Journal of Computer and System Sciences 140 (2024) 103492

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

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

Performance modeling and analysis for randomly walking mobile users with Markov chains

Keqin Li

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

ARTICLE INFO

Article history:
Received 30 June 2022
Received in revised form 15 August 2023
Accepted 4 October 2023
Available online 18 October 2023

Keywords:
Average response time
Computation offloading
Markov chain
Mobility
Performance prediction
Queueing system
Random walk

ABSTRACT

We treat user equipments (UEs) and mobile edge clouds (MECs) as M/G/1 queueing systems, which are the most suitable, powerful, and manageable models. We propose a computation offloading strategy which can satisfy all UEs served by an MEC and develop an efficient method to find such a strategy. We use discrete-time Markov chains, continuous-time Markov chains, and semi-Markov processes to characterize the mobility of UEs, and calculate the joint probability distribution of the locations of UEs at any time. We extend our Markov chains to incorporate mobility cost into consideration, and are able to obtain the average response time of a UE with location change penalty. We can algorithmically predict the overall average response time of tasks generated on a UE and also demonstrate numerical data and examples. We consider the power constrained MEC speed setting problem and develop an algorithm to solve the problem for two power consumption models.

1. Introduction

1.1. Motivation

Mobile edge computing (i.e., multi-access edge computing) enables cloud computing capabilities and service environments available at the edge of any network. By running applications closer to the cellular customers, network congestion can be reduced and applications can perform better with shorter response time. Mobile edge computing has been extensively used in collaborative computing, connected cars, content delivery, edge video caching, healthcare, mobile big data analytics, service function chaining, smart enterprises, smart grids, and smart venues.

There are multiple challenges in modeling and analyzing the performance of mobile user equipments (UEs) in a mobile edge computing (also called fog computing) environment with multiple heterogeneous mobile edge clouds (MECs). Adequate and analytical formulation of UEs and MECs with dynamicity, interaction, mobility, randomness, and cost using mathematically tractable models and methods is critical in performance analysis and prediction for mobile users. (1) Dynamicity – A UE has an infinite sequence of tasks dynamically generated for the UE to process, i.e., a UE does not just process a finite and static set of tasks. (2) Interaction – There are multiple UEs which share the computing and communication resources in the MECs. An MEC does not just serve one UE, but many UEs. (3) Mobility – UEs move among the MECs' service areas, i.e., the location of a UE can change, and a UE may request for services from different MECs at different times. (4) Randomness – Task computation and communication requirements, task execution times, and the movement of UEs are all random

E-mail address: lik@newpaltz.edu.

https://doi.org/10.1016/j.jcss.2023.103492

Table 1
Overview of mathematical models and analytical methods.

Challenge	Model and Method
Mobility and Cost	Markov chains: DTMC, CTMC, SMP, and extensions, Performance prediction
Dynamicity and Randomness	Queueing systems: M/M/1, M/G/1, M/M/m, M/G/m, Multi-variable optimization, Randomized online offloading
Interaction (Computation offloading)	Non-cooperative games, Combinatorial optimization, Lyapunov optimization

and unknown in advance. (5) Cost – When a UE moves from one MEC to another, it may incur mobility cost for service adjustment, i.e., temporarily losing service from the MECs for certain amount of time.

Table 1 gives an overview of the mathematical models and analytical methods that have been adopted to handle the various challenges mentioned above, including non-cooperative games [5,10,14,16,20], combinatorial optimization [17,18], and Lyapunov optimization [7] to deal with interaction and computation offloading; queueing systems [15], multi-variable optimization [8,9], and randomized online offloading [19] to handle dynamicity and randomness. However, very little model and method has been developed for mobility and cost. The motivation of this paper is to construct the model of Markov chains and to apply the method of performance prediction for mobile UEs, which are novel in the literature.

1.2. Contributions

In this paper, we model, analyze, and predict the performance of mobile users in fog computing using queueing systems and Markov chains. We consider a mobile edge computing environment with multiple mobile UEs and multiple heterogeneous MECs. The main contributions of the paper are summarized as follows.

- First, to handle task dynamicity and randomness, we treat the UEs and the MECs as M/G/1 queueing systems, which are the most suitable and powerful and manageable models that are able to capture and characterize task infinity and stochasticity.
- Second, for task interaction, we propose a computation offloading strategy which can satisfy all UEs served by an MEC in the sense that all UEs and the MEC have the same average task response time, and develop an efficient method to find such a strategy.
- Third, for random movement, we use discrete-time Markov chains, continuous-time Markov chains, and semi-Markov processes to characterize the mobility of UEs, and are able to calculate the joint probability distribution of the locations of UEs at any time.
- Fourth, we extend our Markov chains to incorporate mobility cost into consideration, and are able to obtain the average response time of a UE with location change penalty, and to examine the impact of deterministic and probabilistic transition time on performance.
- Fifth, based on the above analytical results, we can algorithmically predict the overall average response time of tasks generated on a UE averaged over all UE distributions and times, and also demonstrate numerical data and examples.
- Sixth, we consider the power constrained MEC speed setting problem and develop an algorithm to solve the problem for two power consumption models, which is applicable to all UE mobility models, as well as the random location distribution model.

To the best of the author's knowledge, this is the first paper which adopts Markov chains to formulate random mobility of UEs and at the same time, applies queueing systems to predict the performance of randomly mobile UEs. The paper makes tangible contributions to accurate and analytical performance prediction for randomly walking mobile users in fog computing based on solid and rigorous mathematical models and methods.

The paper is organized as follows. In Section 2, we describe our mathematical models. In Section 3, we develop our analytical methods. In Section 4, we predict the performance of randomly walking mobile UEs. In Section 5, we demonstrate some numerical data and examples. In Section 6, we derive performance predictions in closed-form for homogeneous UEs and MECs. In Section 7, we consider mobility cost and service delay for location change. In Section 8, we discuss speed setting for MECs with power consumption constraint. In Section 9, we mention extensibility of our models and methods. In Section 10, we comment on some related work. Finally, we conclude the paper in Section 11.

2. Mathematical models

In this section, we describe our mathematical models. Table 2 provides a list of notations and their definitions used in this paper.

2.1. Queueing systems

In this section, we present queueing models for UEs and MECs.

Table 2
Summary of notations and definitions.

Notation	Definition
m	the number of mobile user equipments
n	the number of mobile edge clouds
λ_i	the task arrival rate of UE_i
$\hat{\lambda}_i$	the arrival rate of the substream of tasks of UE_i processed on an MEC
r_i	the random execution requirement of a task generated on UE_i
d_i	the random amount of data to be communicated between UE_i and an MEC
s_i	the computation speed of UE_i
x_i	the random task execution time on UE_i
ρ_i	the utilization of UE_i
W_i	the average task waiting time of UE_i
T_i	the average task response time of UE_i
I_j	the set of indices of UEs at MEC_j
$\tilde{\lambda}_j$	the task arrival rate of MEC_j
$\tilde{s}_j$	the computation speed of MEC_j
$c_{i,j}$	the communication speed between UE_i and MEC_j
$\tilde{x}_j$	the random task execution time on MEC_j
$\tilde{\rho}_j$	the utilization of MEC_j
$\tilde{W}_j$	the average task waiting time of MEC_j
$\tilde{T}_{i,j}$	the average response time of tasks offloaded from UE_i to MEC_j
$\tilde{T}_j$	the average task response time of MEC_j
$p_i(j, j')$	the transition probability of UE_i from MEC_j to $MEC_{j'}$
$\boldsymbol{P}_i$	the $n \times n$ transition probability matrix of UE_i
J	$= (j_0, j_1, ..., j_{m-1})$, a distribution of the UEs, where UE_i is at MEC_{j_i}
$p(J, J')$	the transition probability from J to J'
$\boldsymbol{P}$	the joint $n^m \times n^m$ transition probability matrix of m UEs
$q_i(j, j')$	the transition rate of UE_i from MEC_j to $MEC_{j'}$
$\boldsymbol{Q}_i$	the $n \times n$ transition rate matrix of UE_i
$q(J, J')$	the transition rate from J to J'
$\boldsymbol{Q}$	the joint $n^m \times n^m$ transition rate matrix of m UEs
$\tau_{i,j}$	the random sojourn time of UE_i at MEC_j
$\pi_i^{(t)}(j)$	the probability that UE_i is at MEC_j at time t
$\pi_i^{(t)}$	$= [\pi_i^{(t)}(0), \pi_i^{(t)}(1), ..., \pi_i^{(t)}(n-1)]$, the probability vector of UE_i at time t
π_i	$= [\pi_i(0), \pi_i(1), ..., \pi_i(n-1)]$, the stationary probability vector of UE_i
$\pi^{(t)}(J)$	the probability of distribution J at time t
$\pi^{(t)}$	$= [\pi^{(t)}(0), \pi^{(t)}(1), ..., \pi^{(t)}(N-1)]$, the joint probability vector at time t
π	$= [\pi(0), \pi(1), ..., \pi(N-1)]$, the joint stationary probability vector
$T_i(J)$	the average response time of tasks generated on UE_i under J
$T_i^{(t)}$	the instantaneous average response time of tasks generated on UE_i at time t
T_i^*	the overall average response time of tasks generated on UE_i
$T_i^{(\infty)}$	the stationary average response time of tasks generated on UE_i
Δ	mobility cost, transition time, service delay, and location change penalty
$\tilde{T}_j^{(t)}$	the instantaneous average response time of MEC_j at time t
$\tilde{T}_j^{(\infty)}$	the stationary average response time of MEC_j
$\xi_j, \alpha_j, P_j^*, \beta_j$	parameters of power consumption models
$B_j^{(t)}$ (B_j)	the probability that MEC_j is busy at time t (in a stationary state)

We consider a mobile edge computing environment with multiple UEs and multiple MECs. There are m heterogeneous UEs: UE_0, UE_1, ..., UE_{m-1}, and there are n heterogeneous MECs: MEC_0, MEC_1, ..., MEC_{n-1}. Each UE and MEC is modeled as an M/G/1 queueing system. Fig. 1 shows UE_{i_1}, UE_{i_2}, ..., UE_{i_b}, which are in the service area of MEC_j. Notice that the M/G/1 queueing systems for UE_{i_1}, UE_{i_2}, ..., UE_{i_b}, and MEC_j are not tandem, in the sense that a task received by UE_{i_k} is either processed on UE_{i_k}, or instantly offloaded to MEC_j and processed on MEC_j (see Section 2.2).

Each UE_i has a Poisson stream of tasks with arrival rate λ_i (measured by the number of tasks per second). Task computation requirements are independent and identically distributed (i.i.d.) random variables r_i (measured by the number of billion instructions (BI)) with mean $\overline{r_i}$ and second moment $\overline{r_i^2}$. Task communication requirements are i.i.d. random variables d_i (measured by the number of million bits (MB)) with mean $\overline{d_i}$ and second moment $\overline{d_i^2}$. Therefore, our model can accommodate an infinite sequence of dynamically and stochastically generated random tasks, instead of a finite collection of deterministic and known tasks.

(Notation: $\overline{z}$ and $\overline{z^2}$ stand for the mean and the second moment of a random variable z.)

The computation speed of UE_i is s_i (measured by BI/second). If a task generated on UE_i is executed locally on UE_i, the execution time (measured by second) is a random variable

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

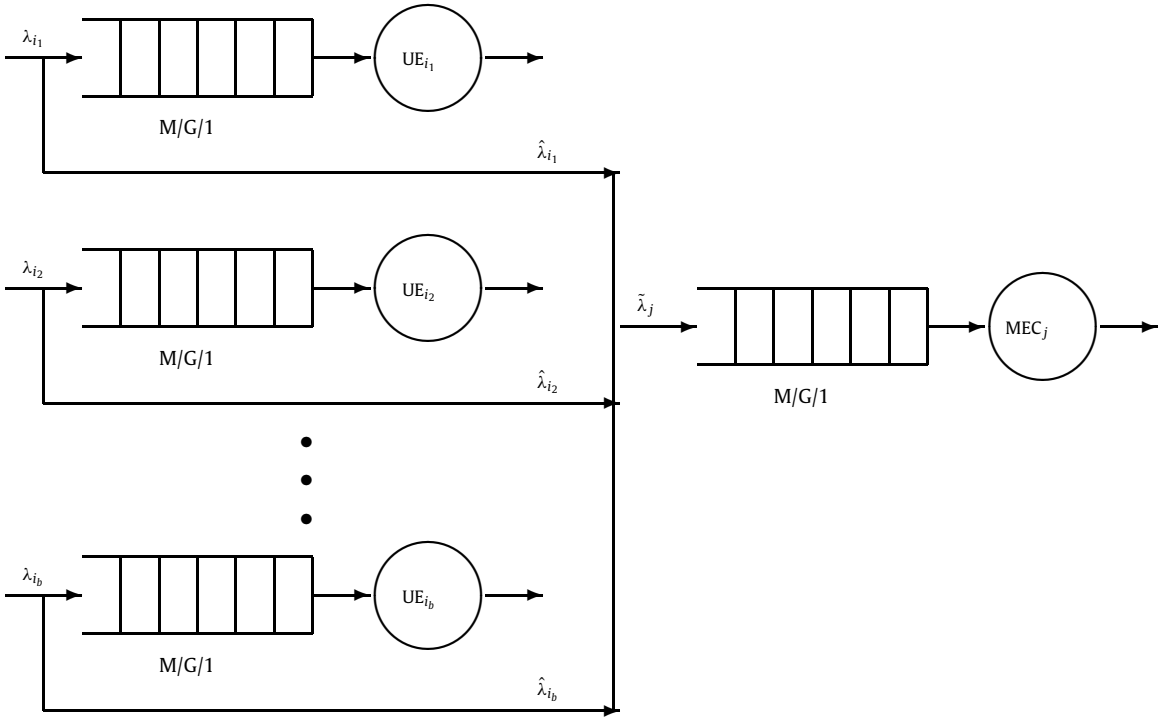

Fig. 1. A queueing model for multiple UEs in the service area of MEC_j with $I_j = \{i_1, i_2, ..., i_b\}$. These UEs offload tasks to MEC_j.

with mean

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

and second moment

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

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

The computation speed of MEC_j is $\tilde{s}_j$ (measured by BI/second). The communication speed between UE_i and MEC_j is $c_{i,j}$ (measured by MB/second). If a task generated on UE_i is executed 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 $d_i/c_{i,j}$, with mean

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

and second moment

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

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

2.2. Resource sharing

Let $I_j = \{i_1, i_2, ..., i_b\}$ denote the set of indices of UEs at MEC_j. All UEs at MEC_j share and compete for the computing power and service capacity of MEC_j by offloading computation tasks to MEC_j. These UEs do not have common interest

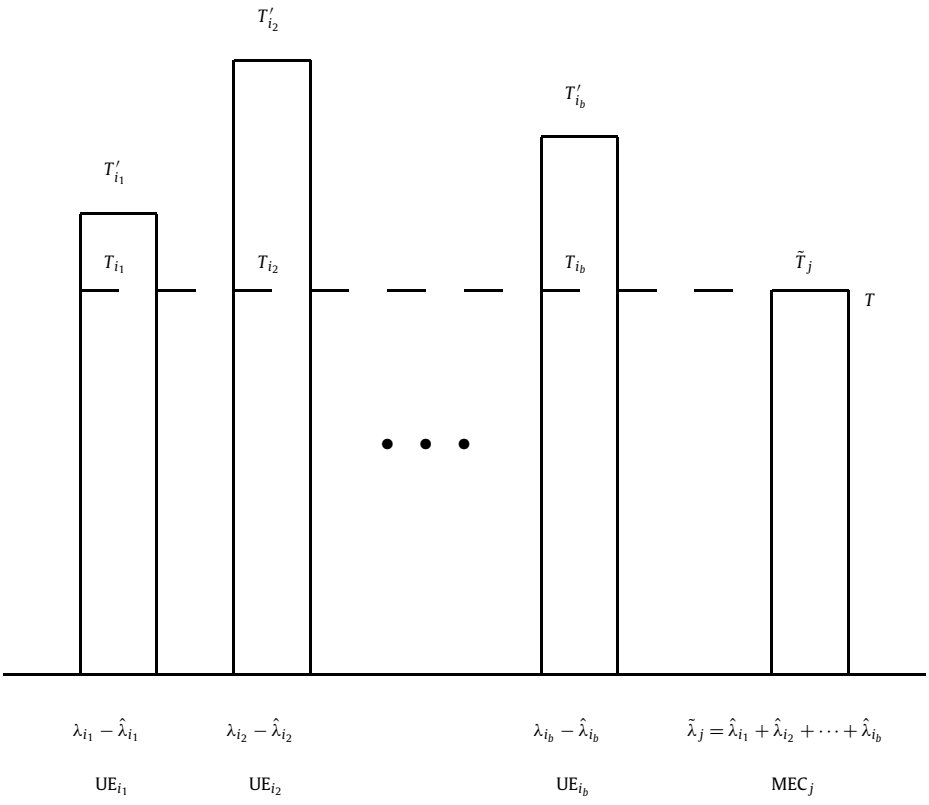

Fig. 2. A computation offloading strategy which makes all UEs served by MEC_j and MEC_j itself having the same average task response time.

in optimizing their combined performance, but each UE is interested in optimizing its own performance. While a non-cooperative game can be played by the UEs [14], a Nash equilibrium is analytically hard to characterize and capture.

We use $\hat{\lambda}_i$ to represent the arrival rate of the substream of tasks of UE_i processed on MEC_j. Let T_i be the average task response time of UE_i, and $\tilde{T}_j$ be the average task response time of MEC_j. For an MEC_j, our *computation offloading strategy* is to find $\hat{\lambda}_i$, $i \in I_j$, and to get T, such that all UEs served by MEC_j and MEC_j itself have the same average task response time, i.e., $T_i = \tilde{T}_j = T$, for all $i \in I_j$. Such a strategy treats all UEs equally and satisfies all of them.

The above computation offloading strategy is illustrated in Fig. 2. Let T'_{i_k} be the average task response time of UE_{i_k} without any offloading. In the beginning, $T_{i_k} = T'_{i_k}$. By offloading $\hat{\lambda}_{i_k}$ amount of workload to MEC_j (see Fig. 1), T_{i_k} is reduced, while $\tilde{T}_j$ is increased. As illustrated by the dashed line in Fig. 2, it is clear that there exist $\hat{\lambda}_{i_1}, \hat{\lambda}_{i_2}, ..., \hat{\lambda}_{i_b}$, such that $T_{i_1} = T_{i_2} = \cdots = T_{i_b} = \tilde{T}_j = T$.

2.3. Random walks

In this section, we discuss mobility modeling for randomly walking UEs.

The main concern is the geographical relationship between a UE and the MECs. In mobile edge computing, the location of a UE is essentially the MEC to which the UE can offload tasks. It is assumed that each MEC has certain service area (i.e., the big circles in Fig. 3). The service areas of different MECs do not overlap. At any moment, a UE is in the service area of only one MEC (i.e., the UE is at the MEC) and the UE can offload its tasks to the MEC.

When a UE walks, it essentially moves from the service area of one MEC to the service area of another MEC, i.e., changes its location from one MEC to another MEC. Moving in the same service area is considered as still. Movements of different UEs are independent of each other. The trajectory of a UE is entirely random and not available in advance.

We would like to mention that when a UE moves from one MEC to another MEC, the service areas of both MECs are affected, since all UEs in the service areas of both MECs need to decide their new computation offloading strategies. It may take a little amount of time for both MECs to re-stabilize. To make our analysis in Sections 3.1–3.2 manageable, we will assume that such time overhead is small and its impact is negligible.

Markov chains are suitable, convenient, and effective for modeling random walks. We will consider both *discrete-time Markov chains* (DTMC) and *continuous-time Markov chains* (CTMC), and their extension, i.e., *semi-Markov processes* (SMP), which make it possible to calculate the joint probability distribution of the locations of UEs at any time. The knowledge of

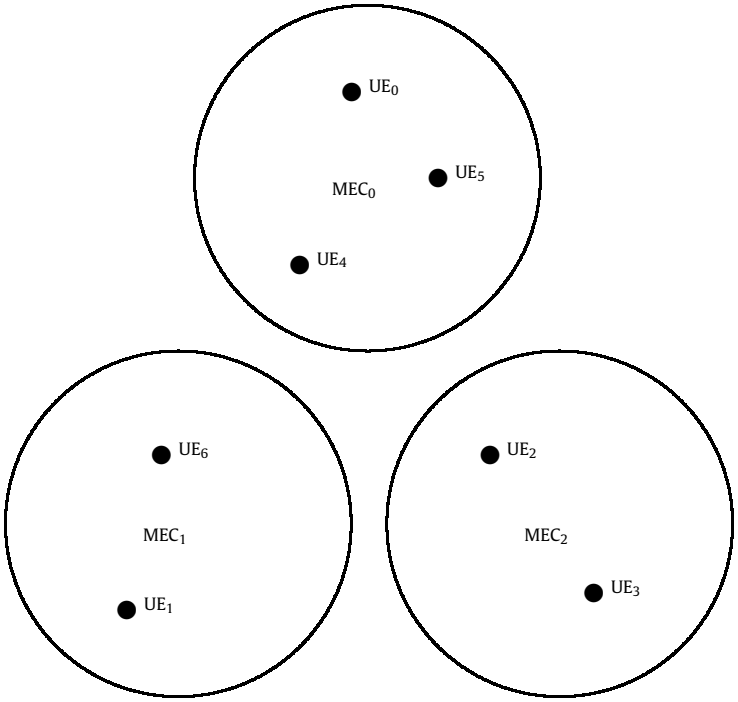

Fig. 3. Each MEC has certain service area (i.e., the big circle). Mobile UEs randomly walk among the circles. $I_0 = \{0, 4, 5\}$. $I_1 = \{1, 6\}$. $I_2 = \{2, 3\}$.

random distribution of UEs among MECs is a key component in and the most challenging part of mobility modeling of UEs, and decisive to performance analysis of mobile UEs.

2.3.1. Discrete-time Markov chains

In our DTMC, time is divided into slots of equal length (e.g., 10 minutes) numbered as $t = 1, 2, 3, \ldots$. At the beginning of each time slot, a UE may move from one MEC to another MEC. The movement of UE_i is governed 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 MEC_j to $\text{MEC}_{j'}$ in a time slot. This means that when UE_i is at MEC_j, it stays there (but still can move in the service area of MEC_j) for a random number of time slots (which is geometrically distributed with parameter $1 - p_i(j, j)$ and mean $1/(1 - p_i(j, j))$) and then transitions to another MEC.

A Markov chain defined by $\boldsymbol{P}_i$ can be represented by a directed graph $G_i = (V, E_i)$, with vertex $v_j \in V$ representing MEC_j, and edge $(v_j, v_{j'}) \in E_i$ with weight $p_i(j, j') \neq 0$ representing the transition probability from MEC_j to $\text{MEC}_{j'}$. We assume that G_i is strongly connected, i.e., there is a directed path from every vertex to every other vertex, for all $0 \le i \le m-1$.

2.3.2. Continuous-time Markov chains

In our CTMC, a UE can change its location at any time. The movement of UE_i is governed 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 MEC_j to $\text{MEC}_{j'}$. Additionally, we have

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

for all $0 \le j \le n-1$. This means that when UE_i is at MEC_j, it stays there for $y_{j'}$ amount of time and then transitions to $\text{MEC}_{j'}$, if $y_{j'} = \min_{j'' \neq j}\{y_{j''}\}$, where $y_{j''}$ is an exponential random variable with parameter $q_i(j, j'')$. Equivalently, this also means that when UE_i is at MEC_j, it stays there for a random amount of time (called the *holding time*), which is exponentially distributed with parameter

$$\sum_{j' \neq j} q_i(j, j') \quad \text{or} \quad -q_i(j, j),$$

and then transitions to $\text{MEC}_{j'}$ with probability

$$p'_i(j, j') = \frac{q_i(j, j')}{\sum_{j'' \neq j} q_i(j, j'')}.$$

The transition probability matrix

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

defines an *embedded DTMC* of the CTMC defined by $\boldsymbol{Q}_i$. $\boldsymbol{P}'_i$ does not have any self-loop.

Similar to DTMC, we can also construct a directed graph $G_i = (V, E_i)$ for the Markov chain defined by $\boldsymbol{Q}_i$, which is actually identical to the underlying directed graph of $\boldsymbol{P}'_i$. Furthermore, we assume that the directed graph is strongly connected.

2.3.3. Semi-Markov processes

A DTMC or a CTMC can be extended to an SMP, where the sojourn time $\tau_{i,j}$ of UE_i at MEC_j can have an arbitrary (not necessarily exponential) probability distribution. At the moment when UE_i changes its location, the motion is controlled by an embedded DTMC with transition probability matrix $\boldsymbol{P}'_i$. The movement of UE_i is specified by $\tau_{i,0}$, $\tau_{i,1}$, ..., $\tau_{i,n-1}$, and $\boldsymbol{P}'_i$.

If $\tau_{i,j}$ is fixed at some constant τ for all $0 \leq i \leq m-1$ and $0 \leq j \leq n-1$, an SMP becomes a DTMC. If $\tau_{i,j}$ is an exponential random variable, an SMP becomes a CTMC.

3. Analytical methods

In this section, we develop our analytical methods.

3.1. Average response time

In this section, we give the average response time of each UE and MEC.

The arrival rate of the substream of tasks of UE_i processed locally at UE_i is $\lambda_i - \hat{\lambda}_i$. The average task response time T_i of UE_i is

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

where W_i is the average task 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 $0 \leq i \leq m-1$ ([11], p. 190).

Let $I_j = \{i \mid \text{UE}_i \text{ is at } \text{MEC}_j\}$ be the set of indices of UEs at MEC_j. All these UEs offload their tasks to MEC_j. The task arrival rate of MEC_j is

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

The average task response time $\tilde{T}_j$ of MEC_j is

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

where $\tilde{W}_j$ is the average task waiting time of MEC_j:

$$\tilde{W}_j = \frac{\tilde{\lambda}_j \overline{\tilde{x}_j^2}}{2(1 - \tilde{\rho}_j)},$$

and

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

and

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

and $\tilde{\rho}_j$ is the utilization of MEC_j:

$$\tilde{\rho}_j = \tilde{\lambda}_j \overline{\tilde{x}_j},$$

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

3.2. Computation offloading

In this section, we consider our computation offloading strategy.

The main challenge is how to find $\hat{\lambda}_i$ and T, such that $\tilde{\rho}_j < 1$, and $T_i = \tilde{T}_j = T$, for all $i \in I_j$. In the following, we develop an efficient method for this purpose.

The following theorem characterizes our computation offloading strategy.

Theorem 1. *Our computation offloading strategy for MEC_j is*

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

for all $i \in I_j$, where T satisfies

$$f(T) = T - \frac{1}{\sum_{i \in I_j} \left(\lambda_i - \frac{2(T - \overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T - \overline{x_i})} \right)} \sum_{i \in I_j} \left(\lambda_i - \frac{2(T - \overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T - \overline{x_i})} \right) \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}} \right) \tag{2}$$

$$+ \frac{\sum_{i \in I_j} \left(\lambda_i - \frac{2(T - \overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T - \overline{x_i})} \right) \left(\overline{\frac{r_i^2}{\tilde{s}_j^2}} + 2 \frac{\overline{r_i} \overline{d_i}}{\tilde{s}_j c_{i,j}} + \frac{\overline{d_i^2}}{c_{i,j}^2} \right)}{2 \left(1 - \sum_{i \in I_j} \left(\lambda_i - \frac{2(T - \overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T - \overline{x_i})} \right) \left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}} \right) \right)} = 0.$$

Proof. The proof is given in Appendix A. □

Notice that

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

that is, $\hat{\lambda}_i$ is a decreasing function of T. As T increases, $\hat{\lambda}_i$ decreases, so is $\tilde{T}_j$. Therefore, $f(T) = T - \tilde{T}_j$ is an increasing function of T, and the equation $f(T) = 0$ can be solved by using the bisection search algorithm.

Since $0 \le \hat{\lambda}_i \le \lambda_i$, we obtain

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

and

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

for all $i \in I_j$. This means that T can be searched in the following interval $[T_{lb}, T_{ub}]$ with

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

Let $\Delta = T_{ub} - T_{lb}$ be the length of the above search interval. Assume that the search terminates when the length of the search interval is no more than ε (i.e., the accuracy requirement). Then, the standard bisection search algorithm can be completed in $O(\log(\Delta/\varepsilon))$ time.

We would like to mention that the $\hat{\lambda}_i$'s are obtained simultaneously for all $i \in I_j$ using Eq. (1), and these $\hat{\lambda}_i$'s are dependent of each other through T, which is obtained by solving Eq. (2).

3.3. Mobility analysis

In this section, we analyze randomized location distribution of UEs, which is the result of random walks of UEs. For both DTMC and CTMC, and a special type of SMP, we are able to calculate the joint probability distribution of the locations of UEs at any time, which is crucial for performance analysis and prediction of mobile UEs.

3.3.1. Discrete-time Markov chains

Let the set of possible locations of UE_i be

$$N_i = \{j_i \mid 0 \le j_i \le n-1\},$$

for all $0 \le i \le m-1$. Then, the Cartesian product

$$\mathcal{N} = N_0 \times N_1 \times \cdots \times N_{m-1}$$

gives the set of *location distributions* of the UEs, where

$$J = (j_0, j_1, ..., j_{m-1}) \in \mathcal{N}$$

represents a location distribution of the UEs, such that UE_i is at MEC_{j_i}. For convenience, $J = (j_0, j_1, ..., j_{m-1})$ is treated as an m-digit radix-n integer $(j_0 j_1 \cdots j_{m-1})_n$ in the range $0, 1, ..., N-1$:

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

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

To include interaction among UEs into consideration, we need to consider the *joint random walk* of the m UEs, which is a *joint DTMC* with the *joint transition probability matrix*:

$$\boldsymbol{P} = [p(J, J')],$$

where $p(J, J')$ is the transition probability from J to J'. Let us assume that $J = (j_0, j_1, ..., j_{m-1})$ and $J' = (j'_0, j'_1, ..., j'_{m-1})$. Notice that J is changed to J' if and only if UE_i changes from MEC_{j_i} to $\text{MEC}_{j'_i}$ (which happens with probability $p_i(j_i, j'_i)$), for all $0 \le i \le m-1$, simultaneously:

$$\begin{array}{lccccc} J = & (j_0, & j_1, & ..., & j_{m-1}) \\ & \downarrow \text{UE}_0 & \downarrow \text{UE}_1 & & \downarrow \text{UE}_{m-1} \\ J' = & (j'_0, & j'_1, & ..., & j'_{m-1}) \end{array}$$

Since the UEs move independently, we have

$$p(J, J') = \prod_{i=0}^{m-1} p_i(j_i, j'_i),$$

for all $0 \le j_0, j_1, ..., j_{m-1}, j'_0, j'_1, ..., j'_{m-1} \le n-1$. Note that each J can transition to n^m different J'.

Let $\pi^{(t)}(J)$ be the probability of J at time t, and

$$\pi^{(t)} = [\pi^{(t)}(0), \pi^{(t)}(1), ..., \pi^{(t)}(N-1)]$$

be the *joint probability vector* (i.e., the joint probability distribution of the locations of the UEs) at time t. (Note: If the location distribution of the UEs is treated as a random variable with $\mathcal{N}$ as the support, then $\pi^{(t)}$ is actually the *probability mass function* (pmf) of this random variable at time t.) Then, $\pi^{(t)}$ can be calculated by using the following equation:

$$\pi^{(t)} = \pi^{(0)} \boldsymbol{P}^t,$$

for all $t = 1, 2, 3, ...$, where $\pi^{(0)}$ gives the initial location distribution of the UEs at time 0 ([11], p. 32). Unfortunately, due to the large size $(n^m \times n^m)$ of matrix $\boldsymbol{P}$, even one matrix multiplication takes $O(n^{3m})$ time, which is excessively long even for moderate values of m and n.

The following theorem suggests a more efficient way to calculate

$$\boldsymbol{P}^t = [p^{(t)}(J, J')]$$

based on

$$\boldsymbol{P}_i^t = [p_i^{(t)}(j, j')],$$

where $0 \le i \le m-1$.

(Note: Recall that the Kronecker product of two $n \times n$ matrices $\boldsymbol{A} = [a_{ij}]$ and $\boldsymbol{B}$ is an $n^2 \times n^2$ matrix

$$\boldsymbol{A} \otimes \boldsymbol{B} = [a_{ij}\boldsymbol{B}],$$

that is, a matrix of matrices.)

Theorem 2. *For all $J = (j_0, j_1, ..., j_{m-1})$ and $J' = (j'_0, j'_1, ..., j'_{m-1})$ and $t \ge 1$, we have*

$$p^{(t)}(J, J') = \prod_{i=0}^{m-1} p_i^{(t)}(j_i, j'_i).$$

Equivalently, $\boldsymbol{P}^t$ is the Kronecker product of the $\boldsymbol{P}_i^t$'s:

$$\boldsymbol{P}^t = \boldsymbol{P}_1^t \otimes \boldsymbol{P}_2^t \otimes \cdots \otimes \boldsymbol{P}_m^t.$$

Proof. The joint random walk changes from J to J' in t steps if and only if UE_i changes from MEC_{j_i} to $\mathrm{MEC}_{j'_i}$ in t steps, for all $0 \le i \le m-1$, simultaneously. Hence, we get

$$p^{(t)}(J, J') = \prod_{i=0}^{m-1} p_i^{(t)}(j_i, j'_i),$$

due to the independence of the UEs.

For Kronecker product, we can verify that if

$$\boldsymbol{B} = [b(J, J')] = \boldsymbol{A}_1 \otimes \boldsymbol{A}_2 \otimes \cdots \otimes \boldsymbol{A}_m,$$

where

$$\boldsymbol{A}_i = [a_i(j, j')],$$

and $0 \le i \le m-1$, then, we have

$$B(J, J') = \prod_{i=0}^{m-1} a_i(j_i, j'_i),$$

for all $J = (j_0, j_1, ..., j_{m-1})$ and $J' = (j'_0, j'_1, ..., j'_{m-1})$. □

To reduce the computation time, we may first calculate

$$\boldsymbol{P}_i^t = [p_i^{(t)}(j, j')],$$

for all $0 \le i \le m-1$, each can be computed in $O(n^3 \log t)$ time with $O(\log t)$ matrix multiplications. Then, we calculate

$$\boldsymbol{P}^t = [p^{(t)}(J, J')],$$

where

$$p^{(t)}(J, J') = \prod_{i=0}^{m-1} p_i^{(t)}(j_i, j'_i),$$

for $J = (j_0, j_1, ..., j_{m-1})$ and $J' = (j'_0, j'_1, ..., j'_{m-1})$. Actually, $\boldsymbol{P}^t$ is the Kronecker product of the $\boldsymbol{P}_i^t$'s:

$$\boldsymbol{P}^t = \boldsymbol{P}_1^t \otimes \boldsymbol{P}_2^t \otimes \cdots \otimes \boldsymbol{P}_m^t.$$

Hence, based on $\boldsymbol{P}_i^t$, where $0 \le i \le m-1$, $\boldsymbol{P}^t$ can be computed in $O(mn^{2m})$ time (much shorter than $O(n^{3m})$ time before), where we notice that $\boldsymbol{P}^t$ has n^{2m} components, each can be computed in $O(m)$ time.

Let $\pi_i^{(t)}(j)$ be the probability that UE_i is with MEC_j at time t, and

$$\pi_i^{(t)} = [\pi_i^{(t)}(0), \pi_i^{(t)}(1), ..., \pi_i^{(t)}(n-1)]$$

be the probability vector of UE_i (i.e., the probability mass function of the random location of UE_i with support N_i) at time t.

The following theorem suggests an even more efficient way to calculate the joint probability vector.

Theorem 3. *For all $t = 1, 2, 3, ..., \pi_i^{(t)}$ can be calculated by using the following equation:*

$$\pi_i^{(t)} = \pi_i^{(0)} \boldsymbol{P}_i^t, \tag{3}$$

where $\pi_i^{(0)}$ is the initial location distribution of UE_i at time 0. If $J = (j_0, j_1, ..., j_{m-1})$, we have

$$\pi^{(t)}(J) = \prod_{i=0}^{m-1} \pi_i^{(t)}(j_i). \tag{4}$$

Proof. The computation of $\pi_i^{(t)}$ is standard (see, e.g., [11], p. 32). The computation of $\pi^{(t)}(J)$ is based on the fact that the joint random walk has location distribution J at time t if and only if UE_i is with MEC_{j_i} at time t, for all $0 \le i \le m-1$. □

Based on $\pi_i^{(t)}$, where $0 \le i \le m-1$, $\pi^{(t)}$ can be computed in $O(mn^m)$ time (much shorter than $O(n^{2m})$ time before), where we notice that $\pi^{(t)}$ has n^m components, each can be computed in $O(m)$ time. Further time reduction seems difficult, since $\pi^{(t)}$ has n^m components to compute.

Furthermore, let

$$\pi_i = \lim_{t\to\infty} \pi_i^{(t)} = [\pi_i(0), \pi_i(1), ..., \pi_i(n-1)]$$

be the stationary probability vector of UE_i and

$$\pi = \lim_{t\to\infty} \pi^{(t)} = [\pi(0), \pi(1), ..., \pi(N-1)]$$

be the *joint stationary probability vector*. According to the Fundamental Theorem of Markov Chains ([2], p. 66), if G_i is strongly connected, there is a unique stationary probability vector π_i.

Theorem 4. *π_i can be obtained by solving the linear system of equations:*

$$\pi_i = \pi_i \boldsymbol{P}_i, \tag{5}$$

with the condition

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

for all $0 \le i \le m-1$. Furthermore, for $J = (j_0, j_1, ..., j_{m-1})$, we have

$$\pi(J) = \prod_{i=0}^{m-1} \pi_i(j_i). \tag{6}$$

Proof. The computation of π_i is well known ([11], p. 31). The computation of $\pi(J)$ is based on the fact that the joint random walk has location distribution J if and only if UE_i is with MEC_{j_i}, for all $0 \le i \le m-1$. □

3.3.2. Continuous-time Markov chains

For m UEs, we need to consider their *joint CTMC* with the *joint transition rate matrix*

$$\boldsymbol{Q} = [q(J, J')],$$

where $q(J, J')$ is the transition rate from J to J'. It is observed that $\boldsymbol{Q}$ is not a straightforward combination of the $\boldsymbol{Q}_i$'s.

Let $J = (j_0, ..., j_i, ..., j_{m-1})$ and $J' = (j_0, ..., j'_i, ..., j_{m-1})$. Since at any instant time, only one UE can move,

$$\begin{aligned} J &= (j_0, \ ..., \ j_i, \quad ..., \ j_{m-1}) \\ &\qquad\quad \Big\downarrow \mathrm{UE}_i \\ J' &= (j_0, \ ..., \ j'_i, \quad ..., \ j_{m-1}) \end{aligned}$$

we have

$$q(J, J') = q_i(j_i, j'_i),$$

for all $0 \le i \le m-1$ and $0 \le j_0, ..., j_i \ne j'_i, ..., j_{m-1} \le n-1$. It is clear that each J can transition to $m(n-1)$ different J'. Furthermore, we have

$$q(J, J) = -\sum_{J' \ne J} q(J, J') = \sum_{i=0}^{m-1} q_i(j_i, j_i) = -\sum_{i=0}^{m-1} \sum_{j'_i \ne j_i} q_i(j_i, j'_i),$$

for all $0 \le J = (j_0, ..., j_i, ..., j_{m-1}) \le N-1$.

Again, let $\pi^{(t)}(J)$ be the probability of J at time t, and

$$\pi^{(t)} = [\pi^{(t)}(0), \pi^{(t)}(1), ..., \pi^{(t)}(N-1)]$$

be the joint probability vector at time t. Then, with the initial location distribution $\pi^{(0)}$ of the UEs, $\pi^{(t)}$ can be calculated by using the following equation:

$$\pi^{(t)} = \pi^{(0)} \exp(\boldsymbol{Q}t),$$

where

$$\exp(\boldsymbol{Q}t) = \boldsymbol{I} + \boldsymbol{Q}t + \frac{(\boldsymbol{Q}t)^2}{2!} + \frac{(\boldsymbol{Q}t)^3}{3!} + \cdots,$$

for all $t > 0$ ([11], p. 51). (Note: $\boldsymbol{I}$ is the identity matrix.) Unfortunately, as we have already known, the above computation is inefficient due to the large size ($n^m \times n^m$) of matrix $\boldsymbol{Q}$.

Let $\pi_i^{(t)}(j)$ be the probability that UE_i is with MEC_j at time t, and

$$\pi_i^{(t)} = [\pi_i^{(t)}(0), \pi_i^{(t)}(1), ..., \pi_i^{(t)}(n-1)]$$

be the probability vector of UE_i at time t.

Theorem 5. *For all $t > 0$, $\pi_i^{(t)}$ can be calculated by using the following equation:*

$$\pi_i^{(t)} = \pi_i^{(0)} \exp(\boldsymbol{Q}_i t), \tag{7}$$

where

$$\exp(\boldsymbol{Q}_i t) = \boldsymbol{I} + \boldsymbol{Q}_i t + \frac{(\boldsymbol{Q}_i t)^2}{2!} + \frac{(\boldsymbol{Q}_i t)^3}{3!} + \cdots, \tag{8}$$

for all $0 \le i \le m-1$. Based on $\pi_i^{(t)}$, we can get

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

for $J = (j_0, j_1, ..., j_{m-1})$.

Proof. The computation of $\pi_i^{(t)}$ is standard ([11], p. 51). The computation of $\pi^{(t)}(J)$ is identical to Eq. (4) in Theorem 3. □

To reduce both memory space and computation time, we first calculate $\pi_i^{(t)}$, for all $0 \le i \le m-1$. Based on $\pi_i^{(t)}$, we can get $\pi^{(t)}(J)$, for all $0 \le J \le N-1$. Due to the small size ($n \times n$) of $\boldsymbol{Q}_i$, the above computation is much faster.

Recall that CTMC $\boldsymbol{Q}_i$ and its embedded DTMC $\boldsymbol{P}'_i$ have the same underlying directed graph G_i. If G_i is strongly connected, there is a unique stationary probability vector π'_i for DTMC $\boldsymbol{P}'_i$, and consequently, there is a unique stationary probability vector π_i for CTMC $\boldsymbol{Q}_i$.

Theorem 6. *The stationary probability vector of UE_i, i.e.,*

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

can be obtained by solving the linear system of equations:

$$\pi_i \boldsymbol{Q}_i = 0, \tag{9}$$

with the condition

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

for all $0 \le i \le m-1$. Furthermore, for $J = (j_0, j_1, ..., j_{m-1})$, we have

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

Proof. The computation of π_i is well known ([11], p. 52). The computation of $\pi(J)$ is identical to Eq. (6) in Theorem 4. □

It can also be verified that if

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

is the unique stationary probability vector of $\boldsymbol{P}_i'$, then we have

$$\pi_i(j) = \frac{\pi_i'(j)\tau_i(j)}{\sum_{0 \le j'' \le n-1} \pi_i'(j'')\tau_i(j'')},$$

where

$$\tau_i(j) = -\frac{1}{q_i(j,j)},$$

is the average holding time when UE_i is at MEC_j, for all $0 \le i \le m-1$ and $0 \le j \le n-1$ (see [26], Section 11.3.2, Theorem 11.3).

In fact, the above argument is also suitable to semi-Markov processes, where the sojourn time at an MEC can be an arbitrary random variable (see [30], Section 7.6, Eq. (7.24)). This makes our methodology acceptable and applicable to much wider scenarios and situations.

3.3.3. Semi-Markov processes

Consider UE_i which is with MEC_{j_0} at time 0. UE_i moves to MEC_{j_1}, MEC_{j_2}, MEC_{j_3}, ..., in the following way: UE_i stays at $MEC_{j_{k-1}}$ for $\tau_{i,j_{k-1}}$ amount of time and then moves to MEC_{j_k}, for $k = 1, 2, 3, ...$, where τ_{i,j_0}, τ_{i,j_1}, τ_{i,j_2}, ... are independent and arbitrary random variables.

At time t, if we know that UE_i has made k location changes, then we can easily calculate $\pi_i^{(t)}$ using Eq. (3), which is

$$\pi_i^{(t)} = \pi_i^{(0)} (\boldsymbol{P}_i')^k,$$

where $\boldsymbol{P}_i'$ is the transition probability matrix of the embedded DTMC. However, we do not know the value of k, since it is a random variable.

We are interested in the probability mass function of k. To this end, we assume that τ_{i,j_0}, τ_{i,j_1}, τ_{i,j_2}, ... are independent and identically distributed (i.i.d.) random variables. That is, the holding times of UE_i at all MEC_j's are i.i.d. random variables τ_i.

A sequence of i.i.d. random variables $\tau_{i,1}$, $\tau_{i,2}$, $\tau_{i,3}$, ... form a *renewal process*. We use $X_i(t)$ to represent the number of renewals by time t. Let

$$S_{i,k} = \tau_{i,1} + \tau_{i,2} + \cdots + \tau_{i,k},$$

where $k = 0, 1, 2, ...$ If $S_{i,k} \le t < S_{i,k+1}$, we have $X_i(t) = k$.

Theorem 7. *For all $t > 0$, $\pi_i^{(t)}$ can be calculated by using the following equation:*

$$\pi_i^{(t)} = \pi_i^{(0)} \left(\sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k] (\boldsymbol{P}_i')^k \right) = \pi_i^{(0)} \left(\sum_{k=0}^{\infty} (F_{S_{i,k}}(t) - F_{S_{i,k+1}}(t)) (\boldsymbol{P}_i')^k \right). \tag{10}$$

Based on $\pi_i^{(t)}$, *we can get*

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

for $J = (j_0, j_1, ..., j_{m-1})$.

(Notation: Let $f_Y(y)$ and $F_Y(y)$ denote the *probability density function* (pdf) and the *cumulative distribution function* (cdf) of a random variable Y respectively. $\mathbf{P}[\cdot]$ is the probability of an event.)

Proof. Since $X_i(t) < k$ if and only if $S_{i,k} > t$, we have

$$\mathbf{P}[X_i(t) < k] = \mathbf{P}[S_{i,k} > t] = 1 - F_{S_{i,k}}(t).$$

Thus, we get

$$\mathbf{P}[X_i(t) = k] = \mathbf{P}[X_i(t) < k+1] - \mathbf{P}[X_i(t) < k] = F_{S_{i,k}}(t) - F_{S_{i,k+1}}(t),$$

for all $k = 0, 1, 2, ...$, with $F_{S_{i,0}}(t) = 1$.

Based on the information of $\mathbf{P}[X_i(t) = k]$, we can calculate

$$\pi_i^{(t)} = \pi_i^{(0)} \left(\sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k] (\boldsymbol{P}_i')^k \right),$$

which is

$$\pi_i^{(t)} = \pi_i^{(0)} \left(\sum_{k=0}^{\infty} (F_{S_{i,k}}(t) - F_{S_{i,k+1}}(t)) (\boldsymbol{P}_i')^k \right).$$

Based on $\pi_i^{(t)}$, we can get $\pi^{(t)}(J)$ using Eq. (4), for all $0 \le J \le N-1$. □

As an interesting special case, let us consider the case when each τ_i is an exponential random variable with parameter q_i. This means that UE_i walks according to a CTMC with

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

where

$$q_i(j, j) = -q_i,$$

for all $0 \le j \le n-1$. It is clear that the corresponding embedded DTMC has

$$\boldsymbol{P}_i' = \frac{\boldsymbol{Q}_i}{q_i} + \boldsymbol{I}.$$

Then, $S_{i,k}$, a summation of k i.i.d. exponential random variables, has an Erlang distribution with

$$f_{S_{i,k}}(t) = \frac{q_i^k t^{k-1} e^{-q_i t}}{(k-1)!},$$

and

$$F_{S_{i,k}}(t) = 1 - e^{-q_i t} \sum_{j=0}^{k-1} \frac{(q_i t)^j}{j!}.$$

Consequently, we have

$$F_{S_{i,k}}(t) - F_{S_{i,k+1}}(t) = e^{-q_i t} \frac{(q_i t)^k}{k!},$$

which yields

$$\begin{aligned}\sum_{k=0}^{\infty}(F_{S_{i,k}}(t)-F_{S_{i,k+1}}(t))(\boldsymbol{P}_i')^k &= e^{-q_i t}\sum_{k=0}^{\infty}\frac{(q_i t)^k}{k!}(\boldsymbol{P}_i')^k\\ &= e^{-q_i t}\sum_{k=0}^{\infty}\frac{(q_i t)^k}{k!}\left(\frac{\boldsymbol{Q}_i}{q_i}+\boldsymbol{I}\right)^k\\ &= e^{-q_i t}\sum_{k=0}^{\infty}\frac{t^k}{k!}(\boldsymbol{Q}_i+q_i\boldsymbol{I})^k\\ &= e^{-q_i t}\exp((\boldsymbol{Q}_i+q_i\boldsymbol{I})t)\\ &= e^{-q_i t}\exp(\boldsymbol{Q}_i t)\exp(\boldsymbol{I}q_i t)\\ &= e^{-q_i t}\exp(\boldsymbol{Q}_i t)\boldsymbol{I}e^{q_i t}\\ &= \exp(\boldsymbol{Q}_i t),\end{aligned}$$

which is identical to Eq. (8) as expected.

Theorem 8. *The stationary probability vector π_i of UE$_i$ is the same as the stationary probability vector π_i' of its embedded DTMC $\boldsymbol{P}_i'$.*

Proof. It is easy to see that UE$_i$'s SMP and its embedded DTMC $\boldsymbol{P}_i'$ have the same underlying directed graph G_i. If G_i is strongly connected, there is a unique stationary probability vector π_i' for DTMC $\boldsymbol{P}_i'$, and consequently, there is a unique stationary probability vector π_i for the SMP. Furthermore, since UE$_i$ has the same expected holding time at all MECs, its stationary probability vector is the same as that of $\boldsymbol{P}_i'$. □

4. Performance predictions

In this section, we predict the performance of randomly walking mobile UEs.

4.1. UE-centric performance measures

In this section, we define performance measures for mobile UEs. (Later in Section 8.1, we will define performance measures for MECs.)

Let $J=(j_0,\ldots,j_i,\ldots,j_{m-1})$ be a UE distribution, in which, UE$_i$ is at MEC$_{j_i}$. Then, we have $I_{j_i}=\{i' \mid j_{i'}=j_i\}$, which includes all UEs at MEC$_{j_i}$ The average response time of tasks offloaded from UE$_i$ to MEC$_{j_i}$ is

$$\tilde{T}_{i,j_i}=\overline{\tilde{x}_{i,j_i}}+\tilde{W}_{j_i}.$$

The average response time $T_i(J)$ of tasks generated on UE$_i$ under J is

$$T_i(J)=\left(\frac{\lambda_i-\hat{\lambda}_i}{\lambda_i}\right)T_i+\frac{\hat{\lambda}_i}{\lambda_i}\tilde{T}_{i,j_i}, \tag{11}$$

which is actually

$$T_i(J)=\left(\frac{\lambda_i-\hat{\lambda}_i}{\lambda_i}\right)T_i+\frac{\hat{\lambda}_i}{\lambda_i}\left(\frac{\overline{r_i}}{\tilde{s}_{j_i}}+\frac{\overline{d_i}}{c_{i,j_i}}+\tilde{W}_{j_i}\right),$$

where

$$\tilde{W}_{j_i}=\frac{\tilde{\lambda}_{j_i}\overline{\tilde{x}_{j_i}^2}}{2(1-\tilde{\rho}_{j_i})}.$$

The *instantaneous average response time* $T_i^{(t)}$ of tasks generated on UE$_i$ at time t is

$$T_i^{(t)}=\sum_{J=0}^{N-1}\pi^{(t)}(J)T_i(J), \tag{12}$$

which is averaged over all J. The *overall average response time* T_i^* of tasks generated on UE$_i$ at times $t_1,t_2,\ldots,t_K$ is

$$T_i^*=\frac{1}{K}\sum_{k=1}^{K}T_i^{(t_k)}, \tag{13}$$

which is averaged over all t_k. The *stationary average response time* $T_i^{(\infty)}$ of tasks generated on UE_i is

$$T_i^{(\infty)} = \sum_{J=0}^{N-1} \pi(J) T_i(J). \tag{14}$$

T_i^* and $T_i^{(\infty)}$ are our ultimate performance predictions for UE_i in a fog computing environment with mobile users.

We would like to emphasize that $T_i^{(\infty)}$ is applicable as long as the joint stationary probability vector π is available, which can be obtained not only for DTMC and CTMC, but also for any semi-Markov process, for which we know the residence time of each UE at each MEC.

4.2. The algorithm

In this section, we devise an algorithmic procedure to obtain our performance predictions.

Algorithm 1 depicts our procedure to predict the performance of randomly walking mobile UEs with DTMC, CTMC, and SMP. The algorithmic procedure essentially summarizes all our models and methods.

The time complexity of the algorithm can be analyzed as follows.

Line (3) takes $O(\log(\Delta/\varepsilon))$ time, where Δ is the upper limit of the search interval and ε is the accuracy requirement. Lines (5)–(6) take constant, and the for-loop in lines (4)–(7) can be done in $O(|I_j|)$ time. The for-loop in lines (2)–(8) can be done $O(n\log(\Delta/\varepsilon)+m)$ time, since $\sum |I_j| = m$, and the overall time complexity of the for-loop in lines (1)–(9) is $O(n^m(n\log(\Delta/\varepsilon)+m))$.

Let $t = \max\{t_1, t_2, ..., t_K\}$. For DTMC, line 12 takes $O(tn^3)$ time. For CTMC, line 12 takes $O(wn^3)$ time by computing the first w terms, where w depends on the accuracy requirement. For SMP, line 12 takes $O(vn^3)$ time, where v is a small constant (see Section 5.1). Line 13 takes $O(n^2)$ time. Hence, the for-loop in lines (11)–(14) can be done in $O(tmn^3)$ time for DTMC and $O(wmn^3)$ time for CTMC and $O(vmn^3)$ time for SMP. Line (15) takes $O(mn^m)$ time. Line (17) takes $O(n^m)$ time. The for-loop in lines (16)–(18) can be done in $O(mn^m)$ time. The overall time complexity of the for-loop in lines (10)–(19) is $O(K(tmn^3+mn^m))$ for DTMC and $O(K(wmn^3+mn^m))$ for CTMC and $O(K(vmn^3+mn^m))$ for SMP.

Line (21) takes $O(K)$ time. The for-loop in lines (20)–(22) can be done in $O(Km)$ time.

Line (24) takes $O(n^3)$ time. The for-loop in lines (23)–(25) can be done in $O(mn^3)$ time.

Line (26) takes $O(mn^m)$ time.

Line (28) takes $O(n^m)$ time. The for-loop in lines (27)–(29) can be done in $O(mn^m)$ time.

To summarize, the time complexity of our algorithmic procedure is $O(n^{m+1}\log(\Delta/\varepsilon)+K(tmn^3+mn^m))$ for DTMC, and $O(n^{m+1}\log(\Delta/\varepsilon)+K(wmn^3+mn^m))$ for CTMC, and $O(n^{m+1}\log(\Delta/\varepsilon)+K(vmn^3+mn^m))$ for SMP.

5. Numerical data and examples

In this section, we demonstrate some numerical data and examples.

5.1. Parameter setting

We consider a mobile edge computing environment with $m = 7$ UEs and $n = 5$ MECs. Our model parameters are set as follows.

For queueing models, 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, $s_i = 2.0 + 0.1i$ 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 $0 \le i \le m-1$; $\tilde{s}_j = 3.5 + 0.2j$ 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 computation offloading to the MECs (i.e., $\hat{\lambda}_i = 0$), each UE has average task response time over 41 seconds.

For mobility models, we consider a situation, where the MECs form a simple topology, i.e., a line. We assume that the initial distribution of the UEs is:

$$\begin{array}{ccccccc} \mathrm{MEC}_0 & & \mathrm{MEC}_1 & \mathrm{MEC}_2 & \mathrm{MEC}_3 & & \mathrm{MEC}_4 \\ \mathrm{UE}_0, \mathrm{UE}_1 & \mathrm{UE}_2 & & \mathrm{UE}_3 & & \mathrm{UE}_4 & \mathrm{UE}_5, \mathrm{UE}_6 \end{array}$$

with

$$J = (0, 0, 1, 2, 3, 4, 4),$$

whose integer value is

$$J = 1 \times 5^4 + 2 \times 5^3 + 3 \times 5^2 + 4 \times 5^1 + 4 \times 5^0 = 974.$$

Algorithm 1: Performance predictions for randomly walking mobile UEs with DTMC/CTMC/SMP.

Input: λ_i, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, s_i, for all $0 \le i \le m-1$; $\tilde{s}_j$, for all $0 \le j \le n-1$; $c_{i,j}$, for all $0 \le i \le m-1$ and $0 \le j \le n-1$; $\pi_i^{(0)}$, $\boldsymbol{P}_i$ for DTMC (or $\boldsymbol{Q}_i$ for CTMC, or $\tau_i, \boldsymbol{P}_i'$ for SMP), for all $0 \le i \le m-1$.

Output: T_i^* and $T_i^{(\infty)}$, for all $0 \le i \le m-1$.

```
for (J = 0; J < N; J++) do
    for (j = 0; j < n; j++) do
        find T by solving Eq. (2);
        for (i ∈ I_j) do
            calculate λ̂_i, i ∈ I_j, by using Eq. (1);
            calculate T_i(J) by using Eq. (11);
        end do;
    end do;
end do;
for (k = 0; k < K; k++) do
    for (i = 0; i < m; i++) do
        calculate P_i^{t_k} for DTMC (or exp(Q_i t_k) for CTMC, or F_{S_{i,k}}(t_k), (P_i')^k for SMP);
        calculate π_i^{(t_k)} by using Eq. (3) for DTMC (or Eq. (7) for CTMC, or Eq. (10) for SMP);
    end do;
    calculate π^{(t_k)} by using Eq. (4);
    for (i = 0; i < m; i++) do
        calculate T_i^{(t_k)} by using Eq. (12);
    end do;
end do;
for (i = 0; i < m; i++) do
    calculate T_i^* by using Eq. (13);
end do;
for (i = 0; i < m; i++) do
    find π_i by solving Eq. (5) for DTMC (or Eq. (9) for CTMC, or Eq. (5) for SMP);
end do;
calculate π by using Eq. (6);
for (i = 0; i < m; i++) do
    calculate T_i^{(∞)} by using Eq. (14);
end do;
return T_i^* and T_i^{(∞)}, for all 0 ≤ i ≤ m − 1.
```

If the UEs do not move, their average response times under $J = 974$ are

$$T_0(J) = 1.34017 \text{ seconds},$$
$$T_1(J) = 1.35181 \text{ seconds},$$
$$T_2(J) = 1.01574 \text{ seconds},$$
$$T_3(J) = 1.01292 \text{ seconds},$$
$$T_4(J) = 1.01119 \text{ seconds},$$
$$T_5(J) = 1.37543 \text{ seconds},$$
$$T_6(J) = 1.38595 \text{ seconds},$$

which are much shorter than the ones without computation offloading.

For DTMC, the transition probability matrices are set as:

$$\boldsymbol{P}_0 = \boldsymbol{P}_5 = \begin{bmatrix} 0.6 & 0.4 & 0.0 & 0.0 & 0.0 \\ 0.4 & 0.3 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.4 & 0.3 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.4 & 0.3 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.6 & 0.4 \end{bmatrix},$$

$$\boldsymbol{P}_1 = \boldsymbol{P}_6 = \begin{bmatrix} 0.4 & 0.6 & 0.0 & 0.0 & 0.0 \\ 0.3 & 0.3 & 0.4 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.3 & 0.4 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.3 & 0.4 \\ 0.0 & 0.0 & 0.0 & 0.4 & 0.6 \end{bmatrix},$$

$$\boldsymbol{P}_2 = \boldsymbol{P}_3 = \boldsymbol{P}_4 = \begin{bmatrix} 0.5 & 0.5 & 0.0 & 0.0 & 0.0 \\ 0.3 & 0.4 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.3 & 0.4 & 0.3 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.4 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.5 & 0.5 \end{bmatrix}.$$

The above matrices are set in such a way that UE_0 and UE_5 move more towards left, UE_1 and UE_6 move more towards right, while UE_2, UE_3, UE_4 move evenly in both directions.

For CTMC, transition rate matrices are set as:

$$\boldsymbol{Q}_0 = \boldsymbol{Q}_5 = \begin{bmatrix} -0.0014 & 0.0014 & 0.0000 & 0.0000 & 0.0000 \\ 0.0016 & -0.0030 & 0.0014 & 0.0000 & 0.0000 \\ 0.0000 & 0.0016 & -0.0030 & 0.0014 & 0.0000 \\ 0.0000 & 0.0000 & 0.0016 & -0.0030 & 0.0014 \\ 0.0000 & 0.0000 & 0.0000 & 0.0016 & -0.0016 \end{bmatrix},$$

$$\boldsymbol{Q}_1 = \boldsymbol{Q}_6 = \begin{bmatrix} -0.0016 & 0.0016 & 0.0000 & 0.0000 & 0.0000 \\ 0.0014 & -0.0030 & 0.0016 & 0.0000 & 0.0000 \\ 0.0000 & 0.0014 & -0.0030 & 0.0016 & 0.0000 \\ 0.0000 & 0.0000 & 0.0014 & -0.0030 & 0.0016 \\ 0.0000 & 0.0000 & 0.0000 & 0.0014 & -0.0014 \end{bmatrix},$$

$$\boldsymbol{Q}_2 = \boldsymbol{Q}_3 = \boldsymbol{Q}_4 = \begin{bmatrix} -0.0015 & 0.0015 & 0.0000 & 0.0000 & 0.0000 \\ 0.0015 & -0.0030 & 0.0015 & 0.0000 & 0.0000 \\ 0.0000 & 0.0015 & -0.0030 & 0.0015 & 0.0000 \\ 0.0000 & 0.0000 & 0.0015 & -0.0030 & 0.0015 \\ 0.0000 & 0.0000 & 0.0000 & 0.0015 & -0.0015 \end{bmatrix}.$$

Again, the above matrices are set in such a way that UE_0 and UE_5 move more towards left, UE_1 and UE_6 move more towards right, while UE_2, UE_3, UE_4 move evenly in both directions. The mean holding time is between 333 seconds and 714 seconds.

For SMP, the random holding time τ_i of UE_i at all MECs has a normal distribution $N(\mu_i, \sigma_i^2)$ with mean $\mu_i = 350 + 50i$ and variance $\sigma_i^2 = 0.01\mu_i^2$ ($\sigma_i = 0.1\mu_i$), whose pdf is

$$f_{\tau_i}(t) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-((t-\mu_i)/\sigma_i)^2/2},$$

and whose cdf is

$$F_{\tau_i}(t) = \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{t} e^{-((t'-\mu_i)/\sigma_i)^2/2} dt',$$

for all $0 \le i \le m-1$. It is well known that

$$X = \frac{\tau_i - \mu_i}{\sigma_i}$$

is a standard normal random variable with mean 0 and variance 1, whose pdf is

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2},$$

and whose cdf is

$$F_X(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-(x')^2/2} dx'.$$

Furthermore,

$$F_{\tau_i}(t) = \Phi\left(\frac{t-\mu_i}{\sigma_i}\right).$$

It is well known that the summation of independent normal random variables is also a normal random variable. Specifically, $S_{i,k}$, which is a summation of k i.i.d. normal random variable, has the normal distribution $N(k\mu_i, k\sigma_i^2)$. Therefore, we get

$$F_{S_{i,k}}(t) = \Phi\left(\frac{t-k\mu_i}{\sqrt{k}\sigma_i}\right),$$

and

$$\mathbf{P}[X_i(t)=k]=F_{S_{i,k}}(t)-F_{S_{i,k+1}}(t)=\Phi\left(\frac{t-k\mu_i}{\sqrt{k}\sigma_i}\right)-\Phi\left(\frac{t-(k+1)\mu_i}{\sqrt{k+1}\sigma_i}\right),$$

for all $k=0,1,2,\ldots$. Since a normal distribution is dominantly in the range $[\mu_i-4\sigma_i,\mu_i+4\sigma_i]$ (probability >0.9999), the summation in Eq. (10) is numerically calculated for k in the range

$$\left(\frac{\sqrt{4\sigma_i^2+\mu_i t}-2\sigma_i}{\mu_i}\right)^2\le k\le\left(\frac{\sqrt{4\sigma_i^2+\mu_i t}+2\sigma_i}{\mu_i}\right)^2-1.$$

The transition probability matrices of the embedded DTMC are set as:

$$\boldsymbol{P}'_0=\boldsymbol{P}'_5=\begin{bmatrix}0.0&1.0&0.0&0.0&0.0\\0.6&0.0&0.4&0.0&0.0\\0.0&0.6&0.0&0.4&0.0\\0.0&0.0&0.6&0.0&0.4\\0.0&0.0&0.0&1.0&0.0\end{bmatrix},$$

$$\boldsymbol{P}'_1=\boldsymbol{P}'_6=\begin{bmatrix}0.0&1.0&0.0&0.0&0.0\\0.4&0.0&0.6&0.0&0.0\\0.0&0.4&0.0&0.6&0.0\\0.0&0.0&0.4&0.0&0.6\\0.0&0.0&0.0&1.0&0.0\end{bmatrix},$$

$$\boldsymbol{P}'_2=\boldsymbol{P}'_3=\boldsymbol{P}'_4=\begin{bmatrix}0.0&1.0&0.0&0.0&0.0\\0.5&0.0&0.5&0.0&0.0\\0.0&0.5&0.0&0.5&0.0\\0.0&0.0&0.5&0.0&0.5\\0.0&0.0&0.0&1.0&0.0\end{bmatrix}.$$

Again, the above matrices are set in such a way that UE_0 and UE_5 move more towards left, UE_1 and UE_6 move more towards right, while UE_2, UE_3, UE_4 move evenly in both directions. The mean holding times are between 350 seconds and 650 seconds. For $i<i'$, UE_i has shorter holding time than $\mathrm{UE}_{i'}$, thus moves more quickly than $\mathrm{UE}_{i'}$.

5.2. Numerical data

We now display and discuss our numerical data.

In Table 3, we display $T_i^{(t)}$, T_i^*, and $T_i^{(\infty)}$, for all $0\le i\le m-1$, and $t=1,2,3,\ldots,10$, for discrete-time Markov chains.

In Table 4, we display $T_i^{(t)}$, T_i^*, and $T_i^{(\infty)}$, for all $0\le i\le m-1$, and $t=500,1000,1500,\ldots,5000$, for continuous-time Markov chains.

In Table 5, we display $T_i^{(t)}$, T_i^*, and $T_i^{(\infty)}$, for all $0\le i\le m-1$, and $t=500,1000,1500,\ldots,5000$, for semi-Markov processes.

It is easily observed that as time goes on, the average response times $T_i^{(t)}$ of some UEs (UE_0, UE_1, UE_2, UE_3 in Table 3, and UE_0, UE_1, UE_2 in Table 4) increase and then decrease, while the average response times of other UEs (UE_4, UE_5, UE_6 in Table 3, and UE_3, UE_4, UE_5, UE_6 in Table 4) always increase. The data in Fig. 5 are very irregular. As time goes on, $T_i^{(t)}$ can jump up and down. For each UE_i, the average response time $T_i^{(t)}$ eventually reaches its stationary value $T_i^{(\infty)}$.

We would like to clarify and confirm that the UEs have identical average task response times only when they are at the same MEC. However, they are likely to have different $T_i^{(t)}$, T_i^*, and $T_i^{(\infty)}$.

One important observation is that for all DTMC, CTMC, and SMP, $T_i^{(t)}$, T_i^*, and $T_i^{(\infty)}$, are longer than $T_i(974)$ when the UEs do not move, for all $0\le i\le m-1$, especially for UE_2, UE_3, UE_4. In other words, random mobility damages the performance of the UEs.

This can be explained using the classic *balls and bins model*. Let us consider m balls and n boxes (or bins), where $m\ge n$. Each ball is independently thrown into a bin that is chosen uniformly at random. A *placement* is a distribution of the balls among the bins. There are n^m placements. We are interested in $F(m,n)$, the number of placements which result in no empty bin.

Theorem 9. *We have the following recurrence relation to calculate $F(m,n)$:*

$$F(m,1)=1,$$
$$F(m,n)=n^m-\left(\binom{n}{1}F(m,n-1)+\binom{n}{2}F(m,n-2)+\cdots+\binom{n}{n-1}F(m,1)\right),\ n>1.$$

Table 3
Performance predictions for randomly walking mobile UEs (DTMC).

	UE_0	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6
$T_i^{(1)}$	1.30274	1.34634	1.36511	1.38769	1.37804	1.38827	1.37487
$T_i^{(2)}$	1.30265	1.36019	1.40392	1.43741	1.41493	1.40779	1.38060
$T_i^{(3)}$	1.32489	1.39143	1.41894	1.44857	1.43055	1.44526	1.40396
$T_i^{(4)}$	1.33948	1.40030	1.42461	1.44829	1.43848	1.46431	1.41975
$T_i^{(5)}$	1.35026	1.40322	1.42789	1.44786	1.44477	1.47805	1.43230
$T_i^{(6)}$	1.35743	1.40103	1.42904	1.44718	1.44938	1.48654	1.44164
$T_i^{(7)}$	1.36248	1.39706	1.42926	1.44676	1.45313	1.49242	1.44902
$T_i^{(8)}$	1.36600	1.39222	1.42886	1.44644	1.45617	1.49641	1.45484
$T_i^{(9)}$	1.36852	1.38728	1.42816	1.44620	1.45868	1.49922	1.45952
$T_i^{(10)}$	1.37034	1.38255	1.42732	1.44602	1.46075	1.50120	1.46329
T_i^{*}	1.34448	1.38616	1.41831	1.44024	1.43849	1.46595	1.42798
$T_i^{(\infty)}$	1.37579	1.35253	1.41923	1.44524	1.47120	1.50671	1.48020

Table 4
Performance predictions for randomly walking mobile UEs (CTMC).

$t = 500$	UE_0	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6
$T_i^{(t)}$	1.33159	1.36164	1.38609	1.38479	1.39440	1.38923	1.39715
$T_i^{(2t)}$	1.34651	1.37859	1.40928	1.42565	1.42024	1.41933	1.42204
$T_i^{(3t)}$	1.35817	1.38800	1.41475	1.43124	1.43136	1.44350	1.44239
$T_i^{(4t)}$	1.36437	1.38999	1.41588	1.43226	1.43874	1.46009	1.45701
$T_i^{(5t)}$	1.36716	1.38816	1.41536	1.43260	1.44408	1.47126	1.46741
$T_i^{(6t)}$	1.36814	1.38486	1.41422	1.43279	1.44802	1.47884	1.47487
$T_i^{(7t)}$	1.36825	1.38130	1.41295	1.43293	1.45094	1.48407	1.48027
$T_i^{(8t)}$	1.36797	1.37801	1.41176	1.43302	1.45313	1.48772	1.48421
$T_i^{(9t)}$	1.36756	1.37520	1.41074	1.43310	1.45476	1.49031	1.48711
$T_i^{(10t)}$	1.36714	1.37291	1.40990	1.43316	1.45598	1.49217	1.48924
T_i^{*}	1.36069	1.37987	1.41009	1.42715	1.43917	1.46165	1.46017
$T_i^{(\infty)}$	1.36530	1.36507	1.40696	1.43332	1.45963	1.49728	1.49543

Table 5
Performance predictions for randomly walking mobile UEs (SMP).

$t = 500$	UE_0	UE_1	UE_2	UE_3	UE_4	UE_5	UE_6
$T_i^{(1t)}$	1.42119	1.43245	1.20672	1.34770	1.18796	1.40132	1.39919
$T_i^{(2t)}$	1.30236	1.24448	1.46930	1.47622	1.33078	1.56516	1.54194
$T_i^{(3t)}$	1.18864	1.56229	1.60712	1.49032	1.25982	1.65518	1.65422
$T_i^{(4t)}$	1.47587	1.39374	1.52190	1.47805	1.56107	1.33897	1.46138
$T_i^{(5t)}$	1.38380	1.33863	1.38043	1.45664	1.39909	1.50085	1.47068
$T_i^{(6t)}$	1.50869	1.49757	1.55216	1.41804	1.61634	1.48060	1.61237
$T_i^{(7t)}$	1.40974	1.29626	1.49056	1.47553	1.35457	1.56245	1.52963
$T_i^{(8t)}$	1.61103	1.41043	1.54908	1.49850	1.67734	1.72303	1.63456
$T_i^{(9t)}$	1.39261	1.33737	1.43638	1.46200	1.48164	1.44412	1.50251
$T_i^{(10t)}$	1.38461	1.33727	1.41982	1.45517	1.48177	1.49569	1.44387
T_i^{*}	1.40785	1.38505	1.46335	1.45582	1.43504	1.51674	1.52503
$T_i^{(\infty)}$	1.39975	1.37962	1.44108	1.46675	1.49237	1.52803	1.50481

Proof. Let $E(m, n, k)$ denote the number of placements which result in exactly k empty bins, where $0 \le k \le n-1$. Then, we have

$$E(m,n,0) + E(m,n,1) + E(m,n,2) + \cdots + E(m,n,n-1) = n^m.$$

It is easy to see that

$$E(m,n,k) = \binom{n}{k} F(m, n-k),$$

for all $0 \le k \le n-1$, where $\binom{n}{k}$ is the number of ways to choose k empty bins from n bins, and $F(m, n-k)$ is the number of placements of the m balls into the remaining $(n-k)$ non-empty bins. Since $F(m,n) = E(m,n,0)$ is the number of placements which result in no empty bin, we have the following recurrence relation to calculate $F(m,n)$:

$$F(m,n) = n^m - \sum_{k=1}^{n-1} E(m,n,k) = n^m - \sum_{k=1}^{n-1} \binom{n}{k} F(m,n-k),$$

for $n > 1$. The base case of $F(m,1) = 1$ is straightforward. □

For instance, when $m = 7$ and $n = 1, 2, 3, 4, 5, 6, 7$, it is straightforward to verify that

$$\begin{array}{lll}
n=1:\ F(7,1)=1, & 1^m=1, & F(7,1)/1^m = 100.0\%;\\
n=2:\ F(7,2)=126, & 2^m=128, & F(7,2)/2^m = 98.4\%;\\
n=3:\ F(7,3)=1806, & 3^m=2187, & F(7,3)/3^m = 82.6\%;\\
n=4:\ F(7,4)=8400, & 4^m=16384, & F(7,4)/4^m = 51.3\%;\\
n=5:\ F(7,5)=16800, & 5^m=78125, & F(7,5)/5^m = 21.5\%;\\
n=6:\ F(7,6)=15120, & 6^m=279936, & F(7,6)/6^m = 5.4\%;\\
n=7:\ F(7,7)=5040=7!, & 7^m=823543, & F(7,7)/7^m = 0.6\%.
\end{array}$$

The above calculation means that for a fixed m, as n increases, the percentage of placements which result in no empty bins reduces quickly.

If the balls and bins are interpreted as UEs and MECs, the placements of balls into the bins are location distributions of the UEs among the MECs. $E(m,n,k)$ is the number of location distributions which result in exactly k idle MECs which serve no UEs. $F(m,n)$ is the number of location distributions which result in no idle MEC, i.e., the MECs are fully utilized. Taking our case of $m = 7$ and $n = 5$ as an example, we have

$$E(7,5,0) = 16800,$$
$$E(7,5,1) = 42000,$$
$$E(7,5,2) = 18060,$$
$$E(7,5,3) = 1260,$$
$$E(7,5,4) = 5.$$

Therefore, the chances to have one and two idle MECs are as high as

$$E(7,5,1)/5^7 = 42000/78125 = 53.8\%,$$
$$E(7,5,2)/5^7 = 18060/78125 = 23.1\%,$$

while the chance to have no idle MEC is only

$$E(7,5,0)/5^7 = 16800/78125 = 21.5\%.$$

The expected fraction of empty bins (i.e., idle MECs) is

$$\left(1 - \frac{1}{n}\right)^m \approx \frac{1}{e^{m/n}} \text{ for large } n,$$

which is $1/e = 36.8\%$ when $m = n$, and $1/e^2 = 13.5\%$ when $m = 2n$. In our case, it is $0.8^7 = 21.0\%$.

Due to random motion and distribution of the UEs, the MECs are not most efficiently utilized. It is not surprising to see that the average response times of randomly walking UEs are longer than the ones when they are still and static, and each MEC serves one or two UEs, such as $J = 974$.

Furthermore, random mobility could result in dense gathering of UEs, and surge and fluctuate the average response time. For instance, when the UEs walk according to SMP, at $t = 4000$,

$$\pi_0^{(t)} = (0.06133, 0.59008, 0.06815, 0.26226, 0.01817),$$
$$\pi_1^{(t)} = (0.06164, 0.15358, 0.23116, 0.34556, 0.20805),$$
$$\pi_2^{(t)} = (0.08904, 0.32192, 0.17808, 0.32192, 0.08904),$$
$$\pi_3^{(t)} = (0.12509, 0.24982, 0.25018, 0.24982, 0.12509),$$
$$\pi_4^{(t)} = (0.03909, 0.42181, 0.07819, 0.42181, 0.03909),$$
$$\pi_5^{(t)} = (0.04449, 0.61816, 0.04943, 0.27474, 0.01318),$$
$$\pi_6^{(t)} = (0.03270, 0.22595, 0.12262, 0.50838, 0.11035).$$

It is observed that the probabilities for UE_0, UE_2, UE_4, UE_5, UE_6 to be at MEC_1 and MEC_3 are very high. Hence, $T_i^{(4000)}$ is noticeably longer than usual for $i = 0, 2, 4, 5, 6$ (see Table 5).

6. Homogeneous UEs and MECs

In this section, we derive performance predictions in closed-form for homogeneous UEs and MECs.

Homogeneous UEs have the same λ_i, $\overline{r_i}$, $\overline{r_i^2}$, $\overline{d_i}$, $\overline{d_i^2}$, s_i, $\boldsymbol{P}_i$ for DTMC (or $\boldsymbol{Q}_i$ for CTMC, or $\tau_i, \boldsymbol{P}'_i$ for SMP), for all $0 \le i \le m-1$, and the same $c_{i,j}$, for all $0 \le i \le m-1$ and $0 \le j \le n-1$. Homogeneous MECs have the same $\tilde{s}_j$, for all $0 \le j \le n-1$.

Furthermore, we assume that UE_i always moves into the MEC_j's with equal probability.

Theorem 10. *For all DTMC/CTMC/SMP, if UE_i moves into the MEC_j's with equal probability, a perfectly balanced location distribution is maintained after each location change and at any time.*

Proof. The proof is given in Appendix A. □

Notice that $\pi_i^{(t)}(j) = 1/n$ for all $0 \le i \le m-1$ and $0 \le j \le n-1$ and $t \ge 0$ implies that $\pi^{(t)}(J) = 1/N$, for all $0 \le J \le N-1$ and $t \ge 0$ (see Eq. (4)), that is, all location distributions are equally probable.

Consider UE_i which is at MEC_j. Let $b = |I_j|$, which is a random variable with support $\{1, 2, ..., m\}$. The following theorem gives its pmf.

Theorem 11. *The probability mass function of b is*

$$\mathbf{P}[|I_j| = b] = \frac{1}{n^{m-1}} \binom{m-1}{b-1} (n-1)^{m-b}, \tag{15}$$

for all $1 \le b \le m$.

Proof. The above result can be explained as follows. Except UE_i, there are $(m-1)$ UEs, which have n^{m-1} different location distributions among the n MECs. There are $\binom{m-1}{b-1}$ different ways to choose $(b-1)$ UEs which join I_j. For the remaining $(m-b)$ UEs, there are $(n-1)^{m-b}$ different location distributions. The result follows if all location distributions are equally probable. □

Let $T_{i,b}$ represent the average response time of UE_i when $|I_j| = b$. A perfectly balanced location distribution gives rise to

$$T_i^{(t)} = T_i^* = T_i^{(\infty)} = \sum_{b=1}^{m} \mathbf{P}[|I_j| = b] T_{i,b}, \tag{16}$$

for all $t \ge 0$.

In the following, we derive a closed-form expression of $T_{i,b}$. Since all UEs are homogeneous, we have $\tilde{\lambda}_j = b\hat{\lambda}_i$. To guarantee $T_i = \tilde{T}_j = T_{i,b}$, we need

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

that is,

$$(\overline{x_i} - \overline{\tilde{x}_j}) - \frac{\overline{x_i^2}(\hat{\lambda}_i - \lambda_i)}{2(\overline{x_i}\hat{\lambda}_i + (1 - \lambda_i\overline{x_i}))} + \frac{b\overline{\tilde{x}_j^2}\hat{\lambda}_i}{2(b\overline{\tilde{x}_j}\hat{\lambda}_i - 1)} = 0,$$

which is

$$\begin{aligned} &2(\overline{x_i} - \overline{\tilde{x}_j})(\overline{x_i}\hat{\lambda}_i + (1 - \lambda_i\overline{x_i}))(b\overline{\tilde{x}_j}\hat{\lambda}_i - 1) \\ &\quad -\overline{x_i^2}(\hat{\lambda}_i - \lambda_i)(b\overline{\tilde{x}_j}\hat{\lambda}_i - 1) + b\overline{\tilde{x}_j^2}\hat{\lambda}_i(\overline{x_i}\hat{\lambda}_i + (1 - \lambda_i\overline{x_i})) = 0. \end{aligned}$$

We view the above equation as a quadratic equation of $\hat{\lambda}_i$ and obtain:

$$\begin{aligned} &2(\overline{x_i} - \overline{\tilde{x}_j})(b\overline{x_i}\overline{\tilde{x}_j}\hat{\lambda}_i^2 + (b\overline{\tilde{x}_j}(1 - \lambda_i\overline{x_i}) - \overline{x_i})\hat{\lambda}_i - (1 - \lambda_i\overline{x_i})) \\ &\quad -\overline{x_i^2}(b\overline{\tilde{x}_j}\hat{\lambda}_i^2 - (b\lambda_i\overline{\tilde{x}_j} + 1)\hat{\lambda}_i + \lambda_i) + b\overline{\tilde{x}_j^2}(\overline{x_i}\hat{\lambda}_i^2 + (1 - \lambda_i\overline{x_i})\hat{\lambda}_i) = 0, \end{aligned}$$

which is reorganized as:

$$(2b\overline{x_i}\overline{\tilde{x}_j}(\overline{x_i}-\overline{\tilde{x}_j})-b\overline{x_i^2}\overline{\tilde{x}_j}+b\overline{x_i}\overline{\tilde{x}_j^2})\hat{\lambda}_i^2$$
$$+(2(\overline{x_i}-\overline{\tilde{x}_j})(b\overline{\tilde{x}_j}(1-\lambda_i\overline{x_i})-\overline{x_i})+\overline{x_i^2}(b\lambda_i\overline{\tilde{x}_j}+1)+b\overline{\tilde{x}_j^2}(1-\lambda_i\overline{x_i}))\hat{\lambda}_i$$
$$-(2(\overline{x_i}-\overline{\tilde{x}_j})(1-\lambda_i\overline{x_i})+\lambda_i\overline{x_i^2})=0,$$

and further rewritten as:

$$A\hat{\lambda}_i^2+B\hat{\lambda}_i+C=0,$$

where

$$A=2b\overline{x_i}\overline{\tilde{x}_j}(\overline{x_i}-\overline{\tilde{x}_j})-b\overline{x_i^2}\overline{\tilde{x}_j}+b\overline{x_i}\overline{\tilde{x}_j^2},$$
$$B=2(\overline{x_i}-\overline{\tilde{x}_j})(b\overline{\tilde{x}_j}(1-\lambda_i\overline{x_i})-\overline{x_i})+\overline{x_i^2}(b\lambda_i\overline{\tilde{x}_j}+1)+b\overline{\tilde{x}_j^2}(1-\lambda_i\overline{x_i}),$$
$$C=-(2(\overline{x_i}-\overline{\tilde{x}_j})(1-\lambda_i\overline{x_i})+\lambda_i\overline{x_i^2}),$$

which gives

$$\hat{\lambda}_i=\frac{\sqrt{B^2-4AC}-B}{2A}. \tag{17}$$

Based on $\hat{\lambda}_i$, we can calculate $T_{i,b}$.

The above discussion can be summarized as follows.

Theorem 12. *For homogeneous UEs which move into the MECs with equal probability, and whose mobility is modeled by DTMC/CTMC/SMP with perfectly balanced location distributions, our performance predictions can be calculated using Eqs. (15), (16), and (17).*

In Fig. 4, we show $T_{i,b}$ as a function of ρ_i for $b=1,2,3,4,5,6,7$, with the following parameter setting: $\overline{r_i}=1.8$ BI, $\overline{r_i^2}=1.1\overline{r_i}^2$ BI^2, $\overline{d_i}=2.6$ MB, $\overline{d_i^2}=1.1\overline{d_i}^2$ MB^2, $s_i=2.3$ BI/second, $\overline{x_i}=\overline{r_i}/s_i$ second, $\overline{x_i^2}=\overline{r_i^2}/s_i^2$ second^2, for all $0\le i\le m-1$; $\tilde{s}_j=3.9$ BI/second, for all $0\le j\le n-1$; and $c_{i,j}=14.0$ MB/second, for all $0\le i\le m-1$ and $0\le j\le n-1$. For a given ρ_i, our performance predictions $T_i^{(t)}$, T_i^*, $T_i^{(\infty)}$ obtained from Eq. (16) are weighted average of the data in Fig. 4 according to the weights from Eq. (15). When $m=7$, we have

$$\mathbf{P}[|I_j|=1]=0.26214,$$
$$\mathbf{P}[|I_j|=2]=0.39322,$$
$$\mathbf{P}[|I_j|=3]=0.24576,$$
$$\mathbf{P}[|I_j|=4]=0.08192,$$
$$\mathbf{P}[|I_j|=5]=0.01536,$$
$$\mathbf{P}[|I_j|=6]=0.00154,$$
$$\mathbf{P}[|I_j|=7]=0.00006.$$

$T_i^{(t)}$, T_i^*, $T_i^{(\infty)}$ are very close the curve for $b=2$.

7. Mobility cost

In this section, we consider mobility cost and service delay for location change.

When UE$_i$ changes its location from MEC$_{j'}$ to MEC$_j$, UE$_i$ loses service from MEC$_{j'}$, and gains service from MEC$_j$ after Δ amount of time (called the *transition time*), which is the time for UE$_i$ to register with MEC$_j$, to establish communication connection with MEC$_j$, and to create a virtual machine in MEC$_j$. During this period of time, UE$_i$ is in the *transition state*; it can still perform its own local processing, but there is no remote service. It is interesting to include such penalty on computing power due to location change into our mathematical models, and evaluate its impact on performance in our analytical methods.

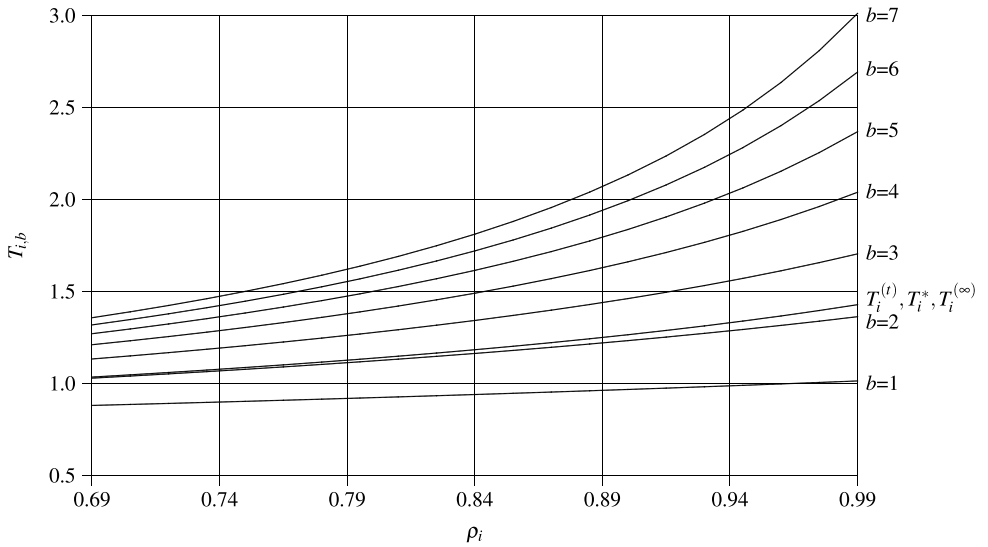

Fig. 4. $T_{i,b}$, $T_i^{(t)}$, T_i^*, $T_i^{(\infty)}$ as functions of ρ_i.

7.1. Discrete-time Markov chains

Consider UE_i which is at MEC_{j_i}. We need to update $T_i(J)$ in Eq. (11) to reflect mobility cost. It turns out that $T_i(J)$ not only depends on location distribution J, but also relies on time t. Thus, our notation is extended to $T_i^{(t)}(J)$, which is the average response time of tasks generated on UE_i under J at time t. It is clear that during time slot t, in the beginning Δ amount of time, not all UE_i's in I_{j_i}, but only those which remain at MEC_{j_i} from time $t-1$ to t, share the service of MEC_{j_i}. In the remaining $1-\Delta$ amount of time, all UE_i's in I_{j_i} share the service of MEC_{j_i}. For this reason, we introduce the notation $T_{i,j_i}(I)$, i.e., the average response time of tasks offloaded from UE_i to MEC_{j_i} when only UEs in I share the service of MEC_{j_i}, which is

$$\tilde{T}_{i,j_i}(I) = \overline{\tilde{x}_{i,j_i}} + \tilde{W}_{j_i}(I),$$

where

$$\tilde{W}_{j_i}(I) = \frac{\tilde{\lambda}_{j_i}\overline{\tilde{x}_{j_i}^2}}{2(1-\tilde{\rho}_{j_i})},$$

with

$$\tilde{\lambda}_j = \sum_{i\in I}\hat{\lambda}_i.$$

The following theorem gives $T_i^{(t)}(J)$.

Theorem 13. *$T_i^{(t)}(J)$ is calculated as follows:*

$$T_i^{(t)}(J) = \frac{1}{\pi^{(t)}(J)}\sum_{J'}\pi^{(t-1)}(J')p(J',J)\Bigg(\Delta((j_i'=j_i)\tilde{T}_{i,j_i}(I_{j_i'}\cap I_{j_i}) + (j_i'\neq j_i)T_i') \tag{18}$$

$$+(1-\Delta)\left(\left(\frac{\lambda_i-\hat{\lambda}_i}{\lambda_i}\right)T_i + \frac{\hat{\lambda}_i}{\lambda_i}\tilde{T}_{i,j_i}(I_{j_i})\right)\Bigg),$$

for all $t \geq 1$. (Recall that T_i' is the average task response time of UE_i without any offloading.)

(Note: A logical expression (i.e., $j_i'=j_i$, $j_i'\neq j_i$) takes value 1 or 0, depending on whether it is true or false.)

Proof. Assume that at certain time t, the environment has location distribution $J=(j_0,...,j_i,...,j_{m-1})$. The probability that at time $t-1$, the environment has location distribution $J'=(j_0',...,j_i',...,j_{m-1}')$ is

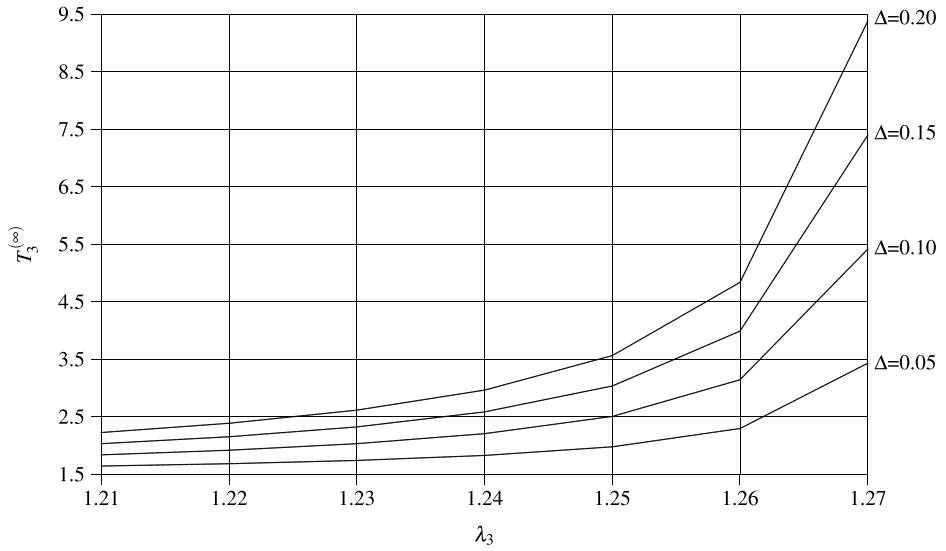

Fig. 5. $T_3^{(\infty)}$ as a function of λ_3 (DTMC).

$$\frac{\pi^{(t-1)}(J')p(J',J)}{\pi^{(t)}(J)}.$$

For DTMC, Δ is actually the percentage of time during which a UE loses remote service due to movement. If $j_i' = j_i$, $I_{j_i'} - I_{j_i}$ is the set of UEs which leave MEC_{j_i}; $I_{j_i'} \cap I_{j_i}$ is the set of UEs which remain in MEC_{j_i}; $I_{j_i} - I_{j_i'}$ is the set of UEs which newly move to MEC_{j_i}. During the beginning Δ amount of time, if $j_i' = j_i$, UE_i does not change its location, i.e., $i \in I_{j_i'} \cap I_{j_i}$. Thus, its average response time is $\tilde{T}_{i,j_i}(I_{j_i'} \cap I_{j_i})$, where only UEs in $I_{j_i'} \cap I_{j_i}$ are served by MEC_{j_i}. If $j_i' \neq j_i$, UE_i changes its location from $\text{MEC}_{j_i'}$ to MEC_{j_i} and has no MEC_{j_i} service. Thus, its average response time is T_i', i.e., the average response time of UE_i without any offloading. During the remaining $1-\Delta$ amount of time, all UEs in I_{j_i} are served by MEC_{j_i}. Thus, UE_i's average response time is the same as that in Eq. (11). Summarizing all the above arguments, we can obtain the theorem. □

For all $t \geq 1$, the instantaneous average response time $T_i^{(t)}$ of tasks generated on UE_i at time t is

$$T_i^{(t)} = \sum_{J=0}^{N-1} \pi^{(t)}(J) T_i^{(t)}(J).$$

Let us consider a mobile edge computing environment in its stationary state π.

Theorem 14. *$T_i^{(\infty)}(J)$ is calculated as follows:*

$$T_i^{(\infty)}(J) = \frac{1}{\pi(J)} \sum_{J'} \pi(J')p(J',J)\Bigg(\Delta((j_i' = j_i)\tilde{T}_{i,j_i}(I_{j_i'} \cap I_{j_i}) + (j_i' \neq j_i)T_i') \tag{19}$$
$$+(1-\Delta)\left(\left(\frac{\lambda_i - \hat{\lambda}_i}{\lambda_i}\right)T_i + \frac{\hat{\lambda}_i}{\lambda_i}\tilde{T}_{i,j_i}(I_{j_i})\right)\Bigg),$$

as $t \to \infty$.

Proof. We only need to notice that as $t \to \infty$, $\pi^{(t)}(J) \to \pi^{(\infty)}(J) = \pi(J)$ and $\pi^{(t-1)}(J') \to \pi^{(\infty)}(J') = \pi(J')$. □

The stationary average response time $T_i^{(\infty)}$ of tasks generated on UE_i is

$$T_i^{(\infty)} = \sum_{J=0}^{N-1} \pi(J) T_i^{(\infty)}(J).$$

In Fig. 5, we show $T_3^{(\infty)}$ as a function of λ_3, for $\Delta = 0.05, 0.10, 0.15, 0.20$, using the same parameter setting in Section 5.1. The value of λ_3 is close to making UE_3 saturated, so that the impact of Δ can be manifested more clearly. It is

easily observed that Δ has strong influence on UE_i's performance, especially when λ_i is large and ρ_i is high. The reason is that during the beginning Δ amount of time, UE_i may lose service and rely on its own computing power, which has long T_i' and results in significant increase of $T_i^{(\infty)}(J)$ and $T_i^{(\infty)}$.

7.2. Continuous-time Markov chains

We assume that the transition time Δ is an exponential random variable with parameter q_Δ.

Let the set of possible states of UE_i be

$$N_i = \{j_i \mid 0 \le j_i \le n-1\} \cup \{\bar{j}_i \mid 0 \le j_i \le n-1\},$$

for all $0 \le i \le m-1$, where both j_i and $\bar{j}_i$ mean that UE_i is at MEC_j. Furthermore, j_i means that UE_i is not in the transition state and has service from MEC_j, while $\bar{j}_i$ means that UE_i is in the transition state without service from MEC_j. Then, the Cartesian product

$$\mathcal{N} = N_0 \times N_1 \times \cdots \times N_{m-1}$$

gives the set of *joint states* of the m UEs. It is clear that $|N_i| = 2n$ and $|\mathcal{N}| = (2n)^m$. For convenience, $J \in \mathcal{N}$ is treated as an m-digit radix-$2n$ integer in the range $0, 1, ..., (2n)^m - 1$, where $\bar{j} = j + n$, for all $0 \le j \le n-1$.

The joint transition rate matrix of the joint CTMC

$$\boldsymbol{Q} = [q(J', J)]$$

can be described as follows. Let $J' = (j_0, ..., j_i', ..., j_{m-1})$ and $J = (j_0, ..., j_i, ..., j_{m-1})$. Since at any instant time, only one UE can move,

$$\begin{aligned} J' &= (j_0, \ ..., \ j_i', \quad ..., \ j_{m-1}) \\ &\qquad\quad \Big\downarrow \mathrm{UE}_i \\ J &= (j_0, \ ..., \ \bar{j}_i, \quad ..., \ j_{m-1}) \end{aligned}$$

we have

$$q(J', J) = q_i(j_i', j_i),$$

for all $0 \le i \le m-1$ and $0 \le j_0, ..., j_i' \ne j_i, ..., j_{m-1} \le n-1$. We also have

$$\begin{aligned} J' &= (j_0, \ ..., \ \bar{j}_i, \quad ..., \ j_{m-1}) \\ &\qquad\quad \Big\downarrow \mathrm{UE}_i \\ J &= (j_0, \ ..., \ j_i, \quad ..., \ j_{m-1}) \end{aligned}$$

that is,

$$q(J', J) = q_\Delta,$$

for all $0 \le i \le m-1$ and $0 \le j_r \le n-1$. Furthermore, we have

$$q(J, J) = -\sum_{J' \ne J} q(J, J'),$$

for all $0 \le J \le (2n)^m - 1$.

The above joint CTMC is actually decomposable. Let the movement of UE_i be governed by an $n \times n$ transition rate matrix

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

The above matrix can be extended to a $2n \times 2n$ transition rate matrix

$$\boldsymbol{R}_i = [r_i(j', j)],$$

where

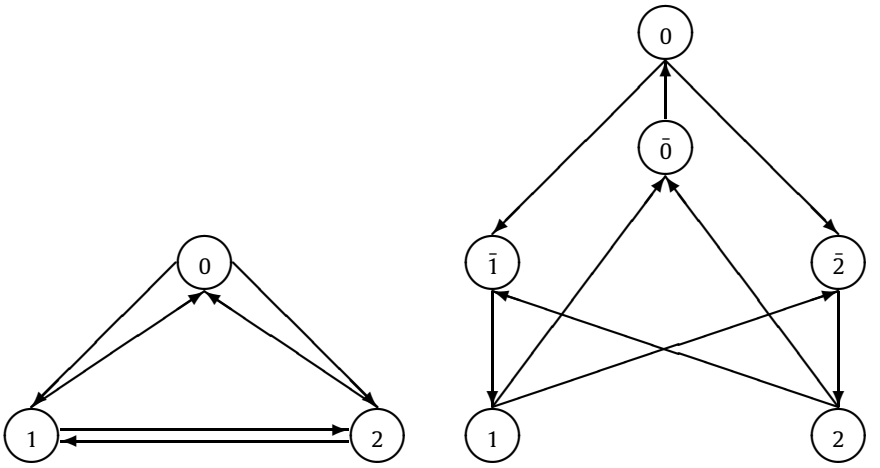

Fig. 6. Illustration of $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ $(n = 3)$.

$$
\begin{aligned}
&r_i(j', \bar{j}) = q_i(j', j),\\
&r_i(j, j) = q_i(j, j) = -\sum_{j' \neq j} q_i(j, j'),\\
&r_i(\bar{j}, j) = q_\Delta,\\
&r_i(\bar{j}, \bar{j}) = -q_\Delta,
\end{aligned}
$$

for all $0 \leq j' \neq j \leq n-1$ and $0 \leq j \leq n-1$. As an example, when $n = 3$, if

$$
\boldsymbol{Q}_i = \begin{bmatrix} q_i(0,0) & q_i(0,1) & q_i(0,2) \\ q_i(1,0) & q_i(1,1) & q_i(1,2) \\ q_i(2,0) & q_i(2,1) & q_i(2,2) \end{bmatrix},
$$

then $\boldsymbol{R}_i$ looks like

$$
\boldsymbol{R}_i = \begin{bmatrix} q_i(0,0) & 0 & 0 & 0 & q_i(0,1) & q_i(0,2) \\ 0 & q_i(1,1) & 0 & q_i(1,0) & 0 & q_i(1,2) \\ 0 & 0 & q_i(2,2) & q_i(2,0) & q_i(2,1) & 0 \\ q_\Delta & 0 & 0 & -q_\Delta & 0 & 0 \\ 0 & q_\Delta & 0 & 0 & -q_\Delta & 0 \\ 0 & 0 & q_\Delta & 0 & 0 & -q_\Delta \end{bmatrix}.
$$

Fig. 6 illustrates $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$ when $n = 3$. For clarity, we do not show self-loops.

With $\boldsymbol{R}_i$, we can get $\pi_i^{(t)}$, $\pi^{(t)}(J)$, π_i, $\pi(J)$, by using Theorems 5 and 6.

Actually, there is close connection between the stationary probability vectors of $\boldsymbol{Q}_i$ and $\boldsymbol{R}_i$. Let

$$
\pi_i = [\pi_i(0), \pi_i(1), ..., \pi_i(n-1), \pi_i(n), \pi_i(n+1), ..., \pi_i(2n-1)]
$$

be the stationary probability vector of $\boldsymbol{R}_i$. It can be verified that

$$
[\pi_i(0), \pi_i(1), ..., \pi_i(n-1)]\boldsymbol{Q}_i = 0.
$$

This implies that if

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

is the stationary probability vector of $\boldsymbol{Q}_i$, then there is a constant $\alpha < 1$, such that

$$
[\pi_i(0), \pi_i(1), ..., \pi_i(n-1)] = \alpha[\pi_i'(0), \pi_i'(1), ..., \pi_i'(n-1)].
$$

It can be verified that

$$
\begin{aligned}
&-(\pi_i(0)q_i(0,0) + \pi_i(1)q_i(1,1) + \cdots + \pi_i(n-1)q_i(n-1,n-1))\\
&\quad = (\pi_i(n) + \pi_i(n+1) + \cdots + \pi_i(2n-1))q_\Delta,
\end{aligned}
$$

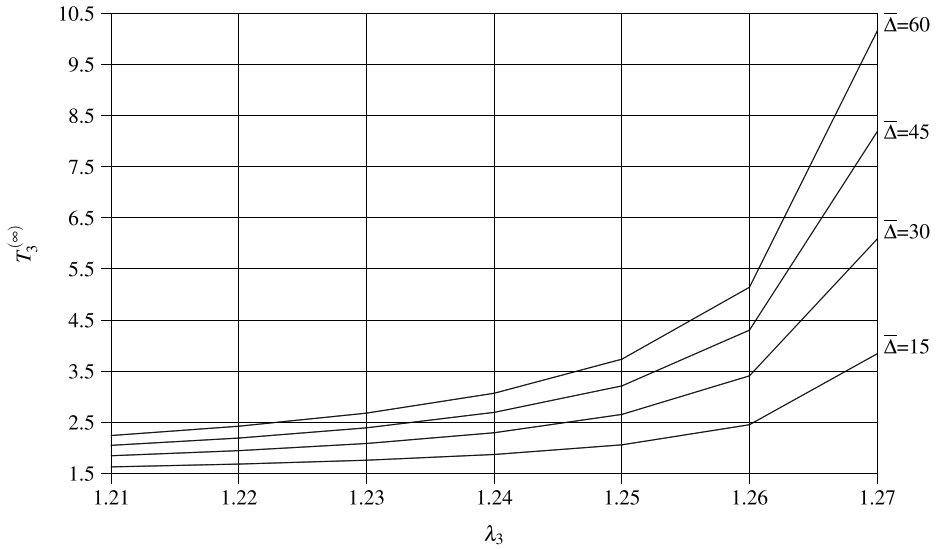

Fig. 7. $T_3^{(\infty)}$ as a function of λ_3 (CTMC).

that is,

$$\begin{aligned}
&-\alpha(\pi_i'(0)q_i(0,0)+\pi_i'(1)q_i(1,1)+\cdots+\pi_i'(n-1)q_i(n-1,n-1))\\
&\quad=(1-(\pi_i(0)+\pi_i(1)+\cdots+\pi_i(n-1)))q_\Delta\\
&\quad=(1-\alpha(\pi_i'(0)+\pi_i'(1)+\cdots+\pi_i'(n-1)))q_\Delta\\
&\quad=(1-\alpha)q_\Delta.
\end{aligned}$$

The last equation yields

$$\alpha=\frac{q_\Delta}{q_\Delta-(\pi_i'(0)q_i(0,0)+\pi_i'(1)q_i(1,1)+\cdots+\pi_i'(n-1)q_i(n-1,n-1))}.$$

If q_Δ is reduced, α is also reduced, while

$$\pi_i(n)+\pi_i(n+1)+\cdots+\pi_i(2n-1)=1-\alpha$$

is increased, which means that the probability for UE_i to be in the transition state is increased.

To evaluate our performance predictions (i.e., $T_i^{(t)}$ and $T_i^{(\infty)}$), we need $T_i(J)$, which is given by the following theorem. Notice that $T_i(J)$ is independent of t.

Theorem 15. *The average response time $T_i(J)$ of tasks generated on UE_i under $J=(j_0,\ldots,j_i,\ldots,j_{m-1})$ is*

$$\begin{aligned}
T_i(J)&=(UE_i\text{ is in transition state, i.e., }j_i\geq n)T_i'\\
&\quad+(UE_i\text{ is not in transition state, i.e., }j_i<n)\left(\left(\frac{\lambda_i-\hat{\lambda}_i}{\lambda_i}\right)T_i+\frac{\hat{\lambda}_i}{\lambda_i}\tilde{T}_{i,j_i}\right),
\end{aligned}\tag{20}$$

with I_{j_i} updated as follows:

$$I_{j_i}=\{i'\mid UE_{i'}\text{ is at }MEC_{j_i}\text{ and }UE_{i'}\text{ is not in transition state, i.e., }j_{i'}<n\}.$$

Proof. Straightforward. □

In Fig. 7, we show $T_3^{(\infty)}$ as a function of λ_3, using the same parameter setting in Section 5.1, where q_Δ is chosen such that the mean transition time $\overline{\Delta}=1/q_\Delta$ is 15, 30, 45, 60 seconds. The curves exhibit similar behavior as those in Fig. 5.

As mentioned earlier, for CTMC, the mean holding time is between 333 seconds and 714 seconds. Therefore, in terms of percentage, the mean transition time in Fig. 7 is shorter than that of DTMC in Fig. 5. However, $T_3^{(\infty)}$ is longer than that in Fig. 5. This means that randomized transition time has stronger impact on performance than constant transition time due to increased uncertainty.

8. MEC speed setting

In this section, we discuss speed setting for MECs with power consumption constraint.

8.1. MEC-centric performance measures

In this section, we define performance measures for MECs.

Let $\tilde{T}_j(I_j)$ be the average response time of MEC$_j$ serving UEs in I_j, which can be calculated by using Theorem 1. The *instantaneous average response time* $\tilde{T}_j^{(t)}$ of MEC$_j$ at time t is

$$\tilde{T}_j^{(t)} = \sum_{J=0}^{N-1} \pi^{(t)}(J)\tilde{T}_j(I_j(J)), \tag{21}$$

where, for a location distribution $J = (j_0, ..., j_i, ..., j_{m-1})$,

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

is the set of UEs at MEC$_j$. (Recall that $N = n^m$ is the number of location distributions.) The *stationary average response time* $\tilde{T}_j^{(\infty)}$ of MEC$_j$ is

$$\tilde{T}_j^{(\infty)} = \sum_{J=0}^{N-1} \pi(J)\tilde{T}_j(I_j(J)). \tag{22}$$

$\tilde{T}_j^{(\infty)}$ is our ultimate performance prediction for MEC$_j$ in a fog computing environment with mobile users.

Both Eqs. (21) and (22) involve $N = n^m$ terms. In fact, both summations can be simplified as follows. Consider $I_j \subseteq \{0, 1, ..., m-1\}$. It is clear that I_j occurs with probability

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

at time t, and 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, $\tilde{T}_j^{(t)}$ can be calculated by

$$\tilde{T}_j^{(t)} = \sum_{I_j \neq \emptyset} P_j^{(t)}(I_j)\tilde{T}_j(I_j), \tag{23}$$

and $\tilde{T}_j^{(\infty)}$ can be calculated by

$$\tilde{T}_j^{(\infty)} = \sum_{I_j \neq \emptyset} P_j(I_j)\tilde{T}_j(I_j), \tag{24}$$

for all $0 \leq j \leq n-1$. Both Eqs. (23) and (24) involve $M = 2^m$ terms.

8.2. Random location distribution model

In this section, we introduce the random location distribution model.

Notice that the evaluation of $\tilde{T}_j^{(\infty)}$ relies on $\pi_i(j)$, i.e., the probability that UE$_i$ is in the service area of MEC$_j$.

Our *random location distribution model* includes the following parameters. Let a geographic area be

$$\Omega = [-\infty, +\infty] \times [-\infty, +\infty].$$

For UE$_i$, we specify a function

$$f_i(x, y) : \Omega \to [0, 1],$$

which gives the probability density function of the location of UE$_i$. For MEC$_j$, we specify its service area $\Omega_j \subseteq \Omega$. Then, it is clear that

$$\pi_i(j) = \iint_{\Omega_j} f_i(x, y)dxdy,$$

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

If a UE is not in the service area of any MEC, which happens with probability

$$1 - \sum_{j=0}^{n-1} \pi_i(j),$$

the UE has no mobile edge computing service.

For MEC_j, it is possible that there is no any UE in its service area, which happens with probability

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

at time t, and

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

in a stationary state. Therefore, the probability that there is at least one UE at MEC_j (i.e., MEC_j is busy) is

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

at time t, and

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

in a stationary state.

Since there is chance for MEC_j to be idle, the average response times given in Eqs. (23) and (24) should be modified as:

$$\tilde{T}_j^{(t)} = \frac{1}{B_j} \sum_{I_j \ne \emptyset} P_j^{(t)}(I_j)\tilde{T}_j(I_j), \tag{25}$$

and

$$\tilde{T}_j^{(\infty)} = \frac{1}{B_j} \sum_{I_j \ne \emptyset} P_j(I_j)\tilde{T}_j(I_j), \tag{26}$$

for all $0 \le j \le n-1$. That is to say, we need to predict the average response time under the condition that MEC_j is working.

8.3. Power constrained speed setting

In this section, we define our power constrained speed setting problem.

To discuss power constrained speed setting, we need power consumption models. We consider two types of server speed and power consumption models.

- In the *idle-speed model*, the power consumption (measured by Watts) of MEC_j is

  $$P_j = \beta_j(B_j\xi_j\tilde{s}_j^{\alpha_j} + P_j^*),$$

  for all $0 \le j \le n-1$.
- In the *constant-speed model*, the power consumption of MEC_j is

  $$P_j = \beta_j(\xi_j\tilde{s}_j^{\alpha_j} + P_j^*),$$

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

In both models, ξ_j and α_j are technology-dependent constants that determine the dynamic component of power consumption, P_j^* is the static component of power consumption, and β_j is the power usage effectiveness (PUE).

Our *power constrained speed setting problem* is to find MEC computing speeds $\tilde{s}_0$, $\tilde{s}_1$, ..., $\tilde{s}_{n-1}$, such that the maximum stationary average response time

$$\tilde{T}_{\max}^{(\infty)} = \max_{0 \le j \le n-1} \{\tilde{T}_j^{(\infty)}\}$$

is minimized and that

$$P_0 + P_1 + \cdots + P_{n-1} = P,$$

where P is a given power constraint.

8.4. The algorithm

In this section, we develop an algorithm to solve the power constrained speed setting problem.

Note that $\tilde{T}_{\max}^{(\infty)}$ is minimized if and only if

$$\tilde{T}_0^{(\infty)} = \tilde{T}_1^{(\infty)} = \cdots = \tilde{T}_{n-1}^{(\infty)} = T.$$

The value of T can be found by bisection search in an interval $[0, T']$, based on the fact that $P_0 + P_1 + \cdots + P_{n-1}$ is a decreasing function of T. For a given T, we can find $\tilde{s}_0$, $\tilde{s}_1$, ..., $\tilde{s}_{n-1}$. The value of $\tilde{s}_j$ can also be found by bisection search in an interval $[0, \tilde{s}'_j]$, based on the fact that $\tilde{T}_j^{(\infty)}$ is a decreasing function of $\tilde{s}_j$. If $\tilde{T}_j^{(\infty)} > T$; we increase $\tilde{s}_j$; otherwise, we decrease $\tilde{s}_j$. Using $\tilde{s}_0$, $\tilde{s}_1$, ..., $\tilde{s}_{n-1}$, we can calculate P_0, P_1, ..., P_{n-1}. If $P_0 + P_1 + \cdots + P_{n-1} > P$, we increase T; otherwise, we decrease T.

The value of T' can be determined as follows. Initially, T' is set at any reasonable value. Then, T' is doubled until $P_0 + P_1 + \cdots + P_{n-1} < P$. The value of $\tilde{s}'_j$ can be determined as follows. Initially, $\tilde{s}'_j$ is set at any reasonable value. Then, $\tilde{s}'_j$ is doubled until $\tilde{T}_j^{(\infty)} < T$.

8.5. Numerical data

In this section, we demonstrate numerical data.

We consider two types of location distributions for the UEs.

- Normal distribution – The x- and y-coordinates of UE_i are independent random variables with normal distributions $N(\mu_{x,i}, \sigma_{x,i}^2)$ and $N(\mu_{y,i}, \sigma_{y,i}^2)$ respectively. Hence, the probability density function of the location of UE_i is

  $$f_i(x, y) = \frac{1}{\sqrt{2\pi}\sigma_{x,i}} e^{-((x-\mu_{x,i})/\sigma_{x,i})^2/2} \times \frac{1}{\sqrt{2\pi}\sigma_{y,i}} e^{-((y-\mu_{y,i})/\sigma_{y,i})^2/2},$$

  which is actually

  $$f_i(x, y) = \frac{1}{2\pi\sigma_{x,i}\sigma_{y,i}} e^{-(((x-\mu_{x,i})/\sigma_{x,i})^2 + ((y-\mu_{y,i})/\sigma_{y,i})^2)/2},$$

  for all $0 \le i \le m-1$. The probability that UE_i is in the service area of MEC_j is

  $$\pi_i(j) = \frac{1}{2\pi\sigma_{x,i}\sigma_{y,i}} \iint_{\Omega_j} e^{-(((x-\mu_{x,i})/\sigma_{x,i})^2 + ((y-\mu_{y,i})/\sigma_{y,i})^2)/2} dxdy,$$

  for all $0 \le i \le m-1$ and $0 \le j \le n-1$.
- Uniform distribution – The location of UE_i is uniformly distributed in a rectangle R_i centered at $(\mu_{x,i}, \mu_{y,i})$ with size $2\sigma_{x,i} \times 2\sigma_{y,i}$:

  $$R_i = \{(x, y) \mid |x - \mu_{x,i}| \le \sigma_{x,i} \text{ and } |y - \mu_{y,i}| \le \sigma_{y,i}\}.$$

  Hence, the probability density function of the location of UE_i is

  $$f_i(x, y) = \frac{1}{4\sigma_{x,i}\sigma_{y,i}}, \quad (x, y) \in R_i,$$

  for all $0 \le i \le m-1$. The probability that UE_i is in the service area of MEC_j is

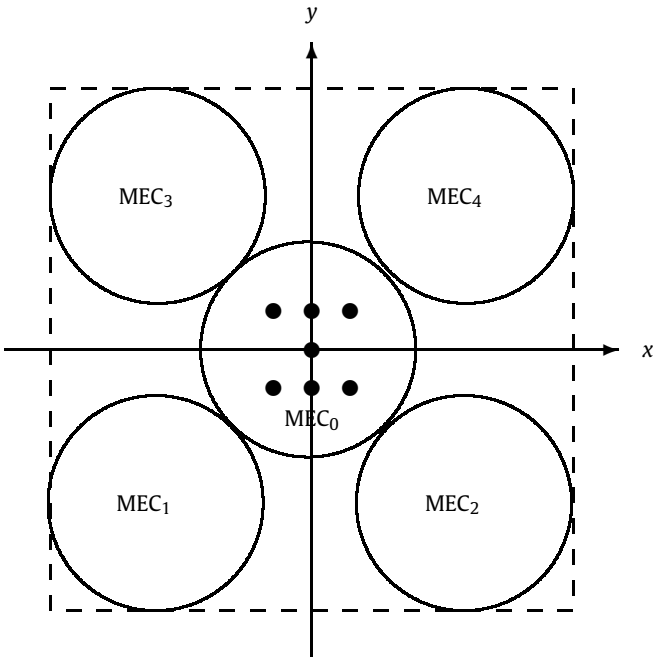

Fig. 8. The service areas of five MECs and the centers of seven UEs' location distributions.

$$\pi_i(j) = \frac{1}{4\sigma_{x,i}\sigma_{y,i}} \iint\limits_{R_i \cap \Omega_j} dxdy,$$

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

We consider $m = 7$ UEs and $n = 5$ MECs with the same parameter setting as Section 5.1.

As shown in Fig. 8, the service area of MEC_j is a circle centered at (x_j, y_j) with radius γ_j:

$$\Omega_j = \{(x, y) \mid (x - x_j)^2 + (y - y_j)^2 \le \gamma_j^2\},$$

for all $0 \le j \le n-1$. The centers are $(x_0, y_0) = (0, 0)$, $(x_1, y_1) = (-2, -2)$, $(x_2, y_2) = (2, -2)$, $(x_3, y_3) = (-2, 2)$, $(x_4, y_4) = (2, 2)$. The radius is $\gamma_j = 1.4$, for all $0 \le j \le n-1$.

UE_0 has a normal location distribution with $\mu_{x,i} = \mu_{y,i} = 0$, and $\sigma_{x,i} = \sigma_{y,i} = 1.5$. UE_1, ..., UE_6 have uniform location distributions in squares of size 6.8×6.8 (i.e., the dashed square in Fig. 8 slightly shifted). For clarity, we only show the centers of the six UEs' location distributions in Fig. 8, which are $(\mu_{x,1}, \mu_{y,1}) = (-0.5, -0.5)$, $(\mu_{x,2}, \mu_{y,2}) = (0, -0.5)$, $(\mu_{x,3}, \mu_{y,3}) = (0.5, -0.5)$, $(\mu_{x,4}, \mu_{y,4}) = (-0.5, 0.5)$, $(\mu_{x,5}, \mu_{y,5}) = (0, 0.5)$, $(\mu_{x,6}, \mu_{y,6}) = (0.5, 0.5)$.

The parameters of the power consumption models are: $\xi_j = 10$, $\alpha_j = 2$, $P_j^* = 5$, $\beta_j = 2$, for all $0 \le j \le n-1$.

In Table 6, we show the probabilities $\pi_i(j)$, for all $0 \le i \le m-1$ and $0 \le j \le n-1$. It seems that there is good chance for a UE to lose service. The probabilities that the MECs are busy are:

$$B_0 = 0.72552,$$
$$B_1 = 0.57417,$$
$$B_2 = 0.57417,$$
$$B_3 = 0.57417,$$
$$B_4 = 0.57417.$$

It seems that there is good chance for an MEC to be idle.

In Tables 7 and 8, we show MEC speed setting for both power consumption models. It is clear that for the same power constraint P, the idle-speed model yields higher computation speed than the constant-speed model due to higher power efficiency. We also observe that since MEC_0 is likely to serve more UEs (not because it is busier than other MECs), its speed is faster than other MECs.

In Fig. 9, we display the maximum stationary average response time $\tilde{T}_{\max}^{(\infty)}$ as a function of the power constraint P. It is clear that for the same power constraint P, the idle-speed model yields lower response time than the constant-speed model

Table 6
Location distributions of the UEs.

	$\pi_i(0)$	$\pi_i(1)$	$\pi_i(2)$	$\pi_i(3)$	$\pi_i(4)$	$\sum \pi_i(j)$
UE_0	0.35307	0.08394	0.08394	0.08394	0.08394	0.68883
UE_1	0.13315	0.13315	0.11705	0.11705	0.10129	0.60170
UE_2	0.13315	0.13315	0.13315	0.11705	0.11705	0.63356
UE_3	0.13315	0.11705	0.13315	0.10129	0.11705	0.60170
UE_4	0.13315	0.11705	0.10129	0.13315	0.11705	0.60170
UE_5	0.13315	0.11705	0.11705	0.13315	0.13315	0.63356
UE_6	0.13315	0.10129	0.11705	0.11705	0.13315	0.60170

Table 7
MEC speed setting (idle-speed model).

	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4
$P = 100$	0.92857	0.90098	0.90207	0.90451	0.90606
$P = 200$	1.62689	1.55995	1.55796	1.55867	1.55811
$P = 300$	2.11577	2.01355	2.00757	2.00542	2.00191
$P = 400$	2.51802	2.38212	2.37179	2.36635	2.35955
$P = 500$	2.86938	2.70070	2.68581	2.67682	2.66657
$P = 600$	3.18620	2.98533	2.96574	2.95304	2.93922
$P = 700$	3.47761	3.24494	3.22056	3.20404	3.18657

Table 8
MEC speed setting (constant-speed model).

	MEC_0	MEC_1	MEC_2	MEC_3	MEC_4
$P = 100$	0.71968	0.70128	0.70266	0.70505	0.70671
$P = 200$	1.25930	1.21472	1.21475	1.21673	1.21760
$P = 300$	1.63601	1.56848	1.56642	1.56709	1.56649
$P = 400$	1.94535	1.85614	1.85171	1.85071	1.84836
$P = 500$	2.21509	2.10495	2.09798	2.09510	2.09084
$P = 600$	2.45798	2.32738	2.31775	2.31285	2.30660
$P = 700$	2.68110	2.53038	2.51801	2.51100	2.50267

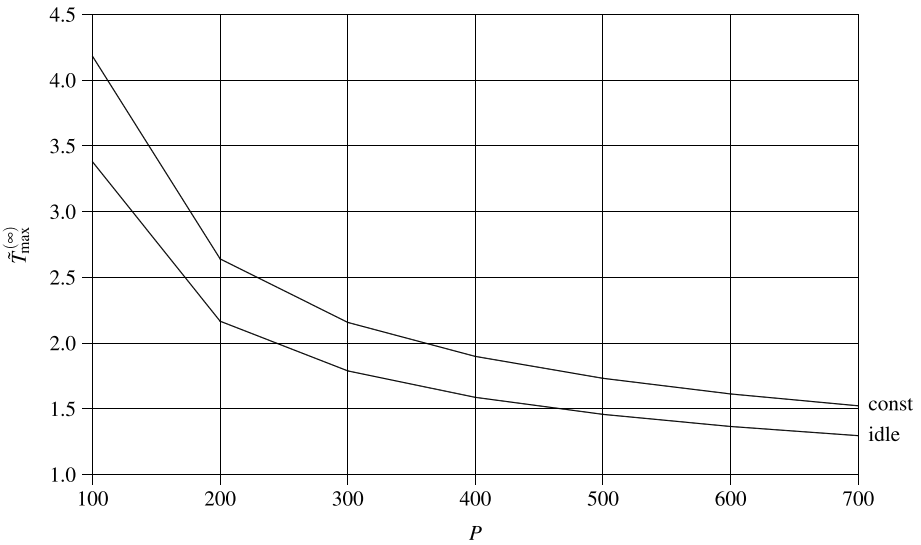

Fig. 9. $\tilde{T}_{\max}^{(\infty)}$ as a function of P.

due to higher MEC computation speed. It is easily observed that increasing P can noticeably reduce the average response time. However, to certain extent, the average response time cannot be reduced anymore, no matter how much power is provided and how fast the MECs are, since based on Theorem 1, we know that $T_i = \tilde{T}_j = T$. Thus, T cannot be lower than $T_{lb} = \max_{i \in I_j}\{\overline{x_i}\}$.

9. Extensions

In this section, we mention extensibility of our models and methods to more sophisticated mobile edge computing environment.

Multiserver MECs. When MECs are multiserver systems, the M/G/m queueing system model can be employed. An accurate analytical expression of the average response time for an M/G/m queueing system is available. Hence, Theorem 1 can be easily extended to accommodate multiserver MECs.

No MEC service. When a UE is not in the service area of any MEC, there is no MEC service for the UE. The Markov chains can be extended by adding a dummy MEC_n, with computation speed $\tilde{s}_n = 0$. When a UE is not at any MEC, it is at MEC_n and cannot offload any task to MEC_n.

Overlapping service areas. When a UE is somewhere with services available from multiple MECs, it is the UE's decision to choose one of the MECs to access mobile edge computing service. In other words, each UE actually decides the servers and defines the service areas by itself, and all such decisions and definitions are reflected in the Markov chains.

Arbitrary computation offloading. Any computation offloading scheme can be considered. For instance, the UEs at the same MEC can play a non-cooperative game to reach a Nash equilibrium. No matter which computation offloading strategy is adopted, each UE_i knows its $T_i(J)$ for each J, and this is exactly what we need for performance prediction.

Correlated mobility. UEs may have correlated mobility. For instance, two UEs may tend to stay together or to depart from each other. Such dependent mobilities are actually represented in the joint random walks $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{P}'$. Of course, these Markov chains are no longer decomposable. For the random location distribution model, a bivariate normal distribution can effectively describe covariance and correlation among the coordinates.

Temporal-dependent mobility. DTMC can handle time-dependent transition probability matrices, and CTMC can handle time-dependent transition rate matrices.

UE groups. A *UE group* (UEG) is a collection of UEs, which walk together. Assume that there are m UEGs: UEG_0, UEG_1, ..., UEG_{m-1}. The mobility of UEG_i is specified by a single Markov chain or a single random location distribution. Let I'_j denote the set of indices of UEGs at MEC_j. Then, the set of indices of UEs at MEC_j is simply

$$I_j = \bigcup_{i \in I'_j} \text{UEG}_i,$$

where UEG_i also represents a set of indices of UEs in UEG_i.

Speed setting. The MEC speed setting problem can be defined based on other criterion, e.g., speed setting with performance constraint, speed setting with optimal power-time product, speed setting with lowest cost-performance ratio, and so on.

10. Related work

In this section, we comment on some related work.

Research involving UE mobility in mobile edge computing can be classified into several categories, including mobility awareness, mobility management, mobility prediction, and mobility modeling.

Mobility awareness. Many computation offloading and task scheduling algorithms have included user mobility into consideration [35]. Ding et al. designed energy consumption minimization algorithms for UEs with known mobility, predictable mobility, and random mobility [6]. Liu et al. considered computation offloading for a mobile UE with precedence constrained tasks, where the speed of the UE is known [21]. Ma et al. focused on providing virtualized network function services to mobile users, by taking into account user mobility and service delay requirements [22]. Wang et al. investigated fast heuristic task scheduling with joint consideration of task properties, user mobility, and network constraints, where the movement paths are available in advance [32]. Zhan et al. studied offloading decision and resource allocation to achieve optimal system-wide user utility, with consideration of mobility in the process of task offloading [36].

Mobility management. The main method for mobility management is service migration. When a UE moves sufficiently away from a serving MEC, the service is migrated to another MEC [23]. Ksentini et al. employed a Markov decision process to study a service migration procedure and formulated a decision policy to a make service migration decision when a UE is far away from an MEC [12]. Ouyang et al. addressed the trade-off between service quality and migration cost by studying the service performance optimization problem under long-term cost budget constraint with unpredictable user mobility [25]. Yang et al. proposed a dynamic and live service migration technique based on an online user movement prediction method that does not rely on prior knowledge such as user trajectories [34].

Mobility prediction. Attempts have been made to adopt predicted mobility into mobile edge computing. Maleki et al. designed novel offloading approaches that consider expected future user locations predicted by a machine learning method [24]. Plachy et al. developed a path selection algorithm for best data delivery between a mobile UE and the MECs based on estimated transmission delay and energy consumption using a Markov decision process [27]. Plachy et al. proposed an algorithm for dynamic virtual machine placement and communication path selection based on predicted and expected user movement [28]. Wang et al. devised an online approximation algorithm for dynamic service placement, whose performance is compared with an offline algorithm that knows the optimal placement with look-ahead time-window [31]. Wu et al.

constructed a factor graph learning model integrating social behavior, network status, and location correlation for location prediction [33].

Mobility modeling. It is well known that trajectories of bank notes are scale-free random walks known as Lévy flights and that human traveling behavior can be described mathematically on many spatiotemporal scales by a two-parameter continuous-time random walk model to a surprising accuracy [3]. Random walks and semi-Markov processes have been successfully applied to mobility modeling and analysis for mobile terminals in cellular wireless communication networks [13]. Various mobility models have been proposed for mobile ad hoc networks (MANETs), including random waypoint model, random walk model, random direction model, and so on [1,4,29]. Unfortunately, little mobility modeling research has been conducted for mobile edge computing. The work in this paper makes significant progress towards this direction.

11. Concluding remarks

11.1. Summary

We have pointed out multiple challenges in modeling and analyzing the performance of UEs in a mobile edge computing environment with multiple heterogeneous MECs. We have proposed to combine mathematically rigorous queueing systems, algorithmically available computation offloading strategies, and analytically tractable Markov chains to simultaneously tackle dynamicity, interaction, mobility, randomness, and cost of UEs. We are able to calculate the joint probability distribution of the locations of UEs at any time. We can evaluate and predict the performance of a UE (i.e., the average response time of tasks generated on the UE) at any time, when the UE dynamically and stochastically generates tasks, competitively shares MEC resources with other UEs, and randomly walks among the MECs. We have also considered closed-form performance predictions for homogeneous UEs and MECs and examined the impact of mobility cost on performance. Furthermore, we addressed power constrained MEC speed setting.

11.2. A note on applications

We believe that the CTMC model is the most applicable to a wide range of scenarios and situations. The DTMC model is limited due to synchronized movement of the UEs. Our SMP model is limited due to i.i.d. random holding times at different MECs.

Our research in this paper is applicable in sophisticated real-world systems in the sense that we have considered task heterogeneity (tasks on different UEs have different arrival rates, computation requirements, and communication requirements), UE heterogeneity (different UEs have different computation speeds, communication speeds, computation offloading strategies, and mobilities), and MEC heterogeneity (different MECs have different computation speeds).

We have made efforts to accommodate UE mobilities which are non-Markovian processes, e.g., semi-Markov processes.

11.3. Future research

We would like to mention several directions for further investigation.

- Smart UEs – When UE_i detects that the current MEC_j is overcrowded with many UEs, i.e., $|I_j|$ is large, UE_i changes its location more quickly. This means that UEs are no longer independent in their movement; instead, the movement of a UE depends on the movement of other UEs. Fortunately, such smart UEs can be modeled using our Markov chains with proper modification. For DTMC, the transition probability from J to J' is not simply a product of UEs individual transition probabilities. For CTMC, the transition rate from J to J' should be adapted and adjusted. For SMP, the transition probabilities of the embedded DTMC should be remedied and revised. The main challenge is how to deal with the large matrix size and to reduce the computational cost.
- Mobility cost – It seems quite challenging to extend our discussion and result in Section 7 to more sophisticated context, including the SMP mobility model and non-exponential randomized mobility cost, transition time, service delay, and location change penalty (i.e., Δ is an arbitrary random variable).
- General SMP – We can extend our SMP in Section 3.3.3 to more general SMP, where each UE can have different random holding times with different probability distributions at different MECs. This is certainly a sophisticated and difficult mathematical problem.

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.

Data availability

Data will be made available on request.

Acknowledgments

The author would like to thank the anonymous referees for their critical review and useful suggestions on improving the paper.

Appendix A. Proofs

Proof of Theorem 1. Since

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

we have

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

that is,

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

and

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

which implies that

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

for all $i \in I_j$.

Furthermore, since

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

we need to find T such that

$$f(T) = 0,$$

where

$$f(T) = T - \frac{1}{\sum_{i\in I_j}\left(\lambda_i - \frac{2(T-\overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T-\overline{x_i})}\right)}\sum_{i\in I_j}\left(\lambda_i - \frac{2(T-\overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T-\overline{x_i})}\right)\left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right)$$
$$+ \frac{\sum_{i\in I_j}\left(\lambda_i - \frac{2(T-\overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T-\overline{x_i})}\right)\left(\frac{\overline{r_i^2}}{\tilde{s}_j^2} + 2\frac{\overline{r_i}\,\overline{d_i}}{\tilde{s}_j c_{i,j}} + \frac{\overline{d_i^2}}{c_{i,j}^2}\right)}{2\left(1-\sum_{i\in I_j}\left(\lambda_i - \frac{2(T-\overline{x_i})}{\overline{x_i^2} + 2\overline{x_i}(T-\overline{x_i})}\right)\left(\frac{\overline{r_i}}{\tilde{s}_j} + \frac{\overline{d_i}}{c_{i,j}}\right)\right)}.$$

This proves the theorem. □

Proof of Theorem 10. A perfectly balanced initial location distribution is

$$\pi_i^{(0)} = \left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right).$$

Essentially, we need to show that for all DTMC/CTMC/SMP, we always have

$$\pi_i^{(t)} = \pi_i^{(0)},$$

for all $t \geq 0$.

For DTMC, we have

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

where $p_i(j, j') = 1/n$, for all $0 \leq j, j' \leq n-1$. For a perfectly balanced initial location distribution:

$$\pi_i^{(0)} = \left(\frac{1}{n}, \frac{1}{n}, ..., \frac{1}{n}\right),$$

we get

$$\pi_i^{(0)} = \pi_i^{(0)} \boldsymbol{P}_i,$$

which implies that

$$\pi_i^{(t)} = \pi_i^{(0)} \boldsymbol{P}_i^t = \pi_i^{(0)},$$

for all $t \geq 0$.

For CTMC, we have

$$\boldsymbol{Q}_i = q_i(\boldsymbol{P}_i' - \boldsymbol{I}),$$

with

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

where $p_i'(j, j') = 1/(n-1)$, for all $0 \leq j \neq j' \leq n-1$. Again, since

$$\pi_i^{(0)} = \pi_i^{(0)} \boldsymbol{P}_i'$$

and

$$\pi_i^{(k)} = \pi_i^{(0)} (\boldsymbol{P}_i')^k = \pi_i^{(0)},$$

for all $k \geq 0$, a perfectly balanced location distribution is maintained after each location change and at any time. Mathematically, we have

$$\begin{aligned}
\exp(\boldsymbol{Q}_i t) &= \exp((\boldsymbol{P}_i' - \boldsymbol{I}) q_i t) \\
&= \exp(\boldsymbol{P}_i' q t) \exp(-\boldsymbol{I} q_i t) \\
&= \exp(\boldsymbol{P}_i' q t) \boldsymbol{I} e^{-q_i t} \\
&= e^{-q_i t} \exp(\boldsymbol{P}_i' q t),
\end{aligned}$$

where we notice that

$$\exp(\boldsymbol{M}_1 + \boldsymbol{M}_2) = \exp(\boldsymbol{M}_1) \exp(\boldsymbol{M}_2),$$

if $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ commute. Furthermore, for all $t \geq 0$,

$$\begin{aligned}
\pi_i^{(t)} &= \pi_i^{(0)} \exp(\boldsymbol{Q}_i t) \\
&= e^{-q_i t} \pi_i^{(0)} \exp(\boldsymbol{P}_i' q t) \\
&= e^{-q_i t} \pi_i^{(0)} \sum_{k=0}^{\infty} \frac{(\boldsymbol{P}_i' q t)^k}{k!} \\
&= e^{-q_i t} \sum_{k=0}^{\infty} \frac{\pi_i^{(0)} (\boldsymbol{P}_i')^k (q t)^k}{k!} \\
&= e^{-q_i t} \sum_{k=0}^{\infty} \frac{\pi_i^{(0)} (q t)^k}{k!}
\end{aligned}$$

$$= e^{-q_i t}\pi_i^{(0)} \sum_{k=0}^{\infty} \frac{(qt)^k}{k!}$$
$$= e^{-q_i t}\pi_i^{(0)} e^{q_i t}$$
$$= \pi_i^{(0)}.$$

For SMP, we have the same $\boldsymbol{P}_i'$ as that of CTMC. Once again, a perfectly balanced location distribution is maintained after each location change and at any time. Mathematically, we notice that for all $t \geq 0$,

$$\pi_i^{(t)} = \pi_i^{(0)} \left(\sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k] (\boldsymbol{P}_i')^k \right)$$
$$= \sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k] \pi_i^{(0)} (\boldsymbol{P}_i')^k$$
$$= \sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k] \pi_i^{(0)}$$
$$= \pi_i^{(0)} \sum_{k=0}^{\infty} \mathbf{P}[X_i(t) = k]$$
$$= \pi_i^{(0)}.$$

The theorem is proved. □

References

[1] F. Bai, A. Helmy, A survey of mobility models in wireless ad hoc networks, in: Wireless Ad Hoc and Sensor Networks, Kluwer Academic Publishers, 2004, pp. 1–29, Chapter 1.
[2] A. Blum, J. Hopcroft, R. Kannan, Foundations of Data Science, Cambridge University Press, United Kingdom, 2020.
[3] D. Brockmann, L. Hufnagel, T. Geisel, The scaling laws of human travel, Nature 439 (26 January 2006) 462–465.
[4] T. Camp, J. Boleng, V. Davies, A survey of mobility models for ad hoc network research, Wirel. Commun. Mob. Comput. 2 (5) (2002) 483–502.
[5] Y. Ding, K. Li, C. Liu, K. Li, A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing, IEEE Trans. Parallel Distrib. Syst. 33 (6) (2022) 1503–1519.
[6] Y. Ding, K. Li, C. Liu, Z. Tang, K. Li, Short- and long-term cost and performance optimization for mobile user equipments, J. Parallel Distrib. Comput. 150 (2021) 69–84.
[7] Y. Ding, K. Li, C. Liu, Z. Tang, K. Li, Budget-constrained service allocation optimization for mobile edge computing, IEEE Trans. Serv. Comput. (2022), in press.
[8] 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.
[9] Z. He, K. Li, K. Li, W. Zhou, J. Liu, Server configuration optimization in mobile edge computing: a cost-performance tradeoff perspective, Softw. Pract. Exp. 51 (9) (2021) 1847–1981.
[10] J. Hu, K. Li, C. Liu, A.T. Chronopoulos, K. Li, Game-based task offloading of multi-MD with QoS in MEC systems of limited computation capacity, ACM Trans. Embed. Comput. Syst. 19 (4) (2020) 29.
[11] L. Kleinrock, Queueing Systems, Volume 1: Theory, John Wiley and Sons, New York, 1975.
[12] A. Ksentini, T. Taleb, M. Chen, A Markov decision process-based service migration procedure for follow me cloud, in: IEEE International Conference on Communications, Sydney, NSW, Australia, 10-14 June 2014.
[13] K. Li, Analysis of distance-based location management in wireless communication networks, IEEE Trans. Parallel Distrib. Syst. 24 (2) (2013) 225–238.
[14] K. Li, A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing, IEEE Trans. Sustain. Comp. (September 2018), https://doi.org/10.1109/TSUSC.2018.2868655.
[15] K. Li, Computation offloading strategy optimization with multiple heterogeneous servers in mobile edge computing, IEEE Trans. Sustain. Comp. (March 2019), https://doi.org/10.1109/TSUSC.2019.2904680.
[16] K. Li, How to stabilize a competitive mobile edge computing environment: a game theoretic approach, IEEE Access 7 (1) (2019) 69960–69985.
[17] K. Li, Heuristic computation offloading algorithms for mobile users in fog computing, ACM Trans. Embed. Comput. Syst. 20 (2) (2021) 11, 28 pp.
[18] K. Li, Distributed and individualized computation offloading optimization in a fog computing environment, J. Parallel Distrib. Comput. 159 (2022) 24–34.
[19] K. Li, Non-clairvoyant and randomized online task offloading in mobile edge computing, Int. J. Parallel Emerg. Distrib. Syst. 37 (4) (2022) 413–424.
[20] C. Liu, K. Li, J. Liang, K. Li, COOPER-MATCH: job offloading with a cooperative game for guaranteeing strict deadlines in MEC, IEEE Trans. Mob. Comput. (June 2019), https://doi.org/10.1109/TMC.2019.2921713.
[21] Y. Liu, C. Liu, J. Liu, Y. Hu, K. Li, K. Li, Mobility-aware and code-oriented partitioning computation offloading in multi-access edge computing, J. Grid Comput. 20 (2) (2022) 11.
[22] Y.Ma.W. Liang, J. Li, X. Jia, S. Guo, Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks, IEEE Trans. Mob. Comput. 21 (1) (2022) 196–210.
[23] M. Mehrabi, H. Salah, F.H.P. Fitzek, A survey on mobility management for MEC-enabled systems, in: IEEE 2nd 5G World Forum, Dresden, Germany, 30 Sept.–2 Oct. 2019.
[24] E.F. Maleki, L. Mashayekhy, S.M. Nabavinejad, Mobility-aware computation offloading in edge computing using machine learning, IEEE Trans. Mob. Comput. (1 June 2021), https://doi.org/10.1109/TMC.2021.3085527.

[25] T. Ouyang, Z. Zhou, X. Chen, Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing, IEEE J. Sel. Areas Commun. 36 (10) (2018) 2333–2345.
[26] H. Pishro-Nik, Introduction to Probability, Statistics, and Random Processes, Kappa Research LLC, 2014, Available at https://www.probabilitycourse.com.
[27] J. Plachy, Z. Becvar, P. Mach, Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network, Comput. Netw. 108 (2016) 357–370.
[28] J. Plachy, Z. Becvar, E.C. Strinati, Dynamic resource allocation exploiting mobility prediction in mobile edge computing, in: IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications, Valencia, Spain, 4-8 Sept. 2016.
[29] A.G. Ribeiro, R. Sofia, A survey on mobility models for wireless networks, SITI Technical Report SITI-TR-11-01, Universidade Lusófona, February 2011.
[30] S.M. Ross, Introduction to Probability Models, 12th ed., Elsevier Inc., 2019.
[31] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, K.K. Leung, Dynamic service placement for mobile micro-clouds with predicted future costs, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1002–1016.
[32] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, R. Wang, User mobility aware task assignment for mobile edge computing, Future Gener. Comput. Syst. 85 (2018) 1–8.
[33] Q. Wu, X. Chen, Z. Zhou, L. Chen, Mobile social data learning for user-centric location prediction with application in mobile edge service migration, IEEE Int. Things J. 6 (5) (2019) 7737–7747.
[34] R. Yang, H. He, W. Zhang, Multitier service migration framework based on mobility prediction in mobile edge computing, Wirel. Commun. Mob. Comput. 2021 (2021) 6638730.
[35] S.K.U. Zaman, A.I. Jehangiri, T. Maqsood, Z. Ahmad, A.I. Umar, J. Shuja, E. Alanazi, W. Alasmary, Mobility-aware computational offloading in mobile edge networks: a survey, Clust. Comput. 24 (2021) 2735–2756.
[36] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, H. Duan, Mobility-aware multi-user offloading optimization for mobile edge computing, IEEE Trans. Veh. Technol. 69 (3) (2020) 3341–3356.