MOTO: Mobility-Aware Online Task Offloading With Adaptive Load Balancing in Small-Cell MEC

Sijing Duan, *Student Member, IEEE*, Feng Lyu, *Senior Member, IEEE*, Huaqing Wu, *Member, IEEE*, Wenxiong Chen, *Member, IEEE*, Huali Lu, *Student Member, IEEE*, Zhe Dong, *Student Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

Abstract—Mobile edge computing is a promising computing paradigm enabling mobile devices to offload computation-intensive tasks to nearby edge servers. However, within small-cell networks, the user mobilities can result in uneven spatio-temporal loads, which have not been well studied by considering adaptive load balancing, thus limiting the system performance. Motivated by the data analytics and observations on a real-world user association dataset in a large-scale WiFi system, in this paper, we investigate the mobility-aware online task offloading problem with adaptive load balancing to minimize the total computation costs. However, the problem is intractable directly without prior knowledge of future user mobility behaviors and spatio-temporal computation loads of edge servers. To tackle this challenge, we transform and decompose the original task offloading optimization problem into two sub-problems, i.e., task offloading control (*ToC*) and server grouping (*SeG*). Then, we devise an online control scheme, named *MOTO* (i.e., Mobility-aware Online Task Offloading), which consists of two components, i.e., Long Short Term Memory based algorithm and Dueling Double DQN based algorithm, to efficiently solve the *ToC* and *SeG* sub-problems, respectively. Extensive trace-driven experiments are carried out and the results demonstrate the effectiveness of *MOTO* in reducing computational costs of mobile devices and achieving load balancing when compared to the state-of-the-art benchmarks.

Index Terms—Mobile edge computing, load balance, mobility-aware task offloading, reinforcement learning

1 Introduction

The proliferation of smart mobile devices (MDs) has brought rich convenience to our lives. However, the limited on-board energy and computing power of MDs have impeded the performance improvement for computation-intensive services, e.g., augmented/virtual reality and autonomous driving [2], [3], [4]. Mobile edge computing (MEC) is a promising paradigm to alleviate the computing burdens of MDs by deploying servers at the network edge [5], [6], [7]. With MEC, users can obtain high-quality computing services with low latency. Recently, integrating MEC with small-cell networks has drawn much attention considering the high-throughput performance of small-cell networks. By deploying edge servers at the small-cell based stations (SBSs), small-cell MEC enables agile service provisioning since MDs can offload their computation tasks to edge servers with reduced communication distance and fast response [8].

Nevertheless, the design of efficient computation offloading strategies in small-cell MEC system is a challenging task. First, MDs with diverse computing capabilities and computing task requirements may have different offloading demands. Second, mobile users are usually unevenly distributed with differentiated service request patterns, resulting in unbalanced computation loads on edge servers. Moreover, the load conditions of edge servers are time-varying since MDs are highly dynamic with frequent access and logout. Therefore, it is crucial to design an efficient mobility-aware task offloading strategy with adaptive load balancing in small-cell MEC systems.

In recent years, there have been many existing works investigating mobile task offloading in small-cell networks and MEC. For example, in [11], an adaptive cooperative and energy-efficient task offloading algorithm is presented for

- *Sijing Duan, Feng Lyu, Huali Lu, and Zhe Dong are with the School of Electrical and Computer Engineering, Central South University, Changsha, Hunan 410083, China. E-mail: {duansijing, fenglyu, huali_lu, rudy_dong}@csu.edu.cn.*
- *Huaqing Wu is with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada. E-mail: huaqing.wu1@ucalgary.ca.*
- *Wenxiong Chen is with the Research Institute of Languages and Cultures and the College of Information Science and Engineering, Hunan Normal University, Changsha, Hunan 410081, China. E-mail: chenwx@hunnu.edu.*
- *Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada. E-mail: sshen@uwaterloo.ca.*

Manuscript received 24 June 2022; revised 9 October 2022; accepted 1 November 2022. Date of publication 8 November 2022; date of current version 5 December 2023.
This work was supported in part by the National K&D Program of China under Grants 2022YFF0604504 and 2022YFC2009805, in part by the National Natural Science Foundation of China under Grant 62002389, in part by Young Elite Scientist Sponsorship Program by CAST under Grant YESS20200238, in part by the Key Research and Development Program of Hunan Province of China under Grant 2022GK2013, in part by the Natural Science Foundation of Hunan Province of China under Grant 2021JJ20079, in part by the Young Talents Plan of Hunan Province of China under Grant 2021RC3004, in part by 111 Project under Grant B18059, in part by Central South University Innovation-Driven Research Programme under Grant 2023CXQD029, in part by Hunan Education Department under Grant 18B043, and in part by the Research Project on Teaching Reform of Ordinary Colleges and Universities in Hunan Province under Grant HNJG-2020-0156. (Corresponding author: Feng Lyu.)
Digital Object Identifier no. 10.1109/TMC.2022.3220720

TABLE 1
Comparison With Some Related Works

Reference	Small cell MEC	Mobility- aware	Data-driven approach	Task offloading	Load balancing	DL-based method
Thananjeyan et al. [9]	No	Yes	No	Yes	No	No
Hu et al. [10]	No	Yes	No	Yes	No	No
Jing et al. [11]	Yes	No	No	Yes	No	No
Dai et al. [12]	No	No	Yes	Yes	No	Yes
Qian et al. [13]	No	No	Yes	Yes	No	Yes
Yang et al. [14]	Yes	No	No	Yes	No	No
Yang et al. [15]	Yes	No	No	Yes	No	No
Huang et al. [16]	Yes	No	No	Yes	No	Yes
Yang et al. [17]	No	No	No	Yes	No	Yes
Li et al. [18]	No	No	No	Yes	Yes	No
Wu et al. [19]	No	No	No	Yes	Yes	Yes
Liu et al. [20]	No	No	No	No	Yes	Yes
Hasan et al. [21]	Yes	No	No	No	Yes	No
Hu et al. [22]	Yes	No	No	No	Yes	No
Mohammad et al. [23]	Yes	No	No	No	Yes	No
Tang et al. [24]	No	No	No	Yes	No	Yes
Zhang et al. [25]	No	No	No	Yes	Yes	No
Yang et al. [26]	No	No	No	No	Yes	Yes
This paper	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

multiple small-cell MEC nodes. In [12], the authors design a data-driven task offloading in MEC-empowered vehicular networks. In [14], a distributed computation offloading strategy is investigated in small-cell networks integrated with MEC. In [15], an offloading strategy for NOMA-enabled hierarchical small-cell MEC is proposed. To minimize the overall energy consumption while ensuring the latency requirements, the authors of [16] focus on the joint design of computation offloading and interference coordination in small cell networks. Furthermore, the joint optimization of task offloading and resource allocation strategies are investigated in MEC systems, aiming to achieve the trade-off between energy efficiency and service delay [10], [27], [28], [29], [30]. However, the user mobility issue and load balancing are not considered in those offloading-related researches. On the other hand, there have been some works focusing on load balancing problems. For example, an optimization problem considering load balancing and task offloading is studied in MEC networks [18]. In [19], [20], load balancing solutions are proposed in vehicular MEC systems and IoT edge systems, respectively. In [21], [22], [23], load balancing issues are considered in small-cell networks. Despite the extensive works, the uneven spatio-temporal load issue and adaptive load balancing in small-cell MEC have not been well studied, calling for further investigations. We summarize the difference between this paper and the existing works in Table 1. The main differences are summarized as follows: (1) most existing works focus on either task offloading or load balancing, while investigation of the integration of both issues is rare; (2) the user mobility characteristics in small-cell MEC have not been well considered; and (3) most existing works mainly focus on theoretical modeling from a mathematical perspective, while paying little attention to the data-driven approach for DL model design.

To bridge this gap, in this paper, we study the mobility-aware online task offloading with adaptive load balancing in small-cell MEC. Specifically, we first justify the research motivations by conducting a comprehensive data analytics on a real-world dataset in a large-scale WiFi system (i.e., a typical example of small-cell MEC system). Based on the analysis on 29,284,966 association records of 21,725 users, we have two major observations. First, the mobility behaviors of most users are highly dynamic with short association durations. Second, the distributions of mobile user loads present uneven and dynamic spatio-temporal characteristics, which motivate us to conduct mobility-aware task offloading with achieving load balancing. To investigate the problem, we then formulate a task offloading optimization (*TOO*) problem, which is intractable directly since the future user mobility behaviors and the spatio-temporal computation loads of MEC servers are unavailable in advance. To this end, we transform and decompose the original problem into two sub-problems, i.e., Task offloading Control (*ToC*) and Server Grouping (*SeG*) with load balancing. Afterwards, we propose an online control scheme, named *MOTO* (i.e., Mobility-aware Online Task Offloading), to solve the two sub-problems. Particularly, *MOTO* consists of two components, i.e., Long short term memory (LSTM)-based algorithm and Dueling Double DQN (D3QN)-based algorithm, respectively solving the *ToC* and *SeG* sub-problems. Finally, we implement our proposed *MOTO* scheme and conduct extensive trace-driven experiments, which demonstrate that the proposed scheme can achieve the superior performance in terms of system computational costs and load balancing.

Our main contributions are summarized as follows:

- We investigate the mobility-aware online task offloading in a practical small-cell MEC system, which is of significant importance for mobile service provisioning to keep pace with the dynamic network conditions. To justify the research motivations, we conduct a comprehensive data analytics on a real-world dataset in a large-scale WiFi system with several major observations.
- Inspired by the observations, we formulate a *TOO* problem to investigate the mobility-aware task

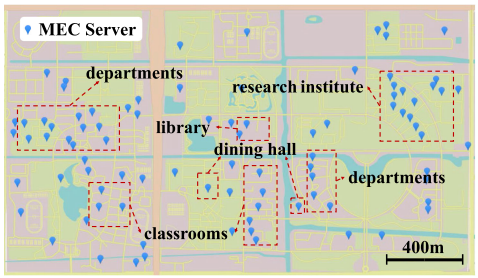
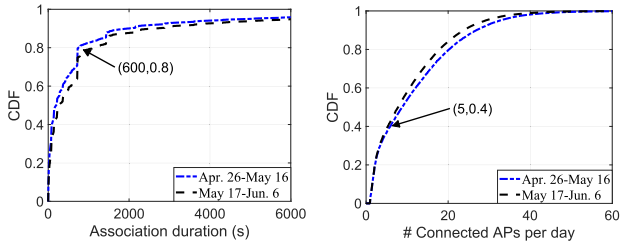


Fig. 1. APs deployment map.



(a) CDFs of association duration (b) CDFs of the number of connected APs for each user

Fig. 2. Overall user association analysis.

offloading control strategy. To solve the non-convex and intractable *TOO* problem, we decompose it into two sub-problems: 1) the task offloading control (*ToC*) sub-problem which optimizes the task offloading decisions to adapt to dynamic user mobility behaviors, and 2) the server grouping (*SeG*) sub-problem that groups MEC servers to address the spatially and temporally uneven computation loads.

- We propose a task offloading control scheme called *MOTO* to minimize the total computation costs. In *MOTO*, we integrate two major techniques: LSTM-based method for *ToC* sub-problem and D3QN-based method for *SeG* sub-problem. The proposed *MOTO* can effectively control online task offloading and achieve adaptive load balancing.

The remainder of this paper is organized as follows. We conduct a systematical data analysis and present our observations and motivations in Section 2. Section 3 gives the system model and problem formulation. We decompose the problem and elaborate on our *MOTO* design in respective Sections 4 and 5. In Section 6, we evaluate the performance of proposed *MOTO* with trace-driven experiments. Section 7 reviews the related work. Finally, we conclude the paper and direct our future work in Section 8.

2 Motivations

In this section, we justify the motivations of our study with some data-driven observations. Specifically, we adopt a public large-scale WiFi dataset[1], which contains 4045 access points (APs) and more than 21,725 active users. When mobile users connect to APs, the system can record network association information, including the connection time, disconnection time, and consumed traffic volume, etc. Fig. 1 visualizes the scope of the WiFi system, which covers classrooms, department buildings, libraries, dining halls, and research institutes in a university.

2.1 User Mobility Issue

We first study the overall distributions of user associations. Fig. 2a shows the cumulative distribution functions (CDFs) of user association duration, where results from Apr. 26 to May 16 and from May 17 to Jun. 6 are plotted. We can achieve two major observations. First, two curves show a quite close trend, indicating similar connection behaviors in these two time periods. Second, more than 80% of the association duration are less than 600s, which means that users are not inclined to keep a long connection with associated AP. Fig. 2b shows the CDFs of the number of connected APs for each user per day. We can observe that 40% of users connect to less than 5 APs per day and about 20% of users associate more than 20 APs, which *indicates that users usually move frequently among multiple geographic locations with a short duration staying in each location*. Likewise, the two curves within different time periods are quite similar, which demonstrate that the observed phenomenons exist with a long time span.

2.2 Dynamics of User Load

We select one department building to study the temporal dynamic characteristics of mobile users. Fig. 3 shows the total number of users, as well as the actual increments/decrements of users connecting/disconnecting the edge server in the building from 11:00 a.m to 13:00 p.m., where the time slot duration is minutely based. We can observe that the number of users has symmetries and reaches the maximum value (i.e., 1,000) around 12 p.m. It further indicates that the users move frequently during certain time periods, e.g., lunchtime. *The highly dynamic temporal variation motivates us to make computation load predictions in advance for better task offloading decisions.*

To better understand the spatio-temporal dynamics of user load distribution, we further analyze the variations in the number of users on different buildings at different time slots. In particular, we count the number of users in all buildings at different time slots on one day, where the sampling interval is 15 minutes. Fig. 4 shows the heatmap of user distribution, where the server ID is ranked in descending order by the number of users at 8 a.m. In addition, users are mainly active between 08:00 a.m and 20:00 p.m. We can observe that most of the users gather in several hot buildings, while the number of users in other buildings is small. *Therefore, we need to consider load balancing among servers for better service provisioning and resource utilization.*

We then examine the spatio-temporal dynamics of the user load in different locations. Fig. 5 visualizes the number of users at two different moments on one day, i.e., 11:00 a.m

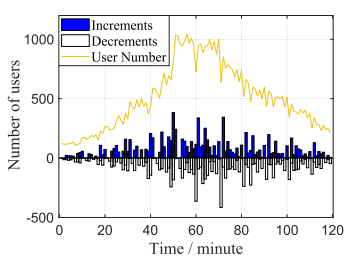


Fig. 3. Dynamics of the number of users in one building.

1. Online available. https://github.com/Intelligent-WiFi/DataSet

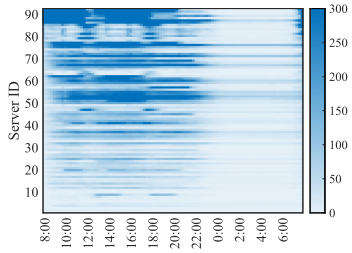
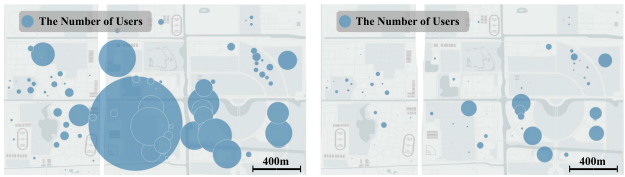


Fig. 4. Heatmap of user distribution.



(a) 11:00 a.m (b) 23:00 p.m

Fig. 5. Spatio-temporal dynamics of computation load.

and 23:00 p.m, where circles denote the number of users and larger size implies more users. We can have the following two major observations. First, the number of users is temporally uneven. Particularly, the computation load at 11:00 a.m is much larger than that of 23:00 p.m[2]. It is reasonable since users are more active in the daytime than nighttime. Second, the user loads are spatially variant. For example, the user load of classrooms is heavier than that of departments and research institutions.

Therefore, the highly dynamic mobility behaviors of users and uneven load distributions motivate us to investigate mobility-aware task offloading with load balancing, to achieve optimal resource utilization.

3 System Model and Problem Formulation

In this section, we first describe the system model and then formulate the mobility-aware task offloading problem.

3.1 System Model

We consider a typical small-cell MEC scenario, where mobile edge servers (MESs) communicate with each other via wireless network[3]. Fig. 6 shows an example of the scenario, where MESs are located in each building. We partition the long-term duration into T consecutive time slots[4]. In each time slot, there are many MDs connecting to MESs for computation service requests through the connected wireless network. Due to the mobility characteristics[5], mobile users may move among buildings and be served by MESs deployed in the corresponding building. Denote by $\mathcal{U} = \{\mathrm{MD}_1, \mathrm{MD}_2, \ldots, \mathrm{MD}_i, \ldots, \mathrm{MD}_N\}$ the set of MDs, and by $\mathcal{P} = \{\mathrm{MES}_1, \mathrm{MES}_2, \ldots, \mathrm{MES}_j, \ldots, \mathrm{MES}_M\}$ the set of MESs, where $N = |\mathcal{U}|$ and $M = |\mathcal{P}|$. The key notations are summarized in Table 2. Each MD can only be served by one MES in each time slot. The uneven distribution of users and differences in service requests result in uneven loads for MESs. For example, the loads of MES A and MES B in crowded places are heavy, while loads of MES C and MES D are light.

2. We assume that the computation loads are proportional to the number of users.

3. The considered problem and the proposed scheme are readily applicable to general WiFi-based and 5G-based small-cell networks.

4. The system is managed and scheduled every time slot, the duration of which can be set flexibly in accordance with the system requirement.

5. Since we do not restrict the users' mobility characteristics in the model design, the proposed scheme can also be applied to scenarios with high-mobility users.

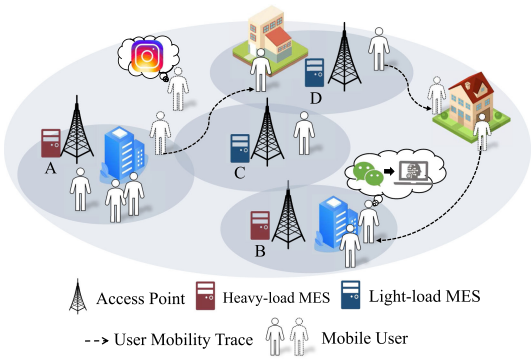


Fig. 6. System architecture of small-cell MEC.

3.1.1 Task Buffer Model

We consider that tasks arrive at MD_i following a Bernoulli process with a parameter λ_i, where the MD has at most one task to be processed in each time slot. Without loss of generality, the computing tasks are atomic, which can be executed locally or be offloaded to an MES. Each task can be described by a tuple $\{\rho, l\}$, where ρ represents the average data size required to be transmitted when the MD offloads the task to an MES, and l indicates the average CPU cycles required to finish the task computation.

For each task, there are two processing options, i.e., local computing or edge computing via offloading. Generally,

TABLE 2
Key Notations

Notation	Definition
$\mathcal{U}, \mathcal{P}$	the set of MDs and MESs
M, N, i, j	the number and indexes of MDs and MESs
ρ, l	the average data size of task, the average CPU cycles required to finish the task computation
f_i, f_j	the CPU frequency of MD_i and MES_j
x_j	the offloading probability of MES_j
μ_i, μ_j	the task processing rate of MD_i and MES_j
μ_g	the sum of computing power of all MESs in group g
p_i	the transmission power of MD_i
D_i^L	the expected time for local computing of MD_i's task
D_i^O	the expected time for MD_i's task executed at edge
E_i^L	the computational energy cost of MD_i
E_i^O	The energy cost to support the transmission of MD_i
C_i^L	the weighted local computing cost of MD_i's task
C_i^O	the weighted expected cost for edge computing of MD_i's task
C_i	the weighted computational costs of MD_i's task
V_j	the set of MDs served by MES_j
r	the average data rate of wireless links between MD and MES
α	the weight coefficient for delay
$\sigma_{a,b}$	the indicator of whether MES_a and MES_b belong to the same group
$\mathcal{G}$	the number of MES groups

MDs will offload as many tasks as possible to the MESs to minimize their computational costs while satisfying the computing capability constraints of MESs. But too many tasks flooding into the MES buffer will reduce the quality of service. Therefore, each MES_j will set a task offloading probability x_j to limit MD task offloading. In each time slot, MES broadcasts the edge computing delay and offloading probability x to all MDs. The goal is to minimize the sum cost of all MDs by making task offloading control decisions at each MES. Specifically, the MESs firstly calculates the offloading probability $\mathbf{X} = \{x_1, x_2, \ldots, x_j, \ldots, x_M\}$ for all MESs, and all MDs associated with MES_j offload their tasks with a probability of $x_j \in [0, 1]$.

3.1.2 Local Computing Model

Denote the CPU frequency of MD_i by f_i, the local task processing rate μ_i can be represented by $\mu_i = \frac{f_i}{l}$. Then, the expected time spent for local computing can be calculated as

$$D_i^L = \frac{1}{\mu_i - (1 - x_j)\lambda_i}. \tag{1}$$

The computational energy cost to support local computing can be calculated by

$$E_i^L = k{f_i}^2 l, \tag{2}$$

where k represents the energy consumption coefficient, which mainly depends on the chip architecture.

Combining Eqs. (1) and (2), the weighted expectation cost for local computing can be obtained by

$$C_i^L = \alpha D_i^L + (1 - \alpha)E_i^L, \tag{3}$$

where $\alpha \in [0, 1]$ is the weight coefficient for delay, and $1 - \alpha$ is for the energy cost. If the MD cares more about the delay performance, a larger α can be set.

3.1.3 Edge Computing Model

Denote by V_j the set of MDs that are currently served by MES_j. The cost for task execution on the MES consists of two parts: 1) the transmission delay and energy cost to offload the task; and 2) the expected computation delay to execute the task at edge.

Let f_j be the CPU frequency of MES_j, and then the task process rate of MES_j can be calculated as $\mu_j = \frac{f_j}{l}$. Denote by r the average data rate of wireless links between MDs and MESs. Then, the expected time spent for the task executed at edge can be calculated as

$$D_i^O = \frac{\rho}{r} + \frac{1}{\mu_j - x_j \sum_{i=1}^{v_j} \lambda_i}, \tag{4}$$

where $v_j = |V_j|$ represents the number of MDs in the set V_j, and x_j should satisfy $\mu_j - x_j \sum_{i=1}^{v_j} \lambda_i \geq 0$ due to the computing capability constraint of MES_j. The energy cost to support the MD transmission can be calculated by

$$E_i^O = p_i \frac{\rho}{r}, \tag{5}$$

where p_i is the transmission power of MD_i.

Combining Eqs. (4) and (5), the weighted expected cost for edge computing can be achieved by

$$C_i^O = \alpha D_i^O + (1 - \alpha)E_i^O. \tag{6}$$

3.2 Problem Formulation

At the beginning of each time slot, task offloading control decisions are made to minimize the computational costs of all MDs in the time slot based on the offloading decision $\mathbf{X}$. We first define the weighted computational cost of MD_i's tasks with Eqs. (3) and (6):

$$C_i = (1 - x_j)C_i^L + x_j C_i^O. \tag{7}$$

The sum of computational costs of all MDs in the system can be calculated as

$$C_{total} = \sum_{i=1}^{N} C_i. \tag{8}$$

Therefore, the task offloading optimization (*TOO*) problem can be formulated as **P1**:

P1 (Original *TOO* Problem):

$$\begin{aligned} \min_{\{\mathbf{X}\}} \quad & C_{total} \\ s.t. \quad & \mathbf{C1}: 0 \leq x_j \leq \min(1, \tfrac{\mu_j}{\sum_{i=1}^{v_j} \lambda_i}), \forall j \in \mathcal{P}. \end{aligned} \tag{9}$$

Directly tackling the above **P1** problem is difficult since the dynamic user mobility behaviors result in spatially and temporally uneven computation load distribution. Furthermore, the status of task arrival rate in the time slot is unavailable in advance.

4 Problem Transformation and Decomposition

In this section, we first introduce an MES grouping approach and transform **P1** into a group-based *TOO* problem to facilitate load balancing. Then, to solve the non-convex and intractable problem, we further decompose the problem into two sub-problems: 1) the task offloading control (*ToC*) sub-problem which optimizes the task offloading decisions to adapt to user mobility behaviors; and 2) the server grouping (*SeG*) sub-problem that addresses the spatially and temporally uneven computation loads to achieve load balancing.

4.1 Problem Transformation

The objective of **P1** is to find the optimal offloading control decision for each MES to serve its associated MDs. However, considering the uneven load of different MESs in the real-world scenario, it is inefficient that each MES only serves its associated MDs. For example, some overloaded MESs are not able to support high-quality service provisioning while some light-loaded MESs's resources are underutilized.

To solve this problem, a potential approach is to transfer parts of tasks from heavy-loaded MESs to light-loaded MESs. For each heavy-loaded MES, we need to determine the target MES for task transfer. Although we can find the target MES via exhaustive searching, the searching delay is unacceptable. In this work, we leverage a grouping-based approach, where several MESs are clustered into one group and the tasks can only be transferred within the group.

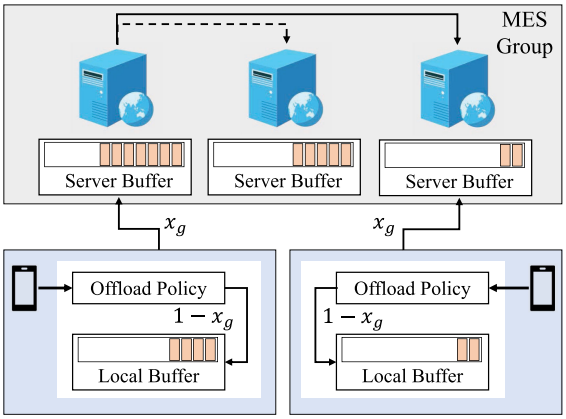


Fig. 7. Illustration of offloading decision in a group.

Denote by $\mathbf{G}=\{\sigma_{a,b}\}$ the grouping decision, where $\mathrm{MES}_a, \mathrm{MES}_b \in \mathcal{P}$ and $\sigma_{a,b} \in \{0,1\}$ indicates whether MES_a, MES_b belong to the same group, i.e.,

$$\sigma_{a,b}=\begin{cases}1 & \text{If MESa and MESb are in the same group;}\\ 0 & \text{Otherwise.}\end{cases} \tag{10}$$

Denote by g an MES group, and $\mathcal{G}$ the set of groups. A group g includes MESs and MDs associated with those MESs. All MESs in g are viewed as a whole, and the probability of offloading tasks to MESs in g is x_g. Let y_g and v_g denote the number of MESs and MDs in g, respectively. The sum of computing rate of all MESs in g can be calculated as

$$\mu_g=\sum_{j=1}^{y_g}\mu_j. \tag{11}$$

After merging servers into a group, the tasks of users belonging to g may be executed on any MES in this group. Fig. 7 shows the task offloading process after grouping. In particular, when MD_i has a task to be processed, it first makes an offloading decision based on x_g. If MD_i chooses local processing, the task will be placed in the local task buffer. Otherwise, the task will be uploaded to MES_a which is directly connected to MD_i, and then MES_b will be determined to perform the task. For simplicity of description, we consider that all MESs in the same group have the same computing rate ($\mu_a=\mu_b$) [6]. It means that the probability of tasks being executed on any MES is equal, and the amount of tasks handled by each server is also equal. In this case, the probability that a task needs to be transferred to another server in the group for execution is $\frac{y_g-1}{y_g}$. Then, the edge computing delay for MD_i belonging to g can be recalculated as

$$D_i^O=\frac{\rho}{r}+\frac{y_g-1}{y_g}\frac{\rho}{r^{trans}}+\frac{1}{\mu_g-x_g\sum_{i=1}^{v_g}\lambda_i}. \tag{12}$$

where r^{trans} represents the average transmission data rate between MESs.

With MES grouping, the sum computational costs of all MDs can be calculated as

$$C_{total}^T=\sum_{g=1}^{|\mathcal{G}|}C_{total}^g, \tag{13}$$

where $|\mathcal{G}|$ is the number of MES groups based on grouping decision $\mathbf{G}$, and C_{Total}^g represents the sum computational costs of all users in g, i.e.,

$$C_{total}^g=\sum_{i=1}^{v_g}C_i. \tag{14}$$

With group-based adaptive load balancing, the original problem **P1** can be transformed into

P2 (Transformed Problem):

$$\begin{aligned}&\min_{\{\mathbf{G},\mathbf{X}\}} \quad C_{total}^T\\ &s.t. \quad \mathbf{C1}: 0\le x_g\le \min(1,\tfrac{\mu_g}{\sum_{i=1}^{v_g}\lambda_i}), \forall g\in\mathcal{G}.\end{aligned} \tag{15}$$

The transformed problem **P2** includes *ToC* and *SeG* sub-problems, which optimize the task offloading decision X and server grouping decision with load balancing G, respectively. Therefore, by solving **P2**, we can achieve the optimal task offloading control decision $\mathbf{X}^*$ and the optimal server grouping decision $\mathbf{G}^*$, where the number of users and computing resources of each group are matched optimally.

4.2 Problem Decomposition

Since $\mathbf{G}^*$ is discrete and $\mathbf{X}^*$ is continuous, **P2** is a typical mixed integer nonlinear programming (MINLP) problem [31]. Generally, the spatial branch and bound (SBB) method can be adopted to solve MINLP problems [32]. However, due to high complexity, this method is not suitable for our problem which requires real-time decisions to adapt to dynamic environments.

In this work, we use the Tammer method to decompose **P2** into two sub-problems to reduce the complexity [33]. Particularly, we first rewrite the transformed problem **P2** as:

$\widetilde{\mathbf{P2}}$ (Equivalent Problem):

$$\begin{aligned}&\min_{\{\mathbf{G}\}}\left(\min_{\{\mathbf{X}\}} C_{total}^T\right).\\ &s.t. \quad \mathbf{C1}.\end{aligned} \tag{16}$$

Since **C1** only constrains the solution of $\mathbf{X}^*$, solving problem $\widetilde{\mathbf{P2}}$ is equivalent to solving the following two sub-problems, i.e., the task offloading control (*ToC*) sub-problem **P2.1** and the server grouping (*SeG*) sub-problem **P2.2**.

P2.1 (*ToC* Sub-Problem):

$$\begin{aligned}&C_{total}^*=\min_{\{\mathbf{X}\}}\sum_{g=1}^{|\mathcal{G}|}C_{total}^g\\ &s.t. \quad \mathbf{C1}.\end{aligned} \tag{17}$$

P2.2 (*SeG* Sub-Problem):

$$\min_{\{\mathbf{G}\}} C_{total}^*. \tag{18}$$

5 Design of *MOTO*

In this section, we elaborate on the design of *MOTO*. Particularly, we first describe its architecture and present the workflow. Then, we concentrate on the major technical components of *MOTO*.

6. If MESs have different computing rates, the system will assign tasks to different MESs based on their computing capabilities.

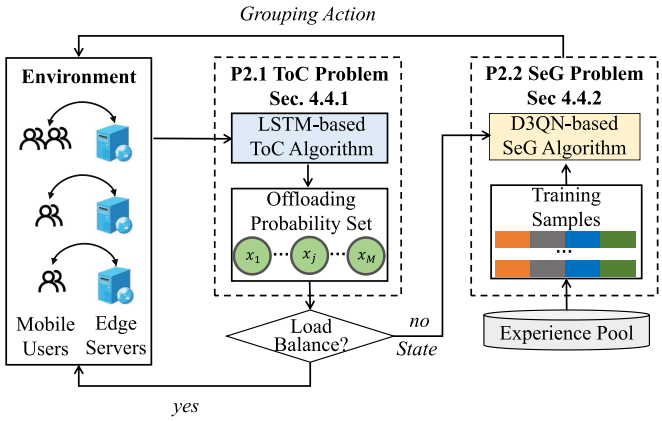


Fig. 8. An overview of the proposed *MOTO*.

5.1 Overview

Fig. 8 shows the workflow and overall architecture of *MOTO*, which consists of two components, i.e., LSTM-based algorithm for *ToC* and D3QN-based algorithm for *SeG*. Since the task arrival rates are unknown in advance, *MOTO* determines task offloading strategy in an online learning manner based on dynamic network environment. Specifically, the system initializes each MES as one group and solves **P2.1** by predicting the task arrival rates of all MDs, and get the optimal offloading probability set $\mathbf{X}^*$. Afterwards, the system determines whether the system is load balanced. If the system is in a load-balanced state, we return the current decision to the environment. Otherwise, the MESs need to be regrouped by solving **P2.2**.

The computation loads of MESs in different locations are uneven in both time and spatial domains. To improve the intelligence of server grouping strategy, we transform **P2.2** into a typical Markov Decision Process (MDP) problem, and design a D3QN-based reinforcement learning (RL) algorithm to solve it. The D3QN algorithm takes the current *state* X^t and samples from the experience pool as the inputs, outputs a grouping *action*, and gets a *reward*. The online learning process continues until the system can make satisfying offloading control decisions in accordance with the dynamic environments. In the following subsections, we will elaborate on the two algorithms.

5.2 LSTM-Based Algorithm for *ToC*

Since task offloading in different groups is independent, solving **P2.1** is equivalent to finding the optimal x_g^* for each group. In this section, we first prove that *ToC* Sub-Problem is convex. Then, we design an LSTM-based algorithm to solve it. To obtain the optimal solution for **P2.1**, we derive Lemma 1 in the following.

Lemma 1. *C_{total}^g is a convex function in the definition domain* $0 \le x_g \le \min(1, \frac{\mu_g}{\sum_{i=1}^{v_g}\lambda_i})$.

Proof. Based on (7) and (14), we can rewrite C_{total}^g as

$$C_{total}^g(x_g) = \sum_{i=1}^{v_g}\left((1-x_g)C_i^L + x_g C_i^O\right). \tag{19}$$

It is easy to see C_{total}^g is a higher-order function of x, including higher-order terms, first-order terms and constant terms. For convenience, we extract the higher-order terms as

$$\psi(x_g) = (1-x_g)\alpha\sum_{i=1}^{v_g}\frac{1}{\mu_i - (1-x_g)\lambda_i} + x_g\alpha\sum_{i=1}^{v_g}\frac{1}{\mu_g - x_g\sum_{i=1}^{v_g}\lambda_i}. \tag{20}$$

The second derivative of $\psi(x_g)$ is calculated as

$$\psi''(x_g) = \sum_{i=1}^{v_g}\left(\frac{2\alpha\lambda_i\mu_i}{(\mu_i-(1-x_g)\lambda_i)^3} + \frac{2\alpha\lambda_g\mu_g}{(\mu_g - x_g\lambda_g)^3}\right). \tag{21}$$

Obviously, λ_i, μ_i, λ_g and μ_g are greater than 0. In addition, for MD_i, the task arrival rate λ_i must be less than the task processing rate μ_i, and x_g is less than 1. So we can get

$$\mu_i - (1-x_g)\lambda_i > 0. \tag{22}$$

Generally, the task processing rate of an MES is greater than its task arrival rate. Thus, $\mu_g - x_g\lambda_g$ is greater than 0 and ψ'' is a positive number, and $C_{total}^g(x_g)$ is a convex function. □

With **Lemma 1**, **P2.1** can be easily solved with given task arrival information, which however is not available beforehand when making the offloading decisions. Therefore, for every group g, we devise an LSTM-based *ToC* algorithm to predict the task arrival rates of all MDs in the group $\lambda_g^{t_0+1}$ in the upcoming time slot, as illustrated in Fig. 9.

LSTM-Based ToC*Algorithm*. LSTM is a variant of Recurrent Neural Network (RNN) to effectively deal with time-series data prediction [34], [35]. Denote by $\mathcal{TN} = \{tn_1, \cdots, tn_t, tn_{ls}\}$ the time series of historical tasks, where ls is the length of time series, and each element tn_t is the number of tasks arrival at time slot t. At each time slot t, the *LSTM* cell can be calculated as

$$\begin{aligned} e_t &= \sigma(W_e[h_{t-1}; tn_t] + b_e),\\ i_t &= \sigma(W_i[h_{t-1}; tn_t] + b_i),\\ o_t &= \sigma(W_o[h_{t-1}; tn_t] + b_o),\\ \hat{m}_t &= \tanh(W_m[h_t - 1; tn_t] + b_m), \end{aligned} \tag{23}$$

where e_t, i_t, o_t, and $\hat{m}_t$ are forget gate, input gate, output gate, and modulated input, $\sigma(\cdot)$ is the Sigmoid function, and tn_t is the input. Then the memory cell and hidden state are updated as

$$\begin{aligned} m_t &= e_t \odot m_{t-1} + i_t \odot \hat{m}_t,\\ h_t &= o_t \odot \tanh(m_t), \end{aligned} \tag{24}$$

where h_t is the output of the *LSTM* cell at step t.

We adopt three LSTM layers in this paper, each LSTM cell takes the vector representation, memory state, and hidden state at time slot $t-1$ as input at time slot t

$$h_t, c_t = LSTM(tn_{t-1}, h_{t-1}, c_{t-1}), \tag{25}$$

where c_{t-1} and h_{t-1} are the memory state and the hidden state, respectively, and the output h_t denotes the task arrival rates at the next time slot, i.e., λ_g^{t+1}. Based on the predicted task arrival information, we can get the optimal task offloading probability x_g^* for each group

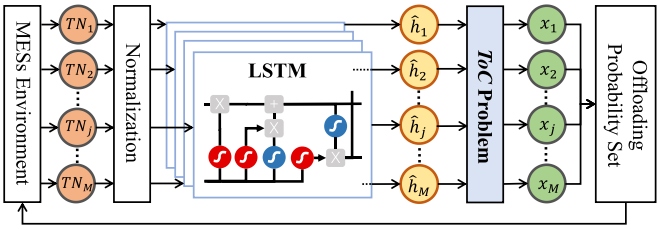


Fig. 9. The architecture of LSTM-based *ToC* algorithm.

$$x_g^* = \mathbf{argmax}_{x_g} C_{total}^g(x_g), \tag{26}$$

which can be solved in polynomial time with the Newton method [36].

Recall that the task offloading probability should satisfy the computing constraint $\mu_j \geq x_g \sum \lambda_i$. For MES_j with limited computation capability, the associated MDs will decrease x_j to avoid MES overload. In this paper, we use $|x_a - x_b|$ to represent the load gap between MES_a and MES_b. Note that for MESs in the same group, their computational loads are balanced since they cooperate to perform computation via task transfer. Considering the highly dynamic network environment, absolute load balancing is difficult to achieve unless all MESs are in the same group. Therefore, we consider that load balancing can be achieved once the load gap (i.e., $x_{max} - x_{min}$) is less than the maximum value of δ. The x_{max} and x_{min} denote the maximum and minimum values of decision $\mathbf{X}$, respectively. When the system service loads are unbalanced, the MESs need to be regrouped.

5.3 D3QN-Based Algorithm for *SeG*

To solve the *SeG* sub-problem, we first model it as an MDP and then propose a D3QN-based algorithm to optimize MES grouping.

Problem Mapping. A typical MDP model consists a tuple with five parameters $\langle S, A, P, R, \gamma \rangle$, representing the state space, action space, state transition probability, reward function, and future reward discount factor, respectively. The value of γ is between 0 and 1, where a larger γ means that more attention is paid to future rewards.

Based on the value of $\mathbf{X}$ obtained from solving the *ToC* sub-problem, if the computational loads are unbalanced, the MES grouping algorithm will be activated to regroup MESs. In other words, the grouping module only works when needed, which means that the time interval between two grouping actions is dynamic. We define the time interval between two consecutive actions as a logical step τ, and let t denote the time interval when MESs update task offloading probabilities. Therefore, one time step τ might contain multiple time slots t.

State Space: Let S denote the collection of all the states of environment. In our model, we set the task offloading probability of all MESs as the state at τ, i.e., $s^\tau = \{x_j^\tau\} \in S$, $1 \leq j \leq M$.

Action Space: Let A be the collection of all actions. The goal of server grouping is to split and regroup all MESs into new MES groups. Considering the tremendous dimension of grouping decisions, we divide the procedure into multiple actions. The derivation process is as follows.

Denote by M the number of MESs. If n_1 MESs are selected as a group from M MESs, the number of combinations is $\frac{M!}{n_1!(M-n_1)!}$. Then, n_2 MESs are selected from the remaining $(M-n_1)$ MESs as a group with the number of combinations being $\frac{(M-n_1)!}{n_2!(M-n_1-n_2)!}$. The above process is repeated until there is no remaining MES, i.e., the server grouping is finished. We assume that all MESs are divided into H groups, then the size of grouping action space can be calculated by:

$$\begin{aligned} |\mathcal{A}| &= \frac{M!}{n_1!(M-n_1)!} \frac{(M-n_1)!}{n_2!(M-n_1-n_2)!} \\ &\cdots \frac{(M-n_1-n_2-\cdots-n_{H-1})!}{n_H!(M-n_1-n_2-\cdots-n_H)!} \\ &= \frac{M!}{n_1! \times n_2! \times \cdots \times n_H!} \end{aligned} \tag{27}$$

According to Equation (27), the searching latency is positively correlated with the number of MESs in the group and the largest possible size of the action space is $M!$ (i.e., each group contains only one MES). Such a large size action space makes the parameter learning process of reinforcement learning very difficult. To solve the above problem, in this paper, each action only combines or splits two MESs during server grouping and regrouping. Specifically, let action space at t_0 be the $a^{t_0} = \{\text{Flag}^{t_0}, MES_i^{t_0}, MES_j^{t_0}\} \in \mathcal{A}$, where $MES_i^{t_0}, MES_j^{t_0} \in M$ and $Flag^{t_0} \in \{0, 1\}$. Particularly, $Flag^{t_0} = 0$ and $Flag^{t_0} = 1$ represent combining and splitting $MES_i^{t_0}$ and $MES_j^{t_0}$, respectively. In this way, the action space is reduced from $M!$ to $2M(M-1)$. Particularly, if the action is to merge MES_i and MES_j which originally belong to different groups, MES_i and MES_j will form a new group. After grouping, when the computation tasks arrive, the serving MES with the shortest task buffers (i.e., with the minimum task loads) within a group will be chosen.

Reward: As mentioned before, a logic step τ may include multiple time slots. The reward of each time slot can be calculated as

$$r_t = \frac{1}{N} \sum_{i=1}^{N} (C_i^L - C_i^O). \tag{28}$$

Then the reward of step τ can be given as

$$r_\tau = \frac{1}{h^\tau} \sum r_t, \tag{29}$$

where h^τ represents the number of time slots in τ, which depends on the speed of environment change. The slower the environment changes at τ, the larger the h^τ.

Based on the MDP model, the grouping sub-problem **P2.2** is transformed into an optimization problem that finds the optimal grouping policy π^* to maximize the reward of all users.

Definition 1 (MES Grouping Policy). *MES Grouping Policy π represents the mapping relationship between S and A, and π^* is the best mapping with the highest reward. With π^*, the model can directly obtain the best grouping action $a^{\tau*} = \{Flag^{\tau*}, MES_i^{\tau*}, MES_j^{\tau*}\} \in A$ at time τ based on the observed state $s^\tau = \{x_j^\tau\} \in S$.*

D3QN-Based SeG Algorithm. A D3QN-based algorithm is designed to find the optimal MES grouping policy, as shown in Fig. 10. D3QN is a classical RL algorithm for Markov Decision Process, which has been proven to achieve

good performance in large-scale discrete state space. D3QN combines the characteristics of Double DQN [37] and Dueling DQN [38]. It contains two neural networks with the same structure, namely, the online network that interacts with the environment and the target network that stores parameters. The design of double networks avoids the overestimation problem in learning.

Algorithm 1. *MOTO*: Mobility-Aware Online Task Offloading Scheme With Adaptive Load Balancing

Initialization: $t, \tau \leftarrow 0$; $x_g^t \leftarrow 1, \forall x_g^t \in \mathcal{X}^t; |g| \leftarrow 1, \forall g \in \mathcal{G}$; $r^\tau \leftarrow 0, s^\tau \leftarrow \mathcal{X}^t, a^\tau \in A$
while $t \rightarrow TZ$ **do**
  for $g \in \mathcal{G}$ **do**
    $n_g^t \leftarrow 0, r^t \leftarrow 0$.
    for $i \in g$ **do**
      Calculate C_i^L, C_i^O.
      if i decides to offload task based on x_g^t **then**
        $n_g^t \leftarrow n_g^t + 1$.
        $r^t \leftarrow r^t + (C_i^L - C_i)$.
      end if
    end for
    Set $\{n_g^{t-len}, n_g^{t-len+1}, ..., n_g^{t-1}\} \cap n_g^t$ as the input of LSTM.
    Get $\hat{n_g}^{t+1}$ as the output of LSTM.
    $x_g^{t+1} = \textbf{argmax}_{x_g} C_{total}^g(x_g) \leftarrow \hat{n_g}^{t+1}$;
    $r^\tau \leftarrow r^\tau + r^t$.
  end for
  if $\mathcal{X}_{max}^{t+1} - \mathcal{X}_{min}^{t+1} > \delta$ **then**
    $s^{\tau+1} \leftarrow \mathcal{X}^{t+1}$
    Store $(s^\tau, a^\tau, r^\tau, s^{\tau+1})$ to D3QN experience pool.
    Set $s^{\tau+1}$ as the state input of online network.
    Select $a^{\tau+1} \in A$ via ϵ-greedy policy.
    Regroup servers with action $a^{\tau+1}$.
    if $(\tau+1)\%\phi = 0$ **then**
      Update θ based on D3QN experience pool.
      $\theta^- \leftarrow \theta$.
    end if
    $\tau = \tau + 1$
  end if
  $t = t + 1$
end while

When the MESs are load unbalanced, $\mathbf{X}^\tau$ is sent as the current state s^τ to D3QN, which will make action a^τ to regroup the MESs. After regrouping, we can obtain the next state value $s^{\tau+1}$ and collect the reward value r^τ. Then, we collect s^τ, a^τ, $s^{\tau+1}$ and r^τ and store them in the experience pool to train the network parameters.

The parameter training process is as follows. We first enter the state s^τ and the action a^τ into the online network to get the estimated action-value as

$$Q(s^\tau, a^\tau; \eta, \alpha, \beta) = V(s^\tau; \eta, \beta) + B(s^\tau, a^\tau; \eta, \omega), \tag{30}$$

where $V(s^\tau; \eta, \beta)$ is the value function, and its output is a scalar. $B(s^\tau, a^\tau; \eta, \omega)$ is an advantage function, which outputs a vector whose length is equal to the size of the action space. η is the input layer and hidden layer parameters, and ω and β are the output layer parameters of the value function and advantage function, respectively. For simplicity of description, we use θ to represent all the parameters of the online network, i.e.,

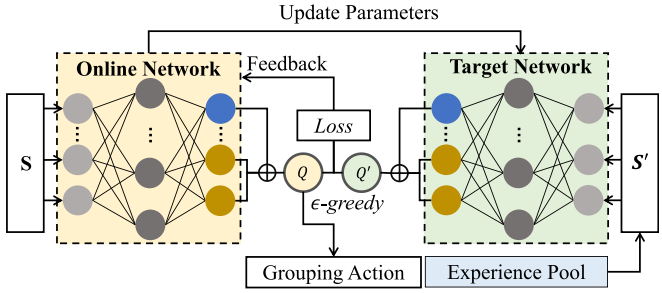


Fig. 10. The architecture of D3QN-based *SeG* algorithm.

$$\theta = \langle \eta, \omega, \beta \rangle. \tag{31}$$

Correspondingly, we use θ^- to represent the parameters of the target network. We input $s^{\tau+1}$ into the online network to get the $argmax_a\{s^\tau, a; \theta\}$ with the largest action-value, and then we can get the target action-value as

$$Q^{target} = r^\tau + \gamma \hat{Q}\big(s^{\tau+1}, argmax_a\{s^\tau, a; \theta\}; \theta^-\big). \tag{32}$$

Equation (32) shows that D3QN uses different functions to select and evaluate an action, which can avoid the over-estimation problem in the original DQN. Based on (30) and (32), we can get the loss function of online network parameter as

$$L(\theta) = \big(Q^{target} - Q(s^\tau, a^\tau; \theta)\big)^2. \tag{33}$$

After obtaining the loss function, we can update the parameter θ in the online network through the gradient back propagation of the neural network. In addition, we need to replace the parameter θ^- in the target network with the parameter θ in online network at regular intervals. This method of asynchronous update of the two networks reduces the correlation between target action-value and estimated action-value, which is conducive to accelerate network convergence. Combining the proposed solutions of the sub-problems, we summarize the details of *MOTO* in *Algorithm 1*.

Algorithm Complexity Analysis. The *MOTO* scheme includes two major stages: 1) offline model training and 2) online decision. In the first stage, the system periodically takes a portion of data from the experience pool and the system costs consist of two components, i.e., the LSTM-based algorithm for task offloading control (*ToC*) and the D3QN-based algorithm for server grouping (*SeG*). For the *ToC*, the prediction step is 1 and each step is 1 *min*, which needs len iterations in total. Denote the termination threshold of Newton's method by ϕ, so the time cost for solving task offloading probability is $O(\frac{1}{\phi})$. Thus, the time cost of *ToC* is $O(len + \frac{1}{\phi})$. For the *SeG*, let T be the number of time steps in each episode and Z denote the number of episodes, then the cost of the D3QN-based algorithm is $O(TZ)$. In the second stage, we assume the number of users and MESs are n and m, respectively. The time costs for *ToC* and *SeG* are within $O(log(\frac{n}{m}))$ and $O(m)$, respectively. Overall, the proposed model can be adaptive in an online manner with low time complexity.

6 Performance Evaluation

In this section, we conduct extensive data-driven experiments to evaluate the performance of *MOTO*. Specifically,

TABLE 3
Network Association Record Description

Field	Value
Association ID	241974033
AP ID	1030
AP Name	MH-D2CY-2F-13
Building Name	MH-D2CY
Client MAC Address	00:00:xx:xx:EF:52
Connection Time	2019-05-08T18:30:31+08:00
Disconnection Time	2019-05-08T18:40:31+08:00
Number of APs: 4045	Number of Records: 29,284,966
Number of Users: 21,725	Time Span: Apr. 26th to Jun. 6th 2019

TABLE 4
Experimental Parameters

Parameters	Value
Task size (ρ)	2×10^3 Bits
CPU cycles required to process a task (l)	4×10^6 Cycles
CPU frequency of MDs (f_i)	2 GHz
CPU frequency of MESs (f_j)	16×3 GHz
Task arrive rate of MDs (λ_i)	[0.05 - 0.15] Tasks/ms
Energy efficiency parameter (k)	1×10^{-27}
Transmission power of MDs (p_i)	1 W
Weights of delay (α)	0.8
Maximum load gap (δ)	0.1
Exploration rate (ϵ)	[0.1, 0.4]
Learning rate (lr)	1×10^{-4}
Reward discount factor (γ)	0.7
D3QN experience pool capacity	1×2^{19}
Batch size	1×2^{10}
Optimizer	Adam
Activation function	Relu

we first elaborate on the evaluation methodology with experiment setup, benchmark strategies design, and metrics definition. Then, we respectively compare the overall performance, the performance of LSTM-based *ToC* algorithm, and the performance of D3QN-based *SeG* algorithm, to evaluate the impact of different parameters.

6.1 Methodology

6.1.1 Experiment Setup

We implement *MOTO* in Python based on Tensorflow, which is an open-source machine learning framework. The experiments are carried out on a server with 4 CPUs each containing 16 Intel(R) Xeon(R) Platinum 8260 CPU @2.40GHz with 24 cores, and one graphics processing unit (NVIDIA Tian V GPU) is used to accelerate the training process. For performance evaluation, we adopt 42 days large-scale real world data samples, i.e., from Apr. 26th, 2019 to Jun. 6th 2019. Table 3 shows an example of an association record, containing the association ID (the unique identification of record), the associated AP ID and name (the unique identification of AP, identifying the campus, building, and floor), the building name where the AP is located, client MAC address (the unique identification of mobile user), connection time, and disconnection time. In particular, we randomly select 35 days and use the data samples as the training data set, and the remaining 7 days data samples are the testing data set. The detailed experimental parameters are shown in Table 4.

6.1.2 Benchmarks

To demonstrate the performance of our proposed *MOTO*, we adopt and implement the following comparable benchmarks strategies for LSTM-based *ToC* algorithm and D3QN-based *SeG* algorithm, respectively.

(1) For LSTM-based ToC algorithm:

- *No offloading (NO)*: in this strategy, all tasks are computed locally.
- *Random offloading (RO)*: users can randomly select local or edge computing.
- *Free offloading (FO)*: users choose local or edge computing with lower total cost.

(2) For D3QN-based SeG algorithm:

- *All group (AG)*: all MESs in the system are in the same group.
- *No group (NG)*: each MES independently provides services for MDs.
- *Max-Min Group (MG)*: this strategy firstly groups the MESs with the heaviest load and the lightest load, and then the grouping rule is repeated until no more groups can be formed.

For the overall performance comparison, we choose the combination of *FO* and three benchmarks for D3QN-based algorithm, i.e., *AG+FO*, *NG+FO*, and *MG+FO*. The reason is that *FO* can always outperform *NO* and *RO*, which will be described in Section 6.3.

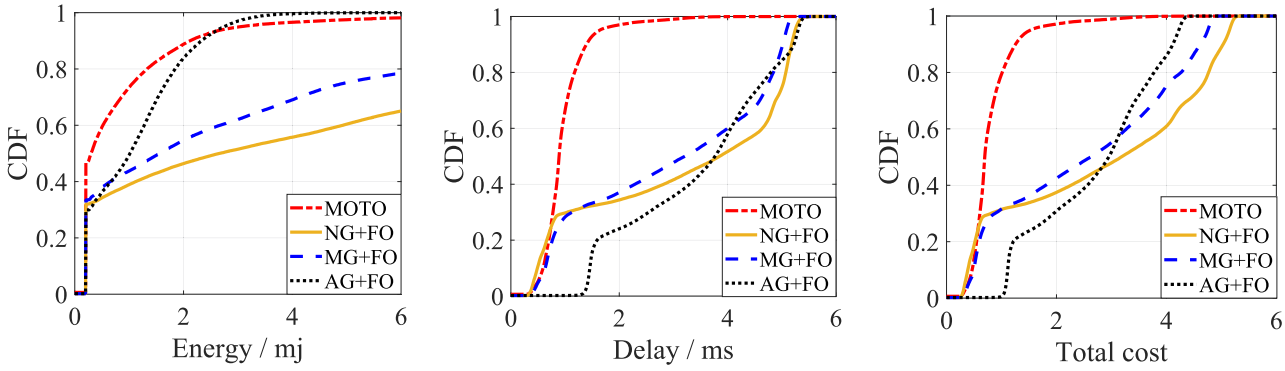
6.1.3 Performance Metrics

The following three metrics are defined to evaluate the offloading control performance with load balancing.

- *Energy Cost (Energy)*: the sum of energy consumption on local computing and transmission when offloading tasks to the edge server.
- *Delay Cost (Delay)*: the sum of delay on transmission and computation.
- *Total Cost*: the weighted cost of energy cost and delay cost.

6.2 Overall Performance Comparison

We first carry out the overall performance comparison between *MOTO* and other benchmark strategies. Figs. 11a, 11b, and 11c show the CDFs of system energy, delay, and total cost achieved by different strategies, respectively. We have the following three major observations. First, *MOTO* can outperform other benchmarks significantly on all metrics. Particularly, the probabilities that the energy cost is lower than 2 *mj* are about 0.88, 0.45, 0.57, and 0.24 in *MOTO*, *NG+FO MG+FO*, and *AG+FO*, respectively. This is because *MOTO* scheme comprehensively considers task offloading control and server grouping with adaptive load balancing, and both modules are coupled to each other. Instead, other baselines are static in load balancing and cannot adjust computing strategy dynamically and adaptively according to the actual system status. Second, we can observe that given the percentile of 80%, the delay costs of *NG+FO*, *MG+FO* and *AG+FO* reach about 5*ms*, 4.7*ms* and 4.7*ms*, respectively, while the *MOTO* scheme only



(a) CDFs of energy cost (b) CDFs of delay (c) CDFs of total cost

Fig. 11. The overall system performance.

consumes 1.3*ms*, reducing the delay cost by 74%, 72.3%, and 72.3% compared to other three benchmarks. Third, since the delay has a higher weight than the energy, the results of total cost show similar trends with delay, and the *MOTO* scheme can achieve the least cost in most cases. Finally, we also find that *AG+FO* has lower energy cost than other baselines but performs worse on delay metric. This is because the system achieves the best load balancing with full edge resource utilization when all servers belong to the same group. In this case, more MDs prefer to offload tasks. However, searching for an optimal MES for MD task offloading leads to high delay costs under *AG+FO* scheme. The reason is that users choose local or edge computing with the lower total cost and all MESs are in the same group with large size. Different from the *AG+FO* scheme, the searching space of our proposed *MOTO* scheme is within a group and the server grouping strategy will avoid forming large-size MES groups. Therefore, the searching delay is very small (i.e., about 1/1000 of the average task execution time) and can be negligible.

Then we evaluate the temporal performance of *MOTO* scheme, i.e., how it performs at different time periods. Fig. 12 shows the temporal box-plots of total cost, where four time periods (i.e., 8:00-11:00, 11:00-14:00, 14:00-17:00, 17:00-20:00) are considered. Under all temporal zones, *MOTO* can achieve a significantly better performance in terms of total cost. By observing the 25^{th}, 50^{th}, 75^{th}, and 100^{th} percentiles, the total costs of the benchmark schemes have large deviations. On the other hand, the performance of *MOTO* varies less than other schemes with different percentiles and with time evolving, demonstrating the stability and adaptiveness of *MOTO*. It is also worth noting that the gaps between the upper and lower quartile position of the four schemes are relatively large during 8:00-11:00 and 11:00-14:00. This is because the users' mobility in these periods are much larger than that in 14:00-17:00 and 17:00-20:00.

Moreover, when observing the performance of benchmarks, the total cost box-plot gap of *AG+FO* scheme is particularly large. The reason is that *AG+FO* requires all users in the same server group, leading to increasing computation costs for all users when resources are insufficient. For *NG+FO* and *MG+FO* schemes, due to the large number of server groups, the user surge for some servers only affects a part of users, and thus the impact on the overall user computation cost is smaller compared to *AG+FO*.

6.3 Performance of LSTM-Based *ToC* Algorithm

In this subsection, we examine the performance of the LSTM-based algorithm for *ToC* sub-problem, considering that each MES independently provides services for MDs. We first evaluate the total cost of different task offloading schemes at different time periods, as shown in Fig. 13. We can observe that the proposed LSTM-based *ToC* algorithm can outperform other baselines with an obvious performance gap. Moreover, *NO* scheme achieves the highest cost and remains the same value at all time intervals since all tasks are computed locally. *RO* scheme allows mobile devices to randomly offload tasks, but its cost is still high. The reason is that the proportion of offloading tasks is low.

Then, we compare total cost with varying weight of delay α, as shown in Fig. 14. Regardless of the α variation, the proposed LSTM-based *ToC* algorithm can outperform other

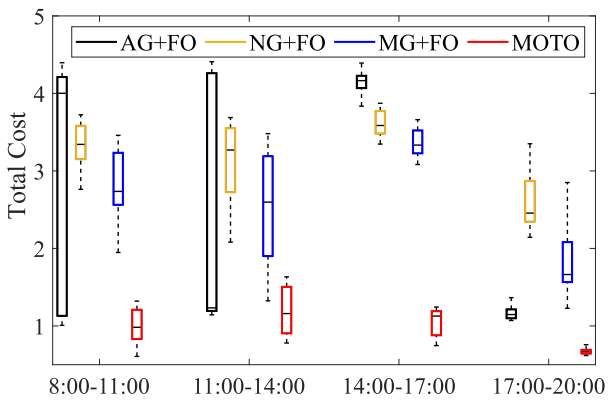


Fig. 12. The total cost versus time.

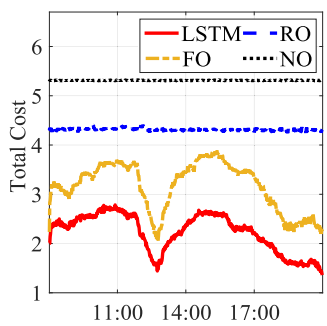


Fig. 13. The offloading cost versus time.

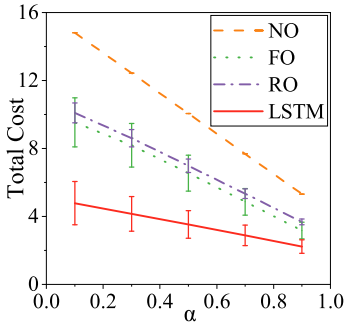
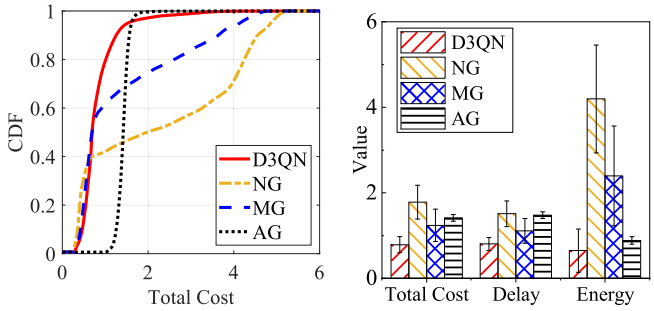


Fig. 14. Impact of different α on the total cost.



(a) CDFs of total cost of server grouping schemes (b) Average performance of server grouping schemes

Fig. 16. Performance of D3QN-based *SeG* algorithm.

baselines with an obvious performance gap. Moreover, the total costs decrease with the increasing value of α, as the delay cost is much larger than the energy cost. Furthermore, we plot the average total cost, delay cost, and energy cost by adopting different values of α in Fig. 15. We can observe that LSTM-based *ToC* algorithm can outperform all the benchmarks under the total cost and delay cost. For the energy cost, LSTM-based *ToC* algorithm achieves higher score than *FO* scheme when $\alpha = 0.9$ and approximately the same score when $\alpha = 0.1$. The reason is that *FO* scheme prefers to offload as many tasks as possible to edge servers, which can reduce the energy cost of MDs. However, *FO* scheme can lead to overloading of edge servers and increase system delay, which can be verified in Fig. 15b. In addition, the standard deviation of *NO* scheme is almost 0 since all computation tasks are computed locally, which leads to the same computation costs for all MDs.

6.4 Performance of D3QN-Based *SeG* Algorithm

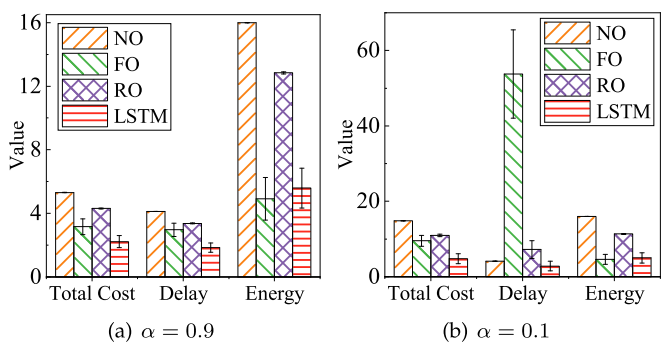
We then verify the performance of D3QN-based *SeG* algorithm. Note that for fairness, LSTM-based *ToC* algorithm is adopted for all benchmarks. We first plot the CDFs of total cost in Fig. 16a, and it can be seen that the proposed D3QN-based *SeG* algorithm can achieve the superior performance. Particularly, given the percentile of 80%, the costs of *AG*, *MG*, and *NG* reach about 1.7, 3, and 4.1, respectively, while the D3QN-based *SeG* algorithm only achieves a score of 0.9, reducing the total cost by 47%, 70%, and 78%, respectively. Fig. 16b shows the average performance of three metrics with error bars, and we have the following observations. For all metrics, D3QN-based *SeG* algorithm can significantly outperform other benchmarks. For instance, the average energy costs for D3QN-based *SeG* algorithm, *NG*, *MG* and *AG* are about 0.8, 4.2, 2.3, and 1 respectively. That means D3QN-based *SeG* algorithm reduces the energy cost by about 80.9%, 65.2%, and 25% compared with other benchmarks.

We next investigate the buffer size (number of tasks in MES buffer) of all MESs without (Red) and with (Blue) load balancing. As shown in Fig. 17, with grouping-based load balancing, the standard deviation of MES load becomes smaller, which shows the effectiveness of the load balancing algorithm. In addition, we can find that the average load of all MESs has increased after load balancing. This is because MESs can handle more tasks and their computing resources are more effectively utilized after load balancing.

6.5 Impact of α and σ on System Performance

In this subsection, we investigate the impact of weight of delay α and the maximum load gap δ on the performance of *MOTO*.

In Fig. 18a, we plot the average results of total cost, delay cost, and energy cost with varying weights of delay α, while fixing the maximum load gap δ to 0.2. With increasing value of α, compared with the energy cost, the delay cost has a more predominant impact on the total cost. In this case, *MOTO* focuses more on minimizing the delay when making task offloading decisions. Therefore, the total cost and delay cost decrease and energy cost increases with a larger α. Specifically, when α increases from 0.1 to 0.3, the total cost and delay cost can decrease from 4.7 and 2.9 to 4.1 and 1.9, respectively.



(a) $\alpha = 0.9$ (b) $\alpha = 0.1$

Fig. 15. Average performance of LSTM-based *ToC* algorithm.

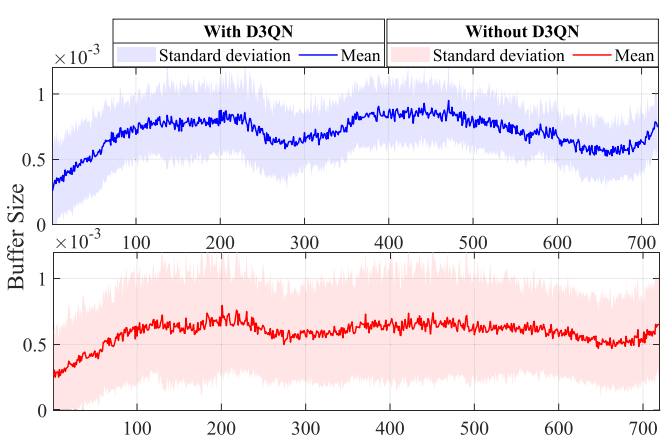
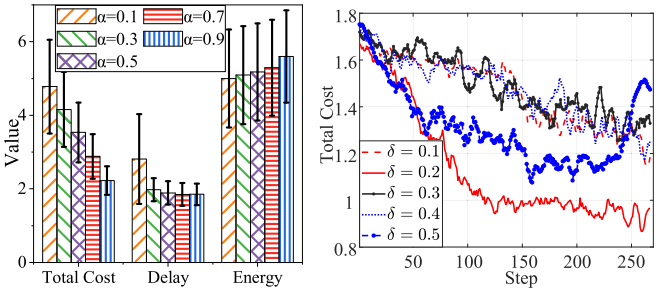


Fig. 17. Mean and standard deviation of MESs buffer size with and without load balancing.



(a) Impact of weight of delay α (b) Impact of maximum load gap δ

Fig. 18. Average performance of *MOTO*.

As shown in Fig. 18b, we plot the convergence curves of total cost with different values of δ, while fixing the weight of delay to 0.9. All curves are smoothed for clear results. We have the following two major observations. First, *MOTO* can achieve the best convergence performance when the $\delta = 0.2$. Second, although *MOTO* also converges fast when $\delta = 0.5$, the curve increases again in the last several training steps. The reason is that a larger value of δ indicates a higher load gap tolerance. In this case, although the algorithm can find the load balancing point easily, it cannot guarantee that the system is stable in the balanced state, thus resulting in some fluctuation.

7 Related Work

We review the related works in two categories, i.e., mobile task offloading and load balancing in MEC.

7.1 Mobile Task Offloading in MEC

To achieve efficient task computation in MEC, mobile task offloading has attracted much research attention recently [9], [11], [12], [14], [15], [16], [17], [24], [39], [40], [41]. For example, Yang et al. [14] considered the interaction of the interests between small cells and mobile devices. They proposed a distributed computation offloading method in a multi-server and multi-device system. To minimize the total energy consumption of user equipment subject to minimum overall throughput of the hierarchical small cell mobile-edge network, the authors of [15] proposed an alternating direction method of multipliers (ADMM)-based distributed offloading algorithm. Huang et al. [16] focused on joint optimization of computation offloading and interference coordination for small-cell MEC. Thus, they proposed a distributed DRL scheme with the objective of minimizing the overall energy consumption while ensuring the latency requirements. Yang et al. [17] devised a multi-task learning approach to solve the computation offloading problems in a multi-access edge computing network. Tang et al. [24] designed a model-free DRL-based distributed algorithm, which can determine the offloading decision without knowing the task models and offloading decision of other devices. Zhou et al. [41] studied the task offloading strategies for computing task selection to maximize effective rewards in uncertain and stochastic 5G small cell networks. Some recent works focused on joint task offloading and resource allocation in MEC [10], [13], [18], [27], [28], [29], [30]. Specifically, Hu et al. [10] studied the computing offloading and resource allocation problems in a MEC-enabled IoT network that supports both mobility and energy harvesting. Jiang et al. [27] proposed an online joint offloading and resource allocation framework under the long-term MEC energy constraint to guarantee the end-user quality of experience. The authors of [30] proposed a utility-based approach to maximize users' quality of experience through jointly optimizing service selection, computation resource allocation, and task offloading decisions. Although these research works proposed effective mobile task offloading methods in MEC, the consideration of user mobility issues and load balancing is still lacking in those MEC offloading strategies. Different from the previous works, in this paper, we design a mobility-aware online task offloading control strategy in a more practical scenario with the consideration of load balancing in the small-cell MEC system.

7.2 Load Balancing in MEC

Load balancing in MEC refers to efficiently distributing mobile computation tasks across a set of edge servers [18], [19], [20], [21], [22], [23], [25], [26]. This technique can avoid unevenly overloading and idle conditions among edge servers and achieve overall computation balance. For example, Li et al. [18] studied an optimization problem to minimize the weighted sum of the total delay and energy consumption of all MDs in the MEC network, which considered multi-dimensional optimization on offloading strategy making, load balancing, computation resource allocation and transmit power control. In [19], the authors proposed a distributed coalition-based algorithm and an incentive algorithm based on DRL, solving the load balance problem in the vehicle-to-vehicle computation offloading problem. To fulfill the communication balancing requirements from IoT networks and the computation balancing requirements from edge servers, Liu et al. [20] proposed a dynamic clustering solution using the DRL-based approach. The authors of [21] proposed a mobility load balancing algorithm for small cell networks by adapting network load status and considering load estimation. Zhang et al. [25] studied parallel offloading and load balancing with multiple cooperative MEC servers and massive delay-sensitive execution workloads. The authors proposed a Lyapunov-based centralized cost management algorithm to maximize the computation efficiency by load balancing. Yang et al. [26] and Wu et al. [42] focused on UAV-enabled MEC. To achieve load balancing and optimal UAV caching for UAVs, the authors developed a deep learning-based algorithm for task scheduling and real-time decision-making. Despite the extensive work, the uneven spatio-temporal load issue and adaptive load balancing in small-cell MEC have not been well studied, thus limiting the system's performance. In this paper, we design a joint task offloading control and load balancing framework to achieve better service provisioning and resource utilization in small-cell MEC.

8 Conclusion and Future Work

In this paper, we have investigated mobility-aware online task offloading with adaptive load balancing in a small-cell MEC system. As the formulated *TOO* problem is intractable directly without knowing the dynamics of mobile users and the computation loads distribution on edge servers, we have proposed *MOTO* to jointly optimize the task offloading and MES

grouping in an online manner. Specifically, we firstpredict the task arrival rates and derive the offloading probabilities based on the LSTM-based approach. Then, a D3QN-based MES grouping approach is devised to achieve load balancing. With the proposed *MOTO* scheme, the spatially and temporally uneven user demands can be effectively managed and utilized for resource-efficient task computation, which can further provide valuable insights on service provisioning in mobile networks. Besides, the problem formulation and optimization process in this work can provide a theoretical basis for future studies related to task offloading in small-cell MEC systems. For future work, we will integrate digital twin techniques to better characterize user dynamics and optimize the deep learning model to further improve the user behavior prediction to facilitate adaptive computing offloading.

References

[1] F. Lyu et al., "Mobility-aware computation offloading with adaptive load balancing in small-cell MEC," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 4330–4335.

[2] L. Zhong et al., "A multi-user cost-efficient crowd-assisted VR content delivery solution in 5G-and-beyond heterogeneous networks," *IEEE Trans. Mobile Comput.*, early access, Mar. 24, 2022, doi: 10.1109/TMC.2022.3162147.

[3] J. Lin, P. Yang, N. Zhang, F. Lyu, X. Chen, and L. Yu, "Low-latency edge video analytics for on-road perception of autonomous ground vehicles," *IEEE Trans. Ind. Informat.*, early access, Jun. 13, 2022, doi: 10.1109/TII.2022.3181986.

[4] F. Lyu et al., "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2586–2602, Jun. 2020.

[5] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.

[6] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-empowered future IoV with enhanced communication, computing, and caching," *Proc. IEEE*, vol. 108, no. 2, pp. 274–291, Feb. 2020.

[7] H. Wu, J. Chen, C. Zhou, J. Li, and X. Shen, "Learning-based joint resource slicing and scheduling in space-terrestrial integrated vehicular networks," *J. Commun. Inf. Netw.*, vol. 6, no. 3, pp. 208–223, 2021.

[8] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.

[9] S. Thananjeyan, C. A. Chan, E. Wong, and A. Nirmalathas, "Mobility-aware energy optimization in hosts selection for computation offloading in multi-access edge computing," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1056–1065, 2020.

[10] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17 541–17 556, Dec. 2021.

[11] Z. Jing, Q. Yang, Y. Wu, M. Qin, K. S. Kwak, and X. Wang, "Adaptive cooperative task offloading for energy-efficient small cell MEC networks," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2022, pp. 292–297.

[12] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[13] Y. Qian, J. Xu, S. Zhu, W. Xu, L. Fan, and G. K. Karagiannidis, "Learning to optimize resource assignment for task offloading in mobile edge computing," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1303–1307, Jun. 2022.

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

[15] L. Yang, S. Guo, L. Yi, Q. Wang, and Y. Yang, "NOSCM: A novel offloading strategy for NOMA-enabled hierarchical small cell mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8107–8118, May 2021.

[16] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Trans. Veh. Technol*, vol. 70, no. 9, pp. 9282–9293, Sep. 2021.

[17] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[18] S. Li, J. Du, D. Zhai, X. Chu, and F. R. Yu, "Task offloading, load balancing, and resource allocation in MEC networks," *IET Commun.*, vol. 14, no. 9, pp. 1451–1458, 2020.

[19] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11 994–12 013, Aug. 2022.

[20] Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in IoT edge systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1491–1502, Jun. 2022.

[21] M. M. Hasan, S. Kwon, and J.-H. Na, "Adaptive mobility load balancing algorithm for LTE small-cell networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2205–2217, Apr. 2018.

[22] J. Hu, H. Zhang, Y. Liu, X. Li, and H. Ji, "An intelligent UAV deployment scheme for load balance in small cell networks using machine learning," in *Proc. IEEE Wirel. Commun. Netw. Conf.*, 2019, pp. 1–6.

[23] M. Javad-Kalbasi and S. Valaee, "Energy and spectrum efficient user association for backhaul load balancing in small cell networks," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[24] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022.

[25] W. Zhang, G. Zhang, and S. Mao, "Joint parallel offloading and load balancing for cooperative-MEC system with delay constraints," *IEEE Trans. Veh. Technol*, vol. 71, no. 4, pp. 4249–4263, Apr. 2022.

[26] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, Aug. 2020.

[27] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, early access, Feb. 11, 2022, doi: 10.1109/TMC.2022.3150432.

[28] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960–1972, Mar. 2022.

[29] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[30] W. Chu, P. Yu, Z. Yu, J. C. Lui, and Y. Lin, "Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach," *IEEE Trans. Mobile Comput.*, early access, Feb. 18, 2022, doi: 10.1109/TMC.2022.3152493.

[31] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: A survey," *Surv. Operations Res. Manage. Sci.*, vol. 17, no. 2, pp. 97–106, 2012.

[32] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, "Branching and bounds tightening techniques for non-convex MINLP," *Optim. Methods Softw.*, vol. 24, no. 4-5, pp. 597–634, 2009.

[33] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach," *Math. Res.*, vol. 35, pp. 376–386, 1987.

[34] W. Wei, H. Gu, and B. Li, "Congestion control: A renaissance with machine learning," *IEEE Netw.*, vol. 35, no. 4, pp. 262–269, Jul./Aug. 2021.

[35] S. Duan et al., "Multitype highway mobility analytics for efficient learning model design: A case of station traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19 484–19 496, Oct. 2022.

[36] J. J. Moré and D. C. Sorensen, "Newton's method," Argonne National Lab., IL USA, Tech. Rep. ANL-82-8, 1982.

[37] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.

[38] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.
[39] S. Yue et al., "TODG: Distributed task offloading with delay guarantees for edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1650–1665, Jul. 2022.
[40] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in MEC-cloud system," *IEEE Trans. Mobile Comput.*, early access, Oct. 11, 2021, doi: 10.1109/TMC.2021.3119200.
[41] R. Zhou et al., "Online task offloading for 5G small cell networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2103–2115, Jun. 2022.
[42] H. Wu, F. Lyu, C. Zhou, J. Chen, L. Wang, and X. Shen, "Optimal UAV caching and trajectory in aerial-assisted vehicular networks: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2783–2797, Dec. 2020.

Sijing Duan (Student Member, IEEE) is currently working toward the PhD degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Her research interests include Internet-of-Things, mobile edge computing, data mining, and data-driven application design.

Feng Lyu (Senior Member, IEEE) received the BS degree in software engineering from Central South University, Changsha, China, in 2013 and the PhD degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018. During respective Sept. 2018-Dec. 2019 and Oct. 2016-Oct. 2017, he worked as a postdoctoral fellow and was a visiting PhD student in BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently a professor with the School of Computer Science and Engineering, Central South University, Changsha, China. His research interests include vehicular networks, beyond 5G networks, Big Data measurement and application design, and edge computing. He is the recipient of the Best Paper Award of IEEE ICC 2019. He currently serves as associate editor for *IEEE Systems Journal* and leading guest editor for *Peer-to-Peer Networking and Applications*, and served as TPC members for many international conferences. He is a member of Communication Society, and Vehicular Technology Society.

Huaqing Wu (Member, IEEE) received the BE and ME degrees from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and 2017, respectively, and the PhD degree from the University of Waterloo, Ontario, Canada, in 2021. She received the prestigious Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship Award in 2021 and worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, MacMaster University, from 2021 to 2022. She is currently an assistant professor with the Department of Electrical and Software Engineering, University of Calgary, Alberta, Canada. Her current research interests include B5G/6G, space-air-ground integrated networks, Internet of vehicles, edge computing/caching, and artificial intelligence (AI) for future networking. She received the Best Paper Award for IEEE GLOBECOM 2018 and *Chinese Journal on Internet of Things* 2020.

Wenxiong Chen (Member, IEEE) received the BS degree in communication engineering and the MS degree in translation from Hunan Normal University, Changsha, China, in 2007 and 2011, respectively, and the PhD degree in management science and engineering from Central South University, Changsha, China, in 2021. He is currently an Associate Researcher with the Research Institute of Languages and Cultures, and associate professor with the College of Information Science and Engineering, Hunan Normal University. His research interests include information management, Big Data measurement and application design.

Huali Lu (Student Member, IEEE) received the MS degree from the College of Computer Science and Electronic Engineering from Hunan University, Changsha, China, in 2020. She is currently working toward the PhD degree with the School of Computer Science and Engineering, Central South University, Changsha, China. Her researches mainly focus on spatial-temporal data mining, compact data collection, and trajectory similarity computing.

Zhe Dong (Student Member, IEEE) received the BSc from the School of Computer Science and Technology, Hainan University, Hainan, China, in 2019 and the MSc degree from the School of Computer Science and Engineering, Central South University, Changsha, China, in 2022. His research interests include edge computing, reinforcement learning, and wireless communication.

Xuemin Shen (Fellow, IEEE) received the PhD degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is a University professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular networks. He is a registered professional engineer of Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, a Chinese Academy of Engineering foreign member, and a distinguished lecturer of the IEEE Vehicular Technology Society and Communications Society. He received the Canadian Award for Telecommunications Research from the Canadian Society of Information Theory (CSIT) in 2021, the R.A. Fessenden Award in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society (ComSoc), and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the technical program committee Cchair/co-chair for IEEE GLOBECOM'16, IEEE INFOCOM'14, IEEE VTC'10 Fall, IEEE GLOBECOM'07, and the chair for the IEEE ComSoc Technical Committee on Wireless Communications. He is the president Elect of the IEEE ComSoc. He was the vice president for Technical & Educational Activities, vice president for Publications, member-at-large on the Board of Governors, chair of the distinguished lecturer Selection Committee, and member of Selection Committee of the ComSoc. He served as the editor-in-chief for *IEEE IoT Journal*, *IEEE Network*, and *IET Communications*.