Breaking Deadlocks in Multi-agent Reinforcement Learning with Sparse Interaction

Toshihiro Kujirai(✉) and Takayoshi Yokota

Tottori University, 4-101, Koyama-cho Minami, Tottori 680-8550, Japan
tkujiraski@gmail.com, yokota@eecs.tottori-u.ac.jp

Abstract. Although multi-agent reinforcement learning (MARL) is a promising method for learning a collaborative action policy that will enable each agent to accomplish specific tasks, the state-action space increased exponentially. Coordinating Q-learning (CQ-learning) effectively reduces the state-action space by having each agent determine when it should consider the states of other agents on the basis of a comparison between the immediate rewards in a single-agent environment and those in a multi-agent environment. One way to improve the performance of CQ-learning is to have agents greedily select actions and switch between Q-value update equations in accordance with the state of each agent in the next step. Although this "GPCQ-learning" usually outperforms CQ-learning, a deadlock can occur if there is no difference in the immediate rewards between a single-agent environment and a multi-agent environment. A method has been developed to break such a deadlock by detecting its occurrence and augmenting the state of a deadlocked agent to include the state of the other agent. Evaluation of the method using pursuit games demonstrated that it improves the performance of GPCQ-learning.

Keywords: Reinforcement learning · Multi-agent · Sparse interaction · Fully cooperative · Deadlock

1 Introduction

Multi-agent reinforcement learning (MARL) is a promising method for learning a collaborative action policy that will enable each agent to accomplish specific tasks (Bloembergen et al. 2015; Vlassis 2007). Each agent tries to learn an optimal action policy, one that maximizes the expected cumulative rewards, while sharing the environment with other agents. Agents that learn their action policy by considering the states and actions of other agents are called joint-action learners. Those that learn it independently are called independent learners (Claus and Boutilier 1998).

If each agent shares the same reward for a task, i.e., a fully cooperative task, independent learners can sometimes learn a collaborative action policy without considering the states and actions of other agents because a random exploration strategy may enable them to learn collaborative actions coincidently (Lauer and Riedmiller 2000; Sen et al. 1994). While joint-action learners may perform better because they take information about other agents into account, they suffer an exponential increase in the state-action space for learning, thereby reducing the learning speed and increasing the cost of communication and the cost of estimating information about other agents (Tan 1993).

In many real-world tasks, agents behave independently most of the time and sometimes must behave cooperatively. For example, consider a task involving multiple robots working together to move a heavy box to a specific position. A rational approach is for them to independently approach the box and then cooperatively move it in the same direction to the final position. Each should decide its actions taking other agents' positions and actions into account only when the robots are close to each other. The basic idea of MARL with sparse interaction is to reduce the state-action space by considering information about other agents only when necessary because a smaller state-action space makes the learning process more efficient. This means that identifying when cooperative actions are required is a key function in MARL with sparse interaction. Melo and Veloso (2009) reported a method in which a pseudo-action, COORDINATE, is added to the action space of each agent. The agents learn when they should consider other agents by estimating the Q-value for the COORDINATE action for each state. Hauwere et al. (2010, 2011) proposed the coordinating Q-learning (CQ-learning) concept. Each agent determines when it should consider the state of other agents by comparing the immediate rewards in a single-agent environment with those in a multi-agent environment. In CQ-learning, the state-action space is partially augmented when an agent detects a difference in the immediate rewards using Student's t-test. Kujirai and Yokota (2018, 2019) reported three methods for improving the performance of CQ-learning: greedily selecting actions (GCQ-learning), switching between Q-value updating equations on the basis of the state of each agent in the next step (PCQ-learning), and their combination (GPCQ-learning). Evaluation using several maze games validated their effectiveness, especially that of GPCQ-learning.

We previously observed that agents using GPCQ-learning sometimes fall into a deadlock if there is no difference in the immediate rewards between a single-agent environment and a multi-agent environment. We have now developed a method for breaking the deadlock by detecting its occurrence and augmenting the state of a deadlocked agent to include the state of the other agent. Evaluation using pursuit games demonstrated that it improves the performance of GPCQ-learning.

The reminder of this paper is organized as follows. Section 2 gives an overview of MARL and discusses related work. Section 3 discusses MARL with sparse interaction and the deadlock caused by GPCQ-learning. Section 4 presents our method for breaking the deadlock. Section 5 describes our evaluation and compares the performance of our proposed method with those of existing methods including GPCQ-learning. Section 6 concludes this paper with a summary of the key points.

2 Multi-agent Reinforcement Learning

2.1 MDP and Reinforcement Learning

A Markov decision process (MDP) is formalized as a problem in which an agent optimizes its action policy by maximizing the expected cumulative reward resulting from the actions it takes in its environment. The MDP is defined as a tuple (S, T, R, π), where S stands for the state space of the agent, $T(= p(s'|s, a))$ and $R(= r(s, a, s'))$ stand for the transition probability matrix and immediate reward matrix for the combinations of state s, action a, and next state s', and $\pi(= p(a|s))$ stands for the action policy of the agent. An optimal policy, i.e., one that maximizes the expected cumulative reward, is described as $\pi*$.

Reinforcement learning is one method for iteratively estimating $\pi*$. Q-learning (Watkins 1989, 1992) is a typical reinforcement learning method. Instead of estimating $\pi*$, Q-learning estimates the optimal Q-value $Q*$ by updating the Q-value using (1), wherein α_t indicates the learning rate and γ indicates the discount rate.

$$\mathrm{Q(s,a)} \leftarrow (1-\alpha_t)\mathrm{Q(s,a)} + \alpha_t[r(s,a) + \gamma max_{a'}\mathrm{Q(s,a)}] \tag{1}$$

2.2 Extended Multi-agent Systems

As shown in Table 1, an MDP can be extended for multi-agent systems in at least four ways. A natural extension is multi-agent MDP (MMDP), in which agents share all the system states (full observability) and rewards. The agents share all their states and actions and obtain the same rewards from the environment as a result of their joint actions (Boutilier 1996).

Table 1. Extended multi-agent systems.

	Full observability	Full joint observability
Shared rewards	MMDP	DEC-MDP
Independent rewards	MG	DEC-MG

Another extension is decentralized MDP (DEC-MDP), in which each agent can observe only its own states, and the agents obtain the same rewards (Melo and Veloso 2011). If they can know the complete state of the environment by sharing their observations, the agents are said to have full joint observability.

These two extensions are called fully cooperative games because the agents obtain the same rewards.

In contrast, in a Markov game (MG) and a decentralized Markov game (DEC-MG), each agent has an independent reward function. This results in a competitive situation (Aras 2004).

2.3 Related Work

The focus here is on fully cooperative games, which have at least one optimal action policy for each agent. The agents can serendipitously learn a cooperative behavior without having any information about the other agents. This is because coincidental actions that lead to high cumulative reward are reinforced (Sen et al. 1994).

For example, assume that an agent randomly selects an action at a certain position in a maze game, and the action results in the agent obtaining a high cumulative reward because the action coincidentally prevents the agent from colliding with another agent. The agent may thereby learn a cooperative action policy without having any information about the other agents. An agent learns a more precise cooperative action policy if it has knowledge of not only its own state-action combinations but also those of other agents. However, the resulting exponential increase in the state-action space and communication cost between agents slows down the learning process.

Figure 1 shows two example maze games in which each agent i tries to find an optimal path from start position S_i to goal G_i. In both games, the goal for each agent is the start position of another agent. They collide if each one simply takes the shortest path. The optimal solution is for one of the agents to take a detour immediately a collision. However, finding this solution requires extensive exploration of potential detours because there are a number of unsuitable detour routes.

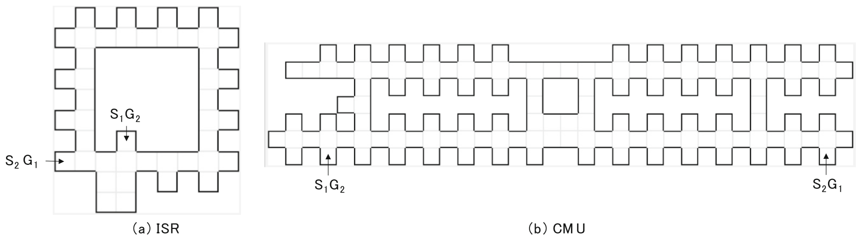


Fig. 1. Example maze games.

Figure 2 shows the average number of steps needed to complete the two games for every 100 episodes using three different learning methods. These methods are straightforward extensions of Q-learning for a multi-agent environment. The first method is independent learning, which is Q-learning itself. Each agent learns its own action policy without having any information about the other agents. The second one is joint-state learning (JSQ-learning), in which each agent always knows the states of the other agents and decides its actions independently on the basis of its own policy. The third is joint-state-action learning (JSAQ-learning), in which one super-agent observes all the states and decides the joint actions for all agents.

As shown in Fig. 2, even the agents using independent learning learned how to avoid collisions and reach their goals unimpeded. In the ISR game, which has a small state-action space, the agents using independent learning converged the fastest although the average number of steps to the goal was the highest because they did not explicitly consider the other agents. In the CMU game, which has a larger state-action space, the agents trained using independent learning had superior performance because 10,000 episodes were not enough for the other methods to learn an optimal policy in the large state-action space.

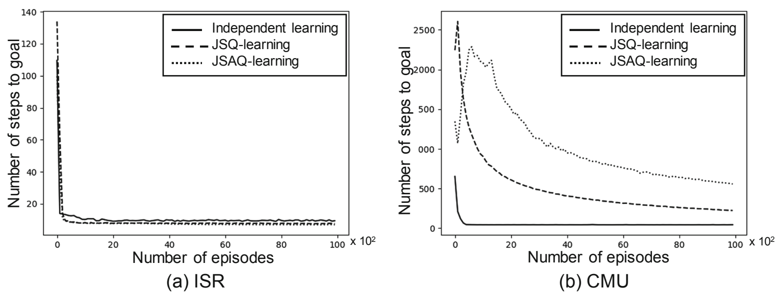


Fig. 2. Learning curves for two maze games.

Policy convergence was slower for the agents trained using JSQ-learning, and the average number of steps to the goal was less than that for the agents trained using independent-learning in the ISR game because these agents consider the other agents when selecting their actions. In the CMU game, the convergence of their policies was slower than that of the agents trained using independent learning because they had a larger state-action space: 43 × 43 × 4 = 7396 in the ISR game and 133 × 133 × 4 = 70,756 in the CMU game for each agent.

The agents trained using JSAQ-learning learned a better policy than the agents trained using the other two methods in the ISR game because the combined state-action space was small enough for each agent to learn an optimal joint-action policy. All the agents had difficulty learning an optimal policy in the CMU game because the combined state-action space was too large (133 × 133 × 4 × 4 = 283,024) for the agents to sufficiently explore all the state-action pairs in the limited number of episodes.

3 Multi-agent Reinforcement Learning with Sparse Interaction

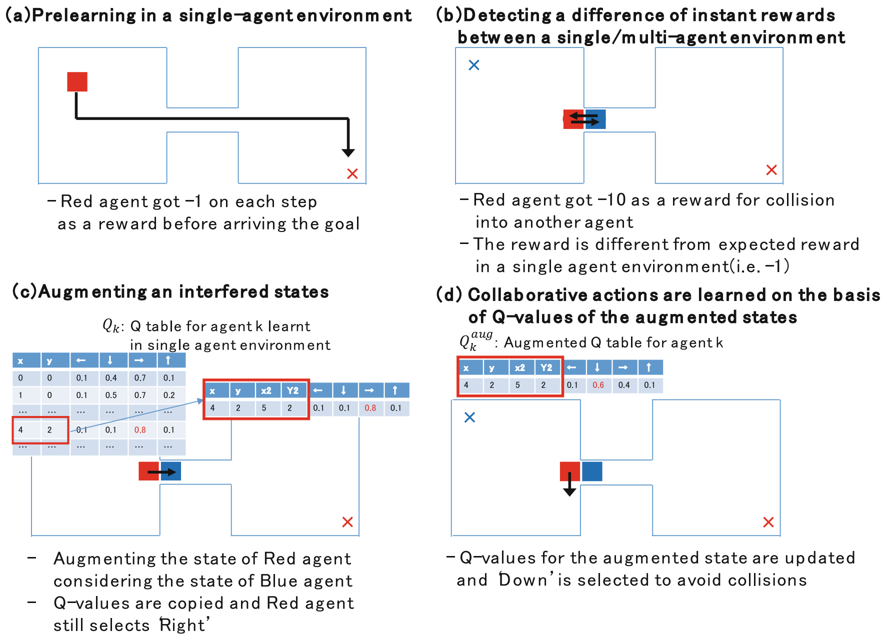
3.1 Existing Methods of MARL with Sparse Interaction

In some fully cooperative games, agents can decide their actions without considering any information about the other agents for most states. That is, an independent optimal action policy might be optimal for most states. In the other states, each agent needs information about the other agents to learn an optimal action policy. Therefore, exponential increases in the state-action space and in the communicational cost can be avoided by having each agent consider information about the other agents only when necessary. This type of framework is called decentralized sparse interaction MDP (DEC-SIMDP) (Melo and Veloso 2011) and is a special case of DEC-MDP. Several methods have been proposed for agents to learn their action policy for DEC-SIMDP.

As mentioned in the introduction, Melo and Veloso (2009) reported a method in which a pseudo-action, COORDINATE, is added to the action space of each agent. When COORDINATE is selected as an action by an agent, the agent obtains information about the other agents and behaves in accordance with that information while suffering the penalty of communication cost. Because the Q-value for selecting

COORDINATE can be obtained with Q-learning, the agent can decide when it should consider the other agents. Setting the cost of COORDINATE for a specific task is a difficult issue because, if the cost is too low, the agents will always choose COORDINATE, and, if the cost is too high, the agents will seldom choose it.

Hauwere et al. (2010, 2011) proposed a method in which the state of an agent is augmented to include the state of another agent if the two agents are likely to interfere with each other. Each agent behaves in accordance with Q-values learned in advance in a single-agent environment. Each agent can identify potential interference with other agents by comparing the distribution of a state's immediate rewards to those for a single-agent environment. Once the state of an agent is augmented, the agent selects an action on the basis of Q-values corresponding to the augmented joint state when the agent and another agent are in an augmented joint state (Fig. 3).


(a)Prelearning in a single-agent environment
- Red agent got -1 on each step as a reward before arriving the goal
(b)Detecting a difference of instant rewards between a single/multi-agent environment
- Red agent got -10 as a reward for collision into another agent
- The reward is different from expected reward in a single agent environment(i.e. -1)
(c)Augmenting an interfered states
Q_k: Q table for agent k learnt in single agent environment
- Augmenting the state of Red agent considering the state of Blue agent
- Q-values are copied and Red agent still selects 'Right'
(d) Collaborative actions are learned on the basis of Q-values of the augmented states
Q_k^{aug}: Augmented Q table for agent k
- Q-values for the augmented state are updated and 'Down' is selected to avoid collisions

x	y	←	↓	→	↑
0	0	0.1	0.4	0.7	0.1
1	0	0.1	0.5	0.7	0.2
…	…	…	…	…	…
4	2	0.1	0.1	0.8	0.1
…	…	…	…	…	…

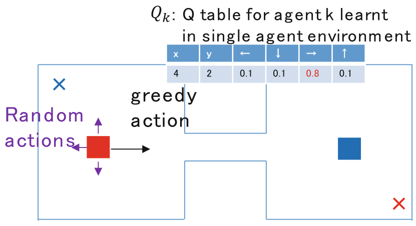
x	y	x2	Y2	←	↓	→	↑
4	2	5	2	0.1	0.1	0.8	0.1

x	y	x2	Y2	←	↓	→	↑
4	2	5	2	0.1	0.6	0.4	0.1

Fig. 3. Illustrated CQ-learning algorithm.

This method is called cooperating Q-learning (CQ-learning). To be more specific, CQ-learning augments the state of agent k s_k, creating augmented joint state $\overrightarrow{s_k} = (s_k, s_l)$ that considers the state of agent l. Its Q-values are represented as $Q_k^{aug}(\overrightarrow{s_k}, a_k)$ when Student's t-test rejects the hypothesis that the distribution of immediate rewards in the state comes from that of a single-agent environment. This partial augmentation of joint states dramatically reduces the state-action space in sparse interaction tasks compared with JSQ-learning and JSAQ-learning and thereby improves the efficiency and optimality of the learned action policy.

Kujirai and Yokota (2018, 2019) pointed out two issues on CQ-learning and proposed an improved method of CQ-learning, called GPCQ-learning. The first issue on CQ-learning is its unnecessary exploration. An agent using CQ-learning selects its action ϵ-greedily even it is not in an interfered state. This causes unnecessary exploration and interferences in a multi-agent environment. In addition to that, taking a random action might coincidently prevent interference with another agent, resulting in a lost opportunity for the agent to identify the difference between a single-agent environment and multi-agent environment (Fig. 4)


Q_k: Q table for agent k learnt
in single agent environment
greedy
action
Random
actions

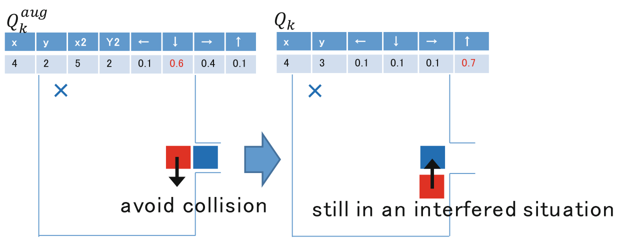
x	y	←	↓	→	↑
4	2	0.1	0.1	0.8	0.1

Fig. 4. Unnecessary exploration by CQ-learning.

The second issue is that CQ-learning optimistically updates the Q-values of an augmented joint state based on the Q-values learned in the single-agent environment as shown in Eq. (2). This updating assumes that after taking the selected action, the agent can behave based on independent Q-values without subsequent interference.

$$Q_k^{aug}(\vec{s_k}, a_k) \leftarrow (1-\alpha_t)Q_k^{aug}(\vec{s_k}, a_k) + \alpha_t[r_k + \gamma max_{a'_k} Q_k(s'_k, a'_k)] \tag{2}$$

This assumption is too optimistic because when an agent is in an interference states with another agent the probability of being in another interference states for the agent can not be neglectable. In Fig. 5 a red agent avoid collision by selecting its action based on its augmented Q-values. CQ-learning assumes that the agent can independently select its action because it has already avoided the collision. However, it is likely that the agent may collide with the same agent because another agent is still in the near location.



x	y	x2	Y2	←	↓	→	↑
4	2	5	2	0.1	0.6	0.4	0.1

x	y	←	↓	→	↑
4	3	0.1	0.1	0.1	0.7

Fig. 5. Example subsequent interferences. (Color figure online)

GPCQ-learning greedily selects an action when an agent is not in an interfered state to avoid unnecessary exploration, and it changes the equation for updating the Q-value of the augmented joint state depending on whether the agent is in an interfered state in the next step in order to avoid optimistic evaluation of the optimal Q-value of the augmented joint state. GPCQ-learning was demonstrated to outperform CQ-learning in several maze games.

3.2 Deadlock Resulting from GPCQ-Learning

In pursuit games, agents (depicted by numbers in Fig. 6) try to move next to a target (depicted by T) in a square field, and the game finishes when all the agents are next to the target. In this paper, the target does not move from the initial position. A state of each agent is represented by a difference in positions between the agent and the target, i.e. $(-6 \leq \mathrm{dx} \leq 6, -6 \leq \mathrm{dy} \leq 6)$. Actions are *Up*, *Down*, *Left*, and *Right* to move. Rewards are designed as −1 for a movement, −10 for a collision with another agent, and 0 for a movement next to the target and a finish.

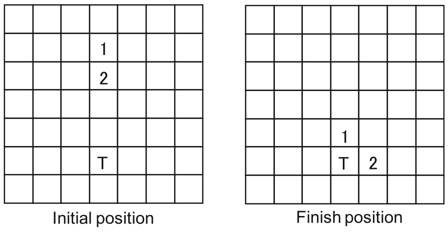


Fig. 6. Example pursuit game.

First, an agent is trained to learn how to move in order to touch the target in a single-agent environment. The initial positions of the target and the agent are randomly selected in every episode. Then, multiple agents try to find an optimal policy for moving in order to touch the target at the same time in a multi-agent environment. The initial positions of the target and the agents were fixed in seven patterns for evaluation, as shown in Fig. 7. For patterns 1–3, the agents can touch the target by greedily selecting their actions without interference with the other agents. For patterns 4–7, an agent collides with another agent if it greedily selects its action on the basis of Q-values learned in a single-agent environment.

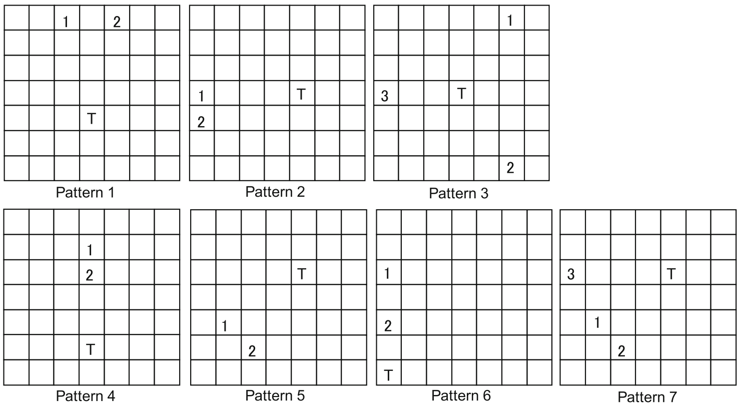


Fig. 7. Seven initial agent/target-position patterns used for evaluation.

For patterns 1–3, the agents using GPCQ-learning find optimal paths resulting in the minimum number of steps to finish while agents using CQ-learning take more steps because they ε-greedily select their actions, resulting in unnecessary exploration. For pattern 4, the agents using GPCQ-learning perform better because they avoid unnecessary augmentation of joint states. For patterns 5–7, the agents using CQ-learning perform much better. For patterns 6–7, in particular, the agents using GPCQ-learning rarely finish the games (depicted as – in Table 2). This is because a repetitive pattern of action-states does not create a difference in the immediate rewards.

Table 2. Comparison of CQ and GPCQ-learning for pursuit games

Patterns	Interference	No. of agents	Min No. of steps	CQ		GPCQ	
				Mean	Std dev.	Mean	Std dev.
1	No	2	4	5.17	1.89	4.00	0.00
2	No	2	4	6.23	8.12	4.00	0.00
3	No	3	4	5.61	13.7	4.00	0.00
4	No	2	4	5.88	1.51	5.14	0.53
5	Yes	2	4	8.53	12.4	274.00	1620.00
6	Yes	2	4	117	219	–	–
7	Yes	3	4	39.5	45.8	–	–

Looking at Fig. 8, we see that agent 1 first greedily selects an action of *upward* in accordance with the prelearned Q-value and detects a difference in immediate rewards because it collides with agent 2 (Fig. 8(a)). It then augments its state with the state of agent 2. For this augmented joint state, agent 1, using GPCQ-learning, decides its action ε-greedily and may select an action *Left* (Fig. 8(b)). After it moves to the left, because it is no longer in an augmented joint state, it greedily selects action *Right* to move back to the previous position in accordance with the prelearned Q-value (Fig. 8 (c)). The reward of selecting action *Right* in a multi-agent environment is the same (i.e. −1) as that in a single-agent environment. Agent 2 also gains the same reward (i.e. 0) as in a single-agent environment because rewards for a touch and finish are the same. Because both rewards are the same as in a single-agent environment, the state of agent 1 is not augmented. Even if agent 1 selects an action of *Right* or *Down*, it may return to the same position because of the same reason. Because any state of the agent is no longer augmented in the situation, it becomes trapped in repetitive movements (i.e. deadlock).

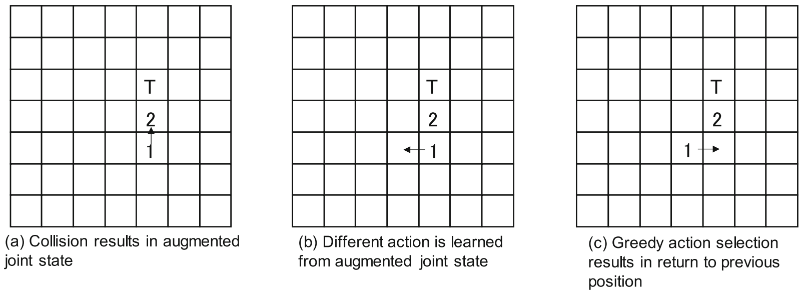

T
2
1
(a) Collision results in augmented joint state
T
2
1
(b) Different action is learned from augmented joint state
T
2
1
(c) Greedy action selection results in return to previous position

Fig. 8. Mechanism of deadlock resulting from GPCQ-learning.

4 Proposed Method

In Algorithm 1, the hatched portions show the differences between CQ-learning and GPCQ-learning and the underlined portion shows the difference between GPCQ-learning and our proposed method. GPCQ-learning selects its action greedily in an unaugmented state (line 13) and changes updating equations based on whether the agent is still in an interfered state in the next step (line 28–29). The proposed method detects repetitive movements between an augmented joint state and a non-augmented state on the basis of states' log and then augments the state of a deadlocked agent to include the state of the other agent, which enables the agent to learn how to avoid the deadlock by updating Q-values.

Although the proposed algorithm can only detect two steps cycle deadlocks, a longer cycle deadlock can be neglectable thanks to the assumption of the sparse interaction.

```
Algorithm 1: Improved GPCQ-learning algorithm
1:Train $Q_k$ independently first, initialize $Q_k^{aug}$ to zero
   and $W_k$=empty
2:Set $t$=0
3:while true do
4:  observe local state $s_k(t)$
5:  if $s_k(t)$ is part of a $\overrightarrow{s_k}$ and the info of $\overrightarrow{s_k}$ present
     in the system state s(t) then
6:    if a set of $\overrightarrow{s_k}$ contains more than two $s_k$ with
       $s_k(t) = s_k$ then
7:      Select an agent $l$ randomly from a set of $\overrightarrow{s_k}$
        with $s_k(t) = s_k$
8:      Select $a_k(t)$ in accordance with $Q_k^{aug}(s_k, s_l)$
        ε-greedily
9:    else
10:     Select $a_k(t)$ in accordance with $Q_k^{aug}$
        ε-greedily
11:   end if
12: else
13:   Select $a_k(t)$ in accordance with $Q_k$ greedily
14: end if
15: observe $r_k = R_k(s(t), a(t))$,  $s'_k$ from  $\mathrm{T}(\mathrm{s(t)}, \mathrm{a(t)})$
16: Store <$s_k(t), a_k(t), r_k(t)$> in $W_k(s_k, a_k)$
17: if p-value of Student's t-test ($W_k(s_k, a_k)$, $\mathrm{E}(R_k(s_k, a_k))$)
    < $p_{th}$ then
18:   Store <$s_k(t), a_k(t), s_l(t), r_k(t)$> in $W_k(s_k, a_k, s_l)$
      for all other agents
19:   for all extra information $s_i$ about another agent $l$
      Present in s(t) do
20:     if p-value of Student's t-test
        $W_k(s_k, a_k, s_l)$, $\mathrm{E}(R_k(s_k, a_k)) < p_{th}$ then
21:       augment $s_k$ with $s_l$ to $\overrightarrow{s_k}$ and add it to $Q_k^{aug}$
22:     end if
23:   end for
24: end if
25: if agent $k$ selected a greedy action and agent $k$
    was/will be in the same augmented joint state
    at $t$=$t$-1/$t$+1 then
26:   augment $s_k$ with $s_l$ to $\overrightarrow{s_k}$ and add it to $Q_k^{aug}$
27: if $s_k(t)$ is part of $\overrightarrow{s_k}$ and information of $\overrightarrow{s_k}$
    is in s(t) then
28:   if $s'_k$ and $s'_l$ is in part of $\overrightarrow{s_k}$ then
29:     $Q_k^{aug}(\overrightarrow{s_k}, a_k) \leftarrow (1 - \alpha_t) Q_k^{aug}(\overrightarrow{s_k}, a_k) +$
          $\alpha_t [r_k + \gamma max_{\alpha'_k} Q_k^{aug}(s'_k, a'_k, s'_l)]$
```

```
30:   else
31:      $Q_k^{aug}(\overrightarrow{s_k}, a_k) \leftarrow (1-\alpha_t)Q_k^{aug}(\overrightarrow{s_k}, a_k) +$
            $\alpha_t[r_k + \gamma max_{\alpha_k'} Q_k(s_k', a_k')]$
32:   end if
33: else
34:   No need to update Q-value
35: end if
36: $t$=$t$+1
37:end while
```

5 Evaluation

We evaluated our proposed learning method in comparison with existing methods: independent learning, JSQ-learning, JSAQ-learning, CQ-learning, and GPCQ-learning. The number of episodes was set to 20,000 for independent learning, JSQ-learning, and JSAQ-learning and 10,000 for CQ-, GPCQ-, and improved GPCQ-learning because CQ-learning and its extensions require prelearning (in this case, 10,000 episodes) in a single-agent environment. In the prelearning, the initial positions of the agents and the target are randomly selected, and ε was set to 0.3 to ensure that the agents could sufficiently explore the environment.

The seven initial agent/target-position patterns shown in Fig. 7 were used for our evaluation. For CQ-, GPCQ-, and improved GPCQ-learning, the length of the window used to calculate the distribution of immediate rewards was set to 20. The threshold of the Student's t-test, p_{th}, was set to 0.01, as was done by Hauwere et al. (2010, 2011).

State-action space of an agent is $13 \times 13 \times 4 = 676$ for an agent using independent learning. If the number of agents is two, the space is $13 \times 13 \times 13 \times 13 \times 4 = 114{,}244$ for the agents using JSQ-learning, $13 \times 13 \times 13 \times 13 \times 4 \times 4 = 456{,}976$ for the agents using JSAQ-learning. If the number of agents is three, the space is $13 \times 13 \times 13 \times 13 \times 13 \times 13 \times 4 = 19{,}307{,}236$ for the agents using JSQ-learning and $13 \times 13 \times 13 \times 13 \times 13 \times 13 \times 4 \times 4 \times 4 = 308{,}915{,}776$ for the agents using JSAQ-learning.

Table 3 shows the number of steps to finish and the standard deviation. For patterns 3 and 7 in which there are three agents, the increase of the state-action space by JSQ-learning and JSAQ-learning clearly reduces the efficiency of finding an optimal action policy. For patterns 1–3, the proposed method, as well as GPCQ-learning, found the optimal paths because there were no interferences between the agents if the agents greedily selected their actions. A slight improvement was obtained for pattern 4. The proposed method substantially outperformed GPCQ-learning for patterns 5–7 while the performance of GPCQ-learning was worst because of deadlocks. For pattern 7, the path found using GPCQ-learning was far from being optimal because there were frequent collisions between the agents in this setting, which is inconsistent with the assumption of sparse interaction. In this case, independent learning, which coincidentally found better paths, performed the best.

Table 3. Evaluation results

	Independent		JSQ		JSAQ		CQ		GPCQ		Proposed	
Patterns	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.	Mean	Std dev.
1	5.07	1.56	5.23	2.35	6.91	9.65	5.17	1.89	4.00	0.00	4.00	0.00
2	5.65	1.76	5.38	2.82	7.11	10.5	6.23	8.12	4.00	0.00	4.00	0.00
3	5.02	1.54	6.87	8.14	108	110	5.61	13.7	4.00	0.00	4.00	0.00
4	6.77	2.56	6.69	4.12	7.73	14.2	5.88	1.51	5.14	0.527	5.09	0.41
5	5.11	1.72	5.22	2.47	7.47	10.7	8.53	12.4	274	1620	4.35	0.87
6	6.40	5.07	6.59	4.91	8.54	31.2	117	219	–	–	5.50	1.02
7	8.32	7.27	11.5	11.8	141	119	39.5	45.8	–	–	9.66	3.65

6 Conclusion

We previously observed that agents using GPCQ-learning sometimes fall into a deadlock if there is no difference in the immediate rewards between a single-agent environment and a multi-agent environment.

Our proposed method breaks such a deadlock by detecting them and augmenting the state of a deadlocked agent to include the state of the other agent.

Evaluation against existing five methods, including GPCQ-learning, using seven initial agent/target-position patterns demonstrated that the proposed method outperforms existing methods for most patterns.

References

Bloembergen, D., Tuyls, K., Hennes, D., Kaisers, M.: Evolutionary dynamics of multi-gent learning: a survey. J. Artif. Intell. Res. **53**(1), 659–697 (2015)

Vlassis, N.: A concise introduction to multiagent systems and distributed artificial intelligence. Synth. Lect. Artif. Intell. Mach. Learn. **1**(1), 1–71 (2007)

Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the 15th National Conference on Artificial Intelligence, pp. 746–752 (1998)

Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: Proceedings of the 17th International Conference on Machine Learning, pp. 535–542 (2000)

Sen, S., Sekaran, M., Hale, J.: Learning to coordinate without sharing information. In: Proceedings of the 12th National Conference on Artificial Intelligence, pp. 426–431 (1994)

Tan, M.: Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the 10th International Conference on Machine Learning, pp. 330–337 (1993)

Melo, F., Veloso, M.: Learning of coordination: exploiting sparse interactions in multiagent systems. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 773–780 (2009)

Hauwere, Y., Vrancx, P., Nowé, A.: Learning multi-agent state space representations. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 715–722 (2010)

Hauwere, Y.: Sparse interactions in multi-agent reinforcement learning. Ph.D. thesis, Vrije Universiteit Brussel (2011)

Kujirai, T., Yokota, T.: Greedy action selection and pessimistic Q-value updates in cooperative Q-learning. In: Proceedings of the SICE Annual Conference, pp. 821–826 (2018)

Kujirai, T., Yokota, T.: Greedy action selection and pessimistic Q-value updating in multi-agent reinforcement learning with sparse interaction. SICE J. Control Meas. Syst. Integr. **12**(3), 76–84 (2019)

Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, Cambridge University (1989)

Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)

Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 195–210 (1996)

Melo, F., Veloso, M.: Decentralized MDPs with sparse interactions. Artif. Intell. **175**(11), 1757–1789 (2011)

Aras, R., Dutech, A., Charpillet, F.: Cooperation through communication in decentralized Markov games. In: Proceedings of the International Conference on Advances in Intelligent Systems - Theory and Applications (2004)