Reinforcement Learning Based Multi-Agent Resilient Control: From Deep Neural Networks to an Adaptive Law

Jian Hou[1], **Fangyuan Wang**[1], **Lili Wang**[2*], **Zhiyong Chen**[3]

[1] School of Information Science & Technology, Zhejiang Sci-Tech University, Hangzhou, China, 310018
[2*] Corresponding Author, Division of Systems Engineering, Boston University, Boston, MA, USA, 02215
[3] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW, 2308
changeleap@163.com, wfy11235813@gmail.com, lili.wang.zj@gmail.com, zhiyong.chen@newcastle.edu.au

Abstract

Recent advances in Multi-agent Reinforcement Learning (MARL) have made it possible to implement various tasks in cooperative as well as competitive scenarios through trial and error, and deep neural networks. These successes motivate us to bring the mechanism of MARL into the Multi-agent Resilient Consensus (MARC) problem that studies the consensus problem in a network of agents with faulty ones. Relying on the natural characteristics of the system goal, the key component in MARL, reward function, can thus be directly constructed via the relative distance among agents. Firstly, we apply Deep Deterministic Policy Gradient (DDPG) on each single agent to train and learn adjacent weights of neighboring agents in a distributed manner, that we call Distributed-DDPG (D-DDPG), so as to minimize the weights from suspicious agents and eliminate the corresponding influences. Secondly, to get rid of neural networks and their time-consuming training process, a Q-learning based algorithm, called Q-consensus, is further presented by building a proper reward function and a credibility function for each pair of neighboring agents so that the adjacent weights can update in an adaptive way. The experimental results indicate that both algorithms perform well with appearance of constant and/or random faulty agents, yet the Q-consensus algorithm outperforms the faulty ones running D-DDPG. Compared to the traditional resilient consensus strategies, e.g., Weighted-Mean-Subsequence-Reduced (W-MSR) or trustworthiness analysis, the proposed Q-consensus algorithm has greatly relaxed the topology requirements, as well as reduced the storage and computation loads. Finally, a smart-car hardware platform consisting of six vehicles is used to verify the effectiveness of the Q-consensus algorithm by achieving resilient velocity synchronization.

Introduction

In recent years, multi-agent Cyber-Physical Systems (CPSs) are ubiquitous in modern infrastructure systems, and attract increasing attention, for instance, in transportation systems, electrical power grids, wireless communication networks, health-care devices and so on (Humayed et al. 2017). The unpredictable and possibly hostile running environments and internal agent faults have brought many challenges, among which fault-tolerance resilient consensus, i.e., reaching global agreement with only local but possibly false data injection interactions, is one of the fundamental concerns.

In the closely related field of Multi-agent Reinforcement Learning (MARL), each agent interacts with the environment and other agents by selecting proper actions to maximize the total rewards. Recent advances in MARL have made it possible to implement various tasks. It therefore seems quite natural to ask whether the MARL techniques can also be beneficial for Multi-agent Resilient Consensus (MARC) problem.

However, MARC presents several challenges from an MARL perspective. Firstly, most successful MARL applications to date adopt the framework of centralized training with decentralized execution, while in MARC, only the local information from neighboring agents in a distributed manner is available for a global target. Secondly, the identities of all the agents as either teammate or opponent are known *a prior* for each agent in MARL such that properly-designed strategies can be implemented. Nevertheless, the main objective in MARC is to distinguish the identities of all the agents. In addition, the unpredictable behaviors of faulty agents may imply non-stationary environment and violate the Markovian assumptions, that is a frequent but tough problem in MARL.

This paper adopts the idea of trial and error from MARL as well as the natural characteristics of the MARC system for consensus to overcome these challenges, aiming at adjusting the adjacent weights, and hence mitigating the influences of the faulty nodes. Firstly, the MARC problem is transformed into a typical MARL problem in which each agent executes DDPG strategy individually to avoid the centralized training, that we call Distributed-DDPG (D-DDPG). By means of consensus for the final objective, a reward function can be well designed via relative state difference, so that teammates and opponents are gradually identified. After the training process for actor and critic networks is completed, MARC can be achieved in just few steps. Secondly, to get rid of neural networks and their time-consuming training process, a Q-learning based algorithm, named Q-consensus, is presented by building a proper reward function and a credibility function for each pair of neighboring agents. And thus, each node (agent) updates its adjacent weights in an adaptive way. Lastly, we apply the proposed Q-consensus algorithm to achieve resilient velocity synchronization in a smart-car

hardware platform. The contributions are listed as follows:

(i) We build the bridge between MARL and MARC, by applying DDPG on each single agent to learn adjacent weights of neighboring agents aiming at reaching resilient consensus (RC) for normal agents in a distributed manner. After neural networks training is completed, MARC can be realized in just few steps regardless of constant and/or random faulty agents.

(ii) To get rid of neural networks and their time-consuming training process, a Q-learning based algorithm, called Q-consensus, is presented so that the adjacent weights can update in an adaptive way through a properly-designed reward function and a credibility function. The proposed Q-consensus algorithm is resistant to faulty agents with constant, random and/or capability of executing D-DDPG.

(iii) In the existing strategies, either tough conditions on topology, e.g., F-total fault model (up to F faulty nodes), F-local fault model (up to F faulty incoming neighbors for each normal node), network connectivity (no less than $2F + 1$), or a mass of historical information and computation loads, is demanded. The proposed Q-consensus algorithm has greatly relaxed the network topology, as well as storage and computation loads.

Related Work

In CPSs, a straightforward method of achieving MARC is to remove the suspicious agents. Generally speaking, this operation can be implemented either at each time temporarily or permanently. By supposing the maximum number F of faulty agents in a network is known *a priori*, each agent just abandons its F largest values as well as the F smallest values among the received information from the neighboring agents temporarily. This scheme is the so-called Mean-Subsequence-Reduced (MSR) algorithm (LeBlanc and Koutsoukos 2011), and then the Weighted-Mean-Subsequence-Reduced (W-MSR) algorithm (LeBlanc and Koutsoukos 2012). It has been further adopted in various MARC applications, including clock synchronization (Kikuya, Dibaji, and Ishii 2017), spacecraft control (Wang et al. 2019), etc. However, it is constrained to network connectivity (Sundaram and Hadjicostis 2011) or network robustness (Zhang and Sundaram 2012), with implementation in practice hardly realizable.

Another common technique is to evaluate the trustworthiness on each agent by the historical data, and then the most suspicious agents are cast away permanently. The typical algorithm called RoboTrust (Mikulski et al. 2012) calculates the trustworthiness by observations and statistical inferences from various historical perspectives, and only the most trustworthy agents are kept for synchronization. This algorithm is then extended to the second-hand evidence in (Liu and Baras 2014). The work in (Baras and Liu 2019) further proposes various local decision rules by local evidence to enhance the robustness of trust evaluation. This trustworthiness based RC strategy avoids the restriction of network connectivity or network robustness, yet raises the storage and computation burden. Meanwhile, the cooperation among faulty agents is not involved.

As a most popular method recently, reinforcement learning (RL) is also applied to solve optimal consensus problem with unknown dynamics, as for the aspect of data-driven. Through establishing performance index via Bellman's principle of optimality and importing neural networks to approximate, the solution of the coupled Hamilton-Jacobi-Bellman (HJB) equation is no longer required, and instead, policy gradient or Q-learning methods are developed to address the optimal control problems in real-time (Zhang et al. 2017; Mu et al. 2019; Peng et al. 2020). As far as we know, the work in (Hou et al. 2020) is the first one that brings the idea of trial and error in MARL into MARC. In (Hou et al. 2020), the adjacent weight between any two neighboring agents is dynamically updated by the credibility measurement, that is determined by the relative state difference at the present step and the credibility value previously. Both fixed and stochastic typologies, as well as the application in clock synchronization are provided to illustrate the effectiveness. Nevertheless, the inner relationship and connection between MARC and MARL are still beyond comprehension.

MARL, as an area of machine learning, derives from RL where an agent learns a proper policy by interacting with a dynamic environment, by extending to multiple agents. The research interest in MARL has been evidenced from single agent games (Mnih et al. 2015), two-player games (Silver et al. 2017; Brown and Sandholm 2018), to scenarios including cooperative as well as competitive agents (Vinyals et al. 2019). One straightforward method is to implement directly the single agent RL algorithm on multi-agent environment. The work in (Tampuu et al. 2017) employs two independent deep Q-networks (DQNs) to train the Atari Pong game, that results in competitive, cooperative, or progression from competitive to collaborative emergent behaviors. The sequential social dilemmas are discussed by each single independent DQN on each agent in (Leibo et al. 2017). However, most results that are obtained by applying an independent RL algorithm directly on multi-agent scenarios are technically straightforward and even infeasible, due to the non-stationary environment and curse of dimensionality (Hernandezleal, Kartal, and Taylor 2019).

In response to the non-stationary environment problem in MARL, Lowe et al. (Lowe et al. 2017) extended DDPG to multi-agent setting (i.e., MADDPG) with the framework of centralized training and decentralized execution. And thus, a general purpose multi-agent learning algorithm is proposed applicable in both cooperative and competitive scenarios. Besides, a training regimen utilizing an ensemble of policies is considered to produce more robust multi-agent policies. This strategy is further extended as Memory-Driven MADDPG (Pesce and Montana 2020) by introducing a shared memory so that each agent's policy depends on both its own observation and its interpretation of the collective memory. Two related algorithms, MiniMax MADDPG (Li et al. 2019) and Message-Dropout MADDPG (Kim, Cho, and Sung 2019) are proposed for robust policy learning so that the agents can still generalize as the opponents' policies change, where the former leverages the minimax concept and produces a minimax learning objective, and the latter takes the dropout technique into account during the training time.

Nowadays, most MARL algorithms are still applied in games, while recent advances begin to give a fruitful sight in various of applications such as traffic lights control(Calvo and Dusparic 2018; Chu et al. 2019), resource management(Noureddine, Gharbi, and Ahmed 2017; Naderializadeh et al. 2020), internet of vehicle(Bacchiani, Molinari, and Patander 2019; Kwon and Kim 2019), electrical power control(Sharma et al. 2019) and recommendation systems(Gui et al. 2019).

Problem Formulation

The MARC problem is studied in this paper. The problem starts with a network consisting of n agents labeled by $1,\ 2,\ \ldots, n$. Normally, the network relation can be represented by a directed graph, $G = (V, E)$, where $V = \{1, 2, \cdots, n\}$ corresponds to the set of agents, and $E \subseteq V \times V$ is the set of edges which describes the neighbor relations. Agent j is a neighbor of agent i, with an edge from node j to node i, if agent i can receive information from agent j. The neighbor set of node i is presented by $N_i = \{j \mid (j, i) \in E\}$. A path from node j to node i is a sequence of distinct nodes $i_0, i_1, \cdots, i_m$, where $i_0 = j$ and $i_m = i$ with $(i_l, i_{l+1}) \in E, 0 \leq l \leq m-1$. A directed graph is called rooted if and only if there exists $i \in V$, so that there is a path from node i to any other node. We use the terms node and agent, network and topology, interchangeably.

A discrete-time model in the following form is considered

$$x_{i,k+1} = x_{i,k} + u_{i,k},\ i \in V \tag{1}$$

where $x_{i,k} \in \mathbb{R}$ and $u_{i,k} \in \mathbb{R}$ denote the state and control input of agent i at time k, respectively. Let $\mathbf{x}_k = [x_{1,k}, \cdots, x_{n,k}]^T$ for any integer $k \geq 0$. The system initial state $\mathbf{x}_0$ is arbitrarily specified.

In the multi-agent consensus problem, all the nodes are trustworthy and cooperating with each other to achieve state consensus (Hou and Zheng 2018), through a common design $u_{i,k}$. In the MARC problem, the concealment of certain faulty nodes may be involved against the system convergence. This paper studies the MARC problem by introducing four types of nodes, including the set of normal nodes V^{n}, the set of persistent faulty nodes V^{p}, the set of constant faulty nodes V^{c}, and the set of learnable faulty nodes V^{l}, with

$$V = V^{\mathrm{n}} \cup V^{\mathrm{p}} \cup V^{\mathrm{c}} \cup V^{\mathrm{l}}.$$

The appearance of faulty nodes sets V^{p}, V^{c} and V^{l} may be produced by system failure or deliberately hostile injection. In the following, we will give the detailed description for each type of nodes.

(i) **Normal Node**: A normal node updates its control input by

$$u_{i,k} = \sum_{j \in N_i} \alpha_{ij,k}(x_{j,k} - x_{i,k}) + \omega_{i,k},\ i \in V^{\mathrm{n}}, \tag{2}$$

where $\alpha_{ij,k} (\geq 0)$ is called the adjacent weight from node j to node i satisfying $\sum_{j \in N_i} \alpha_{ij,k} < 1$, and $\omega_{i,k} \in \mathbb{R}$ is a bounded noise ($|\omega_{i,k}| < \omega$) induced by transmission channel and environment.

(ii) **Persistent Faulty Node (PFN)**: A PFN conducts its input with a random value at every time as follows

$$u_{i,k} = \mathrm{Random},\ i \in V^{\mathrm{p}}, \tag{3}$$

where Random is a random variable that has a specified probability density function f_{Random}.

(iii) **Constant Faulty Node (CFN)**: A CFN holds its input as the initial value as follows

$$u_{i,k} = 0,\ i \in V^{\mathrm{c}}. \tag{4}$$

(iv) **Learnable Faulty Node (LFN)**: All the LFNs may work together by applying machine learning algorithms such as MADDPG to prevent normal nodes from achieving consensus.

With the existence of faulty nodes, the whole network may not achieve consensus. A straightforward method is to isolate suspicious nodes by minimizing their weights to be close to zero. In another word, we expect to design a distributed algorithm to guarantee that the sub-network of normal nodes achieve consensus in the following sense

$$\limsup_{k \to \infty} \max_{i,j \in V^{\mathrm{n}}} |x_{i,k} - x_{j,k}| < \varepsilon \tag{5}$$

for a sufficiently small ε.

The achievement of (5) for the system (1) in the presence of faulty nodes is called MARC in this paper. So, the main *objective* is to propose a distributed algorithm for updating adjacent weights to achieve MARC.

Throughout the paper, we give the following necessary assumption.

Assumption 0.1 *The topology (network) among the normal nodes is fixed and rooted.*

Distributed DDPG Based MARC

The recent development of the MARL technique, especially by feeding sufficient training data and introducing deep neural networks, contributes to the optimization problem of maximizing total return. It therefore motivates us to combine the MARL configuration into the MARC problem aiming at reducing the adjacent weights from suspicious nodes by using stochastic gradient updates.

For each node, we build and apply the DDPG strategy to update all the related adjacent weights. For each node i, $s_{i,k} = \{x_{i,k}, x_{j,k}, \forall j \in N_i\}$ is considered as the system state observed by agent i, and all the related adjacent weights set $a_{i,k} = \{\alpha_{ij,k}, \forall j \in N_i\}$ is defined as the corresponding action. We define the reward function as $r_{i,k} = f(\sum_{j \in N_i} \alpha_{ij,k} |x_{j,k} - x_{i,k}|)$. In general, the closer states for two normal neighboring agents indicate a larger reward value, and thus the reward function should be inversely proportional to the relative distance $|x_{j,k} - x_{i,k}|$. The detailed setting is given in the experiment section. Similarly, we also construct an experience replay buffer R_i to store agent i's experiences at each time, i.e., transition $(s_{i,k}, a_{i,k}, r_{i,k}, s_{i,k+1})$. And a random minibatch of transitions is sampled from the pool to update the corresponding critic and actor networks. After performing experience replay, the agent executes the action according to the current policy and exploration noise.

The complete algorithm, which we call the D-DDPG Algorithm, is presented in Algorithm 1. It is worth mentioning that each node only requires information from its neighbors, so that it is fundamentally a distributed algorithm, and centralized training is avoided. This algorithm is a model-free off-policy algorithm for continuous actions learning.

Q-consensus Based MARC

Fundamentally, the RL based approach is relatively time-consuming and computing-heavy, in particular when the action space is large, that brings long term trial and error episodes. This section introduces the methodology of Q-learning to identify and isolate the faulty nodes in the MARC problem. The main distinction is embodied in the relation characterization we present between a pair of neighboring agents instead of DQN for a single agent.

When agent i receives x_j from its neighbor $j \in N_i$, the relation characterization from j to i is built. First, the *immediate reward* r_{ij} is calculated with the input $x_j - x_i$. In the next place, the *credibility* Q_{ij} is defined based on the immediate reward. And finally, the adjacent weights α_{ij} could be updated subsequently. In the MARC system, the basic rule is that for neighbors with a higher credibility, the adjacent weights are larger, and for any suspicious neighbors with a lower credibility, the adjacent weights $\alpha_{ij,k} \to 0$ as $k \to \infty$. The details on how to get the reward and the credibility will be discussed in the following.

First, a crisp input of the relative distance $x_j - x_i$ is introduced describing the state differences at the current instant. This input is then converted into an immediate reward by

$$r_{ij,k} = f(|x_{j,k} - x_{i,k} + \omega_{ij,k}|),\ \forall j \in N_i$$

where ω_{ij} is the noise on the transmission channel from j to i which is bounded by a positive number ω. Similar to the definition in D-DDPG, each normal node in general recognizes its own state x_i as the true value and the reward value should be inversely proportional to relative distance, e.g.,

$$f(|x_{j,k} - x_{i,k} + \omega_{ij,k}|) = e^{-|x_{j,k} - x_{i,k} + \omega_{ij,k}|\beta_k},$$

with an appropriately designed parameter $\beta_k > 0$. It is noticed that the reward value is bounded between 0 and 1. When the states x_i and x_j are close, the induced reward is close to 1; when they are sufficiently different, the value tends to 0.

Another key element, named credibility Q_{ij}, is initialized as 1 for each neighboring node $j \in N_i$, and then recursively updated according to the associated reward r_{ij} by

$$Q_{ij,k} = Q_{ij,k-1} + \eta_k(r_{ij,k} - Q_{ij,k-1}),\ \forall j \in N_i \qquad (6)$$

that essentially integrates all the historical information from node j to node i. The parameter η_k is a step size that is usually monotonously decreasing and trends to 0 gradually. Compared to the classic Q-learning equation, the TD target component $r + \gamma \max_{a'} Q(s', a')$ is replaced by the reward value $r_{ij,k}$, as no state of next step is involved here. Likewise, the credibility is expected to move towards the reward value to achieve stability.

Since the state x_i is determined by the adjacent weights and the states of neighboring nodes as (2), the last step is to update adjacent weights by credibility. By normalization of the credibility of all neighboring nodes, we define the weight α_{ij} as

$$\alpha_{ij,k} = \frac{Q_{ij,k}}{\sum_{j \in N_i} Q_{ij,k}}(1 - \frac{1}{|N_i| + 1}),\ \forall j \in N_i$$

where $|N_i|$ is the cardinality of N_i. The design obviously satisfies $\sum_{j \in N_i} \alpha_{ij,k} < 1$, and $\alpha_{ii,k} = \frac{1}{|N_i|+1}$ for all k.

The detailed algorithm is summarized in Algorithm 2 as follows.

Remark 0.1 *Regarding the proposed two algorithms, D-DDPG solves the MARC problem directly in an MARL configuration, and thus the DDPG strategy, especially deep neural networks are utilized to train and learn adjacent weights for each single agent. As far as the time-consuming training process with deep neural networks is concerned, an adaptive law is introduced such that for each agent i and its neighboring agent j, the adjacent weight α_{ij} is determined mainly by credibility Q_{ij}, which contains sufficient information to identify the neighboring agents. The update of the credibility Q_{ij} is also adaptive by the modified Q-learning equation* (6).

Experimental Results

In this section, we apply the proposed algorithms to solve the MARC problem in both numerical simulations and hardware experiments.

Resilient Consensus

A fixed directed network is given in Fig. 1 in which 0, 1 and 2 are faulty nodes, and all the rest normal nodes are rooted. In the simulation setting, each of the agents has an arbitrarily initial state between 0 and 1. The noise upper bound is $\omega = 0.01$. It can be observed that neither the network connectivity (Sundaram and Hadjicostis 2011) nor network robustness (Zhang and Sundaram 2012) is satisfied so that the traditional W-MSR algorithms are not available. In the following, we apply both D-DDPG and Q-consensus algorithms on this network, respectively, to check if normal agents can achieve RC.

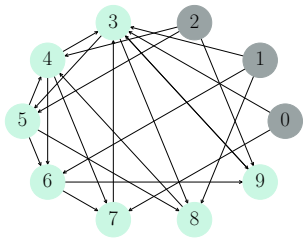

Figure 1: A fixed directed network where 0, 1 and 2 are faulty nodes, and the rest normal nodes are rooted.

D-DDPG We first give the architecture of the corresponding neural network in D-DDPG.

Architectures : Take the actor network as example, and critic network is similar except the dimensions of input

Algorithm 1: Distributed DDPG for a normal node i

Randomly initialize critic network $Q_i(s_i, a_i|\theta^{Q_i})$ and actor network $\mu_i(s_i|\theta^{\mu_i})$ with weights θ^{Q_i} and θ^{μ_i}.
Initialize target network Q_i' and μ_i' with weights $\theta^{Q_i'} \leftarrow \theta^{Q_i}$, $\theta^{\mu_i'} \leftarrow \theta^{\mu_i}$.
Initialize replay buffer R_i.
for *episode = 1 to M* **do**
  Initialize a random process $\mathcal{N}_i$ for action exploration.
  Receive initial observation state $s_{i,1}$.
  for k= *1 to T* **do**
    Select action $a_{i,k} = \mu_i(s_{i,k}|\theta^{\mu_i}) + \mathcal{N}_{i,k}$ according to the current policy and exploration noise.
    Execute action $a_{i,k}$ and observe reward $r_{i,k}$ and new state $s_{i,k+1}$.
    Store transition $(s_{i,k}, a_{i,k}, r_{i,k}, s_{i,k+1})$ in R_i.
    Sample a random minibatch of N transitions $(s_{i,l}, a_{i,l}, r_{i,l}, s_{i,l+1})$ from R_i.
    Set $y_{i,l} = r_{i,l} + \gamma Q_i'\left(s_{i,l+1} \mid \theta^{\mu_i'}\right) \mid \theta^{Q_i'}\Big)$.
    Update critic by minimizing the loss: $L_i = \frac{1}{N}\sum_l \left(y_{i,l} - Q_i\left(s_{i,l}, a_{i,l} \mid \theta^{Q_i}\right)\right)^2$.
    Update actor policy using the sampled policy gradient:

$$\nabla_{\theta^{\mu_i}} J_i \approx \frac{1}{N}\sum_l \nabla_{a_i} Q_i\left(s_i, a_i \mid \theta^{Q_i}\right)|_{s_i=s_{i,l}, a_i=\mu_i(s_{i,l})} \nabla_{\theta^{\mu_i}} \mu_i\left(s_i \mid \theta^{\mu_i}\right)|_{s_{i,l}}$$

    Update the target networks:

$$\theta^{Q_i'} \leftarrow \tau\theta^{Q_i} + (1-\tau)\theta^{Q_i'}$$
$$\theta^{\mu_i'} \leftarrow \tau\theta^{\mu_i} + (1-\tau)\theta^{\mu_i'}$$

  end for
end for

Algorithm 2: Q-consensus for a normal node i

Initialize $\alpha_{ii,0} = \alpha_{ij,0} = \frac{1}{|N_i|+1}$,$Q_{ij,0} = 1, j \in N_i$
for $k = 1$ *to* T **do**
  for $j \in N_i$ **do**
    $r_{ij,k} = f(|x_{j,k} - x_{i,k} + \omega_{ij,k}|)$;
    $Q_{ij,k} = Q_{ij,k-1} + \eta_k(r_{ij,k} - Q_{ij,k-1})$;
  end for
  for $j \in N_i$ **do**
    $\alpha_{ij,k} = \frac{Q_{ij,k}}{\sum_{j\in N_i} Q_{ij,k}}(1 - \frac{1}{|N_i|+1})$;
  end for
  $x_{i,k+1} = x_{i,k} + \sum_{j\in N_i} \alpha_{ij,k}(x_{j,k} - x_{i,k}) + \omega_{i,k}$.
end for

layer and output layer. The neural network consists of 4 layers, including an input layer, two hidden layers and an output layer. In which, the non-output layers use the ReLU activation function, and the output layer utilizes the Tanh activation function so that the output is limited between $(-1, 1)$, and further mapped to $(0, 1)$. It is noted that the output layer of the critic network contains no activation function. In addition, we set replay buffer size as 10000, batch size as 64, discount factor $\gamma = 0.9$, and $\tau = 0.9$. Both the actor learning rate and the critic learning rate are 0.0001.

A training process with 2 random faulty nodes and 1 constant faulty node is shown in Fig. 2 where the adjacent weights are either minimized to 0 or maximized. The reward function in this experiment is defined as $100 \cdot [e^{-20\cdot\sum_{j\in N_i}|x_{j,k}-x_{i,k}+\omega_{ij,k}|\alpha_{ij,k}} - 0.5]$ and all the states are initialized for every 100 steps to accelerate convergence. It is worth mentioning that convergence to the constant node also satisfies the consensus requirement (5) so that the adjacent weights from the constant faulty node may not tend to 0. After the training process is completed, the state evolution of normal nodes is presented in Fig. 3 showing that RC in all four cases, i.e., three random faulty nodes, two random nodes and one constant node, one random node and two constant nodes, and three constant nodes, is achieved in a few steps (the training process for the rest three cases are omitted here).

Q-consensus Similarly, we apply the Q-consensus algorithm on the network shown in Fig. 1. The parameter β_k is set to be $10 + 0.1k$, and η_k is chosen to be 0.01 if $k < 0.8T$ else $\eta_{k+1} = \eta_k - 0.05/T$. The simulation results are presented in Fig. 4 indicating that all the states of normal nodes in the aforementioned four cases can achieve RC.

To show the reliability of the two algorithms, both simulations are run for 1000 times with random initial states. The success rates in terms of the times of consensus achieved by normal nodes of the two algorithms are shown in Fig. 5. It is observed that both algorithms perform well with high success rates.

Furthermore, we consider a scenario of learnable faulty nodes from V^1 executing D-DDPG to interfere system convergence. When all the nodes apply D-DDPG, the reward functions of normal nodes and faulty nodes are defined as $100 \cdot [e^{-20\cdot\sum_{j\in N_i}|x_{j,k}-x_{i,k}+\omega_{ij,k}|\alpha_{ij,k}} - 0.5]$ and $100 \cdot [0.5 -$

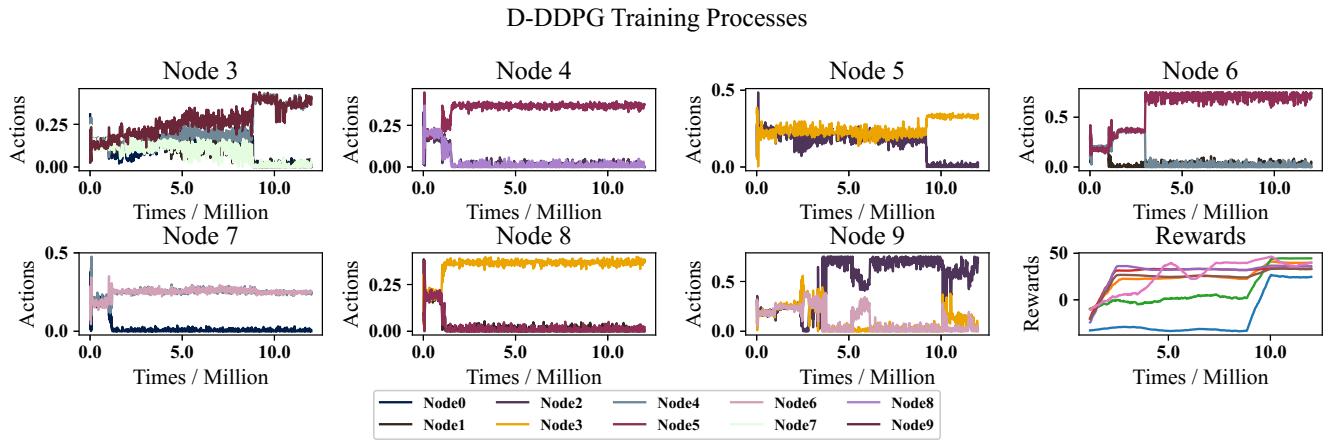

Figure 2: The actions and rewards training for normal nodes under D-DDPG.

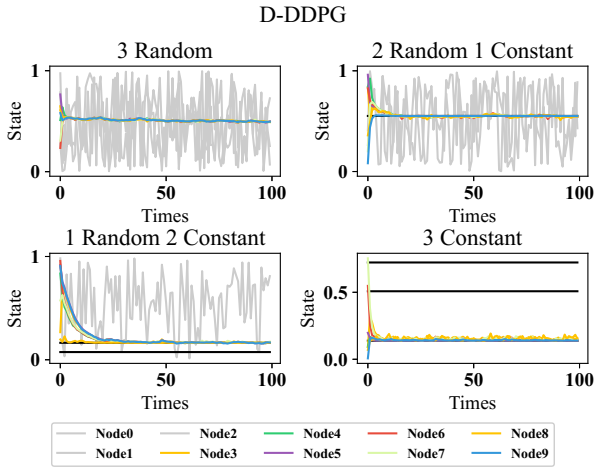

Figure 3: The states of normal nodes under D-DDPG with constant and/or random faulty nodes.

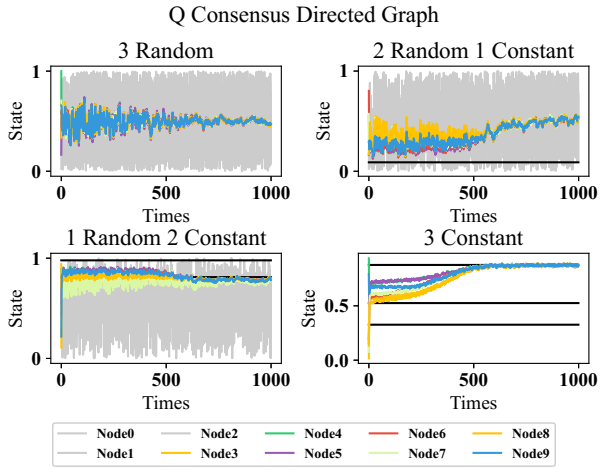

Figure 4: The states of normal states under Q-consensus with constant and/or random faulty nodes.

	D-DDPG	Q-Consensus
Topology requirement	sparse	none
Faulty types	invariant	none
Convergence time	long training	short updating

Table 1: Comparisons of the Two Algorithms

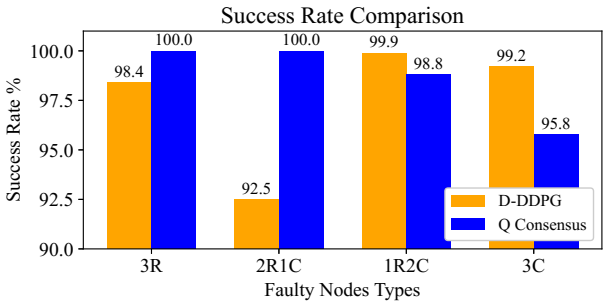

Figure 5: Comparison of success rates of the two proposed algorithms with constant and/or random faulty nodes.

$e^{-20\cdot\sum_{i,j\in V^{\mathrm{n}}}|x_{j,k}-x_{i,k}+\omega_{ij,k}|}$], respectively. The actual input of a learnable normal node is its own state and the states of its neighbors, and the actual input of a learnable faulty node is the states of all the normal nodes. The result in Fig. 6 indicates that RC may not be achieved. Nevertheless, RC can be easily achieved if normal nodes apply Q-consensus algorithm, as a result of much faster convergence time of Q-consensus than training time in D-DDPG. The state evolution is similar to Fig. 4 and we omit it here.

Comparison of the proposed two algorithms is also shown below as Table 1. Owing to the introduction of deep neural networks, it takes a long time to finish the training process for the D-DDPG algorithm, while convergence in the Q-consensus algorithm is much faster as no training process is included. Moreover, D-DDPG is relatively computationally heavy, especially when the topology is not sparse in a large action space. However, as long as the topology for nor-

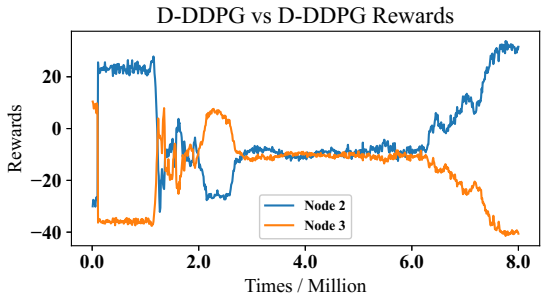

Figure 6: The rewards of a normal node and a faulty node with D-DDPG.

mal nodes is rooted, RC is easily achieved for Q-consensus no matter how complex the topology is. This advantage outperforms trustworthiness based methods, so that it can be applied in large-scale networks with dense topology connections. We omit the corresponding comparison simulations due to the space limit. On the other aspect, Q-consensus also works when the faulty nodes are variable, e.g., either constant or random at different instant, while D-DDPG only works for the fixed types of faulty nodes due to the existence of training process.

Resilient Velocity Synchronization

Lastly, we apply the Q-consensus algorithm in a smart-car hardware platform to achieve resilient velocity synchronization. The platform consists of six Corey Mecanum wheel vehicles labeled by $0,\ 1,\ \ldots, 5$, and each vehicle is equipped with Mecanum wheels, so that it can achieve omnidirectional movement. We use OptiTrack to track and control the movement of the vehicles.

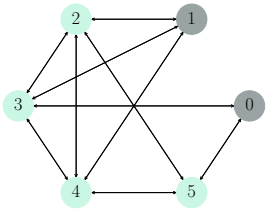

Figure 7: The communication topology of 6 vehicles.

In the experiment, we use a central computer to collect the locations of all vehicles in real time, while each vehicle only utilizes the information of its own and its neighbors to control the movement for the purpose of distributed and local implementation of the algorithm. The moving direction and speed are quantized from the continuous control signal calculated by the proposed control law. The topology of the six vehicles is shown in Fig. 7 where node 0 and node 1 are two faulty nodes with random control, and nodes 2-5 are normal ones.

The actual speeds of all the vehicles are set to be 0.2m/s, while the initial positions and velocity directions are randomly produced as shown in Fig. 8. The experimental result shows that the velocities of the normal nodes are synchronized, that is, all the vehicles move towards the same direction with the same speed. Fig. 9 is one snapshot of the experiment when the velocities of the vehicles are synchronized.

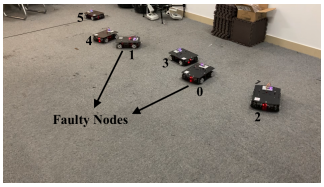

Figure 8: The initial states of the vehicles.

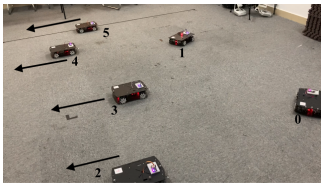

Figure 9: The normal vehicles gradually achieve synchronized velocity.

Conclusions

In this paper, we have introduced the mechanism of MARL to solve the MARC problem through establishing connections between the two concepts. In order to obtain continuous adjacent weights, a policy based strategy, D-DDPG is brought to train and learn via deep neural networks for each single agent in a distributed manner. The reward function is defined by the relative state difference so that the maximized total reward is equivalent to the system goal of consensus. After neural networks training is completed, the weights from suspicious agents are minimized and the corresponding influences eliminated. However, the training in RL is relatively time-consuming, especially in a large action space. On the experience of Q-learning, an adaptive law, called Q-consensus, has been proposed through building a proper reward function and a credibility function.

Three types of faulty nodes were considered in this paper, including persistent ones, constant ones, and learnable ones, where both D-DDPG and Q-consensus have performed well in the presence of the former two, yet only the Q-consensus algorithm has worked well with the learnable faulty nodes. In addition, the proposed Q-consensus has a relative low requirement on topology, as well as reduces the storage and computation loads so that it can be applied in more general applications. Both numerical simulations and hardware platform have been provided to validate the effectiveness.

This paper has considered a fixed topology without time-delay. In practical environment, the topology is commonly time-varying due to newly increased/removed nodes, variable distance among nodes, or package loss during transmission. In addition, time-delay occurs frequently due to the transmission channel jam. Therefore, in future work, we will discuss practical scenarios for time-varying typologies with time-delay. Moreover, we expect to bring the mechanism of multi-agent cooperative systems, such as consensus, formation, and coverage, into the MARL problem working as one aspect of intrinsic reward signal to accelerate the training process.

Acknowledgments

This work is supported by grants from National Natural Science Foundation of China under Grant No. 61803340 and No. 61673344.

References

Bacchiani, G.; Molinari, D.; and Patander, M. 2019. Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning. ArXiv preprint arXiv:1903.01365.

Baras, J. S.; and Liu, X. 2019. Trust is the cure to distributed consensus with adversaries. In *2019 27th Mediterranean Conference on Control and Automation (MED)*, 195–202. Akko, Israel.

Brown, N.; and Sandholm, T. 2018. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* 359(6374): 418–424.

Calvo, J. J. A.; and Dusparic, I. 2018. Heterogeneous multi-agent deep reinforcement learning for traffic lights control. In *The 26th Irish Conference on Artificial Intelligence and Cognitive Science*, 1–12. Dublin, Ireland.

Chu, T.; Wang, J.; Codeca, L.; and et al. 2019. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems* 21(3): 1086–1095.

Gui, T.; Liu, P.; Zhang, Q.; and et al. 2019. Mention recommendation in Twitter with cooperative multi-agent reinforcement learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 535–544. Paris, France.

Hernandezleal, P.; Kartal, B.; and Taylor, M. E. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33(6): 750–797.

Hou, J.; Chen, Z.; Lin, Z.; and Xiang, M. 2020. Resilient consensus via weight learning and its application in fault-tolerant clock synchronization. ArXiv:2002.03541.

Hou, J.; and Zheng, R. 2018. Hierarchical consensus problem via group information exchange. *IEEE Transactions on Cybernetics* 99: 1–7.

Humayed, A.; Lin, J.; Li, F.; and Luo, B. 2017. Cyber-physical systems security - A survey. *IEEE Internet of Things Journal* 4(6): 1802–1831.

Kikuya, Y.; Dibaji, S. M.; and Ishii, H. 2017. Fault-tolerant clock synchronization over unreliable channels in wireless sensor networks. *IEEE Transactions on Control of Network Systems* 5(4): 1551–1562.

Kim, W.; Cho, M.; and Sung, Y. 2019. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1–12. Hawaii, USA.

Kwon, D.; and Kim, J. 2019. Multi-agent deep reinforcement learning for cooperative connected vehicles. In *IEEE Global Communications Conference*, 1–6. Taipei, Taiwan.

LeBlanc, H. J.; and Koutsoukos, X. D. 2011. Consensus in networked multi-agent systems with adversaries. In *Proceedings of the 14th international conference on Hybrid systems: computation and control*, 281–290. Chicago, IL, USA.

LeBlanc, H. J.; and Koutsoukos, X. D. 2012. Low complexity resilient consensus in networked multi-agent systems with adversaries. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, 5–14. Beijing, China.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; and Marecki, J. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, 1–12. Sao Paulo.

Li, S.; Wu, Y.; Cui, X.; Dong, H.; and et al. 2019. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4213–4220. Hawaii, USA.

Liu, X.; and Baras, J. S. 2014. Using trust in distributed consensus with adversaries in sensor and other networks. In *Proceedings of the 2014 International Conference on Information Fusion*, 1–7. Salamanca, Spain.

Lowe, R.; Wu, Y.; Tamar, A.; and et al. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, 6379–6390. Long Beach, California, USA.

Mikulski, D. G.; Lewis, F. L.; Gu, E. Y.; and Hudas, G. R. 2012. Trust method for multi-agent consensus. In *Conference on Unmanned Systems Technology XIV*, 1–15. Baltimore, MD, USA.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; and et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Mu, C.; Zhao, Q.; Gao, Z.; and Sun, C. 2019. Q-learning solution for optimal consensus control of discrete-time multiagent systems using reinforcement learning. *Journal of the Franklin Institute* 356: 6946–6967.

Naderializadeh, N.; Sydir, J. J.; Simsek, M.; and et al. 2020. Resource management in wireless networks via multi-agent deep reinforcement learning. *arXiv preprint arXiv:2002.06215* .

Noureddine, D.; Gharbi, A.; and Ahmed, S. 2017. Multi-agent deep reinforcement learning for task allocation in dynamic environment. In *Proceedings of the 12th International Conference on Software Technologies*, 17–26. Madrid, Spain.

Peng, Z.; Hu, J.; Shi, K.; and et al. 2020. A novel optimal bipartite consensus control scheme for unknown multi-agent systems via model-free reinforcement learning. *Applied Mathematics and Computation* 369: 124821.

Pesce, E.; and Montana, G. 2020. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning* 1–26.

Sharma, M. K.; Zappone, A.; Assaad, M.; and et al. 2019. Distributed power control for large energy harvesting networks: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Cognitive Communications and Networking* 5(4): 1140–1154.

Silver, D.; Schrittwieser, J.; Simonyan, K.; and et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354.

Sundaram, S.; and Hadjicostis, C. N. 2011. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Transactions on Automatic Control* 56(7): 1495–1508.

Tampuu, A.; Matiisen, T.; Kodelja, D.; and et al. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12(4): 1–12.

Vinyals, O.; Babuschkin, I.; Chung, J.; and et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575(7782): 350–354.

Wang, X.; Tan, C.; Wu, F.; and Wang, J. 2019. Fault-tolerant attitude control for rigid spacecraft without angular velocity measurements. *IEEE Transactions on Cybernetics* 1–14.

Zhang, H.; Jiang, H.; Luo, Y.; and Xiao, G. 2017. Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. *IEEE Transactions on Industrial Electronics* 64(5): 4091–4100.

Zhang, H.; and Sundaram, S. 2012. Robustness of information diffusion algorithms to locally bounded adversaries. In *Proceedings of the 2012 American Control Conference*, 5855–5861. Montreal, Quebec, Canada.