Journal of Parallel and Distributed Computing 210 (2026) 105216

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc

Mobility -aware server placement and power allocation for randomly walking mobile users

Keqin Li

Department of Computer Science, State University of New York, 12561, New Paltz, New York, USA

ARTICLE INFO

Keywords:
Average response time
Markov chain
Mobility-awareness
Power allocation
Queueing system
Random walk
Server placement
Task offloading

ABSTRACT

We systematically, quantitatively, and mathematically address the problems of optimal mobility-aware server placement and optimal mobility-aware power allocation in mobile edge computing environments with randomly walking mobile users. The new contributions of the paper are highlighted below. We establish a single-server M/G/1 queueing system for mobile user equipment and a multiserver M/G/k queueing system for mobile edge clouds. We consider both the synchronous mobility model and the asynchronous mobility model, which are described by discrete-time Markov chains and continuous-time Markov chains respectively. We discuss two task offloading strategies for user equipment in the same service area, i.e., the equal-response-time method and the equal-load-fraction method. We formally and rigorously define the optimal mobility-aware server placement problem and the optimal mobility-aware power allocation problem. We develop optimization algorithms to solve the optimal mobility-aware server placement problem and the optimal mobility-aware power allocation problem. We demonstrate numerical data for optimal mobility-aware server placement and optimal mobility-aware power allocation with two mobility models, two task offloading strategies, and two power consumption models. The significance of the paper can be seen from the fact that the above analytical and algorithmic discussion of optimal mobility-aware server placement and optimal mobility-aware power allocation for mobile edge computing environments with randomly walking mobile users has rarely been seen in the existing literature.

1. Introduction

1.1. Background information

The problem of *server placement* in mobile edge computing refers to the strategic assignment of computing resources (servers) to various service areas, where multiserver systems are deployed to support randomly walking mobile users [1]. The goal of server placement is to position these resources closer to end-users to minimize latency, optimize service quality, and maximize resource utilization. An appropriate server placement ensures higher-speed computation and lower-latency communication for real-time applications such as augmented reality, online gaming, and video streaming. A proper server placement assures balanced resource utilization across servers, avoiding overloading certain servers while others remain underutilized.

The problem of *power allocation* in mobile edge computing involves distributing certain available power resources among servers in various service areas [2–4]. Since the server computation speeds determine power consumption, an appropriate power allocation ensures the best choice of server speeds, and thus, optimizing the overall performance of multiple multiserver systems, such as minimizing latency and maximizing throughput. A good power allocation method should consider different power consumption models for realistic applications.

1.2. Challenges and motivation

There are several critical and challenging aspects in the serious and systematic study of optimal mobility-aware server placement and optimal mobility-aware power allocation. First, since both user equipment and mobile edge clouds have computation and communication capabilities, they need to be specified formally as server systems using queueing theory. Second, since the random mobility of mobile users affects server placement and power allocation, it should be characterized mathematically using probability theory. Third, since multiple mobile user equipment in the same service area share and compete for server resources, their interaction needs to be studied analytically in the context of task offloading. Fourth, the optimization goals of optimal server placement and optimal power allocation must be described quantitatively. Fifth, the problems of optimal mobility-aware server placement and optimal mobility-aware power allocation should be clearly formulated and rigorously defined. Sixth, the problems of optimal mobility-

E-mail address: lik@newpaltz.edu

https://doi.org/10.1016/j.jpdc.2025.105216
Received 25 January 2025; Received in revised form 5 November 2025; Accepted 24 December 2025
Available online 6 January 2026

aware server placement and optimal mobility-aware power allocation should be solved algorithmically.

Unfortunately, there has been little such productive investigation within the above framework. The motivation of this paper is to make efforts towards this direction.

1.3. Contributions and significance

In this paper, we systematically, quantitatively, and mathematically address the problems of optimal mobility-aware server placement and optimal mobility-aware power allocation in mobile edge computing environments with randomly walking mobile users. The new contributions of the paper are highlighted below.

- We establish a single-server M/G/1 queueing system for mobile user equipment and a multiserver M/G/k queueing system for mobile edge clouds.
- We consider both the synchronous mobility model and the asynchronous mobility model, which are described by discrete-time Markov chains and continuous-time Markov chains respectively.
- We discuss two task offloading strategies for user equipment in the same service area, i.e., the equal-response-time method and the equal-load-fraction method.
- We formally and rigorously define the optimal mobility-aware server placement problem and the optimal mobility-aware power allocation problem.
- We develop optimization algorithms to solve the optimal mobility-aware server placement problem and the optimal mobility-aware power allocation problem.
- We demonstrate numerical data for optimal mobility-aware server placement and optimal mobility-aware power allocation with two mobility models, two task offloading strategies, and two power consumption models.

The significance of the paper can be seen from the fact that the above analytical and algorithmic discussion of optimal mobility-aware server placement and optimal mobility-aware power allocation for mobile edge computing environments with randomly walking mobile users has rarely been seen in the existing literature. Therefore, the paper has made tangible contributions in this direction.

1.4. Paper organization

In Section 2, we establish our analytical models, including queueing systems for user equipment and mobile edge clouds, and Markov chains for synchronous mobility and asynchronous mobility. In Section 3, we discuss task offloading to an MEC in a service area with multiple UEs. In Section 4, we formulate and solve the mobility-aware server placement problem. In Section 5, we formulate and solve the mobility-aware power allocation problem. In Section 6, we review related research on server placement and power allocation. In Section 7, we conclude the paper.

2. Analytical models

In this section, we establish our analytical models, including queueing systems for user equipment and mobile edge clouds, and Markov chains for synchronous mobility and asynchronous mobility.

2.1. Server modeling: queueing systems

We consider a mobile edge computing system with multiple mobile user equipment UE$_0$, UE$_1$, …, UE$_{m-1}$, which randomly move among multiple service areas SA$_0$, SA$_1$, …, SA$_{n-1}$, where each service area SA$_j$ is equipped with a mobile edge cloud service system MEC$_j$ with k_j servers (see Fig. 1). All UEs and MECs are considered task servers and formulated as queueing systems which are described in detail in this section.

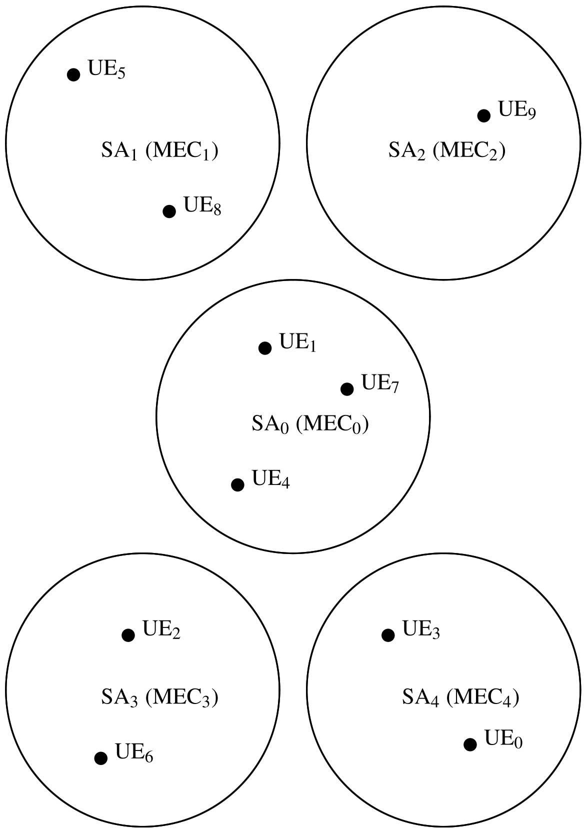

Fig. 1. Mobile UEs randomly walk among the service areas (i.e., the big circles) SA$_0$, SA$_1$, …, SA$_{n-1}$. Each SA$_j$ has a multiserver MEC$_j$, $0 \le j \le n-1$. The UE groups are $I_0 = \{1,4,7\}$, $I_1 = \{5,8\}$, $I_2 = \{9\}$, $I_3 = \{2,6\}$, $I_4 = \{0,3\}$.

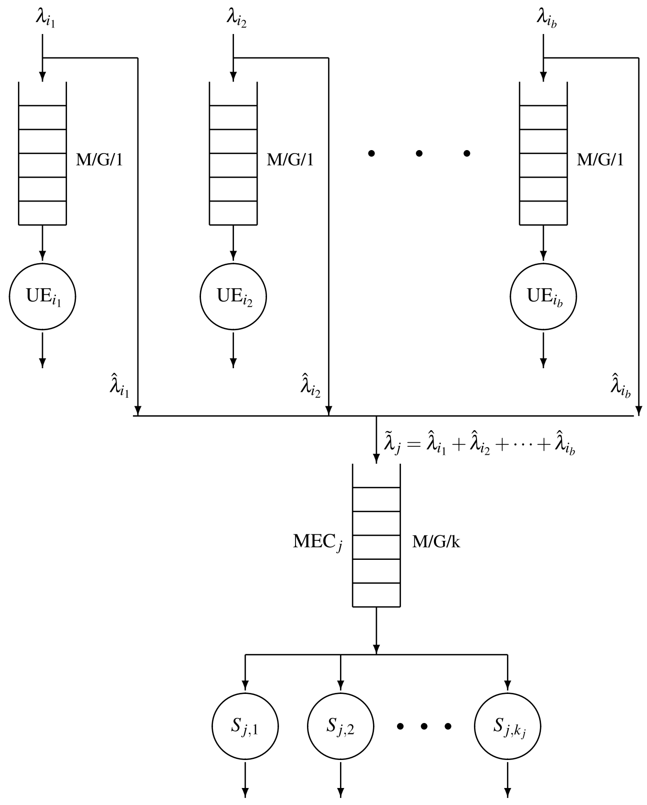

Fig. 2. A service area SA$_j$ with UE$_{i_1}$, UE$_{i_2}$, …, UE$_{i_b}$ and MEC$_j$ with k_j servers $S_{j,1}, S_{j,2}, \ldots, S_{j,k_j}$. Each UE$_i$ is modeled as an M/G/1 queueing system. Each MEC$_j$ is modeled as an M/G/k queueing system.

2.1.1. User equipment: M/G/1 queueing systems

Each *user equipment* UE$_i$, where $0 \le i \le m-1$, is treated as a single-server M/G/1 queueing system with a task-waiting queue of infinite capacity and the first-come-first-serve queueing discipline (see Fig. 2). The workload on UE$_i$ can be described by five parameters: λ_i, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$. Assume that there is a Poisson stream of tasks arriving at UE$_i$. Let λ_i be the arrival rate (measured by the number of tasks per second). We use r_i to denote the independent and identically distributed (i.i.d.) random task computation requirements (measured by the number of billion instructions (BI)) with mean $\overline{r_i}$ and second moment $\overline{r_i^2}$. (Throughout the paper, we use $\overline{z}$ and $\overline{z^2}$ to represent the mean and the second moment of a random variable z.) We use d_i to denote the i.i.d. random task communication requirements (measured by the number of million bits (MB)) with mean $\overline{d_i}$ and second moment $\overline{d_i^2}$. We assume that r_i and d_i are independent of each other.

UE$_i$ has computation speed s_i (measured by BI/second). If a task received by UE$_i$ is processed locally on UE$_i$ itself, the execution time (measured by second) is a random variable

$$x_i = \frac{r_i}{s_i},$$

whose mean is

$$\overline{x_i} = \frac{\overline{r_i}}{s_i},$$

and whose second moment is

$$\overline{x_i^2} = \frac{\overline{r_i^2}}{s_i^2},$$

for all $0 \le i \le m-1$.

2.1.2. Mobile edge clouds: M/G/k queueing systems

Each *mobile edge cloud* MEC$_j$, where $0 \le j \le n-1$, is treated as a multiserver M/G/k queueing system with a task-waiting queue of infinite capacity and the first-come-first-serve queueing discipline. The M/G/k queueing system has k_j (called the *size* of the multiserver system) identical servers $S_{j,1}, S_{j,2}, \ldots, S_{j,k_j}$ (see Fig. 2).

Assume that UE$_i$ is in SA$_j$ so that UE$_i$ can offload its tasks to MEC$_j$. The task arrival stream of UE$_i$ is split into two substreams. The first substream of tasks processed locally on UE$_i$ has arrival rate $\lambda_i - \hat{\lambda}_i$. The second substream of tasks offloaded to and processed remotely on MEC$_j$ is $\hat{\lambda}_i$.

The UEs are divided into *UE groups*: $I_0, I_1, \ldots, I_{n-1}$, where $I_j = \{i \mid \text{UE}_i \text{ is in SA}_j\}$ is the set of indices of UEs in SA$_j$ (see Fig. 1). It is clear that all these UEs can offload their tasks to MEC$_j$. Hence, the task arrival rate of MEC$_j$ is

$$\tilde{\lambda}_j = \sum_{i \in I_j} \hat{\lambda}_i,$$

for all $0 \le j \le n-1$.

MEC$_j$ has computation speed $\tilde{s}_j$ (measured by BI/second). The communication speed between UE$_i$ and MEC$_j$ is $c_{i,j}$ (measured by MB/second). If UE$_i$ is in SA$_j$ and a task received by UE$_i$ is processed remotely on MEC$_j$, the execution time (measured by second) is a random variable

$$\tilde{x}_{i,j} = \frac{r_i}{\tilde{s}_j} + \frac{d_i}{c_{i,j}},$$

which is the computation time $r_i/\tilde{s}_j$ plus the communication time $d_i/c_{i,j}$, whose mean is

$$\overline{\tilde{x}_{i,j}} = \frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}},$$

and whose second moment is

$$\overline{\tilde{x}_{i,j}^2} = \frac{\overline{r_i^2}}{\tilde{s}_j^2} + 2\frac{\overline{r_i}\,\overline{d_i}}{\tilde{s}_j c_{i,j}} + \frac{\overline{d_i^2}}{c_{i,j}^2},$$

for all $0 \le i \le m-1$ and $0 \le j \le n-1$.

The determination of server sizes $k_0, k_1, \ldots, k_{n-1}$ is called *server placement* (see Section 4).

The determination of server speeds $\tilde{s}_0, \tilde{s}_1, \ldots, \tilde{s}_{n-1}$ is called *power allocation and speed setting* (see Section 5).

Both mobility-aware server placement and power allocation depend on mobility modeling.

2.2. Mobility modeling: Markov chains

In this section, we describe our mobility models for UEs and their location distributions. The material in this section is based on [5].

Mobility modeling of a UE$_i$ does not need its detailed trajectory information, but only the knowledge of how UE$_i$ changes its *service area* SA$_j$ and the associated MEC$_j$. When UE$_i$ moves from SA$_j$ to SA$_{j'}$, the task offloading strategy in both SA$_j$ and SA$_{j'}$, the average response time of all UEs in both SA$_j$ and SA$_{j'}$, and the average response time of both MEC$_j$ and MEC$_{j'}$ are changed (see Section 3).

2.2.1. Synchronous mobility: discrete-time Markov chains

In this section, we describe *discrete-time Markov chains* (DTMC) for synchronous mobility.

With *synchronous mobility*, all UEs move simultaneously. It is assumed that time is divided into equal-length slots. At the beginning of each time slot, each UE$_i$ can move from one SA to another SA or remain in the same SA. Such movement is controlled by an $n \times n$ *transition probability matrix*

$$\boldsymbol{P}_i = [p_i(j, j')],$$

where $p_i(j, j')$, $0 \le j, j' \le n-1$, is the transition probability of UE$_i$ from SA$_j$ to SA$_{j'}$ in a time slot, if UE$_i$ is currently in SA$_j$.

Let the *stationary probability vector* of UE$_i$ be denoted as

$$\pi_i = [\pi_i(0), \pi_i(1), \ldots, \pi_i(n-1)],$$

where $\pi_i(j)$ is the stationary probability that UE$_i$ is in SA$_j$. π_i can be obtained by solving the linear system of equations:

$$\pi_i = \pi_i \boldsymbol{P}_i,$$

with the condition

$$\pi_i(0) + \pi_i(1) + \cdots + \pi_i(n-1) = 1,$$

for all $0 \le i \le m-1$.

2.2.2. Asynchronous mobility: continuous-time markov chains

In this section, we describe *continuous-time Markov chains* (CTMC) for asynchronous mobility.

With *asynchronous mobility*, the UEs move at different times. A UE$_i$ can change its service area at any time. Such movement of UE$_i$ is controlled by an $n \times n$ *transition rate matrix*

$$\boldsymbol{Q}_i = [q_i(j, j')],$$

where $q_i(j, j')$, $0 \le j \ne j' \le n-1$, is the transition rate of UE$_i$ from SA$_j$ to SA$_{j'}$, if UE$_i$ is currently in SA$_j$. Notice that

$$q_i(j, j) = -\sum_{j' \ne j} q_i(j, j'),$$

and the *mean holding time* for UE$_i$ to stay in SA$_j$ is $-1/q_i(j, j)$, for all $0 \le j \le n-1$.

The *stationary probability vector* of UE$_i$, i.e.,

$$\pi_i = [\pi_i(0), \pi_i(1), \ldots, \pi_i(n-1)],$$

can be obtained by solving the linear system of equations:

$$\pi_i \boldsymbol{Q}_i = 0,$$

with the condition

$$\pi_i(0) + \pi_i(1) + \cdots + \pi_i(n-1) = 1,$$

for all $0 \le i \le m-1$.

2.2.3. Location distributions

A *location distribution* of the UEs is $J = (j_0, j_1, \ldots, j_{m-1})$, where UE$_i$ is in SA$_{j_i}$, for all $0 \le i \le m-1$. Equivalently, J can be treated as an m-digit radix-n integer $(j_0 j_1 \cdots j_{m-1})_n$ in the range $0, 1, \ldots, N-1$:

$$J = j_0 n^{m-1} + j_1 n^{m-2} + \cdots + j_{m-1} n^0,$$

for all $0 \le j_0, j_1, \ldots, j_{m-1} \le n-1$, where $N = n^m$ is the number of location distributions.

Let $\pi(J)$ be the stationary probability of a location distribution J, where $0 \le J \le N-1$. The stationary location distribution vector

$$\pi = [\pi(0), \pi(1), \ldots, \pi(N-1)]$$

can be calculated as follows. If $J = (j_0, j_1, \ldots, j_{m-1})$, then we have

$$\pi(J) = \prod_{i=0}^{m-1} \pi_i(j_i),$$

for all $0 \le J \le N-1$.

3. Task offloading

In this section, we discuss task offloading to an MEC in a service area with multiple UEs. The discussion in this section forms the basis for the optimization problems to be solved in this paper.

3.1. Average response time

In this section, we derive the average response time for each UE and MEC.

3.1.1. User equipment

Recall that the substream of tasks of UE$_i$ processed locally on UE$_i$ has arrival rate $\lambda_i - \hat{\lambda}_i$. Therefore, by the Pollaczek-Khinchin mean value formula, the *average response time* of UE$_i$ is

$$T_i = \overline{x_i} + W_i,$$

where W_i is the *average waiting time* of UE$_i$:

$$W_i = \frac{(\lambda_i - \hat{\lambda}_i)\overline{x_i^2}}{2(1-\rho_i)},$$

and ρ_i is the *utilization* of UE$_i$:

$$\rho_i = (\lambda_i - \hat{\lambda}_i)\overline{x_i},$$

for all $0 \le i \le m-1$ (see [6], p. 190).

3.1.2. Mobile edge clouds

For a given I_j, the execution time of a task on MEC$_j$ is a random variable $\tilde{x}_j$ with mean

$$\overline{\tilde{x}_j} = \frac{1}{\tilde{\lambda}_j} \sum_{i \in I_j} \hat{\lambda}_i \overline{\tilde{x}_{i,j}} = \frac{1}{\tilde{\lambda}_j} \sum_{i \in I_j} \hat{\lambda}_i \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}} \right),$$

and the second moment

$$\overline{\tilde{x}_j^2} = \frac{1}{\tilde{\lambda}_j} \sum_{i \in I_j} \hat{\lambda}_i \overline{\tilde{x}_{i,j}^2} = \frac{1}{\tilde{\lambda}_j} \sum_{i \in I_j} \hat{\lambda}_i \left(\frac{\overline{r_i^2}}{\tilde{s}_j^2} + 2\frac{\overline{r_i}\overline{d_i}}{\tilde{s}_j c_{i,j}} + \frac{\overline{d_i^2}}{c_{i,j}^2} \right),$$

and the variance

$$\sigma_j^2 = \overline{\tilde{x}_j^2} - \overline{\tilde{x}_j}^2,$$

and the coefficient of variation

$$c_j = \frac{\sigma_j}{\overline{\tilde{x}_j}} = \sqrt{\frac{\overline{\tilde{x}_j^2}}{\overline{\tilde{x}_j}^2} - 1},$$

for all $0 \le j \le n-1$.

The *average response time* of MEC$_j$ is

$$\tilde{T}_j = \overline{\tilde{x}_j} + \tilde{W}_j,$$

where $\tilde{W}_j$ is the average waiting time of MEC$_j$. By Kingman's law of congestion [7], it is known that the *average waiting time* of MEC$_j$ is approximately

$$\tilde{W}_j = \left(\frac{c_j^2 + 1}{2} \right) W_j^*,$$

where W_j^* is the average waiting time of an M/M/k queueing system with the same utilization as the M/G/k queueing system. The *utilization* of MEC$_j$ is

$$\tilde{\rho}_j = \frac{\tilde{\lambda}_j \overline{\tilde{x}_j}}{k_j} = \frac{1}{k_j} \sum_{i \in I_j} \hat{\lambda}_i \overline{\tilde{x}_{i,j}}.$$

Then, we have

$$W_j^* = \overline{\tilde{x}_j} \cdot \frac{p_{j,k_j}}{k_j(1-\tilde{\rho}_j)^2},$$

where

$$p_{j,k_j} = p_{j,0} \frac{(k_j \tilde{\rho}_j)^{k_j}}{k_j!},$$

and

$$p_{j,0} = \left(\sum_{b=0}^{k_j-1} \frac{(k_j \tilde{\rho}_j)^b}{b!} + \frac{(k_j \tilde{\rho}_j)^{k_j}}{k_j!} \cdot \frac{1}{1-\tilde{\rho}_j} \right)^{-1},$$

for all $0 \le j \le n-1$ (see [6], p. 102).

3.2. Task offloading strategies

In this section, we develop two task offloading strategies for user equipment in the same service area. A *task offloading strategy* is to decide the $\hat{\lambda}_i$'s.

3.2.1. Equal-response-time method

For an SA$_j$, the objective of the *equal-response-time* (ERT) method for all the UE$_i$'s in SA$_j$ is to find $\hat{\lambda}_i$ for all $i \in I_j$, such that all the UE$_i$'s served by MEC$_j$ and MEC$_j$ itself have *identical average response time*, i.e., $T_i = \tilde{T}_j = T$, for all $i \in I_j$ and some T. The rationale of the above task offloading strategy is to treat all UEs equally and satisfy all of them.

To get the $\hat{\lambda}_i$'s, we notice that

$$T_i = \overline{x_i} + \frac{(\lambda_i - \hat{\lambda}_i)\overline{x_i^2}}{2(1-(\lambda_i - \hat{\lambda}_i)\overline{x_i})} = T,$$

which gives

$$\hat{\lambda}_i = \lambda_i - \frac{2(T - \overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T - \overline{x_i})},$$

for all $i \in I_j$.

The last equation implies that each $\hat{\lambda}_i$ can be treated as a function $\hat{\lambda}_i(T)$ of T. Since $\tilde{T}_j$ is a function of the $\hat{\lambda}_i$'s, $\tilde{T}_j$ can also be treated as a function of T, i.e., $\tilde{T}_j(\hat{\lambda}_{i_1}(T), \hat{\lambda}_{i_2}(T), \ldots, \hat{\lambda}_{i_b}(T))$, where we assume that $I_j = \{i_1, i_2, \ldots, i_b\}$. Hence, T can be obtained by solving the equation

$$f(T) = T - \tilde{T}_j(\hat{\lambda}_{i_1}(T), \hat{\lambda}_{i_2}(T), \ldots, \hat{\lambda}_{i_b}(T)) = 0,$$

where $\tilde{T}_j$ is derived in Section 3.1.2.

To solve the above equation, we observe that increased T gives reduced $\hat{\lambda}_i$, which in turn, results in reduced $\tilde{T}_j(\hat{\lambda}_{i_1}(T), \hat{\lambda}_{i_2}(T), \ldots, \hat{\lambda}_{i_b}(T))$. Therefore, $f(T)$ is an increasing function of T. This means that the equation $f(T) = 0$ can be solved by using the standard bisection search algorithm. The search interval $[T_{lb}, T_{ub}]$ can be determined as follows. Since

$$\overline{x_i} \le T_i \le \overline{x_i} + \frac{\lambda_i \overline{x_i^2}}{2(1 - \lambda_i \overline{x_i})},$$

we can set

$$T_{lb} = \max_{i \in I_j}\{\overline{x_i}\},$$

and

$$T_{ub} = \min_{i \in I_j}\left\{ \overline{x_i} + \frac{\lambda_i \overline{x_i^2}}{2(1-\lambda_i \overline{x_i})} \right\}.$$

The above procedure takes $O(\log(\Delta/\epsilon))$ time, where $\Delta = T_{ub} - T_{lb}$ is the length of the search interval and ϵ is the accuracy requirement. Since Δ is a reasonably small quantity, the above time complexity will be simply treated as $O(\log(1/\epsilon))$.

3.2.2. Equal-load-fraction method

For an SA$_j$, the objective of the *equal-load-fraction* (ELF) method for all the UE$_i$'s in SA$_j$ is to set

$$\hat{\lambda}_i = F_j \lambda_i,$$

for all $i \in I_j$, such that the *weighted average response time* of all UE$_i$'s and MEC$_j$ is, i.e.,

$$T = \sum_{i \in I_j} \left(\frac{\lambda_i - \hat{\lambda}_i}{\Lambda_j} \right) T_i + \frac{\tilde{\lambda}_j}{\Lambda_j} \tilde{T}_j,$$

is minimized, where

$$\Lambda_j = \sum_{i \in I_j} \lambda_i$$

is the total workload in SA$_j$.

Since

$$\lambda_i - \hat{\lambda}_i = (1 - F_j)\lambda_i,$$

and

$$\tilde{\lambda}_j = \sum_{i \in I_j} \hat{\lambda}_i = \sum_{i \in I_j} F_j \lambda_i = F_j \sum_{i \in I_j} \lambda_i = F_j \Lambda_j,$$

the above T can be rewritten as

$$T = \frac{1}{\Lambda_j} \sum_{i \in I_j} (1 - F_j)\lambda_i T_i + F_j \tilde{T}_j,$$

for all $0 \le j \le n-1$.

It is clear that T can be viewed as a function of F_j, which can be minimized by solving the equation

$$\partial T / \partial F_j = 0,$$

where

$$\frac{\partial T}{\partial F_j} = \frac{1}{\Lambda_j} \sum_{i \in I_j} \left(-\lambda_i T_i + (1 - F_j)\lambda_i \frac{\partial T_i}{\partial F_j} \right) + \tilde{T}_j + F_j \frac{\partial \tilde{T}_j}{\partial F_j},$$

for all $0 \le j \le n-1$.

To derive $\partial T_i / \partial F_j$, we notice that

$$\frac{\partial T_i}{\partial F_j} = \frac{\partial W_i}{\partial F_j},$$

where

$$W_i = \frac{(\lambda_i - \hat{\lambda}_i)\overline{x_i^2}}{2(1 - (\lambda_i - \hat{\lambda}_i)\overline{x_i})} = \frac{(1 - F_j)\lambda_i \overline{x_i^2}}{2(1 - (1 - F_j)\lambda_i \overline{x_i})},$$

and

$$\frac{\partial W_i}{\partial F_j} = -\left(\frac{\lambda_i \overline{x_i^2}(1 - (1 - F_j)\lambda_i \overline{x_i}) + (1 - F_j)\lambda_i^2 \overline{x_i}\,\overline{x_i^2}}{2(1 - (1 - F_j)\lambda_i \overline{x_i})^2} \right),$$

for all $0 \le j \le n-1$.

To derive $\partial \tilde{T}_j / \partial F_j$, we notice that $\overline{\tilde{x}_j}$ and c_j are independent of F_j. Therefore,

$$\frac{\partial \tilde{T}_j}{\partial F_j} = \frac{\partial \tilde{W}_j}{\partial F_j} = \left(\frac{c_j^2 + 1}{2} \right) \frac{\partial W_j^*}{\partial F_j},$$

where

$$\frac{\partial W_j^*}{\partial F_j} = \frac{\overline{\tilde{x}_j}}{k_j} \left(\frac{1}{(1 - \tilde{\rho}_j)^2} \cdot \frac{\partial p_{j,k_j}}{\partial F_j} + \frac{2 p_{j,k_j}}{(1 - \tilde{\rho}_j)^3} \cdot \frac{\partial \tilde{\rho}_j}{\partial F_j} \right),$$

and

$$\frac{\partial p_{j,k_j}}{\partial F_j} = \frac{(k_j \tilde{\rho}_j)^{k_j}}{k_j!} \cdot \frac{\partial p_{j,0}}{\partial F_j} + p_{j,0} \frac{k_j^{k_j} \tilde{\rho}_j^{k_j - 1}}{(k_j - 1)!} \cdot \frac{\partial \tilde{\rho}_j}{\partial F_j},$$

and

$$\frac{\partial p_{j,0}}{\partial F_j} = -p_{j,0}^2 \left(\sum_{b=1}^{k_j - 1} \frac{k_j^b \tilde{\rho}_j^{b-1}}{(b-1)!} + \frac{k_j^{k_j} \tilde{\rho}_j^{k_j - 1}}{(k_j - 1)!} \cdot \frac{1}{1 - \tilde{\rho}_j} + \frac{(k_j \tilde{\rho}_j)^{k_j}}{k_j!} \cdot \frac{1}{(1 - \tilde{\rho}_j)^2} \right) \frac{\partial \tilde{\rho}_j}{\partial F_j},$$

for all $0 \le j \le n-1$. Since

$$\tilde{\rho}_j = \frac{\overline{\tilde{x}_j}}{k_j} \tilde{\lambda}_j = \frac{\overline{\tilde{x}_j}}{k_j} F_j \Lambda_j,$$

we have

$$\frac{\partial \tilde{\rho}_j}{\partial F_j} = \frac{\overline{\tilde{x}_j}}{k_j} \Lambda_j,$$

for all $0 \le j \le n-1$.

We observe that $\partial T / \partial F_j$ is an increasing function of F_j. Consequently, the equation $\partial T / \partial F_j = 0$ can be solved by using the standard bisection search algorithm. Since $0 \le \tilde{\rho}_j < 1$, we have $0 \le F_j < k_j / (\overline{\tilde{x}_j} \Lambda_j)$. Also, $F_j \le 1$. Hence, the search interval of F_j is $[0, \min\{1, k_j / (\overline{\tilde{x}_j} \Lambda_j)\}]$. The time complexity of the procedure is $O(\log(1/\epsilon))$.

Solving the equation $f(T) = 0$ in Section 3.2.1 or the equation $\partial T / \partial F_j = 0$ in Section 3.2.2 is the kernel of all our algorithms in this paper. The value T is called the *service area response time* of SA$_j$, denoted as $\widetilde{T}_j$, for all $0 \le j \le n-1$. Also, we notice that $\widetilde{T}_j$ depends on I_j. Therefore, we will represent $\widetilde{T}_j$ as $\widetilde{T}_j(I_j)$.

(Note: One may consider the situation where different UEs in the same service area SA$_j$ have different fractions of workload to offload to MEC$_j$, such that the weighted average response time is minimized. This is a very complicated optimization problem. Since the focus of this paper is not task offloading, we do not investigate along this direction.)

4. Mobility-aware server placement

In this section, we formulate and solve the mobility-aware server placement problem.

4.1. Problem definition

In this section, we define the mobility-aware server placement problem.

Let $\widetilde{T}_j(I_j)$ be the service area response time of SA$_j$ with UEs in I_j and MEC$_j$, where $\widetilde{T}_j(I_j)$ is obtained in Section 3.2. For a location distribution $J = (j_0, \dots, j_i, \dots, j_{m-1})$, let

$$I_j(J) = \{ i \mid j_i = j \}$$

be the set of UEs in SA$_j$, where $0 \le j \le n-1$. The *expected service area response time* $\widetilde{T}_j$ of SA$_j$ is

$$\widetilde{T}_j = \sum_{J=0}^{N-1} \pi(J) \widetilde{T}_j(I_j(J)).$$

The calculation of $\widetilde{T}_j$ needs $O(N \log(1/\epsilon))$ time.

A more efficient way to calculate $\widetilde{T}_j$ is as follows. Consider $I_j \subseteq \{0, 1, \dots, m-1\}$. It is clear that I_j occurs with probability

$$P_j(I_j) = \prod_{i \in I_j} \pi_i(j) \prod_{i \notin I_j} (1 - \pi_i(j))$$

in a stationary mobile edge computing environment. Therefore, $\widetilde{T}_j$ can be calculated by

$$\widetilde{T}_j = \frac{1}{B_j} \sum_{I_j \neq \emptyset} P_j(I_j)\widetilde{T}_j(I_j),$$

where

$$B_j = \sum_{I_j \neq \emptyset} P_j(I_j)$$

is the probability that MEC$_j$ is busy, and

$$B_j = 1 - P_j(\emptyset) = 1 - \prod_{i=0}^{m-1}(1 - \pi_i(j)),$$

for all $0 \le j \le n-1$. Since there are $M = 2^m$ subsets of $\{0, 1, \dots, m-1\}$, the above equation only involves M terms. Consequently, the calculation of $\widetilde{T}_j$ needs $O(M(m + \log(1/\epsilon)))$, or, $O(M \log(1/\epsilon))$ time, where we notice that $\widetilde{T}_j$ has M terms and each term contains $P_j(I_j)$, which can be calculated in $O(m)$ time, and $\widetilde{T}_j(I_j)$, which can be calculated in $O(\log(1/\epsilon))$ time.

Notice that $\widetilde{T}_j$ can be treated as a function $\widetilde{T}_j(k_j)$ of k_j.

Let $K \ge n$ be the number of available servers to be assigned to MEC$_0$, MEC$_2$, ..., MEC$_{n-1}$.

Our *mobility-aware server placement* (MASP) problem is to assign the K servers to the MEC$_j$'s, i.e., to find server sizes $k_0, k_1, \dots, k_{n-1}$, such that $k_j \ge 1$ for all $0 \le j \le n-1$, and

$$k_0 + k_1 + \cdots + k_{n-1} = K,$$

and the *maximum expected service area response time*, i.e.,

$$T = \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$$

is minimized. (Notice that all the $\tilde{s}_j$'s are identical, i.e., $\tilde{s}_0 = \tilde{s}_1 = \cdots = \tilde{s}_{n-1}$.)

4.2. An optimization algorithm

In this section, we develop an optimization algorithm to solve the mobility-aware server placement problem.

4.2.1. The OMASP algorithm

Our *optimal mobility-aware server placement (OMASP) algorithm* is displayed in Algorithm 1.

To reduce the maximum expected service area response time T (lines (5)–(8)), we consider MEC$_{j_1}$, whose service area has the longest expected response time (line (10)), i.e.,

$$\widetilde{T}_{j_1}(k_{j_1}) = \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}.$$

It is clear that k_{j_1} must be increased, so that $\widetilde{T}_{j_1}(k_{j_1})$ is decreased. The extra server comes from another MEC, whose expected service area response time will be increased. It seems that the best choice is MEC$_{j_2}$, whose service area has the shortest expected response time, i.e.,

$$\widetilde{T}_{j_2}(k_{j_2}) = \min\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}.$$

However, the above MEC$_{j_2}$ does not guarantee a reduced maximum expected service area response time T. Therefore, we consider all possible j_2 (line (12)) such that $j_2 \ne j_1$ and $k[j_2] > 1$ (line (13)). One server is moved from MEC$_{j_2}$ to MEC$_{j_1}$ (lines (14)–(15)). Such a new server placement is acceptable only if the maximum expected service area response time T' (line (16)) is shorter than the previous maximum expected service area response time T (lines (17)–(21)).

The above adjustment of server placement is repeatedly conducted (line (9)) until no MEC$_{j_2}$ can result in a shorter T' (lines (11) and (24)). At that time, we find an optimal server placement that gives rise to the shortest maximum expected service area response time T (line (25)).

In the following, we show some important properties of the OMASP algorithm.

Algorithm 1 Optimal mobility-aware server placement (OMASP).

Input: UE$_i$ with λ_i, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, s_i, $c_{i,j}$ for all $0 \le j \le n-1$, $\boldsymbol{P}_i$ or $\boldsymbol{Q}_i$, for all $0 \le i \le m-1$; MEC$_j$ with $\tilde{s}_j$, for all $0 \le j \le n-1$; K.
Output: $k_0, k_1, \dots, k_{n-1}$, such that $k_j \ge 1$ for all $0 \le j \le n-1$, and $k_0 + k_1 + \cdots + k_{n-1} = K$, and $T = \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$ is minimized.

```
for ($i = 0$; $i < m$; $i++$) do                                          (1)
    calculate $\pi_i$;                                                    (2)
end do;                                                                   (3)
set an initial server placement $k_0, k_1, \dots, k_{n-1}$;               (4)
for ($j = 0$; $j < n$; $j++$) do                                          (5)
    calculate $\widetilde{T}_j(k_j)$;                                     (6)
end do;                                                                   (7)
$T \leftarrow \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$;   (8)
repeat                                                                    (9)
    $j_1 \leftarrow \mathrm{argmax}_{0\le j\le n-1}\{\widetilde{T}_j(k_j)\}$;   (10)
    $done \leftarrow$ true;                                               (11)
    for ($j_2 = 0$; $j_2 < n$; $j_2++$) do                                (12)
        if ($j_2 \ne j_1$ and $k[j_2] > 1$)                               (13)
            $k'_{j_1} \leftarrow k_{j_1} + 1$; calculate $\widetilde{T}_{j_1}(k'_{j_1})$;   (14)
            $k'_{j_2} \leftarrow k_{j_2} - 1$; calculate $\widetilde{T}_{j_2}(k'_{j_2})$;   (15)
            $T' \leftarrow \max\{\widetilde{T}_0(k_0), \dots, \widetilde{T}_{j_1}(k'_{j_1}), \dots, \widetilde{T}_{j_2}(k'_{j_2}), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$;
(16)
            if ($T' < T$)                                                 (17)
                $k_{j_1} \leftarrow k'_{j_1}$; $k_{j_2} \leftarrow k'_{j_2}$; $T \leftarrow T'$;   (18)
                $done \leftarrow$ false;                                  (19)
                break;                                                    (20)
            end if;                                                       (21)
        end if;                                                           (22)
    end do;                                                               (23)
until ($done$);                                                           (24)
return $k_0, k_1, \dots, k_{n-1}$.                                        (25)
```

We claim that the OMASP algorithm guarantees to find the optimal server placement.

Claim 1. *For any initial server placement $k_0, k_1, \dots, k_{n-1}$ in line (4), the OMASP algorithm can find the optimal server placement in lines (9)–(24).*

Proof. For any server placement $k_0, k_1, \dots, k_{n-1}$ with $T = \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$, if T can be reduced, there must be some k_1 and k_2, such that one server must be moved from MEC$_{j_2}$ to MEC$_{j_1}$. According to lines (10)–(23), such an improvement can be found and performed by the OMASP algorithm. □

Let $T(K)$ be the maximum expected service area response time for K servers. We claim that more servers guarantee a shorter maximum expected service area response time.

Claim 2. *For any $K < K'$, we have $T(K) > T(K')$.*

Proof. Let $k_0, k_1, \dots, k_{n-1}$ be the optimal server placement for K servers with $T(K) = \max\{\widetilde{T}_0(k_0), \widetilde{T}_1(k_1), \dots, \widetilde{T}_{n-1}(k_{n-1})\}$ and $j^* = \mathrm{argmax}_{0\le j\le n-1}\{\widetilde{T}_j(k_j)\}$. If more (i.e., K') servers are available, we can increase k_{j^*} and reduce $\widetilde{T}_{j^*}(k_{j^*})$, and thus reduce $T(K')$. □

We claim that as K increases, k_j never decreases, for all $0 \le j \le n-1$.

Claim 3. *Let $k_0, k_1, \dots, k_{n-1}$ be the optimal server placement for K servers and $k'_0, k'_1, \dots, k'_{n-1}$ be the optimal server placement for K' servers. If $K < K'$, then $k_j \le k'_j$, for all $0 \le j \le n-1$.*

Proof. Assume that $k'_j < k_j$. We have $\widetilde{T}_j(k'_j) \le T(K')$. Also, by Claim 2, we have $T(K') < T(K)$. Thus, $\widetilde{T}_j(k'_j) < T(K)$. This means that $k_j - k'_j$ servers from MEC$_j$ can be used to reduce $T(K)$. This is certainly not possible, since $k_0, k_1, \dots, k_{n-1}$ is already the optimal server placement. □

Claims 2 and 3 will be manifested in our numerical data.

4.2.2. Time complexity of the OMASP algorithm

It is noticed that the most time-consuming computation in the OMASP algorithm is the calculation of the $\widetilde{T}_j(k_j)$'s. Lines (5)–(7) calculate n $\widetilde{T}_j(k_j)$'s. The main loop in lines (9)–(24) is repeated no more than K times, and in each repetition, we calculate no more than $2n$ (line (12)) $\widetilde{T}_j(k_j)$'s in lines (14)–(15). Since each $\widetilde{T}_j(k_j)$ requires $O(M\log(1/\epsilon))$ time, the time complexity of the OMASP algorithm is $O(KnM\log(1/\epsilon))$.

4.3. Numerical data

In this section, we present some numerical data to illustrate our algorithm for mobility-aware server placement.

4.3.1. Parameter settings

We consider a mobile edge computing environment with $m = 10$ UEs and $n = 5$ MECs. The SAs have the following layout:

SA$_1$		SA$_2$
	SA$_0$	
SA$_3$		SA$_4$

The parameters of our queueing models are set as follows.

For UE$_i$, we set $\overline{r_i} = 1.5 + 0.1i$ BI, $\overline{r_i^2} = 1.1\overline{r_i}^2$ BI^2, $\overline{d_i} = 2.0 + 0.2i$ MB, $\overline{d_i^2} = 1.1\overline{d_i}^2$ MB^2, $s_i = 1.5 + 0.05i$ BI/second, $\overline{x_i} = \overline{r_i}/s_i$ second, $\overline{x_i^2} = \overline{r_i^2}/s_i^2$ second^2, $\lambda_i = 0.99/\overline{x_i}$ tasks/second, for all $0 \le i \le m-1$.

For MEC$_j$, we set $\tilde{s}_j = 2.5$ BI/second, for all $0 \le j \le n-1$; and $c_{i,j} = (10 + i) + 0.5j$ MB/second, for all $0 \le i \le m-1$ and $0 \le j \le n-1$.

Note that each UE$_i$ has utilization $\rho_i = \lambda_i\overline{x_i} = 0.99$, and without task offloading to the MECs (i.e., $\hat{\lambda}_i = 0$), each UE has an average response time of over 55 s:

$$T_0 = 55.45000 \text{ s},$$
$$T_1 = 57.23871 \text{ s},$$
$$T_2 = 58.91562 \text{ s},$$
$$T_3 = 60.49091 \text{ s},$$
$$T_4 = 61.97353 \text{ s},$$
$$T_5 = 63.37143 \text{ s},$$
$$T_6 = 64.69167 \text{ s},$$
$$T_7 = 65.94054 \text{ s},$$
$$T_8 = 67.12368 \text{ s},$$
$$T_9 = 68.24615 \text{ s}.$$

4.3.2. Synchronous mobility

The transition probability matrices are set as follows:

$$\boldsymbol{P}_0 = \boldsymbol{P}_1 = \boldsymbol{P}_2 = \begin{bmatrix} 0.15 & 0.40 & 0.15 & 0.15 & 0.15 \\ 0.25 & 0.50 & 0.25 & 0.00 & 0.00 \\ 0.25 & 0.50 & 0.25 & 0.00 & 0.00 \\ 0.40 & 0.00 & 0.00 & 0.30 & 0.30 \\ 0.40 & 0.00 & 0.00 & 0.30 & 0.30 \end{bmatrix},$$

$$\boldsymbol{P}_3 = \boldsymbol{P}_4 = \boldsymbol{P}_5 = \boldsymbol{P}_6 = \begin{bmatrix} 0.40 & 0.15 & 0.15 & 0.15 & 0.15 \\ 0.40 & 0.30 & 0.30 & 0.00 & 0.00 \\ 0.40 & 0.30 & 0.30 & 0.00 & 0.00 \\ 0.40 & 0.00 & 0.00 & 0.30 & 0.30 \\ 0.40 & 0.00 & 0.00 & 0.30 & 0.30 \end{bmatrix},$$

$$\boldsymbol{P}_7 = \boldsymbol{P}_8 = \boldsymbol{P}_9 = \begin{bmatrix} 0.15 & 0.15 & 0.15 & 0.15 & 0.40 \\ 0.40 & 0.30 & 0.30 & 0.00 & 0.00 \\ 0.40 & 0.30 & 0.30 & 0.00 & 0.00 \\ 0.25 & 0.00 & 0.00 & 0.25 & 0.50 \\ 0.25 & 0.00 & 0.00 & 0.25 & 0.50 \end{bmatrix}.$$

The above matrices are set in such a way that UE$_0$, UE$_1$, and UE$_2$ move more towards SA$_1$; UE$_3$, UE$_4$, UE$_5$, and UE$_6$ move more towards SA$_0$; while UE$_7$, UE$_8$, and UE$_9$ move towards SA$_4$.

The stationary probability vectors of the UE$_i$'s are:

$$\pi_0 = \pi_1 = \pi_2 = [0.25316, 0.37975, 0.17722, 0.09494, 0.09494],$$
$$\pi_3 = \pi_4 = \pi_5 = \pi_6 = [0.40000, 0.15000, 0.15000, 0.15000, 0.15000],$$
$$\pi_7 = \pi_8 = \pi_9 = [0.25316, 0.09494, 0.09494, 0.17722, 0.37975].$$

In Table 1A, we demonstrate numerical data for optimal mobility-aware server placement with the synchronous mobility model and the equal-response-time method for $5 \le K \le 15$.

In Table 1B, we demonstrate numerical data for optimal mobility-aware server placement with the synchronous mobility model and the equal-load-fraction method for $5 \le K \le 15$.

For each K, we show the server size k_j and the expected service area response time $\widetilde{T}_j(k_j)$) for all $0 \le j \le n-1$. We also show the maximum expected service area response time T. We have the following observations. First, more servers are assigned to service areas which are more crowded with UEs (e.g., SA$_0$ and SA$_1$) and service areas where UEs have more demanding computation and communication requirements (e.g., SA$_3$ and SA$_4$). Second, as more servers are available, the maximum expected service area response time becomes shorter, as shown in Claim 2. Third, as K increases, k_j never decreases, for all $0 \le j \le n-1$, as shown in Claim 3. Fourth, the equal-load-fraction method gives a shorter maximum expected service area response time than the equal-response-time method (a strong observation that is hard to prove).

4.3.3. Asynchronous mobility

The transition rate matrices are set as follows:

$$\boldsymbol{Q}_0 = \boldsymbol{Q}_1 = \boldsymbol{Q}_2 = \begin{bmatrix} -0.05 & 0.02 & 0.01 & 0.01 & 0.01 \\ 0.02 & -0.04 & 0.02 & 0.00 & 0.00 \\ 0.02 & 0.03 & -0.05 & 0.00 & 0.00 \\ 0.03 & 0.00 & 0.00 & -0.05 & 0.02 \\ 0.03 & 0.00 & 0.00 & 0.02 & -0.05 \end{bmatrix},$$

$$\boldsymbol{Q}_3 = \boldsymbol{Q}_4 = \boldsymbol{Q}_5 = \boldsymbol{Q}_6 = \begin{bmatrix} -0.04 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.03 & -0.05 & 0.02 & 0.00 & 0.00 \\ 0.03 & 0.02 & -0.05 & 0.00 & 0.00 \\ 0.03 & 0.00 & 0.00 & -0.05 & 0.02 \\ 0.03 & 0.00 & 0.00 & 0.02 & -0.05 \end{bmatrix},$$

$$\boldsymbol{Q}_7 = \boldsymbol{Q}_8 = \boldsymbol{Q}_9 = \begin{bmatrix} -0.05 & 0.01 & 0.01 & 0.01 & 0.02 \\ 0.03 & -0.05 & 0.02 & 0.00 & 0.00 \\ 0.03 & 0.02 & -0.05 & 0.00 & 0.00 \\ 0.02 & 0.00 & 0.00 & -0.05 & 0.03 \\ 0.02 & 0.00 & 0.00 & 0.02 & -0.04 \end{bmatrix}.$$

The above matrices are set in such a way that UE$_0$, UE$_1$, and UE$_2$ move more towards SA$_1$; UE$_3$, UE$_4$, UE$_5$, and UE$_6$ move more towards SA$_0$; while UE$_7$, UE$_8$, and UE$_9$ move towards SA$_4$.

If the unit of time is minutes, then the *mean holding time* is between 20 min and 25 min.

The stationary probability vectors of the UE$_i$'s are:

$$\pi_0 = \pi_1 = \pi_2 = [0.31579, 0.29323, 0.18045, 0.10526, 0.10526],$$
$$\pi_3 = \pi_4 = \pi_5 = \pi_6 = [0.42857, 0.14286, 0.14286, 0.14286, 0.14286],$$
$$\pi_7 = \pi_8 = \pi_9 = [0.31579, 0.10526, 0.10526, 0.18045, 0.29323].$$

In Table 2A, we demonstrate numerical data for optimal mobility-aware server placement with the asynchronous mobility model and the equal-response-time method for $5 \le K \le 15$. In Table 2B, we demonstrate numerical data for optimal mobility-aware server placement with the asynchronous mobility model and the equal-load-fraction method for $5 \le K \le 15$.

We have similar observations to those in Section 4.3.2.

Table 1A
Numerical data for optimal mobility-aware server placement. (synchronous mobility model, equal-response-time method).

K	MEC$_0$ $(k_0, \widetilde{T}_0(k_0))$	MEC$_1$ $(k_1, \widetilde{T}_1(k_1))$	MEC$_2$ $(k_2, \widetilde{T}_2(k_2))$	MEC$_3$ $(k_3, \widetilde{T}_3(k_3))$	MEC$_4$ $(k_4, \widetilde{T}_4(k_4))$	T
5	1	1	1	1	1	
	2.67342	2.04117	1.89189	1.97678	2.26596	2.67342
6	2	1	1	1	1	
	1.66051	2.04117	1.89189	1.97678	2.26596	2.26596
7	2	1	1	1	2	
	1.66051	2.04117	1.89189	1.97678	1.48415	2.04117
8	2	2	1	1	2	
	1.66051	1.38775	1.89189	1.97678	1.48415	1.97678
9	2	2	1	2	2	
	1.66051	1.38775	1.89189	1.36922	1.48415	1.89189
10	2	2	2	2	2	
	1.66051	1.38775	1.33906	1.36922	1.48415	1.66051
11	3	2	2	2	2	
	1.36449	1.38775	1.33906	1.36922	1.48415	1.48415
12	3	2	2	2	3	
	1.36449	1.38775	1.33906	1.36922	1.28873	1.38775
13	3	3	2	2	3	
	1.36449	1.25711	1.33906	1.36922	1.28873	1.36922
14	3	3	2	3	3	
	1.36449	1.25711	1.33906	1.25275	1.28873	1.36449
15	4	3	2	3	3	
	1.26468	1.25711	1.33906	1.25275	1.28873	1.33906
16	4	3	3	3	3	
	1.26468	1.25711	1.24547	1.25275	1.28873	1.28873
17	4	3	3	3	4	
	1.26468	1.25711	1.24547	1.25275	1.23983	1.26468
18	5	3	3	3	4	
	1.23688	1.25711	1.24547	1.25275	1.23983	1.25711
19	5	4	3	3	4	
	1.23688	1.23388	1.24547	1.25275	1.23983	1.25275
20	5	4	3	4	4	
	1.23688	1.23388	1.24547	1.23314	1.23983	1.24547

Table 1B
Numerical data for optimal mobility-aware server placement. (synchronous mobility model, equal-load-fraction method).

K	MEC$_0$ $(k_0, \widetilde{T}_0(k_0))$	MEC$_1$ $(k_1, \widetilde{T}_1(k_1))$	MEC$_2$ $(k_2, \widetilde{T}_2(k_2))$	MEC$_3$ $(k_3, \widetilde{T}_3(k_3))$	MEC$_4$ $(k_4, \widetilde{T}_4(k_4))$	T
5	1	1	1	1	1	
	2.65283	2.02007	1.87413	1.96117	2.24960	2.65283
6	2	1	1	1	1	
	1.59049	2.02007	1.87413	1.96117	2.24960	2.24960
7	2	1	1	1	2	
	1.59049	2.02007	1.87413	1.96117	1.41618	2.02007
8	2	2	1	1	2	
	1.59049	1.26384	1.87413	1.96117	1.41618	1.96117
9	2	2	1	2	2	
	1.59049	1.26384	1.87413	1.27162	1.41618	1.87413
10	2	2	2	2	2	
	1.59049	1.26384	1.21181	1.27162	1.41618	1.59049
11	3	2	2	2	2	
	1.25087	1.26384	1.21181	1.27162	1.41618	1.41618
12	3	2	2	2	3	
	1.25087	1.26384	1.21181	1.27162	1.17127	1.27162
13	3	2	2	3	3	
	1.25087	1.26384	1.21181	1.08289	1.17127	1.26384
14	3	3	2	3	3	
	1.25087	1.04160	1.21181	1.08289	1.17127	1.25087
15	4	3	2	3	3	
	1.10480	1.04160	1.21181	1.08289	1.17127	1.21181
16	4	3	3	3	3	
	1.10480	1.04160	1.03092	1.08289	1.17127	1.17127
17	4	3	3	3	4	
	1.10480	1.04160	1.03092	1.08289	1.07847	1.10480
18	5	3	3	3	4	
	1.03586	1.04160	1.03092	1.08289	1.07847	1.08289
19	5	3	3	4	4	
	1.03586	1.04160	1.03092	1.01888	1.07847	1.07847
20	5	3	3	4	5	
	1.03586	1.04160	1.03092	1.01888	1.04116	1.04160

Table 2A
Numerical data for optimal mobility-aware server placement. (asynchronous mobility model, equal-response-time method).

K	MEC$_0$ $(k_0, \widetilde{T}_0(k_0))$	MEC$_1$ $(k_1, \widetilde{T}_1(k_1))$	MEC$_2$ $(k_2, \widetilde{T}_2(k_2))$	MEC$_3$ $(k_3, \widetilde{T}_3(k_3))$	MEC$_4$ $(k_4, \widetilde{T}_4(k_4))$	T
5	1	1	1	1	1	
	2.90562	1.98190	1.90084	1.97730	2.13688	2.90562
6	2	1	1	1	1	
	1.77013	1.98190	1.90084	1.97730	2.13688	2.13688
7	2	1	1	1	2	
	1.77013	1.98190	1.90084	1.97730	1.43056	1.98190
8	2	2	1	1	2	
	1.77013	1.36956	1.90084	1.97730	1.43056	1.97730
9	2	2	1	2	2	
	1.77013	1.36956	1.90084	1.36964	1.43056	1.90084
10	2	2	2	2	2	
	1.77013	1.36956	1.34265	1.36964	1.43056	1.77013
11	3	2	2	2	2	
	1.41961	1.36956	1.34265	1.36964	1.43056	1.43056
12	3	2	2	2	3	
	1.41961	1.36956	1.34265	1.36964	1.27053	1.41961
13	4	2	2	2	3	
	1.28694	1.36956	1.34265	1.36964	1.27053	1.36964
14	4	2	2	3	3	
	1.28694	1.36956	1.34265	1.25288	1.27053	1.36956
15	4	3	2	3	3	
	1.28694	1.25284	1.34265	1.25288	1.27053	1.34265
16	4	3	3	3	3	
	1.28694	1.25284	1.24635	1.25288	1.27053	1.28694
17	5	3	3	3	3	
	1.24308	1.25284	1.24635	1.25288	1.27053	1.27053
18	5	3	3	3	4	
	1.24308	1.25284	1.24635	1.25288	1.23612	1.25288
19	5	3	3	4	4	
	1.24308	1.25284	1.24635	1.23316	1.23612	1.25284
20	5	4	3	4	4	
	1.24308	1.23324	1.24635	1.23316	1.23612	1.24635

Table 2B
Numerical data for optimal mobility-aware server placement. (asynchronous mobility model, equal-load-fraction method).

K	MEC$_0$ $(k_0, \widetilde{T}_0(k_0))$	MEC$_1$ $(k_1, \widetilde{T}_1(k_1))$	MEC$_2$ $(k_2, \widetilde{T}_2(k_2))$	MEC$_3$ $(k_3, \widetilde{T}_3(k_3))$	MEC$_4$ $(k_4, \widetilde{T}_4(k_4))$	T
5	1	1	1	1	1	
	2.88205	1.96197	1.88283	1.96116	2.11998	2.88205
6	2	1	1	1	1	
	1.70126	1.96197	1.88283	1.96116	2.11998	2.11998
7	2	1	1	1	2	
	1.70126	1.96197	1.88283	1.96116	1.35038	1.96197
8	2	2	1	1	2	
	1.70126	1.24439	1.88283	1.96116	1.35038	1.96116
9	2	2	1	2	2	
	1.70126	1.24439	1.88283	1.27049	1.35038	1.88283
10	2	2	2	2	2	
	1.70126	1.24439	1.21670	1.27049	1.35038	1.70126
11	3	2	2	2	2	
	1.31497	1.24439	1.21670	1.27049	1.35038	1.35038
12	3	2	2	2	3	
	1.31497	1.24439	1.21670	1.27049	1.13022	1.31497
13	4	2	2	2	3	
	1.14297	1.24439	1.21670	1.27049	1.13022	1.27049
14	4	2	2	3	3	
	1.14297	1.24439	1.21670	1.08115	1.13022	1.24439
15	4	3	2	3	3	
	1.14297	1.03931	1.21670	1.08115	1.13022	1.21670
16	4	3	3	3	3	
	1.14297	1.03931	1.03442	1.08115	1.13022	1.14297
17	5	3	3	3	3	
	1.05796	1.03931	1.03442	1.08115	1.13022	1.13022
18	5	3	3	3	4	
	1.05796	1.03931	1.03442	1.08115	1.05013	1.08115
19	5	3	3	4	4	
	1.05796	1.03931	1.03442	1.01674	1.05013	1.05796
20	6	3	3	4	4	
	1.01476	1.03931	1.03442	1.01674	1.05013	1.05013

5. Mobility-aware power allocation

In this section, we formulate and solve the mobility-aware power allocation problem.

5.1. Problem definition

In this section, we define the mobility-aware power allocation problem.

Power consumption models are necessary to study power allocation and speed setting. There are two types of power consumption models.

- *Idle-speed model* (ISM) – The power consumption (measured by Watts) of MEC$_j$ is
$$P_j = \beta k_j(\xi B_j \tilde{\rho}_j \tilde{s}_j^{\alpha} + P^*),$$
for all $0 \le j \le n-1$.
- *Constant-speed model* (CSM) – The power consumption of MEC$_j$ is
$$P_j = \beta k_j(\xi \tilde{s}_j^{\alpha} + P^*),$$
for all $0 \le j \le n-1$.

In the above models, ξ and α are technology-dependent constants that determine the dynamic component of power consumption, P^* is the static component of power consumption, and β is the *power usage effectiveness* (PUE).

Notice that $\widetilde{T}_j$ can be treated as a function $\widetilde{T}_j(\tilde{s}_j)$ of $\tilde{s}_j$.

Let P be the available power to be allocated to MEC$_0$, MEC$_2$, ..., MEC$_{m-1}$.

Our *mobility-aware power allocation* (MAPA) problem is to allocate P to the MEC$_j$'s, i.e., to find server speeds $\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1}$, such that
$$P_0 + P_1 + \cdots + P_{n-1} = P,$$
and the *maximum expected service area response time*, i.e.,
$$T = \max\{\widetilde{T}_0(\tilde{s}_0), \widetilde{T}_1(\tilde{s}_1), \dots, \widetilde{T}_{n-1}(\tilde{s}_{n-1})\}$$
is minimized.

Notice that the determination of $\tilde{s}_j$ is equivalent to the determination of P_j, since
$$\tilde{s}_j = \left(\frac{1}{\xi B_j \tilde{\rho}_j}\left(\frac{P_j}{\beta k_j} - P^*\right)\right)^{1/\alpha}$$
for the idle-speed model, and
$$\tilde{s}_j = \left(\frac{1}{\xi}\left(\frac{P_j}{\beta k_j} - P^*\right)\right)^{1/\alpha}$$
for the constant-speed model, for all $0 \le j \le n-1$. Since $B_j < 1$ and $\tilde{\rho}_j < 1$, for the same P_j, the idle-speed model yields faster $\tilde{s}_j$ than the constant-speed model.

5.2. An optimization algorithm

In this section, we develop an optimization algorithm to solve the mobility-aware power allocation problem.

5.2.1. The OMAPA algorithm

Our *optimal mobility-aware power allocation (OMAPA) algorithm* is displayed in Algorithm 2.

It is clear that T is minimized if and only if
$$\widetilde{T}_0(\tilde{s}_0) = \widetilde{T}_1(\tilde{s}_1) = \cdots = \widetilde{T}_{n-1}(\tilde{s}_{n-1}) = T.$$
Our algorithm is essentially to find T.

For a given T, we can find the $\tilde{s}_j$'s such that $\widetilde{T}_0(\tilde{s}_0) = \widetilde{T}_1(\tilde{s}_1) = \cdots = \widetilde{T}_{n-1}(\tilde{s}_{n-1}) = T$ (lines (6) and (13)). Once the $\tilde{s}_j$'s are available, we can calculate the P_j's (lines (7) and (14)) and compare $P_0 + P_1 + \cdots + P_{n-1}$ with P (lines (9) and (16)). It is noticed that $P_0 + P_1 + \cdots + P_{n-1}$ is a decreasing function of T. Thus, T can be found by using the standard bisection search algorithm (lines (10)–(21)).

The search interval $[T_{lb}, T_{lb}]$ of T is set as follows (lines (1)–(9)). $T_{lb} = 0$ (line (1)). T_{ub} is determined such that if $\widetilde{T}_j(\tilde{s}_j) = T_{ub}$ for all $0 \le j \le n-1$, then $P_0 + P_1 + \cdots + P_{n-1} < P$ (line (9)). The search interval $[T_{lb}, T_{lb}]$ is adjusted (lines (16)–(20)) in each repetition of the loop in lines (10)–(21).

The equation $\widetilde{T}_j(\tilde{s}_j) = T$ in lines (6) and (13) can be solved as follows. Each $\tilde{s}_j$ can be found by using the standard bisection search algorithm, since $\widetilde{T}_j(\tilde{s}_j)$ is a decreasing function of $\tilde{s}_j$, for all $0 \le j \le n-1$. A reasonable search interval $[s_{lb}, s_{lb}]$ of $\tilde{s}_j$ can be set easily.

Algorithm 2 Optimal mobility-aware power allocation (OMAPA).

Input: UE$_i$ with λ_i, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, s_i, $c_{i,j}$ for all $0 \le j \le n-1$, $\boldsymbol{P}_i$ or $\boldsymbol{Q}_i$, for all $0 \le i \le m-1$; MEC$_j$ with k_j, for all $0 \le j \le n-1$; P.
Output: $\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1}$, such that $P_0 + P_1 + \cdots + P_{n-1} = P$, and $T = \max\{\widetilde{T}_0(\tilde{s}_0), \widetilde{T}_1(\tilde{s}_1), \dots, \widetilde{T}_{n-1}(\tilde{s}_{n-1})\}$ is minimized.

```
$T_{lb} \leftarrow 0$;                                              (1)
$T_{ub} \leftarrow$ some initial value;                             (2)
repeat                                                              (3)
    $T_{ub} \leftarrow 2T_{ub}$;                                    (4)
    for ($j = 0$; $j < n$; $j++$) do                                (5)
        find $\tilde{s}_j$ such that $\widetilde{T}_j(\tilde{s}_j) = T_{ub}$;   (6)
        calculate $P_j$;                                            (7)
    end do;                                                         (8)
until ($P_0 + P_1 + \cdots + P_{n-1} < P$);                         (9)
while ($T_{ub} - T_{lb} > \epsilon$) do                             (10)
    $T \leftarrow (T_{lb} + T_{ub})/2$;                             (11)
    for ($j = 0$; $j < n$; $j++$) do                                (12)
        find $\tilde{s}_j$ such that $\widetilde{T}_j(\tilde{s}_j) = T$;        (13)
        calculate $P_j$;                                            (14)
    end do;                                                         (15)
    if ($P_0 + P_1 + \cdots + P_{n-1} > P$)                         (16)
        $T_{lb} \leftarrow T$;                                      (17)
    else                                                            (18)
        $T_{ub} \leftarrow T$;                                      (19)
    end if;                                                         (20)
end do;                                                             (21)
return $\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1}$.          (22)
```

5.2.2. Time complexity of the OMAPA algorithm

Let $\Delta_T = T_{ub} - T_{lb}$ and $\Delta_s = s_{ub} - s_{lb}$ be the lengths of the search intervals and ϵ is the accuracy requirement.

The most time-consuming computation is to solve the equation $\widetilde{T}_j(\tilde{s}_j) = T$. For a given $\tilde{s}_j$, the computation of $\widetilde{T}_j(\tilde{s}_j)$ needs $O(M \log(1/\epsilon))$ time (see Sections 3.2 and 4.1). The standard bisection search algorithm takes
$$O(M \log(1/\epsilon) \log(\Delta_s/\epsilon))$$
time. Since Δ_s is a reasonably small quantity, the above time complexity will be simply treated as $O(M(\log(1/\epsilon))^2)$.

The main loop in lines (10)–(21) is repeated for $O(\log(\Delta_T/\epsilon))$ times. The inner loop in lines (12)–(15) is repeated for n times. In each repetition of the inner loop, we need to solve the equation $\widetilde{T}_j(\tilde{s}_j) = T$, which takes $O(M(\log(1/\epsilon))^2)$ time. Therefore, the time complexity of the OMAPA algorithm is
$$O(Mn \log(\Delta_T/\epsilon)(\log(1/\epsilon))^2).$$
For a reasonably small Δ_T, the above time complexity is simply $O(Mn(\log(1/\epsilon))^3)$.

5.3. Numerical data

In this section, we present some numerical data to illustrate our algorithm for mobility-aware power allocation.

5.3.1. Parameter settings

We adopt the same parameter settings of our queueing models in Section 4.3.1. The server sizes are $k_j = 2$ for all $0 \le j \le n-1$. The parameters of our power consumption models are set as follows: $\xi = 10$, $\alpha = 2$, $P^* = 5$ W, and $\beta = 2$.

5.3.2. Synchronous mobility

We adopt the same parameter settings of our synchronous mobility model in Section 4.3.2.

In Table 3A, we demonstrate numerical data for optimal mobility-aware power allocation with the synchronous mobility model, the equal-response-time method, and the idle-speed model for $P = 800, 900, \dots, 1500$. In Table 3B, we demonstrate numerical data for optimal mobility-aware power allocation with the synchronous mobility model, the equal-response-time method, and the constant-speed model for $P = 800, 900, \dots, 1500$. In Table 4A, we demonstrate numerical data for optimal mobility-aware power allocation with the synchronous mobility model, the equal-load-fraction method, and the idle-speed model for $P = 800, 900, \dots, 1500$. In Table 4B, we demonstrate numerical data for optimal mobility-aware power allocation with the synchronous mobility model, the equal-load-fraction method, and the constant-speed model for $P = 800, 900, \dots, 1500$.

For each P, we show the server speed $\tilde{s}_j$ and the power consumption P_j (rounded to the nearest integer) for all $0 \le j \le n-1$. We also show the maximum expected service area response time T.

We have the following observations.

- First, more powers are allocated to service areas which are more crowded with UEs (e.g., SA$_0$ and SA$_1$) and service areas where UEs have more demanding computation and communication requirements (e.g., SA$_3$ and SA$_4$).
- Second, as more powers P are available, P_j increases, which results in increased $\tilde{s}_j$, for all $0 \le j \le n-1$, and the maximum expected service area response time becomes shorter.
- Third, for the same P_j, the idle-speed model yields faster $\tilde{s}_j$ than the constant-speed model. Thus, the idle-speed model gives a shorter maximum expected service area response time than the constant-speed model.
- Fourth, the equal-load-fraction method gives a shorter maximum expected service area response time than the equal-response-time method. (A rigorous proof of this claim is interesting but challenging.)

5.3.3. Asynchronous mobility

We adopt the same parameter settings of our asynchronous mobility model in Section 4.3.3.

In Table 5A, we demonstrate numerical data for optimal mobility-aware power allocation with the asynchronous mobility model, the equal-response-time method, and the idle-speed model for $P = 800, 900, \dots, 1500$. In Table 5B, we demonstrate numerical data for optimal mobility-aware power allocation with the asynchronous mobility model, the equal-response-time method, and the constant-speed model for $P = 800, 900, \dots, 1500$. In Table 6A, we demonstrate numerical data for optimal mobility-aware power allocation with the asynchronous mobility model, the equal-load-fraction method, and the idle-speed model for $P = 800, 900, \dots, 1500$. In Table 6B, we demonstrate numerical data for optimal mobility-aware power allocation with the asynchronous mobility model, the equal-load-fraction method, and the constant-speed model for $P = 800, 900, \dots, 1500$.

We have similar observations to those in Section 5.3.2.

Table 3A
Numerical data for optimal mobility-aware power allocation. (synchronous mobility model, equal-response-time method, idle-speed model).

P	MEC$_0$	MEC$_1$	MEC$_2$	MEC$_3$	MEC$_4$	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.48925	1.76615	1.67133	1.78787	2.07635	1.66457
800	262	133	108	120	177	
	2.68283	1.88733	1.77467	1.89731	2.21093	1.59750
900	301	149	119	133	197	
	2.86856	2.00145	1.87080	1.99803	2.33376	1.54430
1000	342	165	130	145	218	
	3.04798	2.10672	1.96055	2.09155	2.44896	1.50122
1100	383	181	141	157	238	
	3.22074	2.20461	2.04442	2.17816	2.55998	1.46599
1200	426	196	151	169	258	
	3.38552	2.29872	2.12285	2.25953	2.66779	1.43691
1300	468	212	161	180	278	
	3.54238	2.39089	2.19596	2.33586	2.77359	1.41260
1400	511	228	171	191	299	
	3.69376	2.48093	2.26702	2.40762	2.87412	1.39185
1500	553	243	181	202	320	

Table 3B
Numerical data for optimal mobility-aware power allocation. (synchronous mobility model, equal-response-time method, constant-speed model).

P	MEC$_0$	MEC$_1$	MEC$_2$	MEC$_3$	MEC$_4$	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.33461	1.66991	1.58695	1.69710	1.96532	1.72810
800	238	132	121	135	174	
	2.51658	1.78329	1.68589	1.80349	2.09570	1.65431
900	273	147	134	150	196	
	2.68981	1.89175	1.77834	1.90115	2.21570	1.59530
1000	309	163	147	165	216	
	2.85720	1.99457	1.86493	1.99198	2.32640	1.54729
1100	347	179	159	179	236	
	3.01983	2.09042	1.94633	2.07696	2.43079	1.50751
1200	385	195	172	193	256	
	3.17717	2.17975	2.02383	2.15648	2.53140	1.47436
1300	424	210	184	206	276	
	3.32846	2.26590	2.09587	2.23187	2.62929	1.44653
1400	463	225	196	219	297	
	3.47326	2.34943	2.16356	2.30187	2.72709	1.42294
1500	503	241	207	232	317	

6. Related research

In this section, we review related research on server placement and power allocation. Tables 7 and 8 summarize related work in server placement and power allocation. (Some related surveys can be found in [8–13].)

6.1. Server placement

Numerous researchers have studied server placement in mobile edge computing with various considerations such as access delay minimization, communication latency reduction, energy consumption optimization, network coverage extension, operation cost reduction, response time minimization, and server workload balancing.

Asghari et al. introduced a two-stage static and dynamic model of cloud resource placement using red deer, Markov game, and Q-learning algorithms with consideration of server portability [14]. Asghari et al. proposed a novel energy-aware server placement method using the trees social relations algorithm and the dynamic voltage and frequency scaling technique to extend network coverage [15]. El-Ashmawi et al. developed a new hybrid natural-inspired algorithm by combining the manta ray foraging algorithm and the uniform crossover operator to find the optimal edge server placement which minimizes access delay [16]. He et al. obtained an optimal server configuration scheme and a suboptimal server placement scheme such that the operational expenditures is minimized while the system performance is maintained at a predetermined

Table 4A
Numerical data for optimal mobility-aware power allocation. (synchronous mobility model, equal-load-fraction method, idle-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.42288	1.79025	1.70395	1.82059	2.09271	1.62836
800	250	136	111	124	179	
	2.59046	1.91628	1.82369	1.94629	2.23329	1.54841
900	282	153	124	139	201	
	2.74843	2.03491	1.93607	2.06395	2.36473	1.48039
1000	315	170	138	154	223	
	2.89845	2.14732	2.04223	2.17484	2.48849	1.42144
1100	348	187	151	168	245	
	3.04184	2.25437	2.14302	2.27987	2.60566	1.36960
1200	382	205	164	183	266	
	3.17960	2.35671	2.23910	2.37977	2.71712	1.32348
1300	415	222	177	198	288	
	3.31249	2.45487	2.33097	2.47512	2.82360	1.28205
1400	449	239	190	212	309	
	3.44109	2.54927	2.41910	2.56639	2.92572	1.24453
1500	483	256	204	227	331	

Table 4B
Numerical data for optimal mobility-aware power allocation. (synchronous mobility model, equal-load-fraction method, constant-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.28542	1.68679	1.60537	1.71686	1.97655	1.70102
800	229	134	123	138	176	
	2.44294	1.80535	1.71831	1.83569	2.10960	1.61832
900	259	150	138	155	198	
	2.59138	1.91697	1.82435	1.94698	2.23406	1.54800
1000	289	167	153	172	220	
	2.73225	2.02277	1.92459	2.05194	2.35132	1.48706
1100	319	184	168	188	241	
	2.86674	2.12359	2.01984	2.15147	2.46243	1.43348
1200	349	200	183	205	263	
	2.99581	2.22006	2.11075	2.24626	2.56817	1.38579
1300	379	217	198	222	284	
	3.12022	2.31267	2.19778	2.33684	2.66921	1.34294
1400	409	234	213	238	305	
	3.24057	2.40182	2.28135	2.42365	2.76610	1.30412
1500	440	251	228	255	326	

Table 5A
Numerical data for optimal mobility-aware power allocation. (asynchronous mobility model, equal-response-time method, idle-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.71927	1.69365	1.64990	1.75369	1.90869	1.68696
800	313	119	106	117	146	
	2.93507	1.80156	1.74913	1.85898	2.02712	1.61967
900	361	132	116	129	162	
	3.14059	1.90370	1.84108	1.95516	2.13551	1.56609
1000	410	145	127	140	178	
	3.33721	1.99982	1.92789	2.04433	2.23566	1.52242
1100	461	158	137	152	193	
	3.52628	2.08959	2.00987	2.12879	2.32901	1.48611
1200	512	171	147	163	207	
	3.70883	2.17429	2.08682	2.20829	2.41739	1.45551
1300	564	183	157	174	222	
	3.88473	2.25560	2.15937	2.28239	2.50293	1.42971
1400	617	196	167	184	236	
	4.05430	2.33273	2.22810	2.35303	2.58662	1.40800
1500	670	208	176	194	251	

Table 5B
Numerical data for optimal mobility-aware power allocation. (asynchronous mobility model, equal-response-time method, constant-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.54028	1.60275	1.56521	1.66265	1.80662	1.75355
800	278	123	118	131	151	
	2.74290	1.70531	1.66086	1.76541	1.92180	1.67898
900	321	136	130	145	168	
	2.93623	1.80213	1.74966	1.85954	2.02773	1.61934
1000	365	150	142	158	184	
	3.12179	1.89438	1.83253	1.94636	2.12585	1.57062
1100	410	164	154	172	201	
	3.30002	1.98172	1.91184	2.02786	2.21704	1.53018
1200	456	177	166	184	217	
	3.47247	2.06421	1.98677	2.10516	2.30261	1.49593
1300	502	190	178	197	232	
	3.63993	2.14283	2.05755	2.17814	2.38357	1.46656
1400	550	204	189	210	247	
	3.80116	2.21723	2.12538	2.24778	2.46280	1.44154
1500	598	217	201	222	263	

Table 6A
Numerical data for optimal mobility-aware power allocation. (asynchronous mobility model, equal-load-fraction method, idle-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.62303	1.73706	1.69457	1.79914	1.94927	1.64136
800	292	124	110	122	151	
	2.80416	1.85910	1.81317	1.92302	2.08096	1.56118
900	331	139	124	136	170	
	2.97510	1.97375	1.92434	2.03886	2.20396	1.49303
1000	370	155	137	151	188	
	3.13765	2.08213	2.02923	2.14791	2.31965	1.43402
1100	409	170	150	165	206	
	3.29314	2.18514	2.12870	2.25111	2.42903	1.38218
1200	449	185	163	180	224	
	3.44257	2.28345	2.22345	2.34922	2.53295	1.33607
1300	489	200	176	194	242	
	3.58670	2.37762	2.31403	2.44283	2.63211	1.29465
1400	529	215	189	208	259	
	3.72612	2.46810	2.40090	2.53245	2.72711	1.25713
1500	569	230	202	222	277	

Table 6B
Numerical data for optimal mobility-aware power allocation. (asynchronous mobility model, equal-load-fraction method, constant-speed model).

P	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4	T
	$(\tilde{s}_0, P_0)$	$(\tilde{s}_1, P_1)$	$(\tilde{s}_2, P_2)$	$(\tilde{s}_3, P_3)$	$(\tilde{s}_4, P_4)$	
	2.46627	1.63108	1.59138	1.69111	1.83431	1.71855
800	263	126	121	134	155	
	2.63583	1.74570	1.70298	1.80793	1.95862	1.63539
900	298	142	136	151	173	
	2.79573	1.85344	1.80767	1.91728	2.07486	1.56472
1000	333	157	151	167	192	
	2.94765	1.95538	1.90655	2.02033	2.18430	1.50353
1100	368	173	165	183	211	
	3.09287	2.05234	2.00042	2.11798	2.28791	1.44976
1200	403	188	180	199	229	
	3.23236	2.14496	2.08993	2.21091	2.38643	1.40194
1300	438	204	195	216	248	
	3.36690	2.23375	2.17557	2.29967	2.48047	1.35899
1400	473	220	209	232	266	
	3.49706	2.31912	2.25779	2.38473	2.57055	1.32008
1500	509	235	224	247	284	

level [17]. Kasi et al. adopted a multi-agent reinforcement learning approach to mobile edge server placement to minimize network latency and balance load on edge servers [18]. Li et al. studied the optimal placement of edge servers on access points in heterogeneous networks with combined consideration of response delay and energy consumption [19]. Li and Wang formulated energy-aware edge server placement as a multi-objective optimization problem and devised a particle swarm optimization-based algorithm to find the optimal solution [20]. Liu et al. attempted to propose an effective edge server placement strategy, taking both communication delay and workload balance into account, and arrived at the appropriate placement method by utilizing machine learning techniques [21]. Wang et al. formulated edge server placement as a multi-objective constraint optimization problem to balance the

Table 7
Summary of related research in server placement.

Ref.	Objective	Technique
[14]	latency reduction, load balancing, and resource reduction	Markov game and Q-learning algorithms
[15]	network coverage extension, energy consumption optimization, and network latency reduction	the trees social relations algorithm and the dynamic voltage and frequency scaling technique
[16]	access delay minimization	a new hybrid natural-inspired algorithm combining the manta ray foraging algorithm and the uniform crossover operator
[17]	operational expenditures minimization	queueing model, bisection algorithm, and genetic algorithm
[18]	network latency minimization and load balancing on edge servers	a multi-agent reinforcement learning approach
[19]	response delay and energy consumption optimization	an adaptive clustering algorithm
[20]	multi-objective optimization	a particle swarm optimization-based algorithm
[21]	combined consideration of communication delay and workload balance	machine learning techniques
[22]	workload balancing and access delay minimization	mixed integer programming
[23]	social media services in industrial cognitive Internet of vehicles	a collaborative method
[24]	transmission delay reduction and server workload balancing	a two-layer graph convolutional network and a differentiable version of K-means clustering
[25]	considerations of QoS latency, computation demands, and rental costs	integer linear programming, a heuristic algorithm

Table 8
Summary of related research in power allocation.

Ref.	Objective	Technique
[26]	presentation of an energy model	evaluation of the energy consumption of different cloud-related architectures
[27]	energy consumption minimization	a Bayesian learning framework
[28]	energy-saving management	a game-theoretical approach
[29]	power consumption minimization	switching the edge servers on and off based on provisioned application needs
[30]	minimization of energy consumption of an edge computing infrastructure	shutting down edge servers during low-demand periods
[31]	presentation of a network simulator focused on energy consumption	testing energy-saving approaches and task placement algorithms
[32]	proposal of an entropy-based power modeling framework	analyzing the major components of an edge server and selecting appropriate parameters
[33]	achieving an efficient plan	an adaptive and decentralized approach
[34]	presentation of a maintenance strategy	reducing power consumption during edge server maintenance
[35]	proposal of an edge intelligent energy modeling approach	joint adoption of Elman neural network and feature selection

workload on edge servers and minimize the access delay, and solved the problem using mixed integer programming [22]. Xu et al. developed a collaborative method for quantification and placement of edge servers for social media services in industrial cognitive Internet of vehicles [23]. Zhang et al. investigated edge server placement to reduce transmission delay and balance server workload by using a two-layer graph convolutional network and a differentiable version of K-means clustering [24]. Zheng et al. studied the efficient placement of edge servers with considerations of QoS latency, computation demands, and rental costs by utilizing integer linear programming, and also proposed a heuristic algorithm with $(\ln n + 1)$-approximation boundary for a size n mobile edge network [25].

6.2. Power allocation

Several researchers have considered the energy efficiency of edge servers from various perspectives such as energy consumption modeling, energy consumption management, and energy consumption minimization.

Ahvar et al. presented an energy model to evaluate the energy consumption of different cloud-related architectures, comprising the full energy consumption of the computing facilities, including cooling systems and network devices [26]. Ayala-Romero et al. established a Bayesian learning framework for jointly configuring services and a radio access network, aiming to minimize the total energy consumption while respecting desirable accuracy and latency thresholds [27]. Cui et al. formulated an energy-efficient edge server management problem and proposed a game-theoretical approach to addressing the problem [28]. Daraghmeh et al. proposed a linear power model to measure the energy consumption of edge servers, and introduced a simple dynamic power management model to minimize power consumption by switching the edge servers on and off based on provisioned application needs [29]. Gómez et al. formulated an optimal orchestration policy to minimize the energy consumption of an edge computing infrastructure and presented a strategy with a polynomial time complexity that reduces the operational energy footprint of edge computing by shutting down edge servers during low-demand periods [30]. Gómez et al. presented a network simulator focused on the energy consumption of edge computing, that allows testing energy-saving approaches and task placement algorithms in realistic large-scale scenarios encompassing entire regions [31]. Li and Li proposed an entropy-based power modeling framework, which weights and combines classical prediction models by analyzing the major components of an edge server and selecting appropriate parameters [32]. Morshedlou and Tajari took an adaptive and decentralized approach for more energy efficiency in edge environments, where edge servers collaborate with each other to achieve an efficient plan [33]. Rubin et al. presented a maintenance strategy that reduces power consumption during edge server maintenance without excessively extending maintenance time or increasing application latency [34]. Zhou et al. proposed an edge intelligent energy modeling approach that jointly adopts Elman neural network and feature selection to optimize the energy consumption on edge servers [35].

6.3. Comments on existing research

Despite the above research, there has been little investigation on power allocation to edge servers across different service areas to optimize the overall performance of all edge servers. Furthermore, there has been little study on mobility-aware server placement and power allocation in mobile edge computing. The work in this paper remedies such a situation and makes significant progress in this area.

7. Concluding remarks

7.1. Summary of the work

We have addressed the problems of optimal mobility-aware server placement and optimal mobility-aware power allocation in mobile edge computing environments with randomly walking mobile users. Our formal discussion is based on two queueing systems for UE and MEC modeling, two Markov chains for mobility modeling, two task offloading methods, and two power consumption models. We analytically define our optimization problems and algorithmically solve these problems. We have also presented extensive numerical data to illustrate optimal mobility-aware server placement and optimal mobility-aware power allocation. Our models and methodologies developed in this paper can be very useful guidelines for proper server placement and power allocation, which are mandates and must for infrastructure deployment, energy-efficiency management, cost reduction, and performance optimization in mobile edge computing.

7.2. A note On applications

Although the primary purpose of this paper is to conduct an analytical and algorithmic discussion of optimal mobility-aware server placement and optimal mobility-aware power allocation for mobile edge computing environments with randomly walking mobile users by using mathematical models such as queueing systems and Markov chains, we would like to mention that the models and methods developed in this paper are readily applicable to various real-world application areas (e.g., augmented and virtual reality, autonomous vehicles, enhanced gaming, real-time healthcare, smart transportation, smart cities, and so on), where all the necessary parameters of our models are available.

7.3. Further research

Expected Maximum Service Area Response Time. Both the MASP and the MAPA problems use the *maximum expected service area response time* as the performance measure to optimize. We can also consider the *expected maximum service area response time*, which is defined as

$$T = \sum_{J=0}^{N-1} \pi(J) \max\{\widetilde{T}_0(I_0(J)), \widetilde{T}_1(I_1(J)), \dots, \widetilde{T}_{n-1}(I_{n-1}(J))\}.$$

The above T can be treated as a function $T(k_0, k_1, \dots, k_{n-1})$ of $k_0, k_1, \dots, k_{n-1}$, or a function $T(\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1})$ of $\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1}$. The function $T(k_0, k_1, \dots, k_{n-1})$ ($T(\tilde{s}_0, \tilde{s}_1, \dots, \tilde{s}_{n-1})$, respectively) is the performance measure to optimize for MASP (MAPA, respectively). These optimization problems seem much more difficult to solve for the expected maximum service area response time.

Different-Load-Fraction Method. As mentioned in Section 3.2.2, we can consider the *different-load-fraction* (DLF) method. The weighted average response time of all UEs and the MEC in the same service area can be optimized by letting each UE decide on a fraction of the workload to offload to the MEC. It is clear that such a minimized weighted average response time may be shorter than that using the equal-load-fraction method. However, such optimization involves solving multiple complicated nonlinear equations simultaneously. An effective and efficient algorithm needs to be developed.

Joint Optimization. In this paper, we have formulated and solved the problems of optimal mobility-aware server placement (for given server speeds) and optimal mobility-aware power allocation (for given server placement) separately. Actually, we can define the *mobility-aware server placement and power allocation* (MASPPA) problem, i.e., to find server placement and server speeds simultaneously, such that the total number of servers does not exceed a given number of servers, the total power consumption does not exceed certain given power consumption, and the maximum expected service area response time or the expected maximum service area response time is minimized. Such joint server size and server speed optimizations are worth further investigation. However, the problem seems quite challenging. Notice that there are $\binom{K-1}{n-1}$ different placements of K servers on n multiserver systems. For instance, if $K = 20$ and $n = 5$, we have $\binom{19}{4} = 3876$. Thus, extensive search in the solution space is inherently inefficient.

CRediT authorship contribution statement

Keqin Li: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The author appreciates the anonymous reviewers for their comments and suggestions on improving the manuscript.

References

[1] Z. He, Y. Xu, K. Li, Resource Allocation and Placement in Multi-Access Edge Computing, Springer, 2024. Resource Management in Distributed Systems.
[2] T.E. Abdoulabbas, S.M. Mahmoud, Power consumption and energy management for edge computing: state of the art, TELKOMNIKA Telecommun. Comput. Electr. Contr. 21 (4) (2023) 836–845.
[3] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, K. Li, A survey of hierarchical energy optimization for mobile edge computing: a perspective from end devices to the cloud, ACM Comput. Surv. 53 (2) (2021) 2055394.
[4] B. Hu, J. Gao, Y. Hu, H. Wang, J. Liu, Overview of energy consumption optimization in mobile edge computing, J. Phys. Conf. Ser. 2209 (2021) 2022.
[5] K. Li, Performance modeling and analysis for randomly walking mobile users with Markov chains, J. Comput. Syst. Sci. 140 (2024) 103492.
[6] L. Kleinrock, Queueing Systems, 1, Theory, John Wiley and Sons, New York, New York, 1975.
[7] https://en.wikipedia.org/wiki/M/G/k_queue.
[8] S. Abedi, M. Ghobaei-Arani, E. Khorami, M. Mojaradd, Dynamic resource allocation using improved firefly optimization algorithm in cloud environment, Appl. Artif. Intell. 36 (1) (2022) 38.
[9] A. Ebrahimi, M. Ghobaei-Arani, H. Saboohi, Cold start latency mitigation mechanisms in serverless computing: taxonomy, review, and future directions, J. Syst. Archit. 151 (2024) 103115.
[10] M. Ghorbian, M. Ghobaei-Arani, A survey on the cold start latency approaches in serverless computing: an optimization-based perspective, Computing 106 (2024) 3755–3809.
[11] M. Ghorbian, M. Ghobaei-Arani, Function Offloading Approaches in Serverless Computing: A Survey, 120, Part C, 2024.
[12] M. Ghorbian, M. Ghobaei-Arani, R. Asadolahpour-Karimi, Function placement approaches in serverless computing: a survey, J. Syst. Archit. 157 (2024) 103291.
[13] M. Tari, M. Ghobaei-Arani, J. Pouramini, M. Ghorbian, Auto-scaling mechanisms in serverless computing: a comprehensive review, Comput. Sci. Rev. 53 (2024) 100650.
[14] A. Asghari, A. Vahdani, H. Azgomi, A. Forestiero, Dynamic edge server placement in mobile edge computing using modified red deer optimization algorithm and markov game theory, J. Ambient Intell. Humaniz. Comput. 14 (2023) 12297–12315.
[15] A. Asghari, H. Azgomi, A.A. Zoraghchian, A. Barzegarinezhad, Energy-aware server placement in mobile edge computing using trees social relations optimization algorithm, J. Supercomput. 80 (2024) 6382–6410.
[16] W.H. El-Ashmawi, A.F. Ali, A. Ali, A modified manta ray foraging algorithm for edge server placement in mobile edge computing, Int. J. Inf. Technol. Decis. Making 23 (4) (2024) 1703–1739.
[17] Z. He, K. Li, K. Li, Cost-efficient server configuration and placement for mobile edge computing, IEEE Trans. Parallel Distrib. Syst. 33 (9) (2022) 2198–2212.

[18] M.K. Kasi, S.A. Ghazalah, R.N. Akram, D. Sauveron, Secure mobile edge server placement using multi-agent reinforcement learning, Electronics 10 (17) (2021) 2098.

[19] B. Li, P. Hou, H. Wu, R. Qian, H. Ding, Placement of edge server based on task overhead in mobile edge computing environment, Trans. Emerging Telecommun. Technol. 32 (9) (2021) 4196.

[20] Y. Li, S. Wang, An energy-aware edge server placement algorithm in mobile edge computing, in: IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2018, pp. 2–7.

[21] H. Liu, S. Wang, H. Huang, Q. Ye, On the placement of edge servers in mobile edge computing, in: International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 2023, pp. 20–22.

[22] S. Wang, Y. Zhao, J. Xu, J. Yuan, C.-H. Hsu, Edge server placement in mobile edge computing, J. Parallel Distrib. Comput. 127 (2019) 160–168.

[23] X. Xu, B. Shen, X. Yin, M.R. Khosravi, H. Wu, L. Qi, S. Wan, Edge server quantification and placement for offloading social media services in industrial cognitive IoV, IEEE Trans. Ind. Inf. 17 (4) (2021) 2910–2918.

[24] S. Zhang, J. Yu, M. Hu, An edge server placement based on graph clustering in mobile edge computing, Sci. Rep. 14 (2024) 29986.

[25] D. Zheng, C. Peng, X. Cao, On the placement of edge server for mobile edge computing, in: 7th International Conference on Computer and Communications (ICCC), Chengdu, China, 2021, pp. 10–13.

[26] E. Ahvar, A.-C. Orgerie, A. Lebre, Estimating energy consumption of cloud, fog, and edge computing infrastructures, IEEE Trans. Sustainable Comput. 7 (2) (2022) 277–288.

[27] J.A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, G. Iosifidis, EdgeBOL: automating energy-savings for mobile edge AI, in: Proceedings of the 17th International Conference on Emerging Networking Experiments and Technologies, the 17th International Conference on Emerging Networking Experiments and Technologies-Germany, 2021, pp. 7–10.

[28] G. Cui, Q. He, X. Xia, F. Chen, Y. Yang, Energy-efficient edge server management for edge computing: a game-theoretical approach, in: Proceedings of the 51st International Conference on Parallel Processing, the 51st International Conference on Parallel ProcessingBordeaux, France, 2022.

[29] M. Daraghmeh, I.A. Ridhawi, M. Aloqaily, Y. Jararweh, A. Agarwal, A power management approach to reduce energy consumption for edge computing servers, in: Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Rome, Italy, 2019, pp. 10–13.

[30] B. Gómez, S. Bayhan, E. Coronado, J. Villalón, A. Garrido, LESS-ON: load-aware edge server shutdown for energy saving in cellular networks, Comput. Netw. 252 (2024) 110675.

[31] B. Gómez, E. Coronado, J. Villalón, A. Garrido, Energy-focused simulation of edge computing architectures in 5G networks, J. Supercomput. 80 (2024) 12564–12584.

[32] G. Li, J. Li, EWM: an entropy-based framework for estimating energy consumption of edge servers, Eng. Rep. 6 (5) (2024) 29986.

[33] H. Morshedlou, A.R. Tajari, A new adaptive approach for efficient energy consumption in edge computing, J. Artif. Intell. 11 (1) (2023) 149–159.

[34] F. Rubin, P. Souza, T. Ferreto, Reducing power consumption during server maintenance on edge computing infrastructures, in: Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, the 38th ACM/SIGAPP Symposium on Applied ComputingTallinn, Estonia, 2023, pp. 27–31.

[35] Z. Zhou, M. Shojafar, J. Abawajy, H. Yin, H. Lu, ECMS: an edge intelligent energy efficient model in mobile edge computing, IEEE Trans. Green Commun. Netw. 6 (1) (2022) 238–247.

Keqin Li received a BS degree in computer science from Tsinghua University in 1985 and a PhD degree in computer science from the University of Houston in 1990. He is a SUNY Distinguished Professor at the State University of New York and a National Distinguished Professor at Hunan University (China). He has authored or co-authored more than 1200 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top few most influential scientists in parallel and distributed computing, regarding single-year impact (ranked #2) and career-long impact (ranked #3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023-2025) and is among the top 0.02% out of over 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022-2025) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022-2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS Fellow, an IEEE Fellow, an AAIA Fellow, an ACIS Fellow, and an AIIA Fellow. He is a Member of the European Academy of Sciences and Arts. He is a Member of Academia Europaea (Academician of the Academy of Europe).