# A novel differentially private advising framework in cloud server environment

Sheng Shen | Tianqing Zhu | Dayong Ye | Minghao Wang | Xuhan Zuo | Andi Zhou

School of Computer Science, Center for Cyber Security and Privacy, University of Technology Sydney, Ultimo, New South Wales, Australia

**Correspondence**
Tianqing Zhu, School of Computer Science, Center for Cyber Security and Privacy, University of Technology Sydney, 15 Broadway, Ultimo NSW 2007, Australia.
Email: tianqing.zhu@uts.edu.au

**Funding information**
Australia Research Council, Grant/Award Number: LP170100123

**Summary**
Due to the rapid development of the cloud computing environment, it is widely accepted that cloud servers are important for users to improve work efficiency. Users need to know servers' capabilities and make optimal decisions on selecting the best available servers for users' tasks. We consider the process of learning servers' capabilities by users as a multiagent reinforcement learning process. The learning speed and efficiency in reinforcement learning can be improved by sharing the learning experience among learning agents which is defined as advising. However, existing advising frameworks are limited by the requirement that during advising all learning agents in a reinforcement learning environment must have exactly the same actions. To address the above limitation, this article proposes a novel differentially private advising framework for multiagent reinforcement learning. Our proposed approach can significantly improve the application of conventional advising frameworks when agents have one different action. The approach can also widen the applicable field of advising and speed up reinforcement learning by triggering more potential advising processes among agents with different actions.

**KEYWORDS**
advising, cloud servers, differential privacy, reinforcement learning

## 1 | INTRODUCTION

Cloud computing, a term coined in late 2007, has developed over the past decade but is currently still a hot topic for research due to its ability to offer flexible dynamic IT infrastructures, QoS guaranteed computing environments and configurable software services. According to Wang et al,[1] who proposed an early definition of cloud computing in 2008, a computing cloud is a set of network enabled services, providing scalable, QoS guaranteed normally personalized, inexpensive computing platforms on demand, which can be accessed in a simple and pervasive way. The current three service models in cloud computing are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).[2] A cloud server is a powerful physical or virtual infrastructure that executes application and information processing and storage. Cloud servers are created using virtualization software to divide a physical server into multiple virtual servers using the IaaS model of cloud computing to process workloads and store information. Users can access virtual server functions remotely through an online interface. Cloud servers provide users stability and security because software problems are isolated from users' environments. Also, cloud servers are stable, fast, and secure as they can avoid hardware limitations. Nowadays, more and more cloud server providers are emerging, that is, A2 Hosting and InMotion. Users select providers and services according to their requirements, such as high processing performance or large storage space.[3,4] The service selection problem can be handled using reinforcement learning.

Reinforcement learning (RL) is widely used today to autonomously learn how to solve sequential decision-making problems interacting with the system environment.[5,6] Regular RL approaches need a large number of interactions with the system environment to learn a policy.[7] Advising

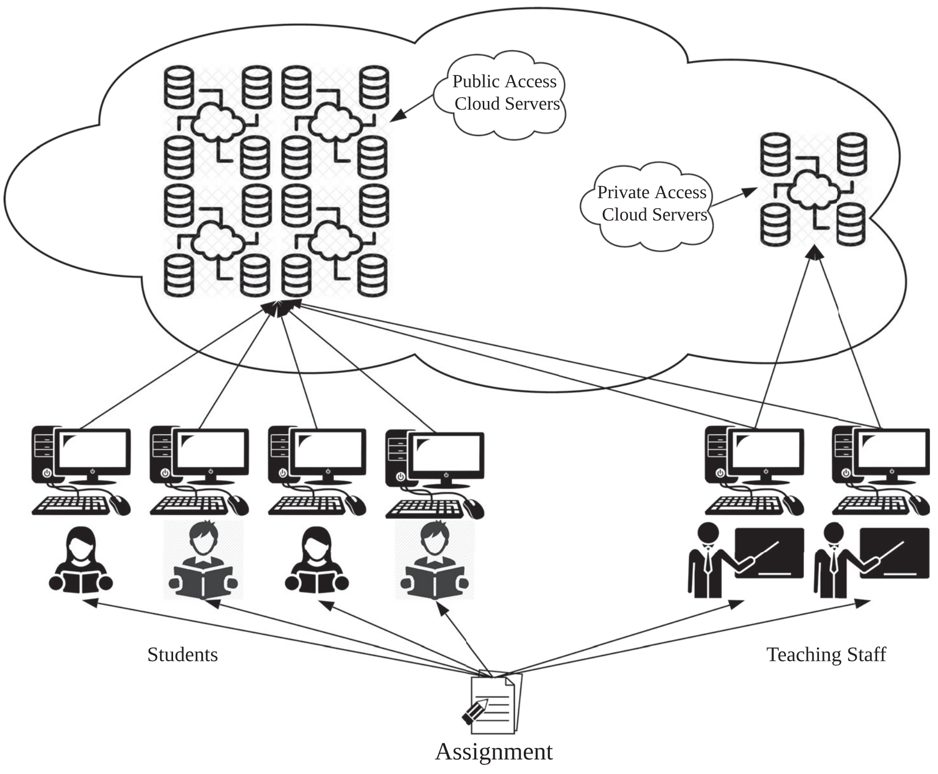

**FIGURE 1** A real-world example: all students and teaching staff of a course need to execute their assignments on a cloud computing servers, but they have different choices available to them

frameworks address the above problem by reusing previous knowledge in a repeated state from other agents. Simultaneously learning agents advising can be used to accelerate learning when all agents start learning in a multiple-state system at the same time.[8-11] An agent's available choices under this state are called actions. Some of learning agents who have visited a state more times than others, can be seen as more experienced in. A single agent can play both roles of teacher who can provide advice, and student who will ask for advice.

Figure 1 shows a real-world example, one assignment in a cloud computing course in a university. Both students and teaching staff need to execute this assignment on the university's cloud servers, but students can only access public servers, and staff can access either public servers or private servers. In this example, the experienced students who have already executed their assignments more times than others can provide advice to other students in choosing the optimal server with the best capability to run their assignments within traditional advising frameworks, and the same applies to staff.

However, traditional advising frameworks are only appropriate for situations in which all agents have the same actions and cannot be applied when agents' actions are not exactly the same. In the real world, it is common that agents have different available actions even though they are solving the same problem.[11] Referring back to Figure 1, considering traditional frameworks, experienced teaching staff cannot advise students which server has a better capability and a better performance due to different available choices for servers even though they are working on the same assignment. Therefore, it is crucial to solve the problem of advising among agents with different available actions. Little previous work has considered experience transfer between two agents with different available actions, and the challenge is how to transfer the experience for the actions which are different among teachers and students.

We find that a differential privacy mechanism can address the above challenge through the property of randomization. We consider all agents' available actions and their experience on each state as independent data sets. The two data sets can be considered as neighboring data sets in terms of differential privacy. In Figure 1, if teaching staff and students record the execution time of tasks as their experience on selecting servers, these records for servers can be seen as data sets. The less time means the better capability of servers. When there is only one only one server accessible only to teaching staff, teachers' data sets have one more record than students.' In this case, the data sets of teachers and students can be considered as neighboring data sets. By adding randomization, these data sets can be considered as the same in a defined scale, and teaching staff and students can share experience with each other. The differential privacy property in our work is to guarantee that the experience for each action can still be used as advice if the two actions differ in, at most, one record. This approach can expand the applicable field of advising frameworks and increase the probability of the occurrence of advising.

We have three main contributions in this article.

1. First, we propose an improved differentially private advising approach based on our previous framework, to improve conventional advising frameworks to address the problem that agents' actions are different.

2. Second, we widen the applicable field of conventional advising frameworks in reinforcement learning by allowing more possible advising to accelerate agents' learning stage in RL.
3. Third, we propose a new approach to decide when advising happens by calculating the Euclidean distance of teacher and student's $Q$-value, $Q(s)$.

We design a comprehensive experiment to demonstrate our approach in simultaneously multiagent RL framework by comparing with conventional advising frameworks under the same setting. We provide a brief analysis of the convergence of agents' learning and why Euclidean distance works here. We also present our experiment results and the analysis of the performance of differential privacy in the method by changing the value of privacy budget, $\epsilon$, in the range from 0.1 to 1.

## 2 | PROBLEM STATEMENT

### 2.1 | Scenario setting

In this section, we describe a detailed scenario as a motivated case study to demonstrate existing challenges we are addressing. The scenario shown in Figure 2 is an extension of above example in a cloud environment. There are many users, $a, b, c, d, \dots$, whom we consider as **agents** participating in a RL environment. There are many tasks, $1, 2, 3, 4, \dots$ can treat as the **states** $s_1, s_2, s_3, s_4, \dots$ in the set of states $S$. Each user is allocated to a random task each time, we call a single cycle for a user to finish its task an *epoch*, from being allocated to a random task to the time before being allocated to the next task. Cloud servers $A, B, C, D, \dots$ are used to solve tasks for users. In this RL environment, servers are actions taken by agents to solve tasks. In our scenario, each agent can access different servers and compose a set of available actions, $A$, that is, for agent $a$, $A_a(S)$:$A, B, C, D$. Each agent can only take one action for each task each epoch. However, each server behaves differently on solving different tasks, and servers behave differently on the same task, that is, server "A" is the best server on executing task 2 with the lowest time cost but the worst on task 4, or server "C"s computation capability is not enough to solve task 3. Combining all these factors, a user obtains a "feedback" to evaluate how this server executes this task, which we call reward $r(s,a)$, meaning the reward from the action $a$ at the state $s$. A higher reward means a server has a better capability on solving the task, and vice versa. Therefore, the aim of users is to solve the task with the optimal server with the maximum reward each epoch.

### 2.2 | Scenario discussion

In the above scenario, each user should repeatedly be allocated tasks and choose servers to solve them. The process can be considered as a learning process in which users need to learn capabilities of servers on solving tasks. We explore the problems of how users can learn servers' capabilities, whether users can share their experience to others during the learning process, and how users share the experience to others even they have different available choices.

To address these questions, each agent needs to know each cloud server's capabilities based on their previous sequence of learning experience, in order to make the best decision next time being allocated to the same tasks. Users continuously expand their knowledge by receiving the rewards

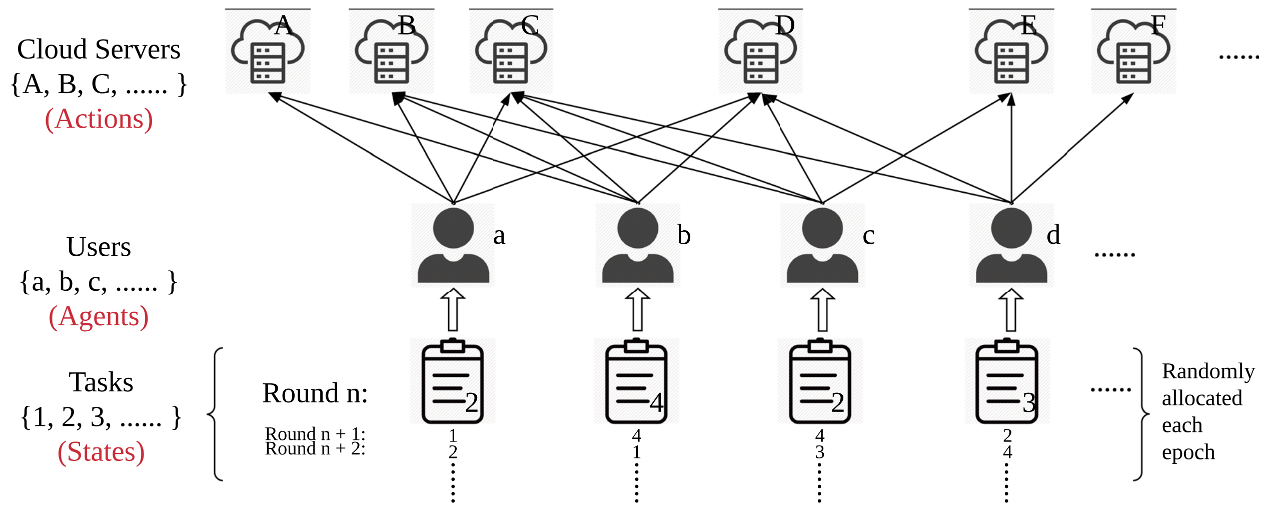

**FIGURE 2** Overview of scenario setting

from previous epochs, and use learned knowledge to make the next decision. The knowledge determines probability distribution for each agents' available actions in each state. The agent should "simunderstand" this reward and update their probability distribution for available actions at states by a transition function $T$, in order to make a better decision next time when they are allocated to this task.

Agents can transfer experience to or ask experience from others to accelerate the learning process. However, due to the randomness of the task allocation to simultaneously learning agents, the above scenario is very likely to generate a knowledge gap at some state. The knowledge gap is an experience gap at a state, based on difference of times that an agent is allocated to a task with others. An agent can be considered as more experienced than another on a task if this agent has met this task enough times more than the other. A threshold is used to define "simmore experienced" with a minimum number of knowledge gap. An agent can ask for advice from the experienced agent while being allocated to this task again. However, agents will inevitably run into due to the limitation we mentioned above, that is, traditional advising frameworks only focus on knowledge transfer among agents with the same actions. For instance, referring back to Figure 2, agent $a$'s available actions are $A_a$: $\{A,B,C,D\}$, and agent $c$'s are $A_c$: $\{B,C,D,E\}$. In conventionally proposed advising frameworks, agents can only advise other agents with the same actions. In other words, agent $a$ and $c$ cannot advise each other due to the difference in their actions sets. Therefore, we propose a differentially private advising framework to break this limitation. Our approach allows agents to advise in the above scenario when RL agents have different available actions, which will accelerate their learning.

## 3 | PRELIMINARY

### 3.1 | Reinforcement learning

Reinforcement learning (RL) techniques are usually used to solve sequential decision-making problems, which model the learning environment as a Markov decision processes (MDP). An MDP model can be depicted as a tuple $\langle S, A, T, R\rangle$, where $S$ is the set of possible environment states, $A$ is the set of agents' available actions, $T$ is the probability of transition to next state $s'$ while performing action $a$ in state $s$, and $R$ is the reward function for agents defining the reinforcement received from performing actions. At each learning step, an agent chooses an action after observing the state $s$, and then the next state is defined by $T$ and a reward signal $r$ can be observed. The aim of RL task for the agent is to find an optimal policy $\pi : S \rightarrow A$, which maps the best action for each possible state.

We focus on RL algorithms with $Q$-value estimates to determine the agent's policy, which is used to map every combination of state and action to an estimate of the long-term reward starting from the current state-action pair: $Q{:}S \times A \rightarrow R$. The policy, $\pi^*(s) = \text{argmax}_a Q^*(s,a)$, is optimal if the $Q$-value is accurate for this policy. The update of the $Q$-value during RL process based on experiences with the MDP. An *experience* can be described by a quadruple $\langle s, a, r, s'\rangle$ where action $a$ is taken in state $s$, resulting in reinforcement reward $r$ and a transition to next state $s'$. The update of $Q$-value follows the rule as: $Q(s,a) \leftarrow Q(s,a) + \alpha[R(s,a) + \gamma Q(s',a') - Q(s,a)]$, where $\alpha$ is learning rate, $\gamma$ is a discount factor for next states, $a$ and $a' \in A$ are the current and next action, respectively, and $s$ and $s' \in S$ are the current and next state, respectively.

### 3.2 | Advising in reinforcement learning

Supposing that an agent has learned an optimal policy $\pi$ for a specific task and become experienced, it can teach another agent which is beginning to learn the same task using this fixed policy. As the student learns, the teacher will observe each state $s$ the student encounters and each action $a$ taken by the student. In $n$ of these states, the teacher can advise the student to take the "simcorrect" action $\pi(s)$ with its own learning experience or policy. The teacher-student framework aims to accelerate a student's training process. However, a budget $b$ can limit the advising process. The teacher cannot provide further advice when the budget $b$ is spent. Therefore, it is critical to define when to give advice to accelerate student's learning.

### 3.3 | Differential privacy

Differential privacy is a provable privacy notation provided by Dwork et al.[12] It provides a strong privacy guarantee that the outputs of the queries on the neighboring data sets will be statistically similar. The formal definition of differential privacy is presented as follows:

**Definition 1** ($\varepsilon$-differential privacy). A randomized algorithm $\mathcal{M}$ gives $\varepsilon$-differential privacy for any pair of *neighboring data sets* $D$ and $D'$, and for every set of outcomes $\Omega$, $\mathcal{M}$ satisfies:

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

$\varepsilon$ is the privacy parameter, which is defined as the *privacy budget*.[13,14] It controls the privacy preservation level. A smaller $\varepsilon$ represents a greater privacy.

**Definition 2** (neighboring data set). The data sets $D$ and $D'$ are neighboring data sets if and only if they differ in one record. This is denoted as $D \oplus D' = 1$.

"$\oplus$" indicates the difference between two data sets.

**Definition 3** (sensitivity). For a query $Q : D \rightarrow \mathbb{R}$, the global sensitivity of $Q$ is defined as follow:

$$S = \max_{D,D'} \|Q(D) - Q(D')\|_1. \tag{2}$$

**Definition 4** (*Laplace* mechanism). Given a function $f : D \rightarrow \mathbb{R}$ over a data set $D$, Equation (4) provides $\varepsilon$-differential privacy.

$$\hat{f}(D) = f(D) + Laplace\left(\frac{S}{\varepsilon}\right). \tag{3}$$

A Laplace mechanism is used for numeric output. Differential privacy is achieved by adding *Laplace* noise to the true answer.[15-17]

## 4 | DIFFERENTIALLY PRIVATE ADVISING FRAMEWORK

We have proposed a differentially private advising framework by combining reinforcement learning, teacher-student advising framework, differential privacy mechanism, and Euclidean distance together. Figure 3 is the overview of our proposed method. An reinforcement learning agent $j$ wants to update its probability distribution by obtaining a reward from an action. Any reason leading to the failure of advising results in the fact that agent $j$ has to choose the action alone based on its own probability distribution. The agent $j$ has a communication budget $b_{ask}$ to control its communication overhead and each successful advising process costs some communication budget from it. $b_{ask}$ is initialized with a constant $C$, and each ask consumes 1 from the budget. When it runs out the budget $b_{ask}$, it leads to the failure of advising. Agents can trigger advising with their neighboring agents whose actions differ in at most one record with the agent. When $b_{ask}$ >0, agent $j$ check if its neighboring agents have communication budget $b_{give}$ for giving advice. $b_{give}$ is also initialized with a constant $C$, and each advice consumes 1 from the budget. If all neighboring agents run out their $b_{give}$, the advising process fails. If some of neighboring agents' $b_{give} > 0$, $j$ starts to calculate the Euclidean distance with each available neighboring agent's $Q$-value. None of Euclidean distance results are greater than a threshold $d$ will lead to the failure of advising process. If one or more than one Euclidean distance is greater than $d$, the agent with the maximum distance with $j$ will become $j$'s teacher to choose an action for $j$ for further updating $Q$-value and probability distribution in this epoch. We will introduce details and algorithms for each part in following subsections.

### 4.1 | Reinforcement learning and normalization

RL algorithm is the base of our proposed method, which is given in Algorithm 1. For each epoch, an agent $j$ selects an action $a_k \in A(j)$ based on its probability distribution over the available actions at state $s$. The agent $g$ obtains a reward $r$ by taking this action and uses this reward to update its $Q$-value of the action $a_k$ at state $s$, which this update is based on (i) $Q(s,a_k)$, the current value of $Q(s,a_k)$, (ii) max $Q(s', a)$, the maximum $Q$-value of an action at the next state $s'$, (iii) $\gamma$, a discount rate to control the effect of max $Q(s', a)$ on update this time, (4) $\alpha$, a learning rate to control the speed of updating $Q$-value and learning, and (5) the reward $r$. An agent should to adjust its probability distribution using the updated $Q$-value. Similar to the

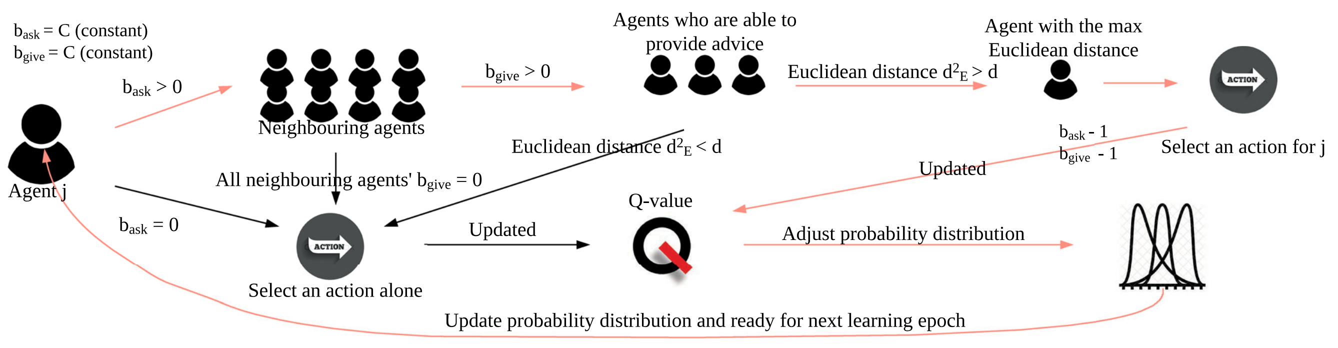
b_ask = C (constant)
b_give = C (constant)
b_ask > 0
Agent j
Neighbouring agents
b_give > 0
Agents who are able to provide advice
Euclidean distance d²_E > d
Agent with the max Euclidean distance
ACTION
b_ask - 1
b_give - 1
Select an action for j
Euclidean distance d²_E < d
All neighbouring agents' b_give = 0
Updated
Q-value
b_ask = 0
ACTION
Select an action alone
Updated
Adjust probability distribution
Update probability distribution and ready for next learning epoch

**FIGURE 3** Overview of framework

process of updating $Q$-value, the agent updates its probability distribution $\pi(s)$ for each action based on the current $\pi(s,a)$, a learning rate $\xi$, and $\bar{r}$ which is the sum of the product of each action's probability and $Q$-value. The update of $\pi(s)$ can only show the trend of experience accumulation and reflect the relationship between each action's $Q$-value; therefore, we require normalization function shown in Algorithm 2 to adjust the probability distribution $\pi(s)$ to satisfy the requirement that the sum of $\pi(s)$ is 1 while keeping the ratio relationship.

**Algorithm 1.** Reinforcement learning

```
Require: Taking agent $j$ at state $s$ as an example
Require: Initialize probability distribution, $Q$-value for each actions
1:  repeat for each epoch
2:      Agent $j$ chooses an action $a_k$, based on the probability distribution: $\pi(s) = \langle \pi(s,a_1), \dots, \pi(s,a_n) \rangle$;
3:      $r \leftarrow R\langle s, a_k \rangle$;
4:      $Q(s,a_k) \leftarrow (1-\alpha) Q(s,a_k) + \alpha \left[ r + \gamma \max_a Q(s\prime, a) \right]$;
5:      $\bar{r} \leftarrow \Sigma_{a \in A(s)} \pi(s,a) Q(s,a)$;
6:      for $a \in A_{(s)}$ do
7:          $\pi(s,a) \leftarrow \pi(s,a) + \xi (Q(s,a) - \bar{r})$;
8:      end for
9:      $\pi(s) \leftarrow Normalization(\pi(s))$;
10:     $s \leftarrow s\prime$;
11: until finish learning
```

**Algorithm 2.** Normalization

```
Require: Taking agent $j$ with $n$ available actions
Require: $d = min_{1 \le k \le n} p(k)$, mapping centre $c_o = 0.5$ and mapping lower bound $\Delta = 0.001$;
1:  if $d > \Delta$ then
2:      $\rho \leftarrow \frac{c_o - \Delta}{c_o - d}$;
3:      for $k = 1$ to $n$ do
4:          $p(k) \leftarrow c_o - \rho(c_o - p(k))$;
5:      end for
6:  end if
7:  $\Pi \leftarrow \Sigma_{1 \le k \le n} p(k)$;
8:  for $k = 1$ to $n$ do
9:      $p(k) \leftarrow \frac{p(k)}{\Pi}$;
10: end for
11: return $p$
```

## 4.2 | When to ask for and give advice

In this section, we introduce when to ask for and give advice and why the differential privacy mechanism is appropriate here. In our advising environment, agents' actions are not private information, which means every agent knows their neighbors and neighbors' actions. Therefore, neighboring relationship has been defined for each client to trigger advising processes. We first present the convergence proof for $Q$-value in our case as a base, and explain why the vector $Q(s,a)$ and our queries on the vector satisfy differential privacy. The solution we used to decide the timing of advising is calculating Euclidean distance $d_E^2(teacher, student)$ between the student's $Q$-value vector and its optimal teacher's $Q$-value vector. A threshold $d$ is set that when the distance $d_E^2(teacher, student) > d$ we consider the teacher to be sufficiently more experienced than the student to be able to advise them.

### 4.2.1 | The convergence of Q-value

Much of the previous research on the proof of $Q$-value has been exploratory in nature. We refer readers who are interested in the proof to References 18-20. In our case, due to the discreteness of our states and actions, it is not necessary to think about the optimal $Q(s', a')$ from the next state

$s'$ while updating current state's $Q(s,a)$, or simply let $Q(s',a')=0$. The equation of updating $Q$-value is transformed to

$$\begin{aligned} Q(s,a) &\leftarrow Q(s,a)+\alpha\cdot(r(s,a)-Q(s,a)) \\ &\leftarrow (1-\alpha)\cdot Q(s,a)+\alpha\cdot r(s,a). \end{aligned}$$

*Claim* 1. Assume that an agent has selected the action $a$ whose reward is $r(s,a)$, at the state $s$ enough times, and updated its $Q$-value $Q(s,a)$ until full convergence, we have

$$\lim_{t\to+\infty} Q(s,a)=r(s,a).$$

*Proof*. Let $q=Q(s,a)$, $r=r(s,a)$, and $t$ is the index of times for updating $q$, so, $q_t$ is the current value of $Q(s,a)$. When the $(t+1)$th time to update $q$, we have

$$q_{t+1}=(1-\alpha)\cdot q_t+\alpha\cdot r,$$

where $\alpha$ is learning rate in the range $0<\alpha<1$. We have

$$\begin{aligned} q_{t+1}-r &= (1-\alpha)\cdot q_t+\alpha\cdot r-r \\ &= (1-\alpha)\cdot q_t-(1-\alpha)\cdot r \\ &= (1-\alpha)\cdot(q_t-r). \end{aligned}$$

So we have

$$\frac{q_{t+1}-r}{q_t-r}=1-\alpha.$$

So $(q_t-r)$ is a geometric progression with the common ratio $(1-\alpha)$, we have

$$q_t=(q_0-r)\cdot(1-\alpha)^t+r.$$

Because when $0<\alpha<1$

$$\lim_{t\to+\infty}(1-\alpha)^t=0.$$

So we have

$$\begin{aligned} \lim_{t\to+\infty} q_t &= (q_0-r)\cdot 0+r \\ &= r. \end{aligned}$$

■

### 4.2.2 | Differential privacy

Assume that all agents have learned enough times, and $Q(s,a)$ and its probability distribution are fully convergent, for the state $s$ and the action $a$, the value of $Q(s,a)$ is fixed and $Q(s,a)=r(s,a)$ for each agent. Consider a single agent, $i$'s $Q$-value, who has $n$ available actions, $Q(s)$: $\{Q(s,a_1),Q(s,a_2),\ldots,Q(s,a_n)\}$. We use a table to record $i$'s $Q(s)$ value as a data set as shown in Table 1. Similarly, an agent $j$'s $Q(s)$ value as shown in Table 2. When $Q(s,a)$ values of agent $i,j$ are fully convergent, Tables 1 and 2 is then transformed to Tables 3 and 4, separately. The difference between Tables 3 and 4 has been highlighted where agent $i$ has a different action $a_k$ at the state $s$ from agent $j$'s action $a_k^*$, and accordingly different convergent $Q(s,a_k)$ and $Q(s,a_k^*)$. According to Definition 2, we consider two data sets in Tables 3 and 4 are neighboring data sets, which satisfy differential privacy mechanism. Because $Q(s)$ maps the probability distribution $\pi(s)$, we consider $\pi(s)$ is a query on the data set $Q(s)$. We define the query $f(Q(s))$ is the process of calculating $\pi(s)$ from $Q(s)$. According to Definition 1, existing a privacy budget $\varepsilon$ and a randomized algorithm $\mathcal{M}$ satisfy

$$\Pr[\mathcal{M}(Q_i(s))\in\Omega]\le\exp(\varepsilon)\cdot\Pr[\mathcal{M}(Q_j(s))\in\Omega].$$

The randomized algorithm $\mathcal{M}$ is Laplace mechanism in our case, which satisfies

$$\hat{f}(Q_i(s))=f(Q_i(s))+Laplace\left(\frac{S}{\varepsilon}\right),$$

**TABLE 1** Agent *i*'s data set of *Q*-value

| Action | Q(*s*) |
|---|---|
| $a_0$ | $Q(s,a_0)$ |
| $a_1$ | $Q(s,a_1)$ |
| $\vdots$ | $\vdots$ |
| $a_k$ | $Q(s,a_k)$ |
| $\vdots$ | $\vdots$ |
| $a_n$ | $Q(s,a_n)$ |

**TABLE 2** Agent *j*'s neighboring data set of *Q*-value

| Action | Q(*s*) |
|---|---|
| $a_0$ | $Q(s,a_0)$ |
| $a_1$ | $Q(s,a_1)$ |
| $\vdots$ | $\vdots$ |
| $a_k^*$ | $Q(s,a_k^*)$ |
| $\vdots$ | $\vdots$ |
| $a_n$ | $Q(s,a_n)$ |

**TABLE 3** Agent *i*'s convergent *Q*-value

| Action | Q(*s*) |
|---|---|
| $a_0$ | $r(s,a_0)$ |
| $a_1$ | $r(s,a_1)$ |
| $\vdots$ | $\vdots$ |
| $a_k$ | $r(s,a_k)$ |
| $\vdots$ | $\vdots$ |
| $a_n$ | $r(s,a_n)$ |

**TABLE 4** Agent *j*'s convergent *Q*-value

| Action | Q(*s*) |
|---|---|
| $a_0$ | $r(s,a_0)$ |
| $a_1$ | $r(s,a_1)$ |
| $\vdots$ | $\vdots$ |
| $a_k^*$ | $r(s,a_k^*)$ |
| $\vdots$ | $\vdots$ |
| $a_n$ | $r(s,a_n)$ |

where the sensitivity *S* satisfies

$$S = \max_{f(Q_i(s)),f(Q_j(s))} \|f(Q_i(s)) - f(Q_j(s))\|_1.$$

### 4.2.3 | Euclidean distance

We use Euclidean distance to calculate the difference of *Q*-value between teacher and student's. Euclidean distance is most commonly used to calculate the distance between two vectors in a Euclidean space due to its simplicity. Let *x*, *y* be two N-dimension vectors and $x=(x^1, x^2, \dots, x^N)$, $y=(y^1, y^2, \dots, y^N)$. The Euclidean distance $d_E^2(x,y)$ is given by

$$d_E^2(x,y) = \sum_{i=1}^{N}(x^i - y^i)^2. \tag{4}$$

We set a threshold *d* that when the distance $d_E^2(teacher, student) > d$, we consider the teacher to be sufficiently more experienced than the student to be able to provide advice. We have presented the convergence proof of *Q*(*s*,*a*) in our setting in the above subsection. *Q*(*s*,*a*) is convergent to $r(s,a)$, so two agents can learn the same knowledge when they are in the completely same setting. $d_E^2(teacher, student)$ is the knowledge gap that indicates how much more the teacher has learned compared with the student.

**Algorithm 3.** Differentially private advising framework

```
Require: Taking student agent $j$ and teacher agent $i$ at state $s$ as an example
Require: Initialize probability distribution, $Q$-value for each actions
Require: $b_{\text{give}} = c$ and $b_{\text{ask}} = c$ for each agent, $c$ is a constant
Require: Sensitivity $S$ and privacy budget $\epsilon$
Require: The distance threshold $d$
 1: function EUCLIDEAN DISTANCE(i, j)
 2:     if $A(i) == A(j)$ then
 3:         $d_E^2(i,j) = \sum_{p=1}^{len(Aj)}(Q_i(s,a_p) - Q_j(s,a_p))^2$
 4:     else if $len(A(i)) < len(A(j))$ then
 5:         $d_E^2(i,j) = \sum_{p=1}^{len(Aj)}[(Q_i(s,a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s,a_p)]^2$
 6:     else if $len(A(i)) > len(A(j))$ then
 7:         $a_{diff}$: the action in $A(j)$ but not in $A(i)$
 8:         $d_E^2(i,j) = \sum_{p=1}^{len(Aj)}[(Q_i(s,a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s,a_p)]^2 + (Q_j(s,a_{diff}) + Laplace(\frac{S}{\epsilon}))^2$
 9:     else if $len(A(i)) == len(A(j))$ and $A(i)! = A(j)$ then
10:         $a_{diff1}$: the action in $A(j)$ but not in $A(i)$
11:         $a_{diff2}$: the action in $A(i)$ but not in $A(j)$
12:         $d_E^2(i,j) = \sum_{p=1}^{len(Aj)-1}[(Q_i(s,a_p) + Laplace(\frac{S}{\epsilon})) - Q_j(s,a_p)]^2 + [(Q_i(s,a_{diff2}) + Laplace(\frac{S}{\epsilon})) - Q_j(s,a_{diff1})]^2$
13:     end if
14:     return $d_E^2(i,j)$
15: end function
16: repeat for each epoch
17:     if $b_{\text{ask}}(j) < 0$ then
18:         $i = [\text{argmax}_{i \in |Neig|}(EuclideanDistance(i,j))$ and $b_{\text{give}}(i) < 0]$
19:         if $d_E^2(i,j) < d$ then
20:             $\bar{r} \leftarrow \Sigma_{a \in A(s)} \pi(s,a)(Q_i(s,a) + Laplace(\frac{S}{\epsilon}))$;
21:             for $a \in A_j(s)$ do
22:                 $\pi\prime(s,a) \leftarrow \pi(s,a) + \xi(Q_i(s,a) + Laplace(\frac{S}{\epsilon}))$;
23:             end for
24:             $i$ selects an action $a_k$ for $j$, based on the new probability distribution: $\pi\prime(s) = \langle \pi\prime(s,a_1), \ldots, \pi\prime(s,a_n) \rangle$;
25:             $r \leftarrow R\langle s, a_k \rangle$;
26:             $Q(s,a_k) \leftarrow (1-\alpha)Q(s,a_k) + \alpha\left[r + \gamma \max_a Q(s\prime, a)\right]$;
27:             $\bar{r} \leftarrow \Sigma_{a \in A(s)} \pi(s,a) Q(s,a)$;
28:             for $a \in A_{(s)}$ do
29:                 $\pi(s,a) \leftarrow \pi(s,a) + \xi(Q(s,a) - \bar{r})$;
30:             end for
31:             $\pi(s) \leftarrow Normalization(\pi(s))$;
32:             $s \leftarrow s\prime$;
33:             $b_{\text{ask}}(j) - 1$;
34:             $b_{\text{give}}(i) - 1$
35:         else Algorithm 1
36:         end if
37:     else Algorithm 1
38:     end if
39: until finish learning
```

### 4.3 | Algorithm of differentially private advising framework

Algorithm 3 is the algorithm of our proposed differentially private advising framework. In this algorithm, we assume the agent *j* as a student and the agent *i* as a teacher, and set the distance threshold *d* to trigger the advising process. We include Euclidean distance, differential privacy mechanism and teacher-student advising framework into the reinforcement learning algorithm. We first propose the function Euclidean distance with differential privacy to decide the timing of advising. There are four possible situations involved in our scenario to calculate Euclidean distance between *i* and *j*'s *Q*-value. First, when two agents have identical actions, they directly calculate Euclidean distance based on their original *Q*(*s*). In the second situation, when the teacher has one action more than the student, we add *Laplace* noise onto the teacher's *Q*(*s*) and only calculate the Euclidean distance of same actions' *Q*-value between teacher and student. Third, when the teacher has one action fewer than the student, we temporarily add the missing action to the teacher with an initial value of *Q*-value, normally 0. Following this, we add *Laplace* noise onto the teacher's new *Q*-value including the additional action to calculate Euclidean distance with the student's *Q*(*s*). The last situation is when the teacher and student have the same number of actions but one of them is different. In this situation, we temporarily consider the teacher's different action as the student's action differing with the teacher, and then add Laplace noise on the teacher's new *Q*(*s*) to calculate Euclidean distance with the student's *Q*(*s*) normally. All of these four situations satisfy the definition of neighboring data set, and are allowed within our proposed framework. This is how we calculate Euclidean distance to determine when advising happens.

We include the function *Euclideandistance*(*i*,*j*) in the main part our algorithm to determine who the teacher agent is. While the agent *j* has not depleted its ask budget $b_{\text{ask}}$, it can ask advice from its "neighboring agents." In the line 18, |*Neig*| is the data set of agent *j*'s neighboring agents who can provide advice to *j*. The agent *j* first checks its neighboring agents to determine whether they still have remaining budget $b_{\text{give}}$, and then calculates the Euclidean distance with every neighbor with nonzero $b_{\text{give}}$. The teacher *i* must satisfy the following three requirements: (i) *i* does not deplete its $b_{\text{give}}$, (ii) the Euclidean distance between *i* and *j* is the greatest among all neighboring agents, and (iii) their Euclidean distance is greater than the threshold *d*. If existing an agent *i* satisfies these three requirements, it can select an action based on its new probability distribution $\pi'(s)$ transitioned from a new *Q*(*s*) during the function *Euclideandistance*(*i*,*j*) (lines 20 to 24). Remarkably, the *i*'s noised *Q*-value is only involved in the Euclidean distance calculation and selecting action for the student, but does not update *i*'s original *Q*-value and affect its learning process. The student agent *j* then gets the reward from this action and then adjusts the *Q*(*s*,*a*) and probability distribution. The advising process costs 1 for student's budget $b_{\text{ask}}$ and teacher's budget $b_{\text{give}}$. If no neighboring agent meets above three requirements, or the agent *j* runs out its asking budget $b_{\text{ask}}$, the agent *j* then proceeds with traditional reinforcement learning as in Algorithm 1.

In general, our algorithm allows more advising processes to occur while guaranteeing the original teacher-student advising framework runs normally in a reinforcement learning system. We use the differential privacy mechanism to eliminate the difference between two agents' actions in a reasonable bound, which facilitates more effective advising with limited communication budgets.

## 5 | EXPERIMENTS

### 5.1 | Experiment setting

In order to demonstrate our method intuitively, we have designed experiments with the scenario introduced in Section 5. There are three parts of experiment. In the first experiment, we investigate the following three strategies:

1. *Traditional reinforcement learning (no advice)*: As reference, we evaluate the SARSA learning algorithm without advising;
2. *Traditional advising framework*: As reference, we evaluate the traditional teacher-student advising framework. Advising only happens between two agents with completely the same actions.
3. *Differentially private advising framework*: Our proposed method. Advising process occurs between two agents with either the same actions or differing by one. This strategy applies differential privacy, which means that students accept teachers knowledge with additional noise to adjust their own knowledge influencing action

We involve our approach and two compared strategies together in experiment 1 via the controlled variable method, where each strategy has the same number of users, servers, and tasks. We also set the same learning rate, rewards, and initial *Q*-value for each action to guarantee the total knowledge is the same among every strategy. The only different setting in our proposed method is that each agent has one different action, (ie, $A(a_1)$: $\{A, B, C, D\}$, $A(a_2)$: $\{A, B, C, E\}$, $A(a_3)$: $\{A, B, D, E\}$, …), but each agent's actions are the same for the other two strategies. The purpose of this experiment is to demonstrate that our method can allow advising to occur between two agents with one different action. The expectation is that our algorithm can have a same performance with traditional advising framework.

Our second experiment is designed to demonstrate the second contribution that our proposed approach can widen the traditional advising framework. We compare two approaches' performance in a completely same setting. In this reinforcement learning system, only a subset of agents have the same available actions and others have one different actions among each other. In this case, only a small amount of agents with the same actions can advise each other in traditional advising framework, but all of agents can advise their neighboring agents in our proposed method. In theory, our method can trigger more advising processes to positively accelerate convergence. We use the traditional advising framework as the base to demonstrate how much performance our differentially private advising framework can improve. In the third experiment, we explore how the privacy budget $\varepsilon$ affect the differential privacy performance in our algorithm. We set the epsilon as 0.1, 0.4, 0.7, 1, and 2 separately.

The comparison criteria is convergence ratio, the accuracy of agents' learning, which is described by the following equation:

$$\frac{\sum_{k\in A}^{n} \pi(s,a_k) * R(s,a_k)}{\sum_{m\in A}^{n} \pi(s,a_m) * R(s,a_m)},$$

where $n$ is each agent, $k$ is each action of an agent, $\pi(s,a_k)$ is the probability of this action, $R(s,a_k)$ is the reward of this action, $m$ is the action with theoretically maximum reward. Due to the fixed reward $r(s,a)$, the change of the ratio is affected by the change of probability. The greater convergence ratio means that agents have the higher probability to select the action with higher reward.

Table 5 shows three experiments' parameters. Note that, we set the reward for the optimal action as 2, while others are 0. The Euclidean distance threshold is set to 0.7, calculated by the theoretical convergent value of $Q$-value regarding related settings. We assign an advising budget of 30 for both giving and asking for each agent during learning processes until convergence.

## 5.2 | Experiment results

Figure 4 shows the result of experiment 1 with the comparison between three strategies with the different setting of actions but in the same numbers. Our proposed approach is shown with the red line, and other two approaches are demonstrated with blue and black lines separately. The result shows that all three strategies are convergent to about 92%, and the red line is above the other two, that is, our proposed method converges the fastest, especially in the early learning stage from 0 to 250 epochs. The details are shown in Figure 5, which is the performance comparison between

**TABLE 5** Parameter setting

| Experiment: | | Experiment 1 | Experiment 2 | Experiment 3 |
|---|---|---|---|---|
| **Parameters** | | | | |
| Users | $n_{\text{users}}$ | 10 | | |
| Servers | $n_{\text{servers}}$ | 12 | | |
| Tasks | $n_{\text{tasks}}$ | 10 | | |
| Privacy budget | $\varepsilon$ | 1 | 1 | (0,2) |
| Privacy sensitivity | $S$ | 1 | | |
| Reward | $r$ | 2, 0 | | |
| Euclidean distance threshold | $d$ | 0.7 | | |
| Budget for giving advice | $b_{\text{give}}$ | 30 | | |
| Budget for asking advice | $b_{\text{ask}}$ | 30 | | |
| Epoch | | 700 | | |

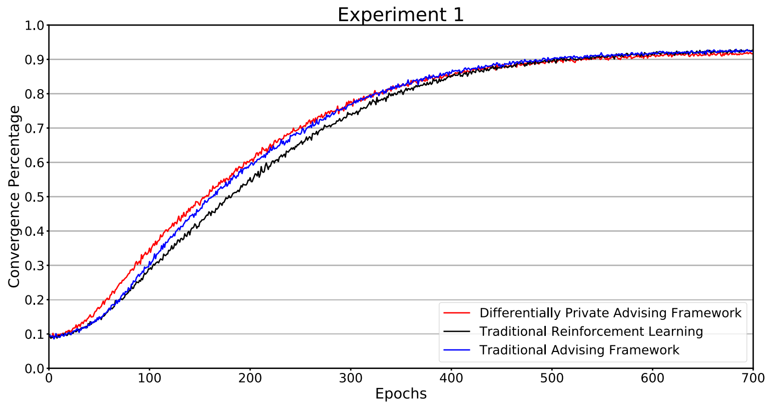

**FIGURE 4** Experiment 1: Comparison between three strategies

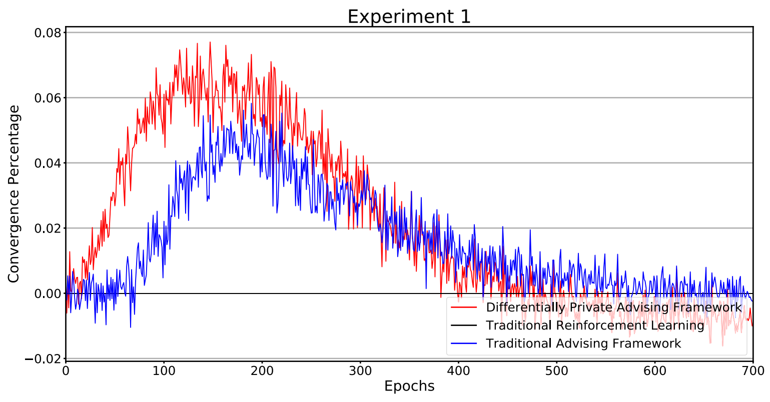

**FIGURE 5** Experiment 1: Performance comparison between three strategies (traditional reinforcement learning as base)

three strategies with the traditional reinforcement learning as a base to checked how much we improve the learning process. Our proposed method has a higher convergence percentage than the other two, and the maximum improvement is around 7% at 100 to 150 epochs, which means our approach can accelerate agents' learning during the whole learning process, especially in the early stage.

Experiment 2 demonstrates our second contribution that our approach allows more advising processes to occur in a reinforcement learning system. Figure 6 shows the result of experiment 2 with the comparison among three mentioned strategies in an identical environment, which differs from experiment 1 in that we only guarantee the total knowledge is the same for three strategies but with different actions in experiment 1. Similar to Figure 5, Figure 7 shows how much we improve the learning speed in this reinforcement learning setting. The performance of the traditional advising framework shown by the blue line is very close to the traditional reinforcement learning strategy, that is, it can only achieve a limited acceleration of learning due to the fact that advising can only occur among few agents with the same actions in the setting. However, our approach shown by the red line, accelerates the learning speed much more in the same setting than the other two strategies.

Experiment 3 explores the performance of differential privacy with different value of epsilon. The experimental results shown in Figures 8 and 9 indicate that the performance is increasingly higher with the greater value of epsilon. When $\epsilon = 0.1$, it even has a negative impact on the performance and performs worse than the traditional reinforcement learning.

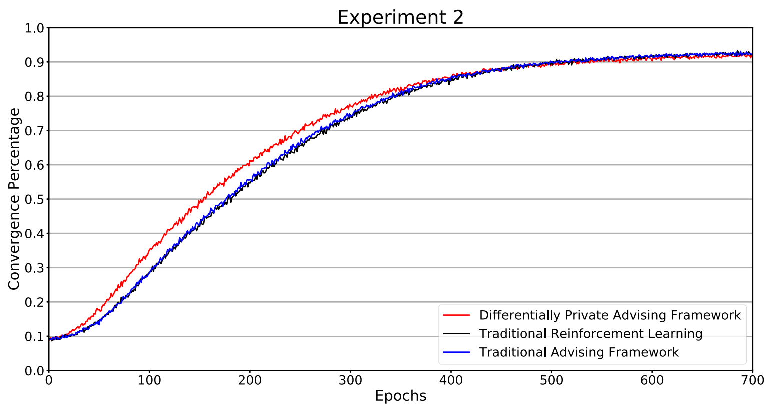

**FIGURE 6** Experiment 2: Comparison between three strategies with the same setting

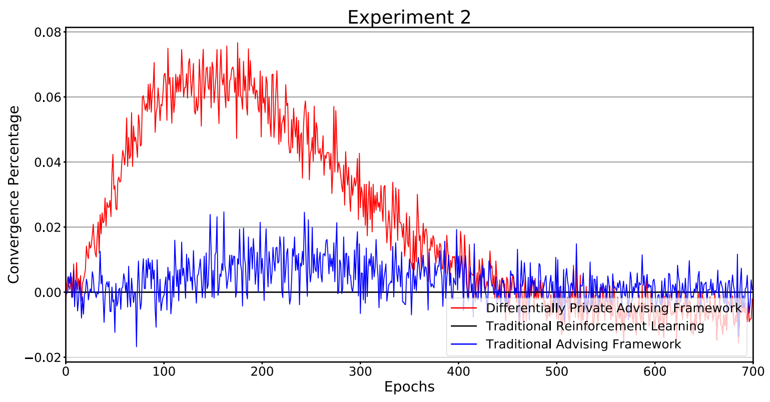

**FIGURE 7** Experiment 2: Performance comparison between three strategies (traditional reinforcement learning as base)

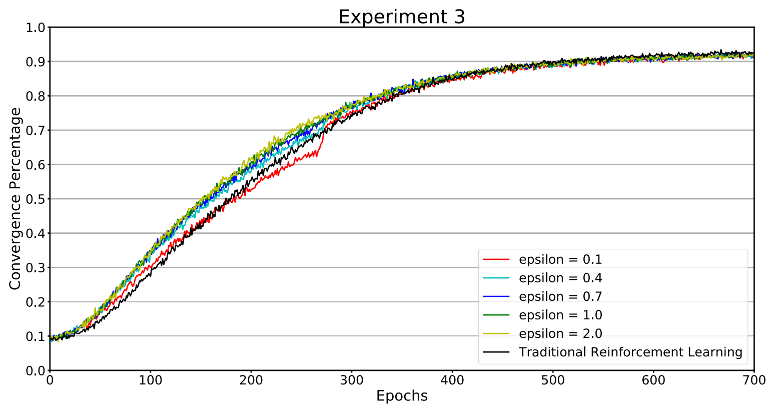

**FIGURE 8** Experiment 3: Comparison between different epsilon value with the same setting

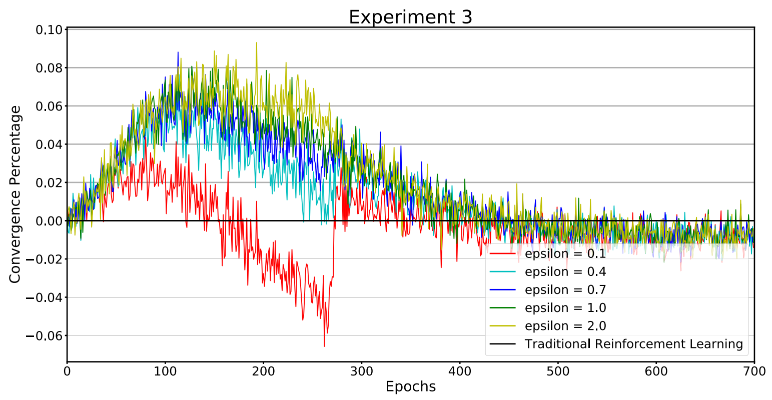

**FIGURE 9** Experiment 3: Performance comparison between different epsilon value (traditional reinforcement learning as base)

## 5.3 | Result discussion

In this section, we will discuss and analyze our experimental results. Our purpose in the first experiment is to demonstrate our first contribution that agents can give advice to or ask advice from others even when they have different actions. Our original expectation for the proposed method was that it would have the same effect as the traditional advising framework. However, the experiment shows that we have a better performance in the early learning state with a higher convergent percentage. We believe that it is positively affected by differential privacy noise. Referring back to our algorithm Algorithm 3, we see that a teacher calculates Euclidean distance with the noised $Q$-value with a student's $Q$-value and transitions the noised $Q$-value to the new probability distribution to make the decision for the student. An appropriate randomized noise positively influences the calculation of Euclidean distance and facilitates more advising occurring in the early learning stage. The noise also keeps the shape of the probability distribution of the teacher, that is, the teacher still has a high probability to choose the best action, and this positively increases the result of Euclidean distance to easily satisfy the threshold to trigger advising process.

Experiment 2 corresponds to our second contribution that our method can widen the applicable field of advising in a reinforcement learning system. Our approach performs much better than the other two compared strategies because our method allows more advising happen by breaking the limitation of the traditional advising framework. As we mentioned earlier, in the environment of this experiment, only a subset of agents have the same actions, and others have one different action with each other. The traditional advising framework performs worse because only a small number of agents with same actions can advise each other, and the effectiveness of advising is not obvious due to limited times that advising is able to occur and the effect from learning rate. However, our proposed method allows more potential advising to occur among agents with the same actions or neighboring agents with one different action. Agents have more opportunities to ask advice from more experienced neighboring agents. The learning process is positively affected by more advice and better advice in our framework. Therefore, our proposed method can widen the applicable field of advising by allowing more potential advising.

Experiment 3 demonstrates the performance of differential privacy in our proposed framework. We set five different values of privacy budget $\varepsilon$ ranging from 0.1 to 2, to explore how the differential privacy noise affects the performance. Referring back to Equation (4) we can see that the scale of the randomized noise is determined by sensitivity $S$ and privacy budget $\varepsilon$, and we have mentioned that in our case $S$ is fixed as 1 because the maximum value of a percentage is 1. So $\varepsilon$ is the only factor affecting the scale of noise with an inversely proportional relationship. When $\varepsilon = 0.1$, our proposed method even performs worse than traditional reinforcement learning because the noise is so large that it negatively impacts on the teacher's selection and student's action. It is obvious that the red line has a sharp increase at epoch 300 because most agents have run out their asking and giving budgets and start independently learning without the negative impact from the randomized noise. Therefore, in Figures 8 and 9, it is clear that our algorithm perform better when the value of $\varepsilon$ is greater within a reasonable range.

## 6 | RELATED WORK

The early advising approaches mostly focused on humans as teachers to provide advice. Clouse and Utgoff[21] proposed that humans can offer advice at any time as an expert monitoring a single student learning a multiple-step decision task via a reinforcement learning. Maclin and Shavlik[22] proposed that the teacher occasionally gives the suggestions to the student while watching the student's learning process. In 2005, Torrey et al[23] proposed a method that requires a human-provided and hand-coded mapping to link the two tasks, to transfer knowledge from one task to another as advice. Clouse[24] proposed to train an RL agent using autonomous teacher that is assumed to perform at a moderate level of expertise for the task. However, this approach may restrict the performance of advising due to receiving too much advice. *Teacher-student* framework is first proposed by Torrey and Taylor[25] and further improved by Taylor et al,[26] which introduces a numeric communication budget to limit the number of times of advising during the learning steps, which becomes an essential part of the advising model to imitate humans' availability and attention capability in real world.[27] Zimmer et al[28] introduced an approach that an agent should learn when to give advice with building a sequential decision-making problem. Even though internal representations of the student are not supposed to be known, the teacher must observe the student reward to solve the problem. All the aforementioned works rely on a single teacher with a fixed policy. Nunes and Oliveira[29] proposed that multiple agents can broadcast their average reward at the end of each learning epoch while learning in the same system, and agents whose average reward is lower than the best one can ask for device from the best agent. Zhan et al[30] considered the possibility of receiving suboptimal advice and combined multiple bits of advice by a majority vote to make this method more robust to against bad advising.

In above approaches, either teacher or student alone can trigger a process of advising in RL stage. Amir et al[31] proposed a jointly initiated framework that both students and teachers need to agree to receive and provide advice simultaneously. Da Silva et al[8] proposed an advising framework among simultaneously learning agents based on *teacher-student* framework, in which agents start learning together without an experienced agent. Their method is also based on jointly initiated teacher-student relations, which are established on demand when a student is not confident enough to make choice alone, and a teacher has enough confidence to provide advice at the same time, which is highly related to our proposed method. Ye et al[9] proposed to use differential privacy to protect benign agents against malicious agents in the advising stage. Zhu et al[32] proposed a partaker-sharer advising framework (PSAF) for cooperative agents under limited communication. The framework allows Q-value transfer from an agent who has visited a state more times to an agent who has visited a state fewer times.

## 7 | CONCLUSION

In this article, we propose a novel differentially private advising framework for multiagent reinforcement learning applying into a real-world cloud server environment. Our proposed method allows agents with at most one different action to advise each other to accelerate learning speed toward convergence in a simultaneously learning setting. We applied differential privacy mechanism to add a randomized noise on the teacher agent's $Q$-value and calculated the Euclidean distance between teacher's noised $Q$-value and student's $Q$-value to decide the timing of advising. If the Euclidean distance is greater than a threshold, the teacher agent selects an action for students based on the new probability distribution transitioned from the new noised $Q$-value.

There are three contributions of our article: (i) we break the limitation of conventional advising framework which only allows advising between agents with the same available actions, (ii) we widen the applicable field of conventional advising framework and thus allow more potential advising in some reinforcement learning systems, and (iii) we design a new approach to decide when advising occurs by calculating the Euclidean distance of teacher and student's $Q$-value.

Our three experimental results demonstrate our contributions and also explore how the differential privacy budget $\varepsilon$ affects the differential privacy in the proposed method. The experimental results indicate that our framework can facilitate potential advising between two agents with one different action and can perform better in the early learning state than the traditional advising framework in the same setting. Our approach may direct some further research in server selection, resource allocation and task allocation problems.

**ACKNOWLEDGMENT**
This work was supported by the Australia Research Council Grant under Grant LP170100123.

**ORCID**
*Sheng Shen* https://orcid.org/0000-0003-4734-1008

## REFERENCES

1. Wang L, Tao J, Kunze M, Castellanos AC, Kramer D, Karl W. Scientific cloud computing: early definition and experience. Paper presented at: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications; 2008:825-830; IEEE.
2. Mell P, Grance T. *The NIST Definition of Cloud Computing*. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8930: National Institute of Science and Technology, Special Publication, 800; 2011.
3. Ye D, Zhang M. A self-adaptive strategy for evolution of cooperation in distributed networks. *IEEE Trans Comput*. 2014;64(4):899-911.
4. Yang M, Zhu T, Liu B, Xiang Y, Zhou W. Machine learning differential privacy with multifunctional aggregation in a fog computing architecture. *IEEE Access*. 2018;6:17119-17129.
5. Littman ML. Reinforcement learning improves behaviour from evaluative feedback. *Nature*. 2015;521(7553):445-451.
6. Ye D, Zhang M, Sutanto D. Cloning, resource exchange, and relation adaptation: an integrative self-organisation mechanism in a distributed agent network. *IEEE Trans Parall Distrib Syst*. 2013;25(4):887-897.
7. Liu WX, Cai J, Wang Y, Chen QC, Tang D. Mix-flow scheduling using deep reinforcement learning for software-defined data-center networks. *Internet Technol Lett*. 2019;2(3):e99. https://doi.org/10.1002/itl2.99.
8. Da Silva FL, Glatt R, Costa AHR. Simultaneously learning and advising in multiagent reinforcement learning. Paper presented at: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems; 2017:1100-1108.
9. Ye D, Zhu T, Zhou W, Philip SY. Differentially Private Malicious Agent Avoidance in Multiagent Advising Learning. *IEEE Transactions on Cybernetics*. 2019;1-14. http://dx.doi.org/10.1109/tcyb.2019.2906574.
10. Ye D, He Q, Wang Y, Yang Y. An Agent-Based Integrated Self-Evolving Service Composition Approach in Networked Environments. *IEEE Transactions on Services Computing*. 2019;12(6):880-895. http://dx.doi.org/10.1109/tsc.2016.2631598.
11. Shen S, Zhu T, Ye D, Yang M, Liao T, Zhou W. Simultaneously advising via differential privacy in cloud servers environment. Paper presented at: Proceedings of the International Conference on Algorithms and Architectures for Parallel Processing; 2019:550-563; Springer.
12. Dwork C, McSherry F, Nissim K, Smith A. Calibrating noise to sensitivity in private data analysis. Paper presented at: Proceedings of the Theory of Cryptography Conference; 2006:265-284; Springer.
13. Dwork C. A firm foundation for private data analysis. *Commun ACM*. 2011;54(1):86-95.
14. Zhu T, Li G, Zhou W, Philip SY. *Differential Privacy and Applications*. New York, NY: Springer; 2017.
15. Zhu T, Xiong P, Li G, Zhou W. Correlated differential privacy: hiding information in non-IID data set. *IEEE Trans Inf Forens Sec*. 2014;10(2):229-242.
16. Zhang T, Zhu T, Xiong P, Huo H, Tari Z, Zhou W. Correlated Differential Privacy: Feature Selection in Machine Learning. *IEEE Transactions on Industrial Informatics*. 2020;16(3):2115-2124. http://dx.doi.org/10.1109/tii.2019.2936825.
17. Zhu T, Philip SY. Applying differential privacy mechanism in artificial intelligence. Paper presented at: Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS); 2019:1601-1609; IEEE.
18. Watkins CJ, Dayan P. Q-learning. *Mach Learn*. 1992;8(3-4):279-292.
19. Tsitsiklis JN, Van Roy B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*. 1997;42(5):674-690. http://dx.doi.org/10.1109/9.580874.
20. Melo FS. Convergence of Q-learning: a simple proof. *Institute of Systems and Robotics Technical Report*. Lisboa, PORTUGAL: Instituto Superior Técnico; 2001:1-4.
21. Clouse JA, Utgoff PE. A teaching method for reinforcement learning. Paper presented at: Proceedings of the Machine Learning; 1992:92-101; Elsevier.
22. Maclin R, Shavlik JW. Creating advice-taking reinforcement learners. *Mach Learn*. 1996;22(1-3):251-281.
23. Torrey L, Walker T, Shavlik J, Maclin R. Using advice to transfer knowledge acquired in one reinforcement learning task to another. Paper presented at: Proceedings of the European Conference on Machine Learning; 2005:412-424; Springer.
24. Clouse JA. Learning from an automated training agent. *Adaptation and Learning in Multiagent Systems*. Springer Verlag: Citeseer; 1996.
25. Torrey L, Taylor M. Teaching on a budget: agents advising agents in reinforcement learning. Paper presented at: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems. International Foundation for Autonomous Agents and Multiagent Systems; 2013:1053-1060.
26. Taylor ME, Carboni N, Fachantidis A, Vlahavas I, Torrey L. Reinforcement learning agents providing advice in complex video games. *Connect Sci*. 2014;26(1):45-63.
27. Miller D, Sun A, Johns M, et al. Distraction becomes engagement in automated driving. Paper presented at: Proceedings of the Human Factors and Ergonomics Society Annual Meeting; vol 59, 2015:1676-1680; SAGE Publications Sage CA, Los Angeles, CA.
28. Zimmer M, Viappiani P, Weng P. Teacher-student framework: a reinforcement learning approach; 2014.
29. Nunes L, Oliveira E. On learning by exchanging advice; 2002. arXiv preprint cs/0203010.
30. Zhan Y, Ammar HB Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer; 2016. arXiv preprint arXiv:1604.03986.
31. Amir O, Kamar E, Kolobov A, Grosz B. Interactive teaching strategies for agent training. Paper presented at: Proceedings of the IJCAI; 2016.
32. Zhu C, Leung Hf, Hu S, Cai Y. A Q-values sharing framework for multiple independent Q-learners. Paper presented at: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems; 2019:2324-2326.