MSM: Mobility-Aware Service Migration for Seamless Provision: A Data-Driven Approach

Wenxiong Chen, *Member, IEEE*, Mingliu Liu, *Member, IEEE*, Fan Wu, *Member, IEEE*, Huaqing Wu, *Member, IEEE*, Yuan Miao, Feng Lyu, *Senior Member, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—Mobile-edge computing (MEC) is a promising approach to support high-quality time-sensitive applications. With the increasing number of mobile devices, achieving efficient service migration management has become nontrivial in MEC. In addition, the service migration issue is difficult to be solved in real time due to user mobility and dynamic network conditions. In this article, we investigate the mobility-aware service migration problem in MEC by introducing a data-driven framework. First, service migration is formulated as an optimization problem for minimizing the long-term system delay that consists of computing, communication, and migration delays. Second, we propose a Mobility-aware Service Migration scheme, named *MSM*, consisting of three layers: 1) the data collection layer; 2) the association patterns analysis layer; and 3) the service migration layer. Specifically, we first collect users' historical Wi-Fi traces to mine the association patterns. We then design a user management mechanism to reduce the complexity of decision making by using user association patterns. Finally, we formulate the service migration as a 2-D-Markov decision process and devise a deep reinforcement learning (DRL)-based algorithm to obtain service migration decisions in a large-scale MEC scenario. Extensive data-driven experiments are conducted to demonstrate the efficacy of *MSM* in reducing the system delay.**

***Index Terms*—Data-driven, mobile-edge computing (MEC), reinforcement learning, service migration, user mobility.**

Manuscript received 20 September 2022; revised 16 January 2023; accepted 25 March 2023. Date of publication 7 April 2023; date of current version 24 August 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62102223 and Grant 62002389; in part by the 111 Project under Grant B18059; in part by the National Key Research and Development Program of China under Grant 2022YFF0604504 and Grant 2022YFC2009805; in part by the 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 Philosophy and Social Science Foundation of Hunan under Grant 18YBQ089; in part by the Research Project on Teaching Reform of Ordinary Colleges and Universities in Hunan Province under Grant HNJG-2020-0156; in part by the Key Scientific and Technological Project of the Higher Education Institutions of Henan Province under Grant 22A520012; in part by the Young Talents Plan of Hunan Province of China under Grant 2021RC3004; and in part by the Philosophy and Social Science Foundation of Hunan under Grant 18YBQ089. *(Corresponding author: Fan Wu.)*

Wenxiong Chen is with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China (e-mail: chenwx@hunnu.edu.cn).

Mingliu Liu is with the State Grid Hubei Electric Power Research Institute, Wuhan 430077, Hubei, China, and also with the School of Electronic Information, Wuhan University, Wuhan 430072, China (e-mail: liumingliu@whu.edu.cn).

Fan Wu, Yuan Miao, and Feng Lyu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: wfwufan@csu.edu.cn; miaoyuancsu@csu.edu.cn; fenglyu@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).

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/JIOT.2023.3265434

I. Introduction

With the rapid development of network technology (e.g., 5G/Wi-Fi6), mobile-edge computing (MEC) has become a promising approach to meet the requirements of emerging delay-sensitive applications [2], such as autonomous driving [3], [4], [5] and intelligent video acceleration [6], [7]. By sinking the computing resources to network edges [8], MEC can provide a low-latency and satisfactory service experience due to the shortened network distance from the nearest edge node. However, service migration issue has become a serious challenge in large-scale MEC scenarios. Fig. 1 shows a user moves from area A to area B at time slot t. If we still place the user's service in the MEC node near area A, the user's perceived latency will be greatly deteriorated due to the increased network distance. Therefore, to maintain the user's seamless experience, we need to make decisions on whether, when, and where to migrate the user's services.

In the large-scale MEC scenario, how to design the service migration policy becomes a serious challenge when considering the service migration problem for multiple users. There are two solutions to address the challenge. The first is designing a service migration policy for each user. When the system arranges service placement for a user, the load task of the associated edge server will increase, and the computing-perceived delay of other users associated with the same edge server will also increase. In addition, when the user moves to a new location, the associated access point (AP) and the communication delay between the user and the original edge server will change. The second solution is designing a centralized service migration policy for all users. In this case, network operators need to collect service request information from all users and make service migration decisions for users at the same time. The multiuser service migration problem is a high-dimensional integer nonlinear programming problem, which is difficult to solve as the number of users increases. Furthermore, the optimal service migration policy decision depends on multidimensional factors, e.g., user

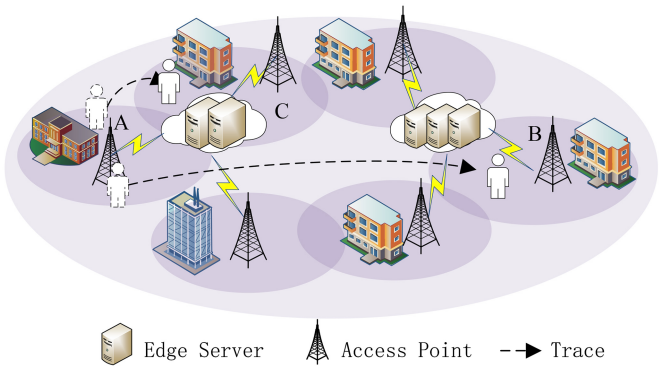

Fig. 1. Users service migration in large-scale MEC scenarios.

mobility, communication channel condition, network load, and available resources on MEC nodes.

Previous works mainly aimed to optimize the service migration by requesting the knowledge of future information [9], [10], [11], [12], [13] or utilizing some methods without any prior knowledge [14], [15], [16]. For schemes which require the prior knowledge of future information, some predictions are generally conducted or assumed that the future information follows a certain mathematical distribution model. For schemes without any prior knowledge, the Lyapunov method has been widely used to decompose the long-term optimization problem into a series of real-time problems. In addition, a large-scale edge cache deployment scheme has been proposed by characterizing the user mobility patterns from a Wi-Fi data set [1], [17]. To mining the user mobility patterns, we further investigate the user mobility characteristics to guide service migration scheme deployment from the collected Wi-Fi traces.

In this article, we devise a data-driven Mobility-aware Service Migration (MSM) scheme to reduce service delay in small-cell MEC scenarios by mining the user mobility patterns. We consider the long-term system delay, which consists of computing delay, communication delay, and migration delay of all users. Specially, MSM is designed with three steps to optimize the service migration (i.e., whether, when, and where to migrate the services) in small-cell MEC scenarios. First, to capture the user association patterns, we collect large-scale association records from a campus Wi-Fi system. The data collection lasts for over three months and the size of the data set is over 8 GB.[1] Second, we mine the user association patterns combined with various metrics to discover user behaviors, such as high dynamics of associations, users' preferences over APs, and association periodicity. Third, we design a user management mechanism and a deep reinforcement learning (DRL)-based algorithm to learn the optimal service migration policy. Specifically, to decrease the complexity of multiuser service migration and reduce the number of service migrations, we propose a user management mechanism consisting of user classification and user grouping. Then, to deal with the unavailability of future information and the high-dimensional action space, we formulate the service migration as a 2-D-Markov decision process (MDP) and design a DRL-based algorithm to explore the dynamic MEC environment by using the historical experience.

In summary, the major contributions of this work are as follows.

1) To minimize the system delay, we model the multiuser service migration as a long-term optimization problem in large-scale MEC scenarios, which considers computing delay, communication delay, and migration delay of all users.
2) We propose an *MSM* scheme to guide service migration design in large-scale MEC scenarios, which manages users efficiently and utilizes a DRL-based algorithm to explore the optimal service migration policy from users' historical experience.
3) We evaluate the performance of MSM with extensive simulations by using the real Wi-Fi traces. Extensive experiments demonstrate the efficacy of MSM on the system delay, which achieves at least 14.6% improvement over other benchmarks.

The remainder of this article is organized as follows. In Section II, we review related works. In Section III, we present the system model and service migration problem formulation. In Section IV, the user association patterns are analyzed from the collected data. In Section V, we introduce user-management mechanism by using the association patterns. In Section VI, we present a DRL-based service migration algorithm. Performance evaluation is carried out in Section VII, and this article is concluded in Section VIII.

II. Related Work

Service placement schemes have been investigated in cloud/edge computing [18], [19], [20], [21], [22], which aimed to reduce service response time, improve resource utilization, balance network loads, etc. Compared with cloud computing, the dynamic service migration across edges should consider user mobility in MEC scenarios. Therefore, the currently proposed service placement methods in cloud computing cannot work well in MEC scenarios.

Prior Knowledge-Based Service Migration: A vital challenge of service migration in MEC systems is the limited knowledge of user mobility. To tackle this issue, some researchers were devoted to predicting future information, such as the cost [9], the distribution of requests [11], the user-centric locations [10], and so on. Another approach to address the unknown future information is making strong assumptions about user mobility or request distribution and other information. Ksentini et al. [12] and Wang et al. [13], [23] modeled the user mobility as 1-D or 2-D random walk, which facilitates the computing of the user's destination location transferring probability. Ouyang et al. [24] made some assumptions on user request distribution and proposed an adaptive user-managed service placement mechanism.

Service Migration Schemes Without Prior Knowledge: Some service migration schemes [25], [26], [27] were designed to work without the request of user mobility knowledge. In addition, Ouyang et al. [14], Ning et al. [15], and Sun et al. [16]

[1]The data set is available at https://github.com/Intelligent-WiFi/DataSet.

proposed schemes to optimize the dynamic service migration problem by using Lyapunov technology. Then, the solutions to optimize service placement are derived in an online scenario. Gao et al. [28] aimed to jointly optimize the access network selection and service placement for MEC. Instead of Lyapunov, they designed an efficient online framework to decompose the long-term optimization problem into a series of one-shot problems. Then, an iteration-based algorithm was proposed to derive a computation efficient solution. Badri et al. [25] proposed a multistage stochastic model to maximize the total QoS of the system in MEC systems. Xu et al. [27] proposed a path-selection algorithm to select the best set of available transferring paths, which can guide service migration in 5G MEC scenarios. Ma et al. [26] proposed a framework that enables efficient live migration of offloading services in MEC scenarios, which provides seamless service with the moving users from the nearest edge node.

RL in MEC Scenarios: DRL methods are widely used in MEC scheme design, such as task offloading [29], [30], [31], [32], [33], [34], [35], resource allocation [29], [36], [37], [38], and edge caching [39], [40], [41], [42], [43], [44]. Particularly, Liu and Cao [31] proposed a DRL-based algorithm to learn the server migration policy based on the observed performance of past server selections, which is combined with a long short-term memory (LSTM)-based neural network. Yuan et al. [34] proposed a multiagent DRL (MADRL) algorithm for vehicular edge computing to maximize the composite system utility in a distributed way. Huang et al. [32] proposed an RL-based online offloading framework for wireless powered MEC to learn the binary offloading decisions from the experience. Qiu et al. [33] designed a DRL-based online computation offloading approach for blockchain-empowered MEC in which both mining tasks and data processing tasks are considered. Liu et al. [45] proposed a vehicle-assisted offloading scheme for vehicle edge computing network. Qiao et al. [43] designed a DRL-based cooperative edge caching scheme for vehicular edge computing. Wang et al. [40] proposed an MADRL-based intelligent video edge caching framework. Wang et al. [23] formulated the service migration problem as MDP and proposed a new algorithm to find the optimal policy for service migration in MEC scenarios.

III. System Model and MSM Design

In this section, we first introduce the considered MEC system model and the service migration issue. We then formulate the issue as a long-term optimization problem and propose MSM scheme to solve it.

A. System Model

In Fig. 2, we consider an edge computing system consisting of N APs and E edge nodes. APs can communicate and transmit data with edge nodes over the network. There are many users in the system, and each user walks and changes positions frequently according to his or her own behavior pattern. The operation mode of the system is the discrete time gap mode, and the large time interval is divided into several same-size short time slots $t \in \mathcal{T} = \{1, 2, 3, \ldots, T\}$.

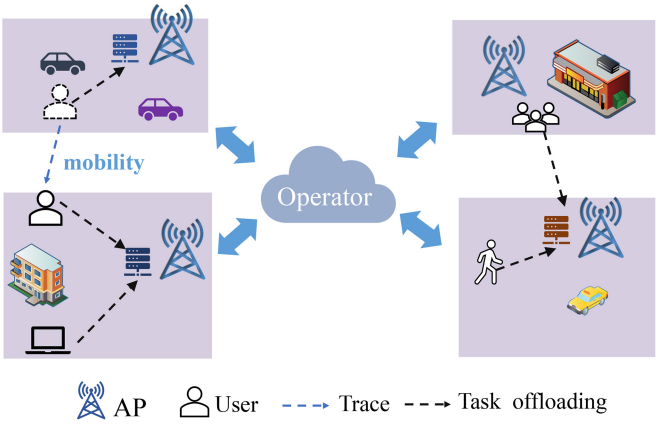

Fig. 2. System model.

TABLE I
List of Main Notations

Notation	Definition
$\mathcal{E}$	Set of edge nodes
$\mathcal{N}$	Set of access points
$\mathcal{U}(t)$	Set of mobile users at time slot t
$x_i^j(t)$	Whether the AP j is selected for user k to access the edge nodes(= 1) or not(= 0)
$y_i^e(t)$	Whether the service of user k is placed on edge node j (= 1) or not(= 0)
$X(t)$	Vector of the AP association variable $x_i^j(t)$
$Y(t)$	Vector of the service placement variable $y_i^e(t)$
R_e	Computing capacity of edge node e
$G_i(t)$	The number of service request produced by user i at time slot t
$\lambda_i(t)$	The total resource demands of user i
$l_j^e(t)$	The communication data rate between edge e and access point j
$f_i^{ke}(t)$	The service of user i migrates from source edge k to destination edge e(=1) or not (=0)
$m_i^{ke}(t)$	The delay caused by user i to migrate service from source edge k to destination edge e at time slot t
C	The total computation delay of users
D	The total communication delay of users
M	The total migration delay of users
$G_i(t)$	The number of services produced by user i
$n_e(t)$	The number of users served by edge e

In each short time slot, we suppose that the user locations and the network scenario will not change. Thus, the AP set, edge node set, and user set in time slot t in the system are denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$, $\mathcal{E} = \{1, 2, \ldots, E\}$, and $\mathcal{U}(t) = \{1, 2, \ldots, |\mathcal{U}(t)|\}$, respectively. There is also a network operator in the system, and at the beginning of each time slot, the user connects to the AP and uploads the service request to the network operator. The network operator collects all the users' requests and selects the optimal edge nodes for them to perform the computing tasks. Table I lists the parameters commonly used in this article.

1) *Service Migration Model:* Let $x_i^j(t)$ and $y_i^e(t)$ denote the dynamic AP association and service placement binary indicators of user i at time slot t, respectively. If user $i \in \mathcal{U}(t)$ associates with AP $j \in \mathcal{N}$ at time slot t, then $x_i^j(t) = 1$; otherwise, $x_i^j(t) = 0$. When the tasks of user i are executed on the edge node $e \in \mathcal{E}$, then $y_i^e(t) = 1$; otherwise, $y_i^e(t) = 0$. Therefore, we define two vectors, i.e.,

$X(t)$ and $Y(t)$, in which $X(t) = [x_1^1(t), \ldots, x_{|\mathcal{U}(t)|}^{|\mathcal{N}|}(t)]$ and $Y(t) = [y_1^1(t), \ldots, y_{|\mathcal{U}(t)|}^{|\mathcal{E}|}(t)]$. In addition, each user can only associate with one AP in the same time. Moreover, we assume that a user's tasks can be executed on only one edge node in the same time slot. Therefore, we can see that the constraints is

$$\sum_{j=1}^{N} x_i^j(t) = 1 \quad \forall i, t \tag{1a}$$

$$\sum_{e=1}^{E} y_i^e(t) = 1 \quad \forall i, t. \tag{1b}$$

When the service of user i is migrated at time slot t, we can see that $\sum_{e=1}^{E} y_i^e(t) y_i^e(t-1) = 0$, where $t > 1$.

2) *QoS Model:* In large-scale MEC scenarios, we consider the QoS model in this article is jointly affected by three major factors, i.e., computing delay, communication delay, and migration delay.

1) Computing Delay: At each time slot t, the user service can be offloaded to any edge node. We take R_e to represent the total computation capacity (i.e., CPU cycles per second) of edge node e. Users served by the same edge node will share the computation resource of the edge. Therefore, for user i, its obtained computation resource is

$$\hat{R}_i(t) = \sum_{e=1}^{|\mathcal{E}|} \frac{R_e y_i^e}{n_e(t)} \tag{2}$$

where $n_e(t) = \sum_{i=1}^{|\mathcal{U}(t)|} y_i^e(t)$ is the number of users served by edge node e. The computing delay of user i at time slot t is calculated as follows:

$$C_i(y_i(t)) = \frac{\lambda_i(t)}{\hat{R}_i(t)} \tag{3}$$

where $\lambda_i(t)$ denotes the resource demands (i.e., the total CPU cycles) of user i at time slot t.

Thus, the overall system computing delay is calculated as follows:

$$C(Y(t)) = \sum_{i=1}^{|\mathcal{U}(t)|} C_i(y_i(t)). \tag{4}$$

2) Communication Delay: It is considered as the transferring delay. Without loss of generality, if the service is executed on the edge server which is attached to the local base station, there is no transferring delay. Otherwise, we need to consider the AP-to-edge delay, which mainly depends on the network distance of transferring path. Let l_j^e denote the unit delay of data transfer from AP j to edge e.

Given the AP association $x_i^j(t)$, service placement $y_i^e(t)$ and the number of service requests $G_i(t)$, we can calculate the communication delay for user i as follows:

$$D_i(y_i(t)) = \sum_{j=1}^{N} \sum_{e=1}^{E} G_i(t) l_j^e(t) x_i^j(t) y_i^e(t). \tag{5}$$

Similarly, given users' AP association vector $X(t)$, service migration vector $Y(t)$, and the service request arrival rate of users, we can calculate the overall communication delay for all services at time slot t as follows:

$$D(Y(t)) = \sum_{i=1}^{|\mathcal{U}(t)|} \sum_{j=1}^{N} \sum_{e=1}^{E} G_i(t) x_i^j(t) y_i^e(t) l_j^e(t). \tag{6}$$

3) Migration Delay: As we know, frequent service migrations will increase the additional migration delay significantly, i.e., the delay of transferring service profiles across edges. $m_i^{ke}(t)$ is the delay of migrating user i's service from source edge k to destination edge e at time slot t. Therefore, the migration delay of user i can be calculated as follows:

$$M_i(y_i(t)) = \sum_{k=1}^{E} \sum_{e=1}^{E} f_i^{ke}(t) m_i^{ke}(t). \tag{7}$$

Moreover, the total migration delay can be computed as follows:

$$M(Y(t)) = \sum_{i=1}^{|\mathcal{U}(t)|} \sum_{k=1}^{E} \sum_{e=1}^{E} f_i^{ke}(t) m_i^{ke}(t) \tag{8}$$

where $f_i^{ke}(t)$ is a binary indicator representing whether user i migrates service from edge k to e at time slot t.

B. Problem Formulation

Given users' AP associations vector $X(t)$, our goal is to find the optimal service migration policy $Y(t)$ to minimize the average user-perceived delay in the long term, which consists of average queuing delay, average communication delay, and average migration delay. Therefore, we can formulate the service migration optimization problem (SMOP) as follows:

$$\min_{Y(t)} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \{C(Y(t)) + D(Y(t)) + M(Y(t))\} \mid_{X(t)}$$
$$\text{s.t. (1b).} \tag{9}$$

The SMOP problem is a sequential decision problem that requires long-term optimization and is time coupled. In this case, the current service migration policy will affect the system performance in future time slots. For example, the service migration decision in the current time slot affects the performance of the system in the next time slot, so it cannot be directly split into individual time slot problems. Therefore, to optimize long-term service migration policy of SMOP, we require complete future system information (e.g., users request distribution or users' mobility). However, the users' future information is difficult to predict accurately in real time. Moreover, when considering the service migration in a large-scale MEC system, SMOP is a high-dimensional nonlinear integer programming problem, which is hard to solve in real time.

C. Architecture of MSM

To address the above challenges, we design a data-driven mobility-aware service migration (MSM) scheme to explore the optimal service migration policy in a large-scale MEC system. In Fig. 3, the workflow of MSM scheme consists of three parts: 1) the collection of user association records; 2) the

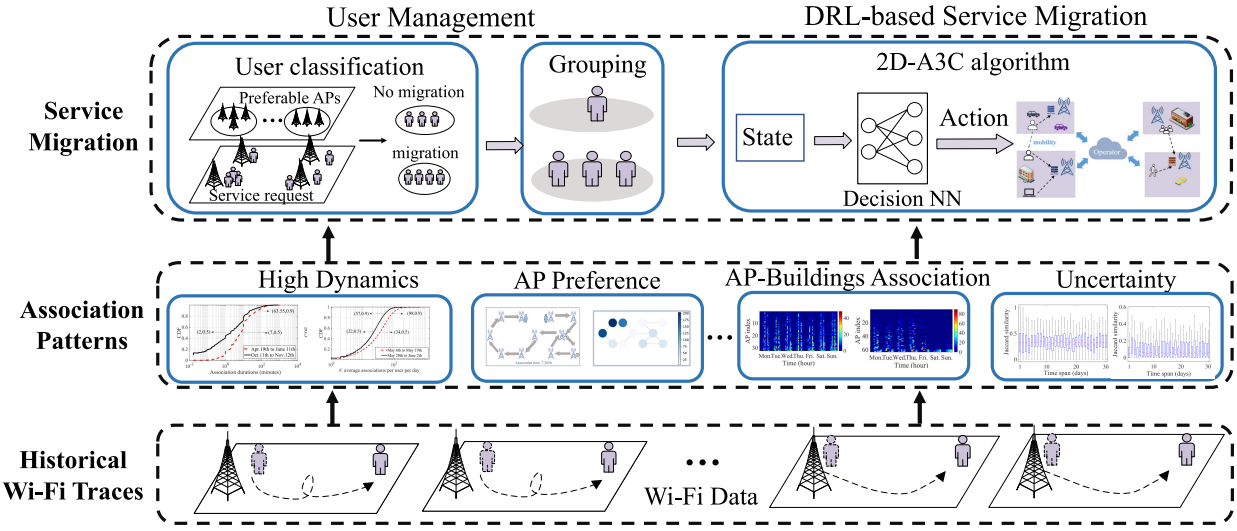

Fig. 3. Design of MSM.

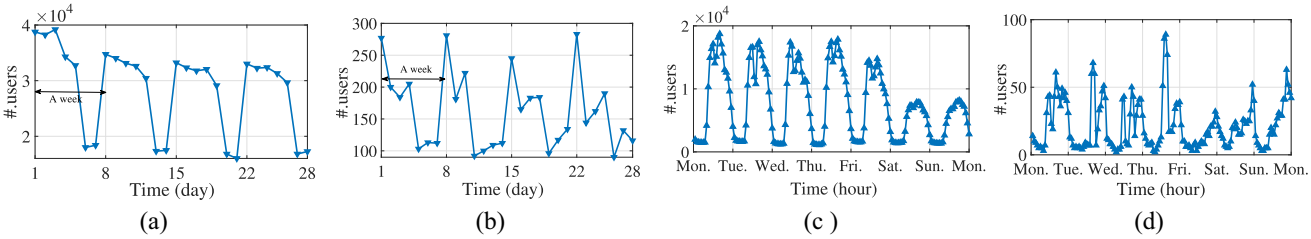

Fig. 4. Number of associated users in different time periods. (a) Whole Wi-Fi system. (b) Randomly selected AP. (c) Whole Wi-Fi system. (d) Randomly selected AP.

analysis of user association patterns; and 3) the design of user management mechanism and DRL-based service migration algorithm. In historical Wi-Fi traces, we have collected a large-scale user association traces from a campus Wi-Fi system, which consists of APs's location, association duration, traffic, users' ID, and connection detail information. In association pattern analysis, we characterize the user activities based on historical user Wi-Fi traces. In user management, we utilize the user classification and user grouping methods to reduce the complexity of service migration decision making. Finally, we utilize the DRL algorithm to explore the optimal service migration policy for the user groups from historical experience, which can further reduce the system delay.

IV. Empirical Study on User Mobility

In this section, we mine the user mobility patterns to guide service migration scheme design by using the collected campus Wi-Fi system data set [17] and data-driven approaches [46]. The used campus Wi-Fi system has about 8000 APs and provides Internet access for 40 000 users. The data set includes total 41 119 940 association records, in which the field consists of asso_id, user_name, client_mac, ap_id, bytes, RSSI, conn_time, and disconn_time.

A. Association Periodicity

We first explore the relationship between the number of associated users and time. In Fig. 4(a) and (b), we plot the number of associated users of all APs and a randomly chosen AP, from 29 April to 3 June, respectively. We can see that the number of associated users varies periodically in different weeks. The system is busier in weekday than in weekends. In Fig. 4(c) and (d), we also plot the number of associated users in terms of hour, from 20 May to 27 May. It is observed that the number of associated users varies periodically by day. On each day, the number is low and stable in the night, and keeps rising between 7:00 A.M. to 16:00 P.M. Starting from 16:00 P.M., the data shows a downward trend all the way.

B. High Dynamics of Associations

1) Frequent Transitions: To understand the user activities in detail, we further study the distributions of user daily associations and durations from the whole. Fig. 5(a) and (b) show the CDFs of the number of daily average associations and association durations, respectively. Note that only when a user moves from an AP and associates to another AP, the number of its associations increases. From above figures, we can get observations as follows: 1) users usually associate to many APs each day as shown in Fig. 5(a), where the mean value (CDF = 0.5) of the number of user daily average associations is 34 in the red curve. Moreover, the users whose daily average associations exceed 98 are not more than 10% and 2) in Fig. 5(b), the duration of associations shows a bipolar distribution. From the red curve, we can observe that median association duration is 7 from 11 April to 11 June, where not less than 10%

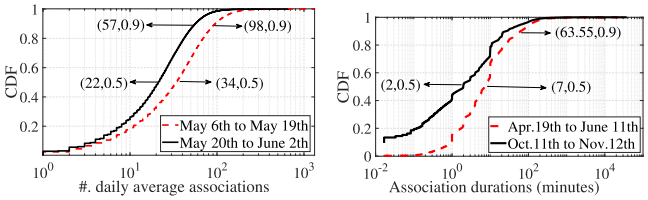

Fig. 5. CDFs of user associations. (a) ♯. associations. (b) Duration (minutes).

associations last for more than 63.55 min. In summary, the short association duration and frequent AP switching dominate the network association behavior of mobile users, i.e., the network association of users is highly dynamic.

2) Uncertainty of Transitions: To observe the transition between days, we compute the Jaccard similarity more clearly. Particularly, let $A_i(t)$ represent the AP set that user i associates to during day t. Therefore, the Jaccard similarity of set $A_i(t)$ and $A_i(t+1)$ can be computed as

$$J(A_i(t), A_i(t+1)) = \frac{A_i(t) \cap A_i(t+1)}{A_i(t) \cup A_i(t+1)}. \tag{10}$$

Fig. 7 shows the box plots of daily Jaccard similarity of user association set. For a special user, its Jaccard similarity at day t is computed by comparing the associated AP sets in two adjacent days t and $t-1$. It is clearly observed that the 75th percent Jaccard similarity is below 0.5 and the median value of Jaccard similarity is about 0.3. This indicates that the user association sets vary differently each day.

Similarly, Fig. 8 shows the box plots of daily Jaccard similarity of user sets associated to each AP. It is observed that the median value is below 0.1, which means that the user sets are significantly different every day. From Figs. 7 and 8, we can conclude that user associations are dynamic with high uncertainty.

C. Users' Preference Over APs

To reveal users' preference over APs, we present Alice's trace and association duration in Figs. 9 and 10, respectively. Fig. 9 shows Alice's association trace (Alice is a randomly selected user) on May 20. We can see that Alice's associations are highly dynamic with frequent AP transitions on May 20. For instance, we can observe that Alice associates with 11 different APs and there are multiple AP transitions between AP1 and AP2. In this case, a user's frequent transitions mean that the user has high mobility.

Fig. 10 shows the cumulative association time distribution of Alice on different APs on May 20, where each point corresponds to the AP at the corresponding position, and the color depth reveals the cumulative association time of the user at that AP. We can observe the polar distribution of cumulative association time from the color of each point, i.e., the cumulative association times of Alice on the three dark APs are no less than 1.5 h, while the average cumulative association time on the other APs is about 14 min, which is consistent with the associated time distribution in Fig. 10. Therefore, although users associate with many APs due to their movement, they obviously prefer some APs and spend most of their time and traffic on these APs. In this work, we measure the user preference over APs by the time spent and the traffic consumed by users from the APs.

D. Spacial–Temporal Regularity of User Associations

In this section, we explore the spacial–temporal regularity of user associations between APs and buildings. We randomly choose two buildings (called "D2CY" and "DXY") to characterize user associations from the associated APs. Fig. 6(a) and (b) show the heatmap of hourly user association in corresponding buildings. For each grid, its value represents the number of average associated users of all APs which locate in the building at the corresponding time. We can see that the busy hours of "D2CY" are from 12:00 P.M. to 1:00 P.M. and from 18:00 P.M. to 19:00 P.M., which is the time for lunch and dinner. It is consistent with D2CY's canteen identity. Similarly, "DXY" is an academic building. In Fig. 6(b), the busy time of "DXY" is concentrated on the daytime of weekday, from 9:00 A.M. to 18:00 P.M., which are working hours.

In Figs. 6 (c) and (d), we plot another heatmap of hourly user association for each AP in the "DXY" and "D2CY", respectively. For each grid, its value represents the number of users associated with an AP at the corresponding time, i.e., each row reflects the change of user associations for a single AP in one week. It is clearly observed that APs located in the same building have similar change rules. Moreover, the APs located in different buildings have significantly different user association patterns.

V. User Management Mechanism

With the above observations, we propose a user management mechanism by mining user association patterns for the SMOP problem. Fig. 11 shows the workflow of user management mechanism in large-scale MEC systems, which mainly consists of four steps: 1) history trace collection; 2) user mobility patterns; 3) user classification; and 4) user grouping. The user mobility patterns step is demonstrated in Section IV. In the user classification step, we only consider migration for users who move to preferable APs, which can reduce the number of service migrations. In the user grouping step, we group the migrated users by using user mobility characteristics.

A. User Classification

With the high mobility of users in frequent AP transitions, the migration delay will greatly increase when the services follow the users, which decrease the system delay significantly in large-scale MEC scenarios. Considering the knowledge of user mobility patterns and users' preferences over APs, we propose a user classification mechanism to find users with similar service migration requirement, which can reduce service migrations.

First, we extract a preferable AP set for each user. Let $\theta_i(j)$ denote the ratio of the association time user i spends on AP j to its overall association time. Therefore, we define the preferable AP set for user i as follows:

$$P_i = \{j | j \in \mathcal{N}, \theta_i(j) > \xi\} \tag{11}$$

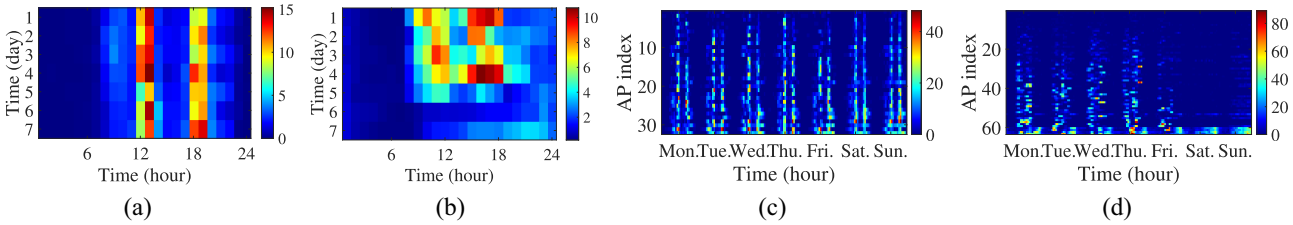

Fig. 6. APs in buildings "D2CY" and "DXY." (a) D2CY. (b) DXY. (c) D2CY. (d) DXY.

Algorithm 1 Lazy User Classification

Input: user locations $X(t)$ at time slot t, user preferable AP sets

Output: user sets $A_1(t), A_2(t)$

for user $i \in U(t)$ **do**
  for AP $j \in \mathcal{N}$ **do**
    if $x_i^j(t) == 1$ **then**
      set $k = j$
  if $\exists j \in \mathcal{N}, x_i^j(t-1) = 1$ **then**
    if $k \in P_i \&\& j! = k$ **then**
      $i \in A_1(t)$, wait for subsequent processing.
    else
      $i \in A_2(t)$, set $y_i(t) = y_i(t-1)$
  else
    if $k \in P_i$ **then**
      $i \in A_1(t)$, wait for subsequent processing.
    else
      $i \in A_2(t)$, the service of user i is placed on the nearest MEC node.
return $A_1(t), A_2(t)$

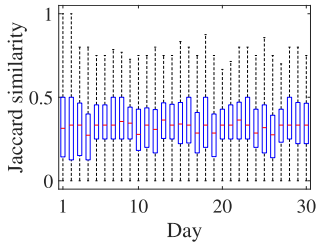

Fig. 7. User association Jaccard similarity.

where ξ is a constant threshold set by referring to the distribution of the user's association time on the AP.

To explore user service migration problem, we then classify users into two categories, i.e., the first category of users needs to consider service migration, and the other category of users does not need to consider service migration. Specifically, we consider that users who have moved to the preferable APs need service migration, because the users may stay in the coverage of the APs for a long time. When users move to the less-preferable APs, it means that the service migration is undesirable due to the short association time and frequent AP transitions. Therefore, we propose an online algorithm for user classification in Algorithm 1.

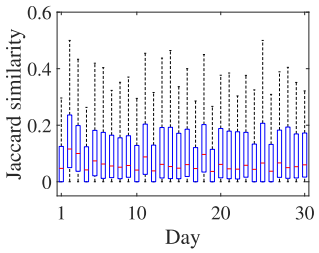

Fig. 8. AP association Jaccard similarity.

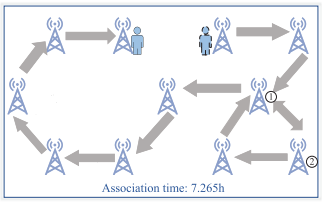

Fig. 9. Alice's trace.

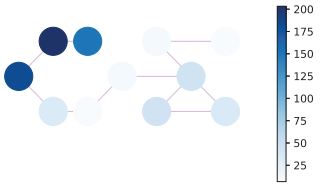

Fig. 10. Alice's association duration.

B. User Grouping

For the large-scale MEC scenario, the number of associated users for each AP may be large and erratic, which is an issue for service migration decision making. To handle the challenge, we group users who are selected to migrate after classification. In other words, the system can make service migration decisions for the user groups to reduce the dimension of the multiuser decisions. Therefore, we can transfer the user service migration problem to the group service migration problem, which can reduce the number of service migrations and improve the system performance in large-scale MEC scenarios.

Generally, each user shows preference for server APs and each AP has its own geographic attributes. From Section IV-D, we can see that user association variations are similar when the APs are from the same building. Otherwise, the user association variations are significantly different for APs from different buildings. These phenomena indicate that APs in the same

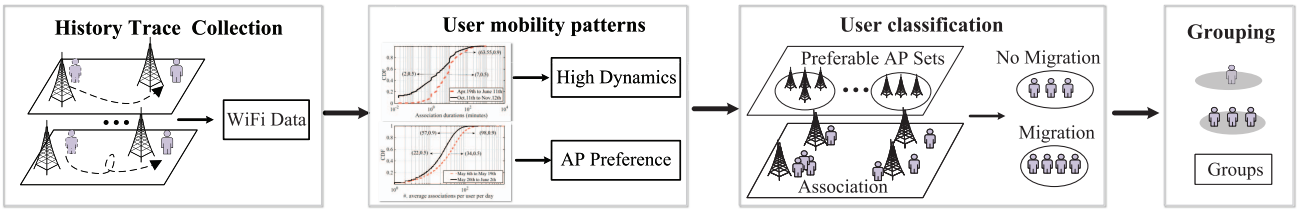

Fig. 11. Workflow of user management mechanism.

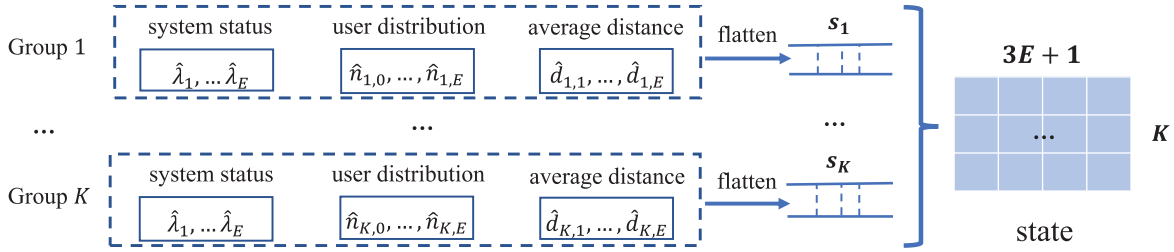

Fig. 12. MDP state.

building share similar user mobility patterns. Combined with the user preference of APs, we group the selected users by using the user association with APs in buildings.

Particularly, we take v_i^n to demonstrate the correlation coefficient between user i and building n. It can be computed as

$$v_i^n = \frac{T_i^n}{T_i} \tag{12}$$

where T_i is the sum of user historical association duration and T_i^n is the time user i spent on the APs in building n. Therefore, user i's correlation coefficient with the whole building set in the system can be described as a vector $V_i = [v_i^1, v_i^2, \ldots, v_i^{N_b}]$, where N_b is the number of buildings. In addition, we have the constraints as follows:

$$\sum_{i=1}^{N_b} v_i = 1$$
$$\text{s.t.} \quad 0 \le v_i^n \le 1. \tag{13}$$

In this article, we adopt the K-means clustering algorithm [47] to group users by measuring the Euclidean distance of the sample to cluster centers, which purpose is to distinguish between different categories of user behavior from the association records.

VI. DRL-Based Service Migration Algorithm

To further improve the system performance, we design a DRL-based service migration algorithm for the group service migration problem in large-scale MEC scenarios.

A. MDP Model

To improve the system performance in a long term, we adopt the DRL algorithm to find the optimal service migration policy in large-scale MEC scenarios. In addition, the DRL algorithm focuses on the impact of current decisions on the future rather than focusing only on immediate rewards, which meets the requirement of the SMOP problem. For the DRL algorithm, we formulate the SMOP problem as an MDP and define the 2-D-state space, action space, and reward function as follows.

1) 2-D-State Space: As shown in Fig. 12, we use $\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t)$, $\hat{\boldsymbol{d}}_{\boldsymbol{k}}(t)$ and $\hat{\boldsymbol{\lambda}}_{\boldsymbol{k}}(t)$ to denote the vectors of the kth group states

$$\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t) = \left[\hat{n}_{k,0}(t), \hat{n}_{k,1}(t), \ldots, \hat{n}_{k,E}(t)\right] \tag{14}$$
$$\hat{\boldsymbol{d}}_{\boldsymbol{k}}(t) = \left[\hat{d}_{k,1}(t), \ldots, \hat{d}_{k,E}(t)\right] \tag{15}$$
$$\hat{\boldsymbol{\lambda}}_{k}(t) = \left[\hat{\lambda}_1(t), \ldots, \hat{\lambda}_E(t)\right] \tag{16}$$

where $\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t)$, $\hat{\boldsymbol{d}}_{\boldsymbol{k}}(t)$, and $\hat{\boldsymbol{\lambda}}_{\boldsymbol{k}}(t)$ denote the number of users in different types, the average network distance between the group and edges, and the size of total requests in each edge node. Specially, we classify the users into $E+1$ types for the state $\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t)$. $\hat{n}_0(t)$ denotes the overall number of users and $\hat{n}_e(t)$ denotes the number of users who run tasks on edge e at time slot $t-1$. In addition, the state $\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t)$ indicates the user distribution and migration information. The state $\boldsymbol{s}_{\boldsymbol{k}}(t)$ of group k at time slot t is flattened into one dimension as follows:

$$\boldsymbol{s}_{\boldsymbol{k}}(t) = \left[\hat{\boldsymbol{n}}_{\boldsymbol{k}}(t), \hat{\boldsymbol{d}}_{\boldsymbol{k}}(t), \hat{\boldsymbol{\lambda}}_{\boldsymbol{k}}(t)\right]. \tag{17}$$

Thus, the system state for each group is flattened into one dimension, and the state $\boldsymbol{s}(t)$ at time slot t is

$$\boldsymbol{s}(t) = [\boldsymbol{s}_{\boldsymbol{1}}(t); \ldots; \boldsymbol{s}_{\boldsymbol{K}}(t)] \tag{18}$$

where K denotes the number of groups.

The state space $\mathcal{S}$ is limited as follows:

$$\mathcal{S} = \left\{ \begin{array}{ll} \hat{n}_{k,e}(t) \ge 0 & \forall k, e, t \\ \hat{\lambda}_e(t) \ge 0 & \forall e, t \\ \hat{d}_{k,e}(t) \ge 0 & \forall k, e, t \end{array} \right\}. \tag{19}$$

2) Action Space: The action $\boldsymbol{a}(t)$ performed at processing period t can is defined as

$$\boldsymbol{a}(t) = [a_1(t), \ldots, a_K(t)] \tag{20}$$

where $\boldsymbol{a}(t)$ is the locations of the group service placement. Therefore, action space $\mathcal{A}$ is

$$\mathcal{A} = \{a_k(t) \in \{1, 2, \ldots, E\} \quad \forall k\} \tag{21}$$

where the size of action space is E^K.

3) Reward Function: In (9), our goal is to decrease the average system delay in long term. Therefore, the reward is equal to the negative value of user perceived delay as follows:

$$R(t) = -1 \times (C(Y(t)) + D(Y(t)) + M(Y(t))). \tag{22}$$

4) Transition Probability: $P(\boldsymbol{s}(t+1)|\boldsymbol{s}(t), \boldsymbol{a}(t))$ denotes the transition probability from the state $\boldsymbol{s}(t)$ to state $\boldsymbol{s}(t+1)$ after taking action $\boldsymbol{a}(t)$. $P(\boldsymbol{s}(t+1)|\boldsymbol{s}(t), \boldsymbol{a}(t))$ is affected by the time-varying factors of $\boldsymbol{s}(t)$ and $\boldsymbol{a}(t)$, which contains the user locations, the number of user requests, and the usage of each edge and so on. When any time-varying factor changes, the state vector will change from $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$.

B. 2-D-Asynchronous Advantage Actor-Critic (2-D-A3C)

Our DRL-based service migration algorithm is based on the A3C algorithm [5], which is a widely used DRL algorithm in MEC scenarios. To adopt the A3C algorithm for multigroup service migration, we need to address the challenge of high-dimensional action space. In this section, we first introduce the high-dimensional action space problem in DRL application, and then introduce the 2-D-A3C-based service migration algorithm.

RL is a traditional approach to solve the corresponding MDP problems. For the problems where state transition probability $P_r(s'|s, a)$ is known as a priori, traditional dynamic planning is used to select the optimal strategy by value iteration and policy iteration. For the problems without $P_r(s'|s, a)$ known as a priori, model-free RL methods, such as Q-learning and Monte Carlo are proposed to explore the correlation between environment and optimal actions. However, such RL methods cannot deal with large-scale problems. In addition, traditional algorithms are unable to optimize sequential decision problems in the long term in the absence of future information such as user movement trajectories. To address the challenge, DRL-based methods are proposed, which utilize a neural network to approximate the strategy function or value function in the MDP model. Moreover, DRL can approximate the long-term optimal solution of the sequential decision-making problem through the agent's exploration on the environment. In other words, the system learns the best real-time migration decision from the historical experience accumulated by the agent, without requiring the user's future information as prior knowledge.

For our MDP problem, the transition probability $P_r(s'|s, a)$ is unknown and the size of action space is E^K. DRL will fall into the dilemma of discrete action space when dealing with the MDP problem: the scale of the network output layer is E^K, which increases explosively with the increase of E or K. To solve the problem, we propose a 2-D-A3C approach to deal with the large action space, which is based on the A3C algorithm. The proposed algorithm can parameterize and iteratively update the service migration policy to find the optimal

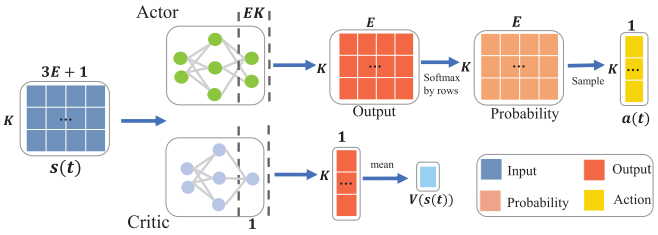

Fig. 13. 2-D-A3C algorithm in Actor–Critic framework.

policy. To further understand the proposed 2-D-A3C approach, we introduce the design and implementation of 2-D-A3C in detail as follows.

1) Design of 2-D-A3C: Compared with the general A3C algorithm, the 2-D-A3C consists of a global network and two local networks (i.e., actor network and critic network). It uses a standard multicore CPU on an independent machine to run multiple threads, and each thread carries an agent and a service migration environment copy. As shown in Fig. 13, each agent consists of two parts: 1) an actor network and 2) a critic network. For the actor network, we parameterize the migration policy as a policy network $\pi(a|s; \theta)$. The agent inputs the observation state $s(t)$ to the actor network, and gets the action probability distribution by the Softmax model. After that, the action $a(t)$ actually taken by the agent is generated according to the probability distribution sampling. For the critic network, we fit it as the value function $V(s(t)|\theta_v)$. Therefore, it can evaluate the quality of the action $a(t)$ according to the obtained reward from environment, and then instruct the actor network to update network parameters θ. In this case, the service migration strategy will be optimized iteratively. Moreover, the A3C algorithm improves the convergence of actor-critical network through an asynchronous parallel mode: each thread interacts with their own environment copy in parallel, updates the global network based on the local parameter gradient, and pull the latest network parameters from the updated global network every certain time step.

In Fig. 13, our 2-D-A3C framework improves the network structure to cope with the high-dimensional discrete action space problem in multiuser service migration scenarios. Specifically, the traditional A3C framework considers the actions of all users as a whole, and uses a 1-D vector as the input and output of networks. For example, if we apply the A3C model into our MDP problem, the input of the local actor network is a 1-D vector of size $(K(3E + 1))$. The output is a 1-D vector of size $(1 \times E^K)$. After that, we can get the $\pi(a|s_t; \theta)$ for all possible actions, thus making the action decision. Instead, we convert the 1-D agent observation into 2-D-state space, where each row represents the information of a user group. The large action space of agent can be further regarded as the joint small action space of multiuser groups. For example, in our local actor network, we obtain a 2-D-state vector $\boldsymbol{s}(t)$ with size of $K \times (3E+1)$ as input and produce a 2-D vector with size of $K \times E$ as output. Softmax is utilized to normalize the output by row, and we will get a possibility matrix $P = [p_1^1, \ldots, p_1^E; \cdots; p_K^1, \ldots, p_K^E]$, where the variable p_k^e represents the possibility of user group k placing the services on

the edge node e. According to the possibility matrix P, the agent action a_t is finally generated by sampling. Moreover, we also utilize a critic network to evaluate the output strategy of actor network. Specially, we input the 2-D-state vector $\boldsymbol{s}(t)$ into the critic network, then we will get a 1-D vector of size $K*1$ as output. To accurately compute gradients normally, we take the mean value of all elements in this vector to generate the variable $V(\boldsymbol{s}(t))$.

2) Implement of 2-D-A3C: The implementation of 2-D-A3C can be divided into two phases: 1) training phase and 2) deployment phase. In the training phase, the 2-D-A3C conducts multithread execution, where each thread maintains a local actor network and critic network. The training flow of each thread is presented as follows. At the beginning of each training event, the local network pulls the latest neural network parameters from the global network and updates themselves. Subsequently, at each time slot t, the agent managed by the operator inputs the state vector $s(t)$ into the actor network. As a result, the agent can get the multigroup service placement decision $a(t)$ according to a series of operations in Fig. 13. When the action $a(t)$ is taken, the environment gets into the next state $s(t)$ and feeds back the reward value $R(s(t), a(t))$ to the agent. Such process will be repeated until the system enters the termination state or reaches the specified number (i.e., $t_{\max}$) of thread iterations. After that, the agent calculates the cumulative gradient parameters for actor and critical networks, and asynchronously uploads them to the global network. The global networks update their parameters, and the system starts the next training event. The process is repeated until the global shared counter reaches the specified value $T_{\max}$. In the deployment phase, the actor and critical network parameters will no longer be updated. In each time slot t, the operator inputs the observation $s(t)$ into the global actor network, and obtains the service migration strategy from the output.

In summary, the proposed 2-D-A3C-based service placement policy uses a general asynchronous and concurrent DRL framework to accelerate convergence efficiency. The detailed algorithm for each agent is presented in Algorithm 2.

Algorithm 2 Service Migration Policy Scheme (2-D-A3C)

// Assume global shared parameter vector θ, θ_v and global shared counter $T = 0$.
// Assume thread-specific parameter vector θ' and θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients:$d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Get a_t according to policy $\pi(a_t|s_t;\theta')$ in the way of Fig. 13
    Perform a_t in the environment
    Receive reward r_t and new state $s_t + 1$
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t or $t - t_{start} == t_{\max}$
  $R = \bar{V}(s_t, \theta'_v)$
  for $i \in t-1, \ldots, t_{start}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients with respect to θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi\left(a_i \mid s_i; \theta'\right)\left(R - \bar{V}\left(s_i; \theta'_v\right)\right)$
    Accumulate gradients with respect to θ'_v: $d\theta_v \leftarrow d\theta_v + \partial\left(R - \bar{V}\left(s_i; \theta'_v\right)\right)^2 / \partial\theta'_v$
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{\max}$

VII. Performance Evaluation

In this section, we verify the effectiveness of MSM with extensive simulations, utilizing the large-scale traces from a campus Wi-Fi system.

A. Methodology

1) Simulation Setup:

1) *Environment Setup:* In this article, the campus Wi-Fi system deploys about 8000 APs and provides Internet access for 40 000 users. We consider a large-scale MEC system, which randomly selected $N = 50$ APs and $E = 10$ edge nodes from the data set. In our simulation, the top ten most popular APs are selected to act as edge nodes. The computation capacity of each edge node is set to be the same, which is $R_e = 5$ GHz. Moreover, to obtain the users' preference over APs, we analyze the user association records from 20 April 2019 to 27 May 2019. In addition, we utilize the data from 27 May to 2 June to train the model, and the data from 3 June to 9 June is used to test the performance of MSM.
2) *Request Setup:* In our simulations, users will generate a certain number of service requests in each time slot, and the service requests between users are independent of each other. To simplify the problem, we consider that the size of computing resources required for each service request is also set to be the same, which is $0.04R_e$.
3) *Communication:* The AP-to-edge communication cost in our simulations is considered as a function of distance (measured by the number of network hops) between different AP nodes and MEC nodes. When a user transfers the service requests from AP node j to MEC node e during time slot t, its unit communication delay can be computed as

$$\mathrm{l}_j^e(t) = \begin{cases} l \cdot \mathrm{dist}(j, e), & \text{if } j \neq e \\ 0, & \text{if } j = e. \end{cases} \tag{23}$$

4) *Service Migration:* Similarly, the migration delay in our simulations is considered as a function of distance (measured by the number of network hops) between different MEC nodes. When a user's service is migrated from MEC node k to MEC node e during time slot t, its migration delay can be computed as

$$\mathrm{m}^{ke}(t) = \begin{cases} \kappa \cdot \mathrm{dist}(k, e), & \text{if } k \neq e \\ 0, & \text{if } k = e. \end{cases} \tag{24}$$

TABLE II
SIMULATION PARAMETERS

Parameters	Value	Definition
N	50	The number of APs
E	10	The number of edge nodes
R_e	5GHz	The computation capacity of edge node e
$G_i(t)$	$[1, 2]$	The number of requests generated by user i
l	0.6s	One hop communication delay
κ	0.6s	One hop migration delay

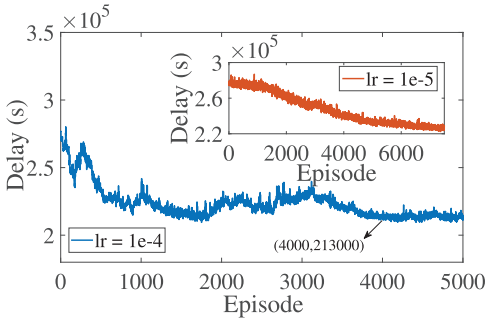

Fig. 14. MSM learning curve.

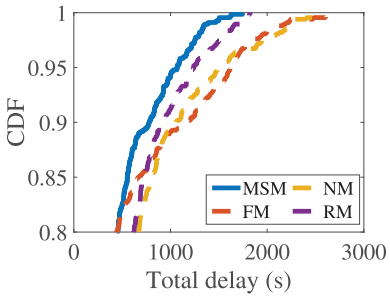

Fig. 15. CDFs of system delay.

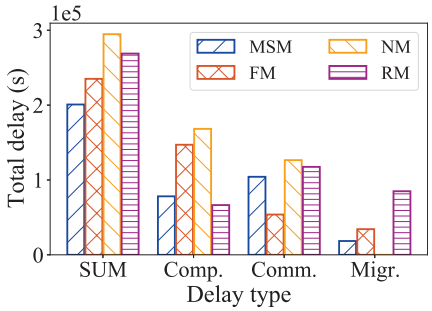

Fig. 16. Accumulated system delay in a week.

The detailed simulation parameters are shown in Table II.

2) Benchmark Schemes:

1) *No Migration (NM):* The NM scheme always keeps the initial service placement policy unchanged for each user.
2) *Follow Mobility (FM):* The FM scheme always migrates the services to the nearest MEC nodes.
3) *Random Migration (RM):* The RM scheme makes random service placement decision for the user.
4) *MSM_NC:* It is the MSM scheme without considering user classification.

3) Convergence Performance of MSM: In our simulations, we set the reward discount factor $\gamma = 0.9$. The number of 2-D-A3C threads is 4 and the size of mini-batch is 16. Both the actor networks and critic networks are a 3-layer fully connected deep neural network, where the number of hidden states is set to be 512. We adopt the Adam [31] optimizer. Moreover, we utilize the PyTorch framework to implement the neural networks of actor network and critic network.

Fig. 14 shows the convergence performance of the proposed MSM scheme. The learning rates of two curves are set to be 10^{-4} and 10^{-5}, respectively. We have two observations as follows.

1) The total delay of two curves both decreases as the number of episodes increases, and then becomes stable when the number of episodes is larger than the number of certain episodes. In this case, we can observe that MSM scheme converges for both cases.
2) With a lower learning rate, the scheme converges more slowly. The learning rate is set to be 10^{-4} in the following performance comparisons.

B. Performance Comparison

1) System Delay: To evaluate the performance of MSM, we consider the collected association records from 3 June 2019 to 9 June 2019. Then, we plot the system delay of MSM and benchmarks in Figs. 15 and 16. Fig. 15 shows the CDF of the system delay in each time slot. In addition, we can see that MSM scheme outperforms other benchmark schemes. For example, the 90th percentile for the service delay is 779.5, 1058.2, 1123.8, and 926.6 s in the MSM, FM, NM, and RM schemes, respectively.

To evaluate the accumulated delay for all users in the week, we present all types of delays as shown in Fig. 16. Then, we take *Comp.*, *Comm.*, *Migr.*, and *SUM* to denote the computation delay, communication delay, migration delay, and the sum of all types of delays, respectively. Therefore, we have five main observations as follows.

1) The accumulated delay of MSM is lower with around 14.6% improvement over FM, 31.9% improvement over NM, and 25.3% improvement over RM.
2) FM achieves the minimum communication time, while incurring large computation and migration delay.
3) The RM scheme makes the random service placement decision for each user, thereby balancing the MEC node loads and minimizing the computation time. However, the incurring large migration delay affects the RM scheme performance.
4) With unchanged service placement locations, the migration delay of NM scheme is 0. Due to the users' highly dynamic mobility, NM scheme incurs large computation delay and communication delay.
5) MSM outperforms other benchmark schemes by reducing the number of service migrations and computation time.

2) User-Perceived Delay: Compared with the system delay, the user-perceived delay is an important metric to measure the users' QoS in service migration policy. Specifically, we

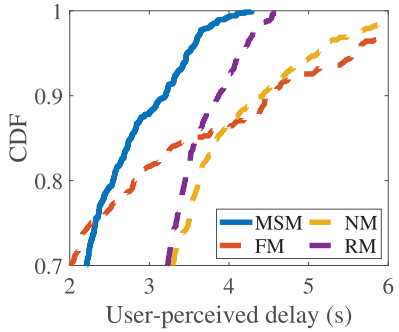

Fig. 17. User-perceived delay.

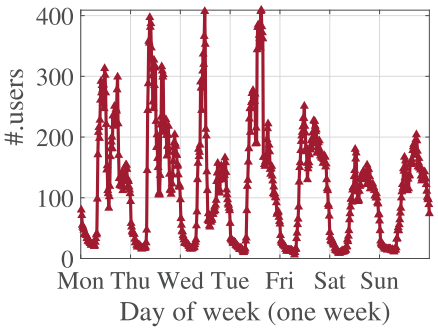

Fig. 18. Number of users.

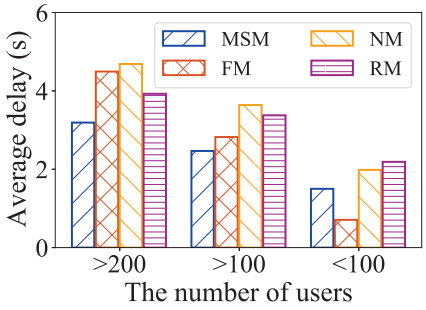

Fig. 19. Avg. user-perceived delay.

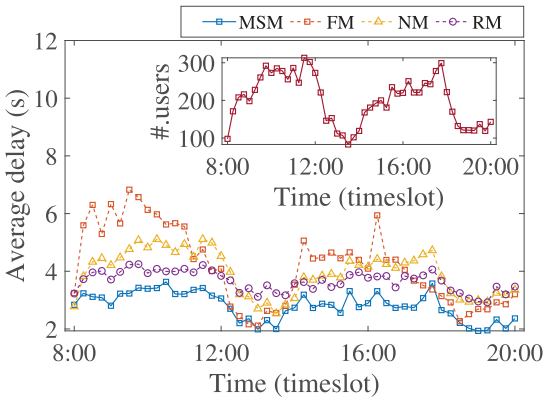

Fig. 20. Overall system delay and number of users in daytime.

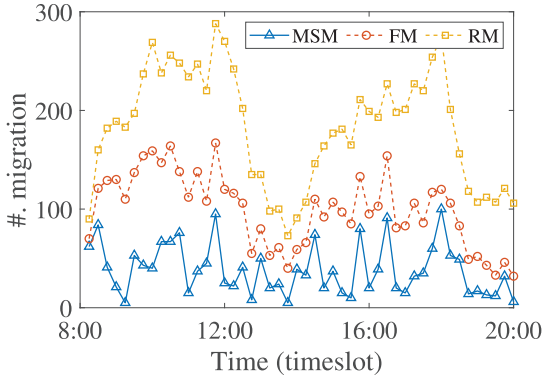

Fig. 21. Number of service migrations.

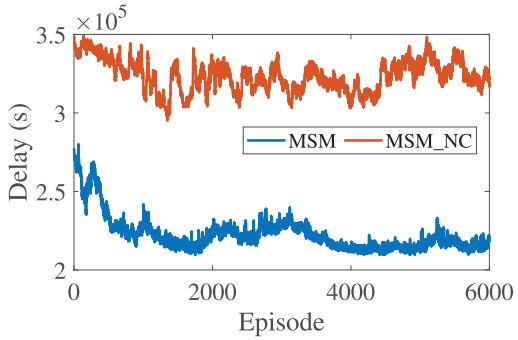

Fig. 22. Training curves.

plot the CDFs of user-perceived delay results for MSM and benchmarks in Fig. 17. In addition, we can see that MSM outperforms other benchmark schemes. For instance, 90th percentile for the average user-perceived delay is 3.2, 3.9, 4.4, and 4.5 s in MSM, RM, NM, and FM schemes, respectively.

As shown in Fig. 18, we plot the number of users in one week for each day. We can see that the number of users is less than 100, which means that the overall system is idle at night. Moreover, we can observe that the number of users is up to 400, which means that the system is busy at the daytime. In this case, the pattern of user activeness is in line with our daily habits. To evaluate the performance of MSM, we show the average user-perceived delay with different numbers of users as shown in Fig. 19. Specifically, we have two observations as follows. First, MSM scheme outperforms other benchmark schemes significantly when the system is busy as shown in Fig. 19. Second, the FM scheme outperforms MSM when the system is empty since it always migrates the services to the nearest edge nodes of users. In light-load scenarios, the trivial number of users cannot highlight the shortcomings of the FM, such as unbalanced loads and unnecessary migrations.

3) Delay Variation Over Time: To further verify the performance of MSM scheme, we explore the performance of MSM in each time slot of daytime. Specifically, we present the overall system delay and the number of users of each time slot from 8:00 A.M. to 8:00 P.M. on a weekday in Fig. 20, respectively. We have two main observations as follows: 1) the MSM outperforms other benchmark schemes in most time slots and 2) the average user-perceived delay keeps pace with the number of users in a time slot.

4) Effect of User Classification: To evaluate the effect of user classification, we plot the learning curves of MSM and MSM_NC, respectively, in Fig. 22. It is clearly observed

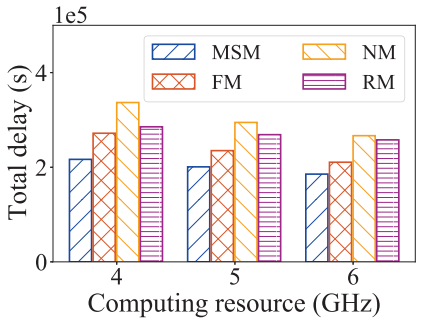

Fig. 23. Impact of computing resources.

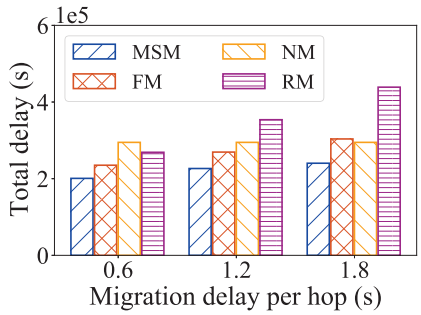

Fig. 24. Impact of migration.

that the user classification effectively improves the MSM performance by accelerating the learning rate and decreasing the service delay. Moreover, we present the number of service migrations of MSM and MSM_NC at different time slots as shown in Fig. 21. We can see that the benchmark schemes migrate services frequently and MSM decreases unnecessary migrations due to the user classification design.

C. Impact Evaluation

1) Impact of Computation Resource R_e: Computing resource is a key factor to affect the performance of computing delay. Fig. 23 shows the accumulated delay in a week of MSM and benchmarks with different edge computing resources. It is observed that the accumulated delays of all schemes increase with less computing resources because of the increasing computing delay. In addition, by utilizing the RL framework to keep pace with the dynamic environment, MSM achieves better performance improvement in less computing resource cases. The main reason is that MSM considers the load status of edge nodes, the DRL can comprehensively consider all observation information to make optimal service migration decisions even when computing resources are less. In addition, we can see that the performance of the FM and NM scheme is worse since they do not consider the load status of edge nodes when making service migration decisions.

2) Impact of Migration Delay Per Hop κ: κ is an important factor to affect the performance of migration delay in the large-scale MEC scenario. Fig. 24 shows the accumulated delay in a week of MSM and benchmarks with different migration delays per hop. With a larger value of κ, the accumulated delays of four schemes all increase due to the increasing migration delay. In addition, MSM scheme outperforms other benchmark schemes significantly. The main reason is that MSM scheme can balance the different delays to minimize the overall system delay in real time in the large-scale MEC system. However, FM, NM, and RM schemes cannot adjust migration policy in real time and do not consider the impact of different delays due to environmental changes.

VIII. Conclusion

To minimize the system delay in the long term, we have proposed an MSM scheme in large-scale MEC scenarios. The MSM scheme can achieve efficient user management in large-scale MEC systems by characterizing user mobility patterns from the real Wi-Fi traces; select the suitable service placement location in real time to keep pace with the dynamic and complicated network scenarios by using the DRL-based approach; and minimize the long-term system delay. For future work, we will study the preservice placement strategy with caching at edge servers by using historical experience to further improve the overall performance of the large-scale MEC system.

References

[1] Y. Miao et al., "Mobility-aware service migration for seamless provision: A reinforcement learning approach," in *Proc. IEEE ICC*, May 2022, pp. 1–6.
[2] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surveys*, vol. 52, no. 6, pp. 1–36, 2019.
[3] 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.
[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] F. Wu, W. Yang, J. Lu, F. Lyu, J. Ren, and Y. Zhang, "RLSS: A reinforcement learning scheme for HD map data source selection in vehicular NDN," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10777–10791, Jul. 2022.
[6] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 787–800, Apr. 2019.
[7] X. Ma, Q. Li, J. Chai, X. Xiao, S.-T. Xia, and Y. Jiang, "Steward: Smart edge based joint QoE optimization for adaptive video streaming," in *Proc. ACM Workshop NOSSDAV*, 2019, pp. 31–36.
[8] Z. Fan, W. Yang, F. Wu, J. Cao, and W. Shi, "Serving at the edge: An edge computing service architecture based on ICN," *ACM Trans. Internet Technol.*, vol. 22, no. 1, pp. 1–27, Oct. 2021.
[9] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.
[10] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile social data learning for user-centric location prediction with application in mobile edge service migration," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7737–7747, Oct. 2019.
[11] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1440–1452, May 2016.
[12] A. Ksentini, T. Taleb, and M. Chen, "A Markov decision process-based service migration procedure for follow me cloud," in *Proc. IEEE ICC*, 2014, pp. 1350–1354.
[13] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. IEEE IFIP*, 2015, pp. 1–9.

[14] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.
[15] Z. Ning et al., "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, Jun. 2021.
[16] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[17] F. Lyu et al., "LeaD: Large-scale edge cache deployment based on spatio-temporal WiFi traffic statistics," *IEEE Trans. Mobile Comput.*, vol. 20, no. 8, pp. 2607–2623, Aug. 2021.
[18] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 2876–2880.
[19] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3051–3063, Nov. 2015.
[20] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1227–1240, Jun. 2020.
[21] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, 1st Quart., 2020.
[22] W. Li, Q. Li, L. Chen, F. Wu, and J. Ren, "A storage resource collaboration model among edge nodes in edge federation service," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9212–9224, Sep. 2022.
[23] S. Wang, R. Urgaonkar, M. Zafer, T. He, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.
[24] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *Proc. IEEE INFOCOM*, 2019, pp. 1468–1476.
[25] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Energy-aware application placement in mobile edge computing: A stochastic optimization approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 4, pp. 909–922, Apr. 2020.
[26] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Trans. Mobile Comput.*, vol. 18, no. 9, pp. 2020–2033, Sep. 2019.
[27] J. Xu, X. Ma, A. Zhou, Q. Duan, and S. Wang, "Path selection for seamless service migration in vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9040–9049, Sep. 2020.
[28] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE INFOCOM*, Paris, France, 2019, pp. 1459–1467.
[29] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.
[30] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021.
[31] H. Liu and G. Cao, "Deep reinforcement learning-based server selection for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13351–13363, Dec. 2021.
[32] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.
[33] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 8050–8062, Aug. 2019.
[34] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, Aug. 2020.
[35] Y. Deng et al., "AUCTION: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1996–2009, Aug. 2022.
[36] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[37] W. Y. B. Lim et al., "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.
[38] F. Wu, F. Lyu, H. Wu, J. Ren, Y. Zhang, and X. Shen, "Characterizing user association patterns for optimizing small-cell edge system performance," *IEEE Netw.*, early access, Sep. 5, 2022, doi: 10.1109/MNET.121.2200089.
[39] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9441–9455, Oct. 2020.
[40] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *Proc. IEEE INFOCOM*, 2020, pp. 2499–2508.
[41] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5G networks: A deep learning based approach," in *Proc. IEEE/ACM IWQoS*, 2018, pp. 1–6.
[42] B. Chen, L. Liu, M. Sun, and H. Ma, "IoTCache: Toward data-driven network caching for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10064–10076, Dec. 2019.
[43] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.
[44] W. Y. B. Lim et al., "Realizing the Metaverse with edge intelligence: A match made in heaven," *IEEE Wireless Commun.*, early access, Jul. 4, 2022, doi: 10.1109/MWC.018.2100716.
[45] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.
[46] F. Wu et al., "Characterizing Internet card user portraits for efficient churn prediction model design," *IEEE Trans. Mobile Comput.*, early access, Jan. 31, 2023, doi: 10.1109/TMC.2023.3241206.
[47] S. Wang, Y. Sun, and Z. Bao, "On the efficiency of K-means clustering: Evaluation, optimization, and algorithm selection," *Proc. VLDB Endow.*, vol. 14, no. 2, pp. 163–175, 2020.

Wenxiong Chen (Member, IEEE) received the B.S. degree in communication engineering and the M.S. degree in translation from Hunan Normal University, Changsha, China, in 2007 and 2011, respectively, and the Ph.D. degree in management science and engineering from Central South University, Changsha, in 2021.

He is currently an Associate Professor with the College of Information Science and Engineering, and an Associate Researcher with the Research Institute of Languages and Cultures, Hunan Normal University. His research interests include information management, big data measurement, and application design.

Mingliu Liu (Member, IEEE) received the Ph.D. and B.E. degrees from the School of Electronic Information, Wuhan University, Wuhan, China, in 2013 and 2019, respectively.

She is currently a Postdoctoral Research Fellow with the State Grid Hubei Electric Power Research Institute, Wuhan, and with the School of Electronic Information, Wuhan University. Her research interests include information search service, Internet of Things, cloud/edge caching, and big data driven applications.

Prof. Liu is a member of the IEEE Computer and IEEE Communication Society.

Fan Wu (Member, IEEE) received the Ph.D. degree in computer science and technology from Central South University, Changsha, China, in 2020.

From 2018 to 2019, he was a visiting Ph.D. student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He worked as a Postdoctoral Fellow with the Department of Computer Science and Technology, Tsinghua University, Beijing, China, from July 2020 to November 2022. She is currently an Assistant Professor with the School of Computer Science and Engineering, Central South University. His current research interests include Internet of Things, mobile-edge computing, data mining, and big data.

Huaqing Wu (Member, IEEE) received the B.E. and M.E. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 2014 and 2017, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2021.

She worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, MacMaster University, Hamilton, ON, Canada, from 2021 to 2022. He is currently an Assistant Professor with the Department of Electrical and Software Engineering, University of Calgary, Calgary, AB, Canada. Her current research interests include B5G/6G, space–air–ground integrated networks, Internet of Vehicles, edge computing/caching, and artificial intelligence for future networking.

Dr. Wu received the Best Paper Awards at the IEEE GLOBECOM 2018, the *Chinese Journal on Internet of Things* 2020, and the IEEE GLOBECOM 2022, and the prestigious Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship Award in 2021.

Yuan Miao received the B.E. and M.E. degrees from Central South University, Changsha, China, in 2019 and 2022, respectively.

Her main research interests include big data, deep reinforcement learning, and edge computing.

Feng Lyu (Senior Member, IEEE) received the B.S. degree in software engineering from Central South University, Changsha, China, in 2013, and the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018.

From September 2018 to December 2019 and from October 2016 to October 2017, he worked as a Postdoctoral Fellow and was a visiting Ph.D. student with the BBCR Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. He is currently a Professor with the School of Computer Science and Engineering, Central South University. His research interests include vehicular networks, beyond 5G networks, big data measurement and application design, and edge computing.

Prof. Lyu is the recipient of the Best Paper Award of the IEEE ICC 2019. He currently serves as an Associate Editor for IEEE SYSTEMS JOURNAL and a leading Guest Editor for *Peer-to-Peer Networking and Applications*, and served as a TPC member for many international conferences. He is a member of the IEEE Computer Society, Communication Society, and Vehicular Technology Society.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada.

Prof. Shen received the R. A. Fessenden Award from IEEE, Canada, in 2019; the Award of Merit from the Federation of Chinese Canadian Professionals, Mississauga, ON, Canada, in 2019; the Technical Recognition Award from the Wireless Communications Technical Committee in 2019; the James Evans Avant Garde Award from the IEEE Vehicular Technology Society in 2018; the Education Award in 2017 and the Joseph LoCicero Award in 2015 from the IEEE Communications Society; the AHSN Technical Committee in 2013; the Excellent Graduate Supervision Award from the University of Waterloo in 2006; and the Premier's Research Excellence Award from the Province of Ontario, Canada, in 2003. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, IEEE VTC'10 Fall, and IEEE Globecom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was the Vice President for the Technical and Educational Activities and Publications, a Member-at-Large on the Board of Governors, and the Chair of the Distinguished Lecturer Selection Committee. He is also a Registered Professional Engineer in Ontario, Canada. He is also the President Elect of the IEEE Communications Society. He served as the Editor-in-Chief for IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He is 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.