A Concurrent Federated Reinforcement Learning for IoT Resources Allocation with Local Differential Privacy

Wei Zhou, Tianqing Zhu*, Dayong Ye, Wei Ren, Kim-Kwang Raymond Choo

***Abstract*—Resource allocation in an edge-based Internet of Things (IoT) systems can be a challenging task, especially when the system contains many devices. Hence, in recent years, scholars have devoted some attention to designing different resource allocation strategies. Among these strategies, reinforcement learning is considered to be one of the best methods for maximizing the efficiency of resource allocation schemes. In a typical reinforcement learning scheme, the edge hosts would be required to upload their local parameters to a central server. However, this process has privacy implications given some of the data processed by the edge hosts is likely to be highly sensitive. To tackle this privacy issue, we developed a concurrent joint reinforcement learning method based on local differential privacy. Our approach allows the edge host to add noise during local training to preserve privacy, and to make joint decisions with the central server to devise an optimal resource allocation strategy. Experiments show that this approach yields high performance while preserving the privacy of the edge hosts.**

***Index Terms*—resource allocation; local differential privacy; deep reinforcement learning, Internet-of-Things**

I. Introduction

The need to maximize resource use in contemporary IoT systems has become increasingly pronounced in our data-centric society. However, creating an efficient resource allocation scheme can be a complex task. Resource optimization strategies are not only dynamic, they also need to consider a diverse range of system settings, such as different operating systems, various operational requirements, including privacy regulations, and performance mandates, such as contractual service level agreements.

One potential approach to maximizing efficiency is to use reinforcement learning [1] to learn an optimal allocation strategy. However, while reinforcement learning may be good at crafting optimal strategies, the methods involved may not prevent privacy violations. In fact, existing reinforcement learning schemes fall into two categories: those that do not consider privacy issues, and those that do. Most resource allocation methods based on deep learning today fall into the first category [2]. These generally yield better performance, but they do not work well in settings where privacy protection is mandated [3], such as in Europe with the General Data Protection Regulation [4]. Further, such systems are vulnerable to a malicious server seeking to compromise the system or the data as is the case with model inversion or data reconstruction attacks [5], [6].

Of those schemes that do consider privacy, most follow a federated learning framework where privacy protection is supported for the edge hosts. Some even support privacy protection for the server [7]. Federated learning is a distributed machine learning framework that builds an aggregated model from gradient parameters uploaded by the clients. As such, the clients are not required to upload raw data [8]. Federated reinforcement learning combines both reinforcement learning and federated learning so as to support optimal decision-making while preserving data privacy [9]. However, federated reinforcement learning alone is not sufficiently secure [10], [11]. For example, when a client uploads its model gradient to the server, the server can glean private data from this process. In addition, Zhu et al. [11] have shown that an attacker can reconstruct a model's input data using only the model's gradients.

In terms of dealing with malicious servers, local differential privacy seeks to address issues with untrusted third-party servers. In these schemes, the user data is disturbed locally prior to being uploaded to the third-party server, which preserves the privacy of the data.

Focusing on an edge-based IoT setting, we explored the potential of using local differential privacy [12] to provide a theoretical guarantee for the privacy protection of edge hosts. Similar to previous work, we relied on a federated learning structure to organize the resources. In this setting, local differential privacy can potentially prevent a malicious server from performing a model inversion attack on data uploaded by the edge hosts. However, this approach may have a significant impact on system performance. To overcome this problem, reinforcement learning is used to support a decision-making process that maximizes part of the cumulative reward through interactions with the edge agents. This process allows the agent to learn and improve its actions in a trial-and-error manner, ultimately resulting in an optimal decision.

Hence, a concurrent federated reinforcement learning scheme is built by adding noise to the local training gradients on each edge host. We coin the approach 'Concurrent Federated Reinforcement Learning with Local Differential Privacy' (LDP-CFRL). Within the approach, an edge host will only share noisy outputs with the server, and an untrusted

*Tianqing Zhu is the corresponding. Email: tianqing.zhu@IEEE.org. Wei Zhou and Wei Ren are with School of Computer Science, China University of Geosciences, Wuhan, China, 430074. Dayong Ye and Tianqing Zhu are with the Centre for Cyber Security and Privacy and the School of Computer Science, University of Technology, Sydney, Australia, 2007. Kim-Kwang Raymond is with Department of Information Systems and Cyber Security, University of Texas at San Antonio, US.

server will not be able to determine the private data of the edge hosts through a model inversion or data reconstruction attack. In comparison to typical deep reinforcement learning approaches, the concurrency within our approach ensures good performance.

A summary of our contributions to the literature follows.

- This paper presents a novel resource allocation method for IoT computing that includes a privacy guarantee.
- We introduce the notion of concurrency to federated reinforcement learning, where the agents and the server make decisions jointly without the need for either party to share their models – only the outputs and rewards are shared. In this way, a similar algorithm could be applied to other similar collaborative scenarios.

The remainder of this paper is organized as follows. Section II reviews the extant literature and related approaches. Section III provides background information relevant to federated, reinforcement learning, and differential privacy. Section IV presents the design of our proposed LDP-CFRL method. Section V analyses the privacy, usability and convergence of the approach, and Section VI describes the experiment setup and provides the results of our comparative tests. The paper concludes in Section VII.

II. Related Work

Reinforcement learning has been widely used in edge computing resource allocation problems [13]–[15]. Existing resource allocation methods fall into two categories based on the degree of privacy protection offered. The two categories are deep reinforcement learning methods and federated reinforcement learning methods. Stand-out studies relating to these two types of methods are discussed below.

A. Resource allocation based deep reinforcement learning

Algorithms based on deep reinforcement learning usually have good utility when dealing with resource allocation problems in edge computing. However, these methods do not protect user privacy.

Wang et al. [16] presented a smart resource allocation method based on deep reinforcement learning for adaptively allocating computing resources in edge computing systems. Their approach, which is based on a deep Q network framework, balances resource utilization with average service times. Xiong et al. [17] considered a mobile edge computing system deployed in a cellular network, where each edge host allocates resources independently. They treat task processing as a Markov decision process and use a deep reinforcement learning algorithm with multiple replay memories to both improve the efficiency of the resource utilization and to ensure that all tasks are completed as quickly as possible.

B. Resource allocation based federated reinforcement learning

In federated reinforcement learning schemes, the edge hosts do not need to upload raw data to the server. Rather, the server generates an aggregation model from the parameters uploaded by the edge hosts. Although this method provides a certain degree of privacy for the edge hosts, model inference attacks and data reconstruction attacks can still cause privacy leaks [11], [18], [19].

Zhan et al. [20] put forward an experience-driven computation resource allocation method based on federated reinforcement learning that operates at minimum system cost. This cost is defined as the weighted sum of the learning time and the energy consumed. The main idea of this algorithm is to reduce the CPU-cycle frequency of the faster mobile devices in the training group so as to achieve a balance between energy consumption and latency.

Yu et al. [21] devised an intelligent ultra-dense edge computing framework and used it to study problems related to joint computation offloading, resource allocation, and service caching. To reduce the overhead associated with computing and resource allocation, they designed a two-timescale deep reinforcement learning approach consisting of two timescales operating at different speeds. Additionally, a federated learning architecture was used to provide privacy protection to the edge devices.

Zhu et al. [22] proposed a resource allocation method called concurrent federated reinforcement learning. Here, the edge hosts and the server collaborate to allocate resources. The server adjusts the resources of different edge hosts according to feedback about the resource allocation policy from the edge hosts. One of the main benefits of this method is that the edge hosts only need to share limited information with the server. Thus, the scheme provides a stronger privacy guarantee than standard federated reinforcement learning.

C. Local Differential Privacy in FL and RL

One way to defend against model inversion and data reconstruction attacks is to add artificial noise. Differential privacy combined with a federated learning framework is an excellent mechanism for providing a high level of privacy protection. In scenarios where the third party is not trusted, local differential privacy can provide higher privacy protection.

Hu et al. [23] put forward a personalized federated learning approach based on differential privacy to ensure that a malicious server could not access private user information from the received messages. Further, Gaussian noise is added to the gradient in the neural network layer when users upload their local updates. Thus, as long as no auxiliary information is provided, Hu et al.'s model can effectively avoid information leakage.

Truex et al. [24] integrated condensed local differential privacy (α-CLDP) into federated learning, allowing participants to perturb their gradients when training local deep neural network instances. Their method performs well against inference attacks on complex models, and participants do receive personalized local differential privacy, but α-CLDP requires a relatively large privacy budget, which will lead to an insufficient privacy guarantee.

Ono et al. [25] applied local differential privacy to reinforcement learning. They proposed a private gradient collection (PGC) framework, where the agent uploads noisy gradients

to a central aggregator, and the central aggregator updates the global parameters. Their method can also learn robust policies while satisfying local differential privacy.

D. Discussion of related work

Deep reinforcement learning typically performs well in resource allocation for edge computing systems, but it overlooks the data security of edge hosts.

The IoT environment in which edge computing operates is actually quite fragile, as indicated by Peng et al.'s research [26], which highlights the security challenges it faces, such as inherent vulnerability, programming vulnerabilities, and attacks. As a successful example of a secure framework, Federated Learning (FL) continues to emerge, but it faces the heterogeneity of data, models, and devices in the IoT field. Lu et al.'s research proposed the PFL framework to overcome these issues [27], applying Federated Learning in the IoT field and achieving excellent results.

Therefore, federated reinforcement learning methods can protect the raw data of edge hosts to some extent [28], [29]. In some studies, the federated learning framework is directly used as a defense mechanism. For example, Yang et al. proposed an asynchronous federated learning framework that allows local model training without transmitting raw device data to the server [30]. Zhang et al. applied federated learning and deep reinforcement learning to manage data generated by IoT devices to improve efficiency and protect privacy [31]. Chen et al. proposes a federated learning algorithm (FL-DDPG) based on DDPG framework to minimize energy consumption of IoT devices by jointly optimizing task offloading and resource allocation [32], while protecting privacy and improving training performance.

These methods protect data privacy to some extent, but we will show in privacy analysis that using only the basic federated learning framework is not absolutely secure. Especially in the case of an untrusted server, these methods do not have the robustness of model inversion attacks and still carry the risk of privacy leakage. Wang et al.'s research combines local differential privacy mechanisms with federated learning [33], providing strong data privacy protection while ensuring high availability of the method. However, their research did not apply this concept to the field of resource allocation. Therefore, we introduce local differential privacy mechanisms into the resource allocation method, which can protect training data from model inversion attacks and membership inference attacks. By combining federated reinforcement learning with local differential privacy, our LDP-CFRL method should provide more horizontal privacy protection while maintaining excellent performance.

III. Preliminaries

In this section, we present the preliminaries and background on federated learning, reinforcement learning, and differential privacy.

A. Deep reinforcement learning

Reinforcement learning is a self-teaching process that is very effective at solving complex sequential decision problems. Reinforcement learning can simulate the decisions of agents as they adapt to dynamic environments. It can be viewed as a Markov decision process consisting of a 4-tuple $\langle S, A, T, R\rangle$, where S is a set of states, A is a set of actions, T is the transition model, and R is the reward function [34].

In reinforcement learning, the policy π refers to the probability distribution of actions in a state. Here, the agent chooses the next action to perform according to $\pi(a|s)$. Thus, the main goal of reinforcement learning is to find the optimal policy π^* that maximizes the rewards returned $\mathbb{E}_{\pi^*}[R]$. The reinforcement learning part of our proposed algorithm is based on the DQN algorithm. The DQN algorithm is derived from Q-Learning, which is a value-based method with an action-value function $Q(s,a)$. $Q(s,a)$ is used to represent the expected reward of the agent for taking action a in state s. The action-value function $Q(s,a)$ is defined as:

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[R(t) \mid s_t = s, a_t = a\right]. \tag{1}$$

There is a basic connection between the policy π^* and the optimal action-value function Q^*, in that if π^* is performed, Q^* can be calculated. Similarly, if Q^* can be calculated (by maximization), π^* can be found. This is formulated as follows:

$$\pi^* = \arg\max_\pi Q^*(s,a). \tag{2}$$

Given the connection between π^* and Q^*, finding one term will lead to finding the other term. Therefore, Q^* can be rewritten as a Bellman optimal equation:

$$Q^*(s,a) = \mathop{\mathbb{E}}_{s'\sim T}\left[r(s,a) + \gamma \max_{a'} Q^*\left(s',a'\right)\right]. \tag{3}$$

Where γ is a discount rate ranging from 0 to 1. The action-value function $Q(s,a)$ can be calculated by iterating through the temporal difference update, given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)]. \tag{4}$$

The reason why DQN can adapt to a variety of complex scenarios is that it replaces the Q-table in Q-Learning with a neural network $Q(s,a,\theta)$. This network is used to learn the weights θ so as to approximate the action-value function $Q(s,a)$. The loss function in this neural network is defined as:

$$\mathcal{L}(\theta) = \mathbb{E}\left[\underbrace{\left(r\left(s,a\right) + \gamma \cdot \max_{a}, Q\left(s',a',\theta\right)\right)}_{\text{Target}} - Q(s,a,\theta)^2)\right]. \tag{5}$$

where s' is the next state and a' is the next action. Minimizing $\mathcal{L}(\theta)$ means reducing the mean squared error. Updating the weights θ through a stochastic gradient descent algorithm is a common implementation.

B. Federated learning

Google first proposed federated learning [35] as a way to train language models in a client-server architecture. Federated learning is a form of distributed learning that allows multiple clients to work together with a central server. The clients do not need to share their private data with the server. Rather, they upload their training parameters to the server for aggregation. With this feature, federated learning provides a more secure and reliable learning service to enhance machine learning in distributed scenarios [36].

However, simply decoupling federated learning from accessing the client's private data cannot completely prevent privacy leaks. If the server is not trusted, private data can still be derived through the intermediate parameters shared by the client [11], [18], [19]. So, to safely share the parameters needed to the train models, the federated learning scheme needs to incorporate an additional privacy-preserving mechanism that can prevent privacy leaks caused by parameter sharing. Some examples of these privacy-preserving mechanisms include secure multi-party computation [37], homomorphic encryption [38] or differential privacy [39]. The first two require a huge computational overhead, while the cost of differential privacy is low. Differential privacy is acceptable if the model will ensure that the model's accuracy is not greatly reduced after the data has been perturbed by noise.

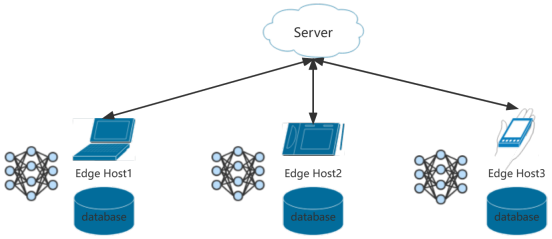


Fig. 1: A federated reinforcement learning framework.

C. Local Differential Privacy

In traditional differential privacy, a trusted server adds noise to the raw data uploaded by the device. But, in most cases, finding a trusted server is very difficult.

In local differential privacy, each device's data is first perturbed locally and then sent to the server, so the server processes the perturbed data instead of the original data. Therefore, even if the server has a large amount of device information, it cannot infer sensitive data by observing the output of the device. In other words, local differential privacy enables the devices to process private data themselves. Intuitively, local differential privacy ensures that the server cannot infer the absolute output of the device by changing the input of the device.

Formally, local differential privacy can be expressed as:

$$Pr[\mathcal{M}(D) \in Y] \leq \exp(\epsilon) \cdot Pr\left[\mathcal{M}\left(D'\right) \in Y\right] + \delta \quad (6)$$

Note that the classical definition of pure differential privacy does not include the additional parameter δ, Dwork et al. [40] was the first to propose approximate differential privacy, which involves the parameter δ, where δ is the probability that pure differential privacy can be broken.

IV. Concurrent Federated Reinforcement Learning Methodology

This section begins with an overview of our system. The learning process and the elements of the edge host and server are then introduced, and the section concludes with an analysis of the algorithm. Table I summarizes the symbols used throughout the paper.

A. Overview of the System

As shown in Fig. 2, edge computing systems consist of three types of entities: IoT devices, edge hosts, and a server.

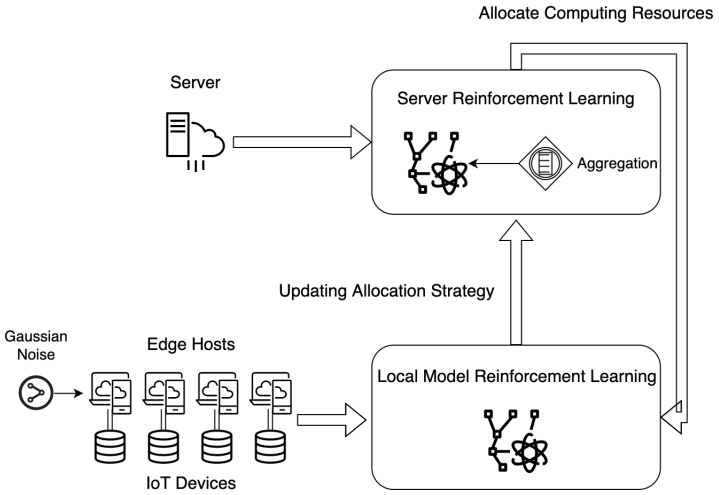


Fig. 2: A framework based on LDP-CFRL. Edge hosts add Gaussian noise to the local training process and shares the perturbed allocation policies with the server. The server aggregates these policies and adjusts the computing resources of the edge hosts.

- IoT devices typically have limited computing power and storage capacity, so they regularly upload tasks to edge hosts;
- The edge hosts store the received tasks in a task queue and process them according to a first-in-first-out policy (FIFO). The edge hosts and the server jointly formulate resource allocation strategies based on the tasks to be processed. However, the edge hosts do not fully trust the server, and so noise is added to the learning process to protect the data privacy of the IoT devices. When the edge hosts receive a task that it cannot process, it uploads the task to the server for processing.
- The server is an incompletely trusted third-party entity. It adjusts the number of resources allocated to each edge host according to the resource allocation strategy. The system's goal is to complete the tasks in the queue as quickly and efficiently as possible.

To accomplish this global objective, resource allocation is formulated as a Markov decision process that comprises three

main elements: a state, an action, and a reward. On the edge host side, the environment is itself. The state depends on how the task is processed. The action refers to the resources assigned to the task. And the reward depends on the size and processing time of the task. On the server side, the environment is itself and all edge hosts. The state depends on the resource allocation strategy submitted by the edge hosts. The action refers to how the edge hosts' resources are adjusted. And the reward depends on feedback from the edge hosts. The components of the system are described in the following subsections.

TABLE I: The meaning of each notation

Notation	Description
S	state space
A	action space
T	transition function
R	reward function
D	replay memory space
$\tilde{D}$	a batch of samples from D
n_e	the number of edge hosts
B	gradient norm bound
$\mathbf{C}$	state matrix for edge host
$\boldsymbol{\tau}$	queue of tasks waiting to be processed
m	computing resources owned by edge hosts
n	length of sliding window owned by edge host
c^{τ_i}	computing resources assigned to task τ_i
t	time step
u^{τ_i}	utility of handling tasks τ_i
r_e	reward of an edge host
r_s	rewards obtained by the server
I	maximal iteration
$\mathcal{L}$	loss function
ϵ	privacy budget
$\mathcal{M}$	randomized mechanism for DP
ε	the parameter of ε-greedy method in reinforcement learning
α	learning rate
γ	discount rate
λ	the mean rate of a Poisson distribution
z	Gaussian noise

B. The Edge Hosts' Learning Process

The learning process of the edge host is a process of building an allocation strategy based on a deep reinforcement learning algorithm. In this process, the edge hosts schedule tasks in the queue. This includes allocating resources for local processing or sending tasks to the server for processing. Since the server is not completely trusted, the edge hosts need to add enough noise to the gradient in the training phase to satisfy local differential privacy. One of the most straightforward ways to craft a noisy gradient is simply to add Gaussian noise to the gradient. However, sensitivity issues with stochastic gradients need to be considered. So first, gradient clipping is applied and then noise is added:

$$\overline{\mathbf{g}_t}(x_i) = \frac{\mathbf{g}_t(x_i)}{\max\left\{1, \frac{\|\mathbf{g}_t(x_i)\|_2}{B}\right\}} \tag{7}$$

where $\mathbf{g}_t(x_i)$ is the gradient vector for the i-th sample of $\tilde{D}$ and B is the clipping threshold. Gradients are clipped with the ℓ_2 norm, while ensuring that, for any two clipped gradients, $\|\overline{\mathbf{g}_1} - \overline{\mathbf{g}_2}\|_2 \leq B$. Then the Gaussian noise z is generated such that:

$$\mathbb{P}(z_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{z^2}{2\sigma^2}\right) \tag{8}$$

With the Gaussian noise z added, the edge host crafts the noisy gradient as follows:

$$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{\tilde{D}}\left(\sum_i \overline{\mathbf{g}}_t(x_i) + z\right) \tag{9}$$

The added noise z satisfies the (ϵ, δ)-differential privacy once the following equation is established:

$$\sigma \leq \frac{\sqrt{2\ln(1.25/\delta)}\Delta f}{\epsilon} \tag{10}$$

Last, the generated perturbed strategy is handed over to the server. The edge hosts receive a reward for completing the task and the DQN model is updated.

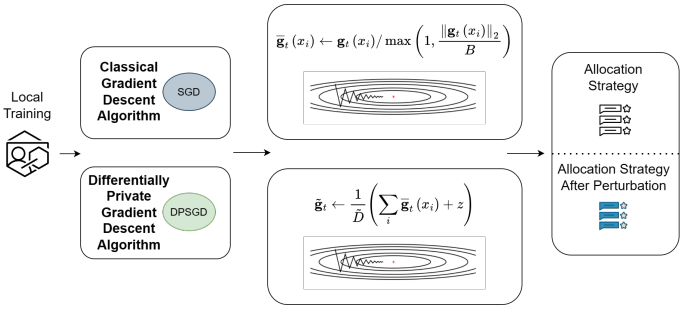


Fig. 3: The process of adding Gaussian noise to an edge host in formulating a local allocation policy.

C. The Server's Learning Process

The server's learning process is also based on a deep reinforcement learning algorithm. Its aim is to adjust the resources of different edge hosts. In this process, the server learns according to the resource allocation strategies uploaded from the edge hosts and, in return, it sends a resource adjustment strategy back to the edge hosts. The server determines the rationality of its resource allocation strategy by receiving feedback from the edge hosts. When a reasonable resource allocation strategy is made, the edge hosts provide higher reward values as feedback. And as a last step, it updates the DQN model.

D. State

The state s of the edge host shows the state information of the Markov decision process, which is defined as:

$$S = \{s | s = \langle \mathbf{C}, \boldsymbol{\tau} \rangle\}, \tag{11}$$

where $\mathbf{C}$ represents the processing state of a task in the edge host, and $\boldsymbol{\tau}$ represents the queue of tasks to be processed. The size of the task in the queue will follow a discrete uniform distribution with a mean of $\bar{X}$. Fig. 4 shows the details of the state s.

The matrix $\mathbf{C}$ consists of computing cells, with the number of rows m representing the computing resources of the host and the number of columns n representing the length of the sliding window of the host. Each cell in the state represents a computing resource. Computing resources have three different states – allocated, available, and unavailable, which

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3312118

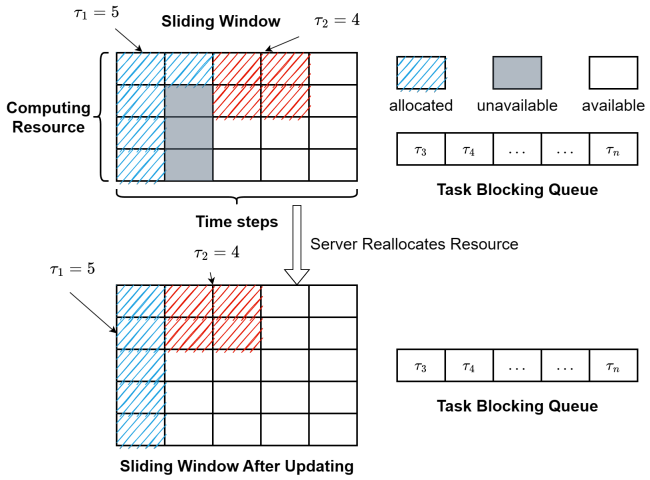


Fig. 4: The state of edge host(s)

are recorded in the matrix as 0, 1 and -1, respectively. For example, Fig. 4 shows that there are 4 resources allocated to processing task τ_1 at time steps $t1$ and $t2$, and 2 resources allocated to processing task τ_2 at time steps $t3$ and $t4$. The resource request in the first column is processed after each time step, and the sliding window moves one column to the left.

Fig. 4 also shows the state space of the edge hosts after the server adjusts the edge host's computing resources. An appropriate adjustment strategy can improve resource utilization and speed up task completion. For example, in Fig. 4, the server adjusts the computing resources of the edge host from 4 to 5. With this adjustment, the edge host will be able to complete task τ_1 in 1 time unit, improving the overall utilization rate.

The queue $\boldsymbol{\tau}$ represents the tasks queue uploaded by the IoT devices that are waiting to be processed by the edge host. The value in the queue represents the size of each task. In other words, it represents how many computing resources each task requires.

In our model, the server has a different state space than the edge hosts. The state S' represents the state of the Markov decision process on the server, which is defined as follows:

$$S' = \{s' | s' = \langle \mathbf{C}', \mathbf{\Omega} \rangle\}, \tag{12}$$

The server's state space $\mathbf{C}'$ is composed of the allocation policies of the edge hosts. The number of rows of the state is determined by the number of hosts connected to the server, and the number of columns indicates the number of tasks to be processed by the edge hosts. The queue $\mathbf{\Omega}$ refers to the number of resources that the server has allocated to the edge host at this time. For example, in Fig. 4, the edge host has 5 resources, so it can allocate no more than 5 resources to the tasks that need to be processed.

E. Action

Because the state spaces of edge hosts and the server are different, their action spaces are also different. In the edge hosts, each action allocates one or more resources to a task. Hence, the action space consists of the number of resources allocated to tasks and the timing of those allocations. Thus, the action space A can be defined as:

$$A_e = \{a | a \in \{1, ..., m\}\}, \tag{13}$$

where a represents the number of resources allocated by the edge host to the current task. The processing time of the task in the edge host is determined by this action a. For example, as shown in Fig. 5, suppose there is a task with a size of 6, which means the task requires 6 computing resources to be completed. If action $a = 3$ is performed, the task will be completed in 2 time steps. The tasks are placed in matrix $\mathbf{C}$ for processing as close to the current time slice as possible. However, if the edge host thinks the current task cannot be processed locally, it will upload the task to the server. Action $a = 0$ indicates that the task will be uploaded.

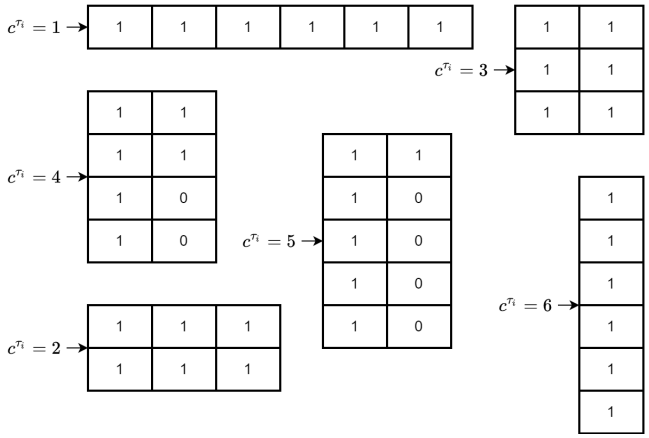


Fig. 5: Possible processing of a task requiring 6 resources

In terms of the server, the action space A' contains the serial number of the edge host and the actions associated with adjusting the number of resources allocated to the edge host. The action space A' can be defined as:

$$A' = \{a' | a' = \langle d, n_{e_i} \rangle\}, \tag{14}$$

where $d \in \{-1, 0, 1\}$ and $1 < n_{e_i} < n_e$. Each action a' is a two-tuple, which denotes adjusting d resources on the $n_e - th$ edge host. For example, assume there is a server connected to three edge hosts. Each edge host has 10 initial resources, and the server can perform an action $a' = \langle -1, 2 \rangle$ to reduce the allocation to the second edge host by one resource.

F. Reward

The design of the reward signal is consistent with the ultimate goal of the system, which is to complete the tasks uploaded by IoT devices as quickly as possible. In this system, delegating tasks to server for processing will result in lower or even negative rewards, because the server assumed in this system is not completely trustworthy. Therefore, delegating tasks to servers for processing is a very dangerous behavior, and the edge host will try to avoid uploading tasks. As a result, there are differences in the reward signals for the edge host and the server. The reward for the edge host is based on the number

of time steps and the size of the task used to complete the task. Completing the task in less time or completing a larger task will result in higher rewards. Formally, the reward r is related to the action $a \in \{1, ..., m\}$ and the state $s = \langle \mathbf{C}, \boldsymbol{\tau} \rangle$. The reward function r_e is defined as follows:

$$r_e = \begin{cases} \mu u^{\tau_i} - \nu \frac{\tau_i}{c^{\tau_i}} & \text{when edge host processing } \tau_i; \\ \mu(u^{\tau_i} - u^{\tau_{max}}) - 1 & \text{when server processing } \tau_i \end{cases} \tag{15}$$

where u^{τ_i} represents the system utility after completing the task, which is related to the size of the task. Completing a larger and more complex task will yield a higher u^{τ_i}. $\frac{\tau_i}{c^{\tau_i}}$ denotes the time required to process the tasks τ_i. μ and ν are weighting coefficients between the processing time spent and the size of the task. According to this definition, if a task is processed locally by the edge host, the edge host's reward will be the difference between the size of the task and the number of time steps it takes to process. As each cell in the state matrix $\mathbf{C}$ of the edge host represents a computing resource, if the size of a task exceeds the number of cells, the edge host will consider the task as one that cannot be completed locally in the short term and upload it to the server, and the value of the reward depends on the difference between the task size and the maximum size of the task. The maximum size of the task is twice $\bar{X}$. By adjusting the weight coefficients μ and ν, the reward signal is set to a negative number, and the edge host will continuously optimize the resource allocation strategy in order to avoid an excessive penalty.

The reward function for the server differs from that of the edge hosts. It mainly considers two scenarios: receiving tasks from edge hosts and not receiving tasks from edge hosts. For example, given an action, the server's reward can be defined as follows:

$$r_s = \begin{cases} -\sum_{i=1}^{n_e} (d_i + c^{\tau_i}) \\ \text{when no tasks are received;} \\ \mu u^{\tau_i} - v \left[\frac{\tau_i}{c^{\tau_i}}\right] - \sum_{i=1}^{n_e} (d_i + c^{\tau_i}) \\ \text{when received task from } n_{e_i} \end{cases} \tag{16}$$

The server allocates resources to edge hosts based on $\langle d_i, n_{e_i} \rangle$. If the server does not receive any tasks, it will be penalized for wasting resources. When a task is received, the server will be rewarded based on two factors: the processing time and difficulty of the task. Constructing a reward function r_s for the server in this way helps it allocate resources while retaining reasonable resource usage.

G. Algorithm

The algorithm for the edge hosts is given in Algorithm 1. As an edge host interacts with their environment, the first thing observed is the state s^t. Then an action to execute is selected based on the ε-greedy principle, i.e., a random value φ is generated and compared to a threshold in the ε-greedy principle. If φ is greater than the threshold, the edge host selects a greedy strategy. Otherwise, it selects a random action, and the threshold will gradually decrease as the number of interactions increases (Lines 7-14). Once the selected action has been performed, the edge host receives its reward for completing the task, and the state s^t transforms to s^{t+1} (Lines 15-16). The transition sample $\langle s^t, a^\tau, r_e, s^{t+1} \rangle$ is stored in D. The edge host takes a batch of samples from memory D and calculates the loss function L. DQN uses a differentially private stochastic gradient descent algorithm to update the gradients. Gaussian noise is added to the gradient using Algorithm 2 (Lines 17-20). Finally, the edge host uploads the output of the local model to the server.

Algorithm 1: The noising edge host's Algorithm

1 **Input**: resource allocated by the server;
2 Initialize a random weight θ for DQN;
3 Initialize the state S by local computing resource;
4 Initialize the replay memory $D = \emptyset$;
5 Initialize action-value function Q with θ;
6 **for** $t = 1$ *to maximal iteration number* I **do**
7 Observe the current state s^t;
8 **if** *task queue* = $\emptyset$ **then**
9 break;
10 Generate random number φ;
11 **if** $\varphi > \varepsilon$ **then**
12 select $a^\tau = argMax_{a^\tau} Q(s^t, a^\tau; \theta)$;
13 **else**
14 Randomly select an action a^τ from the action space A;
15 Perform action a^τ and transform state s^t to s^{t+1};
16 Receive the reward r_e;
17 Store transition $\langle s^t, a^\tau, r_e, s^{t+1} \rangle$ into D;
18 Randomly selected transition samples from replay memory$D$;
19 Using Equation 5 to calculate the loss function $\mathcal{L}(\theta)$ and updating θ' with differentially private stochastic gradient descent based on Algorithm 2;
20 Send the action-value function Q vector $\boldsymbol{Q}(s^t)$ to the server;

Algorithm 2: Gaussian mechanism for gradient updating

1 **Input**: gradient vector $\mathbf{g}$, privacy budget ϵ, gradient norm bound B, noise scale σ;
2 **Clipping Gradient**
3 $\overline{\mathbf{g}_t}(x_i) \leftarrow$ clipping gradient $\mathbf{g}$ using Eq. 7
4 **Generating Noise**
5 $z \leftarrow$ generating Gaussian noise z using Eq. 8
6 **Output**: Noisy gradient $\tilde{\mathbf{g}}_t = \frac{1}{\tilde{D}} \left(\sum_i \overline{\mathbf{g}}_t (x_i) + z \right)$;

The server algorithm (Algorithm 3) also uses deep reinforcement learning to allocate resources to the edge hosts. In each interaction with the environment, the server first forms the state space o^t based on the Q-value vector $\boldsymbol{Q}(s^t)$ uploaded by edge hosts (Line 6). Next, the server selects the action using the same ε-greedy principle as the edge host algorithm (Lines 7-12). Executing the action a_s sends a resource adjustment command to the edge hosts, and the server obtains the reward

r_s based on the feedback from those hosts (Lines 13-14). Then, the server stores the transition sample into memory D and randomly selects a batch of samples for training the Q-network (Lines 15-18). Finally, the server distributes the allocation policies to the edge hosts.

Algorithm 3: The server's Algorithm

1 **Input**: the edge host's Q-value vector $\boldsymbol{Q}(s^t)$;
2 Initialize a random weight θ' for DQN;
3 Initialize the replay memory $D' = \emptyset$;
4 Initialize action-value function Q with θ';
5 **for** $t = 2$ *to maximal iteration number* I **do**
6 Combine all vectors $\boldsymbol{Q}(s^t)$ and assemble into a state matrix o^t;
7 Observe the current state o^t;
8 Generate random number φ;
9 **if** $\varphi > \varepsilon$ **then**
10 select $a_s = argMax_{a_s} Q(o^t, a_s; \theta')$;
11 **else**
12 randomly select an action a_s;
13 Send the resource increase or decrease signal to the edge host according to action a_s;
14 transform state to o^{t+1} and receive the reward r_s;
15 Store transition $\langle o^t, a^s, r_s, o^{t+1} \rangle$ into $D^{'}$;
16 Randomly selected transition samples from replay memory$D^{'}$;
17 Using Equation 5 to calculate the loss function $\mathcal{L}(\theta)$ and updating θ' with stochastic gradient descent;
18 **Output**: Distribute allocation policies to edge hosts;

H. Case Study

Exactly how our scheme protects privacy is best illustrated through a case study. Suppose we have an IoT+edge system designed to deal with route planning problems. The system comprises dozens of IoT devices and an edge computer. Each IoT device needs to plan its own optimal route from the current location to the target location, with each optimal route computation being considered as a task. It is assumed that the edge computer knows the starting locations for each IoT device and that it is not trusted. On the contrary, it will use a backward attack to derive either the local computational model or the route details based on the output of the IoT device. Typically, the location information will be a collection of floating point latitude and longitude values.

If the resource allocation method does not consider privacy protection, each IoT device would directly upload the gradient information of its local model. The edge computer would then iteratively transform the obtained gradient to its negative value to find the local minimum of the differentiable function. Depending on the need of the attack, the latitude and longitude information is computed using gradient inversion and then processed for location optimization, such as latitude and longitude offset correction, which will recover the route information.

However, with our scheme, noise is added when performing gradient descent, and a noise-perturbed gradient is uploaded to the server, which will not cause a leak in the local computational model data. Hence, the route information is protected even if the edge computer launches an attack.

V. Theoretical analysis

Existing federated learning-based methods cannot be completely private without a trusted third-party server, which is impractical in the real-world. However, our method includes a solution to this problem, which means we do not need a trusted third-party. The following analysis demonstrates how from three perspectives – privacy, utility, and convergence.

A. Privacy Analysis

As mentioned in I, the current resource allocation methods are prone to privacy leaks in the face of inversion attacks when the server is not trusted. However, as the following analysis shows, our proposed method is able to provide a theoretical guarantee of privacy. The analysis covers three different aspects of privacy. First, we show that the local models on the edge hosts satisfy (ϵ, δ)-LDP. Second, we show that the policies aggregated by the server do not break differential privacy. Third, we show how our approach preserves privacy based on an improved federated learning framework. Finally, we compare and analyze the differences in privacy protection between the LDP-CFRL method and the CFRL method.

Theorem 1. *The local models on the edge hosts satisfy (ϵ, δ)-LDP.*

Proof. The first step in proving that the local models on the edge hosts satisfy (ϵ, δ)-LDP is to analyze Algorithm 2, which is the mechanism that generates the noise. Each edge host is given θ, which contains a noise gradient $\tilde{\mathbf{g}}_t$ and an output o_t. With a Gaussian mechanism and gradient clipping, for any $\mathbf{g}_t$ and $\mathbf{g}'_t$, the following inequality holds for any $\mathbf{g}_t$ and $\mathbf{g}'_t$:

$$\frac{P\left(o_t \mid \mathbf{g}_t\right)}{P\left(o_t \mid \mathbf{g}'_t\right)} \leq e^{\epsilon} \tag{17}$$

Let $\mathbb{P}\left(z_i|\mathrm{g_t}\right)$ and $\mathbb{P}\left(z_i|\mathrm{g}'_\mathrm{t}\right)$ be the probability distribution of Gaussian noise, given $\mathbf{g}_t$ and $\mathbf{g}'_t$, and we have

$$\begin{aligned}
&\mathrm{Pr}_{z\sim\mathcal{N}(0,\sigma^2)}[\mathbb{P}\left(z_i|\mathrm{g_t}\right) + z \in O] = \\
&\mathrm{Pr}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\mathbb{P}\left(z_i|\mathrm{g_t}\right) + z \in O_1\right] \\
&+ \mathrm{Pr}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\mathbb{P}\left(z_i|\mathrm{g_t}\right) + z \in O_2\right] \\
&\leq \mathrm{Pr}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\mathbb{P}\left(z_i|\mathrm{g_t}\right) + z \in O_1\right] + \delta \\
&\leq e^{\varepsilon}\left(\mathrm{Pr}_{z\sim\mathcal{N}(0,\sigma^2)}\left[\mathbb{P}\left(z_i|\mathrm{g}'_\mathrm{t}\right) + z \in O_1\right]\right) + \delta,
\end{aligned}$$

This yields Gaussian-based (ϵ, δ)-LDP. □

Theorem 2. *The server aggregates and updates the parameters uploaded by the edge hosts, which does not violate the (ϵ, δ)-LDP satisfied for each edge host.*

Proof. The server only receives the perturbation assignment policies uploaded by the edge hosts, which means that the policies are processed and output through the LDP model on the edge hosts. Due to the post-processing invariance of the

differential privacy mechanism, the server's processing of the policy does not violate the (ϵ, δ)-LDP satisfied by each local agent. □

Theorem 3. *Our improved federated learning framework can defend against model inversion attacks and model extraction attacks.*

Proof. In order to overcome a model inversion attack, our method does not share the gradient parameters of the local model with the server. Rather, it shares the allocation strategy output by the local model, which effectively prevents the server from reconstructing the model according to the gradient. Taking a possible model extraction attack on the shared outputs as an example, suppose there is a linear regression model f, which is used to solve a binary classification problem, expressed as $f(x^*) = \sigma(\mathbf{w} \cdot x^* + \beta)$. Its output is $\sigma(t) = \frac{1}{1+e^{-t}}$. The weight w of the model can be cracked from the output, and, once the weight has been obtained, the model can be reconstructed. However, in our method, according to Theorem 1, the server cannot judge whether the output of the edge host is the real output and cannot compute the correct weight w based on the output. Consequently, the method can defend against model extraction attacks. □

Comparing the CFRL method with the LDP-CFRL method, it can be seen that the main difference between them is that the LDP-CFRL method adds differential privacy noise during the local training phase of edge hosts. We use w_i to represent the parameters of the i-th edge host during the local training phase, and B is the clipping threshold used for the boundary w_i. For the CFRL method, assuming that the replay memory during local training of edge hosts is D, the local training process of the i-th edge host can be defined as follows:

$$\mathbf{w}_i = \arg\min_{\mathbf{w}} F_i(\mathbf{w}, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \arg\min_{\mathbf{w}} F_i(\mathbf{w}, \mathcal{D}_j) \tag{18}$$

D_j represents the j-th batch selected from the replay memory. For the LDP-CFRL method, we use s^D to represent the sensitivity of edge hosts, and express it as follows:

$$\begin{aligned} \Delta s^{\mathcal{D}} &= \max_{\mathcal{D},\mathcal{D}'} \left\| s^{\mathcal{D}} - s^{\mathcal{D}'} \right\| \\ &= \max_{\mathcal{D},\mathcal{D}'} \| \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \arg\min_{\mathbf{w}} F_i(\mathbf{w}, \mathcal{D}_j) \\ &\quad - \frac{1}{|\mathcal{D}'|} \sum_{j=1}^{|\mathcal{D}'|} \arg\min_{\mathbf{w}} F_i\left(\mathbf{w}, \mathcal{D}'_j\right) \| = \frac{2B}{|\mathcal{D}|} \end{aligned} \tag{19}$$

Where D' is the adjacent replay memory of D, with equal size and only one sample difference between them. Based on the above results, the sensitivity in the LDP-CFRL method can be defined by $\Delta s \triangleq \max\left\{\Delta s^{\mathcal{D}}\right\}$.

B. Utility Analysis

To analysis the utility of our model, we choose RL-specific assumptions (Linear Markov Decision Process [41], [42]) to establish the utility guarantees. U is the utility symbol, which can be defined as $U = \sum_{t=0}^{I} \gamma^t R(s_t)$. The ultimate goal of the model is to maximize the U.

The LDP-CFRL method has a discrete state space S, a probability distribution $P(s' \mid s, a)$ for the transition T, a learning rate γ, and reward function R. The value function v can be calculated by $v = \mathbb{E}[U_t \mid S_t = s] = \sum_{a \in A} Q(a \mid s)\left(R_s^a + \gamma \sum_{s' \in S} P_{S_{t+1}} v(s')\right)$, and optimal value function v^* can be found by the optimal action value function, we can obtain the optimal value function v^* by solving the following Bellman equation:

$$v^* \geq \gamma P v + R \tag{20}$$

The utility loss is expressed as:

$$\mathbb{E}\left[\frac{1}{n} \left\| v' - v^* \right\|_1\right] \leq \frac{2\sqrt{2}\sigma}{\sqrt{n\pi}(1-\gamma)} \tag{21}$$

Here, v' denotes the value function learned by the algorithm, while v^* denotes the optimal value function. If and only if the model has discrete state space and the Bellman equation 20 holds the v^* can be found.

The LDP-CFRL method has discrete state space setting so the optimal value function v^* can be found. The utility of the model can be guaranteed if we prove the convergence of the model.

C. Convergence Analysis

This convergence analysis covers both the convergence of local models on the edge hosts and the convergence of the server's federated reinforcement learning algorithm.

The analysis of the models on the edge hosts is based on Algorithm 1. To the best of our knowledge, the classical Q-learning algorithm relies on the Banach fixed-point theorem to ensure convergence. However, DQN introduces a neural network with a nonlinear activation function for function approximation, and, to date, no-one has theoretically proven that the Q-Loss must converge. In fact, with the objective function $(r(s,a) + \gamma \cdot \max_a, Q(s', a', \theta) - Q(s, a, \theta)^2)$ and the policy π^*, the classical Q-learning's Bellman backup operator is $[T^*Q](s,a) = \mathbb{E}_{S \sim T}[r + \gamma \max_a Q(s,a)]$. However, DQN's update process may be $\sqcap T^* Q^\theta$, which means that Q-Loss' eventual convergence cannot be guaranteed. According to the advice given by OpenAI, after the first step of the gradient descent, Q-Loss is no longer connected to any performance improvements. The only metric that needs to be focused on is the average return reward.

The convergence analysis of the server's model is based on Algorithm 3. Similar to Algorithm 1, Algorithm 3 also uses a DQN model for training, but its input is the Q-value vectors $\boldsymbol{Q}(s^t)$ output by the edge hosts. Here, $\boldsymbol{Q}(s^t)$ is denoted as $\boldsymbol{w}_t^{(k_i)}$, and $o^t = \boldsymbol{w}_t^{(k)} \mid k \in [N]$. The server's $Q(o^t, a_s)$ is also an estimated value from the neural network and so, again, the convergence of Q-Loss cannot be guaranteed. But its utility can be guaranteed by averaging the rewards returned.

VI. Experimental results

A. Environment Setup

To quantitatively evaluate the performance of our approach, we developed a Python simulator via OpenAI Gym 0.18.0 and

PyTorch 1.9.1. The main configurations of the experiment are listed in Table II.

TABLE II: The experimental settings

Notation	Description	Value
α	the learning rate	0.01
γ	the discount rate	0.99
D	the memory space size	512
I	the maximal iteration	2000
$d_{initial}$	the initial resource of edge host	10
ε_{start}	the initial parameter of ε-greedy method	0.9
ε_{end}	the final parameter of ε-greedy method	0.05
ϵ	privacy budget	10

For all experiments, a server was assumed to be connected to n_e edge hosts, with each edge host having m initial resources. The queue of tasks that each edge host needs to process was dynamically generated in the first 5 time steps, following a Poisson distribution. This means that, in the first 5 time steps, each time step would generate λ tasks into the queue with the generated task size following a discrete uniform distribution with a mean of $\bar{X}$.

We evaluated LDP-CFRL's performance in three different scenarios – first, with the server connected to a different number of edge hosts. Here, n_e was set to 5, 10, and 15, while the total number of tasks was fixed at 10, and $\bar{X}$ was fixed at 10.

In the second scenario, we varied the number of tasks processed by the edge host. We set the mean rate of the Poisson distribution for generating the tasks λ to 5, 10, and 15. The number of edge hosts n_e was fixed at 5 and $\bar{X}$ was fixed at 10.

In the third scenario, we varied the size of the tasks on the edge hosts. The mean value of the discrete uniform distribution of the task size $\bar{X}$ was set to 5, 10, and 15, while the number of edge hosts n_e was fixed at 5, and λ was fixed at 10.

In all scenarios, the initial number of resources for edge hosts was set to 10, and the initial sliding window lengths for edge hosts and the server were set to 6 and 10, respectively.

Further, in all three scenarios, we compared LDP-CFRL to the following two methods:

1. Concurrent federated reinforcement learning (CFRL) [22]. This method sees the edge hosts and the server cooperating to manage task processing and resource allocation. CFRL performs well in terms of task completion speeds and resource utilization. However, it does not come with theoretical guarantee of data privacy for the edge hosts. By comparison, LDP-CFRL allows each edge host to implement a differential noise mechanism in the local training phase, which is acceptable if it does not significantly reduce performance.
2. Deep reinforcement learning (DRL) [17]. This is a widely used method for resource allocation in edge computing systems. This method is similar to CFRL in the behavior of edge hosts, but the server does not dynamically allocate resources to the edge hosts.

We used the following three metrics to evaluate performance:

- **average utility**, which reflects the reward for completing a task;
- **average completion time step**, which shows the speed at which the edge hosts completed their series of tasks;
- **average resource utilization**, which represents the ratio of resources actually used by each task to the amount allocated.

B. Results

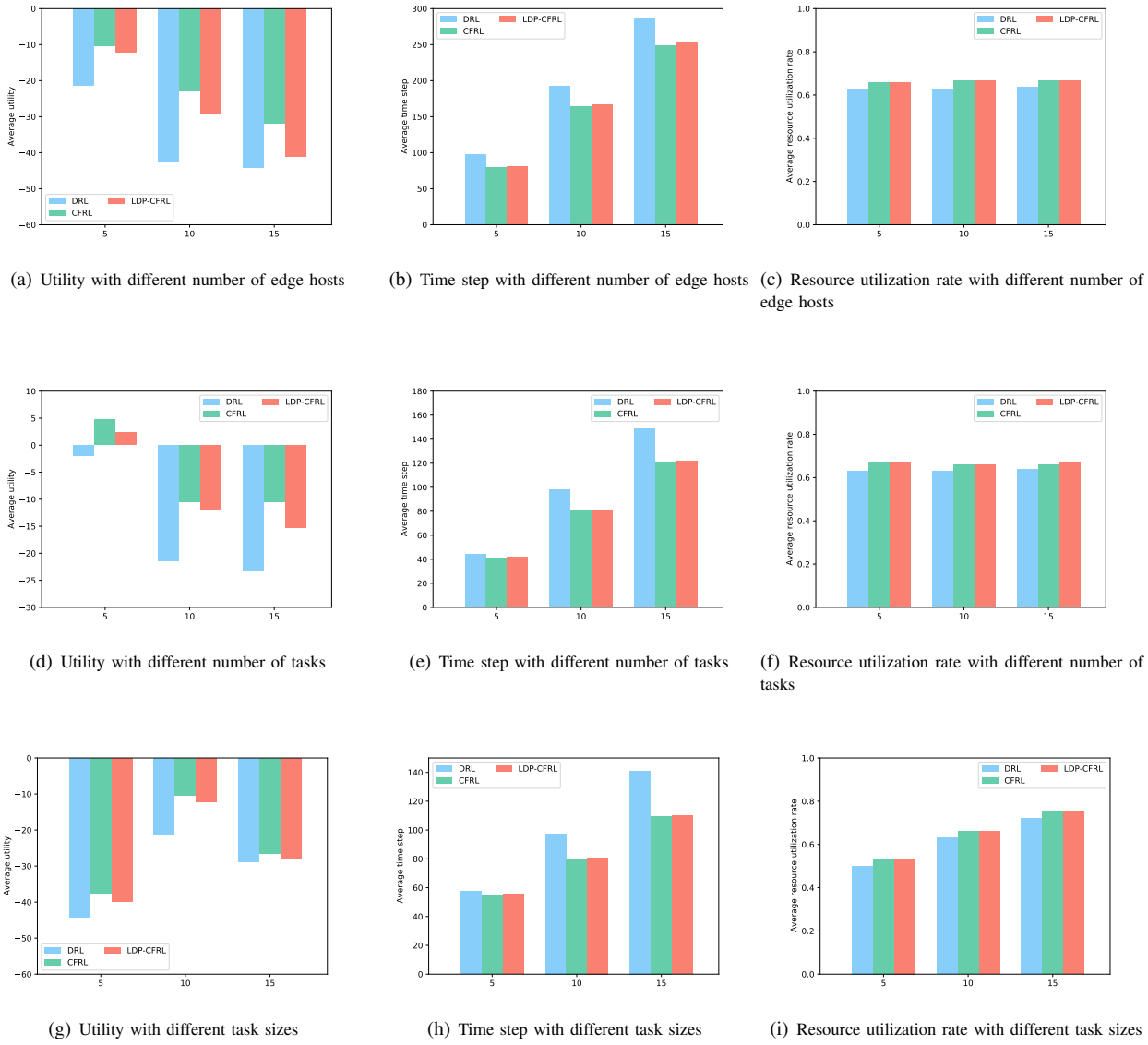
Fig. 6 shows the average utility, average time steps, and average resource utilization of the three methods after training in all three scenarios. As can be seen, LDP-CFRL's average utility was slightly lower than CFRL's in each scenario but higher than DRL's. Further, both the methods that use a joint decision-making strategy, LDP-CFRL and CFRL, took less time to complete all tasks. The reason for DRL's sluggishness is that it is incapable of planning a global strategy. Notably, LDP-CFRL produced acceptable results despite adding noise during local training.

Turning to each of the scenarios, Fig. 6(a) shows how the number of edge hosts affects average utility. Interestingly, as the number of hosts increased, the difference between DRL and LDP-CFRL decreased. This is because as more edge hosts join the joint decision-making process, it becomes more difficult to make decisions. Additionally, the difference in average utility between LDP-CFRL and CFRL also increased. This is because each edge host adds noise during the local training phase. So, as the number of edge hosts increases, the amount of noise in the system increases, resulting in a decrease in utility.

In terms of the second scenario, Fig. 6(d) shows that as the number of tasks increases, the difference in average utility between all three methods decreases. This is because the edge hosts can transfer tasks to the server, but doing so costs the host part of its reward. Therefore, as the number of tasks increases, more tasks get transferred to the server, and, the average rewards received by the edge hosts decreases. Fig. 6(f) shows an interesting result. Here, the average resource utilization rate of LDP-CFRL is slightly higher than for CFRL. This is because adding noise alleviates over-fitting in the CFRL method.

Fig. 6(d) shows that the differences in average utility between the three methods decreases as the number of tasks increases. This is because more tasks bring about more complex resource allocation decision-making problems, and algorithms have better performance when processing low-complexity situations. It can also be seen that in the scenario of small-number of tasks, the system utility of CFRL and LDP-CFRL methods is positive. The system reward function of this paper is negative in most cases, except when generating very reasonable resource allocation strategies, which indicates that in the case of small-number of tasks, the algorithm proposed in this paper can reasonably complete the task. Fig. 6(e) shows that the system time of LDP-CFRL method does not decrease significantly as a result of adding noise. In comparison, the system time of DRL method is significantly lower in multi-task scenarios. This is because DRL method cannot obtain global information of the system, and the complex decision-making scenarios in multi-task scenarios make it unable to

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3312118



(a) Utility with different number of edge hosts (b) Time step with different number of edge hosts (c) Resource utilization rate with different number of edge hosts

(d) Utility with different number of tasks (e) Time step with different number of tasks (f) Resource utilization rate with different number of tasks

(g) Utility with different task sizes (h) Time step with different task sizes (i) Resource utilization rate with different task sizes

Fig. 6: Performance comparison between LDP-CFRL method, CFRL method, and DRL method

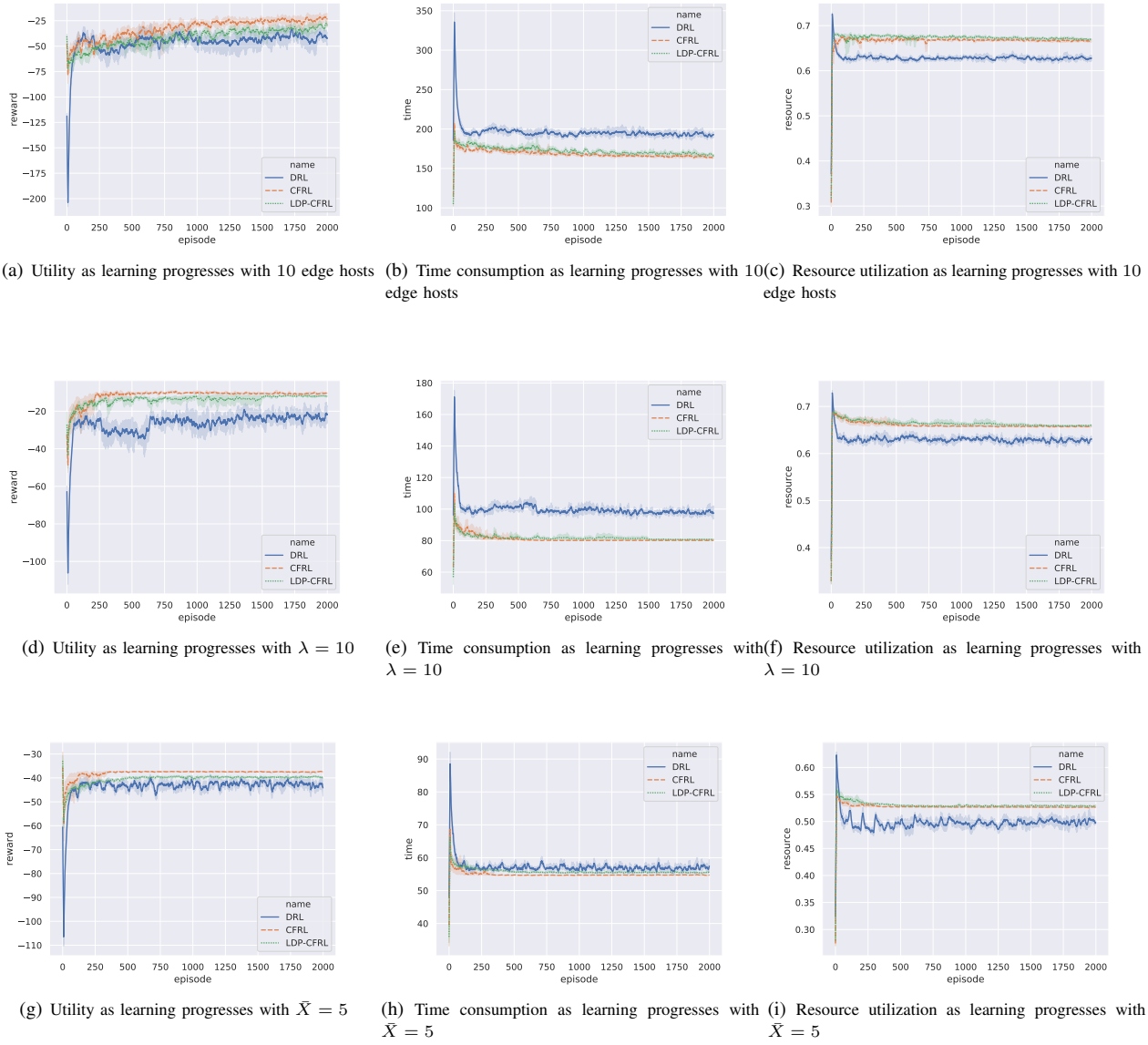
generate optimal resource allocation strategies. Fig. 6(f) shows an interesting result. In the scenario of large number of tasks, the average resource utilization rate of LDP-CFRL method is slightly higher than CFRL. This is because the added noise to the training process alleviates the overfitting problem of CFRL method to some extent, resulting in a small increase in resource utilization rate. In both scenarios of small and large number of tasks, LDP-CFRL method has a certain advantage over DRL method, which demonstrates the effectiveness of our method in protecting privacy while ensuring high resource utilization rate.

In the third scenario, Fig. 6(g) shows that all three methods perform better with an average task size of 10. This is because each edge host was given 10 initial resources. Hence, we see better utility when processing tasks with a similar mean to the discrete uniform distribution. In Fig. 6(i), we see the resource utilization rate for all three methods improve as the size of the tasks processed by the edge hosts increases. Tasks with small means are prone to wasting resources. For example, if 8 resources are allocated to a task of size 5, the resource utilization rate is only 0.625.

Fig. 7 shows the three metrics during the training process. Note that all three methods are based on a framework of deep Q-learning, and all training processes were run for 2000 epochs. We trained each model 5 times and averaged the results to smooth the performance curves.

Figs. 7(a), 7(d) and 7(g) show the average utility at each training epoch. We can observe that LDP-CFRL and CFRL exhibit similar trends. In the early stages of training, the utility values are lower. This is because, during this stage, the agents focus more on exploring their environment. Subsequently, the agents adopt a greedy strategy to increase their utility. The added Gaussian noise leads to slightly lower overall utility for LDP-CFRL compared to CFRL. However, compared to

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3312118



(a) Utility as learning progresses with 10 edge hosts (b) Time consumption as learning progresses with 10 edge hosts (c) Resource utilization as learning progresses with 10 edge hosts

(d) Utility as learning progresses with $\lambda = 10$ (e) Time consumption as learning progresses with $\lambda = 10$ (f) Resource utilization as learning progresses with $\lambda = 10$

(g) Utility as learning progresses with $\bar{X} = 5$ (h) Time consumption as learning progresses with $\bar{X} = 5$ (i) Resource utilization as learning progresses with $\bar{X} = 5$

Fig. 7: Performance comparison between LDP-CFRL method, CFRL method and DRL method with training iterations

DRL, LDP-CFRL demonstrates higher exploration efficiency, faster utility improvement, and still maintains a high level of usability. As shown in Fig. 7 (b), 7(e) and 7(h), the high number of average time steps to complete their tasks at the beginning of training also reflects that the agents were still in the exploratory stage at this time.

Further, Figs. 7(c), 7(f) and 7(i) show the average resource utilization rate at each training epoch. At the beginning of training, the resource utilization rate of the DRL method reached a very high level and then dropped sharply. This is because DRL must sacrifice part of its resource utilization rate to improve average utility and reduce time costs. Moreover, it can be seen from the performance curve that the resource utilization rate of LDP-CFRL was slightly higher than that of the CFRL method. This indicates that adding noise improves generalization, while the fault tolerance of the model alleviates over-fitting.

VII. Conclusion

In this paper, we addressed the challenge of resource allocation in scenarios where the server is partially untrusted. We proposed a novel federated reinforcement learning algorithm called LDP-CFRL, which integrates local differential privacy mechanisms into the previously proposed CFRL algorithm. Our algorithm introduces Gaussian noise satisfying local differential privacy to the training gradients of edge hosts, effectively protecting data privacy even in the presence of an untrusted server. The privacy guarantees of our algorithm are theoretically analyzed and proven.

Our key contribution lies in providing a practical solution that ensures both efficient resource allocation and strong privacy guarantees. By integrating local differential privacy into the federated reinforcement learning framework, we enable edge hosts to collaboratively train a resource allocation

model while preserving the privacy of their local data. This approach offers a higher level of privacy protection compared to traditional allocation algorithms.

Furthermore, our experimental results demonstrate that the LDP-CFRL algorithm achieves notable performance in terms of efficiency and task processing speed. Through extensive evaluations, we have shown that our approach outperforms existing allocation algorithms in terms of both privacy preservation and resource utilization.

In summary, our work makes a significant contribution by presenting a novel federated reinforcement learning algorithm, LDP-CFRL, that addresses the resource allocation challenge in partially untrusted server scenarios. Our algorithm integrates local differential privacy mechanisms, providing a robust privacy guarantee for edge hosts and ensuring the privacy of the entire system. We believe that our research opens new avenues for privacy-preserving resource allocation in edge computing environments.

ACKNOWLEDGEMENT

The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship.

REFERENCES

[1] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[2] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep reinforcement learning for mobile 5g and beyond: Fundamentals, applications, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 44–52, 2019.

[3] T. Zhu, D. Ye, W. Wang, W. Zhou, and P. Yu, "More than privacy: Applying differential privacy in key areas of artificial intelligence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2824–2843, 2020.

[4] A. Vlahou, D. Hallinan, R. Apweiler, A. Argiles, J. Beige, A. Benigni, R. Bischoff, P. C. Black, F. Boehm, J. Céraline *et al.*, "Data sharing under the general data protection regulation: time to harmonize law and research ethics?" *Hypertension*, vol. 77, no. 4, pp. 1029–1035, 2021.

[5] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, "Evaluating gradient inversion attacks and defenses in federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7232–7241, 2021.

[6] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 2019, pp. 2512–2520.

[7] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441–9455, 2020.

[8] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[9] H.-K. Lim, J.-B. Kim, J.-S. Heo, and Y.-H. Han, "Federated reinforcement learning for training control policies on multiple iot devices," *Sensors*, vol. 20, no. 5, p. 1359, 2020.

[10] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu, "More than privacy: Adopting differential privacy in game-theoretic mechanism design," *ACM Computing Surveys (CSUR)*, vol. 54, no. 7, pp. 1–37, 2021.

[11] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[12] M. Yang, L. Lyu, J. Zhao, T. Zhu, and K.-Y. Lam, "Local differential privacy and its applications: A comprehensive survey," *arXiv preprint arXiv:2008.03686*, 2020.

[13] X. Liu, Z. Qin, and Y. Gao, "Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning," in *ICC 2019-2019 IEEE international conference on communications (ICC)*, 2019, pp. 1–6.

[14] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource Allocation with Edge Computing in IoT Networks via Machine Learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020.

[15] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-Agent Reinforcement Learning for Resource Allocation in IoT Networks with Edge Computing," *China Communications*, pp. 220–236, 2020.

[16] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart Resource Allocation for Mobile Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 03, pp. 1529–1541, 2021.

[17] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource Allocation Based on Deep Reinforcement Learning in IoT Edge Computing," *IEEE Journal of Selected Areas in Communications*, vol. 38, no. 6, pp. 1133–1146, 2020.

[18] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 937–16 947, 2020.

[19] B. Zhao, K. R. Mopuri, and H. Bilen, "idlg: Improved deep leakage from gradients," *arXiv preprint arXiv:2001.02610*, 2020.

[20] Y. Zhan, P. Li, and S. Guo, "Experience-Driven Computational Resource Allocation of Federated Learning by Deep Reinforcement Learning," in *Proc. of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 234–243.

[21] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When Deep Reinforcement Learning Meets Federated Learning: Intelligent Multi-Timescale Resource Management for Multi-access Edge Computing in 5G Ultra Dense Network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2020.

[22] T. Zhu, W. Zhou, D. Ye, Z. Cheng, and J. Li, "Resource allocation in iot edge computing via concurrent federated reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1414–1426, 2022.

[23] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.

[24] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, 2020, pp. 61–66.

[25] H. Ono and T. Takahashi, "Locally private distributed reinforcement learning," *arXiv preprint arXiv:2001.11718*, 2020.

[26] K. Peng, M. Li, H. Huang, C. Wang, S. Wan, and K.-K. R. Choo, "Security challenges and opportunities for smart contracts in internet of things: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 004–12 020, 2021.

[27] J. Lu, H. Liu, R. Jia, J. Wang, L. Sun, and S. Wan, "Towards personalized federated learning via group collaboration in iiot," *IEEE Transactions on Industrial Informatics*, pp. 1–10, 2022.

[28] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Federated deep reinforcement learning for resource allocation in o-ran slicing," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*, 2022, pp. 958–963.

[29] A. A. Khalil and M. A. Rahman, "Fed-up: Federated deep reinforcement learning-based uav path planning against hostile defense system," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 268–274.

[30] H. Yang, J. Zhao, Z. Xiong, K.-Y. Lam, S. Sun, and L. Xiao, "Privacy-preserving federated learning for uav-enabled networks: Learning-based joint scheduling and resource management," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3144–3159, 2021.

[31] P. Zhang, C. Wang, C. Jiang, and Z. Han, "Deep reinforcement learning assisted federated learning algorithm for data management of iiot," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8475–8484, 2021.

[32] X. Chen and G. Liu, "Federated deep reinforcement learning-based task offloading and resource allocation for smart cities in a mobile edge network," *Sensors*, vol. 22, no. 13, p. 4738, 2022.

[33] C. Wang, X. Wu, G. Liu, T. Deng, K. Peng, and S. Wan, "Safeguarding cross-silo federated learning with local differential privacy," *Digital Communications and Networks*, vol. 8, no. 4, pp. 446–454, 2022.

[34] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

[35] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[36] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.
[37] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1175–1191.
[38] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.
[39] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated Learning With Differential Privacy: Algorithms and Performance Analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.
[40] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual international conference on the theory and applications of cryptographic techniques*, 2006, pp. 486–503.
[41] Y. Chen and M. Wang, "Stochastic primal-dual methods and sample complexity of reinforcement learning," *arXiv preprint arXiv:1612.02516*, 2016.
[42] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations research*, vol. 51, no. 6, pp. 850–865, 2003.

Wei Zhou received his MA.Eng degree in Computer Science from School of Computer Science and Technology, China University of Geosciences, Wuhan in 2023. Before that, he received his Bachelor of Science degree of Electrical Engineering from Changchun University of Science and Technology, in 2020. His research interests include reinforcement learning and differential privacy.

Tianqing Zhu received her B.Eng. degree and her M.Eng. degree from Wuhan University, Wuhan, China, in 2000 and 2004, respectively. She also holds a PhD in computer science from Deakin University, Australia (2014). She is currently an associate professor in university of technology Sydney, Australia. Prior to that, she was a Lecturer with the School of Information Technology, Deakin University, from 2014 to 2018. Her research interests include privacy-preserving and cyber security.

Dayong Ye Dayong Ye received his MSc and PhD degrees both from University of Wollongong, Australia, in 2009 and 2013, respectively. Now, he is a research fellow of Cyber-security at University of Technology, Sydney, Australia. His research interests focus on differential privacy, privacy preservation, and multiagent systems.

Wei Ren Wei Ren is a full Professor at the School of Computer Science, China University of Geosciences Wuhan, China since 2013. He was with Department of Electrical and Computer Engineering, Illinois Institute of Technology, USA in 2007 and 2008. He was with School of Computer Science, University of Nevada Las Vegas, USA in 2006 and 2007. He was with the Department of Computer Science, Hong Kong University of Science and Technology, in 2004 and 2005. He obtained Ph.D. degree in Computer Science from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2006. He published more than 150 refereed papers, 2 monographs, and 5 textbooks. He obtained 30 patents and 6 innovation awards. He is a Distinguished Member of China Computer Federation.

Kim-Kwang Raymond Choo Kim-Kwang Raymond Choo received his Ph.D. in Information Security in 2006 from the Queensland University of Technology, Australia. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio (UTSA). In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher), 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, Outstanding Associate Editor of 2018 for IEEE Access, British Computer Society's 2019 Wilkes Award Runnerup, 2019 EURASIP Journal on Wireless Communications and Networking (JWCN) Best Paper Award, Korea Information Processing Society's Journal of Information Processing Systems (JIPS) Survey Paper Award (Gold) 2019, IEEE Blockchain 2019 Outstanding Paper Award, IEEE TrustCom 2018 Best Paper Award, ESORICS 2015 Best Research Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, and Co-Chair of IEEE Multimedia Communications Technical Committee's Digital Rights Management for Multimedia Interest Group.