Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.066975

ARTICLE

An Online Optimization of Prediction-Enhanced Digital Twin Migration over Edge Computing with Adaptive Information Updating

Xinyu Yu[1], Lucheng Chen[2,3], Xingzhi Feng[2,4], Xiaoping Lu[2,4,*], Yuye Yang[1] and You Shi[5,*]

[1]School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China
[2]State Key Laboratory of Massive Personalized Customization System and Technology, Qingdao, 266100, China
[3]COSMOPlat Institute of Industrial Intelligence (Qingdao) Co., Ltd., Qingdao, 266100, China
[4]COSMOPlat IoT Technology Co., Ltd., Qingdao, 266103, China
[5]College of Electrical Engineering and Control Science, Nanjing Tech University, Nanjing, 211816, China

*Corresponding Authors: Xiaoping Lu. Email: xiaoping.luhaier@outlook.com; You Shi. Email: shiyou@njtech.edu.cn

Received: 22 April 2025; Accepted: 11 July 2025; Published: 23 September 2025

ABSTRACT: This paper investigates mobility-aware online optimization for digital twin (DT)-assisted task execution in edge computing environments. In such systems, DTs, hosted on edge servers (ESs), require proactive migration to maintain proximity to their mobile physical twin (PT) counterparts. To minimize task response latency under a stringent energy consumption constraint, we jointly optimize three key components: the status data uploading frequency from the PT, the DT migration decisions, and the allocation of computational and communication resources. To address the asynchronous nature of these decisions, we propose a novel two-timescale mobility-aware online optimization (TMO) framework. The TMO scheme leverages an extended two-timescale Lyapunov optimization framework to decompose the long-term problem into sequential subproblems. At the larger timescale, a multi-armed bandit (MAB) algorithm is employed to dynamically learn the optimal status data uploading frequency. Within each shorter timescale, we first employ a gated recurrent unit (GRU)-based predictor to forecast the PT's trajectory. Based on this prediction, an alternate minimization (AM) algorithm is then utilized to solve for the DT migration and resource allocation variables. Theoretical analysis confirms that the proposed TMO scheme is asymptotically optimal. Furthermore, simulation results demonstrate its significant performance gains over existing benchmark methods.

KEYWORDS: Digital twin; edge computing; proactive migration; mobility prediction; two-timescale online optimization

1 Introduction

The digital twin (DT) is a promising paradigm for creating a virtual representation of a physical entity. By enabling simulation and behavior analysis in the digital space, a DT functions as a virtual sandbox to facilitate fine-grained monitoring and intelligent decision-making for complex tasks originating from its physical counterpart [1]. For instance, in the domain of autonomous driving, DTs support realistic testing by virtualizing vehicle behavior and complex road environments, thereby enabling safe and efficient validation of driving tasks via edge communication technologies [2]. In other applications, DTs and multi-agent systems are leveraged to model and coordinate warehouse robots for cargo handling tasks in a smart cyber-physical environment [3]. Similarly, Lv et al. [4] studied a DT-assisted framework for medical delivery tasks using an unmanned aerial vehicle (UAV). Consequently, owing to its capability to address complex real-world

challenges, the DT is recognized as a key enabling technology for applications such as Industry 4.0, the metaverse, and smart cities, garnering significant research attention [5].

A DT-enabled task processing framework typically involves numerous DTs and their associated PTs. In this context, PTs represent real-world entities such as humans, vehicles, or devices, with DTs functioning as their virtual counterparts [6]. Ensuring energy-efficient, low-latency task execution for a PT necessitates the meticulous construction and management of its corresponding DT. These requirements motivate the use of edge computing [7], where DTs are deployed on ESs at the network periphery to execute tasks for their PTs. While recent studies have explored related topics, such as industrial DTs and service deployment on ESs, implementing mobile DT-assisted task execution presents unique challenges. First, unlike stationary entities in industrial plants, the high mobility of PTs results in unstable PT-DT connectivity. Second, in contrast to a limited number of shared service applications, a large number of exclusive DTs must concurrently execute complex tasks for their associated PTs. This intensive computation relies on the limited resources of the host ESs [8], leading to potentially high task response latency and energy consumption. Therefore, to maintain seamless PT-DT connectivity while ensuring low-latency, energy-efficient task execution, the dynamic migration of DTs among ESs becomes essential [9].

However, addressing the aforementioned issues is challenging due to several underlying reasons:

1) Since the transmission of PT status information for mobility prediction is energy-intensive, these updates should occur at a lower frequency. In contrast, the constant movement and task generation of PTs demand that DT migration and resource allocation be optimized more frequently. This suggests that the overall optimization must be handled asynchronously across different timescales.
2) The frequency of uploading PT status data governs a trade-off between task response latency and energy consumption. Shorter upload intervals can reduce task latency but concurrently increase energy usage, whereas longer intervals have the opposite effect. This implies that the upload frequency must be determined adaptively based on feedback regarding system performance.
3) The high mobility of PTs, coupled with the need for frequent DT migrations, means that conventional reactive strategies can lead to severe task response latency and even service interruptions. Therefore, it is necessary to proactively migrate DTs by employing a predictive mobility model [10].

To tackle these challenges, we introduce a novel two-timescale online optimization scheme for a DT-assisted task execution system in an edge computing environment. Specifically, we consider a scenario where each PT has an exclusive DT deployed on an ES to execute its complex tasks. To account for PT mobility, the corresponding DTs are proactively migrated among ESs. These migration decisions are based on the PTs' future trajectories, which are forecast by a predictive model hosted in a cloud center. To support this model, PTs must upload their status information (e.g., moving speed, direction) as predictive inputs. Our primary objective is to minimize the long-term average task response latency of all DTs, subject to a stringent energy consumption constraint and inherent system uncertainties such as unpredictable PT mobility. To this end, we formulate the problem as a two-timescale online optimization problem. This problem involves dynamically optimizing two sets of decisions. The first set, decided at the larger timescale, is the status information uploading frequency for each PT. The second set, decided at the smaller timescale, includes the DT migration strategy and the allocation of computational and communication resources at the ES. To solve this problem, we develop a novel two-timescale mobility-aware online optimization approach based on extended Lyapunov optimization theory. This approach first decomposes the long-term problem into a sequence of subproblems corresponding to the different timescales. An online learning method is then incorporated to adaptively determine the status information update frequency. Subsequently, for each subproblem within the smaller timescale, we first employ a GRU-based scheme to predict the PT's trajectory. Based on this prediction, an AM-based algorithm is proposed to optimize the small-timescale decisions.

The key contributions of this paper are summarized as follows:

- We address the problem of proactive DT migration at the network edge for executing complex PT tasks. To this end, we formulate a two-timescale mobility-aware online optimization problem that jointly optimizes three key decisions, including the adaptive status update frequency of PTs, the proactive DT migration decision, and the allocation of computational and communication resources on the ESs.
- We propose a novel solution framework, termed TMO, which first decomposes the long-term problem into a sequence of subproblems. To solve these subproblems across their different timescales, the TMO framework integrates two main components. An online learning method is employed to address the large-timescale problem, while an AM algorithm, augmented by a GRU-based prediction scheme, is utilized for the small-timescale problems.
- We conduct both theoretical analysis and extensive simulations to validate the performance of our proposed approach. The results demonstrate that the TMO scheme significantly outperforms existing benchmark methods in reducing both task response latency and overall system energy consumption.

The following section provides a review of related work. The system model and the corresponding problem formulation are detailed in Section 3. Section 4 presents our proposed two-timescale mobility-aware online optimization approach and its theoretical analysis. Section 5 discusses the simulation results, and Section 6 concludes the paper.

2 Related Work

As an accurate digital representation of physical entities, the deployment of DTs enables real-time interaction and shows significant potential for supporting emerging 6G applications. For example, Wang et al. [11] propose a DT-based method for attack detection in the internet of things by fusing spatio-temporal features to enhance identification accuracy within dynamic network environments. Similarly, Zhao et al. [12] designed a DT-based application system to improve the accuracy and efficiency of network management. Jyeniskhan et al. [13] proposed a framework for DT systems in additive manufacturing that incorporates machine learning techniques. However, these studies did not fully consider the impact of a dynamic network environment or varying user status on the quality of the PT-DT interaction.

Edge computing is a crucial technology for delivering latency-sensitive services in wireless networks by offloading computation and storage tasks to the network edge. Recently, research efforts have increasingly focused on DT management within edge environments. For instance, Wen et al. [14] proposed an improved artificial potential field method for edge computing environments to achieve cooperative control and DT monitoring of multi-AUG. Lu et al. [15] proposed a DT-assisted wireless edge network framework designed to facilitate low-latency computation and seamless connectivity. In another study, Zhang et al. [16] proposed a framework for DT systems within wireless-powered communication networks, examining adaptive placement and transfer schemes for the DTs. However, these studies typically assume that DTs are either stationary or designed for a single, specific purpose. This assumption makes their approaches inadequate for supporting mobile PTs that have diverse service demands. Beyond DT-specific management, another stream of relevant research focuses on general service migration at the edge. For instance, He et al. [17] presented an efficient planning and scheduling framework for edge service migration that handles live migrations while ensuring low latency for mobile users. Mustafa et al. [18] proposed a hybrid SQ-DDTO algorithm within a three-layer vehicular edge computing framework to optimize task offloading and resource allocation. The same authors [19] also proposed a PPO-based task offloading algorithm for vehicular edge computing. Their approach addresses task dependencies and dynamic network conditions, aiming to enhance policy stability while minimizing delays and the task drop ratio. Nevertheless, these general migration studies assume a

limited set of service entity (SE) types, where a single SE can serve multiple users. This contrasts with the inherent exclusivity of a DT, which is dedicated to handling tasks for only its specific PT.

Lyapunov-based optimization is a widely utilized tool for online optimization problems characterized by system uncertainties, as it can guarantee long-term performance stability without requiring future information [20–22]. However, most existing works in this area focus on decisions made within a single timescale. More recently, some studies have employed a two-timescale Lyapunov method [7,23], where the original problem is decomposed and decisions are subsequently made across different timescales. In a different approach, Huang et al. [24] combined the multi-armed bandit (MAB) method with Lyapunov optimization to schedule queueing systems without prior knowledge of instantaneous network conditions. Nevertheless, these schemes cannot be directly applied to our problem. This is primarily due to two factors. First, our model incorporates a flexible duration for the large timescale. Second, it features a complex coupling of decision variables within both the objective function and the problem constraints.

In contrast to previous works, this paper proposes a novel mobility-aware, two-timescale online optimization approach for edge computing environments. This approach is designed to jointly optimize three coupled decisions, i.e., the uploading frequency of PT status information, the DT migration decision, and the allocation of computational and communication resources.

3 System Model and Problem Formulation

To begin with, an overview on the DT-assisted task execution system is provided. After that, the DT proactive migration together with PT task execution are described more specifically.

3.1 System Overview

As presented in Fig. 1, we present a DT-assisted task execution system built on edge computing. The system comprises a set of I PTs, denoted as $\mathscr{I}$, and M geographically distributed ESs, represented by $\mathscr{M}$. A cloud center functions as the central controller and employs a GRU-based model to enable proactive migration of DTs. PTs generate a stream of complex tasks and require exclusive DTs deployed at the edge for task execution. Due to the complex mobility patterns of PTs, the associated DT should be proactively migrated from one ES to another, based on the mobility prediction model constructed on the cloud center, and each PT should switch its access to the ES that its associated DT newly migrated to. It is worth noting that each ES can support multiple DTs for different PTs, with communication and computation resources shared among all these DTs. Once the DT migration is completed, the PTs can offload their tasks to the associated DTs for execution.

In practice, due to the frequent mobility and task dynamics of PTs, DT migration and the corresponding resource allocation need to be performed more often than the energy-intensive status information uploading. Therefore, as shown in Fig. 2, in the proposed online optimization problem, we consider that the status update frequency of each PT is determined on a large timescale, while DT migration and the associated communication and computation resource allocation on each ES are handled on a small timescale. To be specific, we first segment the timeline into $T \in \mathbb{N}^+$ fine-grained time slots and denote $t \in \mathscr{T}$ as the index of each time slot. The interval between two consecutive status information updates (also defined as large-grained time frame) may contain a varying number of time-slots. Let $\mathscr{K}$ be the total number of time frames, and the first time slot of time frame k is denoted as $t^{(k)}$.

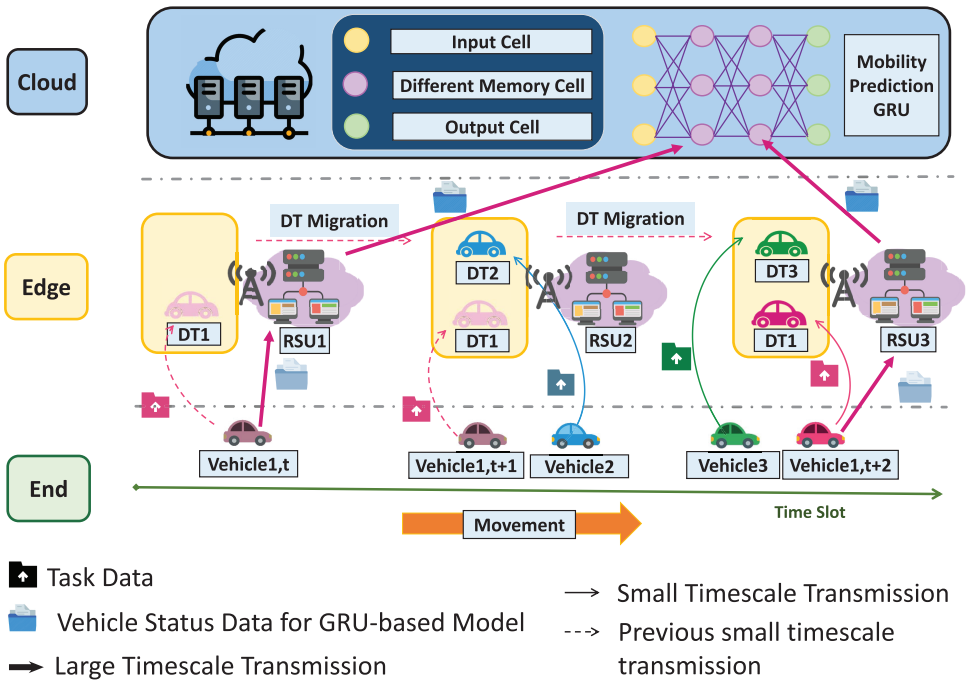

Figure 1: An illustration of DT-assisted task execution system

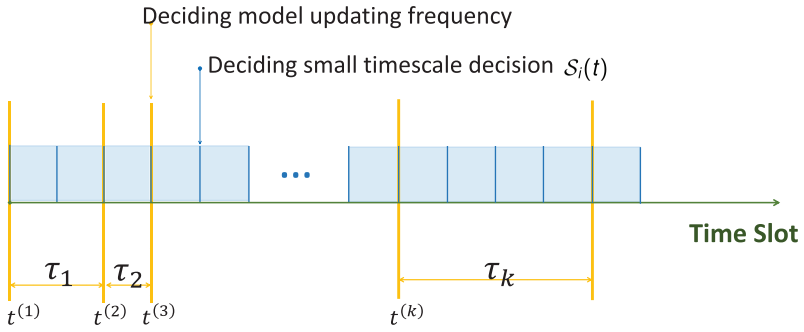

Figure 2: Design of the online two-timescale approach

In general, we aim to optimize the task response latency under the stringent energy consumption constraint by addressing: i) How to adjust the real-time PT status updating of the mobility prediction model. ii) Within every time slot, which ES should be the migration target of each DT and how to allocate the communication and computation resource among the deployed DTs. The detailed system model is described as follows, and Table 1 lists the important notations employed in this paper.

Table 1: Notations of system model

Notations	Meaning
$\mathscr{I}$	Set of ESs
$\mathscr{M}$	Set of mobile users
τ_k	Length of the k-th time frame

(Continued)

Table 1 (continued)

Notations	Meaning
$a_{i,m}(t)$	DT i's migration decision in time slot t
D_i	Model size of DT i
$J_i(t)$	Mobility state information data size of PT i in time slot t
$E_{i,m}^{mig}(t)$	Energy consumption of migrating DT i to ES m in time slot t
$E_{i,c}^{upl}(t)$	Energy consumption of PT i uploading mobility state information to the cloud center in time slot t
$E_{i,m}^{tra}(t)$	Energy consumption of PT i transmitting task data to ES m in time slot t
$E_{i,m}^{exe}(t)$	Energy consumption of DT i's task execution in time slot t
$r_{i,c}(t)$	Transmission rate from PT i to cloud center in time slot t
$r_{i,m}(t)$	Transmission rate from PT i to ES m in time slot t
$\lambda_i(t)$	Data size of the task generated by PT i in time slot t
$u_i(t)$	Communication resource allocation for PT i in time slot t
$f_i(t)$	Computation resource allocation for PT i in time slot t
F_m	Total computation resource of ES m for all DTs
$T_{i,m}^{tra}(t)$	Latency of PT i transmitting task data to ES m in time slot t
$T_{i,m}^{exe}(t)$	Latency of DT i's task execution in time slot t
Vbv	Lyapunov tuning parameter

3.2 DT Proactive Migration

To achieve seamless DT-assisted task execution for PTs, each DT should be timely migrated to the new ES in each time slot $t \in \mathscr{T}$. Let $a_{i,m}(t) \in \{0,1\}$ be the migration decision of DT i in time slot $t \in \mathscr{T}$ indicating whether the corresponding DT of PT $i \in \mathscr{I}$ is deployed on ES $m \in \mathscr{M}$. Note that each PT $i \in \mathscr{I}$ has one unique DT which can only be deployed on one ES $m \in \mathscr{M}$ simultaneously, and ESs are connected through wired links [15]. The energy consumption associated with wired transmission latency between edge servers (ESs) is strongly influenced by the physical distance between them, which can be defined as

$$E_{i,m}^{mig}(t) = \phi D_i d(\pi_i(t-1), m), \tag{1}$$

where ϕ indicates the per-unit energy cost for data transmission per unit distance, D_i is the model size of DT $i \in \mathscr{I}$, and $\pi_i(t) = \arg\max_m a_{i,m}(t)$ is the ES which PT $i \in \mathscr{I}$ is accessed to in time slot $t \in \mathscr{T}$.

To achieve proactive DT migration, it is required to obtain the movement trajectories of mobile PTs in the future. Therefore, we construct a GRU-based model at the cloud center for mobility prediction, which is described in detail in Section 4. To make predictions of PTs' mobile trajectories, the GRU-based model needs to receive status information transmitted from PTs to ES and subsequently uploaded to the cloud. We denote $(x_i(t), y_i(t))$ as the location of PT $i \in \mathscr{I}$ in time slot t, (x_m, y_m) as the fixed location of ES $m \in \mathscr{M}$. In accordance with Shannon's theorem, the data uploading rate between PT $i \in \mathscr{I}$ and ES $m \in \mathscr{M}$ is given by

$$r_{i,m}(t) = u_i(t) W_m \log\left(1 + \frac{(d_{i,m}(t))^{-\theta} p_i |h_{i,m}(t)|^2}{N_0 u_i(t) W_m}\right), \tag{2}$$

in which $u_i(t)$ is the allocation of bandwidth resource for PT $i \in \mathscr{I}$ in time slot $t \in T$, $d_{i,m}(t) = \sqrt{(x_i(t) - x_m)^2 + (y_i(t) - y_m)^2}$ is the distance from PT $i \in \mathscr{I}$ to ES $m \in \mathscr{M}$, $h_{i,m}(t)$ captures the Rayleigh fading effect between PT $i \in \mathscr{I}$ and ES $m \in \mathscr{M}$ in time slot $t \in T$. Note that the bandwidth allocation $u_i(t)$ represents the portion of spectrum resource assigned to each PT, which reflects the multi-user spectrum contention at the ES. Let $J_i(t)$ be the mobility state information data size of PT $i \in \mathscr{I}$ in time slot $t \in \mathscr{T}$, the latency and energy consumption of uploading the status information from PT $i \in \mathscr{I}$ to ES $m \in \mathscr{M}$ in time slot $t \in \mathscr{T}$ can thus be denoted as $T^{upl}_{i,m}(t) = \frac{J_i(t)}{r_{i,m}(t)}$ and $E^{upl}_{i,m}(t) = \frac{J_i(t)}{r_{i,m}(t)} p_i$, where p_i is the transmission power of PT $i \in \mathscr{I}$. Correspondingly, the energy consumption of uploading the status information from ES $m \in \mathscr{M}$ to the cloud center in time slot $t \in \mathscr{T}$ can be calculated as $E^{upl}_{m,c}(t) = \frac{J_i(t)}{r^m} p^m$, where p^m and r^m respectively represents for unit transmission power and the uplink transmission rate via a fiber link channel from ES m to the cloud center. Noting that fiber link is a wired connection with stable communication environment and abundant bandwidth resource, r^m and p^m are therefore considered as constants [25]. Therefore, the total energy consumption of status information uploading by PT $i \in \mathscr{I}$ can be derived as

$$E^{upl}_{i,c}(t^{(k)}) = \sum_{m \in \mathscr{M}} (E^{upl}_{i,m}(t^{(k)}) + E^{upl}_{m,c}(t^{(k)})). \tag{3}$$

3.3 DT-Assisted Task Execution

During each time slot $t \in T$, each PT can offload tasks to the ES where its corresponding DT is proactively migrated to. The latency and energy consumption for uploading a task from PT $i \in \mathscr{I}$ to its DT in each time slot $t \in T$ are respectively expressed as

$$T^{tra}_{i,m}(t) = \frac{\lambda_i(t)}{r_{i,m}(t)}, \tag{4}$$

$$E^{tra}_{i,m}(t) = T^{tra}_{i,m}(t) p_i. \tag{5}$$

Let $\lambda_i(t)$ denote the amount of task data produced by PT $i \in \mathscr{I}$ during time slot $t \in \mathscr{T}$. The latency experienced by PT i when executing its task with the assistance of its DT on ES $m \in \mathscr{M}$ at time t is given by

$$T^{exe}_{i,m}(t) = \frac{\lambda_i(t) C_m}{f_i(t) F_m}, \tag{6}$$

where C_m denotes CPU cycles' number of ES $m \in \mathscr{M}$ to execute one data unit, F_m represents the CPU processing speed (in cycles per second) of ES $m \in \mathscr{M}$, and $f_i(t)$ is the allocation of computation resource for PT $i \in \mathscr{I}$ in time slot $t \in T$. In addition, the energy consumption is calculated as

$$E^{exe}_{i,m}(t) = \rho_m (F_m)^2 \lambda_i(t) C_m, \tag{7}$$

where ρ_m represents the capacitance of ES $m \in \mathscr{M}$.

3.4 Problem Formulation

The small-timescale and large-timescale energy consumption is derived as

$$E^{sts}(t) = \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} a_{i,m}(t) (E^{exe}_{i,m}(t) + E^{mig}_{i,m}(t) + E^{tra}_{i,m}(t)), \tag{8}$$

$$E^{lts}(t^{(k)}) = \sum_{i \in \mathscr{I}} E^{upl}_{i,c}(t^{(k)}). \tag{9}$$

To assess the fundamental performance of the DT-assisted task execution system deployed over edge computing, we adopt the long-term average service response latency across all time slots as the evaluation metric, which is computed as

$$T^{resp}(t) = \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t)(T^{tra}_{i,m}(t) + T^{exe}_{i,m}(t) + T^{upl}_{i,m}(t^{(k)})). \tag{10}$$

We construct a two-timescale online optimization problem that jointly determines i) the adaptive status information updating frequency τ_k in time frame k, ii) DT migration decision $a_{i,m}(t)$, iii) ES communication resource allocation decision $u_i(t)$, and iv) ES computation resource allocation decision $f_i(t)$ in time slot t. The problem is expressed as

$$[\mathscr{P}_1]: \min_{\tau_k, \mathscr{S}_i(t)} \sum\nolimits_{t\in\mathscr{T}} T^{resp}(t)$$

$$s.t., \ \sum\nolimits_{m\in\mathscr{M}} a_{i,m}(t) = 1, \forall i \in \mathscr{I}, \forall t \in \mathscr{T}, \tag{11a}$$

$$\sum\nolimits_{i\in\mathscr{I}} u_i(t) a_{i,m}(t) \le 1, \forall m \in \mathscr{M}, \forall t \in \mathscr{T}, \tag{11b}$$

$$\sum\nolimits_{i\in\mathscr{I} f_i(t)} a_{i,m}(t) \le 1, \forall m \in \mathscr{M}, \forall t \in \mathscr{T}, \tag{11c}$$

$$\tau_k \in [1, \tau_{max}], \forall k, \tag{11d}$$

$$\sum\nolimits_{t\in\mathscr{T}} E^{sts}(t) + \sum\nolimits_{k\in\mathscr{K}} E^{lts}(t^{(k)}) \le C, \tag{11e}$$

where $\mathscr{S}_i(t) = \{a_{i,m}(t), u_i(t), f_i(t)\}$ is the small-timescale decisions determined in each time slot; constraint (11a) ensures that each PT is associated with only one ES per time slot; constraints (11b) and (11c) guarantees that the total allocated computation and communication resource do not exceed the capacity of any ES in any time slot t; constraint (11d) restricts the maximum time slot number of any time frame; (11e) is the long-term system overall energy consumption constraint.

Obviously, solving $\mathscr{P}_1$ directly is very challenging because: i) The decisions involve continuous and discrete variables, and are all integrated in the constraints and objective function, meaning that $\mathscr{P}_1$ is a non-convex optimization problem involving mixed-integer variables, which is NP-hard. ii) The constant mobility of PTs and the changing network dynamics make it extremely hard to acquire the system statics in advance. This implies that the problem must be addressed in an online manner. iii) τ_k is not a simple dicision, because its determination should be learned from previous response latency and system energy feedback, necessitating a learning algorithm compatible with the online optimization approach.

4 A Two-Timescale Online Optimization Algorithm

In this section, we proposed a novel two-timescale mobility-aware optimization approach, namely TMO, to jointly optimize the uploading frequency of PT status information, DT migration decision and corresponding resource allocation on ESs over edge computing. Specifically, in particular, we begin by breaking down the long-term optimization task into multiple short-term subproblems using an enhanced Lyapunov-based strategy. Next, we apply an online learning algorithm to adaptively adjust the status information updating frequency based on historical feedback in large-timescal. After that, we design a GRU-based prediction network to predict the location of PT and develop an AM-based method to solve for the remaining small-timescale decisions.

4.1 Problem Decomposition

We begin by defining the energy deficit queue $Q(t)$, which reflects the deviation of total system energy consumption from the long-term average budget C/T. The evolution of this queue over time can be

represented as follows:

$$Q(t+1) = max\left[Q(t) + E^{sts}(t) + E^{lts}(t)|_{t=t^{(k)}} - \frac{C}{T}, 0\right]. \tag{12}$$

Traditional Lyapunov optimization requires fixed constraints to ensure a bounded performance gap. However, since the PT status information updating only occurs when each time frame starts, e.g., at $t \in t^{(k)}$, the total energy consumption will have an abrupt change. To guarantee the stability of $Q(t)$, we distribute the status information uploading energy consumption over a large time frame evenly across all the time slots within it. Thus, for any time slot t in τ_k, the evolution of $Q(t)$ can be modified as

$$Q(t+1) = max\left[Q(t) + E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k} - \frac{C}{T}, 0\right]. \tag{13}$$

Subsequently, the Lyapunov function is defined as $L(\Theta(t)) = \frac{1}{2}Q(t)^2$, which is regarded as the quantitative indicator of queue congestion. To maintain queue stability, this function should be driven toward its minimum value. As discussed in [26], let $\mathbb{E}[\cdot]$ be the expectation operator, and the conditional Lyapunov drift is given by $\Delta(\mathbf{\Theta}(t)) = \mathbb{E}[L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)]$. The term $\Delta(\mathbf{\Theta}(\tau))$ represents the variation in Lyapunov function across successive time slots. Minimizing this drift effectively curbs queue growth, ensuring compliance with the energy consumption constraint.

Correspondingly, we defined $\Delta(\mathbf{\Theta}(t)) + V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)]$ as the Lyapunov drift-plus-penalty expression, in which $V > 0$ is a tuning parameter that can be tuned to make a trade-off between optimality and queue stability. A larger V indicates a greater emphasis on meeting the energy consumption constraint, whereas a smaller V suggests a willingness to relax this constraint to some extent to reduce the task response latency.

Theorem 1. *Let $V > 0$, and at any given time slot $t \in \tau_k$, the drift-plus-penalty is bounded under all feasible decisions, i.e.,*

$$\begin{aligned}&\Delta(\mathbf{\Theta}(t)) + V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)] \le G + \mathbb{E}[Q(t)(E^{sts}(t)\\ &+ \frac{E^{lts}(t^{(k)})}{\tau_k} - \frac{C}{T})|\mathbf{\Theta}(t)] + V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)],\end{aligned} \tag{14}$$

where $G = \frac{1}{2}[E^{sts}(max) + \frac{E^{lts}(max)}{\tau_k} - \frac{C}{T}]^2$.

Proof. Please see Appendix A. □

Theorem 1 demonstrates the upper bound of the drift-plus-penalty in each time slot $t \in \tau_k$. Then, by aggregating both sides of (14) over all time slots within time frame k, we obtain

$$\begin{aligned}&\sum_{t\in\tau_k}\Delta(\mathbf{\Theta}(t)) + V\sum_{t\in\tau_k}\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)]\\ &\le G\tau_k + \sum_{t\in\tau_k}\mathbb{E}\left[Q(t)\left(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k} - \frac{C}{T}\right)|\mathbf{\Theta}(t)\right] + V\sum_{t\in\tau_k}\mathbb{E}\left[T^{resp}(t)|\mathbf{\Theta}(t)\right].\end{aligned} \tag{15}$$

Therefore, $\mathscr{P}_1$ can be decomposed into a series of subproblems, where the right side of (15) is opportunistically minimized in each time frame $k \in \mathscr{K}$ as

$$[\mathscr{P}_2]: \min_{\mathscr{S}_i(t)} V \sum_{t\in\tau_k} \mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)] + \sum_{t\in\tau_k} \mathbb{E}\left[Q(t)(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k}|\mathbf{\Theta}(t)\right].$$
$$s.t. (11a), (11b), (11c). \tag{16}$$

Note that although problem $\mathscr{P}_2$ concentrates on the optimization within one time frame, it still contains variables in two timescales, making it challenging to solve directly. However, it can be observed that decision $S_i(t)$ can be solved on top of the determination of adaptive status information uploading frequency. Thus, we divide the problem $\mathscr{P}_2$ into two subproblems (i.e., large-timescale and small-timescale subproblems), and solve them sequentially.

4.2 Solution for Large-Timescale Decision

Due to the fact that the future network dynamics including channel conditions, DT migration together with resource allocation decisions in each time slot have not been revealed when we need to determine τ_k when each large time frame starts, we develop an online learning approach to adaptively decide τ_k based on the revealed response latency and system energy from the previous time frames. Specifically, we first construct a multi-armed bandit (MAB) problem to decide the τ_k at the beginning of each time frame k by defining a selectable arm set $\{1, 2, \cdots, \tau_{max}\}$ for each arm τ_k and set the objective function of $[\mathscr{P}_2]$ as the loss of pulling each arm, i.e.,

$$[\mathscr{P}_3]: \min_{\tau_k\in\{1,2,\cdots,\tau_{\max}\}} V \sum_{t\in\tau_k} T^{resp}(t) + \sum_{t\in\tau_k} Q(t)(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k}. \tag{17}$$

Next, to solve the MAB problem $[\mathscr{P}_3]$, we apply the upper confidence bound (UCB) method to decide the optimal τ_k based on historical energy consumption and task response latency. To be specific, the loss of choosing τ_k is given by

$$l(\tau) = V \sum_{t\in\tau} T^{resp}(t) + \sum_{t\in\tau} Q(t)\left(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau}\right). \tag{18}$$

Then, let $\bar{l}(\tau)$ be the average revealed loss of previous time frames, and for each feasible arm in set $\{1, 2, \cdots, \tau_{max}\}$, we can calculate the UCB of each arm as

$$-\bar{l}(\tau) + \sqrt{2logK/T_\tau}. \tag{19}$$

Considering both the average reward and an exploration term that grow with time, the optimal τ of each largr time frame is calculated as

$$\tau = \text{argmax}\{-\bar{l}(\tau) + \sqrt{2logK/T_\tau}\}, \tau \in [1, \tau_{max}]. \tag{20}$$

4.3 Solution for Small-Timescale Decisions

4.3.1 GRU-Based Mobility Prediction Network

After the status information uploading frequency is determined, in each time slot, to optimize small-timescale decisions, the location of mobile PTs to support proactive DT migration is required. Different from traditional time series prediction with input available in each time step, since the PTs only upload their

status information at the beginning of each time frame, we need to predict their locations in each time slot regardless of whether PTs' status information is uploaded. To this end, we extend the conventional GRU model into an input-autoregressive PT mobility prediction model, and its state update equations are designed as:

$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}),\\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}),\\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t(W_{hn}h_{t-1} + b_{hn})),\\
h_t &= (1 - z_t)n_t + z_t h_{t-1},\\
y_t &= W_o h_t + b_o,\\
x_{t+1} &= y_t.
\end{aligned}
$$

In this model, W and b represent the trainable parameters. In time slot t, z_t, r_t, n_t are respectively the update, reset and new gates. h_t is the hidden state at time slot t, while x_t and y_t correspond to the input and output variables. The function $\sigma(\cdot)$ refers tp the sigmoid activation. The state update process is shown in Fig. 3.

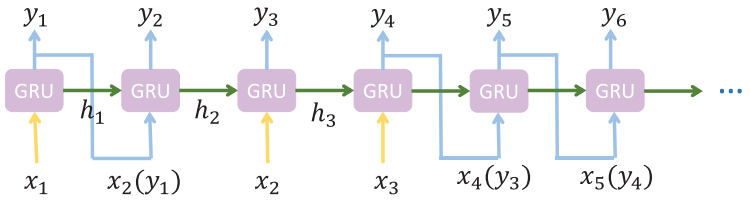

Figure 3: Workflow of the auto-regressive GRU-based model

To train the above-designed GRU-based model for PT mobility prediction, we increase the auto-regressive inference capability during the training phase to address the issue of unavailable ground truth data at each time step. Specifically, we replace the real input data with the model's previous output based on a probability Pr, which starts from 0 and gradually increases with the number of epochs. Finally, Pr reaches its maximum value of 0.8 after 500 epochs.

At the beginning of each epoch, batches of trajectory samples are fetched from the dataset. For each time step, a random number $r \in [0, 1]$ is drawn. if $r \geq Pr$, the real input is used; otherwise, the model's last output serves as input to the next step. This process is repeated over all sequences and batches in each epoch to enhance the model's robustness and generalization to various trajectory patterns. The optimization of model parameters is performed using the Adam optimizer with a learning rate of 0.05 and mean squared error (MSE) as the loss function. The GRU network consists of two hidden layers with 30 units each and a dropout rate of 0.2. Training is conducted for 500 epochs with a batch size of 256. To prevent overfitting and improve convergence, a learning rate scheduler reduces the learning rate by half if the validation loss does not improve for three consecutive epochs. The detailed pseudocode is provided in Algorithm 1.

Algorithm 1: The training procedure of GRU-based mobility prediction network

Input: $\boldsymbol{X}^{state} = (x_{t1}^{state}, x_{t2}^{state}, ..., x_{tn}^{state})$
Output: Weights of the prediction network θ

Collect $\boldsymbol{X}^{state}$ from the cloud center database;
$epoch \leftarrow 0$;
repeat
  $p_r \leftarrow 0.8 \times \frac{epoch}{500}$;
  $epoch \leftarrow epoch + 1$;
  $batch \leftarrow 0$;
  repeat
    $batch \leftarrow batch + 1$;
    for each timestep t in the sequence **do**
      Draw a random number from $[0, 1]$;
      if $r \geq p_r$ **then**
        Extract x_t^{state} from $\boldsymbol{X}$ as the input;
      else
        Use the previous output y_t^{state} as the input;
      end if
    end for
    perform a forward pass through GRU cells and get the prediction results;
  until $batch \geq \lceil |Epoch|/256 \rceil$
until $epoch \geq 500$

4.3.2 Algorithm for Small-Timescale Decisions

Given τ_k and the predicted positions of PTs in each time slot of the kth time frame, we further determine the small-timescale decisions $\mathscr{S}_i(t) = \{a_{i,m}(t), u_i(t), f_i(t)\}$ by minimizing the term $V\mathbb{E}[T^{resp}(t)|\Theta(t)] + \mathbb{E}[Q(t)E^{sts}(t)|\Theta(t)]$ in (14), which is formulated as

$$[\mathscr{P}_4]: \min_{\mathscr{S}_i(t)} VT^{resp}(t) + Q(t)(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k}) \quad s.t. (11a), (11b), (11c). \tag{21}$$

The subproblem $\mathscr{P}_4$ is a a nonlinear mixed-integer programming problem with coupled decision variables. To solve the problem, we deploy an AM-based method, which divides $\mathscr{P}_4$ into two subproblems and alternately optimize one set of variables with the others fixed until the objective of $\mathscr{P}_4$ converges.

DT Migration Decision: Fix $u_i(t)$ and $f_i(t)$ and solve for $a_{i,m}(t)$:

$$[\mathscr{P}_{4-1}]: \min_{a_{i,m}(t)} VT^{resp}(t) + Q(t)(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k}) \quad s.t. (11a), (11b), (11c).$$

Communication and Computation Resource Allocation: Fix $a_{i,m}(t)$ and solve for $u_i(t)$ and $f_i(t)$:

$$[\mathscr{P}_{4-2}]: \min_{u_i(t), f_i(t)} VT^{resp}(t) + Q(t)(E^{sts}(t) + \frac{E^{lts}(t^{(k)})}{\tau_k})$$

$s.t.$(11b), (11c).

Following the decomposition of $\mathscr{P}_3$, the resulting subproblem $\mathscr{P}_{4\text{-}1}$ becomes a linear integer program that can be efficiently handled using standard optimization solvers. For subproblem $\mathscr{P}_{4-2}$, since $u_i(t)$ and $f_i(t)$ are not coupled in both object function and constraint, it can be further divided into two independent subproblems, where $u_i(t)$ and $f_i(t)$ are computed separately:

$$[\mathscr{P}_{4-2-1}]: \min_{u_i(t)} V \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t) T^{tra}_{i,m} + Q \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t)(E^{tra}_{i,m}(t) + \frac{E^{upl}_{i,m}(t^{(k)})}{\tau_k})$$
$$s.t.(11b).$$
$$[\mathscr{P}_{4-2-2}]: \min_{f_i(t)} V \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t) T^{exe}_{i,m}$$
$$s.t.(11c).$$

Theorem 2. *Problem $\mathscr{P}_{4-2-1}$ and $\mathscr{P}_{4-2-2}$ are both convex.*

Proof. Details can be found in Appendix B. □

Building on Theorem 2, we can slove problems $\mathscr{P}_{4-2-1}$ and $\mathscr{P}_{4-2-2}$ efficiently by utilizing standard optimization solvers such as Gurobi. It is important to note that the small-timescale problem is addressed through an iterative process, which concludes when further improvements to the objective of problem $\mathscr{P}_{4-2}$ are no longer possible.

4.4 Analyses of Proposed TMO Scheme

In our proposed two-timescale mobility-aware optimization framework (TMO), we begin by utilizing an extended Lyapunov technique to divide the problem in the long term into a sequence of short-term subproblems. We then incorporate an online learning algorithm to dynamically adjust the update frequency for status information in the large timescale, guided by historical feedback. Subsequently, a GRU-based prediction model is employed to predict the location of each PT, and an alternating minimization (AM) strategy is used to determine the remaining decisions in the small timescale. The overall structure of TMO is illustrated in Fig. 4.

Theorem 3. *Our proposed TMO approach can achieve convergence after finite iterations.*

Proof. Since the alternation process is only applied in solving small-timescale problem $[\mathscr{P}_4]$, we prove the convergence of the AM based method by deriving the partial derivatives of the objective function of subproblem $\mathscr{P}_4$ as

$$\nabla_{\boldsymbol{a}}(\mathscr{P}_{4-1}) = V \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} (T^{tra}_{i,m}(t) + T^{exe}_{i,m}(t) + T^{upl}_{i,m}(t^{(k)})) \tag{22}$$
$$+ Q(t) \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} (E^{tra}_{i,m}(t) + E^{exe}_{i,m}(t) + E^{mig}_{i,m}(t) + (E^{upl}_{i,m}(t^{(k)}) + E^{upl}_{m,c}(t^{(k)}))),$$
$$\nabla_{\boldsymbol{u}}(\mathscr{P}_{4-2-1}) = -V \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t) \frac{\lambda_i}{u_i^2(t) r_{i,m}(t)} - Q \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t) p_i \frac{\lambda_i}{u_i^2(t) r_{i,m}(t)}, \tag{23}$$
$$\nabla_{\boldsymbol{f}}(\mathscr{P}_{4-2-2}) = -V \sum_{i\in\mathscr{I}} \sum_{m\in\mathscr{M}} a_{i,m}(t) \frac{\lambda_i(t) C_m}{f_i^2(t) F_m}. \tag{24}$$

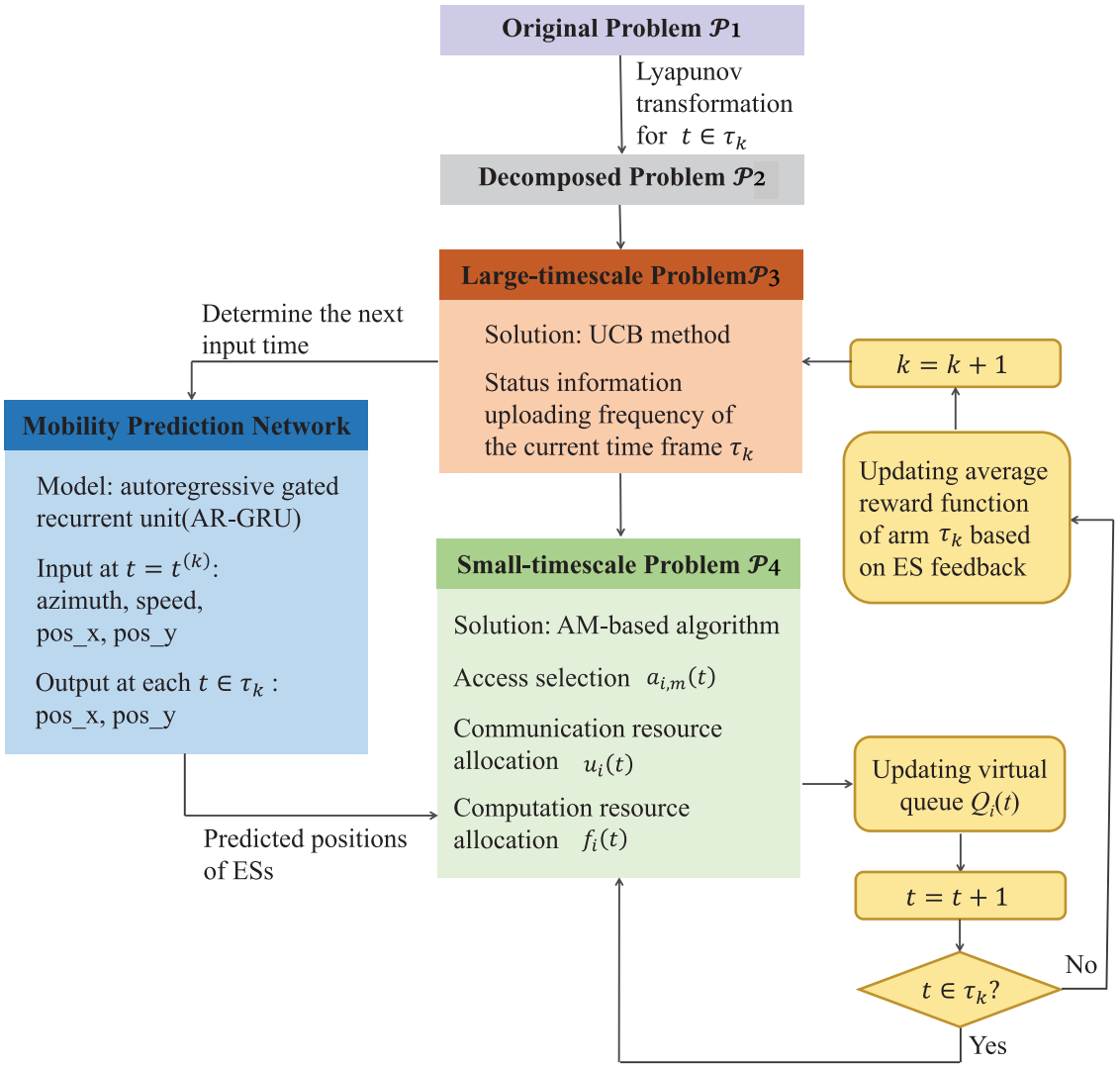

Figure 4: Flowchart of the proposed TMO approach

It is evident that the gradient in (22) is constant, while (23) and (24) exhibit bounded first-order derivatives under the constraints of the original problem. This implies that all relevant components are L-Lipschitz continuous. Accordingly, the proposed method converges within finite iterations. □

Theorem 4. *The proposed TMO scheme's computational complexity is expressed as* $O(S^{max}K + TN^{max}((k^{I+M}+2)I^3))$. *I denotes the PT number, M the ES number, K the arms' number in the UCB algorithm,* N^{max} *the maximum iteration count in the AM-based algorithm,T the number of time slots, and* S^{max} *the maximum number of time frames.*

Proof. The computationa complexity of the TMO approach mainly determined by the contributions of both UCB method in large timescale and the AM-based algorithm in small timescale. For a standard UCB method, the computational complexity of selecting a arm is $O(1)$, and of updating the expected reward is $O(K)$. Besides, the computational complexity for solving small-timescale decisions can be calculated as $O((k^{I+M}+2)I^3)$, by leveraging the interior point scheme [27] in Gurobi solver, where k reflects the efficiency of the branch-and-bound method's pruning. □

In summary, the TMO algorithm has a computational complexity of $O(S^{max}K + TN^{max}((k^{I+M} + 2)I^3))$.

Theorem 5. *Given Lyapunov control parameter V and UCB arms number K, the performance gap between the TMO approach and the theoretical optimum to problem $\mathscr{P}_1$ is*

$$\frac{1}{T}\sum_{t\in\mathscr{T}}\mathbb{E}[\sum_{i\in\mathscr{I}}\sum_{m\in\mathscr{M}}a_{i,m}(t)(T^{tra}_{i,m}(t)+T^{exe}_{i,m}(t))]-\mathscr{X}\leq\frac{G}{V}+\frac{\Lambda}{VT} \tag{25}$$

where $\mathscr{X}$ represents the optimal solution in theoretical, Λ is the performance gap of UCB algorithm, which can be expressed as $O(\sqrt{K\log K})$.

Proof. The upper-bound of the performance gap (e.g., regret) between our TMO approach and the optimum can be decomposed into two gaps, respectively. The first gap is bounded by UCB algorithm under feedbacks of choosing τ_k as the arm. The second gap is bounded by the extended Lyapunov optimization theory with periodically changed constraints due to modified τ_k in each time frame.

To analyze the gap bounded by Lyapunov optimization theory, inequality (14) can be intuitively expanded and rewritten as

$$\begin{aligned}
&\triangle(\Theta(t))+V\mathbb{E}[T^{resp}(t)|\Theta(t)]\\
&\leq G+\mathbb{E}\left[Q(t)\left(E^{sts}(t)+\frac{E^{lts}(t^{(k)})}{\tau_k}-\frac{C}{T}\right)|\Theta(t)\right]+V\mathbb{E}[T^{resp}(t)|\Theta(t)]\\
&\leq G+V\cdot\mathscr{X}+\Lambda
\end{aligned} \tag{26}$$

Then, by aggregating (26) over T time slots, we obtain

$$\begin{aligned}
&(G+V\cdot\mathscr{X}+\Lambda)\cdot T\\
&\geq\sum_{t\in\mathscr{T}}(\Delta(\mathbf{\Theta}(t))+V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)])\\
&=\mathbb{E}[L(\mathbf{\Theta}(T))-L(\mathbf{\Theta}(0))]+V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)]\\
&=\mathbb{E}[L(\mathbf{\Theta}(T))]-\mathbb{E}[L(\mathbf{\Theta}(0))]+V\mathbb{E}[T^{resp}(t)|\mathbf{\Theta}(t)]
\end{aligned} \tag{27}$$

By rearranging inequality (27), i.e., subtracting $\mathbb{E}[L(\Theta(0))]$ from both sides and then dividing by T, we derive inequality (25).

We then analyze the optimal gap Λ brought by UCB algorithm. According to [27], it is equal to $4\sqrt{2K\tau_{max}\log K}+\sum_{k=1}^{\tau_{max}}(\triangle_k+\frac{\pi^2\triangle_k}{3})$, where $\Delta_k\overset{\text{def}}{=}\mu^*-\mu_k$. Note that μ_k is the reward expectation for arm k and μ^* is any maximal element in the set $\{\mu_1,\ldots,\mu_K\}$, thus $\triangle_k$ is constant and it can be defined as $O(\sqrt{K\log K})$. □

5 Simulation Results

In this section, we conducted extensive simulations to verify the advantages of our proposed TMO scheme for optimizing the PT state information uploading frequency and the DT migration decision together with allocation of computation and communication resource over edge computing. All simulations are conducted based on the real-world traffic scenario dataset, and the results are averaged more than 1000 independent runs with varying parameter setting.

5.1 Simulation Settings

Consider a DT-assisted task execution system in a 15 km × 18 km region with $I = 40$ PTs and $M = 10$ ESs. The moving trajectories and state information used for mobility prediction are extracted from the "Real-World Bologna" dataset [28]. Which is derived from a real scenario in the city of Bologna and provided in the form of a SUMO simulation package. This dataset involves more than 8000 mobile entities operating in a realistic urban environment. In our experiments, a simulation of over 5000 s was conducted, during which the output file was generated at 1-second intervals to record the spatio-temporal states (object type, position coordinates, travel angle, speed, road gradient, street identifier, and relative distance from the entrance of the current street) of all entities. Table 2 lists the key parameters and their values, most of which are also are consistent with those commonly adopted in prior works [7,29]. Additionally, to assess the performance of the proposed TMO approach, we benchmark it against the following schemes. It is worth noting that, due to the original objectives of these benchmarks differ from ours, to ensure a fair comparison, we have adjusted these benchmark schemes. Specifically, we modified their settings and optimization objectives to align with our proposed framework.

- SO [30]: Single-timescale optimization. This approach optimizes service migration and task routing to balance the online system performance and service migration cost, yet executed synchronously in a single timescale
- FTO [31]: Fixed two-timescale optimization. This method optimizes service migration, as well as computing and communication resource allocation asynchronously in different timescales. However, the length of each time frame is fixed.

Table 2: Simulation parameters

Parameter	Value	Parameter	Value
p_i	500 mW	W_m	5 MHz
W^c	20 MHz	F_m	20 GHz
θ	2	C_m	300 cycles/bit
N_0	−174 dBm/Hz	ρ_m	10^{-27}
$D_i(t)$	[8, 10] Mbits	λ_i	[10, 20] Mbits
τ_{max}	7	$J_i(t)$	[25, 35] Mbits

5.2 Performance Evaluation

Fig. 5 illustrates the stability of our TMO approach by showcasing the energy consumption queue backlog $Q(t)$, which is under the influence of the Lyapunov decomposition for different values of V. The figure reveals that the queue backlog decreases and stabilizes quickly over time. This is because TMO prioritizes minimizing total energy consumption to optimize the objective function of problem $\mathscr{P}_3$, which effectively reduces the queue backlog and achieves a balance between task response latency and energy consumption. Additionally, it is shown in the figure that the average energy consumption increases with V. The reason is that the introduction of the Lyapunov parameter V, which controls the trade-off between minimizing DT task response latency and enforcing energy consumption constraints; a larger V places greater emphasis on reducing response latency.

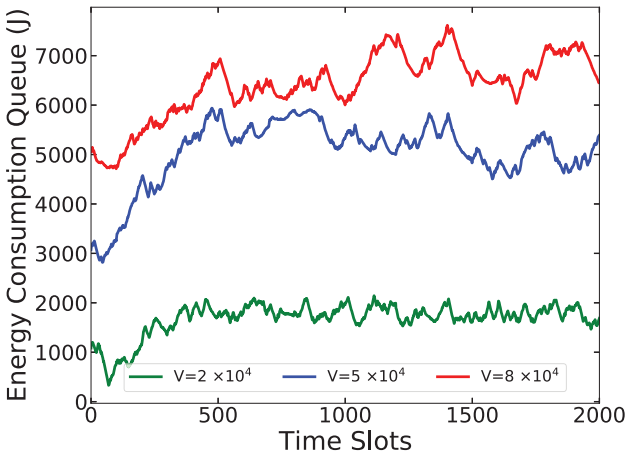

Figure 5: Queue backlogs w.r.t. V

Fig. 6 shows the impact of V's value on average task reponse latency (i.e., $\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} a_{i,m}(t)(T^{tra}_{i,m}(t)+T^{exe}_{i,m}(t))+T^{upl}_{i,m}(t^{(k)})/TI)$ and average energy consumption (i.e., $\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}[E^{upl}_{i,c}$ $\sum_{m\in\mathcal{M}} a_{i,m}(t)(E^{exe}_{i,m}(t)+E^{mig}_{i,m}(t)+E^{tra}_{i,m}(t))]/TI)$. It can be seen that with V increasing from from 10^2 to 10^6, the average energy consumption rises while the task response latency shrinks. The reason is that TMO lays more emphasis on the task response latency than the energy consumption with a larger V.

Fig. 7 illustrates the exploration process of τ_k of our proposed TMO approach. Initially, τ_k exhibits significant fluctuations, reflecting the exploration phase where the algorithm actively tests various values of τ_k to gather sufficient information about their performance. This exploration is guided by the Upper Confidence Bound (UCB) principle, which balances the trade-off between exploring less-tested values and exploiting values with lower empirical losses. As the algorithm progresses, the confidence intervals around the estimated losses for different τ_k values shrink, leading to a gradual stabilization of τ_k. This stabilization indicates that the algorithm has identified the value of τ_k that minimizes the long-term average loss, transitioning from exploration to exploitation.

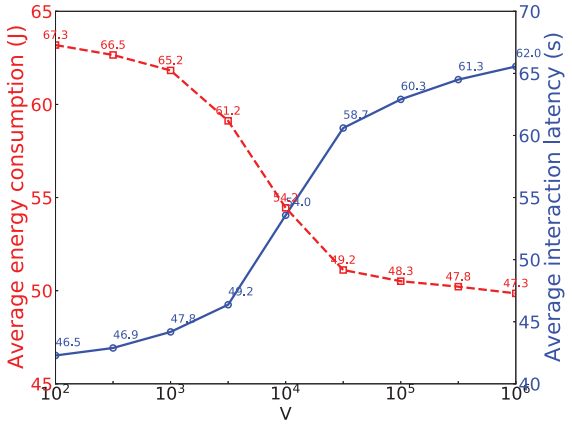

Figure 6: The impact of V on average task response latency and energy consumption

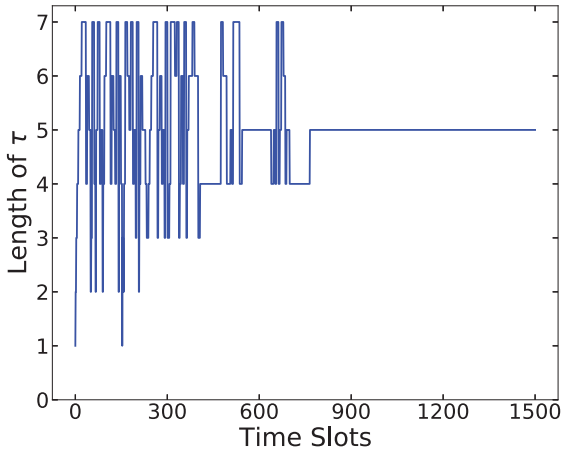

Figure 7: Decisions of τ_k in TMO

Fig. 8 examines the impact of PT numbers on the average response latency and energy consumption for SO, FTO, and the proposed TMO approach. Both figures demonstrate an exponential increase in response latency and energy consumption for all approaches as I grows, because larger I means more task requests and competition for edge resources. We can tell from the pictures that SO has the worst performance, while FTO and TMO have better performance of both task response latency and average energy consumption than SO because of the less frequent data uploading. Besides, TMO has the best performance on account of more flexible and adjustable data uploading frequency, which can reduce the cost of data transmission mostly.

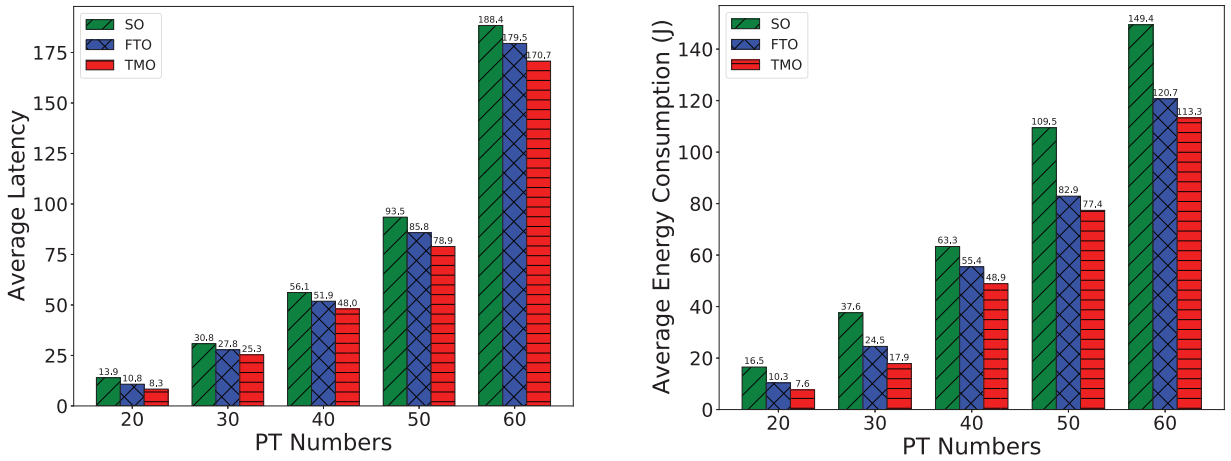

Figure 8: Comparison on average task response latency and energy consumption with varying PT numbers

6 Conclusion

In our work, we explore proactive DT migration over edge computing. Specifically, aiming to minimize the average task response latency of PTs under system uncertainties (e.g., PT mobility), we construct a two-timescale online optimization problem to jointly optimize the PT status information updating frequency, DT migration, and the allocation of communication and computation resources. We introduce a novel solution approach, termed TMO, which first break down the long-term online optimization problem into a sequence of instant subproblems. Additionally, we develop an online learning approach and an AM-based algorithm

supported by a GRU-based mobility prediction model, to solve the two subproblems at different timescales. Both theoretical analysis and extensive simulations demonstrate that TMO outperforms existing approaches in minimizing task response latency and reducing total system energy consumption.

Acknowledgement: Not applicable.

Funding Statement: This research was funded by the State Key Laboratory of Massive Personalized Customization System and Technology, grant No. H&C-MPC-2023-04-01.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Lucheng Chen and Xingzhi Feng; methodology, Xinyu Yu and You Shi; validation, Xinyu Yu, Xiaoping Lu and Yuye Yang; formal analysis, Xinyu Yu and Lucheng Chen; investigation, Lucheng Chen and Xiaoping Lu; writing—original draft preparation, Xinyu Yu; writing—review and editing, Xinyu Yu and Yuye Yang; supervision, Xiaoping Lu and You Shi; project administration, Xingzhi Feng and You Shi. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

Nomenclature

TMO	Two-timescale mobility-aware optimization algorithm
UCB	Upper Confidence Bound
MAB	Multi-Armed Bandit
GRU	Gated Recurrent Unit
AM	Alternate Minimization

Appendix A

Squaring both expressions in Eq. (13) describing the energy queue yields

$$Q^2(t+1) = \left[Q(t) + E^{sts}(t) + E^{lts}(t) - \frac{C}{T}\right]^{+2}$$
$$\leq Q^2(t) + \left(E^{sts}(t) + E^{lts}(t) - \frac{C}{T}\right)^2 + 2Q(t)\left(E^{sts}(t) + E^{lts}(t) - \frac{C}{T}\right).$$

By subtracting $Q^2(t)$ from both sides, dividing by 2, we have

$$\frac{1}{2}(Q^2(t+1) - Q^2(t))$$
$$\leq \frac{1}{2}\left(E^{sts}(t) + E^{lts}(t) - \frac{C}{T}\right)^2 + Q(t)\left(E^{sts}(t) + E^{lts}(t) - \frac{C}{T}\right). \tag{A1}$$

Lastly, by adding $V \cdot T^{resp}(t)$ to both sides of (A1) and taking the expectation of both sides of $\Theta(t)$, inequality (14) can be derived.

Appendix B

For subproblem $\mathscr{P}_{4-2-1}$, we denote $\boldsymbol{\gamma} = \{\gamma(t), \forall i \in \mathscr{I}, \forall t \in \mathscr{T}\}$ as the corresponding Lagrangian multiplier. The lagrangian function can be calculated as

$$\mathscr{L}_1(\boldsymbol{u}, \boldsymbol{\gamma}) = V \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} a_{i,m}(t) T_{i,m}^{tra} + Q \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} a_{i,m}(t) \left(E_{i,m}^{tra}(t) + \frac{E_{i,m}^{upl}(t^{(k)})}{\tau_k} \right) + \sum_{i \in \mathscr{I}} \gamma_i(t)(a_{i,m}(t) u_i(t) - 1). \tag{A2}$$

Since $\frac{1}{r_{i,m}(t)}$ in (A2) decreases monotonically, it is convex when $u_i(t) \in (0, 1]$, thus subproblem $\mathscr{P}_{4-2-1}$ is convex.

For subproblem $\mathscr{P}_{4-2-2}$, we denote $\boldsymbol{\eta} = \{\eta(t), \forall i \in \mathscr{I}, \forall t \in \mathscr{T}\}$ as the corresponding Lagrangian multiplier. The Lagrangian function can be calculated as

$$\mathscr{L}_2(\boldsymbol{f}, \boldsymbol{\eta}) = V \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} a_{i,m}(t) T_{i,m}^{exe} + \sum_{i \in \mathscr{I}} \eta_i(t)(a_{i,m}(t) f_i(t) - 1). \tag{A3}$$

Taking the first-order and second-order derivatives yields

$$\frac{\partial \mathscr{L}_2}{\partial f_i(t)} = -V \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} \frac{a_{i,m}(t) \lambda_i(t) C_m}{f_i^2(t) F_m} + \sum_{i \in \mathscr{I}} \eta_i(t) a_{i,m}(t), \tag{A4}$$

$$\frac{\partial^2 \mathscr{L}_2}{\partial (f_i(t))^2} = 2 \sum_{i \in \mathscr{I}} \sum_{m \in \mathscr{M}} \frac{a_{i,m}(t) \lambda_i(t) C_m}{f_i^3(t) F_m}. \tag{A5}$$

Obviously, when $f_i(t) \in (0, 1]$, $\frac{\partial^2 \mathscr{L}_2}{\partial (f_i(t))^2} \geq 0$, thus subproblem $\mathscr{P}_{4-2-2}$ is also convex.

References

1. Lin X, Kundu L, Dick C, Obiodu E, Mostak T, Flaxman M. 6G digital twin networks: from theory to practice. IEEE Commun Mag. 2023;61(11):72–8. doi:10.1109/mcom.001.2200830.
2. Niaz A, Shoukat MU, Jia Y, Khan S, Niaz F, Raza MU. Autonomous driving test method based on digital twin: a survey. In: 2021 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube); 2021 Oct 26–27; Quetta, Pakistan: IEEE; 2021. p. 1–7.
3. Marah H, Challenger M. Madtwin: a framework for multi-agent digital twin development: smart warehouse case study. Ann Math Artif Intell. 2024;92(4):975–1005. doi:10.1007/s10472-023-09872-z.
4. Lv Z, Chen D, Feng H, Zhu H, Lv H. Digital twins in unmanned aerial vehicles for rapid medical resource delivery in epidemics. IEEE Trans Intell Transp Syst. 2022;23(12):25106–14. doi:10.1109/tits.2021.3113787.
5. Khan LU, Saad W, Niyato D, Han Z, Hong CS. Digital-twin-enabled 6G: vision, architectural trends, and future directions. IEEE Commun Mag. 2022 Jan;60(1):74–80. doi:10.1109/mcom.001.21143.
6. Chen R, Yi C, Zhou F, Kang J, Wu Y, Niyato D. Federated digital twin construction via distributed sensing: a game-theoretic online optimization with overlapping coalitions. arXiv:2503.16823. 2025.
7. Yang Y, Shi Y, Yi C, Cai J, Kang J, Niyato D, et al. Dynamic human digital twin deployment at the edge for task execution: a two-timescale accuracy-aware online optimization. IEEE Trans Mob Comput. 2024;23(12):12262–79. doi:10.1109/tmc.2024.3406607.
8. Chen J, Yi C, Okegbile SD, Cai J, Shen X. Networking architecture and key supporting technologies for human digital twin in personalized healthcare: a comprehensive survey. IEEE Commun Sur Tutor. 2024;26(1):706–46. doi:10.1109/comst.2023.3308717.

9. Okegbile SD, Cai J, Wu J, Chen J, Yi C. A prediction-enhanced physical-to-virtual twin connectivity framework for human digital twin. IEEE Trans Cogn Commun Netw. 2024;PP(99):1–1. doi:10.1109/tccn.2024.3519331.
10. Wang C, Peng J, Cai L, Peng H, Liu W, Gu X, et al. AI-enabled spatial-temporal mobility awareness service migration for connected vehicles. IEEE Trans Mob Comput. 2024;23(4):3274–90. doi:10.1109/tmc.2023.3271655.
11. Wang H, Di X, Wang Y, Ren B, Gao G, Deng J. An intelligent digital twin method based on spatio-temporal feature fusion for iot attack behavior identification. IEEE J Sel Areas Commun. 2023;41(11):3561–72. doi:10.1109/jsac.2023.3310091.
12. Zhao J, Xiong X, Chen Y. Design and application of a network planning system based on digital twin network. IEEE J Radio Freq Identif. 2022;6:900–4. doi:10.1109/jrfid.2022.3210750.
13. Jyeniskhan N, Keutayeva A, Kazbek G, Ali MH, Shehab E. Integrating machine learning model and digital twin system for additive manufacturing. IEEE Access. 2023;11:71113–26. doi:10.1109/access.2023.3294486.
14. Wen J, Yang J, Li Y, He J, Li Z, Song H. Behavior-based formation control digital twin for multi-AUG in edge computing. IEEE Trans Netw Sci Eng. 2023;10(5):2791–801. doi:10.1109/tnse.2022.3198818.
15. Lu Y, Maharjan S, Zhang Y. Adaptive edge association for wireless digital twin networks in 6G. IEEE Internet Things J. 2021;8(22):16219–30. doi:10.1109/jiot.2021.3098508.
16. Zhang Y, Zhang H, Lu Y, Sun W, Wei L, Zhang Y, et al. Adaptive digital twin placement and transfer in wireless computing power network. IEEE Internet Things J. 2024;11(6):10924–36. doi:10.1109/jiot.2023.3328380.
17. He T, Toosi AN, Buyya R. Efficient large-scale multiple migration planning and scheduling in SDN-enabled edge computing. IEEE Trans Mob Comput. 2024;23(6):6667–80. doi:10.1109/tmc.2023.3326610.
18. Mustafa E, Shuja J, Rehman F, Namoun A, Bilal M, Bilal K. Deep reinforcement learning and SQP-driven task offloading decisions in vehicular edge computing networks. Comput Netw. 2025;262:111180. doi:10.1016/j.comnet.2025.111180.
19. Mustafa E, Shuja J, Rehman F, Namoun A, Bilal M, Iqbal A. Computation offloading in vehicular communications using PPO-based deep reinforcement learning. J Supercomput. 2025;81(4):1–24. doi:10.1007/s11227-025-07009-z.
20. Shi Y, Yi C, Chen B, Yang C, Zhu K, Cai J. Joint online optimization of data sampling rate and preprocessing mode for edge–cloud collaboration-enabled industrial IoT. IEEE Internet Things J. 2022;9(17):16402–17. doi:10.1109/jiot.2022.3150386.
21. Jia Y, Zhang C, Huang Y, Zhang W. Lyapunov optimization based mobile edge computing for internet of vehicles systems. IEEE Trans Commun. 2022;70(11):7418–33. doi:10.1109/tcomm.2022.3206885.
22. Lin X, Wu J, Li J, Yang W, Guizani M. Stochastic digital-twin service demand with edge response: an incentive-based congestion control approach. IEEE Trans Mob Comput. 2023;22(4):2402–16. doi:10.1109/tmc.2021.3122013.
23. He Y, Ren Y, Zhou Z, Mumtaz S, Al-Rubaye S, Tsourdos A, et al. Two-timescale resource allocation for automated networks in IIoT. IEEE Trans Wirel Commun. 2022;21(10):7881–96. doi:10.1109/twc.2022.3162722.
24. Huang J, Golubchik L, Huang L. When lyapunov drift based queue scheduling meets adversarial bandit learning. IEEE/ACM Trans Netw. 2024;32(4):3034–44. doi:10.1109/tnet.2024.3374755.
25. Mohammadi M, Suraweera HA, Tellambura C. Uplink/Downlink rate analysis and impact of power allocation for full-duplex cloud-RANs. IEEE Trans Wirel Commun. 2018;17(9):5774–88. doi:10.1109/twc.2018.2849698.
26. Georgiadis L, Neely MJ, Tassiulas L. Resource allocation and cross-layer control in wireless networks. Found Trends® Netw. 2006;1(1):1–144. doi:10.1561/1300000001.
27. Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning. 2002;47(2):235–56.
28. Bieker-Walz L, Krajzewicz D, Morra A, Michelacci C, Cartolano F. Traffic simulation for all: a real world traffic scenario from the city of bologna. In: Modeling mobility with open data. Cham: Springer; 2015. p. 47–60. doi:10.1007/978-3-319-15024-6_4.
29. Ouyang T, Zhou Z, Chen X. Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing. IEEE J Sel Areas Commun. 2018;36(10):2333–45. doi:10.1109/iwqos.2018.8624174.

30. Chen X, Bi Y, Chen X, Zhao H, Cheng N, Li F, et al. Dynamic service migration and request routing for microservice in multicell mobile-edge computing. IEEE Internet Things J. 2022;9(15):13126–43. doi:10.1109/jiot.2022.3140183.
31. Shi Y, Yi C, Wang R, Wu Q, Chen B, Cai J. Service migration or task rerouting: a two-timescale online resource optimization for MEC. IEEE Trans Wireless Commun. 2024;23(2):1503–19. doi:10.1109/twc.2023.3290005.