Differentially Private Malicious Agent Avoidance in Multiagent Advising Learning

Dayong Ye, Tianqing Zhu, *Member, IEEE*, Wanlei Zhou, and Philip S. Yu, *Fellow, IEEE*

***Abstract*—Agent advising is one of the key approaches to improve agent learning performance by enabling agents to ask for advice between each other. Existing agent advising approaches have two limitations. The first limitation is that all the agents in a system are assumed to be friendly and cooperative. However, in the real world, malicious agents may exist and provide false advice to hinder the learning performance of other agents. The second limitation is that the analysis of communication overhead in these approaches is either overlooked or simplified. However, in communication-constrained environments, communication overhead has to be carefully considered. To overcome the two limitations, this paper proposes a novel differentially private agent advising approach. Our approach employs the Laplace mechanism to add noise on the rewards used by student agents to select teacher agents. By using the differential privacy technique, the proposed approach can reduce the impact of malicious agents without identifying them. Also, by adopting the privacy budget concept, the proposed approach can naturally control communication overhead. The experimental results demonstrate the effectiveness of the proposed approach.**

***Index Terms*—Agent advising, differential privacy, multiagent reinforcement learning (MARL).**

I. Introduction

A MULTIAGENT system is composed of a loosely coupled group of agents. These agents can collaboratively solve complex problems by interacting with one another [1]. In multiagent systems, multiagent reinforcement learning (MARL) is one of the fundamental research issues [2]. In MARL, an agent can learn behavior through trial-and-error interactions with an environment or with other agents [3]. Regular MARL approaches may need a large number of interactions between agents and an environment to learn proper behaviors [4]. To improve agent learning performance, agent advising is proposed, where agents are allowed to ask for advice between each other [5]. Recently, a number of agent advising approaches has been developed [5], [6]. These approaches, however, have two common limitations.

The first limitation is that all the agents in a system are assumed to be cooperative and always provide genuine advice to other agents. In the real world, malicious agents may exist and provide false advice to hamper the learning performance of other agents [6]. An intuitive solution against malicious agents is to build a trust model to identify them [7], [8]. However, building a trust model in a large system is time-consuming and incurs extra communication overhead [8], [9].

The second limitation is that the analysis of communication overhead in existing approaches is either overlooked or simplified. In communication-constrained environments, communication overhead has to be carefully considered. For example, in sensor networks, communication is the most power-consuming operation to sensors [10]. To prolong the lifetime of sensors, communication among sensors has to be effectively managed.

This paper proposes a novel differentially private agent advising approach, which takes both the malicious agent and communication overhead problems into account. Our approach addresses the two problems from a very different perspective.

For the malicious agent problem, we incline to decrease the impact of malicious agents rather than identifying them. The methodology we use is differential privacy, which is a promising privacy model. This model has been mathematically proven that an individual record being stored in or removed out of a dataset makes little difference on an analytical output of the dataset [11]. Inspired by this property of differential privacy, to the best of our knowledge, we are the first to adopt differential privacy to address the malicious agent problem. The property of differential privacy can guarantee that a malicious agent being in or out of a multiagent system has little impact on the benefit of the system.

For the communication overhead problem, the privacy budget concept can be adopted. In differential privacy, privacy budget is applied to control the privacy level. In this paper, privacy budget can naturally be used to control communication overhead, so that only a limited number of advices is allowed in a system. In our approach, differential privacy

Manuscript received November 11, 2018; revised February 6, 2019 and March 15, 2019; accepted March 15, 2019. This work was supported in part by the Australian Research Council Linkage Project under Grant LP170100123, in part by NSF under Grant IIS-1526499, Grant IIS-1763325, and Grant CNS-1626432, and in part by NSFC under Grant 61672313. This paper was recommended by Associate Editor J. Liu. *(Corresponding author: Tianqing Zhu.)*

D. Ye and W. Zhou are with the School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: dayong.ye@uts.edu.au; wanlei.zhou@uts.edu.au).

T. Zhu is with the School of Computer Science, China University of Geosciences, Wuhan 430074, China, and also with the School of Software, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: tianqing.e.zhu@gmail.com).

P. S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: psyu@uic.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2906574

is used from a novel perspective to improve agents' learning performance rather than to preserve agents' privacy. In summary, the contribution of this paper is threefold.

1) The proposed approach overcomes the common limitations of existing approaches by properly addressing the malicious agent and communication overhead problems.
2) By using the differential privacy technique, the proposed approach can reduce the impact of malicious agents without identifying them. Thus, the time and communication overhead used to build a trust model can be conserved.
3) By adopting the privacy budget concept, the proposed approach naturally can control communication overhead without sacrificing the learning performance.

II. Related Work

In this section, we first review the related studies and then discuss their limitations.

A. Review of the Related Works

Torrey and Taylor [12] introduced a teacher–student framework, where a teacher agent provides advice to a student agent to accelerate the student's learning. Torrey and Taylor's framework is teacher-initiated, where the teacher determines when to give the student advice. They also developed a set of heuristic approaches for the teacher agent to decide when to provide advice.

Zimmer *et al.* [13] extended Torrey and Taylor's framework by building a sequential decision-theoretic problem in the framework in order to learn when the teacher should provide advice. Their framework requires that the teacher continuously observes the student's states.

Amir *et al.* [14] proposed an interactive student-teacher framework, where the student and the teacher jointly determine when to ask/provide advice. Specifically, the interactive framework combines the properties of both Clouse's student-initiated framework [15] and Torrey and Taylor's teacher-initiated framework [12]. They then developed a set of heuristic approaches for both the student and the teacher to decide when to ask for/provide advice.

da Silva *et al.*'s teacher–student framework [5] is the first one which takes simultaneity into consideration. In their framework, agents simultaneously learn and an agent can be both a teacher and a student at the same time. In their setting, they proposed an algorithm for the student to decide when to ask for advice and an algorithm for the teacher to decide when to give advice. Then, advice acquisition is carried out via broadcasting. A student agent broadcasts an advice request to all its neighboring agents and then selects an advice through a majority vote, where the advice provided by majority of the neighboring agents is selected.

Wang *et al.* [16] designed a teacher–student framework to accelerate the lexicon convergence speed among agents. In Wang *et al.*'s framework, a student agent uses a multicast manner to probabilistically ask each neighboring agent for advice. The asking probability is based on the distance between the student agent and the teacher agent. After receiving multiple pieces of advice, the student agent selects advice based on degree of the teacher agents, where a teacher agent with a higher degree is assumed to have more expertise.

B. Discussion of the Related Works

The existing approaches have two common limitations: 1) neglecting malicious agents and 2) overlooking or simplifying the analysis of communication overhead.

1) Malicious Agents: As stated by da Silva *et al.* [5], [6], the existence of malicious agents is a key concern in agent advising but has been neglected by most of the literature. To deal with malicious agents, typical techniques include: trust models, intrusion detection, and game theory. These techniques, however, cannot be used in this paper due to the following three reasons. First, building a trust model is time-consuming and incurs extra communication overhead [8], [9]. In this paper, communication overhead has to be carefully considered. Second, detecting agent malicious behaviors [17], [18] is difficult in this paper, as malicious agents probabilistically provide good advice to hide their malicious identity. Third, game theory-based approaches focus on efficient resource allocation to discover external adversarial attacks [19], [20]. In this paper, malicious agents have been mixed with regular agents. Thus, malicious agents are internal adversaries rather than external.

2) Communication Overhead: As discussed in Section I, communication overhead has to be carefully considered in communication-constrained environments, e.g., sensor networks. However, the analysis of communication overhead is overlooked in [15]. Although communication budget is used in the teacher–student frameworks [5], [12]–[14] to limit the number of advice that an agent can ask for or give to, they still have limitations. In [12] and [14], only one teacher and one student interaction is studied, which may hinder the applicability of their frameworks to complex multiagent environments. In [13], the teacher has to continuously observe the student's states, which may be infeasible in privacy intensive environments because the teacher may invade the privacy of the student. In [5], broadcast is used for advice acquisition, which may incur heavy communication overhead. In [16], multicast is used for advice acquisition. Although multicast costs less communication overhead than broadcast to some extent, multicast still incurs heavy communication overhead in extreme situations.

In summary, our approach can overcome the aforementioned two limitations. We handle malicious agents and communication control through a different but effective way by taking the advantage of differential privacy.

III. Motivation Example

Fig. 1 shows a sensor network, which consists of a group of sensors. These sensors are deployed in a battlefield and connected through wireless communication. Each sensor, denoted as an agent, measures its ambient conditions and sends the signal packets to the base station for further processing [21]. Due to the communication range limitation of each sensor, packets usually cannot be directly sent to the base station, but have to

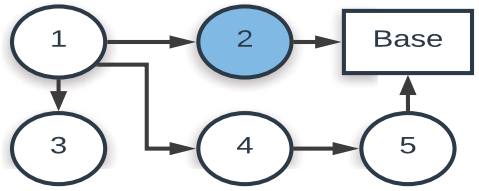

Fig. 1. Motivation example.

be relayed to the base station via multiple sensors. This is the packet routing problem in sensor networks.

The packet routing problem can be efficiently solved using reinforcement learning [22]. To improve the learning performance of sensors, agent advising can be adopted [23]. However, if malicious sensors exist, the learning performance of sensors may be hampered. For example, when sensor 1 plans to send a packet to the base station, sensor 1 may ask advice from its neighbors, sensors 2–4. Obviously, the fastest way to the base station is via sensor 2. However, if sensor 2 has been replaced by the enemy with a malicious sensor [24], [25], sensor 2 may provide false advice to sensor 1. Then, the packet from sensor 1 might be routed to sensor 3, and thus is difficult to reach the base station.

In addition, communication overhead has also to be carefully considered in packet routing, because communication is the most power-consuming operation to sensors [21]. Since each sensor's battery power is limited and cannot be recharged, sensors have to communicate as little as possible to conserve their battery power. Thus, the allowed number of advice requests of each sensor must be set up properly.

In this example, two factors have to be taken into account. First, when malicious sensors exist, packets may not reach their destinations on time due to false advice from malicious sensors. Second, the battery power of sensors is limited and usually un-rechargeable. Existing agent advising approaches neglect malicious agents and simplify the analysis of communication overhead. They may not be applicable in this example. Hence, a novel agent advising approach is developed in this paper which takes the two factors into account.

IV. Preliminaries

A. Background of Reinforcement Learning

Reinforcement learning is usually used to solve sequential decision-making problems, where an agent learns through a trial-and-error manner in an environment. Formally, the sequential decision-making process can be modeled as a Markov decision process (MDP) [26], which is depicted as a tuple $\langle S, A, T, R\rangle$. In this tuple, S is the set of states, A is a set of actions available to the agent, T is the transition function, and R is the reward function.

At each step, the agent observes the state $s \in S$ of the environment, and selects an action $a \in A$ based on its policy π. After performing the action, the agent receives a real-valued reward r, and the environment changes to a new state $s' \in S$. The agent then updates its policy π based on the reward r and the new state s'. Through this way, the agent can gradually accumulate knowledge and improve its policy to maximize its total long-term expected reward.

The policy π can be represented as a Q-function $Q(s, a)$ which estimates the reward that the agent will earn in state s by performing action a [3]. Suppose that there are t sequential time steps, Q-function can be given in an iterative manner as $Q(s_t, a_t) \leftarrow (1-\alpha)\cdot Q(s_t, a_t) + \alpha[r_t + \gamma \cdot \max_a Q(s_{t+1}, a)]$, where α is the learning rate, γ is the discount factor, s_t and s_{t+1} are the current and next state, respectively, a_t is the action taken in state s_t, r_t is the immediate reward obtained by the agent in state s_t after taking action a_t, and $\max_a Q(s_{t+1}, a)$ is a function which outputs the maximum Q-value in state s_{t+1}.

An MDP can be extended to model an MARL process as a tuple $\langle S, A_{1,\ldots,n}, T, R_{1,\ldots,n}\rangle$, where S is the Cartesian product of the sets of local states from each agent, $S = S_1 \times S_2 \times \cdots \times S_n$, A_i is the available action set of agent i, T is the transition function, and R_i is the reward function of agent i [4]. In MARL, state transitions are based on the joint actions of all the agents. Each agent's individual reward is also based on the joint actions of all the agents. Moreover, each agent's policy π_i and Q-function Q_i depend on all the agents' joint policy π and joint Q-function Q, respectively. If all the agents have the same goal, agents are cooperative, e.g., multiple robots cleaning a room. If agents have opposite goals, agents are competitive, e.g., various chess and card games. If some of the agents have the same goal and other agents have opposite goals, agents are semi-cooperative, e.g., multirobot soccer games. Application examples of MARL include: 1) sensor network packet routing [27]; 2) multirobot coordination [28]; and 3) power system restoration [29].

B. Background of Differential Privacy

Differential privacy is a prevalent privacy model that can guarantee that any individual record being stored in or removed out of a dataset makes little difference on an analytical output of the dataset [11], [30], [31]. Two datasets D and D' are *neighboring* datasets if they differ in only one record. A *query* f is a function that maps dataset D to an abstract range $\mathbb{R} : f : D \to \mathbb{R}$. A group of queries is denoted as $F = \{f_1, \ldots, f_m\}$, and $F(D)$ denotes $\{f_1(D), \ldots, f_m(D)\}$.

The maximal difference on the results of query f is defined as the *sensitivity* ΔS, which determines how much perturbation is required for the private-preserving answer. To achieve the target, differential privacy provides a mechanism $\mathcal{M}$, which is a randomized algorithm that accesses the datasets.

The target of *differential privacy* is to mask the difference in the answer of query f between the *neighboring datasets*. In ϵ-differential privacy, parameter ϵ is defined as the *privacy budget*, which controls the privacy guarantee level of mechanism $\mathcal{M}$. A smaller ϵ represents a stronger privacy. The formal definition of differential privacy is presented as follows.

Definition 1 (ϵ-Differential Privacy): A randomized algorithm $\mathcal{M}$ gives ϵ-differential privacy for any pair of *neighboring datasets* D and D', and for every set of outcomes Ω, $\mathcal{M}$ satisfies

$$\Pr[\mathcal{M}(D) \in \Omega] \leq \exp(\epsilon) \cdot \Pr\big[\mathcal{M}(D') \in \Omega\big]. \tag{1}$$

TABLE I
MEANING OF EACH NOTATION

notations	meaning
S	a set of states of the environment
A	a set of actions available to an agent
T	a transition function
R	a reward function
r	an immediate reward obtained by an agent for taking an action in a state
$Q(s,a)$	a reinforcement value for an agent to take action a in state s
teacher/adviser	an agent which provides advice to another agent
student/advisee	an agent which asks for advice from another agent
$\boldsymbol{\pi}$	a probability distribution over the available actions of an agent
$p(j)$	the probability of selecting agent j as an adviser
α, ζ	learning rates, both of which are in $(0,1)$
γ	a discount factor which is in $(0,1)$
ΔS	the sensitivity of a query
ϵ	privacy budget
b	the scale parameter in Laplace mechanism
θ	a parameter used in *Logarithmic Mechanism*
$Neig_i$	the set of neighbouring agents of agent i
n_{visit}	the number of times that an agent has visited a state
C	the communication budget of an agent
P_{ask}	the probability that an agent asks an advice
P_{give}	the probability that an agent gives an advice
e	the $epsilon$ value in the $epsilon$-greedy approach

Sensitivity is a parameter determining how much perturbation is required in the mechanism with a given privacy level.

Definition 2 (Sensitivity): For a query $f : D \to \mathbb{R}$, the sensitivity of f is defined as

$$\Delta S = \max_{D,D'} \left\| f(D) - f(D') \right\|_1. \tag{2}$$

The Laplace mechanism adds Laplace noise to the true answer. We use Lap(b) to represent the noise sampled from the Laplace distribution with the scaling b. The mechanism is defined as follows.

Definition 3 (Laplace Mechanism): Given a function $f : D \to \mathbb{R}$ over a dataset D, (3) provides the ϵ-differential privacy

$$\widehat{f}(D) = f(D) + \text{Lap}\left(\frac{\Delta S}{\epsilon}\right). \tag{3}$$

Application examples of differential privacy include: 1) privacy preserving data publishing [30] and 2) over-fitting avoidance in machine learning [32].

Table I describes the notations and terms used in this paper.

V. DIFFERENTIALLY PRIVATE AGENT ADVISING APPROACH

Our approach is developed in a *teacher–student* framework [5], where multiple agents simultaneously learn in a shared environment. This framework is independent of specific reinforcement learning algorithms. In this framework, each agent can be a teacher or a student or both. Moreover, each agent i has a communication budget C_i to control its communication overhead. Everytime an agent, i, asks for advice from or give advice to another agent, agent i's communication budget is deducted by 1. When agent i runs out of the communication budget, agent i's reinforcement learning process continues without advice.

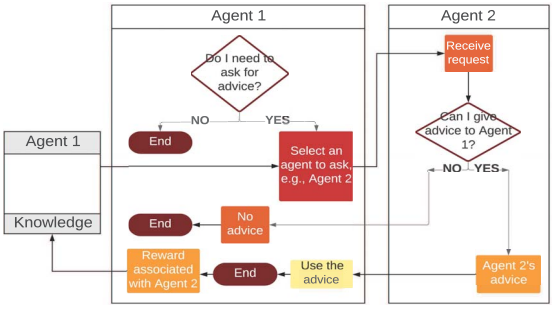

Fig. 2. Overview of the algorithm.

In this framework, each agent has the following knowledge: 1) its available actions in each state; 2) the Q-value of each available action; 3) the existence of its neighboring agents; and 4) a reward table which records the rewards received by using other agents' advice. This knowledge is private and is unknown to other agents.

A. Overview of the Approach

In our approach, during agent advising, each agent has to address the following four subproblems: 1) when to ask for advice; 2) which adviser to ask; 3) when to give advice; and 4) how to give advice.

The overview of the proposed approach is shown in Fig. 2. At a time step, agent 1 decides whether it needs to ask for advice (Section V-B). If agent 1 decides to ask for advice, it also has to decides which adviser to ask (Section V-C). Suppose that agent 1 selects agent 2 to ask for advice. Agent 1 sends an advice request message to agent 2. This message includes: 1) the current state of agent 1 and 2) the available actions of agent 1. Once agent 2 receives the request message, agent 2 makes a decision on whether to give advice to agent 1 (Section V-D). On the one hand, if agent 2 does not have suitable advice, it sends a reject message back to agent 1. Agent 1 selects an action based on its own knowledge. The advising process ends. On the other hand, if agent 2 has suitable advice, it sends advice to agent 1. The advice includes the best action in agent 1's state according to agent 2's knowledge. Agent 1 performs the action suggested by agent 2. The advising process ends.

If agent 2 is malicious and has communication budget, it always gives advice to agent 1 by probabilistically advising agent 1 an action (Section V-E). Agent 1 performs the action suggested by agent 2. The advising process ends.

After agent 1 uses agent 2's advice, agent 1 receives a reward. Agent 1 associates this reward with agent 2 and stores the reward in agent 1's knowledge base for future use.

The advising approach is described in detail in the following sections.

B. Deciding When to Ask Advice

In each state, an agent, i, as an advisee, needs to decide whether to ask for advice from its communication reachable

agents, that is, neighboring agents Neig_i, as long as agent i's communication budget, C_i, is greater than 0. This decision is based on the confidence of agent i in the current state. The confidence of agent i in state s depends on how many times agent i has visited state s, recorded as n_{visit}. A higher n_{visit} value means a higher confidence and results in a lower probability of asking advice. Formally, the decision of asking advice is presented as a probability shown in (4), where square root is used to reduce the decay speed of P_{ask}

$$P_{\text{ask}} = \begin{cases} \frac{1}{\sqrt{n_{\text{visit}}}}, & n_{\text{visit}} > 0 \\ 1, & n_{\text{visit}} = 0. \end{cases} \tag{4}$$

C. Deciding Which Adviser Is Asked

When an agent, i, decides to ask advice, it also has to decide which adviser is asked. Considering that malicious agents may exist in the environment, a novel adviser selection approach has to be developed. We first use multiarmed bandit to model the adviser selection problem. Then, we develop a differentially private algorithm to solve the adviser selection problem in both static and dynamic environments.

The well-known stochastic multiarmed bandit [33] involves a player, which sequentially choosing among a set of K arms, $Ag = \{1, \ldots, K\}$. At each time step, the player selects an arm $ag \in Ag$ to pull and receives a reward r. The reward r is based on a fixed distribution, which is unknown to the player. The goal of the player is to pull arms, based on a calculated probability distribution $\boldsymbol{p}$, to maximize its total reward after a predefined number of time steps. $\boldsymbol{p}$ defines a probability distribution over the arms at time step t given the history of previously selected arms and the corresponding rewards. Formally, $p = p(ag_t | ag_{1:t-1}, r_{1:t-1})$, where $ag_{1:t-1} = ag_1, \ldots, ag_{t-1}$ and $r_{1:t-1} = r_1, \ldots, r_{t-1}$.

In our problem, for advisee agent i, its neighboring agents are modeled as arms, and selecting a neighboring agent to ask advice is modeled as pulling an arm. After using the advice, that is, pulling the arm, the advisee agent will get a reward, r. We develop a differentially private algorithm to achieve the probability distribution $\boldsymbol{p}$. Differential privacy was originally proposed by Dwork [11]. It defines the amount of information about the input of an algorithm that is leaked to an observer of its output. In our algorithm, the input is a sequence of rewards, and the output is one of the neighboring agents which will be an adviser. Formally, we have the following definition.

Definition 4 [(ϵ)-Differentially Private Algorithm]: An algorithm is (ϵ)-differentially private if for all sequences $r_{1:t-1}$ and $r'_{1:t-1}$ that differ in at most one time step, we have

$$p\big(ag_t \in N' | ag_{1:t-1}, r_{1:t-1}\big) \leq p\big(ag_t \in N' | ag_{1:t-1}, r'_{1:t-1}\big) \cdot e^{\epsilon} \tag{5}$$

for all $N' \subseteq N$, where N is the set of all agents in the environment and N' is a subset of N which could be the neighboring agents of an agent.

Based on the analysis of Tossou and Dimitrakakis [34], by using a differentially private algorithm, any individual agent changing reward r_t will not change too much the best agent selected at time step t or later on. Also, any individual agent will not affect too much the output of the algorithm.

Algorithm 1: Adviser Selection of an Advisee Agent i in Static Environments

1 **Initialisation**: $p(j) = \frac{1}{|Neig_i|}$; $t = 0$; $r_0 = 0$;
2 Select an adviser agent based on probability distribution $\boldsymbol{p} = (p_1, ..., p_{|Neig_i|})$;
3 $t \leftarrow t + 1$;
4 Observe reward r_t;
5 Instantiate $r_1, ..., r_{t-1}$ with *Logarithmic Mechanism* $\mathcal{L}$ (refer to **Algorithm 2**);
6 Observe the output of *Logarithmic Mechanism*, $\mathcal{L}_1, ..., \mathcal{L}_{t-1}$;
7 /* update probability distribution $\boldsymbol{p}$ */
8 **for** *each agent* $j \in Neig_i$ **do**
9 $\quad sum_j \leftarrow 0$;
10 $\quad$ **for** $1 \leq k \leq t-1$ **do**
11 $\quad\quad$ **if** $\mathcal{L}_k$ *is from the advice of agent* j **then**
12 $\quad\quad\quad sum_j \leftarrow sum_j + \mathcal{L}_k$;
13 $\overline{sum} \leftarrow \sum_{j \in Neig_i} (p(j) \cdot sum_j)$;
14 **for** *each agent* $j \in Neig_i$ **do**
15 $\quad p(j) \leftarrow p(j) + \eta \cdot (sum_j - \overline{sum})$;
16 $\boldsymbol{p} \leftarrow Normalise(\boldsymbol{p})$; /* ref **Algorithm 3** */

The privacy parameter ϵ determines the extent to which an individual agent affects the output.

Such properties of differentially private algorithms can mitigate the impact of malicious agents to a large extent without identifying them. Specifically, differential privacy is a privacy model which guarantees that an individual record is stored in or removed out of a database will have no difference between the analytic output. The model can be achieved using a differential privacy mechanism, which adds calibrated noise to the analytical output so that adversaries cannot tell a specific record is in or out of the database. In this paper, we use the properties of differential privacy to mitigate the impact of malicious agents by adding calibrated noise to the rewards, so that the selection probability of a malicious agent can be reduced.

1) Static Environments: The differentially private adviser selection algorithm in static environments is formally given in Algorithm 1. In line 1, the probability of selecting each neighboring agent as an adviser is uniformly initialized as $p(j) = (1/|\text{Neig}_i|)$, and time step t and reward r are initialized as 0. Then, in line 2, agent i selects an adviser based on probability distribution over each neighboring agent, that is, p_1, p_2,..., $p_{|\text{Neig}_i|}$, where an agent with a higher probability is more likely to be selected. After agent i adopts the advice, it receives a reward, r_t (line 4). Agent i, then, uses *logarithmic mechanism* $\mathcal{L}$ [35] to instantiate the stream $r_1, \ldots, r_{t-1}$ (Algorithm 2), and observes the output stream $\mathcal{L}_1, \ldots, \mathcal{L}_{t-1}$ (lines 5 and 6). Then, based on the output stream, agent i updates the probability distribution $\boldsymbol{p} = (p_1, \ldots, p_{|\text{Neig}_i|})$ (lines 8–15). Specifically, in lines 8–12, for each neighboring agent j, agent i calculates how much reward that agent j contributes in the last $t-1$ time steps. It should be noted

Algorithm 2: Logarithmic Mechanism [35]

1 **Input**: $r_1, \ldots, r_{t-1}$;
2 **Output**: $\mathcal{L}_1, \ldots, \mathcal{L}_{t-1}$;
3 **for** $1 \le i \le t-1$ **do**
4 $\quad \theta \leftarrow 0$;
5 $\quad \theta \leftarrow \theta + r_i$;
6 $\quad$ **if** $i = 2^k$ *for* $k \in \{0, 1, \ldots\}$ **then**
7 $\quad\quad \theta \leftarrow \theta + Lap(\frac{\Delta S \cdot \lceil \ln t \rceil}{\epsilon})$;
8 $\quad$ Output $\mathcal{L}_i \leftarrow \theta$;

Algorithm 3: *Normalize*()

1 For agent i, there are $|Neig_i|$ neighbouring agents;
2 Let $d = min_{1 \le k \le |Neig_i|} p(k)$, mapping center $c_0 = 0.5$ and mapping lower bound $\Delta = 0.001$;
3 **if** $d < \Delta$ **then**
4 $\quad \rho \leftarrow \frac{c_0 - \Delta}{c_0 - d}$;
5 $\quad$ **for** $k = 1$ *to* $|Neig_i|$ **do**
6 $\quad\quad p(k) \leftarrow c_0 - \rho \cdot (c_0 - p(k))$;
7 $\Pi \leftarrow \sum_{1 \le k \le |Neig_i|} p(k)$;
8 **for** $k = 1$ *to* $|Neig_i|$ **do**
9 $\quad p(k) \leftarrow \frac{p(k)}{\Pi}$;
10 **return** $\boldsymbol{p}$;

that the reward in line 12 has been added with Laplace noise which is random in each invocation of Algorithm 2. Thus, the accumulated reward sum_j cannot be stored for future use. In line 13, agent i calculates the expected reward, and then uses this expected reward to update the probability of selecting each neighboring agent j (lines 14 and 15). The update rule in line 15 is an approximation of policy gradient [36], which is based on: 1) the probability of selecting each neighbor and 2) the difference of the reward contributed by that neighbor and the average reward of all the neighbors. Also, η is the learning rate which determines to what extent the reward difference affects the policy. This update rule encourages agents to select the advisors whose contributed reward is above the average reward. Finally, in line 16, agent i uses Algorithm 3 to normalize probability distribution $\boldsymbol{p}$, that is, to ensure that each $p(j)$ is in $[0, 1]$, $1 \le j \le |\text{Neig}_i|$, and $\sum_{1 \le j \le |\text{Neig}_i|} p(j) = 1$.

Algorithm 2 was developed by Chan *et al.* [35], which is a differentially private mechanism used to release a data stream. The input of Algorithm 2 is a stream with $t-1$ time steps. The output is a noisy stream with the same length as the input stream. To conserve the privacy budget, noise is added in a binary tree structure. Specifically, in line 4, a parameter θ is initialized to 0, which is used to keep track of the approximate reward at each time step. In line 5, when a reward, r_i, comes, the reward value is added to θ. When the time step, indexed by i, is equal to a power of 2, a randomness $\text{Lap}([\Delta S \cdot \lceil \ln t \rceil]/\epsilon)$ is added to θ as an output (lines 6–8). Here, $\text{Lap}([\Delta S \cdot \lceil \ln t \rceil]/\epsilon)$ is the Laplace distribution to satisfy with differential privacy. ΔS is the sensitivity of the average reward. As the current reward directly has the impact on the selection probability $\boldsymbol{p}$ in Algorithm 1, the sensitivity ΔS is determined by the average reward. For example, if the rewards scaling is from 1 to 5, the average will be 3 and the maximum difference between neighboring set will be 1. Then $\Delta S = 1$ in this case. As there are t steps in the stream, the privacy budget will be divided by t originally. In the *logarithmic mechanism*, the noise will be added to the parent nodes in a tree structure, which only has $\ln t$ steps involved. Therefore, the privacy budget will be divided by the rounds number of $\ln t$ to upper integer, which is denoted as $\lceil \ln t \rceil$.

In Algorithm 3, to normalize an invalid probability distribution to a valid one, proportion-based mapping is used. The invalid probability distribution contains the obtained knowledge of agent i and in order to preserve the knowledge, the normalized probability distribution should be as "close" as possible to the un-normalized one. Proportion-based mapping can adjust each invalid probability into the range $(0, 1)$ and meanwhile, can keep the relative magnitude of the probabilities.

2) Dynamic Environments: In dynamic environments, new agents may join the environment and existing agents may leave the environment. Thus, Algorithm 1 has to be extended to accommodate dynamic situations. The extended version of Algorithm 1 is given in Algorithm 4.

The major part of Algorithm 4 is the same as Algorithm 1, except lines 2–13 in Algorithm 4. In line 2, agent i observes its neighbors to check whether a new neighbor joins or an existing neighbor leaves. If a new neighboring agent joins, the new agent is assigned an average probability $[1/(|\text{Neig}_i| + 1)]$ because there is no past knowledge about this new agent (lines 3 and 4). Then, each of the rest neighboring agents is reassigned a new probability based on its current probability (lines 5 and 6). The new agent is then added into the neighbor set of agent i (line 7). If an existing neighboring agent leaves, its current probability will be distributed to the rest neighboring agents. This distribution is based on the current probability of each of the rest neighboring agents (lines 8–12). The left agent is then removed from the neighbor set of agent i (line 13).

D. Deciding When to Give Advice

When an agent, j, receives advice request from another agent. Agent j, as an adviser, needs to decide whether to give advice to the advisee agent, as long as agent j's communication budget C_j is greater than 0. This decision is based on the confidence of agent j in advisee's current state. Similar to the asking advice situation, a higher confidence of adviser agent j means a higher probability of giving advice. Formally, the decision of giving advice is presented as a probability shown in

$$P_{\text{give}} = \begin{cases} 1 - \frac{1}{\sqrt{n_{\text{visit}}}}, & n_{\text{visit}} > 0 \\ 0, & n_{\text{visit}} = 0. \end{cases} \tag{6}$$

E. Deciding How to Give Advice

If an agent, j, is cooperative, it always advises the best action to the advisee, i, that is, the action which has the highest Q-value in agent i's current state s, $Q^*(s)$. If an agent, j, is malicious, it does not always advise its worst action to the

Algorithm 4: Adviser Selection of an Advisee Agent i in Dynamic Environments

Initialisation: $p(j) = \frac{1}{|Neig_i|}$; $t = 0$; $r_0 = 0$;
Observe $Neig_i$;
if *a new agent joins* **then**
  $p(|Neig_i|+1) \leftarrow \frac{1}{|Neig_i|+1}$;
  for $0 \le j \le |Neig_i|$ **do**
    $p(j) \leftarrow \frac{|Neig_i|}{|Neig_i|+1} \cdot p(j)$;
  $Neig_i \leftarrow Neig_i \cup \{agent_{new}\}$;
if *an existing agent leaves* **then**
  Suppose that the left agent is agent $|Neig_i|$;
  $\rho \leftarrow \frac{1}{1-p(|Neig_i|)}$;
  for $0 \le j \le |Neig_i| - 1$ **do**
    $p(j) \leftarrow \rho \cdot p(j)$;
  $Neig_i \leftarrow Neig_i - \{agent_{|Neig_i|}\}$;
Selects an adviser agent based on probability distribution $\boldsymbol{p} = (p_1, ..., p_{|Neig_i|})$;
$t \leftarrow t + 1$;
Observe reward r_t;
Instantiate $r_1, ..., r_{t-1}$ with *Logarithmic Mechanism* $\mathcal{L}$; /* ref **Algorithm 2** */
Observe the output of *Logarithmic Mechanism*, $\mathcal{L}_1, ..., \mathcal{L}_{t-1}$;
/* update probability distribution $\boldsymbol{p}$ */
for *each agent* $j \in Neig_i$ **do**
  $sum_j \leftarrow 0$;
  for $1 \le k \le t-1$ **do**
    if $\mathcal{L}_k$ *is from the advice of agent* j **then**
      $sum_j \leftarrow sum_j + \mathcal{L}_k$;
$\overline{sum} \leftarrow \sum_{j \in Neig_i} (p(j) \cdot sum_j)$;
for *each agent* $j \in Neig_i$ **do**
  $p(j) \leftarrow p(j) + \eta \cdot (sum_j - \overline{sum})$;
$\boldsymbol{p} \leftarrow Normalise(\boldsymbol{p})$; /* ref **Algorithm 3** */

advisee, that is, the action which has the lowest Q-value. This is because, as described in Section V-C, an advisee, i, selects an adviser based on the accumulated reward that i obtains by using the adviser's advice in the past $t-1$ time steps. If a malicious agent, j, always advises the worst action to the advisee, the malicious agent will be set a very low probability of selection by the advisee in very early stages, which implies that the malicious agent has a very high probability not to be selected as an adviser. To avoid this situation, the malicious agent should not always advise the worst action.

We develop an *epsilon*-greedy approach for the malicious agent, j, to probabilistically select an action as the advice. This approach is to increase the difficulty of avoiding malicious agents and is independent of differential privacy. The regular *epsilon*-greedy approach, with a fixed *epsilon* value denoted as e, is first proposed by Watkins [37], and has been proved to be efficient in various fields [38], [39]. The regular *epsilon*-greedy approach has also been extended in several ways, e.g., *epsilon*-decreasing [40], with a decreasing e value, and adaptive *epsilon*-greedy [41], with a changing e value based on the e value in the last time step. In this paper, the e value can be changed by a malicious agent based on the Q-values of its available actions in the current time step. Therefore, existing *epsilon*-greedy approaches cannot be used. Our *epsilon*-greedy approach is given in

$$P(a) = \begin{cases} (1 - e(t)) + \frac{e(t)}{n}, & \text{action, } a, \text{ has the lowest } Q\text{-value} \\ \frac{e(t)}{n}, & \text{otherwise} \end{cases} \tag{7}$$

where $0 < e(t) < 1$ is a coefficient, t means the current time step, and n is the number of available actions. The value of $e(t)$ may change in different time steps. Equation (7) demonstrates that the malicious agent advises the worst action with probability $(1 - e(t)) + (e(t)/n)$ while advises each of the rest actions with probability $(e(t)/n)$.

Equation (7) was originally developed in [39]. In the original version of this equation, however, the e value is fixed, while in this paper, the e value is able to change. The details about how to change the $e(t)$ value are discussed in Remark 3 in Section VI-B.

F. Comparison Between This Paper and the Related Works

Existing works mainly study the three subproblems of agent advising: "when to ask for advice," "which adviser to ask," and "when to give advice." The fourth subproblem, "how to give advice," has not been studied in existing works, because they do not take malicious agents into account. In existing works, for "when to ask for/give advice," the decision can be based on: 1) the learning stage of the student; 2) the importance of the student's current state; 3) the intended action of the student; or 4) the student's and the teacher's confidence on the current state. For which adviser to ask, the decision can be based on the majority vote or the degrees of advisers. In comparison, our contribution mainly focuses on which adviser to ask and how to give advice. For when to ask for/give advice, our approach is based on students' and teachers' confidence on current states. For which adviser to ask, our approach is based on students' historical rewards incurred by teachers. For how to give advice, our approach distinguishes normal agents and malicious agents. Normal agents always provide genuine advice, whereas malicious agents probabilistically provide false advice.

VI. Theoretical Analysis

In this section, theoretical analysis about the proposed approach is presented.

A. Differential Privacy Analysis

Theorem 1: The proposed approach satisfies with ϵ-differential privacy.

Proof: See the supplementary material. ■

B. Malicious Agents Utility Analysis

Suppose that malicious agent j has n available actions in an advisee's current state. Each action a_k is associated with a Q-value, Q_k, where $1 \leq k \leq n$. Without loss of generality, suppose that $Q_1 \leq Q_2 \leq \cdots \leq Q_n$. As the aim of a malicious agent is to provide bad advice to other agents, the utility of a malicious agent is inversely proportional to Q-value of the advised action. Thus, the utility of a malicious agent for selecting each action as advice can be set to $u_1 = Q_n, u_2 = Q_{n-1}, \ldots, u_n = Q_1$, and obviously $u_1 \geq u_2 \geq \cdots \geq u_n$. Then, we have the following theorem.

Theorem 2: To guarantee the discounted maximum utility of a malicious agent, the upper bound value of $e(t)$ is $[((1-\mu)\cdot u_1)/(u_1 - [(\sum_{1\leq k\leq n} u_k)/n])]$, where $0 < \mu < 1$ is the discount factor and u_1 is the maximum among $u_1, \ldots, u_n$.

Proof: See the supplementary material. ■

Remark 1 (Discounted Maximum Utility): In Theorem 2, we use discounted maximum utility $\mu \cdot u_1$, instead of real maximum utility, u_1. This is because the malicious agent is aware of the fact that if it advises the worst action, it will have a high probability not to be selected as an adviser in the next time step. Therefore, the real maximum utility is not sustainable and has to be added a discount factor.

Remark 2 (The Value of Discount Factor μ): As $e(t) < 1$, it should be held that $[((1-\mu)\cdot u_1)/(u_1 - [(\sum_{1\leq k\leq n} u_k)/n])] < 1$. Then, after simplifying the in-equation, we have $\mu > [(\sum_{1\leq k\leq n} u_k)/(u_1 \cdot n)]$. Thus, the lower bound of μ is $[(\sum_{1\leq k\leq n} u_k)/(u_1 \cdot n)]$. The choice of the value of μ is left to malicious agents. A high μ value implies a greedy malicious agent.

Remark 3 [The Value of e(t)]: According to Theorem 2, the upper bound value of $e(t)$ is $[((1-\mu)\cdot u_1)/(u_1 - [(\sum_{1\leq k\leq n} u_k)/n])]$. Since $u_1, \ldots, u_n$ may change in different time steps, the upper bound value of $e(t)$ may also change in different time steps. The value of $e(t)$ can be any number in $[0, [((1-\mu)\cdot u_1)/(u_1 - [(\sum_{1\leq k\leq n} u_k)/n])]]$. When $e(t) = 0$, a malicious agent always advises the worst action. The malicious agent receives high utility in early stages, but it will be avoided by student agents very soon, because student agents receive low rewards by using the malicious agent's advice. The probability of a student agent selecting an adviser is based on the accumulated reward brought by the adviser. Therefore, in each time step t, $e(t)$ is set to the upper bound: $[((1-\mu)\cdot u_1)/(u_1 - [(\sum_{1\leq k\leq n} u_k)/n])]$, which can guarantee the discounted maximum utility of a malicious agent and can increase the probability that the malicious agent is selected as an adviser.

Remark 4 (Malicious Agent Advising Approaches): In the real world, malicious agents may use different approaches to give advice in order to avoid being identified. However, no matter what approaches malicious agents would use, their target is to disturb the normal behavior and hinder the performance of other agents. This target can be reflected by the total reward of a system. This paper focuses on the total reward of a system. This means that the approaches used by malicious agents are not important, as long as our approach can increase the total reward of a system (see the following system utility analysis).

C. Systems Utility Analysis

The utility of a multiagent system can be estimated by the overall reward gained by all the agents. We will demonstrate that with a very high probability, the utility of a multiagent system with differential privacy is greater than a multiagent system without differential privacy.

We apply a widely used utility definition in differential privacy, suggested by Blum *et al.* [42]. The utility is measured by the maximum difference between the reward of differentially private system and the original system of histogram on each time step.

Definition 5 [(δ,β)-Useful]: The differentially private multiagent system is (δ,β)-useful if we have $\Pr(R' - R > \delta) \geq 1 - \beta$, where R' and R are the overall rewards of the multiagent system with and without differential privacy, respectively, and $\delta > 0$ and $0 < \beta < 1$.

Theorem 3: For all $\beta > 0$, with probability at least $1 - \beta$, the overall reward, that is, utility, of the multiagent system with differential privacy is greater than the multiagent system without differential privacy. The utility difference is bounded by δ, which satisfies $0 < \delta < -[(\Delta S \cdot \lceil \ln t \rceil)/\epsilon] \cdot \ln(2 - 2(\beta/m)^{(1/t)})$.

Proof: See the supplementary material. ■

It should be noted that a privacy price is needed when using differential privacy. The price is that those high rewards from normal agents may be partially randomized by adding noise. However, as agents always incline to select agents as advisers with high rewards, but not necessarily the highest reward, the selection results of normal agents will not be affected too much. In other words, agents with high rewards have "reward budget" to be compromised.

Remark 5 (The Upper Bound of δ): The upper bound of δ, $-[(\Delta S \cdot \lceil \ln t \rceil)/\epsilon] \cdot \ln(2 - 2(\beta/m)^{(1/t)}$ relies on t. Let $u(t) = -[(\Delta S \cdot \lceil \ln t \rceil)/\epsilon] \cdot \ln(2 - 2(\beta/m)^{(1/t)}$. Hence, $u(t)$ is monotonically decreasing with the increase of t. Specifically, when $t \to 0$, $u(t) \to \infty$, and when $t \to \infty$, $u(t) \to 0$. This means that in early stages, using DP can significantly improve agents' performance. However, as time progresses, the performance improvement by using DP decreases. This is due to the fact that agents have to consume privacy budget to ask for/give advice. When privacy budget is used up, agents cannot communicate any more. In this situation, using DP cannot improve performance.

D. Convergence Analysis

Theorem 4: Algorithm 1 is convergent if: 1) the reward of selecting each neighbor is set properly and 2) learning rate η is decayed gradually.

Proof: See the supplementary material. ■

E. Communication Complexity Analysis

Theorem 5: The overall communication complexity of the proposed approach is $O(C \cdot m)$, where m is the number of agents and C is the maximum communication budget among $(C_1, \ldots, C_m)$.

Proof: See the supplementary material. ■

In comparison, the overall communication complexity of the broadcast-based approach [5] is $O(C \cdot m^2)$. In [5], for each

agent, two communication budgets are set: 1) communication budget for asking advice, C_{ask} and 2) communication budget for giving advice, C_{give}. When an agent, i, decides to ask advice, it sends a request to all the neighboring agents. In the extreme case, all the agents in the system are agent i's neighbors. Again, suppose that the number of agents in the environment is m. Thus, the complexity of communication between agent i and its neighboring agents is $O(C^i_{\text{ask}} + C' \cdot (m-1)) = O(C \cdot m)$, where $C' = \max(C^1_{\text{give}}, \ldots, C^{i-1}_{\text{give}}, C^{i+1}_{\text{give}}, \ldots, C^m_{\text{give}})$ and $C = \max(C^i_{\text{ask}}, C')$. There are m agents in the environment. Thus, the overall communication complexity is $O(C \cdot m^2)$.

The communication complexity of the multicast-based approach [16] is the same as the broadcast-based approach [5]. This is because in the multicast-based approach, a student agent probabilistically asks each of its neighbors for advice. In the extreme case, all the agents in the system are the student agent's neighbors, and the student agent asks all of them for advice. This situation is the same as the situation in the broadcast-based approach. In comparison, our approach is more efficient than both the broadcast-based and multicast-based approaches in communication.

In our approach, moreover, the setting of C can be controlled by privacy budget ϵ. Each agent sets ϵ/C_i as their privacy budget in each time step and stops when ϵ is used up. When $C > t$, the system can guarantee to complete all communication steps. However, more noise will be added to the system. When $C < t$, the system has the probability to stop before finishing communication steps. However, the adding noise will be limited in the whole system. When $C = t$, the system will stop when all communication steps are completed. Therefore, by adjusting the privacy budget ϵ and the parameter C, we can control the communication overhead of a multiagent system.

F. Time Complexity Analysis

Theorem 6: The overall time complexity of the proposed approach is $O(t \cdot m)$, where m is the number of agents and t is the number of time steps.

Proof: See the supplementary material. ■

Remark 6 (Reducing Time Complexity): To reduce the time complexity, we can either limit the number of neighbors of each agent or reduce the value of t. To limit the number of neighbors of each agent, we can properly set the connections among agents. To reduce the value of t, we can shrink the range of statistics. For example, when $t = 1000$, the range of statistics can be shrunk to $100 \leq t \leq 1000$, namely that the rewards, $r_1, \ldots, r_{99}$, are removed. This shrink can also increase the scalability of our approach, because a reinforcement learning algorithm may have thousands of episodes and maintaining a record of all the previous rewards is a heavy load.

VII. Experiment and Analysis

In this section, we first describe the approaches used for evaluation comparison. Then, we present the evaluation scenarios and the experimental setup. After that, the experimental results are given.

A. Description of the Compared Approaches

The proposed differential privacy-based approach, abbreviated as *DP-based*, is evaluated in comparison with three other related approaches.

1) Broadcast-Based Approach (BC-Based) [5]: In *BC-based* approach, an agent broadcasts a request for advice to its neighbors and selects one advice based on majority vote, where the advice which is suggested by most of the neighboring agents is selected.

2) Simplified Version of DP-Based Approach (DP-Removed): *DP-removed* approach is also developed by us, where the differential privacy part is removed. *DP-removed* approach is used to demonstrate the effectiveness of differential privacy.

3) Regular Learning Approach (Regular): *Regular* approach does not involve any advising strategies. Agents learn in a system based only on received rewards without any advice from other agents. *Regular* approach is used as a standard in the experiments.

In all of the four approaches, agents use a regular Q-learning algorithm as described in Section IV. In the algorithm, the Q-value update function is $Q(s_t, a_t) \leftarrow (1-\alpha) \cdot Q(s_t, a_t) + \alpha[r_t + \gamma \cdot \max_a Q(s_{t+1}, a)]$ and the policy update function is $\pi(s_t, a_t) \leftarrow \pi(s_t, a_t) + \zeta \cdot (Q(s_t, a_t) - \hat{r}_t)$, where ζ is a learning rate and $\hat{r}_t$ is the average reward. $\hat{r}_t$ can be calculated as $\hat{r}_t = \sum_A \pi(s_t, a_t) Q(s_t, a_t)$, where A is the set of available actions in state s_t. The details of the regular Q-learning algorithm can be found in [43]. In the experiments, the values of the reinforcement learning parameters are set to $\alpha = 0.1$, $\gamma = 0.9$, and $\zeta = 0.95$. Moreover, in each state s, the Q-value of each available action a is initialized to $Q(s, a) = 1$, and the probability of selecting each available action is initialized to $\pi(s, a) = (1/|A|)$, where $|A|$ is the number of available actions in state s. These values are empirically chosen.

B. Evaluation Scenarios

Two scenarios are used in the experiments to evaluate the aforementioned four approaches.

1) Packet Routing in Sensor Networks: In this scenario, we use an agent network to model a sensor network, where a sensor is modeled as an agent. Each sensor has a set of packets which will be routed to a base station.

The problem in this scenario is how to efficiently route packets to the base station. When applying reinforcement learning to this scenario, the reward of an agent is based on the number of hops, with which a packet reaches a base station. A less number of hops means a higher reward, where reward r can be set to $r = (10/\text{hops})$. This information is provided by the base station to the agent.

This scenario is used to evaluate the performance of the four approaches in cooperative environments, where agents, except malicious agents, collaboratively work toward a common goal. There are three evaluation metrics in this scenario.

1) *Success Rate:* The ratio between the successfully transmitted packets and the total number of packets.
2) *Average Hops:* The average number of hops through which packets are transmitted to the base station.

3) *Average Communication Overhead:* The ratio between the total number of communication messages generated during the experiments and the total number of agents.

2) Evolution of Cooperation: In this scenario, each agent plays the prisoner's dilemma (PD) game with its neighboring agents. Each agent has two available actions: 1) cooperate and 2) defect. For any pair of neighboring agents, at each time step, if both cooperate, both earn Re as payoff. If both defect, both receive Pu as payoff. If one player opts for defection and the other for cooperation, the defector receives payoff Te and the cooperator receives payoff Su. Generally, $Te > Re > Pu > Su$ and $2Re > Te + Su$.

The problem in this scenario is how to increase the proportion of cooperators. When applying reinforcement learning to this problem, the reward of an agent is the accumulated payoff received by playing the game with its neighbors in the current time step.

This scenario is used to evaluate the performance of the four approaches in competitive environments, where each agent, except malicious agents, attempts to maximize its own benefit. It should be noted that maximizing own benefits does not mean hampering others' benefits which is the aim of malicious agents. There are three evaluation metrics in this scenario.

1) *Proportion of Cooperators:* The ratio between the agents which are cooperators and the total number of agents.
2) *Average Payoff:* The ratio between the total payoff received by all the agents and the total number of agents.
3) *Average Communication Overhead:* The ratio between the total number of communication messages generated during the experiments and the total number of agents.

C. Experimental Setup

The experiments are operated in two types of networks: 1) scale-free networks [44] and 2) random networks.

1) A scale-free network is a network where a couple of agents have many connections while many other agents have only a small number of connections [44]. In the experiments, the distribution of agent degrees follows a power law, $n_d \propto \deg^{-\tau}$, where n_d is the number of agents that have a degree deg and $\tau > 0$ is a constant (typically $\tau \in [2, 3]$). τ is set to 2.5, which makes the average degree of the network approximately 4.
2) A random network is a network where a link between two agents is set with a connection probability p. The probability that an agent has k neighbors follows a binomial distribution $B(n-1, p)$, where n is the number of agents in the network. For a large n and a small p, the degree distribution can be approximated by a Poisson distribution $P(x = k) = e^k \cdot (\lambda^k / k!)$ with $\lambda = n \cdot p$. The connection probability, p, is set to 0.015, which leads to the average degree of the network approximately 4.

In the first scenario, packet routing in sensor networks, random networks are used, because sensors are randomly deployed in some situations, e.g., battlefield [45]. The scale of the networks changes from 20 agents to 120 agents, and the proportion of malicious agents changes from 0 to 0.1. The communication budget of each agent is set to $C = 100$.

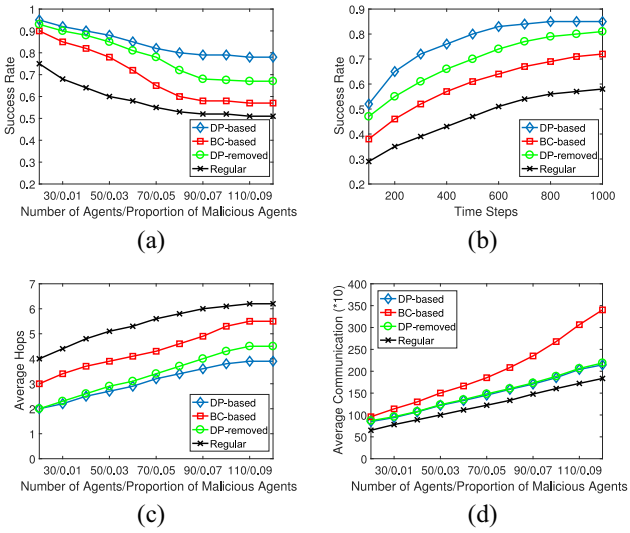

Fig. 3. Performance of the four approaches in static scenario 1. (a) Success rate. (b) Success rate with progress of time steps. (c) Average hops. (d) Average communication overhead.

In the second scenario, evolution of cooperation, scale-free networks are used, because this scenario is similar to a social network and the typical topology of a social network is scale-free [46]. The scale of the networks changes from 100 agents to 600 agents, and the proportion of malicious agents changes from 0 to 0.2. The initial proportion of cooperators changes from 0.1 to 0.3, and the four game theory parameters, Te, Re, Pu, and Su, are set to 5, 4, 0, and -1, respectively. The communication budget of each agent is set to $C = 300$.

Moreover, in both scenarios in dynamic environments, at each time step, with probability 0.01, an existing agent leaves and a new agent joins in the experimental agent network. Each set of experiments consists of 1000 time steps and the results are obtained by averaging 100 runs.

D. Experimental Results

1) Experimental Results in the First Scenario: Fig. 3 demonstrates the performance of the four approaches in the first scenario, that is, packet routing in sensor networks, in static environments.

In Fig. 3(a), it can be seen that with the increase of the network size and the proportion of malicious agents, the success rates of the four approaches gradually decrease. This is mainly due to the property of sensor networks. When the network size increases, a packet usually needs more hops to reach its destination. However, a packet is generally set with a small hop-limit, that is, time-to-live (TTL), in order to conserve sensors' energy. Hence, a part of packets may not reach their destinations within the hop-limit in large networks, and thus the success rate of packet routing decreases.

It can also be seen that *Regular* approach achieves the least success rate among the four approaches, about 7% less than *BC-based* approach, 12% less than *DP-removed* approach, and 15% less than *DP-based* approach, thanks to the strength of agent advising.

It should be noted that with the increase of the network size, the difference between *BC-based* approach and *Regular* approach reduces. As described above, with the increase of the network size, packets need more hops to reach their destinations. Every time a packet takes a hop, the sender agent may ask advice from its neighbors. Therefore, as *BC-based* approach is based on broadcast which is very communication-intensive, most of the communication budget may be used in early stages. Once the communication budget is used up, *BC-based* approach is degraded to *Regular* approach. Thus, they have a similar success rate.

It should also be noted that the difference between *DP-based* approach and *DP-removed* approach enlarges with the increase of malicious agents. This is because *DP-removed* approach does not involve differential privacy and thus is difficult to resist the affect of malicious agents. Agents, then, may be misled by malicious agents to transfer packets far away from destinations. This situation hinders the successful transmission of packets.

In Fig. 3(b), where the number of agents is 70 and the proportion of malicious agents is 0.06, the success rates of all the four approaches increase with progress of time steps, but the convergence speed of *Regular* approach is slower than the other three approaches due to its lack of agent advising.

In Fig. 3(c), with the increase of the network size and the proportion of malicious agents, under the four approaches, the average number of hops used to transfer packets also increases. The explanation of Fig. 3(c) is akin to Fig. 3(a). Generally, with the increase of the network size, the average distance between the original agent of a packet and a destination base station enlarges. Thus, the average number of hops increases.

It should be noted that when the number of agents is larger than 100, the average number of hops keeps relatively steady under the four approaches. As the network size is large, a number of packets may have to use up their hop-limit, that is, TTL, to reach their destinations, and hence, the average number of hops has little space to increase.

In Fig. 3(d), it is shown that *BC-based* approach uses the most communication overhead. In comparison, *DP-based* and *DP-removed* approaches use much less communication overhead, because of their advising mechanisms.

Fig. 4 demonstrates the performance of the four approaches in the first scenario, that is, packet routing in sensor networks, in dynamic environments, where sensor agents dynamically join in or leave the network. Comparing Fig. 4 to Fig. 3, it can be found that under the four approaches, both the success rate [Fig. 4(a)] and the average number of hops [Fig. 4(c)] decrease, whereas the communication overhead [Fig. 4(d)] increases. This is due to the dynamism of the network. When an agent leaves, a route may be broken. Thus, the packets which use that agent as a relaying node have to find other routes. This situation may incur extra hops and increase communication overhead. Then, due to the extra hops, the packets may not reach their destinations within their hop-limits, that is, TTL, and therefore, the success rate decreases.

In Fig. 4(b), the convergence speed of the four approaches is slower compared to Fig. 3(b). This is because when an agent leaves, other agents' knowledge, which is related to this

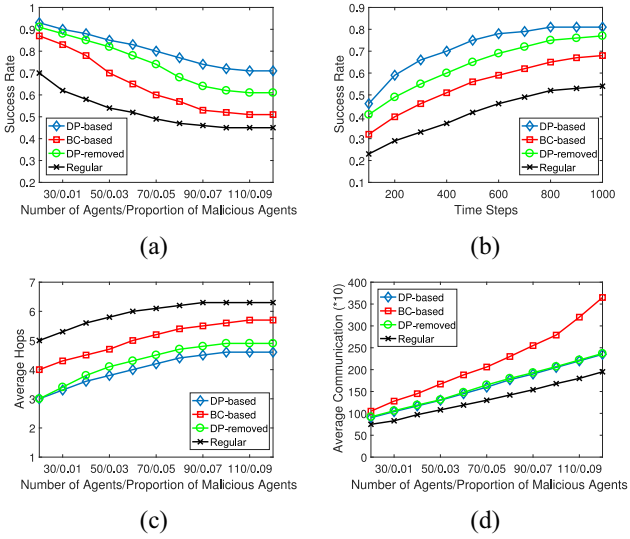

Fig. 4. Performance of the four approaches in dynamic scenario 1. (a) Success rate. (b) Success rate with progress of time steps. (c) Average hops. (d) Average communication overhead.

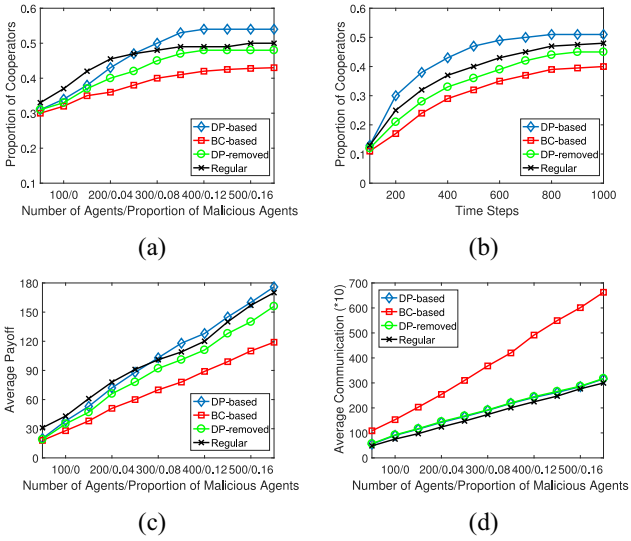

Fig. 5. Performance of the four approaches in static scenario 2. (a) Proportion of cooperators. (b) Proportion of cooperators with progress of time steps. (c) Average payoff. (d) Average communication overhead.

leaving agent, is outdated. These agents have to learn new knowledge. This situation slows down the convergence speed.

2) Experimental Results in the Second Scenario: Fig. 5 demonstrates the performance of the four approaches in the second scenario, that is, evolution of cooperation, in static environments.

In Fig. 5(a), it is interesting to see that *BC-based* and *DP-removed* approaches achieve less proportion of cooperators than *Regular* approach. This situation implies that in evolution of cooperation, using advice may hinder agent learning performance. This can be explained by the fact that evolution of cooperation is in a competitive environment, where the neighbors of an agent are opponents to that agent. Thus, the action, favored by an agent's neighbors, may not be good for that agent. As the initial proportion of cooperators is set to at

most 0.3, there are a large number of defectors in the network. According to the description in Section VII-B2, when a cooperator meets a defector, the defector can obtain a very high payoff, *Te*, whereas the cooperator gets only a very low payoff, *Su*. Then, when the cooperator asks advice from its defecting neighbors, the cooperator may get an advice to become a defector because being a defector can obtain a high payoff.

It is also interesting to see that the difference between *BC-based*/*DP-removed* approaches and the *Regular* approach reduces from about 10% to 3%, with the increase of the number of malicious agents in the network. Due to the aforementioned fact, a cooperator may be advised to become a defector if the majority of the cooperator's neighbors are defectors. However, if a part of the neighbors are malicious agents, they may advise the cooperator to remain a cooperator, because malicious agents can keep utilizing the cooperator so that the cooperator will have a very low payoff. Such a behavior of malicious agents accidentally increases the proportion of cooperators.

Our *DP-based* approach achieves less proportion of cooperators than *Regular* approach when the proportion of malicious agents is less than 0.1 due to the above-mentioned reason, but outperforms *Regular* approach when the proportion of malicious agents is larger than 0.1. Due to the introduction of differential privacy, Laplace noise is added to the payoffs which are used by agents to adjust the policy of selecting advisers. The addition of Laplace noise will affect agents' selection of advisers. For example, as described in Section VII-B2, when a cooperator meets a defector, the cooperator gets payoff $Su = -1$ and the defector gets payoff $Te = 5$. When two defectors meet, both of them get payoff $Pu = 0$, and when two cooperators meet, both of them get payoff $Re = 4$. Then, if a cooperator agent selects a defector as an adviser and the defector is malicious, the defector may advise the cooperator to remain a cooperator, as this can maximize the defector's payoff. If the cooperator uses *DP-removed* approach which does not involve differential privacy, the cooperator may change the adviser in the next time step as the cooperator's payoff, $Su = -1$, is very low. However, if the cooperator uses *DP-based* approach, it may keep using the defector as an adviser in the next time step because the cooperator's payoff $Su = -1$, with the addition of a Laplace noise, may be close to the punishment payoff $Pu = 0$. Thus, the incentive of the cooperator to become a defector is reduced. Similarly, if the cooperator agent selects a cooperator neighbor as an adviser and the cooperator neighbor is malicious, the cooperator neighbor may also advise the cooperator agent to remain a cooperator. Otherwise, the cooperator agent's payoff will be increased from $Re = 4$ to $Te = 5$, and the cooperator neighbor's payoff will be significantly reduced from $Re = 4$ to $Su = -1$. In this situation, moreover, the cooperator agent's payoff $Re = 4$, with the addition of a Laplace noise, may be close to the temptation payoff $Te = 5$. Thus, the incentive of the cooperator to become a defector is reduced. On the contrary, by using *DP-based* approach, a defector has high incentive to become a cooperator. This is because the addition of a Laplace noise can make reward payoff *Re* close to temptation payoff *Te*, and thus being a defector cannot receive much more benefit than being a cooperator. Therefore, it can be found that *DP-based* approach can help the regular *Q*-learning algorithm promote the proportion of cooperators.

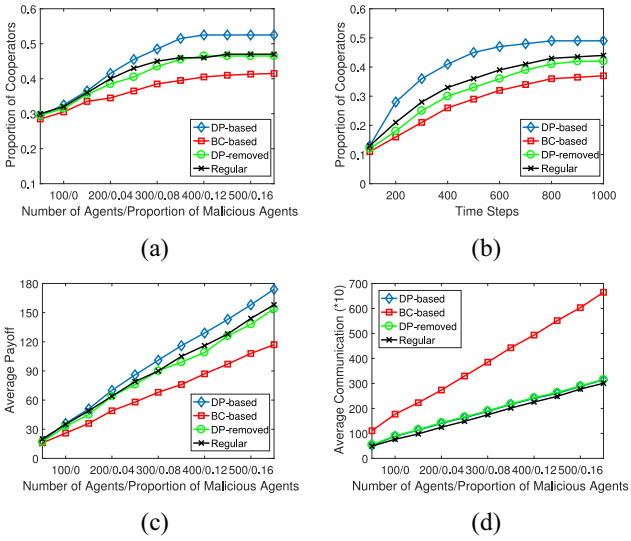

Fig. 6. Performance of the four approaches in dynamic scenario 2. (a) Proportion of cooperators. (b) Proportion of cooperators with progress of time steps. (c) Average payoff. (d) Average communication overhead.

Fig. 5(b) exhibits the change of proportion of cooperators with the progress of time steps under the four approaches, where the number of agents is 350 and the proportion of malicious agents is 0.1. It can be seen that the convergence speed of our *DP-based* approach is the fastest. This is because under *DP-based* approach, agents are not as easy to be swayed by malicious agents as under *DP-removed* and *BC-based* approaches.

Fig. 5(c) shows the average payoff obtained by using the four approaches. Fig. 5(c) has a similar trend to Fig. 5(a). This can be explained by the fact that with the increase of the proportion of cooperators, the average payoff also increases, because two agents can earn more if both of them cooperate than if they choose different options and share the total payoff, that is, $2Re > Te + Su$.

Fig. 5(d) displays the average communication overhead spent by using the four approaches. It can be seen that *BC-based* approach uses the most communication overhead which is about 50% more than the three other approaches. This is owing to the broadcasting property of *BC-based* approach. *DP-removed* and *DP-based* approaches have nearly the same communication overhead which is slightly higher than *Regular* approach (about 3%). This is because *DP-removed* and *DP-based* approaches have the same advice requesting mechanism, while *Regular* approach does not allow agents to ask for advice, which conserves communication overhead.

Fig. 6 demonstrates the performance of the four approaches in the second scenario, that is, evolution of cooperation, in dynamic environments, where agents dynamically join in or leave the network.

In Fig. 6(a), compared to Fig. 5(a), it can be seen that our *DP-based* approach achieves more proportion of cooperators, about 7%, than *Regular* approach, even when the proportion of malicious agents is less than 0.1. Also, *DP-removed* approach

achieves almost the same proportion of cooperators as *Regular* approach, which is higher than *BC-based* approach about 5%. This can be explained by the fact that in *Regular* approach, due to the dynamism of the network, an agent, i, is difficult to learn a stable action update strategy. Every time a new agent joins in as a neighbor of agent i, agent i has to learn a new action update strategy only on itself. In contrast, in the three other approaches, agents are allowed to ask for advice from other agents. Then, agent i can ask advice from its neighbors to update its action. For example, suppose that agent 1 has two neighbors: agents 2 and 3. At a time step, agent 4 joins in. Now, the neighbors of agent 1 are agents 2, 3, and 4. In *Regular* approach, agent 1 has to learn a new action update strategy, whereas in the three other approaches, agent 1 can directly ask for advice from another agent, e.g., agent 2, to update its action. Therefore, the dynamism of the network does not affect agent 1 much.

Fig. 6(b) exhibits the change of proportion of cooperators with the progress of time steps under the four approaches. Compared to Fig. 5(b), the convergence speed of the four approaches is slower in Fig. 6(b) due to the dynamism of the environment.

Fig. 6(c) shows the average payoffs obtained by the four approaches in dynamic environments. Fig. 6(c) has a similar trend to Fig. 6(a), because average payoff is mainly and positively affected by the proportion of cooperators, namely that a higher proportion of cooperators yields a higher average payoff.

Fig. 6(d) displays the communication overhead spent by the four approaches in dynamic environments. Compared to Fig. 5(d), the four approaches in dynamic environments use almost the same communication overhead as in static environments, because communication overhead depends on the number of agents in the network and the advice requesting mechanism. In the experiments, as an agent joins in or leaves the network with the same probability at each time step, the number of agents in the network keeps almost steady. Also, agents do not change advice requesting mechanisms in dynamic environments. Thus, the communication overhead is not affected much by the dynamism of environments.

E. Discussion and Summary

1) Communication Overhead: In Figs. 3(d), 4(d), 5(d), and 6(d), it can be seen that, with the increase of the number of agents, the communication overhead of the four approaches does not converge. According to Theorem 5, the communication complexity of *DP-based* and *DP-removed* approaches is $O(C \cdot m)$, whereas the communication complexity of BC-based approach is $O(C \cdot m^2)$. Thus, if communication budget, C, is fixed, with the increase of the number of agents, m, the communication overhead will inevitably increase. Moreover, although *Regular* approach does not involve advising, communication overhead is still used in *Regular* approach on packet routing (scenario 1) and game playing (scenario 2).

2) Summary: According to the experimental results, the proposed *DP-based* approach achieves the best results in most situations among the four approaches with a little more communication overhead, about 5%, than *Regular* approach. The simplified version of *DP-based* approach, that is, *DP-removed* approach, achieves about 90% performance of *DP-based* approach while spends nearly the same communication overhead as *DP-based* approach. *BC-based* approach achieves about 80% performance of *DP-based* approach while spends about 50% more communication overhead than *BC-based* approach. Such achievement of the proposed *DP-based* approach is significant especially in some situations where the communication resource is limited, e.g., sensor networks.

VIII. Conclusion

This paper proposes a novel differentially private agent advising approach to improve agent learning performance. This approach is the first one which takes malicious agents into consideration, and it is communication efficient. Compared to the benchmark broadcast-based approach, our approach achieves better performance, such as a more proportion of cooperators and a higher success rate, and spends much less communication overhead.

In the future, we intend to improve our approach by allowing agents to estimate advice. In this paper, when a student agent asks for advice from a teacher agent, the student must take that teacher agent's advice. This assumption can be relaxed by enabling the student agent to estimate the advice. If the advice is significantly different from the student agent's knowledge, the student agent may refuse to take the advice.

References

[1] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.

[2] 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.

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. AI Res.*, vol. 4, pp. 237–285, May 1996.

[4] L. Buşoniu, R. Babuŝka, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[5] F. L. da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proc. AAMAS*, São Paulo, Brazil, May 2017, pp. 1100–1108.

[6] F. L. da Silva, M. E. Taylor, and A. H. R. Costa, "Autonomously reusing knowledge in multiagent reinforcement learning," in *Proc. IJCAI*, Stockholm, Sweden, Jul. 2018, pp. 5487–5493.

[7] I. Pinyol and J. Sabater-Mir, "Computational trust and reputation models for open multi-agent systems: A review," *Artif. Intell. Rev.*, vol. 40, no. 1, pp. 1–25, 2013.

[8] H. Yu, Z. Shen, C. Leung, C. Miao, and V. R. Lesser, "A survey of multi-agent trust management systems," *IEEE Access*, vol. 1, pp. 35–50, 2013.

[9] J. Granatyr *et al.*, "Trust and reputation models for multiagent systems," *ACM Comput. Surveys*, vol. 48, no. 2, 2015, Art. no. 27.

[10] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.

[11] C. Dwork, "Differential privacy," in *Proc. ICALP*, 2006, pp. 1–12.

[12] L. Torrey and M. E. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proc. AAMAS*, 2013, pp. 1053–1060.

[13] M. Zimmer, P. Viappiani, and P. Weng, "Teacher–student framework: A reinforcement learning approach," in *Proc. AAMAS Workshop Auton. Robots Multirobot Syst.*, 2014, pp. 1–18.

[14] O. Amir, E. Kamar, A. Kolobov, and B. Grosz, "Interactive teaching strategies for agent training," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 804–811.

[15] J. A. Clouse, "Learning from an automated training agent," in *Adaptation and Learning in Multiagent Systems*. Heidelberg, Germany: Springer-Verlag, 1996.
[16] Y. Wang, W. Lu, J. Hao, J. Wei, and H. Leung, "Efficient convention emergence through decoupled reinforcement social learning with teacher–student mechanism," in *Proc. AAMAS*, 2018, pp. 795–803.
[17] R. P. D. Wegner, "Multi-agent malicious behaviour detection," Ph.D. dissertation, Faculty Grad. Stud., Univ. Manitoba, Winnipeg, MB, Canada, 2012.
[18] M. S. Alam, "An intelligent multi-agent based detection framework for classification of android malware," Ph.D. dissertation, Faculty Grad. Postdoctoral Stud., Univ. British Columbia, Vancouver, BC, Canada, 2016.
[19] M.-F. Balcan, A. Blum, and S.-T. Chen, "Diversified strategies for mitigating adversarial attacks in multiagent systems," in *Proc. AAMAS*, 2018, pp. 407–415.
[20] D. Klaška, A. Kučera, T. Lamser, and V. Řehák, "Automatic synthesis of efficient regular strategies in adversarial patrolling games," in *Proc. AAMAS*, 2018, pp. 659–666.
[21] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 6–28, Dec. 2004.
[22] L. Jin, "Reinforcement learning based energy efficient routing protocols for underwater acoustic wireless sensor networks," Ph.D. dissertation, School Elect. Electron. Comput. Eng., Univ. Western Australia, Perth, WA, Australia, 2012.
[23] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse, "Transferring knowledge of activity recognition across sensor networks," in *Proc. Int. Conf. Pervasive Comput.*, 2010, pp. 283–300.
[24] M. Winkler, K. D. Tuchs, K. Hughes, and G. Barclay, "Theoretical and practical aspects of military wireless sensor networks," *J. Telecommun. Inf. Technol.*, vol. 2, pp. 37–45, Feb. 2008.
[25] H. Dai, H. Liu, and Z. Jia, "Dynamic malicious node detection with semi-supervised multivariate classification in cognitive wireless sensor networks," *Concurrency Comput. Pract. Exp.*, vol. 27, no. 12, pp. 2910–2923, 2015.
[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[27] D. Ye and M. Zhang, "A self-adaptive sleep/wake-up scheduling approach for wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 979–992, Mar. 2018.
[28] M. Liu, K. Sivakumar, S. Omidshafiei, C. Amato, and J. P. How, "Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, 2017, pp. 1853–1860.
[29] D. Ye, M. Zhang, and D. Sutanto, "A hybrid multiagent framework with Q-learning for power grid systems restoration," *IEEE Trans. Power Syst.*, vol. 26, no. 4, pp. 2434–2441, Nov. 2011.
[30] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially private data publishing and analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, Aug. 2017.
[31] T. Zhu, P. Xiong, G. Li, W. Zhou, and P. S. Yu, "Differentially private model publishing in cyber physical systems," *Future Gener. Comput. Syst.*, Apr. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17325554
[32] Y.-X. Wang, J. Lei, and S. E. Fienberg, "Learning with differential privacy: Stability, learnability and the sufficiency and necessity of ERM principle," *J. ML Res.*, vol. 17, pp. 6353–6392, Apr. 2016.
[33] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, nos. 2–3, pp. 235–256, 2002.
[34] A. C. Y. Tossou and C. Dimitrakakis, "Algorithms for differentially private multi-armed bandits," in *Proc. AAAI*, 2016, pp. 2087–2093.
[35] T.-H. H. Chan, E. Shi, and D. Song, "Private and continual release of statistics," in *Proc. Automata Lang. Program.*, 2010, pp. 405–417.
[36] C. Zhang, V. Lesser, and P. Shenoy, "A multi-agent learning approach to online distributed resource allocation," in *Proc. IJCAI*, 2009, pp. 361–366.
[37] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Univ. Cambridge, Cambridge, U.K., 1989.
[38] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. ECML*, 2005, pp. 437–448.
[39] E. R. Gomes and R. Kowalczyk, "Dynamic analysis of multiagent Q-learning with ϵ-greedy exploration," in *Proc. ICML*, 2009, pp. 943–948.
[40] N. Cesa-Bianchi and P. Fischer, "Finite-time regret bounds for the multiarmed bandit problem," in *Proc. ICML*, 1998, pp. 100–108.
[41] M. Tokic, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences," in *Proc. 33rd Annu. German Conf. AI*, 2010, pp. 203–210.
[42] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *Proc. STOC*, Victoria, BC, Canada, 2008, pp. 609–618. [Online]. Available: http://doi.acm.org/10.1145/1374376.1374464
[43] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[44] A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
[45] P.-J. Wan and C.-W. Yi, "Coverage by randomly deployed wireless sensor networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2658–2669, Jun. 2006.
[46] A. Hernando, D. Villuendas, C. Vesperinas, M. Abad, and A. Plastino, "Unravelling the size distribution of social groups with information theory in complex networks," *Eur. Phys. J. B*, vol. 76, no. 1, pp. 87–97, 2010.

Dayong Ye received the M.Sc. and Ph.D. degrees in computer science from the University of Wollongong, Wollongong, NSW, Australia, in 2009 and 2013, respectively.

He is currently a Research Fellow of cybersecurity with the University of Technology Sydney, Ultimo, NSW, Australia. His current research interests include privacy preserving, artificial intelligence, and cloud computing.

Tianqing Zhu (M'11) received the B.Eng. degree in applied chemistry and the M.Eng. degree in automation from Wuhan University, Wuhan, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science from Deakin University, Geelong, VIC, Australia, in 2014.

She was a Lecturer with the School of Information Technology, Deakin University, Geelong, VIC, Australia, from 2014 to 2018. She is currently a Senior Lecturer with the School of Software, University of Technology Sydney, Ultimo, NSW, Australia. Her current research interests include privacy preserving, data mining, and network security.

Wanlei Zhou received the B.Eng. and M.Eng. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1982 and 1984, respectively, the Ph.D. degree in computer science from Australian National University, Canberra, ACT, Australia, in 1991, and the D.Sc. degree (a higher Doctorate degree) from Deakin University, Geelong, VIC, Australia, in 2002.

He is currently the Head of the School of Software, University of Technology Sydney, Ultimo, NSW, Australia. He has published over 400 papers in refereed international journals and refereed international conferences proceedings, including many articles in IEEE TRANSACTIONS and journals. His current research interests include security and privacy, bioinformatics, and e-learning.

Philip S. Yu (F'93) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, and the M.B.A. degree from New York University, New York, NY, USA.

He is a Distinguished Professor of computer science with the University of Illinois at Chicago, Chicago, IL, USA, and also holds the Wexler Chair in Information Technology. He has published over 1000 papers in refereed journals and conferences. He holds or has applied for over 300 U.S. patents. His current research interest includes big data, including data mining, data stream, database and privacy.

Dr. Yu is the Editor-in-Chief of the *ACM Transactions on Knowledge Discovery From Data*. He is a fellow of ACM.