IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 November 2023, accepted 6 December 2023, date of publication 7 December 2023, date of current version 14 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340867

RESEARCH ARTICLE

An Efficient Centralized Multi-Agent Reinforcement Learner for Cooperative Tasks

DENGYU LIAO[1], ZHEN ZHANG[1], TINGTING SONG[2], AND MINGYANG LIU[1]
[1]Shandong Key Laboratory of Industrial Control Technology, School of Automation, Qingdao University, Qingdao 266071, China
[2]Qingdao Metro Group Company Ltd., Operating Branch, Qingdao 266000, China

Corresponding author: Zhen Zhang (tbsunshine8@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61903209, and in part by the Science and Technology Support Plan for Youth Innovation of Universities in Shandong Province under Grant 2019KJN033.

ABSTRACT Multi-agent reinforcement learning (MARL) for cooperative tasks has been extensively researched over the past decade. The prevalent framework for MARL algorithms is centralized training and decentralized execution. Q-learning is often employed as a centralized learner. However, it requires finding the maximum value by comparing the Q-value of each joint action a' in the next state s' to update the Q-value of the last visited state-action pair (s,a). When the joint action space is extensive, the maximization operation involving comparisons becomes time-consuming and becomes the dominant computational burden of the algorithm. To tackle this issue, we propose an algorithm to reduce the number of comparisons by saving the joint actions with the top 2 Q-values (T2Q). Updating the top 2 Q-values involves seven cases, and the T2Q algorithm can avoid traversing the Q-table to update the Q-value in five of these seven cases, thus alleviating the computational burden. Theoretical analysis demonstrates that the upper bound of the expected ratio of comparisons between T2Q and Q-learning decreases as the number of agents increases. Simulation results from two-stage stochastic games are consistent with the theoretical analysis. Furthermore, the effectiveness of the T2Q algorithm is validated through the distributed sensor network task and the target transportation task. The T2Q algorithm successfully completes both tasks with a 100% success rate and minimal computational overhead.

INDEX TERMS Multi-agent reinforcement learning, reinforcement learning, Q-learning, multi-agent system.

I. INTRODUCTION

Reinforcement learning [1] a prevalent method for optimization of a Markov Decision Process (MDP). It obtains the optimal strategy by trial and error without the a priori knowledge of the state transition function and the reward function. In recent years, it has been applied to predicting the battery capacity fade [2], remaining-useful-life (RUL) maintenance [3] and optimization in the path planning and management of unmanned ships [4].

A variety of complex large-scale problems can be modeled as multi-agent systems (MASs). Multi-agent reinforcement learning (MARL) [5], [6],has been applied to many problems that can be modeled as MASs, such as dynamic web service composition [7], multi-intersection traffic signal control [8], bike rebalancing [9] and path planning in autonomous surface vehicles [10]. In the multi-agent setting, the agent constantly changes its strategy during the training process, which leads to the problem of non-stationarity. Centralized training and decentralized execution (CTDE) has become a prevalent framework to deal with non-stationarity, and Q-learning is a classical reinforcement learning algorithm with a convergence guarantee.

The associate editor coordinating the review of this manuscript and approving it for publication was Siddhartha Bhattacharyya.

A. MOTIVATION

The efficiency of reinforcement learning algorithms is crucial for real-world decision-making in scenarios like robot path planning and traffic signal control. The computational load has always posed a challenge for reinforcement learning, primarily stemming from two factors. First, the number of samples used for learning is influenced by the exploration strategy adopted by the agents. Second, given the samples used for learning, the computational operations of the algorithms themselves can be time-consuming. This paper specifically addresses the second challenge.

With the availability of abundant computing resources, the training speed of deep reinforcement learning (DRL) has significantly improved. However, reinforcement learning, whether in tabular form or in the context of deep learning, still requires substantial time during the training phase. While DRL algorithms can benefit from GPU acceleration, this is not the case for tabular reinforcement learning algorithms. Many tabular Q value-based reinforcement learning algorithms utilize the Bellman equation to update the Q-value function. The most time-intensive operation during this enhancement is the computation of the maximum Q-value within the joint action space. When the joint action space is extensive, this operation can become a major computational burden for the algorithm.

B. CONTRIBUTION
We propose an algorithm to reduce the times of comparing by saving the joint actions with the top 2 Q-values (T2Q). The updating of the top 2 Q-value includes seven cases. The T2Q algorithm can avoid traversing the Q-table to update the Q-value in five of the seven cases, thus reducing the computational burden. Theoretical analysis reveals that the upper bound of the ratio of expected execution times when comparing T2Q to Q-learning decreases as the number of agents increases. To validate these findings, the running times of T2Q and Q-learning are evaluated in fictitious two-stage stochastic games. The simulation results align with the theoretical analysis, providing further support. Moreover, the effectiveness of the T2Q algorithm is demonstrated through two fully cooperative tasks, where the T2Q algorithm attains a 100 The rest of this article is organized as follows.

C. ORGANIZATION
Section II reviewsthe related work. Section III introduces stochastic games, Q-learning and notations used in this paper. Section IV elaborates the T2Q algorithm and analyzes the expected computation burden needed. Section V studies the effectiveness of the T2Q algorithm empirically. Section VI summarizes the conclusions.

II. PREVIOUS WORK
We focus on fully cooperative MARL algorithms. In a fully cooperative scenario, the objective is to obtain the maximum expected cumulative return of all agents. The reviewed MARL algorithms are classified into joint action learners (JALs) and independent learners. A JAL needs to evaluate the utility of joint actions, while an independent learner evaluate the the utility of its own actions.

A JAL seeks the optimal joint action at each state. Team Q-learning [11] assumes that each state corresponds to the only optimal joint action to avoid coordination mechanisms. Reference [12] requires each agent to build a model of the other agent to promote coordination. OAL [13] is one of the few JALs that have been proved to converge to the optimal Nash equilibrium (NE), which corresponds to the optimal joint strategy in cooperative stochastic games. In PMR-EGA [14], the Q-value function of the joint action is used to estimate the gradient information of joint strategy [15]. It has been proved that PMR-EGA converges in repeated games with two optimal joint action without common component actions [16].

An indepedent learner does not require the observation of the actions of the other agent. Reference [17] simply extends Q-learning from a single-agent environment to a multi-agent environment. However, this method does not solve the nonstationary problem caused by the concurrent learning. In [18], [19], and [20] the dynamics of IQL with Boltzmann action selection and varepsilon$-greedy action selection in double-agent double-action repeated games are analyzed. FMQ [21] employs the maximum reward in history and the frequency of obtaining the maximum reward to achieve coordination. FMRQ [22] and EAQR [23]adopt the frequencies used in FMQ and performs well in several stochastic games. SOoN [24]optimizes the joint strategy by estimating the farsighted frequency of the maximum reward. FMQ is used for repeated games only, while FMRQ, EAQR, and SOoN can be used for both repeated games and stochastic games.

Multi-agent deep reinforcement learning (MADRL) [23] is a hot research topic in recent years. CTDE is an important idea to alleviate the problem of the exponential growth of the joint action space with the increase of the number of agents. MADDPG [24] uses decentralized critical networks for each agent but still needs to choose joint actions during the stage of concentralized learning. In COMA [25], the Q-value function is estimated by a centralized critic network, and each agent selects an action through a decentralized actor network. However, when the number of agents increases, training acritic with joint actions as input is difficult. Therefore, a variety of Q-value function decomposing method arise. QMIX [26] utilizes hybrid networks to implement the individual-global-max (IGM) principle and considers the impact of the global state. VDNs [27] trains the global Q-network like DQN and the global Q-value function is decomposed into the sum of each independent Q-network QTRAN [28] trains a value netwok as the gap between the global Q-value function and the sum of the local Q-value functions. The efficiency of multi-agent algorithms is crucial for addressing real-world problems.

The CME-AO algorithm [29] incorporates a novel parallel communication protocol to enhance the process of multi-robot space exploration, while also reducing both computational complexity and time. Meanwhile, the Antares algorithm [30] leverages the characteristics of ant systems to create a decentralized and self-organized P2P information system within computational grids, thereby enhancing the effectiveness of both basic and range queries.

The T2Q algorithm proposed in this paper has the following characteristics. First, compared with most of the JALs and independent learners, the convergence of the T2Q algorithm can be guaranteed because the Q-value updating rule of this algorithm is the same as Q-learning. Second, compared with Q-learning, the T2Q algorithm requires much less computation burden when the number of agents is large.

III. PRELIMINARIES

A. NOTATIONS

TABLE 1. Notations.

Symbol	Meanings
t	time step
n	the number of agents
m	the number of actions of each homogeneous agent
$Q_t(s,a)$	the Q-value conditioned on the global state s and the joint action a
$a, a(s)$	the executed joint action in s
$[a(s)]_i$	the executed action of agent i in s
$a_t^*(s)$	the optimal joint action in s at time step t
$[a_t^*(s)]_i$	the optimal action of agent i in s
$a_t^{**}(s)$	the joint action with the second largest Q-value in s
$a_{t+1}^*(s)$	the optimal joint action in s at time step $t+1$
$a_{t+1}^{**}(s)$	the joint action with the second largest Q-value in s at time step $t+1$
$\overline{a}_t(s)$	any joint action other than $a_t^*(s)$ and $a_t^{**}(s)$
$\pi_i(s)$	the strategy of agent i, it is deterministic in this paper
$\pi(s)$	the joint strategy composed of $\pi_i(s)$, for i=1,2,...,n
$Q_t^*(s)$	the maximum Q-value in s
$Q_t^{**}(s)$	the second largest Q-value in s
$Q_{t+1}^*(s)$	the maximum Q-value in s at time step $t+1$
$Q_{t+1}^{**}(s)$	the second largest Q-value in s at time step $t+1$
α	the learning rate, a number within (0,1)
γ	the discount factor that balances the rewards received in the near future and the far future, a number within (0,1)

B. STOCHASTIC GAMES

A stochastic game involves a tuple $< S, A_1, \ldots A_n, T, r_1, \ldots r_n >$ where n is the number of agents, S represents the set of the states, A_i denotes the action set of agent i, $A = A_1 \times A_2, \ldots, \times A_n$ is the joint action set, $T : S \times A \times S \to [0, 1]$ is the state transition function that is the probability distribution of transition to the next state, and the reward function r_i: $S \times A \times S \to R$ is the immediate reward obtained by agent i. In fully cooperative tasks, the common reward function $r = \sum_{i=1}^{n} r_i$ is used to maximize the expected cumulative reward as follows:

$$E_{s_0}^{\pi}\{R(s(t))\} = E_{s_0}^{\pi}\{\sum_{k=0}^{K} \gamma^k r(s(t+k+1)) \quad (1)$$

where $E_{s_0}^{\pi}\{\cdot\}$ represents the expectation w.r.t. the Markov chain $\{s(t)\}_{t\geq 0}$ induced by the joint strategy profile π with the initial state $s(0) = s_0$, K is the length of duration of an episode.According to the theory of dynamic programming [31], [32], there is at least one optimal pure strategy in a MDP.

C. Q-LEARNING

Q-learning [33]is a table-based reinforcement learning algorithm that has been proven to converge to the optimal strategy. The Q-value function is updated as follows:

$$Q(s, a) \leftarrow (1-\alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \quad (2)$$

where $\alpha \in (0, 1)$ is the learning rate. During the learning stage, some action exploration scheme such as ε -greedy [1] are used to choose an action.

IV. THE T2Q ALGORITHM

A. FORMULATION OF THE ALGORITHM

The T2Q algorithm is applicable for solving fully cooperative stochastic games where both the state space and action space are discrete. In these scenarios, states, actions, and rewards can be shared among all participating agents. This algorithm is well-suited for situations with a constrained number of discrete states and collective actions, making it effective even when dealing with heterogeneous agents that possess varying discrete action spaces.

T2Q adopts the CTDE paradigm in which the component action is chosen by each agent, and the Q-value function of the joint actions is learned in a centralized manner. In T2Q, each agent observes the global state s and chooses its action according to:

$$[a(s)]_i \leftarrow \left\{ \begin{array}{ll} \pi_i(s), & \textit{with probability of } \varepsilon \\ \textit{a random action other than } \pi_i(s), & \textit{with probability of } \frac{1-\varepsilon}{|A_i|-1} \end{array} \right\} \quad (3)$$

where $\varepsilon \in (\frac{1}{|A_i|}, 1)$ is the probability of selecting the component action of the optimal joint action. As with Q-learning, after each state transition (s, a, r, s'), the Q-value is updated as follows:

$$Q_{t+1}(s, a) \leftarrow Q_t(s, a) + \alpha \left[r(s, a, s') + \gamma Q_t(s', a^*(s')) - Q_t(s, a) \right] \quad (4)$$

where s is the state at time step t, and s' is state at time step t+1. The pseudocode for the T2Q algorithm is shown in algorithm 1, where line 4-6 is for decentralized execution, and line 9-39 is for centralized training.

The updating of $a_{t+1}^*(s)$, $a_{t+1}^{**}(s)$, $Q_{t+1}^{**}(s)$ and $Q_{t+1}^{**}(s)$ can be classified into 7 cases as shown in Fig. 1-7.Take case1, case 3, and case 5 as examples to explain the updating rules.

Case 1: Agent i selects the action $a = a_t^*(s)$ in state s, and $Q_{t+1}(s, a) < Q_t^{**}(s)$ ($Q_{t+1}(s, a)$) is obtained by (4).). This

Algorithm 1 The T2Q Algorithm for Stochastic Games

```
1:  Set $t$=0. Initialize strategy $\pi_i(s)$ randomly for each agent $i$ for $i = 1, 2, \ldots, n$. Initialize $Q_t(s,a) \leftarrow 0$,
    $Q_t^*(s) \leftarrow 0, Q_t^{**}(s) \leftarrow 0, a_t^*(s) \leftarrow \arg\max_a Q_t(s,a)$,and $a_t^{**}(s) \leftarrow \arg\max_{a \neq a_t^*(s)} Q_t(s,a)$.
2:  repeat
3:    repeat for each episode
4:      for each agent $i$, do
5:        Select an action according to (3) under the current state $s$.
6:      end for each agent
7:      Observe the state of transition and the common immediate reward $r$.
8:      Update $Q_{t+1}(s,a)$ according to (4).
9:      if $a = a_t^*(s)$ then
10:       if $Q_{t+1}(s,a) < Q_t^{**}(s)$ then
11:         $a_{t+1}^*(s) \leftarrow a_t^{**}(s), Q_{t+1}^*(s) \leftarrow Q_t^{**}(s)$
12:         $a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_{t+1}^*(s)} Q_{t+1}(s,a''), Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$
13:       else
14:         $a_{t+1}^{**}(s) \leftarrow a_t^{**}(s), Q_{t+1}^{**}(s) \leftarrow Q_t^{**}(s)$
15:         $a_{t+1}^*(s) \leftarrow a_t^*(s), Q_{t+1}^*(s) \leftarrow Q_t^*(s)$
16:       end if
17:     else if $a = a_t^{**}(s)$ then
18:       if $Q_{t+1}(s,a) < Q_t^*(s)$ then
19:         $a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_t^*(s)} Q_{t+1}(s,a''), Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$
20:         $a_{t+1}^*(s) \leftarrow a_t^*(s), Q_{t+1}^*(s) \leftarrow Q_t^*(s)$
21:       else
22:         $a_{t+1}^{**}(s) \leftarrow a_t^*(s), Q_{t+1}^{**}(s) \leftarrow Q_t^*(s)$
23:         $a_{t+1}^*(s) \leftarrow a, Q_{t+1}^*(s) \leftarrow Q_{t+1}(s,a)$
24:       end if
25:     else
26:       if $Q_{t+1}(s,a) > Q_t^*(s)$ then
27:         $a_{t+1}^{**}(s) \leftarrow a_t^*(s), Q_{t+1}^{**}(s) \leftarrow Q_t^*(s)$
28:         $a_{t+1}^*(s) \leftarrow a, Q_{t+1}^*(s) \leftarrow Q_{t+1}(s,a)$
29:       else if $Q_{t+1}(s,a) > Q_t^{**}(s)$ then
30:         $a_{t+1}^{**}(s) \leftarrow a, Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s,a)$
31:         $a_{t+1}^*(s) \leftarrow a_t^*(s), Q_{t+1}^*(s) \leftarrow Q_t^*(s)$
32:       else
33:         $a_{t+1}^{**}(s) \leftarrow a_t^{**}(s), Q_{t+1}^{**}(s) \leftarrow Q_t^{**}(s)$
34:         $a_{t+1}^*(s) \leftarrow a_t^*(s), Q_{t+1}^*(s) \leftarrow Q_t^*(s)$
35:       end if
36:     end if
37:     for each agent $i$, do
38:       $\pi_i(s) \leftarrow [a_{t+1}^*(s)]_i$
39:     end for each agent
40:     $s \leftarrow s'$
41:     $t \leftarrow t+1$
42:   until the episode is over
43: until the predefined number of episodes have been played
44: return $\pi_i(s)$ for each agent.
```

means that $Q_{t+1}^*(s) < Q_t^{**}(s)$ in time step $t+1$. Therefore, $a_t^{**}(s)$ becomes the optimal joint action and $Q_t^{**}(s)$ becomes the largest Q-value in state s in time step $t+1$. Thus the operation $Q_{t+1}^*(s) \leftarrow Q_t^{**}(s)$ and $a_{t+1}^*(s) \leftarrow a_t^{**}(s)$ is performed in Fig. 1. Then the joint action with the second largest Q-value $a_{t+1}^{**}(s)$ and its Q-value $Q_{t+1}^{**}(s)$ in state s need to be updated by traversing the Q-table conditioned on state s, namely, performing the operation of $a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_{t+1}^*(s)} Q_{t+1}(s,a'')$ and $Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$.

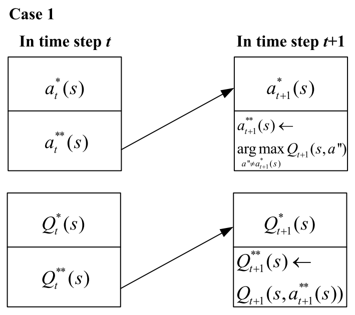

Case 1
In time step t
In time step t+1
$a_t^*(s)$
$a_t^{**}(s)$
$a_{t+1}^*(s)$
$a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_{t+1}^*(s)} Q_{t+1}(s,a'')$
$Q_t^*(s)$
$Q_t^{**}(s)$
$Q_{t+1}^*(s)$
$Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$

FIGURE 1. *When $a = a_t^*(s)$ and $Q_{t+1}(s,a) < Q_t^{**}(s)$.*

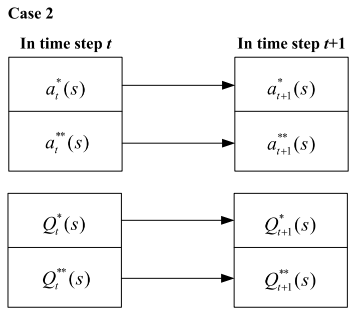


FIGURE 2. *When $a = a_t^*(s)$ and $Q_{t+1}(s,a) \geq Q_t^{**}(s)$.*

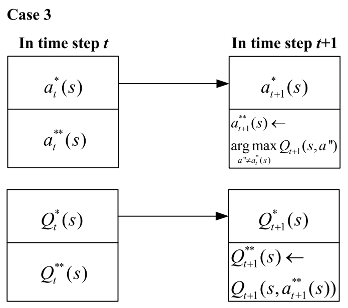

Case 3
In time step t
In time step t+1
$a_t^*(s)$
$a_t^{**}(s)$
$a_{t+1}^*(s)$
$a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_t^*(s)} Q_{t+1}(s,a'')$
$Q_t^*(s)$
$Q_t^{**}(s)$
$Q_{t+1}^*(s)$
$Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$

FIGURE 3. *When $a = a_t^{**}(s)$ and $Q_{t+1}(s,a) < Q_t^*(s)$.*

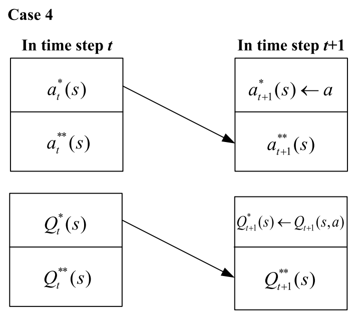


FIGURE 4. *When $a = a_t^{**}(s)$ and $Q_{t+1}(s,a) \geq Q_t^*(s)$.*

Case 3: Agent i selects the action $a = a_t^{**}(s)$ in state s and $Q_{t+1}(s,a) < Q_t^*(s)$ ($Q_{t+1}(s,a)$ is obtained by (4).). This means that $Q_{t+1}^*(s) < Q_t^*(s)$ in time step $t+1$. Therefore, $a_t^*(s)$ is still the optimal joint action and $Q_t^*(s)$ is still the largest Q-value in state s in time step $t+1$.

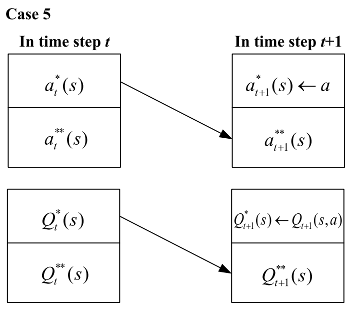


FIGURE 5. ***When $a = \bar{a}_t(s)$ and $Q_{t+1}(s, a) > Q_t^*(s)$.***

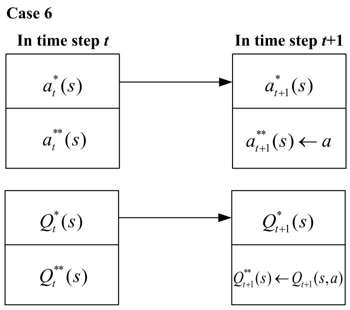


FIGURE 6. ***When $a = \bar{a}_t(s)$ and $Q_t^{**}(s) < Q_{t+1}(s, a) \leq Q_t^*(s)$.***

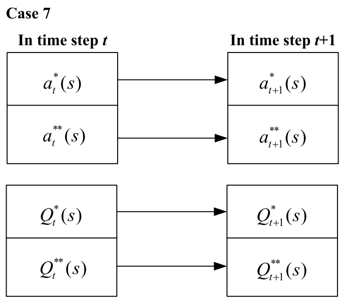
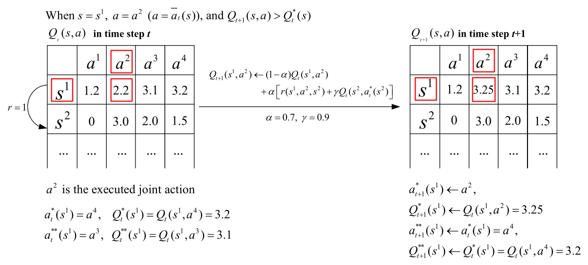


FIGURE 7. ***When $a = \bar{a}_t(s)$ and $Q_{t+1}(s, a) \leq Q_t^{**}(s)$.***

Thus the operation $Q_{t+1}^*(s) \leftarrow Q_t^*(s)$ and $a_{t+1}^*(s) \leftarrow a_t^*(s)$ is performed in Fig. 3. In the meanwhile, the joint action with the second largest Q-value $a_{t+1}^{**}(s)$ and its Q-value $Q_{t+1}^{**}(s)$ in state s needs to be updated by traversing the Q-table conditioned on state s, Namely, perform the operation of $a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_{t+1}^*(s)} Q_{t+1}(s, a'')$ and $Q_{t+1}^{**}(s) \leftarrow Q_{t+1}(s, a_{t+1}^{**}(s))$.

Case 5: Agent i selects the action $a = \bar{a}_t(s)$ in state s and $Q_{t+1}(s, a) > Q_t^*(s)$ ($Q_{t+1}(s, a)$ is obtained by (4).). This means that $Q_{t+1}^*(s) > Q_t^*(s)$ in time step t+1. Therefore, $a = \bar{a}_t(s)$ becomes the optimal joint action and $Q_{t+1}(s, a)$ becomes the largest Q-value in state s in time step t+1. Thus the operation $Q_{t+1}^*(s) \leftarrow Q_{t+1}(s, a)$ and $a_{t+1}^*(s) \leftarrow a$ is performed in Fig. 5. In the meanwhile, the joint action with the second largest Q-value $a_{t+1}^{**}(s)$ and its Q-value $Q_{t+1}^{**}(s)$ in state s becomes $a_t^*(s)$ and the maximum Q-value $Q_t^*(s)$ respectively. Thus the operation of $Q_{t+1}^{**}(s) \leftarrow Q_t^*(s)$ and $a_{t+1}^{**}(s) \leftarrow a_t^*(s)$ needs to be performed.


When $s = s^1$, $a = a^2$ ($a = \bar{a}_t(s)$), and $Q_{t+1}(s,a) > Q_t^*(s)$
$Q_t(s,a)$ in time step t
$Q_{t+1}(s,a)$ in time step t+1
$r = 1$
$Q_{t+1}(s^1,a^2) \leftarrow (1-\alpha)Q_t(s^1,a^2) + \alpha[r(s^1,a^2,s^2) + \gamma Q_t(s^2,a_t^*(s^2))]$
$\alpha = 0.7$, $\gamma = 0.9$
a^2 is the executed joint action
$a_t^*(s^1) = a^4$, $Q_t^*(s^1) = Q_t(s^1,a^4) = 3.2$
$a_t^{**}(s^1) = a^3$, $Q_t^{**}(s^1) = Q_t(s^1,a^3) = 3.1$
$a_{t+1}^*(s^1) \leftarrow a^2$,
$Q_{t+1}^*(s^1) \leftarrow Q_t(s^1,a^2) = 3.25$
$a_{t+1}^{**}(s^1) \leftarrow a_t^*(s^1) = a^4$,
$Q_{t+1}^{**}(s^1) \leftarrow Q_t^*(s^1) = Q_t(s^1,a^4) = 3.2$

	a^1	a^2	a^3	a^4
s^1	1.2	2.2	3.1	3.2
s^2	0	3.0	2.0	1.5
...	...	...	...	...

	a^1	a^2	a^3	a^4
s^1	1.2	3.25	3.1	3.2
s^2	0	3.0	2.0	1.5
...	...	...	...	...

FIGURE 8. **The updating of the Q-value function and the critical variables in T2Q in case 5.**

Fig. 8 presents the detailed updating process of the Q-value function and the critical variables($a_{t+1}^*(s)$, $a_{t+1}^{**}(s)$, $Q_{t+1}^{**}(s)$, and $Q_{t+1}^{**}(s)$) in the T2Q algorithm. By reserving the value of these variables, we can avoid traversing the Q-table to update the specified Q-value of some state in case 2, case 4, case 5, case 6, and case 7. The selected joint action is a^2, and the common immediate reward from state s^1 to state s^2 is 1. It can be seen that the updated Q-values obtained by T2Q and Q-learning are identical in case 5. It is easy to verify the identity in the other cases.

B. ANALYSIS OF THE ALGORITHM

Compared with Q-learning, the main advantage of T2Q is that it reduces computation burden in the updating process of the Q-value function. The most time-consuming part of the T2Q algorithm is the comparing operation in the operation $a_{t+1}^{**}(s) \leftarrow \arg\max_{a'' \neq a_{t+1}^*(s)} Q_{t+1}(s, a'')$. Thus the expected times of comparing for T2Q is our concern.

Let n denote the number of agents, m denote the number of joint actions, ε denote the probability of selecting the action $\pi_i(s)$, N denote the times of comparing in one update of $a_{t+1}^*(s)$, $a_{t+1}^{**}(s)$, $Q_{t+1}^{**}(s)$ and $Q_{t+1}^{**}(s)$ (line 9-36 in Algorithm 1). Each agent uses the equivalent value of $\varepsilon > \max_i 1/|A_i|$. The expected times of comparing can be obtained by

$$
\begin{aligned}
&E[N] \\
&= E[N|a = a_t^*(s), Q_{t+1}\left(s, a_t^*(s)\right) < Q_t^{**}(s)] && \text{(5-a)} \\
&\quad \times p(a = a_t^*(s), Q_{t+1}\left(s, a_t^*(s)\right) < Q_t^{**}(s)) \\
&\quad + E[N|a = a_t^{**}(s), Q_{t+1}\left(s, a_t^{**}(s)\right) < Q_t^*(s)] \\
&\quad \times p(a = a_t^{**}(s), Q_{t+1}\left(s, a_t^{**}(s)\right) < Q_t^*(s)) \\
&= E[N|a = a_t^*(s), Q_{t+1}\left(s, a_t^*(s)\right) < Q_t^{**}(s)] \\
&\quad \times p(a = a_t^*(s))p(Q_{t+1}\left(s, a_t^*(s)\right) < Q_t^{**}(s)) \\
&\quad + E[N|a = a_t^{**}(s), Q_{t+1}\left(s, a_t^{**}(s)\right) < Q_t^*(s)] \\
&\quad \times p(a = a_t^{**}(s))p(Q_{t+1}\left(s, a_t^{**}(s)\right) < Q_t^*(s)) \\
&\leq (m-1)\,\varepsilon^n p_1 + (m-1)\,\varepsilon^{n-1}\left(\frac{1-\varepsilon}{|A_k|-1}\right)p_2 && \text{(5-b)} \\
&\leq (m-1)\,\varepsilon^n + (m-1)\,\varepsilon^{n-1}\left(\frac{1-\varepsilon}{|A_k|-1}\right) && \text{(5-c)}
\end{aligned}
$$

$$= (m-1)\,\varepsilon^n \left(1 + \frac{1-\varepsilon}{\varepsilon}\frac{1}{|A_k|-1}\right)$$
$$\le 2\,(m-1)\,\varepsilon^n \qquad \text{(5-d)}$$
$$\text{(5)}$$

Equation 5-(a) holds because the operation $a^{**}_{t+1}(s) \leftarrow \underset{a\neq a^{*}_{t+1}(s)}{\arg\max}\ Q_{t+1}(s,a)$ occurs in case 1 and case 3 only. Let $p_1 = p(Q_{t+1}(s, a^*_t(s)) < Q^{**}_t(s))$ and $p_2 = p(Q_{t+1}(s, a^{**}_t(s)) < Q^{*}_t(s))$. Then we have inequation 5-(b), because $p(a = a^{**}_t) \le \varepsilon^{n-1}(\frac{1-\varepsilon}{|A_k|-1})$ where $|A_k| = \min_i |A_i|$. Inequation 5-(c) holds because $p_1 \le 1$ and $p_2 \le 1$. Inequation 5-(d) holds because from $\varepsilon > \max_i 1/|A_i|$ and $|A_k| = \min_i |A_i|$, we have $\frac{1-\varepsilon}{\varepsilon}\frac{1}{|A_k|-1} < 1$.

In practice, ε will be increased from a small positive value to a value close to 1, allowing agents to explore more in the early learning stage and utilize more in the later learning stage. Consider the value of ε grows linearly as follows:

$$\varepsilon_l = \varepsilon_{ini} + \frac{(\varepsilon_{end} - \varepsilon_{ini})}{n_{episode}} l \qquad (6)$$

where l is the elapsed number of episodes and $n_{episode}$ is the total number of episodes, ε_{ini} is the initial value of ε, and ε_{end} is the terminal value of ε. Let N_l denote the times of comparing in one update of $a^*_{t+1}(s)$, $a^{**}_{t+1}(s)$, $Q^{**}_{t+1}(s)$ and $Q^{**}_{t+1}(s)$ in the l-th episode. According to (5), the expected average times of comparing for one update in $n_{episode}$ episodes satisfy

$$E[\frac{1}{n_{episode}}\sum_{l=1}^{n_{episode}} N_l] = \frac{1}{n_{episode}}\sum_{l=1}^{n_{episode}} E[N_l]$$
$$\le \frac{1}{n_{episode}}\sum_{l=1}^{n_{episode}} 2(m-1)\varepsilon_l^n. \qquad (7)$$

When $n_{episode}$ is large, the upper bound in (7) can be approximated by

$$\int_{\varepsilon_{ini}}^{\varepsilon_{end}} 2(m-1)\varepsilon^n d\varepsilon$$
$$= 2(m-1)\frac{\varepsilon^{n+1}}{n+1}\Big|_{\varepsilon_{ini}}^{\varepsilon_{end}}$$
$$= \frac{2(m-1)}{n+1}(\varepsilon_{end}^{n+1} - \varepsilon_{ini}^{n+1})$$
$$\le \frac{2(m-1)}{n+1} \qquad (8)$$

Q-learning performs m times of comparing for one update of Q-value function. Thus we have that $m/\frac{2(m-1)}{n+1} = \frac{n+1}{2}\frac{m}{m-1} \approx \frac{n+1}{2}$ (when the number of joint actions m is large), which means that the computation needed by Q-learning is at least $\frac{n+1}{2}$ times than that of T2Q.

C. THE T2Q ALGORITHM FOR STOCHASTIC GAMES

The performance of the T2Q algorithm is verified by experiments in fictitious two-stage stochastic games in which each agent has the same number of actions. Each stage i has only one state s_i, namely, the state transition is fixed. The reward function is defined as follows:

$$r(s,a) = \frac{m}{index(a)+1} + B \qquad (9)$$

where $index\,(a) \in \{x|x \in \mathbb{Z}^+, 0 \le x \le m-1\}$ is the index of the executed joint action a, B is a random number that follows the uniform distribution on [-10,10], and m is the number of joint actions. The aim is to maximize the common cumulative reward. To validate the convergence of the T2Q algorithm, the immediate reward without B is also recorded to evaluate the cumulative reward without B. The optimal joint action is to select the action with index 0 at both the stages, and the maximal cumulative reward without B is $2m$. The number of agents is $n_{agent} = 2, 3, 4, 5, 6, 7$ and the number of each agent's actions varies from 2 to 5. The performance indices include the success rate and the execution time. The result is an average of 50 runs, each of which includes 500,000 learning episodes and 1 evaluation episode. A run is successful if the optimal undiscounted cumulative reward without B – $2m$ is obtained in the evaluation episode. The execution time is the running time for one complete run. Both T2Q and Q-learning use the parameters $\gamma = 0.9$, $\alpha = 0.2$, and ε following (6) with $\varepsilon_{ini} = 0.35$ and $\varepsilon_{end} = 1.0$. The simulation is conducted on a PC with Windows 10 and I7-7700HQ CPU with 2.80GHz.

Table 2 shows that T2Q obtains a 100% success rate in all cases, which indicates it converges to the optimal joint strategy. Table 3 records the average execution time of one run with 500,000 learning episodes and 1 evaluation episode. The ratio of the execution time of Q-learning to T2Q is used to approximate the ratio of the computation of Q-learning to T2Q. When the number of the joint actions is small, the comparing operation does not dominate the required computation. In this situation, the advantage of T2Q is not obvious. When the number of agents is 6 and 7 and the number of each agent's action is 4 and 5, the execution time of T2Q decreases evidently and the ratio of the execution time of Q-learning to T2Q is larger than $(n+1)/2$, which is consistent with our analysis in section IV-B.The experiment indicates that the upper bound $(n+1)/2$ is much larger than the supremum of the ratio.

TABLE 2. Success rate of the T2Q algorithm in two-stage stochastic games (runs = 50).

	$n_{agent} = 2$	$n_{agent} = 3$	$n_{agent} = 4$	$n_{agent} = 5$	$n_{agent} = 6$	$n_{agent} = 7$
$\lvert A_i\rvert = 2$	100%	100%	100%	100%	100%	100%
$\lvert A_i\rvert = 3$	100%	100%	100%	100%	100%	100%
$\lvert A_i\rvert = 4$	100%	100%	100%	100%	100%	100%
$\lvert A_i\rvert = 5$	100%	100%	100%	100%	100%	100%

V. EMPIRICAL STUDIES FOR COOPERATIVE TASKS

We investigate the of the T2Q algorithm's effectiveness in cooperative tasks through the distributed sensor network (DSN) task and the target transportation task. The difference between them is that the DSN task has a larger joint action

TABLE 3. Average execution time of T2Q | Q-learning (runs = 50,unit: second).

	$n_{agent}=2$	$n_{agent}=3$	$n_{agent}=4$	$n_{agent}=5$	$n_{agent}=6$	$n_{agent}=7$
$\|A_i\|=2$	0.25|0.25	0.36|0.37	0.49|0.51	0.61|0.67	0.74|0.84	0.83|1.08
$\|A_i\|=3$	0.27|0.25	0.41|0.40	0.50|0.62	0.63|1.10	0.75|2.37	0.92|5.75
$\|A_i\|=4$	0.29|0.26	0.41|0.47	0.51|1.00	0.66|2.83	0.79|10.46	1.00|36.24
$\|A_i\|=5$	0.27|0.29	0.43|0.58	0.49|1.79	0.65|7.63	0.91|34.68	1.86|168.4

space and the target transportation task has a larger state space. The T2Q algorithm is in comparison with Q-learning and three prevalent MARL algorithms – MADDPG, QMIX and QTRAN Plus [34]. To accommodate discrete action space, we use the Gumbel-Softmax reparametrization trick [35] for the MADDPG algorithm.

A. TASK 1: DISTRIBUTED SENSOR NETWORK

The objective of the DSN task [36] is to coordinate eight sensors (agents) to catch two targets within the minimum time step and obtain the maximum cumulative reward. Fig. 9 shows that eight sensors are distributed around a grid composed of three cells, in which two targets move randomly. In each step, each target randomly chooses one of three actions – move left, move right, or stay still. The targets take actions sequentially. If a target tries to move into a cell occupied by another target or move out of the grid, the target cannot move and stays still. The location and the energy of the targets constitute the state space of the DSN task, which contains 36 states. All state variables can be sensed by each of the eight sensors.

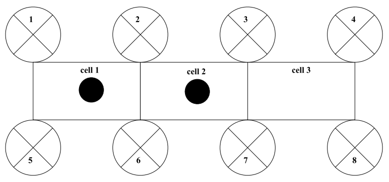


FIGURE 9. Task 1: distributed sensor network.

At each time step, the sensors always take actions before the targets. Each sensor can choose one of the three actions: focus on the left cell, focus on the right cell, or not focus. The amount of joint actions is $3^8 = 6561$. At the beginning of an episode, the energy value of each target is three. When two sensors focus on a target simultaneously, the target's energy decreases by one with a probability of 30% or remains unchanged with a probability of 70%. If a target is focused by three or more sensors, its energy value is reduced by one. If the target's energy goes to 0, it is captured. If both targets are captured or 10 time steps have elapsed, an episode is over.

The reward function of the DSN task is defined as follows. If a target is captured by three sensors, each sensor involved in the capture obtains a reward of 10. If four sensors participate in the capture, only the sensors with the top three indices will be rewarded. Any sensor that performs a focusing action gets an additional immediate reward of -1. The action of not focusing always produces an immediate reward of 0. If two sensors participate in the capture, each sensor can get an additional immediate reward of 6. In the best case, if each target is always focused by two sensors, the sensors can get a common cumulative reward of 52 in 3 steps. However, this situation does not always occur. The optimal joint strategy is to focus on one target for every three sensors. The maximum expected cumulative reward for the DSN task is 42, and the minimum expected number of steps to complete the task is 3. The condition for success is to obtain a cumulative reward of 42 in an evaluation episode.

The experiments are conducted for 50 runs and each run contains $n_{episode}$ learning episodes and 50,000 evaluation episodes. The T2Q algorithm uses the parameters $\gamma = 0.9$, $\alpha = 0.2$, and ε following (6) with $\varepsilon_{ini} = 0.35$ and $\varepsilon_{end} = 1.0$.

The success rate, the average cumulative reward, and the average number of steps are presented in Table 4, Table 5, and Table 6 respectively. Table 4 shows that T2Q, QMIX, QTRAN Plus and Q-learning obtain a success rate of 100% with $n_{episode} = 500{,}000$, which is consistent with the result in Table 5 and Table 6. QTRAN Plus has the highest learning speed because of its generalization ability and the small state space with mere 36 elements. Q-learning has similar performance with T2Q in terms of cumulative reward and steps because they use the equivalent Q-value updating rule. Given more time, MADDPG probably converges to the optimal joint strategy. MADDPG learns slower than QMIX and QTRAN Plus because it has more parameters to be trained. QTRAN Plus, QMIX and MADDPG are MADRL algorithms, thereby requiring much more time for training. The worst case is shown in Table 7 and Table 8. All algorithm shows stable performance for the DSN task with all values of $n_{episode}$.

Table 9 shows the ratio of the execution time used for updating strategy of Q-learning to T2Q, which is used to approximate the ratio of the computation burden of Q-learning to T2Q. Let $t_{Q-learning}/t_{T2Q}$ denote this ratio where $t_{Q-learning}$ is the execution time of updating Q-value according to (2), and t_{T2Q} is the execution time of the code from line 9 to line 39 in Algorithm 1. According to the analysis in Section IV, Q-learning requires at least $(8+1)/2 = 4.5$ times computation compared with T2Q. For task 1, the minimum of $t_{Q-learning}/t_{T2Q}$ is 55, which is much larger than the lower bound 4.5.

TABLE 4. Success rate for task 1.

	$n_{episode}=100{,}000$	$n_{episode}=250{,}000$	$n_{episode}=500{,}000$
T2Q	44.37%	91.68%	**100%**
QMIX	**70.74%**	**96.01%**	**100%**
MADDPG	0%	0%	0%
Q-learning	23.04%	89.29%	**100%**
QTRAN Plus	66.43%	**100%**	**100%**

TABLE 5. Average cumulative reward and standard deviation for task 1.

	$n_{episode}$ = 100, 000	$n_{episode}$ = 250, 000	$n_{episode}$ = 500, 000
T2Q	**42 | 0**	41.99 | 0.05	**42 | 0**
QMIX	41.61 | 0.35	41.96 | 0.11	**42 | 0**
MADDPG	39.64 | 1.32	41.07 | 0.76	41.33 | 0.78
Q-learning	41.98 | 0.07	**42 | 0**	**42 | 0**
QTRAN Plus	41.75 | 0.29	**42 | 0**	**42 | 0**

TABLE 6. Average steps and standard deviation for task 1.

	$n_{episode}$ = 100, 000	$n_{episode}$ = 250, 000	$n_{episode}$ = 500, 000
T2Q	3.88 | 0.70	3.08 | 0.17	**3 | 0**
QMIX	**3 | 0.03**	**3 | 0**	**3 | 0**
MADDPG	3.06 | 0.16	3.19 | 0.24	3.22 | 0.38
Q-learning	4.20 | 0.53	3.11 | 0.20	**3 | 0**
QTRAN Plus	3.32 | 0.47	**3 | 0**	**3 | 0**

TABLE 7. Minimal cumulative reward for task 1.

	$n_{episode}$ = 100, 000	$n_{episode}$ = 250, 000	$n_{episode}$ = 500, 000
T2Q	**42**	41.67	**42**
QMIX	40.66	41.66	**42**
MADDPG	34.51	38.92	39.24
Q-learning	41.66	**42**	**42**
QTRAN Plus	40.52	**42**	**42**

TABLE 8. Maximal steps for task 1.

	$n_{episode}$ = 100, 000	$n_{episode}$ = 250, 000	$n_{episode}$ = 500, 000
T2Q	5.26	3.57	**3**
QMIX	**3.24**	**3**	**3**
MADDPG	3.67	3.80	4.44
Q-learning	5.33	3.55	**3**
QTRAN Plus	4.44	**3**	**3**

TABLE 9. The ratio of the execution time used for updating strategy of Q-learning to T2Q for task 1.

$n_{episode}$ = 100, 000	$n_{episode}$ = 250, 000	$n_{episode}$ = 500, 000
176.97	206.17	55.24

The joint strategy obtained by T2Q with $n_{episode}$ = 500000 learning episodes is illustrated in Fig. 10. The action of focusing is represented by an arrow pointing to the target, and the energy bar above each target is for displaying the energy value. It shows that every three sensors focus on the same target at the same time, which is the optimal joint strategy.

B. TASK 2: TARGET TRANSPORTATION

The target transportation task [37] is illustrated in Fig. 11. In a 7 × 6 grid world, there are two agents represented by circles.

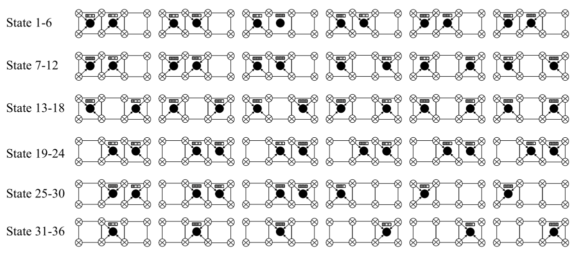


FIGURE 10. The optimal joint strategy of task 1 obtained by T2Q (using 500,000 episodes for training).

The black solid rectangle represents the target. The slanted shaded cells represent obstacles. The home is a 3 × 1 area. The objective of the task is coordinating two agents to grab the target and transport the target to the 3 × 1 home area in the shortest time step.

The global state is defined as: $s = \{x_1, y_1, x_2, y_2, g_1, g_2\}^T$ where $x_i \in \{1, 2, \ldots, 7\}$ and $y_i \in \{1, 2, \ldots, 6\}$ denote the coordinates of agent i, and $g_i = \{FREE, GRAB\_LEFT, GRAB\_RIGHT\}$ denotes the grabbing information. Each agent has five actions – left, right, up, down, and still.

The movement rules of the target transportation task are defined as follows. First, both the agents move at the same time. If an agent moves to the obstacle or collides with another agent, it fails to move and stands still. Second, an agent grabs the target if it is in the right cell or the left cell of the target. Once an agent grabs the target, it cannot move unless the other agent grabs the target as well and they move in the same direction. Third, in the white area, the target moves successfully with a 100% chance if both the agents move in the same direction and no collision occurs. In the gray shaded area, the chance of moving successfully drops to 50%.

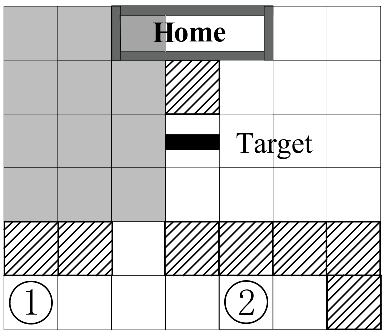


FIGURE 11. Task 2: target transportation.

The reward function is defined as follows: if an agent grabs the target, it receives an immediate reward of 1. If the target reaches the home area, each agent receives an immediate reward of 100. If the target moves in the gray shaded area, each agent receives an immediate reward of 0, if it fails to move in the gray shaded area, each agent receives an

TABLE 10. Success rate for task 2.

	$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
T2Q	**100%**	**100%**	**100%**
QMIX	14%	92%	92%
MADDPG	0%	22%	22%
Q-learning	**100%**	**100%**	**100%**
QTRAN Plus	4%	78%	**100%**

TABLE 11. Average cumulative reward and standard deviation for task 2.

	$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
T2Q	**182 | 0**	**182 | 0**	**182 | 0**
QMIX	-38.4 | 211.97	148.24 | 115.65	148.24 | 115.65
MADDPG	-239.64 | 0.88	-46.44 | 211.12	45.60 | 196.97
Q-learning	**182 | 0**	**182 | 0**	**182 | 0**
QTRAN Plus	-155.68 | 169.15	173.16 | 59.63	**182 | 0**

TABLE 12. Average steps and standard deviation for task 2.

	$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
T2Q	**12 | 0**	**12 | 0**	**12 | 0**
QMIX	68.64 | 54.01	20.64 | 29.60	20.64 | 29.60
MADDPG	120 | 0	70.78 | 53.88	47.46 | 50.28
Q-learning	**12 | 0**	**12 | 0**	**12 | 0**
QTRAN Plus	98.68 | 43.08	14.36 | 15.25	**12 | 0**

TABLE 13. Minimal cumulative reward for task 2.

	$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
T2Q	**182**	**182**	**182**
QMIX	-240	-240	-240
MADDPG	-240	-240	-240
Q-learning	**182**	**182**	**182**
QTRAN Plus	-240	-240	**182**

TABLE 14. Maximal steps for task 2.

	$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
T2Q	**12**	**12**	**12**
QMIX	120	120	120
MADDPG	120	120	120
Q-learning	**12**	**12**	**12**
QTRAN Plus	120	120	**12**

TABLE 15. The ratio of the execution time used for updating strategy of Q-learning to T2Q for task 2.

$n_{episode} = 10,000$	$n_{episode} = 30,000$	$n_{episode} = 50,000$
1.449	1.718	1.283

immediate reward of -4. In the other cases, each agent receives an immediate reward of -1.

The experiment is conducted for 50 runs. Each run contains $n_{episode}$ learning episodes and 5000 evaluation episodes. In each evaluation episode, each agent uses the learned strategy. The minimum expected number of steps to complete the task is 12 and the maximum expected cumulative reward is 182. If the task is completed within 12 steps, an evaluation episode is successful.

The parameters of this experiment are set as follow: $\alpha = 0.2$, $\gamma = 0.9$, and ε following (6) with $\varepsilon_{ini} = 0.4$ and $\varepsilon_{end} = 0.8$.

Table 10, Table 11, Table 12 shows the success rate, the average cumulative reward, and the average number of steps. T2Q and Q-learning exhibit the same performance because they perform the equivalent Q-value update rule. QTRAN Plus, QMIX and MADDPG learn slower because the target transportation task has much more states than the DSN task. The worst case is shown in Table 13 and Table 14. QMIX and MADDPG fail to complete the task within 120 steps in the worst case because they have not been fully trained within 50000 learning episodes.

Table 15 shows the ratio of the execution time used for updating strategy of Q-learning to T2Q. According to the analysis in Section IV, Q-learning requires at least $(2+1)/2 = 1.5$ times computation compared with T2Q. For task 2, the minimum of the ratio is 1.283 which is obtained when $L = 30000$, which approaches but is lower than the lower bound 1.5. This result is still consistent with the theoretical analysis because the lower bound is obtained in the sense of mathematical expectation.

Fig. 12 shows the joint strategy obtained by the T2Q algorithm after $n_{episode} = 50000$ learning episodes. The arrows denote the actions of the agents. This task is completed in 12 steps, which is the minimum expected number of steps for this task. From step 0 to step 3, agent 1 moves to the channel first. From steps 4 to 6, agent 1 goes through the channel to grab the target from the right side of the target, and agent 2 goes through the channel into the gray area. In step 7, agent 2 grabs the target from the left side of the target. The two agents decide to transport the target from the white area into the home area. From step 8 to 12, both agents move in the same direction until the target is transported into the home area. In the best case, the agents will obtain the maximum cumulative reward if they transport the target through the gray area. However, the best case does not always occur. In fact, if they enter the gray area at step 8, they will not be able to achieve the maximum expected cumulative reward. The path of the agents shown in Fig. 12 again verifies the effectiveness of the T2Q algorithm.

The results from both Task 1 and Task 2 indicate that the ratio of execution time between Q-learning and T2Q increases as the number of agents rises from 2 to 8. This implies that T2Q exhibits a more significant advantage over

Q-learning in terms of execution time when a greater number of agents are involved.

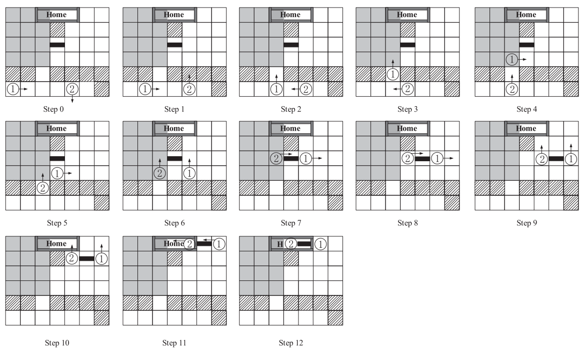


FIGURE 12. The optimal joint strategy of task 2 obtained by T2Q (using 50,000 episodes for training).

VI. CONCLUSION

In this paper, a CTDE based MARL algorithm - T2Q is proposed with the aim of reducing the computation burden during centralized training. The computation for one update of Q-value is analyzed quantitatively.

Theoretical analysis reveals that when the number of the joint action is large, the computation of T2Q is no more than $\frac{2}{n+1}$ of that of Q-learning, where n is the number of joint actions. This theoretical result is verified by two-stage stochastic games. T2Q converges to the optimal joint strategy in all of the stochastic games for testing and the execution time matches the theoretical prediction.

The optimization capability and computational efficiency are further validated through two fully cooperative tasks involving eight agents and two agents, respectively. T2Q is compared against Q-learning, QMIX, MADDPG, and QTRAN Plus. In both tasks, T2Q achieves a 100% success rate with the least computational burden. Simulation results also demonstrate that the ratio of execution time between Q-learning and T2Q aligns with the theoretical findings.

It's worth noting that the computational reduction approach employed in T2Q is applicable only when Q-values are stored in a tabular format, which imposes limitations on its broader applicability. In the future, we plan to address more complex tasks by combining the approach presented in this paper with deep learning techniques. The combination of gradient methods with heuristic approaches may offer a viable solution for obtaining the maximum Q-value in scenarios where the state space and the action space are continuous. Our next objective is to explore the feasibility of reducing computational overhead in such scenarios.

REFERENCES

[1] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *Robotica*, vol. 17, no. 2, pp. 229–235, 1999.

[2] A. Namdari and Z. S. Li, "An entropy-based approach for modeling lithium-ion battery capacity fade," in *Proc. Annu. Rel. Maintainability Symp. (RAMS)*, Palm Springs, CA, USA, Jan. 2020, pp. 1–7.

[3] J. Lee and M. Mitici, "Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics," *Rel. Eng. Syst. Saf.*, vol. 230, Feb. 2023, Art. no. 108908.

[4] D. Wu, Y. Lei, M. He, C. Zhang, and L. Ji, "Deep reinforcement learning-based path control and optimization for unmanned ships," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–8, May 2022. [Online]. Available: https://downloads.hindawi.com/journals/wcmc/2022/7135043.pdf

[5] H. M. Schwartz, "Learning in multiplayer stochastic games," in *Multi-Agent Machine Learning: A Reinforcement Approach*. New York, NY, USA: Wiley, 2014, pp. 73–143.

[6] R. B. L. Busoniu and B. D. Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., C*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[7] H. Wang, X. Wang, X. Hu, X. Zhang, and M. Gu, "A multi-agent reinforcement learning approach to dynamic service composition," *Inf. Sci.*, vol. 363, pp. 96–119, Oct. 2016.

[8] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li, "CityFlow: A multi-agent reinforcement learning environment forlarge scale city trafficscenario," in *Proc. World Wide Web Conf.*, vol. 90, 2019, pp. 3620–3624.

[9] I. Xiao, "A distributed reinforcement learning solution with knowledge transfer capability for a bike rebalancing problem," 2018, *arXiv:1810.04058*.

[10] S. Y. Luis, D. G. Reina, and S. L. T. Marín, "A multiagent deep reinforcement learning approach for path planning in autonomous surface vehicles: The Ypacaraí lake patrolling case," *IEEE Access*, vol. 9, pp. 17084–17099, 2021.

[11] M. L. Littman, "Value-function reinforcement learning in Markov games," *Cognit. Syst. Res.*, vol. 2, no. 1, pp. 55–66, Apr. 2001.

[12] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. AAAI*, Menlo Park, CA, USA, 1998, pp. 746–752.

[13] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team Markov games," in *Proc. NIPS*, Vancouver, BC, Canada, 2002, pp. 1571–1578.

[14] Z. Zhang, D. Wang, D. Zhao, Q. Han, and T. Song, "A gradient-based reinforcement learning algorithm for multiple cooperative agents," *IEEE Access*, vol. 6, pp. 70223–70235, 2018.

[15] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. ICML*, Amherst, MA, USA, 1993, pp. 330–337.

[16] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers, "Evolutionary dynamics of multi-agent learning: A survey," *J. Artif. Intell. Res.*, vol. 53, pp. 659–697, Aug. 2015.

[17] A. Kianercy and A. Galstyan, "Dynamics of Boltzmann Q-learning in two-player two-action games," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 85, no. 4, pp. 1115–1154, Apr. 2012.

[18] M. Babes, M. Wunder, and M. L. Littman, "Dynamics of Boltzmann Q-learning in two-agent two-action games," in *Proc. AAMAS*, Budapest, Hungary, 2009, pp. 659–697.

[19] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in *Proc. AAAI*, Edmonton, AB, Canada, 2002, pp. 326–331.

[20] Z. Zhang, D. Zhao, J. Gao, D. Wang, and Y. Dai, "FMRQ—A multiagent reinforcement learning algorithm for fully cooperative tasks," *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1367–1379, Jun. 2017.

[21] Z. Zhang and D. Wang, "EAQR: A multiagent Q-learning algorithm for coordination of multiple agents," *Complexity*, vol. 2018, pp. 1–14, Aug. 2018.

[22] L. Matignon and G. J. Laurent. (2009). *Coordination of Independent Learners in Cooperative Markov Games*. [Online]. Available: http://hal.archives-ouvertes.fr/docs/00/37/08/89/PDF/Rapport-1.pdf

[23] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 33, no. 6, pp. 750–797, Nov. 2019.

[24] R. Lowe, Y. Wu, A. Tamar, and J. Harb, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[25] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proc. AAAI*, San Francisco, CA, USA, 2017, pp. 2974–2982.

[26] T. Rashid, M. Samvelyan, C. D. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. ICML*, Stockholm, Sweden, 2018, pp. 4292–4301.

[27] P. Sunehag, G. Lever, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, and M. Lanctot, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Conf. Autonomous Agents Multiagent Syst.*, 2018, pp. 2085–2087.
[28] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. ICML*, Long Beach, CA, USA., 2019, pp. 5887–5896.
[29] F. Gul, A. Mir, I. Mir, S. Mir, T. U. Islaam, L. Abualigah, and A. Forestiero, "A centralized strategy for multi-agent exploration," *IEEE Access*, vol. 10, pp. 126871–126884, 2022.
[30] A. Forestiero and C. Mastroianni, "A swarm algorithm for a self-structured P2P information system," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 681–694, Aug. 2009.
[31] R. E. Bellman and S. E. Dreyfus, "Applied dynamic programming," in *RAND Corp.* Princeton, NJ, USA: Princeton Univ. Press, 1962, pp. 750–797.
[32] S. Ross, *Introduction to Stochastic Dynamic Programming*. New York, NY, USA: Academic Press, 1983.
[33] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. thesis, King's College, Cambridge Univ., Cambridge, U.K., 1989.
[34] D. Liao, Z. Zhang, and D. Zhao, "Robot cooperative handling method based on multi-agent deep reinforcement learning," *Electron. Des. Eng.*, vol. 31, no. 23, pp. 7–11, 2023. [Online]. Available: http://kns.cnki.net/kcms/detail/61.1477.TN.20230406.1755.002.html
[35] S. G. E. Jang and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. ICLR*, Toulon, France., 2017.
[36] S. Ali, S. Koenig, and M. Tambe, "Preprocessing techniques for accelerating the DCOP algorithm ADOPT," in *Proc. 4th Int. Joint Conf. Auto. Agents Multiagent Syst.*, Utrecht, The Netherlands, Jul. 2005, pp. 1041–1048.
[37] R. B. L. Busoniu and B. D. Schutter, "Multi-agent reinforcement learning: An overview," in *Innovations in Multi-Agent Systems and Applications*. Berlin, Germany: Springer, 2010, pp. 183–221.

DENGYU LIAO received the B.S. degree from the School of Information and Electrical Engineering, Shandong Jianzhu University, Jinan, China, in 2020. She is currently pursuing the M.S. degree with the School of Automation, Qingdao University, Qingdao, China. Her research interests include reinforcement learning and multi-agent reinforcement learning.

ZHEN ZHANG received the B.S. degree from the China University of Petroleum, Dongying, China, in 2006, the M.S. degree in control theory and control engineering from the Dalian University of Technology, Dalian, China, in 2009, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, in 2013. He has been with the School of Automation, Qingdao University, Qingdao, China, since 2013. He was a Visiting Scholar with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, from August 2018 to February 2019. He has been an Associate Professor with the School of Automation, Qingdao University, since November 2019. His current research interests include reinforcement learning and intelligent optimization methods.

TINGTING SONG received the B.S. and M.S. degrees from the School of Automation, Qingdao University, Qingdao, China, in 2015 and 2018, respectively. She joined Qingdao Metro Group Company Ltd., Operating Branch, Qingdao, in 2018. Her research interests include reinforcement learning and deep learning.

MINGYANG LIU received the B.S. degree from the School of Automation and Electrical Engineering, Linyi University, Linyi, China, in 2021. He is currently pursuing the M.S. degree with the School of Automation, Qingdao University, Qingdao, China. His research interests include transfer learning and reinforcement learning.