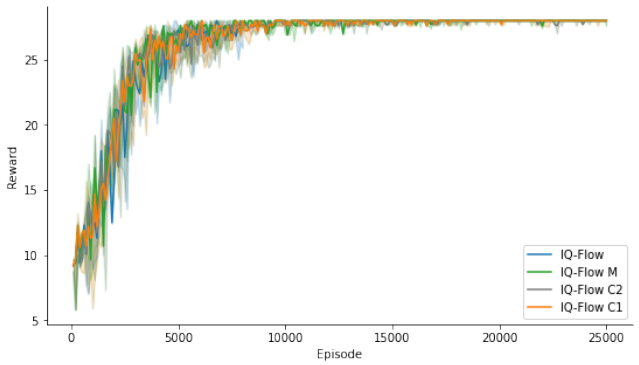

Figure 11: ER(5,2)

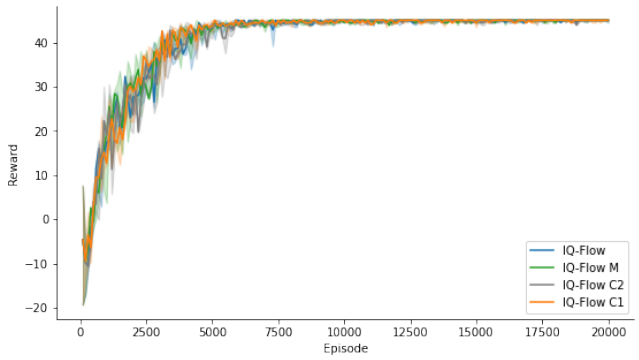

Figure 12: ER(10,5)

C ITERATED MATRIX GAME IQ-FLOW PAYOFF RESULTS

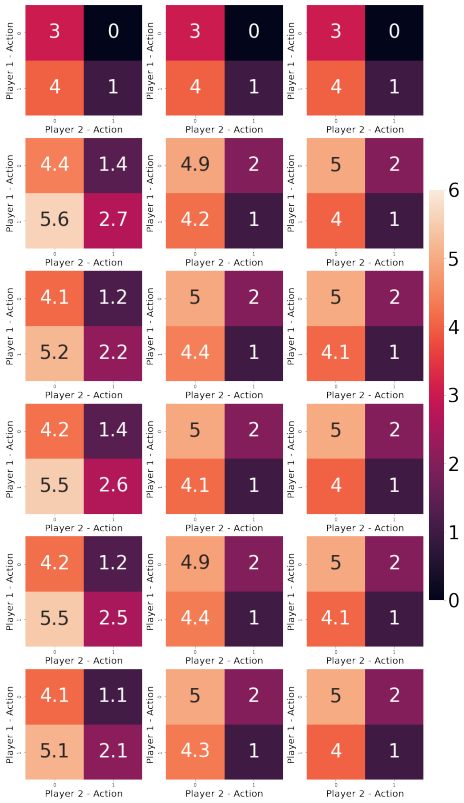

Figure 13: IPD Player 1 Payoff matrices

Figure 14: IPD Player 2 Payoff matrices

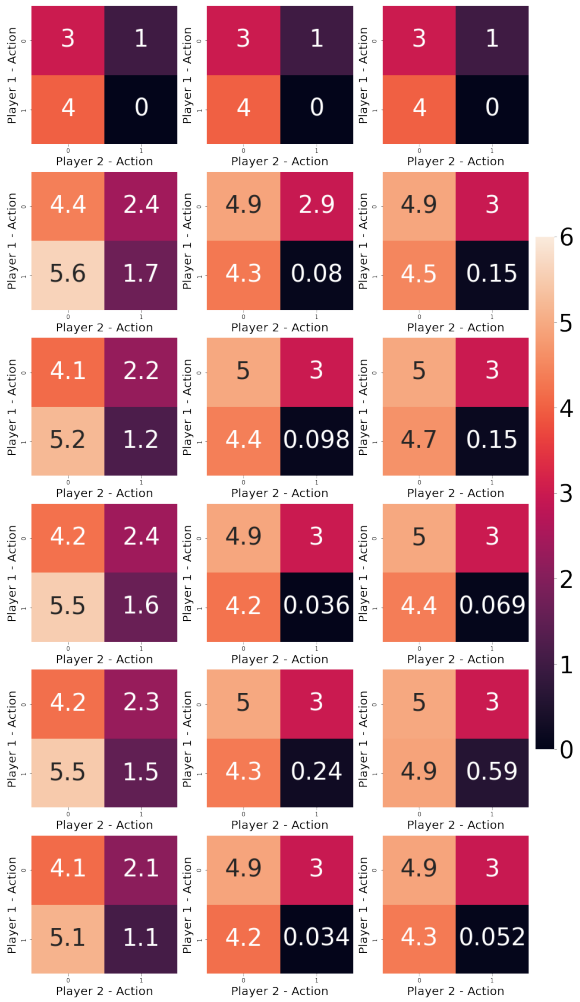

Figure 15: Chicken Game Player 1 Payoff matrices

Figure 16: Chicken Game Player 2 Payoff matrices

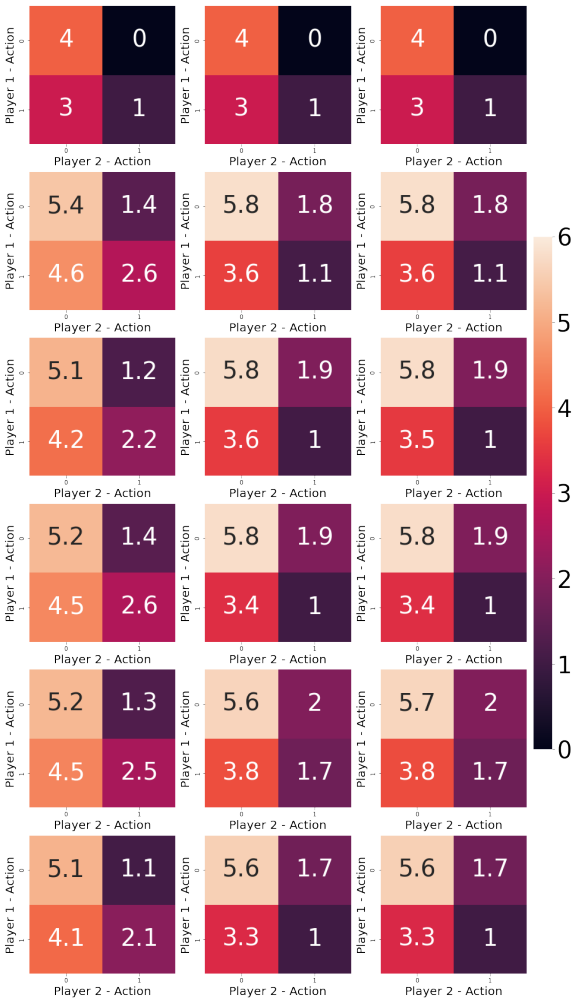

Figure 17: Stag Hunt Player 1 Payoff matrices

Figure 18: Stag Hunt Player 2 Payoff matrices

D ITERATED MATRIX GAME IQ-FLOW Q-VALUE RESULTS

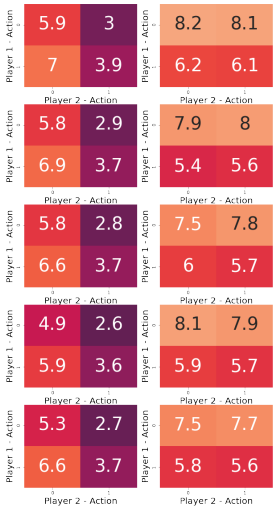

Figure 19: IPD Player 1 Q-Values

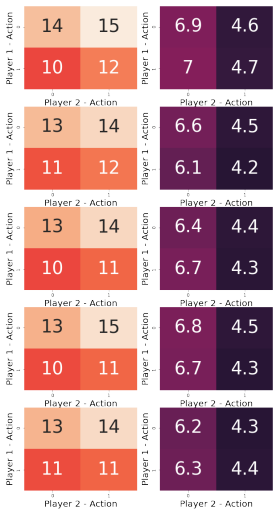

Figure 20: IPD Player 2 Q-Values

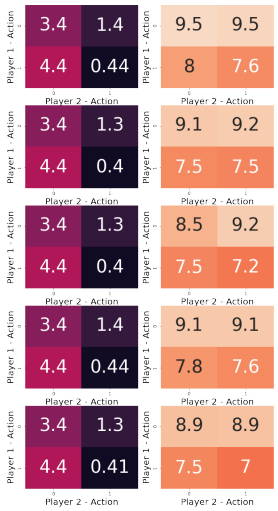

Figure 21: Chicken Game Player 1 Q-Values

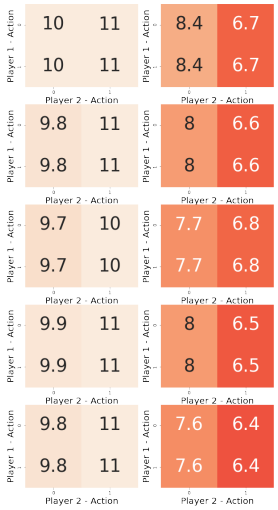

Figure 22: Chicken Game Player 2 Q-Values

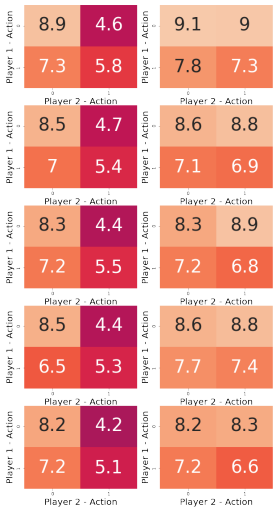

Figure 23: Stag Hunt Player 1 Q-Values

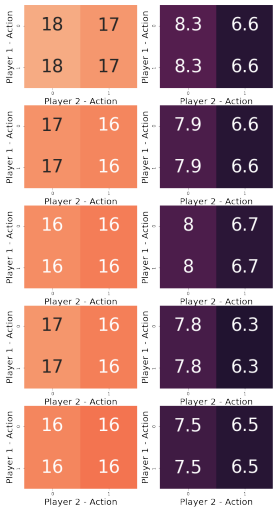

Figure 24: Stag Hunt Player 2 Q-Values

E ITERATED MATRIX GAMES IQ-FLOW R - T AND S - P PLOTS FOR Q-VALUES

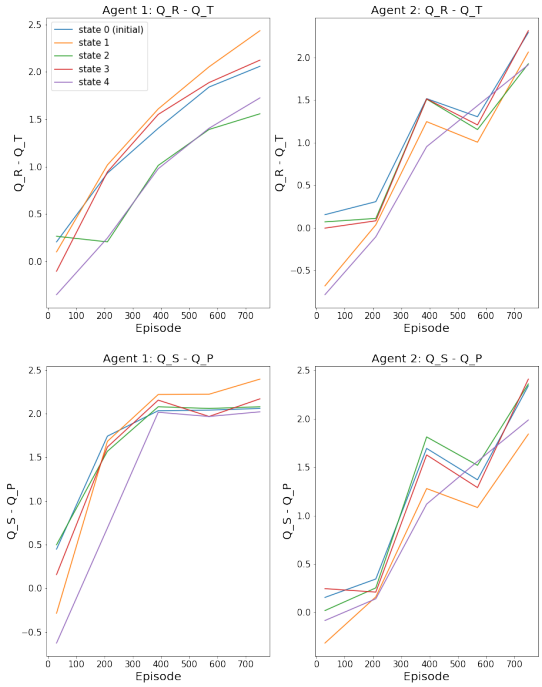

Figure 25: IPD $R - T$ and $S - P$ plot for Q-Values

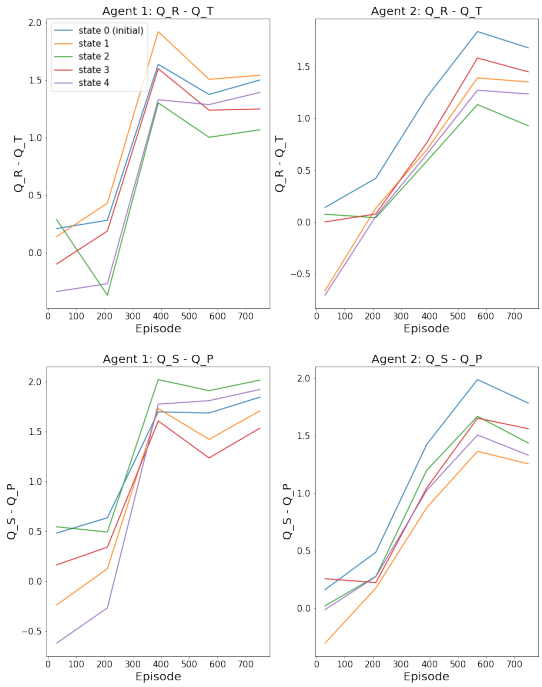

Figure 26: Chicken Game $R-T$ and $S-P$ plot for Q-Values

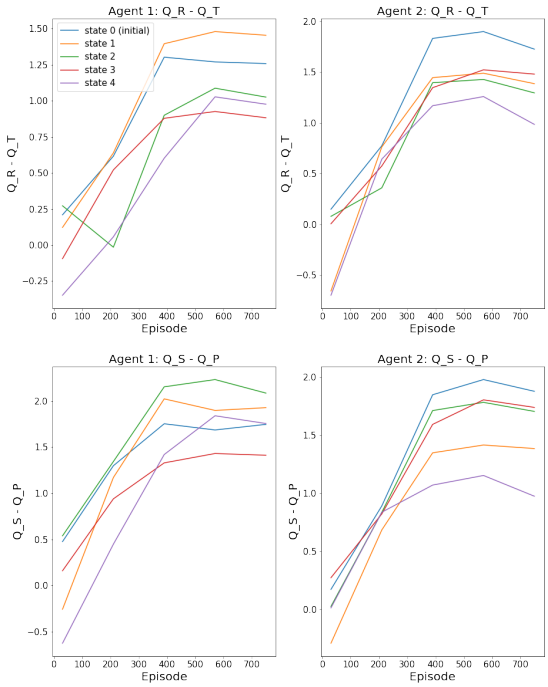

Figure 27: Stag Hunt $R-T$ and $S-P$ plot for Q-Values