Explicitly Coordinated Policy Iteration

Yujing Hu[1] , **Yingfeng Chen**[1] , **Changjie Fan**[1] and **Jianye Hao**[2]
[1]Fuxi AI Lab in Netease
[2]Tianjin University
{huyujing, chenyingfeng1, fanchangjie}@corp.netease.com
jianye.hao@tju.edu.cn

Abstract

Coordination on an optimal policy between independent learners in fully cooperative stochastic games is difficult due to problems such as relative overgeneralization and miscoordination. Most state-of-the-art algorithms apply fusion heuristics on agents' optimistic and average rewards, by which coordination between agents can be achieved implicitly. However, such implicit coordination faces practical issues such as tedious parameter-tuning in real world applications. The lack of an explicit coordination mechanism may also lead to a low likelihood of coordination in problems with multiple optimal policies. Based on the necessary conditions of an optimal policy, we propose the *explicitly coordinated policy iteration* (EXCEL) algorithm which always forces agents to coordinate by comparing the agents' separated optimistic and average value functions. We also propose three solutions for deep reinforcement learning extensions of EXCEL. Extensive experiments in matrix games (from 2-agent 2-action games to 5-agent 20-action games) and stochastic games (from 2-agent games to 5-agent games) show that EXCEL has better performance than the state-of-the-art algorithms (such as faster convergence and better coordination).

1 Introduction

As the extension of reinforcement learning (RL) to the multi-agent domain, multi-agent reinforcement learning (MARL) has been widely studied over the last twenty years [Buşoniu *et al.*, 2008; Shoham *et al.*, 2003]. Different from single-agent RL where an agent maximizes its expected accumulative rewards, MARL is more complicated which involves different types of agent relationship (competition, cooperation, or mixture of the both), various learning goals (e.g., convergence and rationality [Bowling and Veloso, 2002]), and many domain-specific problems (such as equilibrium selection).

Stochastic game (a.k.a. Markov game) [Littman, 1994; Hu and Wellman, 2003] is commonly used to formulate a multi-agent system due to its simplicity and generality. In this paper, we focus on learning in fully cooperative stochastic games where all agents receive the same reward signal from the environment. As distinguished by Claus and Boutilier, joint action learner (JAL) and independent learner (IL) are two major learning paradigms of multi-agent reinforcement learning [Claus and Boutilier, 1998]. In cooperative settings, the main difference between the two paradigms is that a JAL agent can observe the other agents' actions whereas an IL agent cannot. Although JAL agents can utilize more information, combinatorial explosion of joint action space and strong observability assumption [Matignon *et al.*, 2012] make them less scalable than IL agents in practice.

In this paper, we study the independent-learner form of MARL for fully cooperative stochastic games. Early work in this domain can be dated back to the average-based methods such as decentralized Q-learning [Tan, 1993] and the optimistic (or maximum-based) methods such as distributed Q-learning [Lauer and Riedmiller, 2000]. However, both the two categories of methods have limited success due to problems such as relative overgeneralization [Wiegand, 2003], miscoordination [Claus and Boutilier, 1998] and stochasticity. For better handling these problems, later works choose to combine the two categories of methods in different ways, such as using different learning rates [Matignon *et al.*, 2007] and applying decreasing leniency [Panait *et al.*, 2006; Wei and Luke, 2016]. Recently, the combination of deep reinforcement learning (DRL) [Mnih *et al.*, 2015] and existing algorithms have also been investigated [Omidshafiei *et al.*, 2017; Palmer *et al.*, 2018].

To summarize, the key to the success of the state-of-the-art algorithms is the implicit coordination facilitated by different fusion heuristics of optimistic and average rewards. However, achieving excellent performance in practice by such implicit coordination mechanisms is a non-trivial task which requires deep understanding, careful implementation and tedious parameter-tuning of these algorithms. Without an explicit coordination mechanism which actively guides agents, the likelihood of coordination may still be low in problems with multiple optimal policies. In this paper, we propose an algorithm called ***ex**plicitly **c**oordinat**e**d po**l**icy iteration* (EXCEL) which constantly forces agents to coordinate during the learning process. The idea is to always let agents try potentially optimal actions indicated by their separated optimistic and average value functions. Our contributions are as follows.

- Based on an agent's optimistic and average functions, we identify the necessary conditions of an optimal policy, according to which we give the action selection strategy and propose our cooperative IL algorithm.

- We propose three solutions for fitting optimistic value functions by deep neural networks and make extensions of our algorithm to the DRL domain.
- Results from extensive experiments in tabular and function approximation settings show that our algorithm outperforms current state-of-the-art algorithms in many aspects (e.g., higher probability of coordination, faster convergence and better asymptotic performance).

The rest of this paper is organized as follows. Section 2 gives background and reviews related work of this paper. Section 3 provides the details of our algorithm. Section 4 shows experimental results and Section 5 finally makes conclusions.

2 Background

In this section, we review key concepts of multi-agent reinforcement learning and briefly introduce related work.

2.1 Multi-Agent Reinforcement Learning

Stochastic games are widely adopted as the model of multi-agent reinforcement learning (MARL) [Littman, 1994; Greenwald and Hall, 2003; Hu and Wellman, 2003]. It can be treated as an extension of MDP to the multi-agent domain.

Definition 1. *An n-agent ($n \geq 2$) stochastic game is a tuple $\langle N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \{\mathcal{R}_i\}_{i=1}^n, \mathcal{P}\rangle$, where N is the set of agents, $\mathcal{S}$ is the state space, $\mathcal{A}_i$ is the action space of agent i ($i = 1, \ldots, n$). Let $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ be the joint action space. $\mathcal{R}_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the reward function of agent i and $\mathcal{P} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0, 1]$ is the transition function.*

In a stochastic game, the accumulative discounted reward of each agent is determined by the joint policy of all agents. We denote the policy of an agent i ($i \in N$) by $\pi_i : \mathcal{S} \times \mathcal{A}_i \to [0, 1]$ and the joint policy of all agents by $\boldsymbol{\pi} = (\pi_1, ..., \pi_n)$. From agent i's perspective, $\boldsymbol{\pi}$ can be also written as (π_i, π_{-i}), where π_{-i} is the joint policy of all other agents. Specially, a fully cooperative stochastic game is a tuple $\langle N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \mathcal{R}, \mathcal{P}\rangle$ where all agents share the same reward function $\mathcal{R}$.

The majority of MARL algorithms for stochastic games can be organized as joint action learner (JAL) and independent learner (IL). As indicated by the names, JAL agents can perceive other agents' actions (and sometimes rewards as well) and IL agents only make decisions from their local viewpoints. As a consequence, the former can directly attribute each reward signal to the corresponding joint action of all agents while the latter cannot distinguish between different reward signals with the same own action taken. Formally, cooperative JAL agents try to maximize the joint action value function $Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}}\Big\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\Big\}$, where $(s, \boldsymbol{a})$ is a state-joint-action pair, γ is the discount rate, t is any time step, and r_{t+k} is the reward at time step $(t + k)$. As IL agents only have local action observation, their value functions are often defined as a projection from the joint value function space to the local value function space $q_i(s, a_i) = \Psi_i Q(s, \boldsymbol{a})$, where q_i is the local value function of agent i, (s, a_i) is a state-local-action pair, Ψ_i denotes any projection for agent i, and $\boldsymbol{a}$ is a joint action containing a_i.

2.2 Related Work

The earliest attempt of IL algorithm in multi-agent systems is decentralized Q-learning [Tan, 1993] which uses an *average projection* to construct each agent's local value function: $q_i(s, a_i) = \sum_{a_{-i}} \pi_{-i}(a_{-i}) Q^{\boldsymbol{\pi}}(s, a_i, a_{-i})$. The relative overgeneralization problem caused by the average projection and the lack of coordination limit the availability of decentralized Q-learning. Distributed Q-learning [Lauer and Riedmiller, 2000] uses an *optimistic projection* $q_i(s, a_i) = \max_{a_{-i}} Q^*(s, a_i, a_{-i})$ to address the relative overgeneralization problem and explicitly coordinates agents by a policy locking mechanism. Despite of the theoretical guarantee in deterministic environments, being highly optimistic makes distributed Q-learning vulnerable to stochasticity. For better handling problems such as relative overgeneralization, miscoordination, and stochasticity, later works apply optimism in a more cautious way. Representative methods include hysteretic Q-learning (HYQ) which uses two different learning rates to perform Q-value update with positive and negative TD-errors respectively[Matignon *et al.*, 2007], the FMQ algorithm which adopts Boltzmann action selection based on the combination of average and optimistic rewards [Kapetanakis and Kudenko, 2002], and lenient learner which shifts gradually from an optimistic learner to an average learner by applying decreasing leniency [Panait *et al.*, 2006; Wei and Luke, 2016]. With the development of deep MARL [Lowe *et al.*, 2017; Foerster *et al.*, 2018], there have been attempts which extend existing algorithms with deep learning models [Omidshafiei *et al.*, 2017; Palmer *et al.*, 2018].

3 Methodology

This section introduces the methodology of this work. We begin with our motivation and then provide algorithm details.

3.1 Motivation

Generally, most of the state-of-the-art algorithms [Kapetanakis and Kudenko, 2002; Matignon *et al.*, 2007; Matignon *et al.*, 2008; Wei and Luke, 2016] can be treated as the combination of optimistic methods and average-based methods. On one hand, learning optimistic values (or utility) can filter penalty resulted from other agents' exploratory behaviors and distinguish between optimal and suboptimal actions. On the other hand, learning average values can deal with stochastic rewards and transitions. By applying simple or complex combination heuristics (e.g., decreasing leniency), coordination is achieved implicitly. However, achieving such implicit coordination in practice seems complicated and difficult. As shown in Figure 1, one can imagine that the learning process is a line segment where the implicit coordination between agents is achieved at some point. But where exactly the point is highly depends on algorithm hyperparameters and cannot be ensured before coordination really occurs. As a result, coordinating agents within predefined learning rounds often requires deep understanding, careful implementation and tedious parameter-tuning of

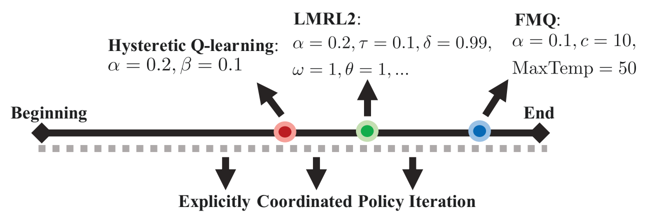

LMRL2:
$\alpha = 0.2, \tau = 0.1, \delta = 0.99,$
$\omega = 1, \theta = 1, ...$
FMQ:
$\alpha = 0.1, c = 10,$
$\mathrm{MaxTemp} = 50$
Hysteretic Q-learning:
$\alpha = 0.2, \beta = 0.1$
Beginning
End
Explicitly Coordinated Policy Iteration

Figure 1: The illustration of different IL algorithms. The solid line is the learning process. The red, green, and blue circles are the implicit coordination points of HYQ, LMRL2, and FMQ, which highly depend on the hyperparameters. The gray dotted line is our algorithm which explicitly and constantly coordinate agents.

the algorithms. Moreover, without an explicit coordination mechanism which tells agents the way of improving policies, the likelihood of coordination may still be low if there are multiple optimal policies in a problem.

3.2 Algorithm

Instead of combining each agent's optimistic and average value functions into one to achieve implicit coordination, we propose to maintain the two value functions separately and explicitly coordinate agents by comparing the two value functions. For a given cooperative stochastic game $\langle N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \mathcal{R}, \mathcal{P}\rangle$, we define the optimistic value function of any agent i ($i \in N$) as

$$q_i^{\text{opt}}(s, a_i) = \max_{a_{-i}} Q^*(s, a_i, a_{-i}) \tag{1}$$

and the average value function of agent i as

$$q_i^{\boldsymbol{\pi}}(s, a_i) = \sum_{a_{-i}} \pi_{-i}(a_{-i}) Q^{\boldsymbol{\pi}}(s, a_i, a_{-i}), \tag{2}$$

where $s \in \mathcal{S}$ is a state, $a_i \in \mathcal{A}_i$ is a local action of agent i, Q^* and $Q^{\boldsymbol{\pi}}$ are the underlying joint action value functions of an optimal joint policy and a given joint policy $\boldsymbol{\pi} = (\pi_1, ..., \pi_n)$, a_{-i} and π_{-i} are the joint action and joint policy of the other agents. For any policy π_i of agent i and any state s, we define $\mathcal{A}_i^{\pi_i(s)} = \{a_i | \pi(s, a_i) > 0\}$ to be the support of $\pi_i(s)$. Formally, an optimal joint policy can be defined as follows.

Definition 2. *A joint policy $\boldsymbol{\pi}$ of a cooperative stochastic game $\langle N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \mathcal{R}, \mathcal{P}\rangle$ is optimal if and only if for any agent $i \in N$ and any state $s \in \mathcal{S}$, it holds that*

(1) $q_i^{\boldsymbol{\pi}}(s, a_i) = \max_{a_i'} q_i^{\boldsymbol{\pi}}(s, a_i'), \forall a_i \in \mathcal{A}_i^{\pi_i(s)}$,

(2) $q_i^{opt}(s, a_i) = \max_{a_i'} q_i^{opt}(s, a_i'), \forall a_i \in \mathcal{A}_i^{\pi_i(s)}$.

The first equation indicates that $\boldsymbol{\pi}$ is a Nash equilibrium and the second one indicates that this Nash equilibrium is optimal. Therefore, from each agent i's individual point of view, the two equations can be used as a necessary condition for determining whether the joint policy $\boldsymbol{\pi}$ is optimal.

Action Selection

For any policy π_i of agent i, define $\pi_i^{s:a_i}$ to be a policy identical to π_i except that agent i always takes action a_i in state s. We utilize a corollary of Definition 2 to guide the action selection of each agent.

Corollary 1. *For any optimal joint policy $\boldsymbol{\pi}$ of a fully cooperative stochastic game $\langle N, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, \mathcal{R}, \mathcal{P}\rangle$, any agent $i \in N$, any state $s \in \mathcal{S}$, and any action $a_i \in \mathcal{A}_i^{\pi_i(s)}$, the joint policy $(\pi_i^{s:a_i}, \pi_{-i})$ is also optimal.*

This corollary provides a way of policy improvement according to each agent's optimistic and average value functions: choosing the actions with maximal optimistic and average values since they could be a part of an optimal policy. Also, Corollary 1 indicates that there is no need for each agent i to reconstruct the underlying policy π_i from its value functions and sample actions accordingly, because at least one pure-strategy optimal policy can be derived from any mixed-strategy optimal policy.

For an agent i, we define the set of actions with maximal optimistic values in state s as

$$\mathcal{M}_{i,s}^{\text{opt}} = \Big\{a_i | q_i^{\text{opt}}(s, a_i) = \max_{a_i'} q_i^{\text{opt}}(s, a_i')\Big\}. \tag{3}$$

Denote our algorithm by $\mathbb{A}$ and the average value function maintained by $\mathbb{A}$ for agent i by $q_i^{\mathbb{A}}$. Similarly, we define the set of actions with maximal average values in state s as

$$\mathcal{M}_{i,s}^{\mathbb{A}} = \Big\{a_i | q_i^{\mathbb{A}}(s, a_i) = \max_{a_i'} q_i^{\mathbb{A}}(s, a_i')\Big\}. \tag{4}$$

According to our discussion, agent i can firstly choose one action from the intersection $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}}$. However, $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}}$ may be empty during the learning process, which indicates that it is impossible to derive an optimal policy from the current value functions. In this case, we suggest choosing the action with maximal average value from the set $\mathcal{M}_{i,s}^{\text{opt}}$ since $\mathcal{M}_{i,s}^{\text{opt}}$ must contain all optimal actions in state s if q_i^{opt} is well learnt. In summary, the principle is to increase the likelihood of finding an optimal policy by always choosing the action with the highest potential to be optimal.

Learning Value Functions

The average value function $q_i^{\mathbb{A}}$ of agent i can be updated directly by a Q-learning rule [Watkins and Dayan, 1992]. Upon receiving an experience (s, a_i, r, s'), $q_i^{\mathbb{A}}$ is updated by

$$q_i^{\mathbb{A}}(s, a_i) \leftarrow (1-\alpha) q_i^{\mathbb{A}}(s, a_i) + \alpha(r + \gamma \max_{a_i'} q_i^{\mathbb{A}}(s', a_i')), \tag{5}$$

where α is a learning rate. The way of updating the optimistic value function q_i^{opt} is similar to that of distribute Q-learning, which changes the value $q_i^{\text{opt}}(s, a_i)$ for state-action pair (s, a_i) only when an increase occurs:

$$\begin{aligned} q_i^{\text{opt}}(s, a_i) \leftarrow & (1-\alpha) q_i^{\text{opt}}(s, a_i) + \alpha \max \Big\{q_i^{\text{opt}}(s, a_i), \\ & r + \gamma \max_{a_i'} q_i^{\text{opt}}(s', a_i')\Big\}. \end{aligned} \tag{6}$$

Taking stochastic rewards and state transitions into account, a learning rate α is also introduced here. In contrast, distributed Q-learning has no learning rate and is only applicable in deterministic environments. However, Equation (6) cannot deal with stochasticity alone since $q_i^{\text{opt}}(s, a_i)$ always increases when a maximal target (i.e., $r + \gamma \max_{a_i'} q_i^{\text{opt}}(s', a_i')$)

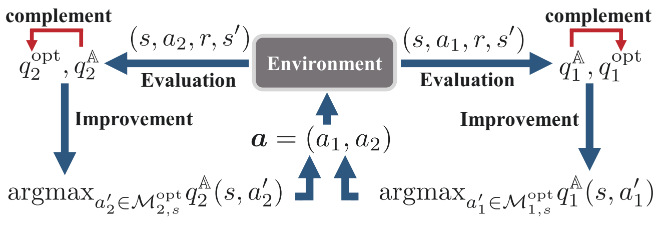

complement
$q_2^{\text{opt}}, q_2^{\mathbb{A}}$
(s, a_2, r, s')
Evaluation
Environment
(s, a_1, r, s')
Evaluation
complement
$q_1^{\mathbb{A}}, q_1^{\text{opt}}$
Improvement
$\boldsymbol{a} = (a_1, a_2)$
Improvement
$\text{argmax}_{a_2' \in \mathcal{M}_{2,s}^{\text{opt}}} q_2^{\mathbb{A}}(s, a_2')$
$\text{argmax}_{a_1' \in \mathcal{M}_{1,s}^{\text{opt}}} q_1^{\mathbb{A}}(s, a_1')$

Figure 2: The main flow of EXCEL in a 2-agent problem

is sampled and will finally converge to its upper bound rather than the maximal expected value. To address this issue, note that for an optimal joint policy $\boldsymbol{\pi}$, it must hold that

$$q_i^{\boldsymbol{\pi}}(s, a_i) = q_i^{\text{opt}}(s, a_i), \forall i, \forall s, \forall a_i \in \mathcal{A}_i^{\pi_i(s)}. \qquad (7)$$

This motivates the idea that using the average value $q_i^{\mathbb{A}}(s, a_i)$ to complement the optimistic value $q_i^{\text{opt}}(s, a_i)$. If the intersection $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}}$ is empty, which indicates that suboptimal actions might be overestimated due to their higher upper bounds of optimistic values, then the optimistic values of the actions in $\mathcal{M}_{i,s}^{\text{opt}}$ will be reduced by

$$q_i^{\text{opt}}(s, a_i) \leftarrow (1 - c)q_i^{\text{opt}}(s, a_i) + cq_i^{\mathbb{A}}(s, a_i), \forall a_i \in \mathcal{M}_{i,s}^{\text{opt}},$$

where c is a complementary factor that is very small in the beginning and increases every time $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}} = \emptyset$ occurs. The reason that c should increase from a small value is that $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}} = \emptyset$ may also happen before the value functions are sufficiently evaluated. Thus, c can be also treated as a measure of environment stochasticity. If $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}} \neq \emptyset$, the value complement will be conducted on all actions. On one hand, this would not change the agent's behaviors if the actions in the intersection really belong to an optimal policy. On the other hand, the overestimation problem can be alleviated if the environment is stochastic.

Explicitly Coordinated Policy Iteration

The rules for action selection and updating value functions together form a typical policy iteration loop. We name our method ***explicitly coordinated policy iteration*** (EXCEL) since it explicitly forces agents to coordinate in every state. Figure 2 intuitively depicts the main flow of EXCEL in a 2-agent problem. The details of EXCEL are shown in Algorithm 1 from agent i's point of view. To compute the greedy action a_i^* (at line 7), there is no need to compute the set $\mathcal{M}_{i,s}^{\mathbb{A}}$ since a_i^* must be in $\mathcal{M}_{i,s}^{\mathbb{A}}$ if $\mathcal{M}_{i,s}^{\text{opt}} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}} \neq \emptyset$. The value complement mechanism (lines 15 to 23) is similar to the recursive frequency maximum Q-value (RFMQ) heuristic [Matignon *et al.*, 2012] since they both have the form of linear interpolation and both their learning steps are a measure of stochasticity. However, the two mechanisms are essentially different because the former is a way of overcoming the overestimation of optimistic values based on the property of an optimal policy (i.e., Eq. (7)) while the latter is a way of biasing the evaluation of actions with optimistic rewards.

Algorithm 1: Explicitly Coordinated Policy Iteration

```
   Input: Learning rate $\alpha$, discount rate $\gamma$, exploration factor $\epsilon$,
          maximal complementary factor $c_{\max}$, increment of
          complementary factor $\Delta c$
1  $\forall s \in S, \forall a_i \in \mathcal{A}_i, q_i^{\text{opt}}(s, a_i) \leftarrow 0, q_i^{\mathbb{A}}(s, a_i) \leftarrow 0$;
2  foreach episode do
3      Initialize state $s$;
4      $c \leftarrow 0$;
5      repeat
6          $\mathcal{M}_{i,s}^{\text{opt}} \leftarrow \left\{\hat{a}_i | q_i^{\text{opt}}(s, \hat{a}_i) = \max_{a_i'} q_i^{\text{opt}}(s, a_i')\right\}$;
7          $a_i^* \leftarrow \arg\max_{a_i' \in \mathcal{M}_{i,s}^{\text{opt}}} q_i^{\mathbb{A}}(s, a_i')$;
8          $a_i \leftarrow \epsilon$-greedy$(a_i^*)$;
9          Receive the experience $(s, a_i, r, s')$;
10         $\delta^{\mathbb{A}} \leftarrow r + \gamma \max_{a_i'} q_i^{\mathbb{A}}(s', a_i') - q_i^{\mathbb{A}}(s, a_i)$;
11         $q_i^{\mathbb{A}}(s, a_i) \leftarrow q_i^{\mathbb{A}}(s, a_i) + \alpha\delta^{\mathbb{A}}$;
12         $\delta^{\text{opt}} \leftarrow r + \gamma \max_{a_i'} q_i^{\text{opt}}(s, a_i') - q_i^{\text{opt}}(s, a_i)$;
13         if $\delta^{opt} > 0$ then
14             $q_i^{\text{opt}}(s, a_i) \leftarrow q_i^{\text{opt}}(s, a_i) + \alpha\delta^{\text{opt}}$;
15         $\mathcal{M}_{i,s}^{\mathbb{A}} \leftarrow \left\{\hat{a}_i | q_i^{\mathbb{A}}(s, \hat{a}_i) = \max_{a_i'} q_i^{\mathbb{A}}(s, a_i')\right\}$;
16         Recompute $\mathcal{M}_{i,s}^{\text{opt}}$;
17         if $\mathcal{M}_{i,s}^{opt} \bigcap \mathcal{M}_{i,s}^{\mathbb{A}} = \emptyset$ then
18             $c \leftarrow \min\{c + \Delta c, c_{\max}\}$;
19             foreach $\hat{a}_i \in \mathcal{M}_{i,s}^{opt}$ do
20                 $q_i^{\text{opt}}(s, \hat{a}_i) \leftarrow (1-c)q_i^{\text{opt}}(s, \hat{a}_i) + cq_i^{\mathbb{A}}(s, \hat{a}_i)$;
21         else
22             foreach $\hat{a}_i \in \mathcal{A}_i$ do
23                 $q_i^{\text{opt}}(s, \hat{a}_i) \leftarrow (1-c)q_i^{\text{opt}}(s, \hat{a}_i) + cq_i^{\mathbb{A}}(s, \hat{a}_i)$;
24         $s \leftarrow s'$;
25     until $s$ is a terminal state;
```

3.3 DRL Extensions

For applications in high-dimensional state space problems, we also extend our algorithm to the deep reinforcement learning domain. We adopt the same framework of deep Q-learning network (DQN) [Mnih *et al.*, 2015] for learning the agents' optimistic and average value functions. The learning tricks such as experience replay and target networks are also utilized. However, we found that using only samples with positive TD-errors to update the optimistic value functions may cause the overestimation problem even in deterministic environments. With neural networks as the value function representation, samples with positive TD-errors may result from the output values of the network rather than the true rewards. Before the network is well fitted, such unreliable samples may be generated frequently. Therefore, for learning accurate optimistic values, the samples with negative TD-errors may be also useful. Given agent i's optimistic value function q_i^{opt} and a sample (s, a_i, r, s'), let $\delta = r + \gamma q_i^{\text{opt}}(s, a_i) - \max_{a_i'} q_i^{\text{opt}}(s', a_i')$ be the original TD-error. Let $\mathbb{1}_p$ be the function which is 1 when p is true and 0 when p is false. For utilizing (s, a_i, r, s') in gradient computation when $\delta < 0$, we propose three methods which compute a corrected TD-error δ^{opt} based on δ:

(1) *negative threshold* method which makes the TD-error 0 if δ is less than a negative threshold: $\delta^{\text{opt}} = \mathbb{1}_{\delta \geq \delta_{\min}} \delta$, where $\delta_{\min} < 0$ and $|\delta_{\min}|$ will decay to a small value,

(2) *weighted negative TD-error* method which multiplies negative TD-errors by a weight: $\delta^{\text{opt}} = \max\{\mathbb{1}_{\delta>0}, \beta\}\delta$, where β decays from 1 to a small value,

(3) *negative sampling* method which samples negative TD-errors by a ratio: $\delta^{\text{opt}} = \max\{\mathbb{1}_{u>\eta}\delta, \delta\}$, where u is a random variable uniformly distributed in $[0, 1]$ and η is a sampling ratio decaying from 1 to a small value.

As learning progresses, the three methods gradually reduce the use of negative samples. This is different from previous algorithms which make use of negative samples by a constant negative learning rate [Matignon *et al.*, 2007; Omidshafiei *et al.*, 2017] and an increasing sampling ratio [Wei and Luke, 2016; Palmer *et al.*, 2018]. Besides, the three methods indirectly implements value complement as the utilization of negative samples also means the injection of average rewards into the optimistic value functions. Thus, the linear interpolation of Algorithm 1 don't have to be included in the extension. The other components of Algorithm 1 can be extended straightforwardly so we omit the details.

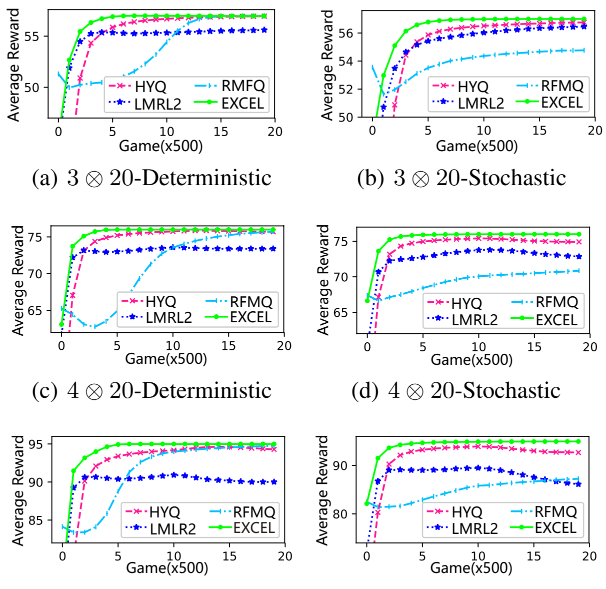
4 Experiments

We conduct two groups of experiments in this paper. The first one is a test in matrix games and the second one is grid world games in both tabular and function approximation settings.

4.1 Matrix Games

Denote the matrix game with n agents and m actions for each agent by $n \otimes m$. In our experiment, n ranges from 2 to 5 and m ranges from 2 to 20. Let a_i be an action of an agent i and $\mathbb{I}(a_i)$ be the index of a_i $(0 \sim m-1)$. The utility $r(\boldsymbol{a})$ of a joint action $\boldsymbol{a} = (a_1, ..., a_n)$ is $n \times (m-1)$ if the indices of all agents' actions are the same. Otherwise, $r(\boldsymbol{a}) = \max\left\{\sum_{i=1}^{n} \mathbb{I}(a_i), n(m-1) - \sum_{i=1}^{n} \mathbb{I}(a_i)\right\}$.

We implement hysteretic Q-learning (HYQ) [Matignon *et al.*, 2007], lenient multi-agent reinforcement learning 2 (LMRL2) [Wei and Luke, 2016], and recursive frequency maximum Q-value (RFMQ) [Matignon *et al.*, 2012] to compare with our algorithm EXCEL. Each test of an algorithm in a specific game contains $20,000$ training periods and each training period contains $10,000$ game plays. We test the performance of each algorithm in both deterministic and stochastic reward settings. Let r denote the original utility of a joint action. In the stochastic reward problem, the utility of a non-optimal joint action is sampled from a Gaussian distribution with mean value r and 0.95 confidence interval $[0.8r, 1.2r]$ while the utility of an optimal joint action is from a Gaussian distribution with mean value r and 0.9 confidence interval $[0.7r, 1.3r]$. Therefore, optimal joint actions have higher utility variances than non-optimal joint actions. All the algorithms except HYQ use a learning rate of 0.2. The positive and negative learning rates of HYQ are 0.05 and 0.02, respectively. The learning rate of frequency of RFMQ is 0.01. The complementary factor of EXCEL increases from 0 to 1 with an increment 0.001. The algorithms EXCEL, HYQ, and RFMQ adopt ϵ-greedy exploration with ϵ decaying from 1.0 to 0.1 exponentially by a factor 0.99977. LMRL2 uses Boltzmann exploration with the temperature of each action decaying from 50 to 0.1 by a factor 0.9. The moderation factors of action selection and lenience for LMRL2 are both 1.0.



(a) $3 \otimes 20$-Deterministic (b) $3 \otimes 20$-Stochastic

(c) $4 \otimes 20$-Deterministic (d) $4 \otimes 20$-Stochastic

(e) $5 \otimes 20$-Deterministic (f) $5 \otimes 20$-Stochastic

Figure 3: Average rewards of each algorithm in matrix games

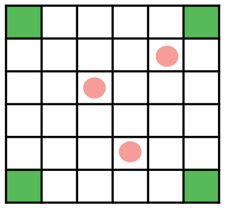
	$3 \otimes 20$-(D/S)	$4 \otimes 20$-(D/S)	$5 \otimes 20$-(D/S)
EXCEL	**0.999/0.999**	**0.995/0.993**	**0.975/0.983**
HYQ	0.960/0.793	0.766/0.440	0.547/0.205
RFMQ	0.999/0.540	0.953/0.255	0.971/0.177
LMRL2	0.353/0.811	0.198/0.497	0.095/0.182

Table 1: The coordination ratio of each algorithm in matrix games

Each game play in a training period is followed by a test game play with deterministic rewards and no exploration. Along a training period, we record the reward of each algorithm in each test game play. We also record how many times that an optimal policy is learnt among the $20,000$ rounds of training, namely the coordination ratio. Due to space limitation, we only show the results of the most difficult games $3 \otimes 20$, $4 \otimes 20$, and $5 \otimes 20$ in Figure 3 and Table 1. Figure 3 shows that EXCEL converges faster and finally achieves higher reward values than the other algorithms. The performance gap between the algorithms is more obvious with respect to the coordination ratio. As shown in Table 1, EXCEL always succeeds to coordinate agents (with at least 0.97 coordination ratio) while the coordination ratios of the other algorithms decay rapidly as the number of agents increases.

4.2 Grid World Games

We use the grid world games shown in Figure 4 to test the performance of our algorithm in stochastic games. The left one in Figure 4 is a grid map for testing tabular algorithms and the right one is a modified predator-prey (PP) game [Lowe *et al.*, 2017] for testing DRL algorithms. The agents (the red circles) are initially located in the central area and are required to reach the same goal among the 4 goals (the green areas).


(a) Tabular Grid World

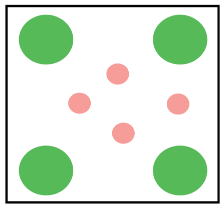
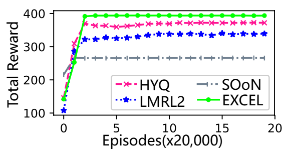

(b) Predator-Prey Game

Figure 4: The grid world games adopted in the second experiment

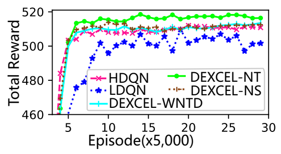
In each step, the agents can choose to move in one of the four directions *left*, *right*, *up*, and *down* or *stay*. Once an agent reaches any of the four goals, the team reward will be added by 10. If all agents reach the same goal, the team reward will be directly set to $15n$, where n is the number of agents.

In the tabular grid world games, we implement the EXCEL, HYQ, LMRL2 algorithms and the swing between optimistic or neutral algorithm (SOoN) [Matignon *et al.*, 2009]. The number of agents ranges from 2 to 4 and the corresponding map sizes are 10×10, 6×6, and 5×5, respectively. One episode of each game contains 25 steps of state transitions. We make stochastic state transitions by assigning a 0.2 failure probability to the execution of actions. The team reward is also made stochastic by sampling from a Gaussian distribution with mean value r and 0.8 confidence interval $[0.5r, 1.5r]$, where r is the original reward. The learning rates of all algorithms are 0.2. The negative learning rate of HYQ is 0.08. The learning rate of the farsighted frequency of SOoN is 0.3. The EXCEL, HYQ, and SOoN algorithms use ϵ-greedy exploration strategy. In the 2-agent and 3-agent games, ϵ decays from 1 to 0.05 while in the 4-agent game it decays to 0.25 for ensuring sufficient exploration. The decay factor of ϵ is 0.9999 and the discount rate is 0.9. The other hyperparameters are the same as those in the matrix game experiment. We record the total rewards of the agent team in each episode.

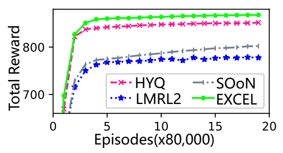
In the modified predator-prey (PP) game, we implement our deep EXCEL algorithm (DEXCEL) with negative threshold (NT), weighted negative TD-error (WNTD), and negative sampling (NS), respectively. We also implement the hysteretic deep Q-network (HDQN) [Omidshafiei *et al.*, 2017] and the lenient deep Q-network (LDQN) [Palmer *et al.*, 2018] for comparison. All these algorithms adopt the same Q-network structure of two fully connected hidden layers with 64 units and *relu* activation functions. The network parameters are optimized by Adam with a learning rate of 10^{-3} and a batch size of 1024. The threshold $\delta_{\min}$ of the NT method increases from -20 to 0 in $10,000$ episodes. The weight coefficient β of the WNTD method and the sampling ratio η of the NS method both decay from 1.0 to 10^{-4} in $100,000$ episodes. The coefficient for negative update of HDQN is 0.4. LDQN uses retroactive temperature decay schedule and the corresponding hyperparameters are $\rho = -0.01$, $d = 0.95$, $\mu = 0.9995$, $v = 1$. For exploration, all the algorithms uses ϵ-greedy strategy with ϵ decaying from 1 to 0.2 in $60,000$ episodes. We range the number of agents from 3 to 5 and test the total rewards of each algorithm in each episode.



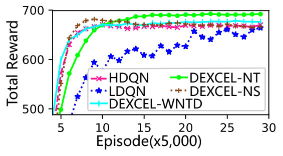
(a) 2-Agent Tabular Game



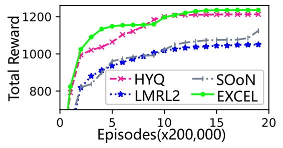
(b) 3-Agent PP Game



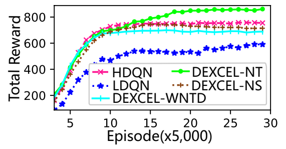
(c) 3-Agent Tabular Game



(d) 4-Agent PP Game



(e) 4-Agent Tabular Game



(f) 5-Agent PP Game

Figure 5: Average rewards of each algorithm in matrix games

Figure 5 contains the results of the tabular grid world and the modified PP game. From the left column, it can be found that our algorithm EXCEL again performs the best among all the tested algorithms. The reward value that EXCEL converges to in each of the three tabular games is very close to the corresponding optimal value. The right column of Figure 5 shows that the negative threshold is the most effective method for utilizing negative samples because DEXCEL-NT significantly outperforms the other algorithms. DEXCEL-WNTD and DEXCEL-NS also perform better than (or similar to) HDQN and LDQN in the 3-agent and 4-agent PP games. However, the two algorithms fail to outperform HDQN in the 5-agent PP game. But generally, the results demonstrate that EXCEL has the ability to coordinate agents well.

5 Conclusion

In this paper, we propose an independent learner algorithm called explicitly coordinated policy iteration (EXCEL) for fully cooperative stochastic games. Instead of using an implicit coordination mechanism, the EXCEL algorithm explicitly forces agents to coordinate by utilizing the necessary conditions derived from the optimistic and average value functions of an optimal policy. For applications in large-scale problems, we propose three versions of deep EXCEL algorithms which utilize negative samples differently. We conduct extensive experiments in matrix games and stochastic games. The empirical results show that EXCEL converges faster, achieves higher coordination ratio and better asymptotic performance than the state-of-the-art algorithms.

Acknowledgements

We thank the reviewers for the valuable comments. Jianye Hao is supported by the National Natural Science Foundation of China (Grant Nos.: 61702362, U1836214).

References

[Bowling and Veloso, 2002] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[Buşoniu *et al.*, 2008] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, 2008.

[Claus and Boutilier, 1998] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 746–752, 1998.

[Foerster *et al.*, 2018] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18)*, pages 2974–2982, 2018.

[Greenwald and Hall, 2003] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 242–249, 2003.

[Hu and Wellman, 2003] Junling Hu and Michael P. Wellman. Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.

[Kapetanakis and Kudenko, 2002] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02)*, pages 326–331. York, 2002.

[Lauer and Riedmiller, 2000] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the 17th International Conference on Machine Learning*, pages 535–542, 2000.

[Littman, 1994] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[Matignon *et al.*, 2007] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'07)*, pages 64–69, 2007.

[Matignon *et al.*, 2008] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. A study of FMQ heuristic in cooperative multi-agent games. In *Proceedings of Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domains Workshop in the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 77–91, 2008.

[Matignon *et al.*, 2009] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Coordination of independent learners in cooperative markov games. Technical report, Institut FEMTO-ST, UniversitPé de Franche-Comté, 2009.

[Matignon *et al.*, 2012] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[Omidshafiei *et al.*, 2017] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning Under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, pages 2681–2690, 2017.

[Palmer *et al.*, 2018] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'18)*, pages 443–451, 2018.

[Panait *et al.*, 2006] Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In *Proceedings of the 15th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*, pages 801–803, 2006.

[Shoham *et al.*, 2003] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. *Unpublished survey. http://robotics. stanford. edu/shoham*, 2003.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, volume 337, pages 330–337, 1993.

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

[Wei and Luke, 2016] Ermo Wei and Sean Luke. Lenient learning in independent-learner stochastic cooperative games. *The Journal of Machine Learning Research*, 17(1):2914–2955, 2016.

[Wiegand, 2003] R. Paul Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, 2003.