User-Oriented Edge Node Grouping in Mobile Edge Computing

Qing Li, Xiao Ma, *Member, IEEE,* Ao Zhou, *Member, IEEE,* Xiapu Luo, *Member, IEEE,* Fangchun Yang, *Senior Member, IEEE,* and Shangguang Wang, *Senior Member, IEEE*

Abstract—In mobile edge computing networks, densely deployed access points are empowered with computation and storage capacities. This brings benefits of enlarged edge capacity, ultra-low latency, and reduced backhaul congestion. This paper concerns edge node grouping in mobile edge computing, where multiple edge nodes serve one end user cooperatively to enhance user experience. Most existing studies focus on centralized schemes that have to collect global information and thus induce high overhead. Although some recent studies propose efficient decentralized schemes, most of them did not consider the system uncertainty from both the wireless environment and other users. To tackle the aforementioned problems, we first formulate the edge node grouping problem as a game that is proved to be an exact potential game with a unique Nash equilibrium. Then, we propose a novel decentralized learning-based edge node grouping algorithm, which guides users to make decisions by learning from historical feedback. Furthermore, we investigate two extended scenarios by generalizing our computation model and communication model, respectively. We further prove that our algorithms converge to the Nash equilibrium with upper-bounded learning loss. Simulation results show that our mechanisms can achieve up to 96.99% of the oracle benchmark.

Index Terms—Mobile edge computing, edge node grouping, game theory, reinforcement learning.

1 Introduction

Mobile edge computing provides cloud-like capabilities in proximity to end-users by equipping the network edge with computational and storage resources [1, 2]. Endowing densely deployed access points with edge servers is one of the main forms of edge computing deployment in 5G networks [3]. It brings many benefits, such as augmented computation capacity, ultra-low latency, and reduced backhaul congestion [4]. In such edge computing networks, it is hard for any individual edge node to provide guaranteed quality of service for users due to the limited computation capacity, radio coverage, and unpredicted wireless environment. With the dense deployment of edge nodes, it becomes possible that multiple edge nodes serve a single user cooperatively such that user experience can be enhanced.

This paper concerns the edge node grouping problem where each user selects multiple edge nodes to form a group to execute tasks in parallel. While cooperative computation brings many benefits, the dynamics of the wireless environment and limitation of edge node capacities can influence user experience. Besides, when multiple users offload tasks to an identical edge node, competition for the communication and computation resources will deteriorate cooperative computation performance. Hence, it is non-trivial to properly choose an edge node group for each user to adapt to wireless environment dynamics and peer competition. Most pioneering studies [5–22] adopt the centralized architecture, which requires collecting global information (from users, edge nodes, and network environment) and thus causes high overhead, especially in large scale networks. Distributed schemes [23–27] have been recently proposed based on game theory, where each user makes its computation offloading decision autonomously. These works have one key limitation that they did not consider system uncertainty from both the wireless environment and other users, which is usually impractical. First, the wireless environment is highly time-varying and hard to learn in advance while most decentralized schemes [23–26] assume that wireless environment information is known before making decisions. Second, selfish users are reluctant to share their strategies when competing for resources while all the decentralized schemes assume that each user knows the strategies of the other users.

To address the aforementioned limitations, we focus on edge node grouping under dual uncertainty from both the wireless environment and the other users. First, since the wireless environment information is unknown before making decisions, each user has to learn edge node grouping strategies from the feedback of historical decisions. Second, when multiple users make edge node grouping decisions simultaneously without joint strategy profiles, they are oblivious to others' actions, which may undermine the convergence and incur large learning performance loss. Third, feedback may be delayed due to the poor wireless environment or the limited computational capacity in edge nodes. This delay may cause disorder and absent feedback and result in learning performance loss.

In this paper, we propose a novel decentralized learning mechanism to solve the edge node grouping problem in mobile edge computing by leveraging game theory and

- *Qing Li, Xiao Ma, Ao Zhou, Fangchun Yang, and Shangguang Wang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China.*
E-mail: {q_li;maxiao18;aozhou;fcyang;sgwang}@bupt.edu.cn. Xiao Ma is the corresponding author.
- *Xiapu Luo is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China.*
E-mail: csxluo@comp.polyu.edu.hk.

reinforcement learning. Specifically, we model the multi-user edge node grouping problem as a game, where each user selects an edge node group to maximize its long-term energy-aware utility. We prove that the game is an exact potential game with a unique Nash Equilibrium (NE). To cope with the unknown wireless environment information, we introduce the reinforcement learning mechanism into the game model. In the game, each user learns its strategy from historical feedback (of edge nodes) via trial-and-error interactions with other users and the dynamic wireless environment. To tackle the absence of other users' strategies, we employ better-response dynamics [28] and no-regret dynamics [29] to guide each user to respond to other users based on feedback. Besides, we further investigate the edge node grouping problem in two extended scenarios. In the first extended scenario, feedback is delayed due to limited computation capacity and we enhance our algorithms by adding a feedback queue for each strategy to adapt to the scenario with delayed feedback. In the second extended scenario, we prove that our algorithms can be applied to another common access scenario where each edge node has multiple frequency bands.

In summary, we have the following major contributions.

- We investigate decentralized mechanisms for edge node grouping. We formulate this problem as a non-cooperative game and prove that it is an exact potential game with a unique NE.
- To deal with the system uncertainty from both wireless environment and other users, we propose a novel decentralized learning-based edge node grouping algorithm by leveraging better-response dynamics and no-regret dynamics. It is proved that the mechanism admits NE with bounded learning loss.
- We extend our study to two scenarios i.e., the scenario with delayed feedback and the scenario where each edge node has multiple frequency bands.
- Simulations demonstrate that our mechanisms can converge to NE with no-regret and achieve 96.99% of the oracle benchmark.

The remainder of this paper is organized as follows. We introduce the related work in Section 2. We describe the system model and formulate the edge node grouping game in Section 3. In Section 4, we design the distributed learning-based edge node grouping algorithm. In Section 5, we discuss two extended scenarios. The simulation results are illustrated in Section 6. We conclude in Section 7.

2 Related Works

In this section, we analyze the related works which can be divided into two categories: centralized schemes and game theory based decentralized schemes.

Centralized Cooperative Computation Schemes [5–22]. Cooperative computation has obvious advantages in reducing computation delay by parallelizing the algorithm of computation and distributing the computation workload to different parts. We divide these works into two categories: offline schemes and online schemes. The offline schemes are as follows. Gong [5] provides some useful insights for the optimal computation-communication co-design for parallel computing to reduce delays in mobile edge computing. Kim *et al.* [8] develop an optimization methodology for communication resource management in wireless D2D networks to maximize energy efficiency. Zhu *et al.* [9] study multiple servers cooperatively computing a set of interdependent tasks. Chen *et al.* [13] propose a tripartite dynamic cooperation framework among unmanned aerial vehicles, unmanned ground vehicles, and base stations during the post-disaster rescue. Funai *et al.* [17] provide insight into how communication and computation complexity affect the distribution of tasks in a cooperative multi-hop ad hoc network, and show the achievable trade-offs in terms of computational delay and network lifetime. Lin *et al.* [10] exploit abundant computation resources distributed at massive wireless devices for cooperative computing to enhance the computation performance. All the above works make offloading decisions at the compile-time, where all required information is available. It needs complete information about the system and provides solutions that satisfy the given requirements. For example, Gong [5] assumes full information knowledge before decision making, such as orders of forward and backward communications between edge devices, delays of forward and backward communications, the total computation workload, and computation rates of edge devices.

For online schemes, He *et al.* [22] propose an auction-based incentive mechanism, where users participate in the system dynamically. Kao *et al.* [18] propose a novel algorithm that learns the performance of unknown devices and channel qualities through exploratory probing and makes task assignment decisions by exploiting the gained knowledge. Feng *et al.* [15] develop a cooperative computation offloading and resource allocation algorithm to maximize the computation rate of mobile edge computing systems. He *et al.* [19] explore a novel cooperative offloading mechanism based on deep reinforcement learning to improve the QoE of users. Cui *et al.* [20] propose a software-defined cooperative offloading model and design an online task scheduling algorithm to save mobile devices' energy and reduce the traffic on access links. Chen *et al.* [21] study cooperation among edge nodes via workload peer offloading. All these works make offloading decisions during the run-time of the system. The decisions are based on both process characteristics and the current state of the system. We notice that online schemes can accommodate dynamic changes in the system without requiring information about future system dynamics by leveraging the Lyapunov technique, but they cannot make the best use of system resources to provide optimal offloading decisions.

All the above works focus on centralized cooperative computation and require global information about user demands and system resources to make offloading decisions. The centralized schemes can potentially find a globally optimal solution. However, they are vulnerable regarding the scalability and the computational complexity issue.

Decentralized Schemes Based on Game Theory [23–27]. Unlike centralized schemes, decentralized schemes consider multiple authority and orchestrator nodes to make offloading decisions. Generally distributed in the network, the management elements compute offloading decisions based

TABLE 1: Related Work Comparison.

References	Centralized/ Decentralized	Online/ Offline	Mathematical method	Key assumptions
[5–7]	Centralized	Offline	Heuristic method	Global information
[8–12]	Centralized	Offline	Convex optimization	Global information
[13, 14]	Centralized	Offline	Mixed integer linear programming	Global information
[15]	Centralized	Offline	Matching theory	Global information
[16–19]	Centralized	Online	Reinforcement learning	Global information
[20, 21]	Centralized	Online	Stochastic optimization	Global information
[22]	Centralized	Online	Auction mechanism	Global information
[23–26]	Decentralized	Offline	Game theory	Local complete information
[27]	Decentralized	Online	Game theory and reinforcement learning	Unknown CSI
Ours	**Decentralized**	**Online**	**Game theory and reinforcement learning**	**Unknown CSI and joint strategy**

on local resources and information. This is more flexible and can be more efficient to handle the dynamic changes of infrastructure like edge computing without resorting to network-wide computations. The decentralized schemes help to address the scalability and the locality awareness issues and allow offloading decisions that best fit with the local context. However, no guarantees are provided regarding the global optimality of the computed solutions. The authors design a decentralized potential game based offloading algorithm to minimize the cost of each mobile device in both [23] and [24]. Chen *et al.* [25] devise a distributed computation offloading algorithm that achieves an NE of the multi-user computation offloading game. They derive the upper bound of the convergence time under mild conditions. Gao *et al.* [26] propose a two-level decentralized approach in network coding assisted D2D communications. Dinh *et al.* [27] adopt an unknown payoff game framework and propose a reinforcement learning offloading mechanism that helps users learn their long-term offloading strategies to maximize their long-term utilities. All these works [23–27] cannot address the system uncertainty from both wireless environment and other users. To address these limitations, our work devotes efforts to grouping edge nodes without knowing both the wireless channel state and other users' strategies. We summarize the comparison of related works from four important dimensions: online/offline, centralized/decentralized, the mathematical method they applied, and information collecting assumptions as in Table 1.

3 System Model

In the mobile edge computing networks, the macro base station confirms the complete coverage of User Equipments (UEs). Both the macro base station and other access points (e.g., micro base stations) are equipped with edge servers and named edge nodes. The edge nodes provide computation and storage capabilities for UEs. We investigate grouping among densely deployed edge nodes to enable flexible cooperative computation for each UE dynamically. To better capture the system dynamics, the system is assumed to operate in a slotted structure and its timeline is discretized into time slots $\{1 \leq t \leq T\}$. Each time slot is a decision round with a duration of T^{dur}. Each UE has a computation-intensive task with a large amount of data to offload to edge nodes for processing. Autonomous UEs choose proper edge nodes to form an edge node group and schedule the transmission power among edge nodes in its group. We consider that the task is divisible without task dependency. Typical examples are virus scanning and video streaming analytics [27]. For these tasks, the execution time can be much longer (e.g., several hours) compared with the duration of each decision round (e.g., tens of milliseconds). In each time slot, the cooperative computation procedure works as follows.

Cooperative Computation Procedure. 1) *Task Preprocessing*: Each UE divides the tasks into sub-tasks for cooperative computation. 2) *Edge Node Grouping*: Each UE performs the edge node grouping scheme shown as follows. 3) *Task Transmitting*: Each UE offloads the sub-tasks to the edge nodes in its group. 4) *Task Processing*: edge nodes in the group of each UE process offloaded tasks. 5) *Result Returning*: edge nodes return the results to UEs after completing the computation. In addition, UEs are informed of the transmitted data size to each edge node, which is called *feedback*.

Edge Node Grouping Scheme. 1) *Preparation*: When a UE has task requests, it selects several edge nodes as a candidate set according to the received signal strength. 2) *Decision Making*: According to the proposed algorithm in Section 4, the UE chooses a group of edge nodes from the candidate set and allocates its transmission power among all the edge nodes in its group. If the size of the candidate set is smaller than the group size, UE chooses the candidate set as the serving group directly. If the candidate set is empty, the UE offloads tasks to the macro base station. 3) *Execution*: The UE and edge nodes in its group complete the connection process assisted by the macro base station.

3.1 Edge Node Grouping Strategy

We assume that there are a UE set $\mathcal{N} = \{1, 2, \cdots, N\}$ and an edge node set $\mathcal{M} = \{1, 2, \cdots, M\}$. Let $\mathbf{a}_i^t = (a_{i,1}^t, \ldots, a_{i,M}^t)$ denote the M dimensional association vector between UE i and edge nodes in slot t. If UE i offloads its task to edge node j $(1 \leq j \leq M)$ at time slot t, then $a_{i,j}^t = 1$, otherwise $a_{i,j}^t = 0$. Edge node grouping modes can be classified into number-based cooperation modes [30] and distance-based cooperation modes [31]. In the former, each UE offloads its task to a constant number L edge nodes simultaneously. In the latter, each UE is served by all edge nodes within a certain distance. It is hard to implement the distance-based cooperation mode in practice because a user does not directly know its distances to edge nodes when making edge node grouping decisions [32]. Hence, this paper adopts the number-based cooperation mode for edge node grouping, i.e.,

$$\sum_{j=1}^{M} a_{i,j}^{t} = L, \forall i \in \mathcal{N}, 1 \leq t \leq T. \tag{1}$$

where L is the edge node group size. We assume that the transmission power is quantified into discrete levels. UE i adjusts transmission power $p_{i,j}^{t}$ on the wireless link of edge node j in time slot t. The sum of transmission power on the wireless links of different edge nodes cannot exceed the maximal transmission power $P_{i,\max}$,

$$\sum_{j=1}^{M} p_{i,j}^{t} \leq P_{i,\max}, \forall i \in \mathcal{N}, 1 \leq t \leq T. \tag{2}$$

We denote the edge node grouping strategy of UE i at time slot t as $\mathbf{s}_i^t = \{a_{i,j}^t, p_{i,j}^t\}$. The joint strategy of all UEs at time slot t is $\mathbf{s}^t = (\mathbf{s}_1^t, \cdots, \mathbf{s}_i^t, \cdots, \mathbf{s}_N^t)$ and lies in $\mathcal{S} = \prod_{i \in \mathcal{N}} \mathcal{S}_i$ where $\mathcal{S}_i$ refers to the domain of $\mathbf{s}_i^t, (1 \leq t \leq T)$. The joint strategy across all time slots is $\{\mathbf{s}^1, \cdots, \mathbf{s}^t, \cdots, \mathbf{s}^T\}$.

3.2 Communication Model

In the wireless communication model, edge nodes are assumed to transmit tasks over non-overlapping frequency bands such that inter-cell interference can be avoided in the uplink [33]. Similar to [25, 27], we focus on exploring the wireless interference model where multiple UEs connected to the same edge node share the spectrum band of the edge node. This can be realized by some physical layer channel access scheme (e.g., code division multiple access). We compute the total transmission rate of UE i as [27],

$$R_i^t(\mathbf{s}^t) = \sum_{j=1}^{M} W_j \log_2 \left(1 + \frac{a_{i,j}^t p_{i,j}^t h_{i,j}^t}{N_0 + \sum\limits_{m \in \mathcal{N} \setminus \{i\}} a_{m,j}^t p_{m,j}^t h_{m,j}^t}\right), \tag{3}$$

where W_j is the channel bandwidth, N_0 represents the background noise power, and $h_{i,j}^t$ is the channel gain between UE i and edge node j. Uplink inter-UE interference occurs when multiple UEs simultaneously transmit to the same edge node. The transmission energy consumption of UE i is

$$E_i^t(\mathbf{s}_i^t) = T^{\text{trans}} \sum_{j=1}^{M} a_{i,j} p_{i,j}^t, \tag{4}$$

where T^{trans} is the duration of the transmission period and identical for all UEs. We will analyze the scenario in which each edge node has multiple frequency bands in Section 5.2.

3.3 Computation Model

We consider the elastic capacity model where edge nodes have sufficient computation resources and the computation capacities can scale with task requests. The computation delay at each edge node is relatively small and the computation result and feedback can be returned timely in the current time slot. We denote the amount of the received CPU cycles by UE i in time slot t as,

$$F_i^t(\mathbf{s}^t) = \kappa T^{\text{trans}} R_i^t(\mathbf{s}^t), \tag{5}$$

where κ means the amount of CPU cycles needed for processing one unit of data. We will discuss the extended scenario where each edge node has a fixed computation capacity, which can lead to delayed feedback in Section 5.1.

3.4 Utility Function

Since tasks of each UE will last a long time, it makes sense for each UE to opportunistically obtain as many CPU cycles to finish the task earlier. The amount of received CPU cycles is linear with the data transmission rate as in (5). Therefore, each UE will increase its transmission power levels to maximize the data transmission rate in each time slot according to (3). This will increase energy consumption but may not improve performance due to communication interference among UEs. Hence, we define an energy-aware utility model,

$$\mathcal{U}_i^t(\mathbf{s}^t) = \lambda_1 F_i^t(\mathbf{s}^t) - \lambda_2 E_i^t(\mathbf{s}_i^t), \tag{6}$$

where $\lambda_1, \lambda_2 \in [0, \infty)$ are weight factors. The weight factors reflect the relative importance between the received CPU cycles and energy consumption. We define payoff as the long-term average utility,

$$\bar{\mathcal{U}}_i(\mathbf{s}^1, \cdots, \mathbf{s}^T) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathcal{U}_i^t(\mathbf{s}^t). \tag{7}$$

3.5 Problem Formulation

In this section, we introduce the repeated game with unknown payoff framework, where players repeatedly play the same one-shot game simultaneously with unknown payoff [28]. The one-shot game is called a stage game. Each time slot is a game stage. We formulate the problem as a repeated edge node grouping game characterized by the tuple $\mathcal{G} = \{N, \{\mathbf{s}^1, \cdots, \mathbf{s}^T\}, \{\bar{\mathcal{U}}_i\}_{i \in \mathcal{N}}\}$, where each UE i makes edge node grouping strategy (including edge node grouping and transmission power allocation) repeatedly to maximize its long-term averaged utility over all T time slots.

$$\forall i \in \mathcal{N} \max_{\mathbf{s}^1, \cdots, \mathbf{s}^T \in \mathcal{S}} \bar{\mathcal{U}}_i. \tag{8}$$

We define the stage game of $\mathcal{G}$ as $\mathcal{G}' = \{N, \{\mathbf{s}_i\}_{i \in \mathcal{N}}, \{\bar{\mathcal{U}}_i\}_{i \in \mathcal{N}}\}$, where we omit the superscript t. In repeated games with perfect information, a player has observations on the environment information and historical joint strategies of all the players before choosing his strategy for the current stage game. In this paper, UEs cannot obtain full Channel State Information (CSI) and edge node grouping strategies of other UEs. Hence, we focus on the edge node grouping game with imperfect information. Fortunately, the influence of the wireless environment and other UEs can be obtained by the signal to interference and noise ratio. Therefore, we can drive an aggregated information of CSI and other UEs' strategies from the feedback about the data transmission rate. For edge node j with $a_{i,j} = 1$, the aggregated information of other UE strategies $I_{-i,j}$ in edge node j at time slot t can be expressed as,

$$I_{-i,j}^t = \frac{h_{i,j}^t}{N_0 + \sum\limits_{m \in \mathcal{N} \setminus \{i\}} a_{m,j}^t p_{m,j}^t h_{m,j}^t}. \tag{9}$$

$$\psi(\mathbf{s}_i,\mathbf{s}_{-i}) = T^{\text{trans}}\left(\lambda_1\kappa\int_{\mathbb{H}}\sum_{j=1}^{M}W_j\log_2\left(N_0+\sum_{i=1}^{N}a_{i,j}p_{i,j}h_{i,j}^t\right)f_{\mathbf{H}}dh_{1,1}^t\cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t-\lambda_2\sum_{i=1}^{N}\sum_{j=1}^{M}a_{i,j}^tp_{i,j}^t\right). \quad (13)$$

From (3), the aggregated information can be calculated as

$$I_{-i,j}^t = \frac{2^{\frac{R_{i,j}^t}{W_j}}-1}{a_{i,j}^tp_{i,j}^t}, \quad (10)$$

where $R_{i,j}^t$ denotes the data transmission rate between UE i and edge node j, and can be calculated from the feedback following (10). For edge node j with $a_{i,j}=0$, UE i transmits pilot signals with low power and the aggregated information can also be obtained from (10). Consequently, the aggregated information of other UEs' strategies can be calculated. Note that collecting the aggregated information causes neglected overhead in communication for each UE.

4 Algorithm Design

4.1 Preliminaries

We will design algorithms based on the game structure. It is helpful to analyze the property of the repeated game before designing algorithms. Since the repeated game consists of many stage games, we start by analyzing the stage game $\mathcal{G}'$.

Definition 1. *A strategy profile $\mathbf{s}^*=(\mathbf{s}_1^*,\cdots,\mathbf{s}_N^*)$ is a **Nash equilibrium** of the edge node grouping stage game $\mathcal{G}'$, if no UE can further increase its utility by unilaterally changing its strategy, i.e.,*

$$\mathcal{U}_i(\mathbf{s}_i^*,\mathbf{s}_{-i}^*)\geq\mathcal{U}_i(\mathbf{s}_i',\mathbf{s}_{-i}^*),\forall\mathbf{s}_i'\in\mathcal{S}_i. \quad (11)$$

Definition 2. *A game is an exact potential game if there exists a potential function $\psi(\mathbf{s}):\mathbb{S}\mapsto\mathbb{R}$ such that for every UE $i\in\mathcal{N}$ $s_{-i}\in\prod_{m\neq i}\mathcal{S}_m$, $\mathbf{s}_i,\mathbf{s}_i'\in\mathcal{S}_i$, it is satisfied that*

$$\mathcal{U}_i(\mathbf{s}_n',\mathbf{s}_{-i})-\mathcal{U}_i(\mathbf{s}_i,\mathbf{s}_{-i})=\psi(\mathbf{s}_i',\mathbf{s}_{-i})-\psi(\mathbf{s}_i,\mathbf{s}_{-i}), \quad (12)$$

Theorem 1. *The stage game $\mathcal{G}'=\{N,\{\mathbf{s}_i\}_{i\in\mathcal{N}},\{\bar{\mathcal{U}}_i\}_{i\in\mathcal{N}}\}$ is an exact potential game with the potential function in (13), where $\mathbf{H}=\{h_{i,j}^t\}$ is the matrix of the channel gain. $f_{\mathbf{H}}$ is the joint distribution of $\mathbf{H}$.*

Proof. The key idea is to prove the potential function $\psi(\mathbf{s})$ in (13) satisfies Definition 2. From (3)-(6), we have (14), where $\mathbf{H}_{-i}$ is the vector of $\mathbf{H}$ without $(h_{i,1}^t\cdots h_{i,M}^t)$; and $f_{\mathbf{H}_{-i}}$ is the joint distribution of $\mathbf{H}_{-i}$. $\bar{\mathcal{U}}_i(\mathbf{s}_i',\mathbf{s}_{-i})-\bar{\mathcal{U}}_i(\mathbf{s}_i,\mathbf{s}_{-i})$ can be derived in (15). $\psi(\mathbf{s}_i',\mathbf{s}_{-i})-\psi(\mathbf{s}_i,\mathbf{s}_{-i})$ can be derived in (16). We can derive that $\bar{\mathcal{U}}_i(\mathbf{s}_i',\mathbf{s}_{-i})-\bar{\mathcal{U}}_i(\mathbf{s}_i,\mathbf{s}_{-i})=\psi(\mathbf{s}_i',\mathbf{s}_{-i})-\psi(\mathbf{s}_i,\mathbf{s}_{-i})$. Thus, $\mathcal{G}'$ is an exact potential game, which always has an NE and finite improvement property [34]. □

Theorem 1 guarantees the existence and uniqueness of NE of game $\mathcal{G}'$ which has finite improvement property [34]. From Theorem 2 of [35], if s_t^* is the unique NE of the t_{th} stage game $\mathcal{G}'$, $\{s_t^*\}_{t=1}^T$ is the unique NE of the repeated game $\mathcal{G}$.

If each UE has full CSI information and joint strategies of other UEs before choosing their strategies for the current stage game, the repeated game $\mathcal{G}$ can be transformed into the stage game $\mathcal{G}'$ repeated in each time slot. In this case, the best-response dynamics based edge node grouping algorithm [34] can provide the performance upper bound theoretically. We summarize the oracle benchmark for the stage game in Algorithm 1. In the beginning, each UE randomly selects its strategy and receives the initial utility (Line 2). At each iteration of the stage game, the algorithm randomly chooses a UE to perform the best-response to the strategies of other UEs (Line 9) as,

Algorithm 1 Best-Response Based Edge Node Grouping

```
1:  for $t = 1,\cdots,T$ do
2:      $\forall i\in\mathcal{N}$, $\mathbf{s}_i$ is selected with probability $\frac{1}{|\mathcal{S}_i|}$;
3:      while s is not an NE (s must hold for at leat $T_{iter}$
        iterations to be an NE) do
4:          UE $i$ is randomly chosen to alter its action;
5:          for $\mathbf{s}_i' = 1,\cdots,|\mathcal{S}_i|$ do
6:              $\mathbf{s}\leftarrow\{\mathbf{s}_i',\mathbf{s}_{-i}\}$;
7:              Calculate the utility according to (6);
8:          end for
9:          Calculate $\mathbf{s}_i^*$ according to (17).
10:         $\forall i\in\mathcal{N},\mathcal{U}_i\leftarrow\mathcal{U}_i(\mathbf{s})$.
11:     end while
12: end for
```

$$\mathbf{s}^*=\arg\max_{\mathbf{s}_i\in\mathcal{S}_i}\mathcal{U}_i(\mathbf{s}_i,\mathbf{s}_{-i}). \quad (17)$$

According to the finite improvement property of potential games and Theorem 2 of [35], Algorithm 1 can achieve the NE of game $\mathcal{G}$ by iteratively updating their strategies.

4.2 Algorithm Overview

The best-response dynamics fails in our scenario with unknown CSI and unavailable joint strategies of other UEs. We propose a decentralized learning-based algorithm (i.e., Algorithm 3) to solve this problem. Two common evaluation criteria for learning algorithms in games are convergence [27] and regret [36]. Convergence measures if players can reach equilibrium. Regret measures how worse an algorithm performs compared to the best static strategy. We seek to improve both criteria by proposing Algorithm 3. To address the issue of convergence with unknown CSI, we combines the better-response with inertia dynamics and Q-learning in the primary learning phase following previous work [28]. To address the issue of unavailable joint strategies of other UEs, we exploit the aggregated inference information to know the influence of other UEs (Section 4.3.1). To further reduce the learning loss, we propose a secondary learning phase based on no-regret dynamics (Section 4.3.2).

4.2.1 Q-Learning Better-Response Process with Inertia

We first review the better-response with inertia process defined in [28]. In each time slot t, with probability η_i (the inertia factor), the UE i selects the same strategy $\mathbf{s}_i^{t-1}$ as in the previous slot. With probability $1-\eta_i$, the UE i selects the current strategy $\mathbf{s}_i^t$ according to a distribution that puts positive probability only on strategies that are better

$$\bar{\mathcal{U}_i} = \mathbb{E}_{\mathbb{H}}[\mathcal{U}_i^t] = \lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + \sum_{i=1}^{N} a_{i,j}^t p_{i,j}^t h_{i,j}^t \right) f_{\mathbf{H}}(\cdot) dh_{1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t - \lambda_2 T^{\text{trans}} \sum_{j=1}^{M} a_{i,j} p_{i,j} \tag{14}$$

$$-\lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + \sum_{m \in \mathcal{N} \setminus \{i\}} a_{m,j} p_{m,j} h_{m,j}^t \right) f_{\mathbf{H}_{-i}}(\cdot) dh_{1,1}^t \cdots dh_{i-1,1}^t \cdots dh_{i-1,|\mathcal{M}|}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t.$$

$$\begin{aligned} \bar{\mathcal{U}_i}(\mathbf{s}_i', \mathbf{s}_{-i}) - \bar{\mathcal{U}_i}(\mathbf{s}_i, \mathbf{s}_{-i}) &= \lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + a_{i,j}' p_{i,j}' h_{i,j}^t + \sum_{m \in \mathcal{N} \setminus \{i\}} a_{m,j} p_{m,j} h_{m,j}^t \right) f_{\mathbf{H}}(\cdot) dh_{1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t \\ &\quad -\lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + a_{i,j} p_{i,j} h_{i,j}^t + \sum_{m \in \mathcal{N} \setminus \{i\}} a_{m,j} p_{m,j} h_{m,j}^t \right) f_{\mathbf{H}}(\cdot) dh_{1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t \\ &\quad -\lambda_2 T^{\text{trans}} \left(\sum_{j=1}^{M} a_{i,j}' p_{i,j}' - \sum_{j=1}^{M} a_{i,j} p_{i,j} \right). \end{aligned} \tag{15}$$

$$\begin{aligned} \psi(\mathbf{s}_i', \mathbf{s}_{-i}) - \psi(\mathbf{s}_i, \mathbf{s}_{-i}) &= \lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + a_{i,j}' p_{i,j}' h_{i,j}^t + \sum_{m \in \mathcal{N} \setminus \{i\}} a_{m,j} p_{m,j} h_{m,j}^t \right) f_{\mathbf{H}}(\cdot) dh_{1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t \\ &\quad -\lambda_1 \kappa T^{\text{trans}} \int_{\mathbb{H}} \sum_{j=1}^{M} W_j \log_2 \left(N_0 + a_{i,j} p_{i,j} h_{i,j}^t + \sum_{m \in \mathcal{N} \setminus \{i\}} a_{m,j} p_{m,j} h_{m,j}^t \right) f_{\mathbf{H}}(\cdot) dh_{1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|}^t \\ &\quad -\lambda_2 T^{\text{trans}} \left(\sum_{j=1}^{M} a_{i,j}' p_{i,j}' - \sum_{j=1}^{M} a_{i,j} p_{i,j} + \sum_{m \in \mathcal{N} \setminus \{i\}} \left(\underbrace{\sum_{j=1}^{M} a_{m,j} p_{m,j} - \sum_{j=1}^{M} a_{m,j} p_{m,j}}_{=0} \right) \right). \end{aligned} \tag{16}$$

Algorithm 2 Q-Learning Better-Response Process with Inertia

```
1: for $t = 1, \cdots, T$ do
2:     At $t = 1$, all UEs start with a random strategy.
3:     At $t > 1$, each UE selects its new strategy as follows.
4:     • With probability $\epsilon_i^t$, $\mathbf{s}_i^t$ is chosen with probability $|\frac{1}{\mathcal{S}_i}|$.
5:     • With probability $1 - \epsilon_i^t$,
6:         ○ with probability $\eta_i$, $\mathbf{s}_i^t \leftarrow \mathbf{s}_i^{t-1}$,
7:         ○ with probability $1 - \eta_i$, choose a strategy from $\mathcal{B}_i^{t-1}(\mathbf{s}_{-i})$ randomly. If $\mathcal{B}_i^{t-1}(\mathbf{s}_{-i}) = \varnothing$, $\mathbf{s}_i^t \leftarrow \mathbf{s}_i^{t-1}$.
8: end for
```

than the previous strategy $\mathbf{s}_i^{t-1}$. Better-response with inertia dynamics is proven to converge to the NE of potential games [28]. Then, we introduce the notation $\mathcal{B}_i^t$ as the better-response set for UE i at t given other UEs' strategies $\mathbf{s}_{-i}$,

$$\mathcal{B}_i^t(\mathbf{s}_{-i}) = \{\mathbf{s}_i' | \mathbf{s}_i' \in \mathcal{S}_i, \bar{\mathcal{U}}_i^t(\mathbf{s}_i', \mathbf{s}_{-i}) > \bar{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})\}. \tag{18}$$

We consider a stochastic better-response with inertia dynamics by adding an exploration process as shown in Algorithm 2, where $\epsilon_i^t = \epsilon_i = c_\epsilon t^{-\frac{1}{|\mathcal{N}|}}$ $(c_\epsilon > 0)$ is the exploration factor.

The stochastic better-response with inertia dynamics can converge to the NE of $\mathcal{G}'$ [28]. It can only apply to the case where both the average utility $\bar{\mathcal{U}}_i^t$ and joint strategies of other UEs $\mathbf{s}_{-i}$ are known to UE i. However, neither the average utility nor joint strategies are known in our scenario when we construct the better-response set. To address the unknown CSI, we resort to Q-learning following [28]. The action is the strategy $\mathbf{s}_i$ of each UE i, the reward function is the utility function $\mathcal{U}_i^t$. The state refers to the time-varying wireless channel state. Different from classic Q-learning, our state is unavailable to UEs and the state space will be huge when we model the continuously changing wireless channel state. So we introduce single-state Q-learning [37] where the agent is able to select the action based entirely on its utility update function $\widetilde{\mathcal{U}} : \mathbb{S} \to \mathbb{R}$ [28]

$$\widetilde{\mathcal{U}}_i^t(\hat{\mathbf{s}}) = \widetilde{\mathcal{U}}_i^{t-1}(\hat{\mathbf{s}}) + \nu^t \mathbb{I}_{\mathbf{s}^t = \hat{\mathbf{s}}}(\mathcal{U}_i^t(\mathbf{s}^t) - \widetilde{\mathcal{U}}_i^{t-1}(\hat{\mathbf{s}})), \tag{19}$$

where $\nu^t = (C_\nu + \sharp^t(\hat{\mathbf{s}}))^{-\rho_\nu}$ is the learning parameter, $C_\nu > 0$, $\rho_\nu \in (0, \frac{1}{2}]$, $\hat{\mathbf{s}}$ is an arbitrary joint strategy, $\sharp^t(\hat{\mathbf{s}})$ is the number of times the strategy $\hat{\mathbf{s}}$ has been chosen until time t, and $\mathbb{I}$ is the indicator function. We replace $\bar{\mathcal{U}}_i^t$ with $\widetilde{\mathcal{U}}_i^t$ to construct the better-response set $\mathcal{B}_i^t(\mathbf{s}_{-i})$. The Q-learning better-response process with inertia algorithm is summarized in Algorithm 2.

4.3 Algorithm Enhancement

The Q-learning better-response process with inertia algorithm only addresses the unknown CSI challenge. We seek to enhance the algorithm to address the unavailable joint strategies challenge and further improve the learning performance.

4.3.1 Exploiting Aggregated Information

Still, UEs are aware of the aggregated information $\mathbf{I}_{-i}$ which contains the information of $\mathbf{s}_{-i}$. We can update $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ instead of $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ in (19) according to Claim 1, where $\mathbf{I}_{-i} = (I_{-i,1}, I_{-i,2}, \cdots, I_{-i,M})$ is the vector of aggregated information from each edge node.

Claim 1. *If the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ can achieve the NE $\mathbf{s}^*$, the same holds for the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$.*

Proof. We prove Claim 1 by analyzing the correlation of the aggregated information and strategies of other UEs in two cases: time-invariant CSI and time-variant CSI. We start from the time-invariant CSI. The aggregated information is, $I_{-i,j}(\mathbf{s}) = \frac{h_{i,j}}{N_0 + \sum\limits_{m\in\mathcal{N}\setminus\{i\}} a_{m,j}p_{m,j}h_{m,j}}$. Keep $\mathbf{s}_i$ unchanged, the same $\mathbf{s}_{-i}$ results in the same $I_{-i,j}(\mathbf{s})$ because $h_{i,j}$ is constant over time slots. The mapping relationship between strategies $\mathbf{s}_{-i}$ and the aggregated information $\mathbf{I}_{-i}$ can be divided into two types: one-to-one mapping and many-to-one mapping. In the one-to-one mapping case, it is obvious that the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ is equivalent to the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$. In the many-to-one mapping case, it is probable that different strategies of other UEs, e.g., $\mathbf{s}_{-i}$ and $\mathbf{s}'_{-i}$ have the same $\mathbf{I}_{-i}$ and $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$. We argue that this many-to-one mapping has no influence on convergence to the NE as the strategy updating process towards NE is based on UEs' utility (which is directly determined by $\mathbf{I}_{-i}$). The same utility will lead to the same updating process. Therefore, the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ can also converge to the NE in time-invariant CSI cases if the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ converges.

Then, we extend to the time-variant CSI case. There are three types of mapping relationship between $\mathbf{s}_{-i}$ and $\mathbf{I}_{-i}$ in this case: one-to-one mapping, many-to-one mapping, and one-to-many mapping. The first two have been analyzed in the time-invariant cases and we focus on the one-to-many mapping where the same $\mathbf{s}_{-i}$ may result in different $\mathbf{I}_{-i}$ and $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ due to CSI randomness across time slots. Each UE i may respond differently based on different $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ for the same $\mathbf{s}_{-i}$. Note that different response for the same $\mathbf{s}_{-i}$ can occur due to CSI randomness even when each UE i updates based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$. However, the randomness of different $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ will be averaged to the mean of $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ over time and the one-to-many mapping will vanish, i.e., $\lim\limits_{t\to\infty} \widetilde{\mathcal{U}}_i^t(\mathbf{s}'_i, \mathbf{s}_{-i}) - \widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i}) = \lim\limits_{t\to\infty} \widetilde{\mathcal{U}}_i^t(\mathbf{s}'_i, \mathbf{I}_{-i}) - \widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i}) = \lim\limits_{t\to\infty} \widetilde{\mathcal{U}}_i^t(\mathbf{s}'_i, \mathbf{I}'_{-i}) - \widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}'_{-i})$. Therefore, if the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-fi})$ leads to the NE, the same holds for updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$. □

The better set of each time slot t can be modified as,

$$\widetilde{\mathcal{B}}_i^t(\mathbf{s}_{-i}) = \{\mathbf{s}'_i | \mathbf{s}'_i \in \mathcal{S}_i, \widetilde{\mathcal{U}}_i^t(\mathbf{s}'_i, \mathbf{I}_{-i}) > \widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})\}. \quad (20)$$

The detail is illustrated in the primary learning of Algorithm 3 (Lines 4-10). The primary learning converges to the NE of the game $\mathcal{G}$ as shown in proof of Theorem 1. Next, we will design the secondary learning to assist the primary learning.

4.3.2 No-Regret Dynamics

In a repeated game, a player has regret for a strategy if there exists another strategy whose payoff is greater [36]. The average regret of the strategy $\{\hat{\mathbf{s}}_i^t\}_{t=1}^T$ with respect to strategy $\{\mathbf{s}_i^t\}_{t=1}^T$ is defined as

$$Reg_i^T(\{\hat{\mathbf{s}}_i\}_{t=1}^T) = \frac{1}{T}\sum_{t=1}^{T}(\mathcal{U}_i^t(\hat{\mathbf{s}}_i^t, \mathbf{s}_{-i}^t) - \mathcal{U}_i^t(\mathbf{s}_i^t, \mathbf{s}_{-i}^t)). \quad (21)$$

The main idea of no-regret dynamics is that each player should adapt his strategy such that his regret grows sublinearly as a function of the horizon T, thereby yielding zero average regret asymptotically. No-regret dynamics guarantees that the joint strategy will converge to points of no-regret. These points can be referred to as coarse correlated equilibrium in potential games [29]. Note that the NE is coarse correlated equilibrium. Hence, no-regret algorithms can effectively improve the convergence rate and bound the learning loss of primary learning.

In this section, we adopt the idea of regret matching with fading memory [36] into the better-response with inertia dynamics. Thus, each UE can choose a strategy from the better-response set based on the discounted average regret probability of strategies. First, each UE needs to compute the regret of each unplayed strategy corresponding to the strategies of other UEs. With fading memory, each UE exponentially discounts the influence of past regret, i.e.,

$$Reg_i^{t+1}(\hat{\mathbf{s}}_i) = (1-\alpha)Reg_i^t(\hat{\mathbf{s}}_i) + \alpha(\mathcal{U}_i^t(\hat{\mathbf{s}}_i, \mathbf{I}_{-i}) - \mathcal{U}_i^t(\mathbf{s}_i, \mathbf{I}_{-i}), \quad (22)$$

for all $\hat{\mathbf{s}}_i \in \mathcal{S}_i$, where $\alpha \in (0, 1]$. $(1-\alpha)$ is the discount factor and $Reg_i^0(\hat{\mathbf{s}}_i) = 0$. The aggregated information $\mathbf{I}_{-i}$ is equivalent to $\mathbf{s}_{-i}$ as shown in Claim 1. Then, each UE computes the discounted average regret probability as

$$\Pr_i(Reg_i(\hat{\mathbf{s}}_i)) = \frac{[Reg_i(\hat{\mathbf{s}}_i)]^+}{\sum\limits_{\mathbf{s}_i\in\mathcal{S}_i}[Reg_i(\mathbf{s}_i]^+}. \quad (23)$$

Recall that the stochastic better-response with inertia algorithm chooses a strategy randomly from the better-response set (Line 6 of Algorithm 2). Here, the secondary learning assists the primary learning by choosing a strategy from better-response set $\widetilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i})$ according to regret based probability $\Pr_i(Reg_i^{t-1}(\widetilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i})))$ (Line 15 in Algorithm 3). We describe the algorithm in Algorithm 3. To make the algorithm structure more clear, we provide an algorithm framework illustration in Fig. 1.

4.4 Algorithm Performance Analysis

Theorem 2 shows the convergence guarantee and the upper-bounded learning regret of Algorithm 3.

Theorem 2. *The strategy profiles $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ generated by Algorithm 3 can converge to the NE with no regret.*

Proof. **Convergence**. Algorithm 3 is a variant of Q-learning better-response process with inertia algorithm [28]. We first review the convergence proof of Q-learning better-response process with inertia in **Step 1** and then prove the the convergence of Algorithm 3 in **Step 2**. Suppose there are two strategy profiles $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ and $\{\dot{\mathbf{s}}_i^1, \cdots, \dot{\mathbf{s}}_i^T\}$ generated by Algorithm 3 and Q-learning better-response process with inertia, respectively.

Step 1. The convergence of the Q-learning better-response process with inertia follows from Lemma 5.5 in [28]. We give the proof sketch as follows. Consider a Markov chain generated on state space O by the better-response process with inertia [28]. Let $p_i(\mathbf{s}_i|o)$ be the better-response distribution used by UE i with $p_i(\mathbf{s}_i|o) > 0$ only if $\mathbf{s}_i$ is a better reply to o for i. Let $\mathcal{L} \subseteq \mathcal{N}$ be a set of UEs having inertia and keeping their strategies unchanged at the current time slot. The Markov process transition function for a finite memory better-response process with inertia is

$$P_{oo'} = \prod_{i\in\mathcal{N}}(1-\eta_i)p_i(\mathbf{s}_i|o) + \sum_{\mathcal{L}\subseteq\mathcal{N}}(\prod_{i\in\mathcal{L}}\eta_i)(\prod_{i\notin\mathcal{L}}(1-\eta_i))U_{oo'}^{\mathcal{L}}, \quad (24)$$

Algorithm 3 No-Regret Better-Response Based Edge Node Grouping

```
Input: $\mathcal{M}, \mathcal{N}, \rho, \alpha_i, \forall i \in \mathcal{N}$.
1: Initialization: $\mathbf{s}_i^0$ is selected with probability $\frac{1}{|\mathcal{S}_i|}$ and UEs initialize $\widetilde{\mathcal{U}}_i^0(\mathbf{s}^0) = 0$, $\tilde{\mathcal{B}}_i^0(\mathbf{s}_{-i}) = \varnothing$ and $Reg_i^0 = \mathbf{0}$.
2: for $t = 1, \cdots, T$ do
3:   for $i = 1, \cdots, N$ do
4:     Primary learning.
5:     • With probability $\epsilon_i^t$, $\mathbf{s}_i^t$ is chosen with probability $|\frac{1}{\mathcal{S}_i}|$.
6:     • With probability $1 - \epsilon_i^t$,
7:       ∘ with probability $\eta_i$, $\mathbf{s}_i^t \leftarrow \mathbf{s}_i^{t-1}$,
8:       ∘ with probability $1 - \eta_i$, choose a strategy from $\tilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i})$ according to $\Pr_i(Reg_i^{t-1}(\tilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i})))$. If $\tilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i}) = \varnothing$, $\mathbf{s}_i^t \leftarrow \mathbf{s}_i^{t-1}$.
9:     Observe $F_i^t(\mathbf{s}^t)$ in the "Result Returning" duration.
10:    Update $\widetilde{\mathcal{U}}_i^t(\mathbf{s}^t)$ according to (19).
11:    Secondary learning.
12:    Calculate the aggregated information $I_{-i,j}^t$.
13:    Calculate the utility $\mathcal{U}_i^t(\hat{\mathbf{s}}_i^t, \mathbf{I}_{-i}^t)$ of each strategy $\hat{\mathbf{s}}_i^t$ according to (6).
14:    Update the regret $Reg_i^t(\hat{\mathbf{s}}_i^t)$ of each strategy according to (22).
15:    Calculate the regret based probability.
16:  end for
17: end for
```

where

$$U_{oo'}^{\mathcal{L}} = \begin{cases} \prod_{i\in\mathcal{L}} \mathbb{I}\{\mathbf{s}_i' = \mathbf{s}_i\} \prod_{i\notin\mathcal{L}} p_i(\mathbf{s}_i'|o) & \text{Condition 1} \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

where Condition 1 is that if o is a successor to o' and $\mathbf{s}_i$ (resp. $\mathbf{s}_i'$) is the i_{th} rightmost entry of o (resp. o').

The stochastic better-response process with inertia is a perturbed process of the better-response process with inertia. At each time slot, each UE uses the better-response process with inertia rule with probability $1-\epsilon^t$ or uniformly samples from $\mathcal{S}_i$ with probability ϵ^t. The transition matrix of stochastic better-response process with inertia at time slot t is P^{ϵ^t}. For $\epsilon^t = \epsilon$ fixed, the transition probability matrix is

$$P_{oo'}^{\epsilon} = (1-\epsilon)^{|\mathcal{N}|} P_{oo'} + \sum_{\mathcal{L}'\subseteq\mathcal{N}, \mathcal{L}\neq\emptyset} \epsilon^{|\mathcal{L}|}(1-\epsilon)^{(|\mathcal{N}|-|\mathcal{L}|)} U_{oo'}^{\mathcal{L}'}, \tag{26}$$

where

$$U_{oo'}^{\mathcal{L}'} = \begin{cases} \prod_{i\in\mathcal{L}'} \frac{1}{|\mathcal{S}_i|} \mathbb{I}\{\mathbf{s}_i' = \mathbf{s}_i\} \prod_{i\notin\mathcal{L}} p_i(\mathbf{s}_i'|o) & \text{Condition 1} \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

If $\epsilon^t = \epsilon$ for all t then we have a stationary, irreducible, and aperiodic Markov chain on a finite state space, which is both weakly and strongly ergodic. The stochastic better-response process with inertia with varying ϵ^t converges to the NE following Lemma 5.6 in [28]. The proof is divided into three steps, first showing that the process is weakly ergodic, second, that it is strongly ergodic, and third, that the distribution of historical strategies converges to a unique steady-state distribution μ^{ϵ^t} with $\lim_{t\to\infty, \epsilon^t\to 0} \mu^{\epsilon^t} = \mu^*$. Finally, the

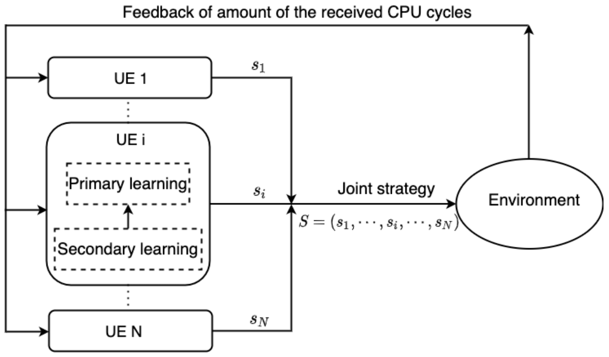


Fig. 1: Framework of Algorithm 3.

convergence of the Q-learning better-response process with inertia follows from Lemma 5.5 in [28].

Step 2. Algorithm 3 is a variant of Q-learning better-response process with inertia, which has two main modifications to fit our scenario. First, we enhance better-response distribution by no-regret dynamics. Second, we replace $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ with $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ in the Q-learning updating process. We prove that Algorithm 3 has the same convergence performance with the Q-learning better-response process with inertia. We begin with the first modification. The Q-learning better-response process with inertia algorithm chooses a strategy from a better-response set randomly while Algorithm 3 chooses a strategy from a better-response set according to regret based probability distribution $\Pr_i(Reg_i^{t-1}(\tilde{\mathcal{B}}_i^{t-1}(\mathbf{s}_{-i})))$. This modification leads to a different better-response distribution $p_i(\mathbf{s}_i|o)$ in (**??**) but has no influence on the ergodicity of the Markov chain. We can analyze the regret-based variant of the stochastic better-response process with inertia in an identical way to the earlier treatment of the stochastic better-response process with inertia. This gives the perturbed transition probability for the regret-based variant of the stochastic better-response process with inertia, which looks identical to (26) but has different values $P_{oo'}$ and $p_i(\mathbf{s}_i|o)$. So the regret-based variant of the stochastic better-response process with inertia converges to the NE. And the convergence of the Q-learning variant also follows from Lemma 5.5 in [28]. Therefore, the first modification has no influence on the convergence performance. Following Claim 1, the second modification has no influence on the convergence performance, either. Therefore, the sequence of profile s^t is a non-stationary ergodic Markov process and converges to the NE.

No Regret. In Algorithm 3, the regret matching method is performed on the better-response set, where the strategies are better in terms of maximizing the utility or minimizing the regret. We can infer that the average regret of strategies generated by Algorithm 3 is less than the regret matching method. From the Blackwells Approachability Theorem in [38], the average regret of regret matching method is of the order of $\frac{1}{\sqrt{t}}$, i.e., $\lim_{T\to\infty} \frac{Reg_i^T(\hat{\mathbf{s}}_i^t)}{T} = 0$. The regret matching method is no-regret. Therefore, Algorithm 3 is no-regret. $\square$

5 Discussion

5.1 Delayed Feedback

In this section, we consider the scenario where computation capacities of edge nodes are fixed. The sum of the received CPU cycles cannot exceed the maximum capacity $F_{j,\max}$,

$$\sum_{i=1}^{N} a_{i,j}^{t} F_{i,j}^{t}(\mathbf{s}^t) \leq F_{j,\max}, \forall j \in \mathcal{M}. \tag{28}$$

Suppose that UE i obtains $\hat{F}_{i,j}^{t}(\mathbf{s}^t)$ CPU cycles in the time slot t from edge node j. The computation time is

$$T_{i,j}^{\text{com}} = \frac{\kappa T^{\text{trans}} R_{i,j}^{t}(\mathbf{s}^t)}{\hat{F}_{i,j}^{t}(\mathbf{s}^t)}, \tag{29}$$

which can be influenced by both the joint strategies of other UEs and the computation resource in edge nodes. When $T_{i,j}^{\text{com}} < T^{\text{dur}}$, the result and feedback are returned in the current slot. Here, we consider the case of $T_{i,j}^{\text{com}} \geq T^{\text{dur}}$. The results and feedback of slot t are returned to UE i in slot $t + \lceil \frac{T_{i,j}^{\text{com}}}{T^{\text{dur}}} \rceil$. Note that $T_{i,j}^{\text{com}}$ is unknown to UE i, and the delays may change the order of feedback, with the feedback of a latter strategy returning before the feedback of an earlier one. UEs need to learn with absent and disordered feedback.

We propose Algorithm 4 by integrating feedback queues [39] in between the primary learning phase and the secondary learning phase to enable Algorithm 4 to adapt the delayed feedback. In the result returning phase, each UE collects feedback information of prior time slots and stores it in separate queues for each strategy (Line 5 in Algorithm 4). If the feedback queue for the current strategy is not empty, $\widetilde{\mathcal{U}}_i^t(\mathbf{s}^t)$ is updated (Line 10 in Algorithm 4) and secondary learning is performed. When no feedback is available (Line 13 in Algorithm 4), the algorithm keeps the same strategy with the last time slot. Thus, Algorithm 3 runs in a simulated non-delayed environment. We describe the algorithm in Algorithm 4 and illustrate the algorithm framework in Fig. 2. Theorem 3 shows the theoretical convergence guarantee and upper-bounded learning regret of Algorithm 4.

Theorem 3. *Suppose $T_{i,j}^{\text{com}} \leq \tau_{\max} (\forall i, j, t)$, where $\tau_{\max} \ll T$, the strategy profiles $\{\check{\mathbf{s}}_i^t\}_{t=1}^{T}$ generated by Algorithm 4 converge to the NE and have no regret.*

Proof. **Convergence**. Suppose there are two strategy profiles $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ and $\{\check{\mathbf{s}}_i^1, \cdots, \check{\mathbf{s}}_i^T\}$ generated by Algorithm 3 and Algorithm 4 respectively. The key idea is to prove $\{\check{\mathbf{s}}_i^1, \cdots, \check{\mathbf{s}}_i^T\}$ is an extension of $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ with arbitrary element $\mathbf{s}_i^t$ repeats itself several times. Thus, $\{\check{\mathbf{s}}_i^1, \cdots, \check{\mathbf{s}}_i^T\}$ has the same convergence performance with $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$. From Theorem 2, $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ converges to the NE. We consider an extreme situation where $T_{i,j}^{\text{com}} = \tau_{\max}$. When the feedback of the played strategy is available in the queue, Algorithm 4 performs the same steps as Algorithm 3. When the feedback is not available, the played strategy repeats itself at most $\tau_{\max}$ times until a feedback returns. Thus, the strategy profiles $\{\check{\mathbf{s}}_i^1, \cdots, \check{\mathbf{s}}_i^T\}$ can be seen as an extension of $\{\mathbf{s}_i^1, \cdots, \mathbf{s}_i^T\}$ where each strategy repeats itself several (at most $\tau_{\max}$) times. With $\tau_{\max} \ll T$, $\{\check{\mathbf{s}}_i^1, \cdots, \check{\mathbf{s}}_i^T\}$ also converges to the NE.

No Regret. To relate the regret of Algorithm 3 and Algorithm 4, we first model the upper-bound of missing

Algorithm 4 Feedback Queues Based Edge Node Grouping

```
Input: $\mathcal{M}, \mathcal{N}, \rho, \alpha_i, \forall i \in \mathcal{N}$.
 1: Initialization. $\mathbf{s}_i^0$ is selected with probability $\frac{1}{|\mathcal{S}_i|}$. UEs
    initialize $\widetilde{\mathcal{U}}_i^0(\mathbf{s}^0) = 0$, $\widetilde{\mathcal{B}}_i^0(\mathbf{s}_{-i}) = \varnothing$, $Reg_i^0 = \mathbf{0}$ and create
    an empty FIFO buffer $Q[\mathbf{s}_i]$ for each $\mathbf{s}_i \in \mathcal{S}_i$.
 2: for time slot $t = 1, 2, \cdots, T$ do
 3:   for $i = 1, \cdots, N$ do
 4:     Primary learning in Algorithm 3.
 5:     Queueing.
 6:     for each delayed feedback do
 7:       Add the feedback to the buffer $Q(\mathbf{s}_i)$.
 8:     end for
 9:     Use $J$ to index the current strategy $J \leftarrow \mathbf{s}_i^t$ .
10:     if $Q[J]$ is not empty then
11:       Update $\widetilde{\mathcal{U}}_i^t(J)$ using $Q[J]$ according to (19).
12:       Secondary learning in Algorithm 3.
13:     else
14:       $\mathbf{s}_i^t \leftarrow \mathbf{s}_i^{t-1}$.
15:       Go to Line 9.
16:     end if
17:   end for
18: end for
```

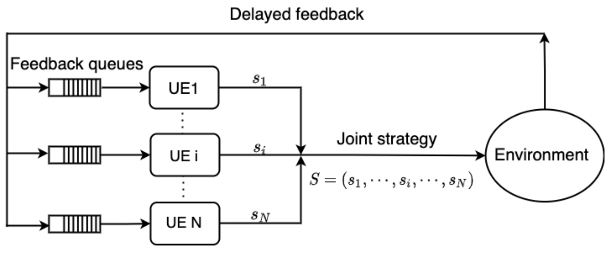


Fig. 2: Framework of Algorithm 4.

feedback times. Then we relate the number of times that the two algorithms run, based on which we relate the regret of the two algorithms. We make some definitions. Let $Y_{\mathbf{s}_i}^t$ denote the number of times $\mathbf{s}_i$ is chosen by the end of time slot t. Let $Z_{\mathbf{s}_i}^t$ denote the number of feedback for $\mathbf{s}_i$ that are received by the end of time slot t. $G_{\mathbf{s}_i}^t = Y_{\mathbf{s}_i}^t - Z_{\mathbf{s}_i}^t$ is the number of missing feedback for $\mathbf{s}_i$ at time slot t. $G_i^{t*} = \max_{\mathbf{s}_i} G_{\mathbf{s}_i}^t$. $\bar{\mathcal{U}}_{\mathbf{s}_i}$ is the expected utility of $\mathbf{s}_i$, $\mathcal{U}_i^*$ is the maximal utility of all strategies, $\xi_{\mathbf{s}_i} = \mathcal{U}_i^* - \bar{\mathcal{U}}_{\mathbf{s}_i}$ is the regret of strategy $\mathbf{s}_i$. The expected regret of UE i is $\mathbb{E}[Reg_i^T] = \sum_{t=1}^{T} (\mathcal{U}_i^* - \bar{\mathcal{U}}_{\mathbf{s}_i^t}) = \sum_{\mathcal{S}_i} \xi_{\mathbf{s}_i} \mathbb{E}[Y_{\mathbf{s}_i}^T]$. Furthermore, let $Y_{\mathbf{s}_i}^{'t'}$ be the number of times that Algorithm 3 has chosen $\mathbf{s}_i$ while being queried t' times. t' is the number of time slots the Algorithm 3 performs in t slots of the delayed problem. Assume that Algorithm 4 runs multiple slots so that Algorithm 3 is queried for t times. Then, since $t' \leq t$, the number of times $\mathbf{s}_i$ is chosen by Algorithm 3, namely $Y_{\mathbf{s}_i}^{'t'}$ can only increase, $Y_{\mathbf{s}_i}^{'t'} \leq Y_{\mathbf{s}_i}^{'t}$. Combined with the Lemma 5 of [39], $0 \leq Y_{\mathbf{s}_i}^t - Y_{\mathbf{s}_i}^{'t'} \leq G_i^{t*}$, we have $\mathbb{E}[Y_{\mathbf{s}_i}^t] \leq \mathbb{E}[Y_{\mathbf{s}_i}^{'t}] + \mathbb{E}[G_i^{t*}]$. The average regret over all T time slots is

$$\sum_{\mathcal{S}_i} \xi_{\mathbf{s}_i} \mathbb{E}[Y_{\mathbf{s}_i}^T] \leq \sum_{\mathcal{S}_i} \xi_{\mathbf{s}_i} \mathbb{E}[Y_{\mathbf{s}_i}^{'T}] + \sum_{\mathcal{S}_i} \xi_{\mathbf{s}_i} \mathbb{E}[G_i^{T*}]. \tag{30}$$

$$R_i^t(\mathbf{s}^t) = \sum_{j=1}^{M}\sum_{k=1}^{K} W_{j,k}\log_2\left(1+\frac{a_{i,j,k}^t p_{i,j,k}^t h_{i,j,k}^t}{N_0+\sum\limits_{m\in\mathcal{N}\setminus\{i\}} a_{m,j,k}^t p_{m,j,k}^t h_{m,j,k}^t}\right). \tag{34}$$

From Lemma 4 of [39], the delayed feedback sequence of each strategy has the same distribution as the sequence of non-delayed feedback sequence. The Algorithm 3 has worked on the first $Y_{\mathbf{s}_i}^{'t}$ of the feedback for each $\mathbf{s}_i$, and has therefore operated for t times in a simulated environment with the same feedback distributions, but without delay. Hence, the first summation in the right-hand side of (30) is $\mathbb{E}[Reg_i^T]$ the expected regret of the Algorithm 3. As $T_{i,j}^{\text{com}} \le \tau_{\max}$, $\mathbb{E}[G_i^{t*}] \le \tau_{\max}$. The average regret of Algorithm 4 $\widetilde{Reg}_i^T$ is upper-bounded by $\mathbb{E}[\widetilde{Reg}_i^T] \le \mathbb{E}[Reg_i^T] + \mathcal{O}(\tau_{\max})$. Based on Theorem 2, $\lim\limits_{T\to\infty} \frac{\mathbb{E}[\widetilde{Reg}_i^T]}{T} = \lim\limits_{T\to\infty} (\frac{\mathbb{E}[Reg_i^T]}{T} + \frac{\mathcal{O}(\tau_{\max})}{T}) = 0$. Therefore, Algorithm 4 is also no-regret. □

5.2 Edge Nodes with Multiple Frequency Bands

In previous sections, we assume that UEs offloading the task to the same edge node share one frequency band. We will extend to the scenario where each edge node has multiple frequency bands denoted as $\mathcal{K} = \{1, \cdots, K\}$. The strategy of UE i is modified as $\mathbf{s}_i^t = \{a_{i,j,k}^t, p_{i,j,k}^t\}$. The constraint on UE-edge node association is modified as,

$$\sum_{j=1}^{M}\sum_{k=1}^{K} a_{i,j,k}^t = L, \forall i \in \mathcal{N}, 1 \le t \le T, \tag{31}$$

$$\sum_{k=1}^{K} a_{i,j,k}^t = 1, \forall i \in \mathcal{N}, \forall j \in \mathcal{M}, 1 \le t \le T, \tag{32}$$

which means each UE occupies one band when connecting to each edge node. The constraint on transmission power allocation is modified as,

$$\sum_{j=1}^{M}\sum_{k=1}^{K} p_{i,j,k}^t \le P_{i,\max}, \forall i \in \mathcal{N}, 1 \le t \le T. \tag{33}$$

We focus on exploring the wireless interference that occurs when multiple UEs connect to the same edge node on the same band. We compute the total transmission rate of UE i as (34), where $W_{j,k}$ is the channel bandwidth, $h_{i,j,k}^t$ is the channel gain between UE i and edge node j on frequency band k. Uplink inter-UE interference occurs when multiple UEs simultaneously transmit to the same edge node over the same band. The transmission energy consumption of UE i is

$$E_i^t(\mathbf{s}_i^t) = T^{\text{trans}} \sum_{j=1}^{M}\sum_{k=1}^{K} a_{i,j,k}^t p_{i,j,k}^t, \tag{35}$$

Then, we will analyze the aggregated information pattern in this extended scenario. For frequency band k of edge node j with $a_{i,j,k}^t = 1$, the aggregated information of other UE strategies $I_{-i,j,k}$ in edge node j on frequency band k at time slot t can be calculated as

$$I_{-i,j,k}^t = \frac{2^{\frac{R_{i,j,k}^t}{W_{j,k}}} - 1}{a_{i,j,k}^t p_{i,j,k}^t}, \tag{36}$$

where $R_{i,j,k}^t$ denotes the data transmission rate between UE i and edge node j on frequency band k, and can be calculated from the feedback following (36). For frequency band k of edge node j with $a_{i,j,k}^t = 0$, UE i transmits pilot signals with low power and the aggregated information can also be obtained from (36). Consequently, the aggregated information of other UE strategies can be calculated. Similarly, we can apply computation models (Sections 3.3 and 5.1) and utility function (Section 3.4) to compute the utility. We can also model the edge node grouping problem as a repeated game. For such repeated game under the multiple frequency bands model, we show that it exhibits the same structural property as the single frequency band model. Both Theorem 1 and Claim 1 hold in this multiple frequency bands scenario.

Proposition 1. *The stage game $\mathcal{G}' = \{N, \{\mathbf{s}_i\}_{i\in\mathcal{N}}, \{\bar{\mathcal{U}}_i\}_{i\in\mathcal{N}}\}$ is an exact potential game with the potential function in (37), where $\mathbf{H} = \{h_{i,j,k}^t\}$ is the matrix of the channel gain. $f_{\mathbf{H}}$ is the joint distribution of $\mathbf{H}$.*

The proof is similar to Theorem 1. In this scenario, we can also update $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$ instead of $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ according to Claim 2, where $\mathbf{I}_{-i} = \{I_{-i,j,k}\}$ is the matrix of aggregated information.

Claim 2. *In the multiple frequency bands scenario, if the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{s}_{-i})$ can achieve the NE $\mathbf{s}^*$, the same holds for the updating based on $\widetilde{\mathcal{U}}_i^t(\mathbf{s}_i, \mathbf{I}_{-i})$.*

The proof is similar to Claim 1. Based on Proposition 1 and Claim 2, Algorithms 3 and 4 can extend to the scenario where each edge node has multiple frequency bands.

5.3 Limitations of Our Algorithms

Reinforcement learning has been applied widely in various scenarios [40–42]. In this work, we design multi-agent Q-learning algorithms where UEs select a joint action and each receives an individual reward. In the context of mobile communication networks, each UE only competes with several neighboring UEs and the size of the candidate edge node set is also limited because only edge nodes with relatively high received signal strength would be chosen to ensure good offloading performance. The proposed algorithms can scale well with network size due to limited action space. However, they may face space explosion problems in other large-scale games. The size of the learning problem grows exponentially with the number of agents. This causes slow convergence and heavy memory overhead, thereby reducing the applicability of our algorithms in other large-scale games.

6 Evaluation

We evaluate Algorithm 3 (labeled as Alg.3) and Algorithm 4 (labeled as Alg.4) by answering 7 research questions (RQs) with 3 metrics, namely, average utility, average received

$$\psi(\mathbf{s}_i,\mathbf{s}_{-i}) = T^{\text{trans}}\left(\lambda_1 \int_{\mathbb{H}} \sum_{j=1}^{M}\sum_{k=1}^{K} W_{j,k}\log_2\left(N_0+\sum_{i=1}^{N} a_{i,j,k}p_{i,j,k}h_{i,j,k}^t\right) f_{\mathbf{H}} dh_{1,1,1}^t \cdots dh_{|\mathcal{N}|,|\mathcal{M}|,|\mathcal{K}|}^t - \lambda_2 \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{K} a_{i,j,k}p_{i,j,k}\right). \quad (37)$$

TABLE 2: Simulation Parameters.

Parameters	Value
Task transmission time T^{trans}	300 ms
Channel bandwidth W_j	0.1 MHz
Transmit power budget $P_{i,\max}$	0.1W
Computation to data ratio κ	1900 cycle/bit
Background noise power N_0	10^{-13} W
Weight factor λ_1	2×10^{-6}
Weight factor λ_2	600
Inertia Parameter η_i	$1/N$
Parameter C_ν	0.01
Parameter c_ϵ	1
Parameter ρ_ν	0.9
Parameter α	0.91

CPU cycle numbers, and average energy consumption. And we share our simulation codes [1] for researchers who are interested in our work.

6.1 Simulation Setup

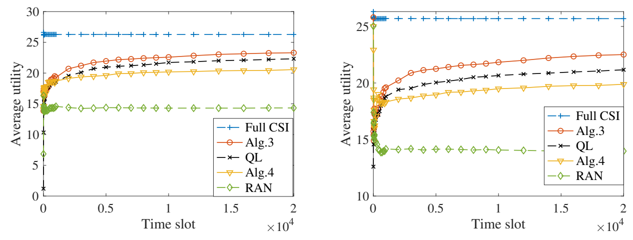
Simulation Parameters. We conduct extensive simulations with various system settings on a MATLAB2016b-based simulator using a PC with Intel Core i5-7200U CPU @ 2.5 GHz processor. We assume that the edge network is deployed in a commercial complex served by a set of edge nodes whose locations are generated by a homogeneous Poisson point process. The UE locations are also generated by the homogeneous Poisson point process. The default channel fast fading between each pair of edge node and UE follows Rayleigh distribution. All simulation results are obtained by averaging over 2×10^4 time slots. In the delayed-feedback case, the feedback delay (measured by the number of time slots) follows exponential distribution $\exp(\mu)$ with $\mu=3$. The feedback delay is $\min(\exp(\mu),\tau_{\max})$. The main parameters follow [27] as shown in TABLE 2.

Baseline Approaches. We compare our algorithms with three baselines as follows.

1) *Oracle benchmark (labeled as Full CSI)*. Each UE knows the full CSI as well as strategies of other UEs and derives its strategy by following Algorithm 1. In each time slot, the algorithm randomly chooses a UE to perform the best-response to other UEs iteratively until converges to the NE. The Oracle benchmark faces no challenge from system uncertainty that we focus on, so it provides a performance upper bound for our algorithms.

2) *Q-Learning (labeled as QL)*. Each UE knows strategies of other UEs with unknown CSI and derives its strategy by following the idea of Algorithm 2 in [27] where each UE learns the offloading strategy by updating the Q-function. Q-Learning is the state-of-art online learning method that is highly related to ours. The superiority of our algorithms can be highlighted by comparing with the Q-Learning method

1. http://sguangwang.com/resources.php



(a) Rician Channel. (b) Rayleigh Channel.

Fig. 3: Performance under different channel models.

[27] in face of system uncertainty from both wireless environment and other users.

3) *Random assignment (labeled as RAN)*. Each UE randomly chooses the edge node grouping strategy in each time slot. The Random assignment method is the simplest and most time-efficient way to make edge node grouping decisions and provides a performance lower bound for our algorithm.

6.2 RQ1: How does channel condition affect our algorithms?

Motivation. Since channel state dynamics can influence learning efficiency. We aim to evaluate the scalability of our algorithms under different channel conditions.

Approach. Rician and Rayleigh channel models are two widely used channel models in wireless communications. The Rician channel assumes that the transmission paths from the transmitter to the receiver are comprised of the dominant line-of-sight path and other scattering paths, whereas the Rayleigh channel consists of scattering channels from the transmitter to the receiver. We test how our algorithms adapt to the two channel models. We plot utility variation with time under Rician distribution and Rayleigh distribution in Fig. 3(a) and Fig. 3(b) respectively.

Result. Alg.3 and Alg.4 achieve average 88.66% and 78.27% utility of Full CSI under Rice channel in Fig. 3(a). Alg.3 and Alg.4 achieve average 87.63% and 77.40% utility of Full CSI under Rayleigh channel in Fig. 3(b). The utility variation trend of our algorithms is nearly the same under different channel models. And both of them can converge to the stable state.

Answer. Our algorithms are robust to the channel condition dynamics.

6.3 RQ2: How does network scale affect our algorithms?

Motivation. Our algorithms are distributed algorithms that can scale well with the edge node numbers. We evaluate how the edge node number affects our algorithms.

Approach. To verify the scalability, we increase the average edge node number from 5 to 25 and increase the average

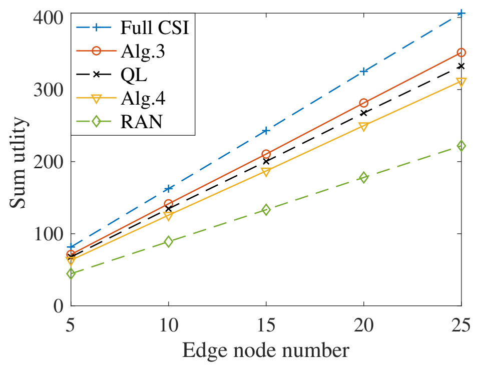


Fig. 4: Performance under different network scales.

UE number from 3 to 15 proportionally. We choose 5 nearest edge nodes as a candidate set for each UE. We plot the sum utility of all the UEs with edge node numbers in Fig. 4.

Result. Alg.3 and Alg.4 achieve average 86.81% (ranging from 86.55% to 87.00%) and 77.12% (ranging from 76.80% to 77.61%) utility of Full CSI within the investigated range in Fig. 4. Although the edge node number scales up, each UE can only choose from the five edge nodes in the candidate set and will be influenced by several UEs nearby. So the algorithm performance will not be influenced by the network size. From Fig. 4, our algorithms can scale linearly with edge node number. Alg.3 improves slightly (5.30%) sum utility of QL for the following reasons. First, Alg.3 performs with less information (without the joint strategy of all UEs), which deteriorates the average utility. However, Alg.3 introduces no-regret dynamics into better-response dynamics, which helps improve the utility. Alg.4 has a 12% performance loss compared with Alg.3 due to the delayed feedback.

Answer. Our algorithms scale well with the network size.

6.4 RQ3: How do weight factors affect our algorithms?

Motivation. Weight factors reflect the relative importance (preference) between the received CPU cycles and energy consumption. We try to explore how weight factors influence our algorithms.

Approach. We set λ_1 as the default value 2×10^{-6} and change λ_2 such that $\lambda_2/\lambda_1 = 10^q$, where $q \in [7.40, 11.10]$. λ_2 shows the dominance variation of the two weight factors, which is achieved by experiment.

Result. The average utility in Fig. 5(a) keeps unchanged when $\lambda_2/\lambda_1 \leq 10^{9.65}$ and drops sharply when $\lambda_2/\lambda_1 \leq 10^{9.65}$. The received CPU cycles in Fig. 5(b) do not change with the variation of λ_2/λ_1 because λ_1 is fixed. The average energy consumption fluctuates slightly when $\lambda_2/\lambda_1 \leq 10^{9.65}$ and drops sharply when $\lambda_2/\lambda_1 > 10^{9.65}$ in Fig. 5(c). The reason is that energy consumption dominates the utility when $\lambda_2/\lambda_1 > 10^{9.65}$.

Answer. The weight factors reflect the relative importance (preference) between the received CPU cycles and energy consumption in the utility function. The received CPU cycles dominate the utility when $\lambda_2/\lambda_1 \leq 10^{9.65}$, otherwise, the energy consumption dominates.

6.5 RQ4: Do our algorithms converge with no regret?

Motivation. Convergence and regret are critical factors to illustrate whether the learning-based algorithm can find the best solutions. Convergence represents the stability of the algorithm. Regret determines the expected performance loss over time. We give theoretical results in Theorem 2 and Theorem 3 and verify them in simulation.

Approach. We run our algorithms over 5000 time slots under a network scale with 25 edge nodes and 15 UEs. Each UE chooses 5 nearest edge nodes as a candidate set and competes with several UEs nearby. We record the average utility of Alg.3, Alg.4, and Q-learning in Fig. 6 (a). We record the average regret of both Alg.3 and Alg.4 in Fig. 6 (b).

Result. From Fig. 6(a), we observe that the average utility of all three algorithms can converge after about 2000 time slots (iterations). The convergence rate is determined by the action space size in our scenarios. Agl.3 and Alg. 4 have the similar convergence rate with the Q-learning algorithm because they have the same action space size. Besides, the enhancements in our algorithms have no influence on the convergence. This verifies the theoretical results as proved by Theorem 2 and Theorem 3. Moreover, Alg.3 and Alg.4 can guarantee zero average regret as shown in Fig. 6(b).

Answer. Alg.3 and Alg.4 converge to the NE with no regret as proved in Theorem 2 and Theorem 3. No UE is willing to change its strategy ultimately at the equilibrium and the learning loss is sublinear.

6.6 RQ5: Is there an optimal edge node group size?

Motivation. Since edge node group size affects the cooperative offloading performance, we try to explore the optimal size.

Approach. We set the number of edge nodes as eight and change the edge node group size from one to seven.

Result. The optimal edge node group size is four in our parameter settings. Alg.3 and Alg.4 can achieve an average of 96.99% and 92.50% utility of Full CSI in our settings. In Fig. 7(a), all the five algorithms have the highest utility when the edge node group size is four. The utility first increases and then decreases with group size. The reason may be that the increase of group size means more data transmission links for one UE, which improves the utility. However, it also leads to more resource contention among UEs, which degrades the utility. Hence, there exists an optimal edge node group size that maximizes the average utility.

Answer. Optimizing edge node group size can further improve offloading performance.

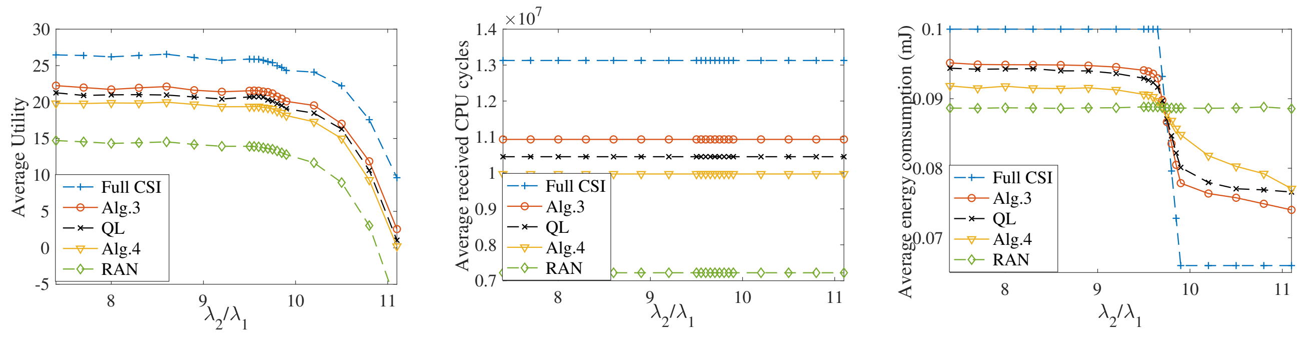
6.7 RQ6: How does feedback delay affect Alg.4?

Motivation. Intuitively, feedback delays affect the learning performance of Alg.4. We try to verify this in simulation.

Approach. We change the expectation of feedback delays μ from one to six. The feedback delays are calculated as $\min(\exp(\mu), \tau_{\max})$ where $\tau_{\max}$ is set as ten.

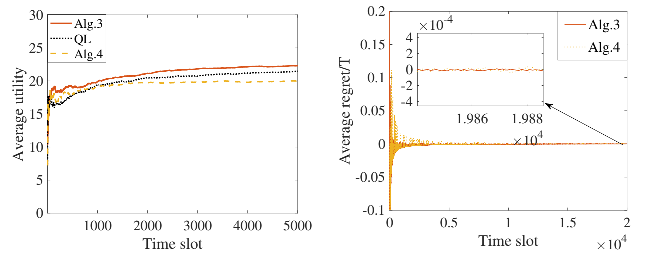
Result. Fig. 7(b) shows that the averaged utility drops with feedback delays because the increased delay leads to more absent feedback and worsens learning performance.

Answer. The feedback delay degrades the learning performance of Alg.4.



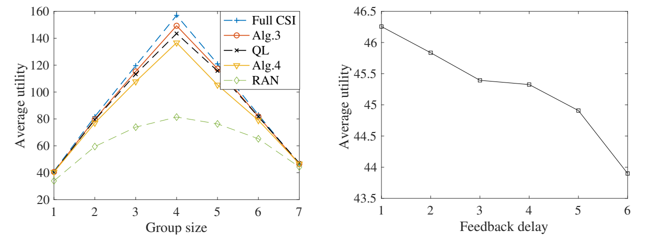
(a) Utility. (b) CPU cycles. (c) Energy consumption.

Fig. 5: Performance under different weight factors.



(a) Convergence. (b) Regret.

Fig. 6: Convergence and regret.



(a) Optimal group size. (b) Feedback delay.

Fig. 7: Impact of system parameters.

6.8 RQ7: How long do our algorithms need to make the edge node grouping decision in each time slot?

Motivation. We seek to evaluate the efficiency of Alg.3 and Alg.4 in terms of time consumption.

Approach. We set the duration of one slot as 500ms. Then, we record the runtime of all algorithms in each time slot on a typical PC and average over 2×10^4 slots.

Result. As shown in TABLE 3, Alg.3 and Alg.4 take 17.80ms and 9.82ms respectively in each time slot. Full CSI incurs the highest overhead, taking 180.87ms on average because Full CSI needs many iterations to reach NE in each time slot. By contrast, other algorithms need much less time, 17.80ms for Alg.3, and 16.43ms for QL because the two algorithms have only one iteration in each time slot and reach NE over multiple time slots. Alg.4 consumes 9.82ms because of feedback absence in some time slots.

TABLE 3: Algorithm Runtime

Algorithm	Runtime
Full CSI	180.87 ms
Alg. 3	**17.80 ms**
Q-Learning	16.43 ms
Alg.4	**9.82 ms**
Random Assignment	0.15 ms
Offloading decision cycle	500ms

Answer. Alg.3 and Alg.4 cause light overhead in practice with performance guarantees.

7 Conclusions

In this paper, we investigate decentralized mechanisms for edge node grouping. We first formulate this problem as a repeated game and prove that it is an exact potential game with a unique NE. Then, we propose a novel decentralized learning algorithm based on better-response dynamics and no-regret dynamics. Our mechanisms can converge to NE with no-regret. To extend to the scenarios (i.e., feedback is delayed and edge nodes have multiple frequency bands), we first enhance the algorithm by introducing feedback queues and then prove that our algorithms can be extended to the multiple frequency bands wireless model. In future work, we will carry out performance analysis to characterize how the number of edge nodes in a group affects the utility function, given the network conditions such as the density of users, the density of edge nodes, CSI dynamics. And we will optimize the edge node group size dynamically in the highly fluctuated network environment to further improve the edge node grouping performance.

References

[1] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[2] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing a key technology

towards 5G," *European Telecommunications Standards Institute white paper*, vol. 11, no. 11, 2015.

[4] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *proc. of IEEE Conference on Computer Communications*, 2020, pp. 2076–2085.

[5] X. Gong, "Delay-optimal distributed edge computing in wireless edge networks," in *proc. of IEEE Conference on Computer Communications*, 2020, pp. 2629–2638.

[6] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 14–19, 2018.

[7] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge computing framework for cooperative video processing in multimedia IoT systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2018.

[8] J. Kim, T. Kim, M. Hashemi, C. G. Brinton, and D. J. Love, "Joint optimization of signal design and resource allocation in wireless D2D edge computing," in *proc. of IEEE Conference on Computer Communications*, 2020, pp. 2086–2095.

[9] Y. Zhu, Y. Hu, and A. Schmeink, "Delay minimization offloading for interdependent tasks in energy-aware cooperative MEC networks," in *proc. of IEEE Wireless Communications and Networking Conference*, 2019, pp. 1–6.

[10] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Communications Letters*, vol. 23, no. 10, pp. 1816–1820, 2019.

[11] Y. Liu, X. Li, F. R. Yu, H. Ji, H. Zhang, and V. C. M. Leung, "Grouping and cooperating among access points in user-centric ultra-dense networks with non-orthogonal multiple access," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 10, pp. 2295–2311, 2017.

[12] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[13] W. Chen, Z. Su, Q. Xu, T. H. Luan, and R. Li, "VFC-based cooperative UAV computation task offloading for post-disaster rescue," in *proc. of IEEE Conference on Computer Communications*, 2020, pp. 228–236.

[14] Y. Wang, X. Tao, X. Zhang, P. Zhang, and Y. T. Hou, "Cooperative task offloading in three-tier mobile computing networks: An ADMM framework," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2763–2776, 2019.

[15] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6214–6228, 2020.

[16] M. Li, J. Gao, N. Zhang, L. Zhao, and X. Shen, "Collaborative computing in vehicular networks: A deep reinforcement learning approach," in *proc. of IEEE International Conference on Communications*, 2020, pp. 1–6.

[17] C. Funai, C. Tapparello, and W. Heinzelman, "Computational offloading for energy constrained devices in multi-hop cooperative networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 60–73, 2020.

[18] Y. H. Kao, K. Wright, P. H. Huang, B. Krishnamachari, and F. Bai, "MABSTA: Collaborative computing over heterogeneous devices in dynamic environments," in *proc. of IEEE Conference on Computer Communications*, 2020, pp. 169–178.

[19] X. He, H. Lu, H. Huang, Y. Mao, K. Wang, and S. Guo, "QoE-based cooperative task offloading with deep reinforcement learning in mobile edge networks," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 111–117, 2020.

[20] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1746–1760, 2017.

[21] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1619–1632, 2018.

[22] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4832–4841, 2020.

[23] S. Jošilo and G. Dán, "Selfish decentralized computation offloading for mobile cloud computing in dense wireless networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 207–220, 2019.

[24] L. Yang, H. Zhang, X. Li, H. Ji, and V. C. M. Leung, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2762–2773, 2018.

[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[26] C. Gao, Y. Li, Y. Zhao, and S. Chen, "A two-level game theory approach for joint relay selection and resource allocation in network coding assisted D2D communications," *IEEE Transactions on Mobile Computing*, vol. 16, no. 10, pp. 2697–2711, 2017.

[27] T. Q. Dinh, Q. D. La, T. Q. S. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Transactions on Communications*, vol. 66, no. 12, pp. 6353–6367, 2018.

[28] A. C. Chapman, D. S. Leslie, A. Rogers, and N. R. Jennings, "Convergent learning algorithms for unknown reward games," *SIAM Journal on Control and Optimization*, vol. 51, no. 4, pp. 3154–3180, 2013.

[29] H. P. Young, *Strategic learning and its limits*. OUP Oxford, 2004.

[30] N. Lee, D. Morales-Jimenez, A. Lozano, and R. W. Heath, "Spectral efficiency of dynamic coordinated beamforming: A stochastic geometry approach," *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 230–241, 2015.

[31] R. Tanbourgi, S. Singh, J. G. Andrews, and F. K. Jondral, "A tractable model for noncoherent joint-transmission

base station cooperation," *IEEE Transactions on Wireless Communications*, vol. 13, no. 9, pp. 4959–4973, 2014.

[32] W. Bao and B. Liang, "Optimizing cluster size through handoff analysis in user-centric cooperative wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 766–778, 2018.

[33] M. Rahman, H. Yanikomeroglu, and W. Wong, "Interference avoidance with dynamic inter-cell coordination for downlink lte system," in *proc. of IEEE Wireless Communications and Networking Conference*, 2009, pp. 1–6.

[34] D. Monderer and L. S. Shapley, "Potential games," *Games and economic behavior*, vol. 14, no. 1, pp. 124–143, 1996.

[35] Z. Zhou, P. Glynn, and N. Bambos, "Repeated games for power control in wireless communications: Equilibrium and regret," in *proc. of IEEE Conference on Decision and Control*, 2016, pp. 3603–3610.

[36] J. R. Marden, G. Arslan, and J. S. Shamma, "Regret based dynamics: convergence in weakly acyclic games," in *proc. of International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007, pp. 1–8.

[37] K. Spiros and K. Daniel, "Reinforcement learning of coordination in cooperative mas," in *proc. of National Conference on AI, Alberta*, 2002, pp. 326–331.

[38] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000.

[39] P. Joulani, A. Gyorgy, and C. Szepesvári, "Online learning under delayed feedback," in *proc. of International Conference on Machine Learning*, 2013, pp. 1453–1461.

[40] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 4, pp. 1179–1189, 2020.

[41] L. Xue, C. Sun, D. Wunsch, Y. Zhou, and F. Yu, "An adaptive strategy via reinforcement learning for the prisoners dilemma game," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 301–310, 2018.

[42] Z. Cao, C. Lin, M. Zhou, and R. Huang, "Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 825–837, 2019.

Acknowledgment

This work was supported by the National Key R&D Program of China (2020YFB1805500), the National Science Foundation of China (61922017, 61921003, 61902036).

Qing Li received the M.S. degree in The State Key Laboratory of Integrated Services Networks from Xidian University, Xi'an, China, in 2017. She is currently a Ph.D. candidate at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include cloud computing and mobile edge computing.

Xiao Ma received her Ph.D. degree in Department of Computer Science and Technology from Tsinghua University, Beijing, China, in 2018. She is currently a lecturer at the State Key Laboratory of Networking and Switching Technology, BUPT. From October 2016 to April 2017, she visited the Department of Electrical and Computer Engineering, University of Waterloo, Canada. Her research interests include mobile cloud computing and mobile edge computing.

Ao Zhou received the P.H.D degrees in Beijing University of Posts and Telecommunications, Beijing, China, in 2015. She is currently an Associate Professor with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. She has published 20+ research papers. She played a key role at many international conferences. Her research interests include Cloud Computing and Edge Computing.

Xiapu Luo is an Associate Professor with the Department of Computing, The Hong Kong Polytechnic University. He received the Ph.D. degree in Computer Science from The Hong Kong Polytechnic University, and was a Post-Doctoral Research Fellow with the Georgia Institute of Technology. His work appears top venues and received seven best paper awards (e.g., INFOCOM'18, ISPEC'17, ISSRE'16, etc.). His current research focuses on mobile and IoT security and privacy, blockchain and smart contracts, network security and privacy, software engineering, and Internet measurement.

Fangchun Yang received his Ph.D. in communications and electronic systems from the Beijing University of Posts and Telecommunication in 1990. He is currently professor at the Beijing University of Posts and Telecommunication, China. He has published 8 books and more than 100 papers. His current research interests include network intelligence, service computing and machine games. He is a fellow of the IET.

Shangguang Wang is a Professor at the School of Computer Science and Engineering, Beijing University of Posts and Telecommunications, China. He received his Ph.D. degree at Beijing University of Posts and Telecommunications in 2011. He has published more than 150 papers. His research interests include service computing, mobile edge computing, and satellite computing. He is currently serving as Chair of IEEE Technical Committee on Services Computing (2022-2013), and Vice-Chair of IEEE Technical Committee on Cloud Computing (2020-). He also served as General Chairs or Program Chairs of 10+ IEEE conferences. He is a Fellow of the IET, and Senior Member of the IEEE. For further information on Dr. Wang, please visit: http://www.sguangwang.com