I2Q: A Fully Decentralized Q-Learning Algorithm

Anonymous Author(s)
Affiliation
Address
email

Abstract

Fully decentralized multi-agent reinforcement learning has shown great potentials for many real-world cooperative tasks, where the global information, *e.g.*, the actions of other agents, is not accessible. Although independent Q-learning is widely used for decentralized training, the transition probabilities are non-stationary since other agents are updating policies simultaneously, which leads to non-guaranteed convergence of independent Q-learning. To deal with non-stationarity, we first introduce stationary ideal transition probabilities, on which independent Q-learning could converge to the global optimum. Further, we propose a fully decentralized method, I2Q, which performs independent Q-learning on the modeled ideal transition function to reach the global optimum. The modeling of ideal transition function in I2Q is fully decentralized and independent from the learned policies of other agents, helping I2Q be free from non-stationarity and learn the optimal policy. Empirically, we show that I2Q can achieve remarkable improvement in a variety of cooperative multi-agent tasks.

1 Introduction

Multi-agent reinforcement learning (MARL) has shown great potentials in real-world applications, including UAV [17], IoT [3], and games [26]. A number of MARL methods have been proposed for training agents to cooperatively maximize the cumulative shared reward, most of which follow the paradigm of centralized training and decentralized execution (CTDE), where the information of all agents is collected and used in the training phase. However, in many industrial applications where agents may belong to different companies, *e.g.*, autonomous vehicles or robots, the actions of other agents may not be accessible, so that CTDE methods cannot work. One way to address this challenge is *fully decentralized learning*, where the agents only use local experiences without the actions of other agents in both training and execution.

Independent Q-learning [25, 24] is one of the most straightforward decentralized methods. However, since other agents are treated as a part of the environment, from the perspective of individual agent, the transition probabilities that also depend on the policies of other agents will change as other agents are updating their policies simultaneously [7]. Whether the agent updates the individual Q-values using the off-policy experiences, which are stored in the decentralized replay buffer, or the on-policy experiences, which are collected under the latest learned policies of all agents, the transition probabilities are non-stationary, thus the convergence of independent Q-learning is not theoretically guaranteed.

To tackle the non-stationarity problem, we propose a novel variant of independent Q-learning. First, we introduce *ideal transition probabilities* for each agent, which is induced by the optimal conditional joint policy of other agents, conditioned on action of this agent. We theoretically show that if all agents independently perform Q-learning on respective ideal transition probabilities, their policies converge to the optimal joint policy when there is only one optimal joint policy. Certainly, such ideal transition probabilities are unknown in advance, but they can be deliberately learned by each

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

agent in a decentralized way. We let each agent learn the QSS-value [6], the value of *state* and *next state*, which will converge on its replay buffer and be equivalent with the optimal joint Q-value in deterministic environments. Then, for each agent, the next state under ideal transition probabilities, given a state and an action, is the one with the highest QSS-value. Therefore, each agent can model an ideal transition function by learning QSS value and perform independent Q-learning on the ideal transition function, which guarantees the convergence to the optimal policy.

The proposed method, ***ideal independent Q-learning*** (**I2Q**), is fully decentralized, without the information of other agents. Although the theoretical proof is built on deterministic environments, I2Q can also be applied in stochastic environments. We theoretically analyze the value gap and experimentally demonstrate its effectiveness in stochastic environments. We evaluate I2Q on a variety of multi-agent cooperative tasks, *i.e.*, matrix games, MPE-based differential games [12], Multi-Agent MuJoCo [5], and SMAC [20], covering fully and partially observable, deterministic and stochastic, discrete and continuous environments. Empirically, I2Q outperforms baselines, verifying the analyzed convergence and optimality, and the modeled ideal transition function.

2 Related Work

CTDE. Recently, a lot of MARL methods have been proposed to solve cooperative tasks, most of which follow the paradigm of centralized training and decentralized execution (CTDE). Policy-based methods [12, 8, 28, 31, 22, 15] extend policy-gradient into multi-agent cases by designing different optimization objectives. Value factorization methods [23, 19, 21, 27, 18] decompose the joint value function into individual value functions according to the Individual-Global-Max condition. In these methods, the information of all agents can be accessed in a centralized way during training, and the convergence can be guaranteed. Unlike these methods, we focus on decentralized learning where global information is not available.

DTDE. When agents cannot obtain the global information, they have to adopt decentralized training and decentralized execution (DTDE). The most straightforward decentralized methods are independent Q-learning [24] and independent PPO [4]. However, since all agents are updating policies simultaneously, from the perspective of each agent, the transition probabilities are non-stationary, and thus the convergence of these methods may not be guaranteed. Fingerprints [7] uses iteration number and exploration rate to alleviate the problem of obsolete experiences in replay buffer for independent Q-learning. Hysteretic-QL [13] and Lenient-QL [14] let the agents be optimistic and attach less importance to the value punishment. However, none of these variants guarantees the convergence of independent Q-learning. Some theoretical methods [2, 32] guarantee the convergence to a Nash equilibrium, but the converged equilibrium may not be the optimal one when there are multiple equilibria [30]. Moreover, there methods cannot use replay buffer and have to re-collect experiences once the policies are updated, which is not practical. Our I2Q is a practical method to learn the optimal policy using replay buffer.

3 Method

We first introduce the problem setting and analyze the non-stationarity in independent Q-learning. Then, we construct ideal transition probabilities, on which the agents are theoretically guaranteed to converge to the optimal joint policy via independent Q-learning. Finally, we propose I2Q, which performs independent Q-learning on the modeled ideal transition function.

3.1 Preliminaries

There are N agents in multi-agent MDP $M_{\text{env}} =< \mathcal{S}, \mathcal{O}, \mathcal{A}, R, P_{\text{env}}, \gamma >$ with the state space $\mathcal{S}$ and the joint action space $\mathcal{A}$. At each timestep, each agent i performs an individual action a_i, and the environment transitions to the next state s' by taking the joint action $\boldsymbol{a}$ with the transition probabilities $P_{\text{env}}\left(s'|s, \boldsymbol{a}\right)$. For simplicity of theoretical analysis, we let all agents obtain the state s [10], though in practice each agent i may learn based on the observation $o_i \in \mathcal{O}$. All agents get a shared reward $r = R\left(s, s'\right)$ and learn to maximize the expected return $\mathbb{E}\sum_{t=0}^{\infty}\gamma^t r_t$, where γ is the discount factor. We consider the fully decentralized learning, where M_{env} is partially observable to each agent since each agent only observes its own action a_i instead of the joint action $\boldsymbol{a}$. From the perspective of each agent i, there is an MDP $M_i =< \mathcal{S}, \mathcal{A}_i, R, P_i, \gamma >$ with the individual action space $\mathcal{A}_i$ and the

transition probabilities

$$P_i\left(s'|s,a_i\right)=\sum_{\boldsymbol{a}_{-i}}P_{\text{env}}\left(s'|s,\boldsymbol{a}\right)\boldsymbol{\pi}_{-i}(\boldsymbol{a}_{-i}|s), \tag{1}$$

where $\boldsymbol{a}_{-i}$ and $\boldsymbol{\pi}_{-i}$ respectively denote the joint action and the joint policy of all agents except agent i. Each agent performs independent Q-learning:

$$Q_i(s,a_i)=\mathbb{E}_{P_i(s'|s,a_i)}\left[r+\gamma\max_{a_i'}Q_i(s',a_i')\right]. \tag{2}$$

In this paper, all Q values denote the optimal values Q^*, we refer to Q^* as Q for simplicity. According to (1), the transition probabilities depend on the policies of other agents. Since other agents are updating their policies continuously, P_i becomes non-stationary during training. Concretely, if each agent i updates Q_i in an off-policy way, using the experiences stored in the replay buffer $\mathcal{D}_i$, which does not contain other agents' actions, the transition probabilities P_i in the replay buffer could be seen as the ones depending on the average policy $\bar{\boldsymbol{\pi}}_{-i}$ along the training process, which is non-stationary and outdated [7]. If each agent learns in an on-policy way, only using the most recent experiences, P_i could be seen as the one under the latest learned policy $\boldsymbol{\pi}_{-i}$, which are not outdated but still non-stationary. The convergence of Q-learning on non-stationary transition probabilities is not guaranteed. *How to solve the non-stationarity?* In the next section, we introduce ideal transition probabilities and prove that if all agents perform independent Q-learning on the ideal transition probabilities, they will converge to the optimal joint policy.

3.2 Ideal Transition Probabilities

Assume there is only one optimal joint policy $\boldsymbol{\pi}^*(s)=\arg\max_{\boldsymbol{a}}Q(s,\boldsymbol{a})$, where Q is the optimal joint Q-function,

$$Q(s,\boldsymbol{a})=\mathbb{E}_{P_{\text{env}}(s'|s,\boldsymbol{a})}\left[r+\gamma\max_{\boldsymbol{a}'}Q(s',\boldsymbol{a}')\right]. \tag{3}$$

The unique optimal policy $\boldsymbol{\pi}^*(s)$ must be deterministic [16]. Thus, $\boldsymbol{\pi}^*$ can be uniquely factorized into deterministic optimal individual policies $\boldsymbol{\pi}^*=\langle\pi_1^*,\pi_2^*,\cdots,\pi_N^*\rangle$. For agent i, we build a deterministic optimal joint policy of other agents which is conditioned on a_i:

$$\boldsymbol{\pi}_{-i}^*(s,a_i)=\arg\max_{\boldsymbol{a}_{-i}}Q(s,a_i,\boldsymbol{a}_{-i}). \tag{4}$$

There may exist multiple $\boldsymbol{\pi}_{-i}^*$, and we can choose one arbitrarily. If other agents act the policy $\boldsymbol{\pi}_{-i}^*(s,a_i)$, the transition probabilities viewed by agent i is $P_i\left(s'|s,a_i\right)=P_{\text{env}}\left(s'|s,a_i,\boldsymbol{\pi}_{-i}^*(s,a_i)\right)$, which are defined as ***ideal transition probabilities***. Then we have the following theorem.

Theorem 1. *If agent i performs Q-learning on the ideal transition probabilities $P_i\left(s'|s,a_i\right)=P_{\text{env}}\left(s'|s,a_i,\boldsymbol{\pi}_{-i}^*(s,a_i)\right)$, its policy will converge to the optimal individual policy π_i^*.*

Proof. According to the fixed-point formulation of Q-learning, we have

$$Q_i(s,a_i)=\mathbb{E}_{P_{\text{env}}\left(s'|s,a_i,\boldsymbol{\pi}_{-i}^*(s,a_i)\right)}\left[r+\gamma\max_{a_i'}Q_i(s',a_i')\right], \tag{5}$$

$$\max_{\boldsymbol{a}_{-i}}Q(s,a_i,\boldsymbol{a}_{-i})=\max_{\boldsymbol{a}_{-i}}\mathbb{E}_{P_{\text{env}}(s'|s,\boldsymbol{a})}\left[r+\gamma\max_{a_i'}\max_{\boldsymbol{a}_{-i}'}Q(s,a_i',\boldsymbol{a}_{-i}')\right] \tag{6}$$

$$=\mathbb{E}_{P_{\text{env}}\left(s'|s,a_i,\boldsymbol{\pi}_{-i}^*(s,a_i)\right)}\left[r+\gamma\max_{a_i'}\max_{\boldsymbol{a}_{-i}'}Q(s,a_i',\boldsymbol{a}_{-i}')\right], \tag{7}$$

where (6) is from taking $\max_{\boldsymbol{a}_{-i}}$ on both sides of (3), and (7) is by folding $\max_{\boldsymbol{a}_{-i}}$ into P_{env}. From (5) and (7), we see $Q_i(s,a_i)$ and $\max_{\boldsymbol{a}_{-i}}Q(s,a_i,\boldsymbol{a}_{-i})$ have the same stationary Bellman operator. According to the contraction mapping theorem, the Bellman operator converges to a unique fixed point. Thus,

$$Q_i(s,a_i)=\max_{\boldsymbol{a}_{-i}}Q(s,a_i,\boldsymbol{a}_{-i}).$$

Then, we have:

$$\max_{a_i}Q_i(s,a_i)=\max_{a_i}\max_{\boldsymbol{a}_{-i}}Q(s,a_i,\boldsymbol{a}_{-i})=Q(s,\boldsymbol{\pi}^*(s)). \tag{8}$$

By contradiction, if an $\tilde{a}_i \neq \pi_i^*(s)$ but satisfies $Q_i(s, \tilde{a}_i) \geq Q_i(s, \pi_i^*(s)) = Q(s, \boldsymbol{\pi}^*(s))$, it contradicts that $\boldsymbol{\pi}^*$ is the unique optimal joint policy. Therefore, $\pi_i^*(s) = \arg\max Q_i(s, a_i)$. □

By extending this theorem, if all agents perform Q-learning on such ideal transition probabilities under the conditional optimal joint policy, they will converge to the respective optimal individual policies, and arrive at the optimal joint policy $\langle \pi_1^*, \pi_2^*, \cdots, \pi_N^* \rangle$.

3.3 I2Q

How to obtain the ideal transition probabilities from the non-stationary replay buffer for each agent? We first introduce the QSS-learning [6]. Following the setting of QSS [6], we assume the environment is deterministic.[1] Each agent i learns a value function $Q_i^{\text{ss}}(s, s')$ using the experiences in its own replay buffer $\mathcal{D}_i$:

$$Q_i^{\text{ss}}(s, s') = r + \gamma \max_{s'' \in \mathcal{N}(s')} Q_i^{\text{ss}}(s', s''), \tag{9}$$

where $\mathcal{N}(s')$ is the neighboring state set of the state s' (the set of all next states of s'). The value function Q_i^{ss} has several advantages. First, it has been proven in QSS [6] that Q_i^{ss} and Q learn equivalent values, which means

$$\max_{s'} Q_i^{\text{ss}}(s, s') = \max_{\boldsymbol{a}} Q(s, \boldsymbol{a}). \tag{10}$$

The equivalence shows that agent i can independently infer the next state under the optimal joint policy according to $\arg\max_{s'} Q_i^{\text{ss}}(s, s')$. Second, as $Q_i^{\text{ss}}(s, s')$ decouples the action a_i from the value and the implied transition probabilities: $P(s'|s, s') = 1, \forall s' \in \mathcal{N}(s)$, are always stationary, Q_i^{ss} converges to the optimal value even on its own replay buffer, without the information of other agents. Third, since all agents act in the same environment, they collect the same states set and the same next state set in their own replay buffers. Therefore, they will converge to the same Q_i^{ss} independently, which builds the consensus between the learned policies of agents.

Then, how to build the ideal transition function from Q_i^{ss}? As the environment P_{env} and the optimal conditional joint policy of other agents $\boldsymbol{\pi}_{-i}^*(s, a_i)$ are deterministc, under the ideal transition probabilities $P_{\text{env}}\left(s'|s, a_i, \boldsymbol{\pi}_{-i}^*(s, a_i)\right)$, the environment from the perspective of agent i will deterministically transition to the next state that has the highest Q_i^{ss} among all neighboring states given a_i:

$$s'^* = \arg\max_{s' \in \mathcal{N}(s, a_i)} Q_i^{\text{ss}}(s, s'), \tag{11}$$

where $\mathcal{N}(s, a_i)$ if the neighboring state set of s given a_i. This is proved in the proof of the following theorem.

Theorem 2. *In deterministic environments, all agents will converge to the optimal policies, if each agent i performs Q-learning on the transition function $s'^* = \arg\max_{s' \in \mathcal{N}(s, a_i)} Q_i^{\text{ss}}(s, s')$.*

Proof. The proof is given in Appendix A, which shows that $s'^* = \arg\max_{s' \in \mathcal{N}(s, a_i)} Q_i^{\text{ss}}(s, s')$ is the ideal transition function. □

To model the stationary ideal transition function $s'^* = \arg\max_{s' \in \mathcal{N}(s, a_i)} Q_i^{\text{ss}}(s, s')$, we train a neural network $f_i(s, a_i)$ to predict s'^* and update $f_i(s, a_i)$ by maximizing:

$$\mathbb{E}_{s, a_i, s' \sim \mathcal{D}_i}\left[\lambda Q_i^{\text{ss}}(s, f_i(s, a_i)) - (f_i(s, a_i) - s')^2\right]. \tag{12}$$

The first term enforces that the predicted next state has the highest Q_i^{ss}, and the second term constrains the predicted next state to be in the set $\mathcal{N}(s, a_i)$. The hyperparameter λ is the coefficient. Based on the transition model $f_i(s, a_i)$, Q_i^{ss} is updated by minimizing the TD-error:

$$\mathbb{E}_{s, a_i, s', r \sim \mathcal{D}_i}\left[\left(Q_i^{\text{ss}}(s, s') - r - \gamma \bar{Q}_i^{\text{ss}}(s', f_i(s', a_i'^*))\right)^2\right], \quad a_i'^* = \arg\max_{a_i'} Q_i(s', a_i'). \tag{13}$$

And Q_i is updated by minimizing:

$$\mathbb{E}_{s, a_i, r \sim \mathcal{D}_i}\left[\left(Q_i(s, a_i) - r - \gamma \max_{a_i'} \bar{Q}_i(f_i(s, a_i), a_i')\right)^2\right]. \tag{14}$$

[1] Although the environment P_{env} is deterministic, from the perspective of each agent i, the viewed environment P_i would still be non-stationary.

Algorithm 1. I2Q for each agent i

Initialize transition model f_i, Q-networks Q_i and Q_i^{ss}, and the target networks $\bar{Q}_i$ and $\bar{Q}_i^{\text{ss}}$.
Initialize the replay buffer $\mathcal{D}_i$.
for $t = 1, \ldots, max_iteration$ **do**
  All agents interact in the environment and store experiences (s, a_i, s', r) in replay buffer $\mathcal{D}_i$.
  Sample a mini-batch from $\mathcal{D}_i$.
  Update f_i by maximizing (12).
  Update Q_i^{ss} by minimizing (13).
  Update Q_i by minimizing (14) or (15).
  Update the target networks $\bar{Q}_i$ and $\bar{Q}_i^{\text{ss}}$.
end for

$\bar{Q}$ is the target network of Q. When updating Q_i (14), the target value is not computed on the next state sampled from replay buffer D_i, but on the one predicted by the modeled ideal transition function $f_i(s, a_i)$. In (14), the reward r is simplified to depending just on s. When r depends on both s and s', given s and a_i, the next state under the ideal transition function (11) satisfies:

$$\begin{aligned} Q_i^{\text{ss}}(s, s'^*) &= r(s, s'^*) + \gamma \max_{s''} Q_i^{\text{ss}}(s'^*, s'') \\ &= r(s, s'^*) + \gamma \max_{\boldsymbol{a}'} Q(s'^*, \boldsymbol{a}') \leftarrow (10) \\ &= r(s, s'^*) + \gamma \max_{a_i'} Q_i(s'^*, a_i') \leftarrow (8) \\ &= Q_i(s, a_i) \leftarrow (\text{deterministic ideal transition probabilities}). \end{aligned}$$

Therefore, we can update Q_i by minimizing:

$$\mathbb{E}_{s, a_i \sim \mathcal{D}_i} \left[\left(Q_i\left(s, a_i\right) - \bar{Q}_i^{\text{ss}}(s, f_i(s, a_i)) \right)^2 \right]. \tag{15}$$

The training procedure of I2Q is summarize in Algorithm 1, where each agent i learns f_i, Q_i^{ss}, and Q_i. Although all the modules update simultaneously, as the convergence of Q_i^{ss} is guaranteed, the transition function f_i derived from Q_i^{ss} will be stationary in the later stage. Thus Q_i will also converge. I2Q can be applied in environments with both discrete and continuous state-action space. In continuous action space, we build I2Q on DDPG [11], where a policy network $\pi_i(s)$ is trained by maximizing $Q_i(s, \pi_i(s))$ as a substitute of $\arg\max_{a_i} Q_i(s, a_i)$. In continuous state space, since $f_i(s, a_i)$ is differentiable, (12) can be maximized by gradient ascent. In discrete state space or large state space, we can map the state space to a continuous embedding space, and apply I2Q on the embedding space.

3.4 Assumptions

We will further discuss the two assumptions. The prime one is that there is only one optimal joint policy, but I2Q can easily solve tasks with multiple optimal joint policies. With multiple optimal actions (with the max $Q_i(s, a_i)$), if each agent arbitrarily selects one of the optimal independent actions, the joint action might not be the optimal. To address this, we can set a performance tolerance ε and introduce a fixed randomly initialized reward function $\hat{r}(s, s')$ in the range $(0, \hat{r}_{\max}]$, where $\hat{r}_{\max} = (1 - \gamma)\varepsilon$. Then all agents perform I2Q on the shaped reward $r + \hat{r}$ and learn the value function $\hat{Q}_i(s, a_i)$ in terms of of $r + \hat{r}$. Since $\hat{r}$ is positive, $\hat{Q}_i(s, a_i) > Q_i(s, a_i)$. In $\hat{Q}_i(s, a_i)$, the maximal contribution from $\hat{r}$ is $\hat{r}_{\max}/(1 - \gamma) = \varepsilon$, so the minimal contribution from r is $\hat{Q}_i(s, a_i) - \varepsilon > Q_i(s, a_i) - \varepsilon$, which means that the maximal performance drop is ε when selecting actions according to $\hat{Q}_i$. Moreover, since the reward function $\hat{r}(s, s')$ is randomly initialized, it is a small probability event to find multiple optimal joint policies on the reward function $r + \hat{r}$. Thus, if ε is set to be small enough, I2Q could solve the task with multiple optimal joint policies.

The secondary one is deterministic environments. In stochastic environments, (10) does not hold, and $Q_i^{\text{ss}}(s, s')$ can be considered the "best possible value" [6]. So $s'^* = \arg\max_{s' \in \mathcal{N}(s, a_i)} Q_i^{\text{ss}}(s, s')$ is the best possible transition function, rather than the ideal transition function. The difference between the learned Q_i of I2Q and the true value under the ideal transition function Q_i^{sto} is guaranteed by the following theorem.

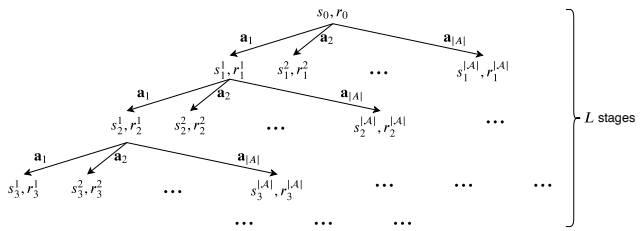
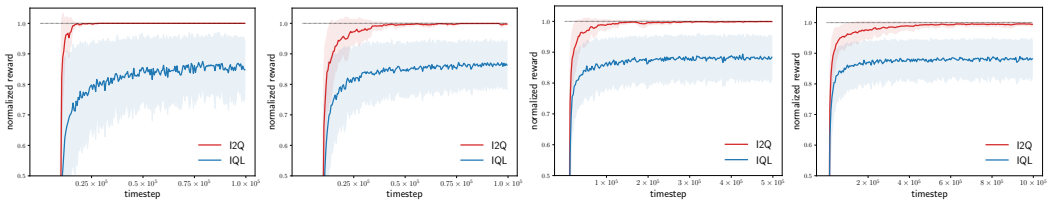


Figure 1: Illustration of the matrix games.



(a) $L = 3$ (b) $L = 4$ (c) $L = 5$ (d) $L = 6$

Figure 2: Learning curves on 100 random matrix games with different stage L.

Theorem 3. $\|Q_i^{sto} - Q_i\|_\infty \leq \frac{\Delta r \left\| 1 - P_{\text{env}}\left(s'^* | s, a_i, \boldsymbol{\pi}^*_{-i}(s, a_i)\right) \right\|_\infty}{(1-\gamma)^2}$, *where* $\Delta r = r_{\max} - r_{\min}$.

Proof. The proof is given in Appendix A. □

Theorem 3 shows that Q_i of I2Q is closer to the true value if the transition probability of s'^* is higher in stochastic environments. Moreover, we empirically show that I2Q can also obtain remarkable performance gain in stochastic tasks.

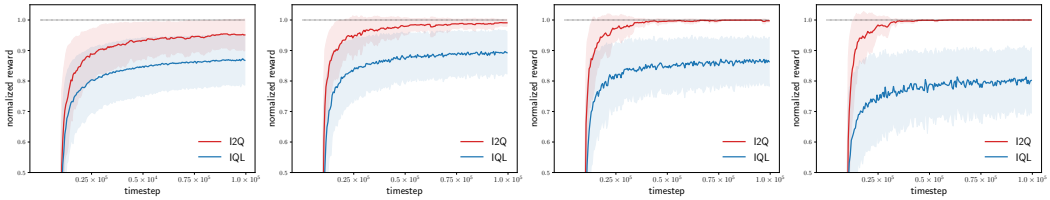
4 Experiments

In experiments, we first evaluate I2Q on a class of randomly generated matrix games to verify our theoretical analysis. Second, we compare I2Q with a series of Q-learning variants on MPE-based [12] differential games to illustrate the convergence and optimality of I2Q on more complex tasks with continuous action. Third, we test I2Q on two popular MARL benchmarks: Multi-Agent MuJoCo [15] and SMAC [20]. The experiments cover both fully and partially observable, deterministic and stochastic, discrete and continuous environments. Since we consider the fully decentralized settings, I2Q and the baselines *do not use parameter-sharing*. The results are presented using mean and standard deviation (std) with different random seeds. More details about hyperparameters are available in Appendix C.

4.1 Matrix Games

To support the theoretical analysis of I2Q, we perform experiments on a class of two-agent matrix games with L stages. The action space of each agent is 3, so the joint action space $|\mathcal{A}| = 9$. The payoff tree is shown in Figure 1, where the reward r is randomly generated in the range $[-1, 1]$ and fixed in each matrix game. Without loss of generality, we randomly generate 100 matrix games for each stage L, train the agents for four different seeds in each game, and plot the mean normalized return (normalized by the optimal return) and std over the 100 matrix games. Since the states are countable, we adopt Q-tables instead of neural networks, where Q_i table and Q_i^{ss} table are updating *simultaneously*, and use ϵ-greedy policies. Figure 2 show the learning curves with different stages L under $\epsilon = 0.5$. IQL cannot converge to the optimal policies and shows large std, because it is afflicted with the non-stationarity. I2Q converges to the global optimum on the matrix games, even when the stage L grows, which confirms our theoretical analysis.

In Figure 3, we show the effect of exploration rate ϵ on the matrix games with $L = 4$. As ϵ increases, I2Q converges faster, while the performance of IQL drops. In conventional decentralized methods like IQL, there is a *dilemma of exploration and exploitation*. If ϵ is small, the weak exploration causes low sample efficiency and slow learning. When ϵ is large, all agents act more randomly. From the perspective of individual agent, the transition probabilities will be much different from the real ones



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.3$ (c) $\epsilon = 0.5$ (d) $\epsilon = 0.7$

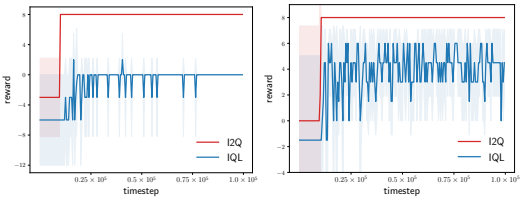
Figure 3: Learning curves with different ϵ on 100 random matrix games with stage L.

a_2 \ a_1	$\mathcal{A}^{(1)}$	$\mathcal{A}^{(2)}$	$\mathcal{A}^{(3)}$
$\mathcal{A}^{(1)}$	**8**	-12	-12
$\mathcal{A}^{(2)}$	-12	0	0
$\mathcal{A}^{(3)}$	-12	0	0

(a) case 1

a_2 \ a_1	$\mathcal{A}^{(1)}$	$\mathcal{A}^{(2)}$	$\mathcal{A}^{(3)}$
$\mathcal{A}^{(1)}$	**8**	-12	-12
$\mathcal{A}^{(2)}$	-12	6	0
$\mathcal{A}^{(3)}$	-12	0	6

(b) case 2



(c) curves on case 1 (d) curves on case 2

Figure 4: Learning curves on two specific one-stage matrix games.

under the learned policies of other agents, which causes bad performance. That is the reason why IQL drops when ϵ increases. I2Q can avoid the dilemma, because $Q_i^{\text{ss}}(s, s')$ is independent from agent policies and converges faster with larger ϵ as it sees more states. Thus, I2Q can use large ϵ, *e.g.*, 0.5 or 0.7, to promote sample efficiency.

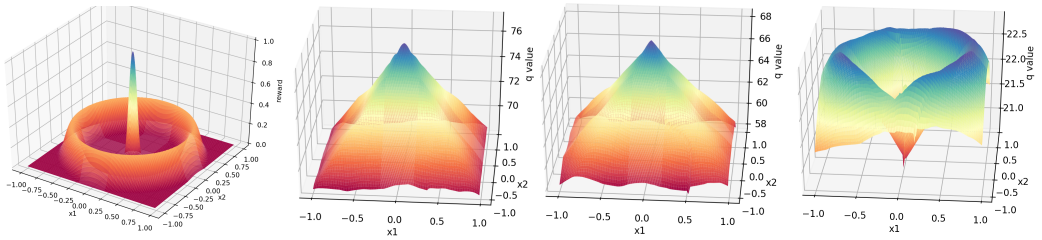
Moreover, we test I2Q on two one-stage matrix games proposed in QTRAN [21] and QPLEX [27] as shown in Figure 4, which are special cases of Figure 1. In case 1, IQL converges to the local optimum, and in case 2, IQL does not converge. Our I2Q can converge to the global optimum easily, in a fully decentralized way.

4.2 MPE

To investigate the effectiveness of I2Q in complex continuous environments with neural network implementation, we design a class of MPE-based differential games, where N agents can move in the range $[-1, 1]$. In each timestep, agent i acts the action $a_i \in [-1, 1]$, and the position of agent i is updated as $x_i = \text{clip}(x_i + 0.1 \times a_i, -1, 1)$ (*i.e.*, the updated position is clipped to $[-1, 1]$). The state is the position vector $\{x_1, x_2, \cdots, x_N\}$. The reward function of each timestep is defined as

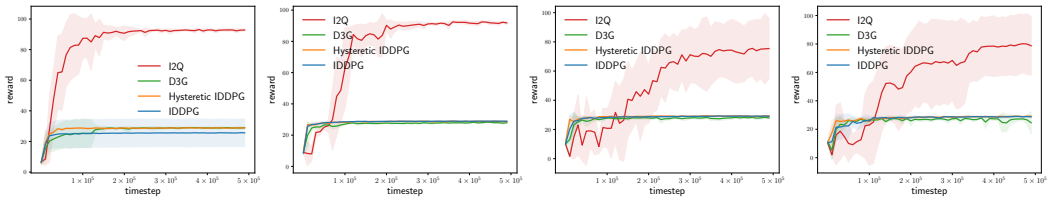
$$r = \begin{cases} 0.5\cos(l\pi/m) + 0.5 & \text{if } l \leq m \\ 0 & \text{if } m < l \leq 0.6 \\ 0.15\cos(5\pi(l - 0.8)) + 0.15 & \text{if } 0.6 < l \leq 1.0 \\ 0 & \text{if } l > 1.0 \end{cases}, \quad l = \sqrt{\frac{2}{N}\sum_{i=0}^{N} x_i^2}, \quad m = 0.13(N-1).$$

The visualization of reward function of two-agent case is shown in Figure 5a. There is only one global optimum ($l = 0$ and $r = 1$) but infinite sub-optimum ($l = 0.8$ and $r = 0.3$), and the region with $r > 0.3$ is very narrow and surrounded by the region with $r = 0$. So it is hard to learn the optimal policies in a decentralized way. The episode contains 100 timesteps, and the agents' positions are randomly initialized at the beginning of each episode. To verify the scalability, we test I2Q in the settings with different agent numbers N, and train the agents for eight random seeds in each setting. The results are shown in Figure 6. IDDPG always falls into local optimum due to the non-stationarity and outdated transition probabilities. To be optimistic towards the value punishment is an important technique in decentralized MARL. Although adopting the optimistic update, Hysteretic IDDPG [13] still falls into local optimum due to the non-stationarity problem. D3G [6] applies the idea of QSS in single agent learning. To extend D3G into decentralized MARL, we train each agent independently using D3G, without the information of other agents. However, D3G cannot learn the optimal solution either, since the forward model in D3G requires the transition probabilities P_i to be deterministic. However, when learning on the decentralized replay buffer, P_i is the average of non-stationary transition probabilities and cannot be deterministic. Thus, D3G is not applicable to decentralized multi-agent learning. Learning on the ideal transition function inferred from $Q_i^{\text{ss}}(s, s')$, I2Q can escape from local optimum and significantly outperforms others. To thoroughly demonstrate the effectiveness of I2Q, in Figure 5, we visualize the learned values of agent 1 in the two-agent case.



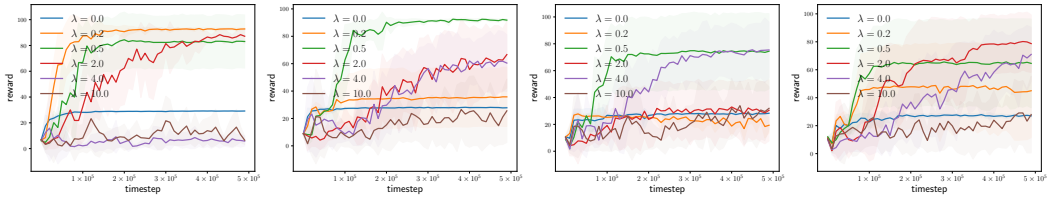
(a) reward function (b) Ideal (c) I2Q (d) IDDPG

Figure 5: (a): Visualization of reward function in differential game with two agents. (b)-(d): Visualizations of learned values of agent 1 in differential game with two agents. x1-axis and x2-axis: agents 1 and 2's positions, respectively.



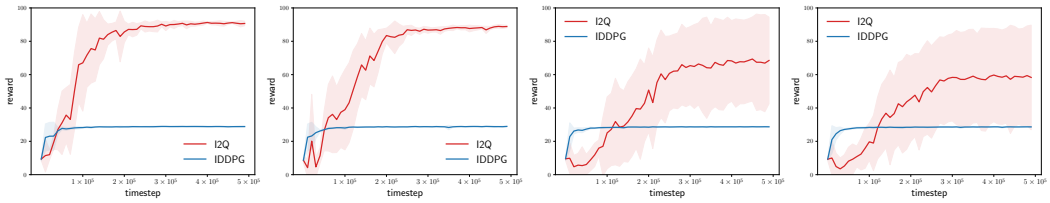
(a) $N = 2$ (b) $N = 3$ (c) $N = 4$ (d) $N = 5$

Figure 6: Learning curves on MPE-based differential games with different agent numbers N.



(a) $N = 2$ (b) $N = 3$ (c) $N = 4$ (d) $N = 5$

Figure 7: Learning curves on MPE-based differential games with different λ.

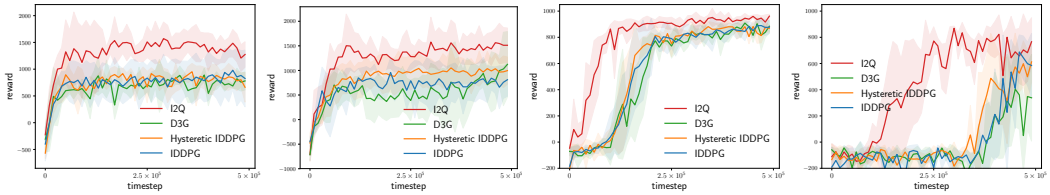


(a) $\xi = 0.02$ (b) $\xi = 0.03$ (c) $\xi = 0.04$ (d) $\xi = 0.05$

Figure 8: Learning curves on MPE-based differential games with different noise ξ.

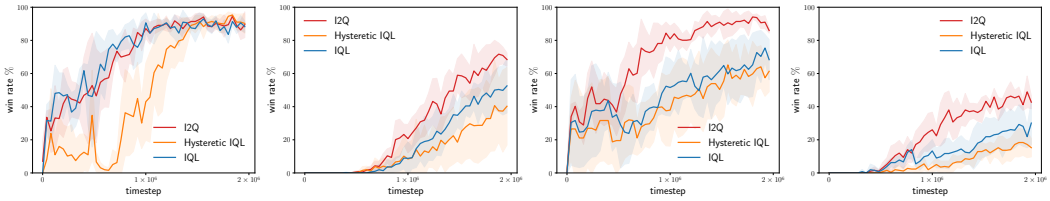
In Figure 5b, we let the agent 2 always act the optimal policy (moving to the position $x_2 = 0$), which means the agent 1 performs Q-learning on the true ideal transition probabilities. Thus, Figure 5b show the optimal values under ideal transition probabilities. In Figure 5c, both agents use I2Q. I2Q values are very similar to the true ideal values, where the value of the center point ($l = 0$) is the highest, which means that the modeled transition function in I2Q is close to the true one. In Theorem 4 (see Appendix A), we theoretically analyze that the difference between ideal values (Figure 5b) and I2Q values (Figure 5c) is bounded by the model error. In Figure 5d, both agents use IQL, resulting in that the value of center point ($l = 0$) is the lowest and the suboptimal points ($l = 0.8$) have the highest values.

To model the ideal transition function, next states predicted by f_i should have the highest Q_i^{ss} values (optimality) and be in the neighboring state set given state-action pair (referred to as neighborhood constraint). To achieve this objective, λ in (12) controls the balance between optimality and neighborhood constraint. Figure 7 shows the effect of λ. When λ is too small, the next state predicted by f_i cannot maximize $Q_i^{\text{ss}}(s, s')$, and thus I2Q gets stuck at local optimum. Especially, when $\lambda = 0$, I2Q degrades into a popular model-based method MBPO [9]. When λ is too large, *e.g.*, $\lambda = 10$, paying much more attention to maximize $Q_i^{\text{ss}}(s, s')$, f_i would generate out-of-neighborhood states, causing that I2Q even cannot converge to local optimum. We notice that $\lambda = 0.5$ can generally achieve a good performance in different settings of N.



(a) 2×3 HalfCheetah (b) $2|4$ HalfCheetah (c) 2×4 Ant (d) $6|2$ Ant

Figure 9: Learning curves on Multi-Agent MuJoCo.



(a) 8m (b) 3s_vs_4z (c) 2s3z (d) 5m_vs_6m

Figure 10: Learning curves on SMAC.

Although we assume deterministic environments in theoretical analysis, I2Q can also obtain performance gain in stochastic environments. To measure the impact of stochastic environments, we add a noise to the position update: $x_i = \text{clip}(x_i + 0.1 \times a_i + \xi \times z, -1, 1)$, where z is a uniform random variable in $[-1, 1]$, and ξ is a constant controling the randomness. Figure 8 shows the performance with different ξ in the three-agent case. In the stochastic environments, IDDPG still converges to local optimum, while I2Q always escapes from local optimum when $\xi = 0.02$ and 0.03, free from the impact of stochastic environments. When $\xi = 0.04$ and 0.05, the effect of stochasticity is nearly half of the effect of action. I2Q cannot always learn optimal policies with the strong stochasticity but still significantly outperforms IDDPG.

4.3 Multi-Agent MuJoCo and SMAC

To investigate the effectiveness of I2Q in *partially observable* environments, we perform experiments on Multi-Agent MuJoCo [15], where each agent independently controls one or some joints of the robot and could only observe the state of its own joints and bodies (with the parameter agent_obsk = 0). The results are shown in Figure 9 with eight random seeds. I2Q achieves higher reward than IDDPG, which indicates that I2Q could be extended to partially observable tasks and obtain performance gain by performing Q-learning on ideal transition probabilities instead of transition probabilities in replay buffer. In the partially observable setting, we only consider two-agent cases. When there are more agents, each agent can only control one or two joints, and the observation range is too limited to learn strong policies. In Appendix B.4, we provide the 6-agent Walker and 8-agent Ant experiments with full observation setting to verify the scalability.

We test I2Q on *partially observable and stochastic* SMAC tasks [20] with the version SC2.4.10, including both easy and hard maps [29]. We adopt the implementation of PyMARL [20] that takes as input the trajectory of partial observations, which is high-dimensional and not differentiable on some dimensions, meaning that directly optimizing (12) is hard. So the QSS model Q_i^{SS} is not built on the trajectory of partial observations, but on the hidden state of Q_i (the output of RNN layer), which is 64-dimension and differentiable, and the transition function f_i predicts the next hidden state. We train the agents for four random seeds. D3G can only be used in continuous action space, so we do not compare with it in SAMC. As shown in Figure 10, I2Q is capable of handling high-dimensional and stochastic tasks.

5 Conclusion

We propose I2Q, a novel variant of independent Q-learning, to deal with the non-stationarity in decentralized MARL. I2Q models the ideal transition function, which depend on the optimal conditional joint policy of other agents and performs independent Q-learning on the modeled ideal transition function. Theoretically, we prove that I2Q converges to the optimal joint policy in deterministic environments. Empirically, I2Q obtains performance gain in a variety of multi-agent tasks.

References

[1] Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018.

[2] Natalia Akchurina. Multiagent reinforcement learning: algorithm converging to nash equilibrium in general-sum discounted stochastic games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 725–732. Citeseer, 2009.

[3] Zilong Cao, Pan Zhou, Ruixuan Li, Siqi Huang, and Dapeng Wu. Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0. *IEEE Internet of Things Journal*, 7(7):6201–6213, 2020.

[4] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is Independent Learning All You Need in The StarCraft Multi-Agent Challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[5] Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Deep Multi-Agent Reinforcement Learning for Decentralized Continuous Cooperative Control. *arXiv preprint arXiv:2003.06709*, 2020.

[6] Ashley Edwards, Himanshu Sahni, Rosanne Liu, Jane Hung, Ankit Jain, Rui Wang, Adrien Ecoffet, Thomas Miconi, Charles Isbell, and Jason Yosinski. Estimating q (s, s') with deep deterministic dynamics gradients. In *International Conference on Machine Learning (ICML)*, 2020.

[7] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2017.

[8] Shariq Iqbal and Fei Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2019.

[9] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[10] Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2022.

[11] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[12] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Information Processing Systems (NeurIPS)*, 2017.

[13] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007.

[14] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient Multi-Agent Deep Reinforcement Learning. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018.

[15] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[16] Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming, 1994.

[17] Han Qie, Dianxi Shi, Tianlong Shen, Xinhai Xu, Yuan Li, and Liujing Wang. Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE access*, 7:146264–146272, 2019.

[18] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2018.

[20] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[21] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning To Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2019.

[22] Kefan Su and Zongqing Lu. Divergence-Regularized Multi-Agent Actor-Critic. *arXiv preprint arXiv:2110.00304*, 2021.

[23] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

[24] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

[25] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning (ICML)*, 1993.

[26] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature*, 575(7782):350–354, 2019.

[27] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[28] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. Dop: Off-policy multi-agent decomposed policy gradients. In *International Conference on Learning Representations (ICLR)*, 2020.

[29] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[30] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

[31] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. FOP: Factorizing Optimal Joint Policy of Maximum-Entropy Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2021.

[32] Zhengyuan Zhou, Panayotis Mertikopoulos, Nicholas Bambos, Peter Glynn, and Claire Tomlin. Multi-agent online learning with imperfect information, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3.4.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] As we provide a new method for decentralized MARL, we do not see any potential negative societal impacts at this stage.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We have clearly stated the assumptions in Section 3.2 and 3.3, and further discussed them in Section 3.4.
   (b) Did you include complete proofs of all theoretical results? [Yes] We include the proof of Theorem 1 in Section 3.2, and include the proofs of Theorem 2,3,4 in Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We have not cleaned the code now, and we will certainly release the code after this paper is accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include the training details in Appendix C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We include the details of computation in Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL?[N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We do not use any dataset in this paper.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]