QoE-Aware Task Executions on Service Models in DT-Assisted Edge Computing

Yuncan Zhang, *Member, IEEE*, Weifa Liang, *Senior Member, IEEE*, Yuanyuan Yang, *Life Fellow, IEEE*

Abstract—Mobile Edge Computing (MEC) shifts the computing power to the edge of core networks and provides important impetus in the flourishment of delay sensitive services at the network edge. Digital Twin (DT) technique enables object's behavior monitoring, analysis, and prediction through data analytics and artificial intelligence, which facilitates inference services provisioning based on machine learning models. In this paper, we deal with the Quality-of-Experience (QoE) issue of user satisfaction on inference services in DT-assisted MEC networks, through executing user tasks locally or offloaded to the MEC network, and we formulate two novel optimization problems: the utility maximization problem and the dynamic utility maximization problem, with the aim to maximize the total utility of user task executions in terms of QoEs and service delays of user satisfaction with the services. We first provide an Integer Linear Programming solution for the utility maximization problem when the problem size is small or medium; otherwise we devise a randomized algorithm with high probability, at the expense of bounded resource violations. We then develop an efficient online heuristic for the dynamic utility maximization problem, and for one of its special case without the bandwidth constraint, we devise an online algorithm with a provable competitive ratio. We finally evaluate the performance of proposed algorithms through simulations. The simulation results show that the proposed algorithms are promising.

Index Terms—Digital twin, task offloading, quality of experience, randomized algorithm and online algorithm, edge computing

1 Introduction

With the rapid development of the Internet of Things (IoT), the last decades witnessed billions of mobile devices such as smart phones, smart watches, tablets connecting to the Internet, paving the way towards a broad set of novel IoT applications and services such as smart city, autonomous vehicles, smart healthcare, AR/VR, etc [1]. Since most IoT devices have limited computing capacity due to their portable size, to expand their functionalities, one feasible way is to offload their computing-intensive tasks to remote clouds for processing. However, this approach incurs an unbearable service delay, due to the remoteness of clouds, thus, it is not applicable to delay sensitive applications such as autonomous driving [2]. Mobile Edge Computing (MEC) has emerged as a key technology to push computing and storage resources in the proximity of end devices. The offloading of tasks to nearby cloudlets in an MEC network becomes a promising paradigm for IoT devices to alleviate their computational burden and mitigate service delay [3]. On the other hand, Digital Twin (DT) as a disruptive technology has bridged physical objects with their powerful virtual representations, which can assist complex task executions [4], [5], [6]. Through DT modeling, the virtual avatar of an object can monitor behavior, record historical traces, make decisive predictions, and provide inference services on the object, by leveraging data analytics, artificial intelligence, and machine learning [7], [8].

In this paper, we deal with the Quality of Experience (QoE) aware task executions in a DT-assisted MEC network, by either offloading tasks to cloudlets in the MEC network for processing against service models in the cloudlets or processing tasks against its requested service models in devices locally, where there are multiple service models associated with each DT to provide inference services to users. To maintain high-fidelity services, each object needs synchronizing with its DT of its status and sensory data continuously [9], [10], [11], [12], and its DT service model needs to retrain quite often, using the update data from the object in order to enhance its service accuracy [11], [13]. One such an example is the traffic map navigation. Usually, users can download maps and navigate on their mobile devices. On the other hand, a navigation service model can be trained by the DT of a regional monitor, which can provide a better route based on real-time monitoring information, such as local traffic incidents, local weather conditions, and so on. This DT-assisted service task execution in an MEC network poses the following challenges.

First, for machine learning-based inference services, e.g., Deep Neural Networks (DNNs), both service delay and service accuracy affect the QoE of users. How to develop a metric to quantify the QoE of user satisfaction on a service is critical. Second, given a set of user task requests for inference services, it is challenging to maximize the accumulative QoE-based utility of task executions, by deciding which tasks to be processed locally in user devices and which tasks to be offloaded to cloudlets for executions, on their requested DT service models, considering the heterogeneity of processing capabilities of different cloudlets and user devices. Finally, DT service models can be enhanced through

- *Y. Zhang is with the School of Software Engineering, Sun Yat-Sen University, Zhuhai, P. R. China. Email: zhangyc29@mail.sysu.edu.cn*
- *W. Liang is with the Department of Computer Science, City University of Hong Kong, P. R. China Emails: weifa.liang@cityu.edu.hk*
- *Y. Yang is with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, USA. Email: yuanyuan.yang@stonybrook.edu*

retraining with the update data from their sources - objects. However, the limited bandwidth on each Access Point (AP) creates a competition between allocating a fractional bandwidth for DT updates versus task offloading, how to allocate the limited bandwidth among objects for their DT updates and users for their task offloading via APs is challenging. The rest of the paper will address the above mentioned challenges.

The novelty of this study lies in addressing the QoE issue of user satisfaction on inference services in a DT-assisted MEC network via task offloading, where the service accuracy of a request can be enhanced when the DT based service model is continually retrained, using the update data from its object. A novel utility metric is proposed to measure the QoE of user satisfaction on a service, which takes into account both service accuracy and service delay jointly. Also, performance-guaranteed algorithms for the problems of concern are devised to maximize the total utility of user task executions in both static and dynamic settings of task request arrivals.

The main contributions of the paper are presented as follows. We study user QoE satisfaction on services in a DT-assisted MEC network through task offloading, where a service request task can be either processed locally or offloaded to a cloudlet in the MEC for processing. To this end, we formulated two novel QoE-aware (static and dynamic) utility maximization problems. We first present an Integer Linear Program (ILP) solution for the utility maximization problem when the problem size is small or medium; otherwise we develop a randomized algorithm with high probability for the problem. We then devise an online algorithm for the dynamic utility maximization problem. Furthermore, for a special case of the dynamic utility maximization problem where the bandwidth capacity on each AP is sufficient, we propose an online algorithm with a provable competitive ratio. We finally evaluate the performance of the proposed algorithms via simulations, and simulation results demonstrate that the proposed algorithms are promising.

The remainder of the paper is organized as follows. Section 2 surveys the related work. Section 3 introduces the system model, cost modeling, metrics, and problem definitions. Section 4 deals with the utility maximization problem. Section 5 devises online algorithms for the dynamic utility maximization problem. Section 6 evaluates the proposed algorithms, and Section 7 concludes the paper.

2 Related work

This section surveys the state of the art on traditional and machine learning model based task offloading in DT-assisted MEC networks.

Traditional task offloading: The task offloading in MEC has been extensively studied in the past decades [14], [15]. For example, Li *et al.* [2] considered offloading delay-sensitive tasks to cloudlets or a remote cloud, for Network Function Virtualization (NFV) instances or Service Function Chain (SFC) services. They formulated two user service satisfaction problems and developed efficient algorithms for the problems. Chai *et al.* [16] investigated the problem of multitask joint computation offloading in satellite IoTs, where there is dependence between different tasks. Zhang *et al.* [17] formulated a cost minimization problem of task offloading in UAV-assisted MEC, and devised a privacy-preserving auction frameworks to schedule offloading tasks on the fly and incentivize UAVs' participation. Chen *et al.* [18] formulated a stochastic optimization problem for dynamic task offloading, which combines task scheduling and computing resource allocation decisions in MEC. Dai *et al.* [19] studied the UAV-assisted vehicular task offloading problem in MEC, with the aim to minimize vehicular task delay. However, they all consider traditional Virtual Network Function (VNF) services, in which the same input to an VNF instance by different users at different times leads to the same output [20].

Inference services based on machine learning models: A few recent studies focused on inference task offloading in MEC with inference accuracy requirements. For example, Xu *et al.* [21] studied offloading DNN inference requests in a 5G-enabled MEC, with the aim to minimize the total energy consumption of mobile devices, cloudlets and 5G base stations. Li *et al.* [22] dealt with partial offloading of a DNN model to MEC to speed up inferences between cloudlets and devices, by reducing the problem of concern into a maximum flow and minimum cut problem and devising an approximate solution to the problem. Ogden *et al.* [23] designed an inference framework - MDINFERENCE with the aim to increase the average accuracy of all models for inference requests while bounding service latency. Fresa *et al.* [3] formulated an ILP problem with the objective to maximize the total inference accuracy subject to a time constraint on the makespan.

Service accuracy and DT-assisted service model retraining: There is growing interest in DT-assisted intelligent service provisioning in MEC networks. The mentioned studies so far did not consider service accuracy dynamically changes over time due to the data drift of service models, not to mention utilizing DTs of data sources for service model training, thereby improving or maintain the service accuracy of service models. For instance, Li *et al.* [9], [12] investigated the Age-of-Information (AoI)-aware user satisfaction on DT-enabled service in MEC, and devised performance-guaranteed approximation algorithm and online algorithm for dynamic DT placements, with the aim to maximize service utility measured by AoI. Li *et al.* [10] studied AoI-aware IoT query services in DT-assisted MEC network. They formulated a minimization problem that explores nontrivial trade-offs between the freshness of query results and service delays. Liang *et al.* [11] considered inference service models retraining in DT-empowered MEC, with the aim to maximize the state freshness of inference service models while minimizing the retraining cost. Gu *et al.* [24] studied DT placement in wireless Digital Twin Network (DTN) to minimize the total cost that is the weighted sum of latency and energy consumption. They developed a multi-armed bandit driven algorithm to tackle the non-convex and non-stationary optimization problem. Yu *et al.* [6] formulated a QoE maximization problem based on attention-based resolution perception in DT-empowered Edge Computing (DTEC) systems, and utilized a continuous reinforcement learning framework to facilitate dynamic resource allocation in DTEC. Zhang *et al.* [13] devised efficient algorithms for DT placement and migration problems in

MEC by considering the mobility of objects, with the aim to minimize the weighted sum of freshness of DT data and the service cost of users requesting DT services. Zhang *et al.* [25] studied DT replica placements for mobility-aware service provisioning in MEC. However, these works did not tackle task offloading services in DT-assisted MEC networks.

DT-assisted task offloading: There are several recent studies on DT-assisted task offloading in MEC. For instance, Liu *et al.* [26] considered edge server selection and task offloading, integrating an MEC and its DT to construct a DT Edge Network (DTEN) and employing the virtual DTEN to monitor the status of the physical network. Cao *et al.* [27] studied task offloading in a DTN and proposed a five-objective optimization model in the DTN. Li *et al.* [28] proposed a DT-assisted optimization method for joint task offloading and resource allocation in multi-device collaborative tasks in Industrial Internet of Things (IIoT). They established a two-layer architecture: physical and DT layers, and formulated an optimization problem by jointly optimizing computational frequency, transmit power, bandwidth, and offloading factors. They then developed an alternating optimization algorithm for the problem by decoupling the problem into four subproblems. Guo *et al.* [29] designed a UAV deployment scheme in a DTN and presented two task offloading schemes using DT to obtain high-fidelity state information. Zhang *et al.* [30] studied task offloading in a DT-driven collaborative MEC, with the aim to maximize the MEC system's total revenue by leveraging the DT technology to mirror the status of the real system. Zhao *et al.* [31] developed a DRL algorithm to train the task offloading strategy in DT-based Vehicle Edge Computing (VEC), where DT technology enables to monitor the state of the VEC network in real time. In contrast, Yang *et al.* [5] offloaded tasks to Virtual Twins (VTs) of a Physical Twin (PT) for the task execution, where a VT consists of a generic model in clouds and a customized model updated by end devices. They jointly considered model placements of VTs, PTs' task offloading, and resource allocation to maximize the accuracy of task execution.

Different from the aforementioned works on task offloading, in this paper we jointly consider task offloading and DT updates in order to provide accurate yet low-delay intelligent services for users in DT-assisted edge computing environments. Specifically, we study two novel QoE-aware utility maximization problems for task offloading in DT-assisted MEC, with the aim of maximizing the total utility of all task executions. Table 1 highlights differences of our work from existing studies.

TABLE 1
Comparison with related works

Feature\Work	[3]	[5]	[6]	[10]	[11]	[19]	[24]	[26]	[27]	[28]	Ours
Delay optimization	×	×	×	✓	×	✓	✓	✓	✓	✓	✓
Accuracy optimization	✓	✓	×	✓	✓	×	×	×	×	×	✓
QoE modeling	×	×	✓	×	×	×	×	×	×	×	✓
DT participation	×	✓	✓	✓	✓	×	✓	✓	✓	✓	✓
Model retraining	×	✓	×	×	✓	×	×	×	×	×	✓

3 PRELIMINARY

In this section we introduce the system model, notions and notations, problem definitions, and NP-hard proof of the defined problems.

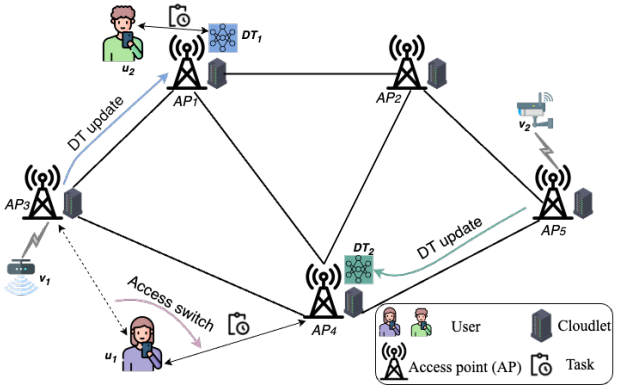


Fig. 1. An illustrative example of task executions on service models in a DT-assisted MEC network.

3.1 System model

We consider an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a set of Access Points (APs), and $\mathcal{E}$ is a set of links between APs. Denote by d_e the delay of transmitting a unit data on link $e \in \mathcal{E}$. Each AP is co-located with a cloudlet and the communication delay between them is negligible. Without loss of generality, we use AP and its co-located cloudlet interchangeably in this paper. Each AP $j \in \mathcal{N}$ is equipped with limited bandwidth capacity B_j, and its co-located cloudlet has an amount C_l of computing capacity. We assume that the Orthogonal Frequency-Division Multiple Access (OFDMA) scheme is adopted and each AP j has L_j orthogonal subchannels available with $L_j \geq 1$. Consider a monitoring time horizon $\mathbb{T}$ that is divided into T equal time slots, i.e., $\mathbb{T} = \{1, 2, \cdots, T\}$. At each time slot at most L_j devices can upload data through AP j.

Denote by V a set of objects that are under the coverage of different APs in the network. We assume that each object $v_i \in V$ has a DT, DT_i, which is a vivid virtual representation of the object. Each DT_i contains all the update data uploaded by its object by that time point. Associated with each DT_i, there is a service model (e.g., inference models) built upon its update data. The service models and DT_i itself are deployed in a cloudlet $n(DT_i) \in \mathcal{N}$. The residual computing capacity after the deployment of DTs and service models in cloudlet j is represented by C'_j. Specifically, the data generated by object v_i is uploaded to its gateway AP, $g(v_i) \in \mathcal{N}$, and then transmitted from $g(v_i)$ to cloudlet $l_i \in \mathcal{N}$ that hosts DT_i of v_i. Fig. 1 gives an illustrative example with two objects v_1 and v_2 and their DTs DT_1 and DT_2 in the network. The task of user u_1 is offloaded to the service model of DT_2 for execution from AP_4.

3.2 DT-assisted task execution for user task requests

Let U denote a set of user devices with different application tasks that need to be executed either by the user device locally or by offloading the task to an appropriate DT that is hosted by a cloudlet [5]. A user device can issue different types of tasks at different time slots. At time slot $t \in \mathbb{T}$, the task generated by user device $k \in U$ is expressed by a tuple $u_k(t) = \langle \mathcal{C}_k(t), s_k(t), r_k(t), c_k(t), D_k(t), \beta_k \rangle$, where $\mathcal{C}_k(t) \subset \mathcal{N}$ is the set of APs within the maximum transmission range of the user device k, $s_k(t)$ is the data size of the task, $r_k(t) \in [1, |V|]$ is the index of DT that

can perform this task, $c_k(t)$ is the amount of demanded computing resource, $D_k(t)$ is the service delay threshold, β_k is a delay tolerable factor with constant $\beta_k \geq 1$, and $\beta_k \cdot D_k(t)$ is the maximum service delay that the user device can tolerate [2]. The service delay is the end-to-end delay of the task from its issue time point to the service result returns. Let $U(t) = \{u_k(t) \mid k \in U\}$ be the set of user tasks arrived at time slot t.

Since a service model deployed in a user device generally does not change after its initial deployment, the service accuracy of a task execution against a service model in the local user device k does not change, and is represented by a_k^{loc} which is within [0,1] [5]. In contrast, when a service task request is offloaded to a cloudlet hosting its requested service model for service, then the service accuracy varies, which is determined by whether the service model is continuously retrained using the updated DT data. We here quantify the accuracy of a service model based on its DT update data as follows. Denote by $vol_i(t)$ the volume of update data generated by object v_i by time slot t since its last uploading. Once object v_i uploads its update data to its DT, the service model built upon the DT will be retrained, using the updated data and the service accuracy of the model will be improved. Denote by $\Phi_i(t)$ the accumulative volume of update data at DT_i by time slot t, which is calculated as follows.

$$\Phi_i(t) = \begin{cases} \Phi_i(t-1) + vol_i(t), & \text{if } v_i \text{ uploads data at } t \\ \Phi_i(t-1), & \text{otherwise} \end{cases} \quad (1)$$

It can be seen that $\Phi_i(t) \geq \Phi_i(t-1)$ and $\Phi_i(0) = 0$ for all $1 \leq i \leq |V|$. Intuitively, the more update data used for a model retraining, the higher the accuracy of the service model. Also, the growth rate of service accuracy of a model will decrease gradually with the increase on the amount of accumulated update data. Therefore, the accuracy of the service model of DT_i can be captured by a submodular, non-decreasing function $f_i(\cdot)$. We further assume that the retraining of each service model can be completed within one time slot. Since a service model cannot process user offloaded tasks during its retraining, the service model served at time slot t is the version that finished training at least by time slot $t-1$, which means that the accuracy of the service model of DT_i at time slot t is $f_i(\Phi_i(t-1))$. For ease of description, we treat each task running in its own device as it runs in a 'virtual' cloudlet 0. The accuracy $A_{k,l}(t)$ of user task $u_k(t)$ that is executed at cloudlet l at time slot t is calculated as follows.

$$A_{k,l}(t) = \begin{cases} a_k^{loc}, & \text{if } l = 0 \\ f_i(\Phi_i(t-1)), & \text{if } l \text{ hosts requested } DT_i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

3.3 Service delay of a user task execution

The service delay of a user task depends on where the task is executed. If task $u_k(t)$ is offloaded to the service model at $DT_{r_k(t)}$, its service delay consists of the uploading delay, the transmission delay of routing the task data from an AP within the transmission coverage of the user device to the cloudlet hosting $DT_{r_k(t)}$, and the processing delay at $DT_{r_k(t)}$, which are detailed in the following.

At time slot t, user device k uploads its task data through an AP $j \in \mathcal{C}_k(t)$. Recall that the OFDMA scheme is adopted at each AP, and assume that there are at most L_j users that can upload their data through AP j at each time slot. The upload rate of user device k to AP j is

$$b_{k,j}(t) = \frac{B_j}{L_j} \cdot \log(\, 1 + \frac{PX_k \cdot H_{k,j}}{\sigma^2}\,) \quad (3)$$

where PX_k is the transmission power of device k, $H_{k,j}$ is the channel gain between device k and AP $j \in \mathcal{N}$, and σ^2 is the white noise. The uploading delay of task $u_k(t)$ to AP j at time slot t is

$$d_{k,j}^{upload}(t) = \frac{s_k(t)}{b_{k,j}(t)}. \quad (4)$$

The transmission delay of routing the update data of task $u_k(t)$ from AP $j \in \mathcal{N}$ to cloudlet l that hosts the service model of $DT_{r_k(t)}$ is

$$d_{j,l_{r_k(t)}}^{comm}(t) = s_k(t) \cdot \sum_{e \in P_{j,l_{r_k(t)}}} d_e, \quad (5)$$

where $P_{j,l}$ is the shortest routing path in $\mathcal{G}$ between AP j and cloudlet l in terms of transmission delay.Task $u_k(t)$ can be offloaded to a cloudlet $r_k(t)$ that hosts $DT_{r_k(t)}$ or cloudlet 0 (device itself) for execution, where $r_k(t)$ is the cloudlet index of DT that can perform this task; otherwise, the task cannot be processed when it is offloaded to other cloudlets with mismatched DTs.

When task $u_k(t)$ is executed by the service model at $DT_{r_k(t)}$, the processing delay is

$$d_k^{comp}(t) = \frac{s_k(t)}{\mu_{r_k(t)}}, \quad (6)$$

where $\mu_{r_k(t)}$ is the processing rate of the service model at $DT_{r_k(t)}$. The service delay of user task $u_k(t)$ that is offloaded to cloudlet l hosting its requested DT through AP j is

$$d_{k,j,l}(t) = d_{k,j}^{upload}(t) + d_{j,l}^{comm}(t) + d_k^{comp}(t). \quad (7)$$

If task $u_k(t)$ is processed by the user device locally, i.e., it is 'offloaded' to itself - cloudlet 0, its service delay is

$$d_k^{loc}(t) = \frac{s_k(t)}{f_k}, \quad (8)$$

where f_k is the processing rate of user device k (cloudlet 0).

The service delay of user task $u_k(t)$ is defined as follows.

$$d_{k,j,l}(t) = \begin{cases} d_{k,j}^{upload}(t) & +d_{j,l}^{comm}(t) + d_k^{comp}(t), \\ & \text{if } l = n(DT_{r_k}) \wedge j \in \mathcal{C}_k(t) \\ d_k^{loc}(t), & \text{if } j = 0 \wedge l = 0 \\ \infty, & \text{otherwise} \end{cases} \quad (9)$$

3.4 User satisfaction on service delay

Users requesting task executions in most applications have their tolerable delay requirements [2]. Specifically, for a user task $u_k(t)$ issued at time slot t, if the actual service delay is within the specified delay threshold $D_k(t)$ of user

k, the user fully satisfies with the service; otherwise, the user satisfaction will be inversely proportional to the delay beyond his/her specified delay threshold. We here use a constant $\beta \geq 1$ to capture a certain degree of delay tolerances by users, where a larger β implies that each user is more tolerable to his/her service delay. When a service delay is beyond the maximum service delay threshold of any user, the user will not be satisfied on the service totally. Thus, we model the user satisfaction on a service delay by a function as follows.

$$\rho_{k,j,l}(t) = \begin{cases} 1, & \text{if } d_{k,j,l}(t) \leq D_k(t) \\ \frac{\beta \cdot D_k(t) - d_{k,j,l}(t)}{\beta \cdot D_k(t) - D_k(t)}, & \text{if } D_k(t) < d_{k,j,l}(t) < \beta D_k(t) \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The intuition behind the utility function $\rho_{k,j,l}(t)$ in (10) is that if the service delay is no greater than the predefined threshold $D_k(t)$ of user u_k, then the user fully satisfies the service with utility 1. Otherwise (if the service delay is within the tolerable range of the user, i.e., the delay is within $(D_k(t), \beta_k \cdot D_k(t)]$), the user satisfies the service partially, the utilization obtained for the service is a fractional value between 0 and 1, and the value is inversely proportional to the service delay. Otherwise (the service delay is beyond the user expectation (i.e., the delay is strictly larger than $\beta_k \cdot D_k(t)$), the value assigned to the utility function is 0, implying that the user does not satisfy the service. No credit will be given to the service provider.

3.5 QoE metric and optimization objective

We measure the QoE of user satisfaction with a service as a utility function of its task execution, which is a linear combination of the accuracy of the service model on the requested DT by the user and the satisfaction of the user on the service delay at time slot t as follows.

$$Q_{k,j,l}(t) = A_{k,l}(t) + \omega \cdot \rho_{k,j,l}(t), \tag{11}$$

where the service delay of a user consists of the user location j (data uploading by AP j) and the task processing in cloudlet l (if cloudlet l hosts the requested service model of the user), $A_{k,l}(t)$ represents the inference accuracy of task $u_k(t)$ on its requested model service at cloudlet l, and ω is a fixed coefficient that is used to balance between the service accuracy and user satisfaction on the service delay of the requested service with $0 \leq \omega \leq 1$.

The QoE metric plays a pivotal role in bridging user perception and system resource optimization. By jointly modeling service accuracy and service delay as user satisfaction through an adjustable constant ω, the QoE metric directly governs two critical aspects in the system model. First, it serves as the optimization objective for resource allocation, where tasks are dynamically assigned to either local devices or cloudlets by the QoE calculation, ensuring fine balancing between service precision (critical for DT model fidelity) and responsiveness (essential for interactive services). Second, parameter ω enables an adaptive control policy for diverse services. For instance, a larger value of ω prioritizes service accuracy for mission-critical applications like medical diagnostics, while a smaller value favors delay-sensitive tasks such as AR/VR.

3.6 Problem definitions

In the following, we consider two novel optimization problems of user task executions in their own devices or cloudlets in the MEC network. We first consider the utility maximization problem, where all user task requests are given.

Definition 1. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, the DTs of a set V of objects that have been deployed in cloudlets already, and a set U of user task requests, each task $u_k \in U$ is represented by a tuple $\langle \mathcal{C}_k, s_k, r_k, c_k, D_k, \beta_k \rangle$. The utility maximization problem in $\mathcal{G}$ is to maximize the utility of all user tasks in U through determining whether each task is executed in its own device or offloaded, subject to the bandwidth capacity on each AP and computing capacity on each cloudlet.*

The utility maximization problem considered a set U of user task requests in a snapshot case, assuming that the accuracy of each service model built on a DT is fixed in that snapshot. In the following, we consider a more generic case, where user task requests arrive one by one for a given time horizon $\mathbb{T}$.

Definition 2. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, the DTs of a set V of objects that are already deployed in cloudlets, and a given finite time horizon $\mathbb{T}$, user tasks arrive one by one without the knowledge of future arrivals. Each arrived user task $u_k(t) \in U(t)$ at time slot $t \in \mathbb{T}$ is represented by a tuple $\langle \mathcal{C}_k(t), s_k(t), r_k(t), c_k(t), D_k(t), \beta_k \rangle$, which needs to be processed locally or offloaded to a DT for task execution immediately. The dynamic utility maximization problem is to maximize the utility sum of all user tasks over the given time horizon $\mathbb{T}$, through scheduling DT updates and task offloading decisions at each time slot t, subject to the bandwidth capacity on each AP and computing capacity on each cloudlet.*

3.7 NP-hardness

Theorem 1. *The utility maximization problem in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is NP-hard.*

Proof. We prove the NP-hardness of the utility maximization problem by a reduction from a well-known NP-hard problem - the maximum profit Generalized Assignment Problem (GAP) [32] that is defined as follows. Given $|\mathcal{N}|+1$ bins and $|U|$ items, if item $u_k \in U$ is packed to bin l, it leads to a profit $p_{k,l}$ and weight $w_{k,l}$. Each bin has a limited capacity. The GAP is to pack as many items as possible to the bins such that the total profit of the packed items is maximized, subject to bin capacities.

We consider a special case of the utility maximization problem, where each AP has sufficient bandwidth to enable all users under its coverage to offload their tasks. We treat each cloudlet l as a bin c_l with capacity C'_l, i.e., the residual computing capacity of cloudlet l after all DT deployments with $l \in \mathcal{N}$. We assume that bin 0 is a virtual cloudlet that corresponds to the device itself to process its task. We treat each task as an item. Let $w_{k,l}$ be the weight of item $u_k(t)$ packed in bin c_l with profit $p_{k,l}(t)$, which is defined as follows.

$$p_{k,l}(t) = \begin{cases} Q_{k,0,0}, & \text{if } l = 0 \\ \max\{Q_{k,j,l}(t) \mid j \in \mathcal{C}_k(t)\}, & \text{if } l = n(DT_{r_k}) \\ 0, & \text{otherwise} \end{cases}$$

It can be seen that this special utility maximization problem is equivalent to the maximum profit GAP. Hence, the total utility maximization problem is NP-hard. □

Theorem 2. *The dynamic utility maximization problem in an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is NP-hard.*

Proof. The utility maximization problem is a special case of the dynamic utility maximization problem when the time horizon $\mathbb{T}$ consists of a single time slot. The dynamic utility maximization problem thus is NP-hard, too. □

For the sake of convenience, the symbols used in this paper are summarized in Table 2.

4 Algorithms for the Utility Maximization Problem

In this section, we deal with the utility maximization problem by formulating an ILP solution when the problem size is small. Otherwise, we develop a randomized algorithm at the expense of a bounded resource violation.

4.1 ILP solution

Let $x_{k,j,l}$ be a binary variable indicating whether user task u_k is offloaded through AP j and processed at cloudlet l ($x_{k,j,l} = 1$) or not ($x_{k,j,l} = 0$). Note that the range of AP index j is $0 \leq j \leq |\mathcal{N}|$, where bin 0 is a virtual AP with unlimited numbers of subchannels that correspond to the local processing of a task. The accuracy and service delay of assigning task u_k to cloudlet j are calculated by Eq. (2) and Eq. (9), respectively. The utility gain $Q_{k,j,l}$ of offloading task u_k through AP j to cloudlet l is calculated by Eq. (11). The ILP for the problem is formulated, and the objective (12) aims to maximize the total utility gain of all user task execution requests in U as follows.

$$\text{Maximize} \sum_{u_k \in U} \sum_{j=0}^{|\mathcal{N}|} \sum_{l=0}^{|\mathcal{N}|} Q_{k,j,l} \cdot x_{k,j,l} \tag{12}$$

$$\text{subject to } (1) - (11),$$

$$\sum_{j=0}^{|\mathcal{N}|} \sum_{l=0}^{|\mathcal{N}|} x_{k,j,l} = 1, \forall u_k \in U \tag{13}$$

$$\sum_{u_k \in U} \sum_{l=0}^{|\mathcal{N}|} x_{k,j,l} \leq L_j, \forall j \in \{0\} \cup \mathcal{N} \tag{14}$$

$$\sum_{u_k \in U} \sum_{j=0}^{|\mathcal{N}|} c_k \cdot x_{k,j,l} \leq C'_l, \forall l \in \{0\} \cup \mathcal{N} \tag{15}$$

$$x_{k,j,l} = 0, \forall u_k \in U, \text{ if } j \neq 0 \wedge j \notin \mathcal{C}_k \text{ or } j = 0 \wedge l \neq 0 \tag{16}$$

$$x_{k,j,l} = 0, \forall u_k \in U, \text{ if } l \neq 0 \wedge l \neq n(DT_{r_k}) \text{ or } l = 0 \wedge j \neq 0 \tag{17}$$

$$x_{k,j,l} \in \{0,1\}, \forall k \in U, \forall j \in \{0\} \cup \mathcal{N} \tag{18}$$

where Constraint (13) ensures that each task is either processed locally or offloaded to its requested DT in a cloudlet for execution. Constraint (14) ensures that the number of user requests allocated to an AP is no greater than the number of its sub-channels. Constraint (15) ensures that the total amount of computing resources demanded by all task executions in any cloudlet cannot exceed its computing capacity, where C'_l indicates the residual computing capacity after DT deployments at cloudlet l. Constraint (16) ensures that each user task can only upload its data through an AP within its transmission range or processed locally ($j = 0, l = 0$). Constraint (17) ensures that a task can only be offloaded to the cloudlet that hosts its requested DT or processed locally, where r_k is the requested DT of user task u_k.

Algorithm 1 Randomized algorithm for the utility maximization problem

Input: An MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a set V of objects with their DTs deployed in $\mathcal{G}$, and a set U of user tasks.
Output: The task offloading decisions to maximize the total utility gain of user tasks.

```
Calculate the utility value Q_{k,j,l} for each task u_k;
Solve the linear relaxation LP of the ILP (12);
Let {x̃_{k,j,l} | u_k ∈ U, j ∈ [0,|N|], l ∈ [0,|N|]} be the optimal LP solution;
x̂_{k,j,l} ← 0; /*initialize the integral solution*/
for each u_k ∈ U do
    g_x ← a randomly generated number ∈ [0,1];
    flag ← false;
    for each j ∈ [0,|N|] do
        for each l ∈ [0,|N|] do
            if Σ_{j'=0}^{j} Σ_{l'=0}^{l} x̃_{k,j',l'} ≥ g_x then
                x̂_{k,j,l} ← 1 /* task u_k is offloaded to cloudlet l
                through AP j exclusively */
                flag ← true;
                break;
            end if
        end for
        if flag is true then
            break;
        end if
    end for
end for
return An integral solution {x̂_{k,j,l} | u_k ∈ U, j ∈ [0,|N|], l ∈ [0,|N|]}
```

4.2 Randomized algorithm

Since finding an ILP solution in (12) takes prohibitive time when the problem size is large, we instead propose a randomized algorithm for the problem. Let $\tilde{x}_{k,j,l}$ be the value of variable $x_{k,j,l}$ in the optimal solution of the LP relaxation of the ILP, where $\tilde{x}_{k,j,l} \in [0,1]$. Let $\{\hat{x}_{k,j,l} \mid u_k \in U, j \in [0,|\mathcal{N}|], l \in [0,|\mathcal{N}|]\}$ be an integral solution of the ILP. In the rounding procedure, each random variable $\hat{x}_{k,j,l}$ is set to 1 with probability of $\tilde{x}_{k,j,l}$. This setting is taken by Constraint (13) in an exclusive manner, which means that associated with each task $u_k \in U$, there is exactly one variable $\hat{x}_{k,j,l}$ to be 1 and the rest are set to 0, for each pair of $j \in [0,|\mathcal{N}|]$ and $l \in [0,|\mathcal{N}|]$.

The randomized algorithm, `Algorithm 1`, for the utility maximization problem is a snapshot case where all model accuracy are given and fixed, and all user service requests are given, too. The detailed description of the randomized algorithm is given in `Algorithm 1`.

TABLE 2
Table of notations

Notation	Descriptions
$\mathcal{G}=(\mathcal{N},\mathcal{E})$	An MEC network with a set $\mathcal{N}$ of APs and a set $\mathcal{E}$ of links
B_j, C_l	The bandwidth capacity of AP_j and the computing resource capacity on its co-located cloudlet l
d_e	The transmission delay per unit data transfer along link $e \in \mathcal{E}$
L_j	The number of sub-channels on AP j
V	A set of objects that are under the coverage of different APs in the network
$g(v_i)$	The gateway AP of object $v_i \in V$
DT_i, $n(DT_i)$	The DT of object $v_i \in V$ and its hosting cloudlet $n(DT_i) \in \mathcal{N}$
$\mathbb{T}$	A monitoring time horizon that is divided into T equal time slots, i.e., $\mathbb{T}=\{1,2,\cdots,T\}$
$U(t)$	The set of user tasks arrived at time slot t
$u_k(t)$	The task generated by user device $k \in U$ at time slot $t \in \mathbb{T}$
$\mathcal{C}_k(t)$	The set of APs within the maximum transmission range of the user device k
$s_k(t)$	The data size of task $u_k(t)$
$r_k(t)$	The index of DT that can perform this task where $r_k(t) \in [1,\|V\|]$
$c_k(t)$	The amount of demanded computing resource of task $u_k(t)$
$D_k(t)$	The service delay threshold of task $u_k(t)$
β_k	A delay tolerable factor with constant $\beta_k \geq 1$ of task $u_k(t)$
a_k^{loc}	The task execution accuracy of local processing on user device k
$vol_i(t)$	The volume of update data generated by object v_i by time slot t since its last uploading
$\Phi_i(t)$	The accumulative volume of update data at DT_i by time slot t
$A_{k,l}(t)$	The accuracy of user task $u_k(t)$ that is executed at cloudlet l at time slot t
$b_{k,j}(t)$, $d_{k,j}^{upload}(t)$	The upload rate and uploading delay of task $u_k(t)$ to AP j at time slot t
$d_{j,l_{r_k(t)}}^{comm}(t)$	The transmission delay of routing the update data of task $u_k(t)$ from AP $j \in \mathcal{N}$ to cloudlet l that hosts the service model of $DT_{r_k(t)}$
$d_k^{comp}(t)$	The processing delay of task $u_k(t)$ executed by the service model at $DT_{r_k(t)}$
$d_{k,j,l}(t)$	The service delay of user task $u_k(t)$ that is offloaded to cloudlet l through AP j
$\rho_{k,j,l}(t)$	The user satisfaction of task $u_k(t)$ on a service delay when it is offloaded to cloudlet l through AP j
$Q_{k,j,l}(t)$	The QoE of task $u_k(t)$ when it is offloaded to cloudlet l through AP j
$x_{k,j,l}$	Binary variable indicating whether user task u_k is offloaded through AP j and processed at cloudlet l or not
$\eta_i(t)$	A metric to measure the potential benefit by retraining the service model of DT_i using the update data of object v_i at time slot $t \in \mathbb{T}$

4.3 Algorithm analysis

In the following, we show the performance guarantee of the proposed randomized algorithm. We also analyze the time complexity of the proposed randomized algorithm.

Lemma 1. *For any user task $u_k \in U$, $\hat{x}_{k,j,l}$ is set to 1 with probability $\tilde{x}_{k,j,l}$ exclusively by* `Algorithm 1`.

Proof. Following `Algorithm 1`, Step 10 ensures that there must exist a pair j and l such that condition $\sum_{j'=0}^{j}\sum_{l'=0}^{l}\tilde{x}_{i,j',l'} \geq g_x$ holds, as the value range of random variable g_x is in $[0,1]$, and random variable $\tilde{x}_{i,j,l}$ of each valid pair of j and l is independent of each other. Therefore, each user task is exclusively assigned to a pair of j and l, and the probability of $\hat{x}_{i,j,l}=1$ is $\tilde{x}_{i,j',l'}$. □

To analyze the approximation ratio of `Algorithm 1` and bound its resource capacity violations on each AP and each cloudlet, we define a positive parameter Λ as follows.

$$\Lambda = \max\{\max\{Q_{k,j,l} \mid u_k \in U, j \in \mathcal{N}\cup\{0\}, l \in \mathcal{N}\cup\{0\}\}, \max\{c_k \mid u_k \in U\}, 1\} \quad (19)$$

Theorem 3. *Given an MEC network $\mathcal{G}=(\mathcal{N},\mathcal{E})$, the DTs of a set V of objects that have been deployed in cloudlets already, and a set U of user task execution requests, there is a randomized algorithm,* `Algorithm 1`*, with approximation ratio $\frac{1}{2}$ for the utility gain maximization problem with high probability of $\min\{1-\frac{1}{|U|}, 1-\frac{1}{|\mathcal{N}|}\}$, the solution is obtained at the expense of no more than twice the computing capacity on any cloudlet and the number of subchannels on any AP, provided that $\widetilde{OPT} \geq 8\Lambda\ln|U|$, $\min\{C_l' \mid l \in \mathcal{N}\} \geq 6\Lambda\ln|\mathcal{N}|$, and $\min\{L_j \mid j \in \mathcal{N}\} \geq 6\Lambda\ln|\mathcal{N}|$, and the algorithm takes $O(((|U|\cdot|\mathcal{N}|^2)^{2+\frac{1}{6}})\log(|U|\cdot|\mathcal{N}|^2))$ time, where Λ is defined in Eq.* (19).

Proof. Let OPT and $\widetilde{OPT}$ be the values of the optimal solution of the ILP in (12) and of the its LP, respectively. As the studied problem is a maximization problem, we have $\widetilde{OPT} \geq OPT$. We first analyze the approximation ratio of the randomized algorithm, `Algorithm 1`. Let $y_{k,j,l} = \frac{Q_{k,j,l}}{\Lambda}\cdot\hat{x}_{k,j,l}$ be a random variable deriving from random variable $\hat{x}_{k,j,l}$.It can be seen that $y_{k,j,l}$ equals $\frac{Q_{k,j,l}}{\Lambda}$ with probability $\tilde{x}_{k,j,l}$; otherwise, $y_{k,j,l}$ equals 0. Due to $\frac{Q_{k,j,l}}{\Lambda}\cdot\hat{x}_{k,j,l} \leq \frac{Q_{k,j,l}}{\max\{Q_{k,j,l} \mid u_k\in U, j\in[0,|\mathcal{N}|], l\in[0,|\mathcal{N}|]\}} \leq 1$, we have $0 \leq y_{k,j,l} \leq 1$. The expected value of $y_{k,j,l}$ is $\mathbb{E}[y_{k,j,l}] = \frac{Q_{k,j,l}}{\Lambda}\cdot\tilde{x}_{k,j,l}$. We have

$$\mathbb{E}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\sum_{l=0}^{|\mathcal{N}|} y_{k,j,l}] = \sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\sum_{l=0}^{|\mathcal{N}|}\frac{Q_{k,j,l}\cdot\tilde{x}_{i,j,m,k}}{\Lambda} = \frac{\widetilde{OPT}}{\Lambda} \quad (20)$$

Let α be a constant with $0 < \alpha \leq 1/2$. By Chernoff Bounds [33], we have

$$\mathbf{Pr}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\sum_{l=0}^{|\mathcal{N}|} Q_{k,j,l}\cdot\hat{x}_{k,j,l} \leq (1-\alpha)\cdot OPT]$$

$$\leq \mathbf{Pr}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\sum_{l=0}^{|\mathcal{N}|} Q_{k,j,l}\cdot \hat{x}_{k,j,l} \leq (1-\alpha)\cdot \widetilde{OPT}],$$

$$= \mathbf{Pr}[\sum_{u_k\in U} Y_k \leq (1-\alpha)\cdot \frac{\widetilde{OPT}}{\Lambda}],$$

$$\text{let } Y_k = \sum_{j=0}^{|\mathcal{N}|}\sum_{l=0}^{|\mathcal{N}|} Q_{k,j,l}/\Lambda \cdot \hat{x}_{k,j,l}$$

$$\leq exp(\frac{-\alpha^2\cdot \widetilde{OPT}}{2\Lambda}), \text{ by the Chernoff Bounds [33]. (21)}$$

Assuming that $exp(\frac{-\alpha^2\cdot\widetilde{OPT}}{2\Lambda}) \leq \frac{1}{|U|}$ and $0<\alpha<1/2$, we have $\alpha \geq \sqrt{\frac{2\Lambda\ln|U|}{\widetilde{OPT}}}$, i.e., $\widetilde{OPT} \geq \frac{2\Lambda\ln|U|}{\alpha^2}$. Because $1-\alpha \geq \frac{1}{2}$, the approximation ratio of `Algorithm` 1 is $\frac{1}{2}$ with high probability $1-\frac{1}{|U|}$.

We then show that the violation ratio of computing capacity on any cloudlet $l \in \mathcal{N}$ is upper bounded by 2 as follows. Let $z_{k,j,l}$ be a random variable derived from random variable $\hat{x}_{k,j,l}$, where $z_{k,j,l} = \frac{c_k}{\Lambda}\cdot\hat{x}_{k,j,l}$. Due to $\frac{c_k}{\Lambda}\cdot\hat{x}_{k,j,l} \leq \frac{c_k}{\max\{c_k \mid u_k\in U\}} \leq 1$, the expectation of $z_{k,j,l}$ is $\mathbb{E}[z_{k,j,l}] = \frac{c_k}{\Lambda}\cdot\tilde{x}_{k,j,l}$ with $0 \leq z_{k,j,l} \leq 1$. For each cloudlet $l\in\mathcal{N}$, we have $\mathbb{E}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} z_{k,j,l}] = \sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\frac{c_k\cdot\tilde{x}_{k,j,l}}{\Lambda} = \frac{\widetilde{C}'_l}{\Lambda}$, where $\widetilde{C}'_l$ is the amount of computing resource consumed on cloudlet l by the LP and there is $\widetilde{C}'_l \leq C'_l$. Let β be a constant with $0<\beta\leq 1$. The total amount of computing resource consumed on cloudlet l is $\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} c_k\cdot\hat{x}_{k,j,l}$ in the solution by `Algorithm` 1. The probability of the computing capacity violation on cloudlet $l\in\mathcal{N}$ is

$$\mathbf{Pr}[\bigvee_{l\in\mathcal{N}}\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} c_k\cdot\hat{x}_{k,j,l} \geq (1+\beta)\cdot C'_l]$$

$$\leq \mathbf{Pr}[\bigvee_{l\in\mathcal{N}}\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} c_k\cdot\hat{x}_{k,j,l} \geq (1+\beta)\cdot \widetilde{C}'_l],$$

$$\leq \sum_{l\in\mathcal{N}}\mathbf{Pr}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} z_{k,j,l} \geq (1+\beta)\cdot\frac{\widetilde{C}'_l}{\Lambda}],$$

let $z_{k,j,l} = \frac{c_k}{\Lambda}\cdot\hat{x}_{k,j,l}$ and by the Union Bound Inequality

$$\leq \sum_{l\in\mathcal{N}}\mathbf{Pr}[\sum_{u_k\in U} Z_k \geq (1+\beta)\cdot\frac{\widetilde{C}'_l}{\Lambda}], \text{ let } Z_k = \sum_{j=0}^{|\mathcal{N}|} z_{k,j,l}$$

$$\leq |\mathcal{N}|\cdot exp(\frac{-\beta^2\cdot\widetilde{C}'_l}{(2+\beta)\cdot\Lambda}), \quad \text{by the Chernoff Bounds [33].} \tag{22}$$

Assuming that $exp(\frac{-\beta^2\cdot\widetilde{C}'_l}{(2+\beta)\cdot\Lambda}) \leq \frac{1}{|\mathcal{N}|^2}$, we have $exp(\frac{-\beta^2\cdot\widetilde{C}'_l}{6\Lambda}) \leq \frac{1}{|\mathcal{N}|^2}$ as $0<\beta\leq 1$. Then, we have $\beta \geq \sqrt{\frac{6\Lambda\ln|\mathcal{N}|}{\widetilde{C}'_l}} \geq \sqrt{\frac{6\Lambda\ln|\mathcal{N}|}{C'_l}}$, due to $\widetilde{C}'_l \leq C'_l$. Since $\beta\leq 1$, we have $C'_l \geq 6\Lambda\ln|\mathcal{N}|$ for any cloudlet $l\in\mathcal{N}$, i.e., $\min\{C'_l \mid l\in\mathcal{N}\} \geq 6\Lambda\ln|\mathcal{N}|$. By Eq. (22) and $C'_l \leq C_l$, we have

$$\mathbf{Pr}[\bigvee_{l\in\mathcal{N}}\sum_{u_k\in U} Z_k \geq (1+\beta)\cdot C_l]$$

$$\leq \mathbf{Pr}[\bigvee_{l\in\mathcal{N}}\sum_{u_k\in U} Z_k \geq (1+\beta)\cdot C'_l] \leq |\mathcal{N}|\cdot\frac{1}{|\mathcal{N}|^2} = \frac{1}{|\mathcal{N}|}.$$

Because $1+\beta\leq 2$, the computing resource consumption on each cloudlet $l\in\mathcal{N}$ is no more than twice its capacity with high probability of $1-\frac{1}{|\mathcal{N}|}$, provided that $\min\{C'_l \mid \forall l\in\mathcal{N}\} \geq 6\Lambda\ln|\mathcal{N}|$.

We thirdly show the violation ratio of the number of subchannels on AP $j\in\mathcal{N}$ is upper bounded by 2 as follows. Let $h_{k,j,l}$ be a random variable derived from random variable $\hat{x}_{k,j,l}$, where $h_{k,j,l} = \frac{1}{\Lambda}\cdot\hat{x}_{k,j,l}$. Due to $\frac{1}{\Lambda}\cdot\hat{x}_{k,j,l}\leq 1$, the expectation of $h_{k,j,l}$ is $\mathbb{E}[h_{k,j,l}] = \frac{1}{\Lambda}\cdot\tilde{x}_{k,j,l}$ with $0\leq h_{k,j,l}\leq 1$. For each AP $j\in\mathcal{N}$, we have $\mathbb{E}[\sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|} h_{k,j,l}] = \sum_{u_k\in U}\sum_{j=0}^{|\mathcal{N}|}\frac{1\cdot\tilde{x}_{k,j,l}}{\Lambda} = \frac{\widetilde{L}_j}{\Lambda}$, where $\widetilde{L}_j$ is the number of subchannels used of AP j in the solution of the LP with $\widetilde{L}_j \leq L_j$. Let λ be a constant with $0<\lambda\leq 1$. The number of subchannels used on AP j is $\sum_{u_k\in U}\sum_{l=0}^{|\mathcal{N}|}\hat{x}_{k,j,l}$ by `Algorithm` 1. The probability of the number of subchannel violations on any AP $j\in\mathcal{N}$ then is

$$\mathbf{Pr}[\bigvee_{j\in\mathcal{N}}\sum_{u_k\in U}\sum_{l=0}^{|\mathcal{N}|} 1\cdot\hat{x}_{k,j,l} \geq (1+\lambda)\cdot L_j]$$

$$\leq \mathbf{Pr}[\bigvee_{j\in\mathcal{N}}\sum_{u_k\in U}\sum_{l=0}^{|\mathcal{N}|} 1\cdot\hat{x}_{k,j,l} \geq (1+\lambda)\cdot \widetilde{L}_j],$$

$$\leq \sum_{j\in\mathcal{N}}\mathbf{Pr}[\sum_{u_k\in U}\sum_{l=0}^{|\mathcal{N}|} h_{k,j,l} \geq (1+\lambda)\cdot\frac{\widetilde{L}_j}{\Lambda}],$$

by $h_{k,j,l} = \frac{1}{\Lambda}\cdot\hat{x}_{k,j,l}$ and the Union Bound Inequality

$$\leq \sum_{j\in\mathcal{N}}\mathbf{Pr}[\sum_{u_k\in U} H_k \geq (1+\lambda)\cdot\frac{\widetilde{L}_j}{\Lambda}], \text{let } H_k = \sum_{l=0}^{|\mathcal{N}|} h_{k,j,l}$$

$$\leq |\mathcal{N}|\cdot exp(\frac{-\lambda^2\cdot\widetilde{L}_j}{(2+\lambda)\cdot\Lambda}), \quad \text{by the Chernoff Bounds [33].} \tag{23}$$

Because $0<\lambda\leq 1$, we assume that $exp(\frac{-\Lambda^2\cdot\widetilde{L}_j}{(2+\Lambda)\cdot\Lambda}) \leq \frac{1}{|\mathcal{N}|^2}$, and we then have $exp(\frac{-\lambda^2\cdot\widetilde{L}_j}{6\Lambda}) \leq \frac{1}{|\mathcal{N}|^2}$, i.e., $\Lambda \geq \sqrt{\frac{6\Lambda\ln|\mathcal{N}|}{\widetilde{L}_j}} \geq \sqrt{\frac{6\Lambda\ln|\mathcal{N}|}{L_j}}$ due to $\widetilde{L}_j\leq L_j$. Since $\lambda\leq 1$, we have $L_j \geq 6\Lambda\ln|\mathcal{N}|$ for any AP $j\in\mathcal{N}$, i.e., $\min\{L_j \mid j\in\mathcal{N}\} \geq 6\Lambda\ln|\mathcal{N}|$. By Eq. (23), we have

$$\mathbf{Pr}[\bigvee_{j\in\mathcal{N}}\sum_{u_k\in U} H_k \geq (1+\lambda)\cdot L_j] \leq |\mathcal{N}|\cdot\frac{1}{|\mathcal{N}|^2} = \frac{1}{|\mathcal{N}|}.$$

We finally analyze the time complexity of `Algorithm` 1. The utility value calculation of $Q_{k,j,l}$ of all tasks takes $O(|U|\cdot|\mathcal{N}|^2)$ time. The linear relaxation LP of the ILP (12) is solved by a solver. According to [34], an LP with n variables can be solved in time $O(n^{2+\frac{1}{6}}\log(\frac{n}{\delta}))$, where δ is the relative accuracy. The LP of the ILP (12) consists of $|U|\cdot|\mathcal{N}|^2$ variables. By setting $\delta=1$ [35], finding a solution for the LP takes $O((|U|\cdot|\mathcal{N}|^2)^{2+\frac{1}{6}})\log(|U|\cdot|\mathcal{N}|^2)$ time. The rounding procedure for all variables takes $O(|U|\cdot|\mathcal{N}|^2)$ time. Therefore, the time complexity of `Algorithm` 1 is $O(((|U|\cdot|\mathcal{N}|^2)^{2+\frac{1}{6}})\log(|U|\cdot|\mathcal{N}|^2))$.

The theorem then follows. □

5 Algorithms for the Dynamic Utility Maximization Problem

In this section, we study the dynamic utility maximization problem. We first devise an online algorithm for the problem with the generic setting. We then develop an online algorithm with a provable competitive ratio for a special case of the problem, where each AP has abundant bandwidth to enable all devices under its coverage to upload their data via the AP to the MEC network.

5.1 Online algorithm for the dynamic utility maximization problem

The basic idea of the proposed online algorithm for the dynamic utility maximization problem is adopting exponential functions of various resource usages to govern task offloading with the aim of balancing dynamic workload at each cloudlet and AP. Specifically, the proposed algorithm proceeds iteratively, and each iteration corresponds to one time slot. Within each time slot, it decides which objects to upload their update data through which APs, and which user devices to offload their tasks through which APs to the cloudlets hosting their requested DT service models. To this end, we show how to schedule DT updates for service model retraining. We make use of a metric $\eta_i(t)$ to measure the potential benefit by retraining the service model of DT_i using the update data from object v_i at time slot $t \in \mathbb{T}$. Denote by $U_i(t)$ the set of user tasks requesting the service model of DT_i at slot t, i.e., $U_i(t) = \{u_k(t) |\ r_k(t) = i, u_k(t) \in U(t)\}$. The metric $\eta_i(t)$ of object v_i is defined as follows.

$$\eta_i(t) = \sum_{t'=1}^{t-1} \sum_{u_k(t') \in U_i(t')} (f_i(\Phi(t'-1) + vol_i(t')) - f_i(\Phi(t'-1))) \tag{24}$$

where $\eta_i(t)$ is the potential benefit brought if object v_i updates DT_i at time slot t, and $f_i(\Phi(t-1) + vol_i(t)) - f_i(\Phi(t-1))$ is the utility gain of one task execution at the retrained service model at DT_i, using the update data $vol_i(t)$.

Because we need reserving bandwidth for task offloading, we only allow up to $\lfloor \sigma_1 \cdot L_j \rfloor$ of subchannels of each AP j to be used for DT updates, where σ_1 is a given threshold with $\sigma_1 \in [0, 1]$. For each AP j, the objects under its coverage are examined one by one, and an object is chosen to upload to update its DT if it meets one of the following two conditions: (i) the object has not updated its DT in the past σ_2 consecutive time slots, or (ii) the value of η_i of object v_i is within top $\sigma_3 \cdot |V|$ among the objects. This procedure continues until all $\lfloor \sigma_1 \cdot L_j \rfloor$ subchannels are allocated or no object meets either of the conditions, where σ_2 is a given integer threshold with $\sigma_2 > 0$ and σ_3 is a given threshold with $\sigma_3 \in [0, 1]$. Note that at the end of time slot $t-1$, we calculate $\eta_i(t)$ and choose the DTs that need to be updated at time slot t, then we can conduct DT updates from the beginning of time slot t.

When a task is offloaded to a cloudlet with its requested DT, that cloudlet needs to allocate the requested amount of computing resource for the task. Therefore, the available computing resource in each cloudlet upon the arrival of a task is its residual computing capacity at that time slot.

Considering dynamic arrivals of task requests, a usage cost of computing resource is introduced to measure the computing resource consumption in each cloudlet, which in turn is used to guide task offloading. Specifically, when task request $u_k(t)$ arrives, its usage cost at cloudlet l is

$$\rho_l(k) = C'_l \cdot (\alpha^{1 - \frac{C_l(k)}{C'_l}} - 1), \tag{25}$$

where $C_l(k)$ is the residual computing capacity of cloudlet l before task request $u_k(t)$ is considered, and α is a tuning parameter with $\alpha > 1$ that models the impact of resource reduction on the resource usage cost. If task $u_(k)$ is offloaded to cloudlet l for processing, then $C_l(k+1) = C_l(k) - c_k(t)$, where $c_k(t)$ is the amount of computing resource demanded by task $u_k(t)$, and $C_l(1) = C'_l$ initially. The normalized computing resource usage cost of offloading task $u_k(t)$ to cloudlet l is

$$\gamma_l(k) = \frac{\rho_v(k)}{C'_l} = \alpha^{1 - \frac{C_l(k)}{C'_l}} - 1. \tag{26}$$

Since task requests may arrive during objects uploading their update data, we determine task offloading by utilizing the residual bandwidth on each AP. As the bandwidth capacity on each AP is limited, the bandwidth resource cost $\varphi_j(k)$ is used to capture the amount of bandwidth consumed by offloading task $u_k(t)$ through AP j, which is defined as

$$\varphi_j(k) = L'_j \cdot (\theta^{1 - \frac{L_j(k)}{L'_j}} - 1), \tag{27}$$

where L'_j is the number of remaining subchannels of AP j after scheduling DT updates, $L_j(k)$ is the number of remaining subchannels of AP j before task $u_k(t)$ is considered, and θ is a tuning parameter with $\theta > 1$. If task $u_k(t)$ uploads its task data through AP j, we have $L_j(k+1) = L_j(k) - 1$, and $L_j(1) = L'_j$ initially. The normalized bandwidth resource usage cost of offloading task $u_k(t)$ through AP j is

$$\tau_j(k) = \theta^{1 - \frac{L_j(k)}{L'_j}} - 1. \tag{28}$$

When user task $u_k(t)$ arrives, the algorithm checks whether the APs in $\mathcal{C}_k$ have remaining subchannels for the user to upload its task data and then transfer the data to cloudlet l hosting its requested DT_{r_k} with sufficient computing resource. Otherwise, task $u_k(t)$ will be processed locally if the residual computing/bandwidth resource is insufficient or one of the following conditions is met: (i) the utility gain obtained by offloading the task to a cloudlet is no greater than the utility gain of its local processing, i.e., $Q_k^{off} \leq Q_{k,0,0}$, where $Q_k^{off} = \max\{Q_{k,j,l}(t) \mid j \in \mathcal{C}_k(t), L_j(k) \geq 1, l = n(DT_{r_k(t)})\}$; or (ii) the normalized computing resource usage cost of cloudlet l hosting the requested DT of task $u_k(t)$ or the normalized bandwidth usage cost of AP j^* with $j^* = \arg\max\{Q_{k,j,l}(t) \mid j \in \mathcal{C}_k(t), L_j(k) \geq 1, l = n(DT_{r_k(t)})\}$ is greater than $|\mathcal{N}| \cdot Q_k^{off}$, i.e., $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$ or $\tau_j(k) > |\mathcal{N}| \cdot Q_k^{off}$. The detailed online algorithm is given in `Algorithm` 2.

Algorithm 2 Online algorithm for the dynamic utility maximization problem

Input: An MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, a set V of objects with their DTs deployed in $\mathcal{G}$, a time horizon $\mathbb{T}$, a set $U(t)$ of user tasks at each time slot $t \in \mathbb{T}$.
Output: Maximize the accumulative utility gain of user tasks within the time horizon $\mathbb{T}$.

$q \leftarrow 0$; /* the accumulative utility gain */
$V^{up} \leftarrow \emptyset$; /* set of objects to update their DTs */
$L_j^{res} \leftarrow L_j$ for each AP j; /* residual subchannels of AP j */
for each $t \in \mathbb{T}$ **do**
  Conduct DT updates for objects in V^{up};
  while task $u_k(t)$ arrives **do**
    if $C_l(k-1) < c_k(t)$ with l hosting $DT_{r_k(t)}$ or $\sum_{j \in \mathcal{C}_k} L_j^{res} = 0$ **then**
      Process task $u_k(t)$ locally, $q \leftarrow q + Q_{k,0,0}$;
    else
      Calculate Q_k^{off}, $\gamma_l(k)$, and $\tau_{j^*}(k)$;
      if $Q_k^{off} \leq Q_{k,0,0}$ or $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$ or $\tau_{j^*}(k) > |\mathcal{N}| \cdot Q_k^{off}$ **then**
        Process task $u_k(t)$ locally, $q \leftarrow q + Q_{k,0,0}$;
      else
        Offload task $u_k(t)$ through AP j^* to cloudlet l;
        $q \leftarrow q + Q_k^{off}$, $L_{j^*}^{res} \leftarrow L_{j^*}^{res} - 1$;
      end if
    end if
  end while
  Calculate $\eta_i(t+1)$ for each object $v_i \in V$;
  $V^{up} \leftarrow \emptyset$; /* determine objects to update their DTs at next time slot */
  for each AP $j \in \mathcal{N}$ **do**
    $L_j^{use} \leftarrow 0$;
    for each object under coverage of AP j **do**
      if DT_i has not been updated in the past σ_2 consecutive time slots or η_i is within the top $\sigma_3 \cdot |V|$ among all objects **then**
        $L^{use} \leftarrow L^{use} + 1$; $V^{up} \leftarrow V^{up} \cup \{v_i\}$;
      end if
      if $L^{use} = \lfloor \sigma_1 \cdot L_j \rfloor$ **then**
        Break;
      end if
    end for
    $L_j^{res} \leftarrow L_j - L_j^{use}$;
  end for
end for
return q.

5.2 Online algorithm for a special dynamic utility maximization problem

We here consider a special case of the dynamic utility maximization problem where each AP has abundant bandwidth to allow all devices under its coverage to upload simultaneously. Under this assumption, each object v_i can synchronize with its DT_i at each time slot, the service model of each DT can be retrained using the update data from its object.

The basic idea is similar to the generic one in the previous subsection. Specifically, the cumulative volume $\Phi_i(t)$ of the update data at DT_i is $\Phi_i(t) = \sum_{t'=1}^{t} vol_i(t')$, and the result accuracy of task execution on a service model of DT_i at time slot t is $f_i(\Phi_i(t-1))$. The utility $Q_{k,j,l}(t)$ of task offloading through AP j and executing at cloudlet l at each time slot t can be calculated by Eq. (11). When task request $u_k(t)$ arrives, the algorithm checks whether cloudlet l hosting its requested DT_{r_k} has sufficient computing resource for the execution of task $u_k(t)$. If there is no adequate computing resource in cloudlet l for task $u_k(t)$, the task must be processed locally. Otherwise whether the task will be offloaded to a cloudlet or processed locally is determined by the offloading control policy: task $u_k(t)$ is processed locally if (i) its utility gain by offloading it to any cloudlet l is no greater than the utility gain of its local processing, i.e., $Q_k^{off} \leq Q_{k,0,0}$, where $Q_k^{off} = \max\{Q_{k,j,l}(t) \mid j \in \mathcal{C}_k(t), l = n(DT_{r_k(t)})\}$; or (ii) the normalized computing resource usage cost of cloudlet l that hosts the requested DT of task $u_k(t)$ is greater than $|\mathcal{N}| \cdot Q_k^{off}$, i.e., $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$. The detailed algorithm for this special case is given in `Algorithm` 3.

Algorithm 3 Online algorithm for the special dynamic utility maximization problem

Input: An MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with APs having abundant bandwidth resource, a set V of objects with their DTs deployed in $\mathcal{G}$, a time horizon $\mathbb{T}$, a set $U(t)$ of user tasks at each time slot $t \in \mathbb{T}$.
Output: Maximize the accumulative utility gain of user tasks within the time horizon $\mathbb{T}$.

$q \leftarrow 0$ /* the accumulative utility gain */
while task $u_k(t) \in U(t)$ **do**
  if $C_l(k-1) < c_k(t)$ with cloudlet l hosting $DT_{r_k(t)}$ **then**
    $q \leftarrow q + Q_{k,0,0}$, task $u_k(t)$ is executed locally;
  else
    Calculate Q_k^{off} and $\gamma_l(k)$;
    if $Q_k^{off} \leq Q_{k,0,0}$ or $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$ **then**
      $q \leftarrow q + Q_{k,0,0}$, task $u_k(t)$ is executed locally;
    else
      $q \leftarrow q + Q_k^{off}$, offloading task $u_k(t)$ to cloudlet l;
    end if
  end if
end while
return q.

Remarks. The distinction between `Algorithm` 2 and `Algorithm` 3 lies in their applicability and theoretical guarantees. `Algorithm` 2 delivers an online solution for the generic dynamic utility maximization problem, while `Algorithm` 3 delivers a performance-guaranteed solution for a special case of the problem where each AP has sufficient bandwidth to allow all devices under its coverage for uploading at each time slot. In `Algorithm` 2, the update frequency of each DT is dynamically determined by bandwidth allocations and the overall system utility optimization. In both algorithms, the system continues serving user requests, using the most recent updated version of each service model during its DT update and model retraining at the current time slot. This ensures that task execution is not stalled by model update activities.

5.3 Algorithm analysis

In the following, we first analyze the time complexity of Algorithm 2 for the dynamic utility maximization problem, and then analyze the competitive ratio of Algorithm 3 for the special dynamic utility maximization problem.

Theorem 4. *Given a DT-assisted MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a finite time horizon $\mathbb{T}$, assume that user tasks arrive one by one without the knowledge of future arrivals over the given time horizon. There is an online algorithm,* Algorithm 2*, for the dynamic utility maximization problem, which deliver a solution to the problem, and the algorithm takes $O(|\mathcal{N}| \cdot |V| + |\mathcal{N}|^2 \cdot |U(t)|)$ time per time slot.*

Proof. As neither the computing resource capacity on any cloudlet nor the bandwidth resource capacity on any AP is violated in the solution delivered by Algorithm 2, the solution is feasible. The rest is to analyze the time complexity of Algorithm 2 as follows.

The algorithm consists of $|\mathbb{T}|$ time slots. At each time slot $t \in \mathbb{T}$, it takes $O(|\mathcal{N}| \cdot |V|)$ time to schedule the DT updates of objects in V through uploading their update data to APs in $\mathcal{N}$. For each user task $u_k(t) \in U(t)$, the time complexity of making the task offloading decision is dominated by the time complexity of calculating Q_k^{off}, while the latter takes $O(|\mathcal{N}|^2)$ time. The time complexity of Algorithm 2 thus is $O(|\mathcal{N}| \cdot |V| + |\mathcal{N}|^2 \cdot |U(t)|)$ per time slot. □

Lemma 2. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a finite time horizon $\mathbb{T}$, user tasks arrive one by one without the knowledge of future arrivals. Let $S(k)$ be the set of tasks offloaded by* Algorithm 3 *prior to task $u_k(t)$. Then, the total resource usage cost of all cloudlets prior to the arrival of $u_k(t)$ is*

$$\sum_{l \in \mathcal{N}} \rho_l(k) \leq 2 \cdot \log_2 \alpha \cdot |\mathcal{N}| \cdot \sum_{u_{k'}(t) \in S(k)} (c_{k'}(t) \cdot Q_{k'}^{off}), \quad (29)$$

where α is a constant with $2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2 \leq \alpha \leq 2^{\frac{C_{min}}{c_{\max}}}$, $C_{min} = \min\{C_l' \mid l \in \mathcal{N}\}$, $Q_{\max}^{off} = \max\{f_i(\Phi_i(|\mathbb{T}| - 1)) + \omega \mid v_i \in V\}$, and $c_{\max} = \max\{c_k(t) \mid u_k(t) \in \cup_{t \in \mathbb{T}} U(t)\}$.

Proof. If task $u_k(t)$ is processed locally, the resource usage of each cloudlet does not change. Otherwise, task $u_k(t)$ is offloaded to cloudlet l that hosts its requested DT $DT_{r_k(t)}$. Recall that $C_l(k)$ is the residual computing capacity on cloudlet l before task $u_k(t)$ is considered. Then, we have $C_l(k+1) = C_l(k) - c_k(t)$.

$$\begin{aligned}
&\rho_l(k+1) - \rho_l(k) \\
&= C_l' \cdot (\alpha^{1 - \frac{C_l(k+1)}{C_l'}} - 1) - C_l' \cdot (\alpha^{1 - \frac{C_l(k)}{C_l'}} - 1) \\
&= C_l' \cdot \alpha^{(1 - \frac{C_l(k)}{C_l'})} \cdot (2^{\frac{c_k(t)}{C_l'} \cdot \log_2 \alpha} - 1) \\
&\leq C_l' \cdot \alpha^{(1 - \frac{C_l(k)}{C_l'})} \cdot \frac{c_k(t)}{C_l'} \cdot \log_2 \alpha \quad (30) \\
&= \log_2 \alpha \cdot c_k(t) \cdot \alpha^{(1 - \frac{C_l(k)}{C_l'})}, \quad (31)
\end{aligned}$$

where Ineq. (30) holds due to that $2^x - 1 \leq x$ with $0 \leq x < 1$.

The difference of the resource usage cost of all cloudlets before and after offloading task $u_k(t)$ to cloudlet l is

$$\begin{aligned}
&\sum_{l' \in \mathcal{N}} (\rho_{l'}(k+1) - \rho_{l'}(k)) = \rho_l(k+1) - \rho_l(k) \\
&\leq \log_2 \alpha \cdot c_k(t) \cdot \alpha^{(1 - \frac{C_l(k)}{C_l'})}, \text{ by Eq. (31)} \\
&= \log_2 \alpha \cdot c_k(t) \cdot ((\alpha^{(1 - \frac{C_l(k)}{C_l'})} - 1) + 1) \\
&= \log_2 \alpha \cdot c_k(t) \cdot (\gamma_l(k) + 1) \\
&\leq \log_2 \alpha \cdot c_k(t) \cdot (|\mathcal{N}| \cdot Q_k^{off} + 1) \quad (32) \\
&\leq 2 \cdot \log_2 \alpha \cdot |\mathcal{N}| \cdot c_k(t) \cdot Q_k^{off}, \quad (33)
\end{aligned}$$

where Ineq. (32) holds since task $u_k(t)$ is offloaded to cloudlet l, by the offloading control policy.

The cost sum of resource usages of all cloudlets prior to the arrival of task $u_k(t)$ is

$$\begin{aligned}
&\sum_{l \in \mathcal{N}} \rho_l(k) = \sum_{k'=1}^{k-1} \sum_{l \in \mathcal{N}} (\rho_l(k'+1) - \rho_l(k')) \\
&= \sum_{u_{k'}(t) \in S(k)} \sum_{l \in \mathcal{N}} (\rho_l(k'+1) - \rho_l(k')) \\
&\leq \sum_{u_{k'}(t) \in S(k)} 2 \cdot \log_2 \alpha \cdot |\mathcal{N}| \cdot c_{k'}(t) \cdot Q_{k'}^{off}, \text{ by Eq. (33)} \\
&= 2 \cdot \log_2 \alpha \cdot |\mathcal{N}| \cdot \sum_{u_{k'}(t) \in S(k)} (c_{k'}(t) \cdot Q_{k'}^{off}).
\end{aligned}$$

The lemma then follows. □

Lemma 3. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a finite time horizon $\mathbb{T}$, user task requests arrive one by one without the knowledge of future arrivals. Denote by $M(k)$ the set of tasks offloaded to cloudlets for processing by an optimal solution but processed locally by* Algorithm 3 *prior to the arrival of task $u_k(t)$. For each task $u_k(t) \in M(k)$, we have $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$, where cloudlet l hosts $DT_{r_k(t)}$, $2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2 \leq \alpha \leq 2^{\frac{C_{min}}{c_{\max}}}$, and $c_{\max} = \max\{c_k(t) \mid u_k(t) \in \cup_{t \in \mathbb{T}} U(t)\}$.*

Proof. Since user task $u_k(t)$ is offloaded to a cloudlet for processing in the optimal solution, we must have $Q_k^{off} > Q_{k,0,0}$. Otherwise, the accumulative utility gain can be improved by processing $u_k(t)$ locally. This contradicts that the solution is optimal. We prove the claim by distinguishing into two cases: Case 1. There is no sufficient computing resource in cloudlet l to process task $u_k(t)$ in the solution by Algorithm 3, and thus task $u_k(t)$ has to be processed locally. Case 2. Although there is sufficient computing resource in cloudlet l to process task $u_k(t)$, offloading task $u_k(t)$ to cloudlet l meets the task offloading control policy.

Case 1. Since there is no sufficient computing resource in cloudlet l prior to the arrival of task $u_k(t)$ by Algorithm 3, i.e., $C_l(k) < c_k(t)$, we have

$$\begin{aligned}
\gamma_l(k) &= \alpha^{1 - \frac{C_l(k)}{C_l'}} - 1 > \alpha^{1 - \frac{c_k(t)}{C_l'}} - 1 \\
&\geq \alpha^{1 - \frac{1}{\frac{1}{\log_2 \alpha}}} - 1, \text{ since } \alpha \leq 2^{\frac{C_{min}}{c_{\max}}} \leq 2^{\frac{C_l'}{c_k(t)}} \\
&= \frac{\alpha}{2} - 1 \geq |\mathcal{N}| \cdot Q_{\max}^{off}, \text{ due to } \alpha \geq 2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2, \\
&\geq |\mathcal{N}| \cdot Q_k^{off}.
\end{aligned}$$

Case 2. Because $Q_k^{off} > Q_{k,0,0}$, condition (i) of control policy is not met. Processing task $u_k(t)$ locally by `Algorithm` 3 needs to meet condition (ii) of the task offloading control policy, namely, $\gamma_l(k) > |\mathcal{N}| \cdot Q_k^{off}$.

The lemma then follows. □

Lemma 4. *Given an MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a finite time horizon $\mathbb{T}$, assume that user task execution requests arrive one by one without the knowledge of future arrivals. Denote by $W(k)$ the set of tasks offloaded to the network so far sequentially prior to the arrival of task $u_k(t)$ by both the optimal solution and `Algorithm` 3. Denote by $\mathbb{Q}_{opt}(k)$ and $\mathbb{Q}(k)$ the cumulative utility gain achieved by an optimal solution and the solution delivered by `Algorithm` 3 prior to the arrival of the task $u_k(t)$ at time slot t, respectively. Then,*

$$\mathbb{Q}_{opt}(k) \leq \mathbb{Q}(k) + \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off}. \tag{34}$$

Proof. Denote by $H_{opt}(k)$ and $H(k)$ the sets of tasks that are processed locally in an optimal solution and in the solution of `Algorithm` 3 prior to the arrival of task $u_k(t)$, respectively. Denote by $H_1(k)$ the set of tasks processed locally by both the optimal solution and `Algorithm` 3 prior to the arrival of task $u_k(t)$, i.e., $H_1(k) = H_{opt}(k) \cap H(k)$. Denote by $H_2(k)$ the set of tasks that are offloaded to cloudlets for processing in the solution by `Algorithm` 3 but processed locally in an optimal solution prior to the arrival of task $u_k(t)$. Then, $H_2(k) = H_{opt}(k) \setminus H_1(k)$ and $H_2(k) \subseteq S(k) \setminus W(k)$. For the optimal solution, we have

$$\begin{aligned}
\mathbb{Q}_{opt}(k) &= \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off} + \sum_{u_{k'}(t) \in W(k)} Q_{k'}^{off} \\
&\quad + \sum_{u_{k'}(t) \in H_{opt}(k)} Q_{k',0,0} \\
&\leq \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off} + \sum_{u_{k'}(t) \in W(k)} Q_{k'}^{off} \\
&\quad + \sum_{u_{k'}(t) \in H_1(k)} Q_{k',0,0} + \sum_{u_{k'}(t) \in S(k) \setminus W(k)} Q_{k',0,0} \qquad (35) \\
&\leq \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off} + \sum_{u_{k'}(t) \in W(k)} Q_{k'}^{off} \\
&\quad + \sum_{u_{k'}(t) \in H_1(k)} Q_{k',0,0} + \sum_{u_{k'}(t) \in S(k) \setminus W(k)} Q_{k'}^{off} \qquad (36) \\
&\leq \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off} + \sum_{u_{k'}(t) \in S(k)} Q_{k'}^{off} \\
&\quad + \sum_{u_{k'}(t) \in H_1(k)} Q_{k',0,0} \\
&\leq \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off} + \sum_{u_{k'}(t) \in S(k)} Q_{k'}^{off} \\
&\quad + \sum_{u_{k'}(t) \in H(k)} Q_{k',0,0} \\
&= \mathbb{Q}(k) + \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off},
\end{aligned}$$

where Ineq. (35) holds due to $H_2(k) = H_{opt}(k) \setminus H_1(k)$ and $H_2(k) \subseteq S(k) \setminus W(k)$, and Ineq. (36) holds due to $Q_{k'}^{off} > Q_{k',0,0}$ if task $u_{k'}(t)$ is offloaded to the cloudlet by `Algorithm` 3. The lemma then follows. □

Theorem 5. *Given a DT-assisted MEC network $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, assuming that each AP has abundant bandwidth to enable all devices under its coverage to upload their update data simultaneously. For a given finite time horizon $\mathbb{T}$, assume that user tasks arrive one by one without the knowledge of future arrivals over the time horizon. There is an online algorithm, `Algorithm` 3 with a competitive ratio of $O(\log |\mathcal{N}|)$ for the special dynamic utility maximization problem, which takes $O(|\mathcal{N}|^2)$ time to process each task when $\alpha = 2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2$ and $Q_{\max}^{off} = \max\{f_i(\Phi_i(|\mathbb{T}| - 1)) + \omega \mid v_i \in V\}$.*

Proof. We analyze the competitive ratio of `Algorithm` 3 as follows. Assume that request $u_k(t)$ is currently being considered one, we have

$$\begin{aligned}
&|\mathcal{N}| \cdot (\mathbb{Q}_{opt}(k) - \mathbb{Q}(k)) \leq |\mathcal{N}| \cdot \sum_{u_{k'}(t) \in M(k)} Q_{k'}^{off}, \text{ by Lemma 4} \\
&= \sum_{u_{k'}(t) \in M(k)} |\mathcal{N}| \cdot Q_{k'}^{off} < \sum_{u_{k'}(t) \in M(k')} \gamma_l(k'), \text{ by Lemma 3} \\
&= \sum_{u_{k'}(t) \in M(k)} \frac{\rho_l(k')}{C'_l} \leq \sum_{l \in \mathcal{N}} \sum_{u_{k'}(t) \in M(k)} \frac{\rho_l(k)}{C'_l} \qquad (37) \\
&= \sum_{l \in \mathcal{N}} \rho_l(k) \sum_{u_{k'}(t) \in M(k)} \frac{1}{C'_l} \leq \sum_{l \in \mathcal{N}} \rho_l(k) \qquad (38) \\
&\leq 2 \cdot \log_2 \alpha \cdot |\mathcal{N}| \cdot \sum_{u_{k'}(t) \in S(k)} (c_{k'}(t) \cdot Q_{k'}^{off}), \text{ by Lemma 2,}
\end{aligned}$$

where Ineq. (37) holds, since the computing resource usage does not decrease, and Ineq. (38) holds due to the assumption that the residual computing resource on each cloudlet is no less than 1. Then, we have

$$\begin{aligned}
\mathbb{Q}_{opt}(k) &\leq \mathbb{Q}(k) + 2 \cdot \log_2 \alpha \sum_{u_{k'}(t) \in S(k)} (c_{k'}(t) \cdot Q_{k'}^{off} \\
&\leq \mathbb{Q}(k) + 2 \cdot \log_2 \alpha \cdot c_{\max} \sum_{u_{k'}(t) \in S(k)} Q_{k'}^{off} \qquad (39)
\end{aligned}$$

We then have

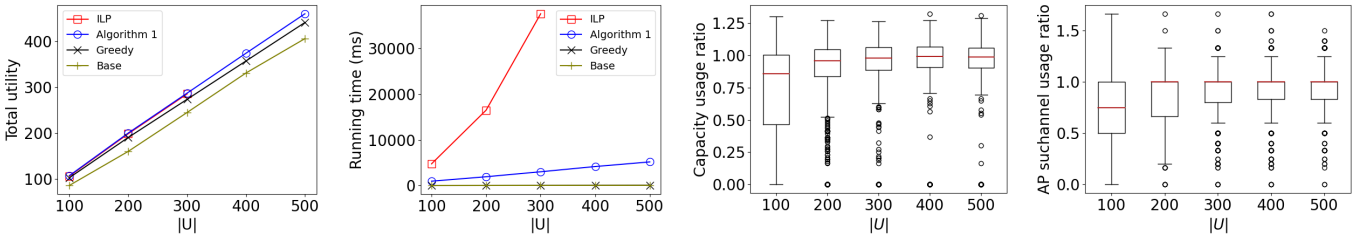
$$\begin{aligned}
\frac{\mathbb{Q}_{opt}(k)}{\mathbb{Q}(k)} &\leq \frac{\mathbb{Q}(k) + 2 \cdot \log_2 \alpha \cdot c_{\max} \sum_{u_{k'}(t) \in S(k)} Q_{k'}^{off}}{\mathbb{Q}(k)} \\
&\leq \frac{\mathbb{Q}(k) + 2 \cdot \log_2 \alpha \cdot c_{\max} \cdot \mathbb{Q}(k)}{\mathbb{Q}(k)} \\
&= 1 + 2 \cdot \log_2 \alpha \cdot c_{\max} = O(\log |\mathcal{N}|), \qquad (40)
\end{aligned}$$

where $\alpha = 2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2$.

It can be seen that the time complexity of `Algorithm` 3 is dominated by the time complexity of calculating Q_k^{off} for each task $u_k(t)$, which needs finding a shortest routing path in $\mathcal{G}$ between a cloudlet in $\mathcal{C}_k(t)$ and the cloudlet hosting $DT_{r_k(t)}$ that takes $O(|\mathcal{N}|^2)$ time. The theorem then follows. □

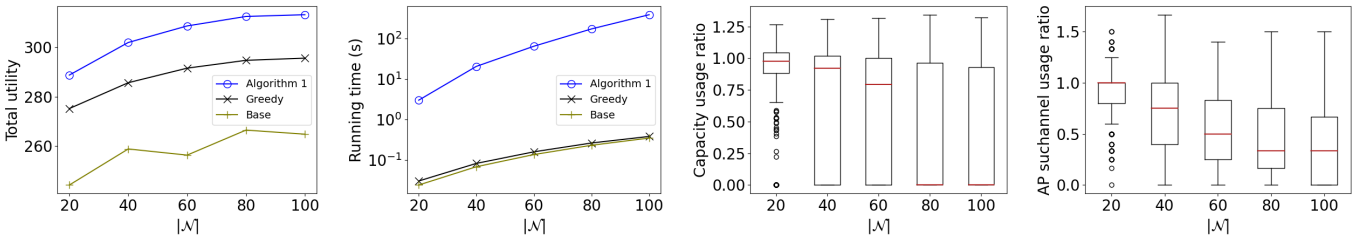
6 Performance Evaluation

In this section, we evaluated the performance of the proposed algorithms for the two investigated problems. Also, we studied the impacts of the major parameters on the performance of the proposed algorithms.



(a) Total utility (b) Running time (c) Computing resource usage by Algorithm 1 (d) Subchannel usage by Algorithm 1

Fig. 2. Performance of different algorithms for the utility maximization problem, by varying the number $|U|$ of user tasks.



(a) Total utility (b) Running time (c) Computing resource usage by Algorithm 1 (d) Subchannel usage by Algorithm 1

Fig. 3. Performance of different algorithms for the utility maximization problem by varying the network size $|\mathcal{N}|$.

6.1 Experimental settings

We considered an MEC network that consists of 20 APs with a co-located cloudlet for each AP, and the topology of the network is generated by NetworkX [36]. The computing capacities on cloudlets are randomly chosen in the range of $[1,000, 1,500]$ MHz [25]. The transmission delay of per MB data on each link is randomly drawn in the range of $[0.02, 0.05]$ ms [25]. The bandwidth capacity on each AP is uniformly distributed in the range of $[20, 40]$ MHz [2], and the number of subchannels on an AP ranges from 3 to 6, by adopting the OFDMA scheme. There are 100 user devices, and the task size of a task of each user device is randomly drawn in the range of $[1, 5]$ MB, and the amount of computing resource required by each user task is randomly drawn from the range of $[200, 400]$ MHz [2]. The service delay threshold of each task is randomly chosen in the range of $[3, 10]$ ms, and the value of β_k ranges from 1 to 3 [2]. The signal-to-noise ratio $\frac{PX_k \cdot H_{k,j}}{\sigma^2}$ between a user device and an AP is randomly drawn in the range of $[10, 30]$ dB [25]. The processing rate of a device is uniformly distributed in the range of $[0.5, 2]$ MB per ms [2]. The accuracy of a service model installed in a device for local processing is within the range of $[0.1, 0.6]$. There are 50 objects and the volume of data generated by each object at each time slot is within the range of $[1, 5]$ MB [13]. A submodular non-decreasing function $f(x) = \log_2(\frac{x}{40} + 1)$ in Eq. (2) will be adopted for our purpose [37], [38], [39]. The processing rates of service models provided by DTs are uniformly distributed in the range of $[1, 3]$ MB per millisecond [2]. The value of ω in Eq. (11) is set at 0.5, and the values of α and θ are set at $2 \cdot |\mathcal{N}| \cdot Q_{\max}^{off} + 2$. The values of σ_1, σ_2, and σ_3 are set to 0.4, 3, and 0.3, respectively. Assume that $\mathbb{T}$ consists of 50 time slots and the duration of each task occupying computing resource in cloudlet is randomly chosen in the range of $[1, 3]$ time slots.

To evaluate `Algorithms` 1, a baseline algorithm `Base` makes its task offloading decision for each user task randomly. Also, a greedy algorithm `Greedy` always admits the user task with the maximum utility among the tasks to be considered, and each task is either executed locally or offloaded to a cloudlet that hosts its requested DT in each iteration.

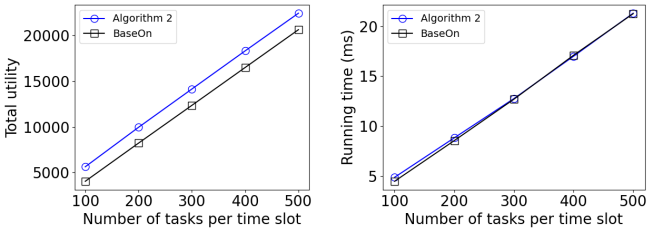
To evaluate `Algorithm` 2, the algorithm `BaseOn` is used to choose objects randomly to update their DTs and to make the task-offloading decision for each user task randomly.

To evaluate `Algorithm` 3, an online version of `Base`, refereed to as `BaseSp`, is adopted, which applies `Base` at each time slot.

The value in each figure is the average of the results of 50 network instances with the same size. The running time of each algorithm is based on a desktop equipped with an Intel(R) Xeon(R) Platinum 8280 2.70GHz CPU, with 10GB RAM. The parameters are adopted in the default settings unless otherwise specified.

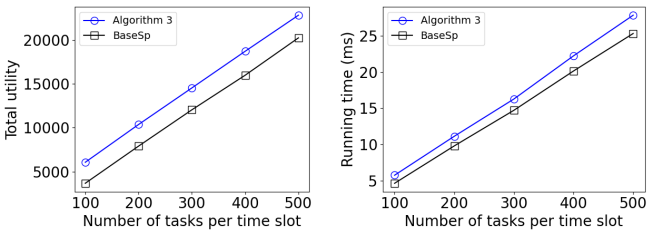
6.2 Performance of different algorithms for the utility maximization problem

We first evaluated the mentioned algorithms for the utility maximization problem, by varying the number of user task execution requests. Figs. 2(a) and 2(b) display the total utilities and running times of different algorithms, demonstrating that the utility delivered by `Algorithm` 1 is close to the optimal one by the ILP within a moderate



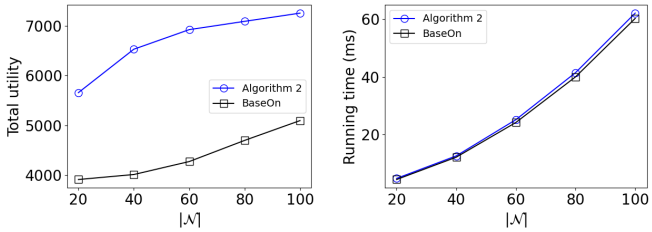
(a) Total utility (b) Running time

Fig. 4. Performance of different algorithms for the dynamic utility maximization problem, by varying the number of incoming tasks per time slot.



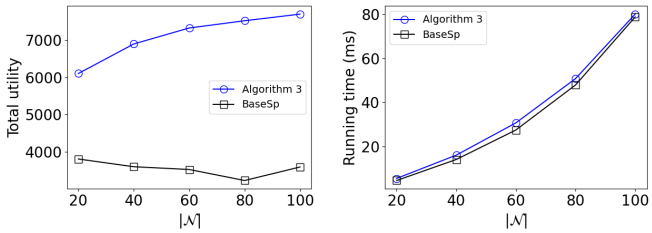
(a) Total utility (b) Running time

Fig. 5. Performance of algorithms for the special dynamic utility maximization problem, by varying the number of incoming tasks per time slot.



(a) Total utility (b) Running time

Fig. 6. Performance of different algorithms for the dynamic utility maximization problem by varying the network size $|\mathcal{N}|$.



(a) Total utility (b) Running time

Fig. 7. Performance of algorithms for the special dynamic utility maximization problem by varying the network size $|\mathcal{N}|$.

running time. It can be seen from Fig. 2(b) that with the increase in the number of tasks, the running time of the ILP grows rapidly, which indicates that it is not scalable when the problem size is large. Fig. 2(c) and 2(d) depict the resource capacity violation ratios of cloudlets and APs, respectively.

We then investigated the impact of network size $|\mathcal{N}|$ on the performance of `Algorithm 1`, `Greedy`, and `Base` while fixing the number of task requests at 300. Fig. 3 shows the performance results. It can be seen from Fig. 3(a) that `Algorithm 1` outperforms `Greedy` and `Base` by at least 5% and 16%, respectively. Fig. 3(c) and Fig. 3(d) depict that the median resource usages of cloudlets and APs decrease, with the increase on network size. This is because a larger network contains more cloudlets and APs, implying that the network is equipped with more computing and bandwidth resources for services.

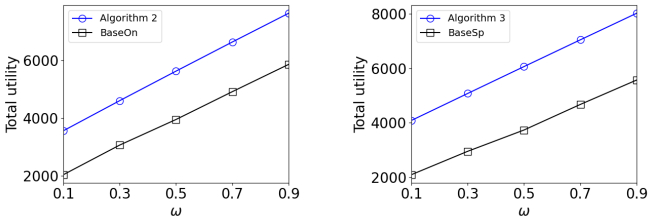
6.3 Performance of different algorithms for the dynamic utility maximization problem

We then investigated the performance of `Algorithm 2` against `BaseOn`, and `Algorithm 3` against `BaseSp` by varying the number of task requests per time slot. Figs. 4 and 5 depict the total utilities and running times of different algorithms for the (special) dynamic utility maximization problem. Fig. 4(a) shows that `Algorithm 2` outperforms `BaseOn` by at least 8%, which demonstrates that `Algorithm 2` schedules DT updates and task offloading more intelligently compared to `BaseOn` under the limited bandwidth constraint. Fig. 5(a) shows that `Algorithm 3` outperforms `BaseSp` by at least 12%, which demonstrates the effectiveness of the task offloading control policy.

We evaluated the impacts of parameters on the performance of the proposed algorithms for the dynamic utility maximization problem with the bandwidth constraints on APs including $|\mathcal{N}|$, ω, σ_1, and σ_3, where ω is a scaling factor that strives for a fine balance between the service accuracy and the service delay, σ_1 reflects the maximum amount of resources on each AP for DT updates, and σ_3 reflects the maximum number of objects that can update their DTs. The number of incoming tasks is fixed at 100 per time slot in the rest of experiments.

Fig. 6 and 7 plot the total utility and running time curves of `Algorithm 2`, `BaseOn`, `Algorithm 3`, and `BaseSp` for the problem by varying the network size $|\mathcal{N}|$ with and without bandwidth constraints. It can be seen from Fig. 6(a) that `Algorithm 2` outperforms `BaseOn` by at least 42% for the given network size. This demonstrates the effectiveness of scheduling DT updates and task offloading by `Algorithm 2` under the bandwidth constraints. It can also be seen from Fig. 7(a) that the total utility by `Algorithm 3` increases with the growth on the network size, because a larger network has more computing resource for more task executions. Without the bandwidth constraints on APs, all objects can upload their updates to their DT service models at each time slot, and their service models are very likely to be retrained to obtain more accurate services, compared with those obtained by local processing only.

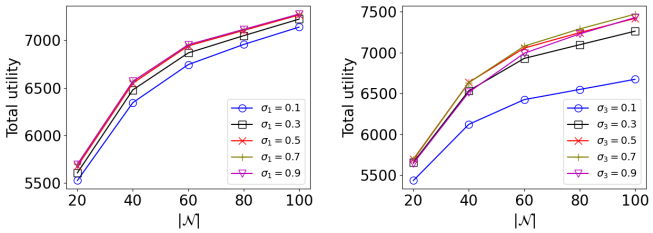
Fig. 8 plots the total utility curves by different algorithms through varying the value of ω. It can be seen from Fig. 8(a) and 8(b) that `Algorithm 2` outperforms `BaseOn` by at least 30%, and `Algorithm 3` outperforms `BaseSp` by at least 44% for a given ω, respectively. It is observed that the total utility by each comparison algorithm increases, with the growth of the value of ω. This indicates that the service delay is more important than the service accuracy in terms of the utility to express user satisfaction with a service for given setting.



(a) Performance of different algorithms for the dynamic utility maximization problem

(b) Performance of different algorithms for the special dynamic utility maximization problem

Fig. 8. The impact of ω on the performance of proposed algorithms.



(a) Impact of σ_1

(b) Impact of σ_3

Fig. 9. Impacts of σ_1 and σ_3 on the performance of `Algorithm 2` for the dynamic utility maximization problem.

Fig. 9(a) and 9(b) illustrate impacts of σ_1 and σ_3 on the performance of `Algorithm 2`, respectively. It can be seen from Fig. 9(a) that for each given network size, the total utility by `Algorithm 2` increases when the value of σ_1 increases from 0.1 to 0.7, while the total utility when $\sigma_1 = 0.7$ is close to the one when $\sigma_1 = 0.9$. This is because a larger σ_1 means that more subchannels are reserved for DT updates. With the increase on the value of σ_1, more objects can update their DTs and then improve the accuracy of their service models through model retraining, resulting in a larger total utility. Since the number of objects that can update their DTs at each time slot is determined by σ_3, the total utility will not increase indefinitely. When the network size is set at 60, Fig. 9(b) indicates that the total utility by `Algorithm 2` increases first when the value of σ_3 increases from 0.1 to 0.7, and then decreases when the value of σ_3 increases from 0.7 to 0.9. The rationale behind this is that a larger σ_3 means that more objects can update their DTs, leading to a higher accuracy of their service models. However, when more and more objects upload their update data to their DTs, less numbers of task requests can be offloaded to cloudlets hosting their requested DTs for executions, due to limited bandwidth on each AP. This results in the reduction of the total utility when $\sigma_3 = 0.9$.

7 Conclusion

In this paper, we studied QoE-aware task execution on inference service models in a DT-assisted MEC network, through either offloading tasks to cloudlets in the MEC network or processing the tasks locally. A novel utility metric to measure the QoE of user satisfaction on services is proposed, which jointly considers service delay and service accuracy of each task execution. Specifically, We formulated two novel QoE-aware utility maximization problems: the (static) utility maximization problem, and the dynamic utility maximization problem, respectively. For the former, we formulated an ILP solution when the problem size is small; otherwise we devised a randomized algorithm with high probability. For the latter, we devised an online algorithm with a provable competitive ratio for a special case of the problem if bandwidth constraints on APs are ignored, otherwise, we proposed a generic online algorithm without performance-guarantees. Finally, we evaluated the performance of the proposed algorithms through simulations. The simulation results demonstrate that the proposed algorithms are promising.

Acknowledgment

The authors appreciate for the three anonymous referees and the Associate Editor for their constructive comments and invaluable suggestions, which have help us to improve the quality and presentation of the paper greatly. The work by Weifa Liang was supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China with grant No: CityU 11202723, CityU 11202824, 7005845, 8730094, 9380137, and the CRF grant No: C1042-23GF, respectively.

References

[1] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[2] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, and J. Zhao, "Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1199–1212, 2022.

[3] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 7, pp. 2025–2039, 2023.

[4] X. Chen, J. Cao, Y. Sahni, M. Zhang, Z. Liang, and L. Yang, "Mobility-aware dependent task offloading in edge computing: a digital twin-assisted reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 24, no. 4, pp. 2979–2994, 2025.

[5] Y. Yang, Y. Shi, C. Yi, J. Cai, J. Kang, D. Niyato, and X. Shen, "Dynamic human digital twin deployment at the edge for task execution: A two-timescale accuracy-aware online optimization," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 262 – 12 279, 2024.

[6] J. Yu, A. Y. Alhilal, T. Zhou, P. Hui, and D. H. Tsang, "Attention-based qoe-aware digital twin empowered edge computing for immersive virtual reality," *IEEE Transactions on Wireless Communications*, vol. 23, no. 9, pp. 11 276–11 290, 2024.

[7] D. Gupta, S. S. Moni, and A. S. Tosun, "Integration of digital twin and federated learning for securing vehicular internet of things," in *Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems*, 2023, pp. 1–8.

[8] Y. Zhang, W. Liang, Z. Xu, W. Xu, and M. Chen, "AoI-aware inference services in edge computing via digital twin network slicing," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3154–3170, 2024.

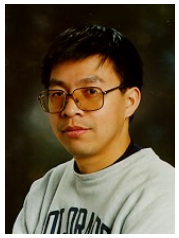
[9] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Z. Xu, and W. Xu, "AoI-aware user service satisfaction enhancement in digital twin-empowered edge computing," *IEEE/ACM Transactions on Networking*, vol. 32, no. 2, pp. 1677–1690, 2024.

[10] J. Li, S. Guo, W. Liang, J. Wu, Q. Chen, Z. Xu, W. Xu, and J. Wang, "AoI-aware, digital twin-empowered IoT query services in mobile edge computing," *IEEE/ACM Transactions on Networking*, 2024.

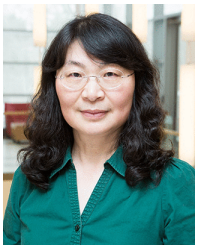
[11] X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia, "Multiple service model refreshments in digital twin-empowered edge computing," *IEEE Transactions on Services Computing*, vol. 17, no. 5, pp. 2672–2686, 2024.

[12] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya, "Digital twin-enabled service satisfaction enhancement in edge computing," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.
[13] Y. Zhang, W. Liang, W. Xu, Z. Xu, and X. Jia, "Cost minimization of digital twin placements in mobile edge computing," *ACM Transactions on Sensor Networks*, vol. 20, no. 3, pp. 1–26, 2024.
[14] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017.
[15] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, 2016.
[16] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, and M. Guizani, "Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite iot," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7783–7795, 2023.
[17] R. Zhang, R. Zhou, Y. Wang, H. Tan, and K. He, "Incentive mechanisms for online task offloading with privacy-preserving in UAV-assisted mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2646 – 2662, 2024.
[18] Y. Chen, J. Xu, Y. Wu, J. Gao, and L. Zhao, "Dynamic task offloading and resource allocation for noma-aided mobile edge computing: An energy efficient design," *IEEE Transactions on Services Computing*, 2024.
[19] X. Dai, Z. Xiao, H. Jiang, and J. C. Lui, "UAV-assisted task offloading in vehicular edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2520–2534, 2024.
[20] Y. Ma, W. Liang, Z. Xu, and S. Guo, "Profit maximization for admitting requests with network function services in distributed clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 5, pp. 1143–1157, 2019.
[21] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu, "Energy-aware inference offloading for DNN-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2021.
[22] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput maximization of delay-aware dnn inference in edge computing by exploring dnn model partitioning and inference parallelism," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 3017–3030, 2023.
[23] S. S. Ogden and T. Guo, "Mdinference: Balancing inference accuracy and latency for mobile applications," in *2020 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2020, pp. 28–39.
[24] J. Gu, Y. Fu, and K. Hung, "On intelligent placement decision-making algorithms for wireless digital twin networks via bandit learning," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 6, pp. 8889–8902, 2024.
[25] Y. Zhang, W. Liang, Z. Xu, and X. Jia, "Mobility-aware service provisioning in edge computing via digital twin replica placements," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 295 – 11 311, 2024.
[26] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1427–1444, 2022.
[27] B. Cao, Z. Li, X. Liu, Z. Lv, and H. He, "Mobility-aware multiobjective task offloading for vehicular edge computing in digital twin environment," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3046–3055, 2023.
[28] S. Li, S. Li, Y. Sun, B. Wang, B. Wang, and B. Zhang, "Digital twin-assisted computation offloading and resource allocation for multi-device collaborative tasks in industrial internet of things," *IEEE Transactions on Network Science and Engineering*, pp. 1–16, 2025.
[29] H. Guo, X. Zhou, J. Wang, J. Liu, and A. Benslimane, "Intelligent task offloading and resource allocation in digital twin based aerial computing networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3095–3110, 2023.
[30] Y. Zhang, J. Hu, and G. Min, "Digital twin-driven intelligent task offloading for collaborative mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3034–3045, 2023.
[31] L. Zhao, Z. Zhao, E. Zhang, A. Hawbani, A. Y. Al-Dubai, Z. Tan, and A. Hussain, "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3386–3400, 2023.
[32] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, no. 4, pp. 162–166, 2006.
[33] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.
[34] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," *Journal of the ACM*, vol. 68, no. 1, pp. 1–39, 2021.
[35] J. Liu, G. Zhao, H. Xu, P. Yang, B. Wang, and C. Qiao, "Toward a service availability-guaranteed cloud through VM placement," *IEEE/ACM Transactions on Networking*, vol. 32, no. 5, pp. 3993 – 4008, 2024.
[36] Networkx. https://networkx.org/. Accessed: Jul., 2022.
[37] J. Li, W. Liang, J. Wang, and X. Jia, "Accumulative fidelity maximization of inference services in DT-assisted edge computing," in *2024 International Conference on Meta Computing (ICMC)*. IEEE, 2024, pp. 64–73.
[38] J. Li, J. Wang, W. Liang, X. Jia, and A. Y. Zomaya, "Inference service fidelity maximization in DT-assisted edge computing," *IEEE Transactions on Mobile Computing*, 2025, doi: 10.1109/TMC.2025.3600390.
[39] X. Ai, W. Liang, Y. Zhang, and W. Xu, "Fidelity-aware inference services in DT-assisted edge computing via service model retraining," *IEEE Transactions on Services Computing*, vol. 18, no. 4, pp. 2089–2102, 2025.

Yuncan Zhang (Member, IEEE) received the Ph.D degree from Kyoto University, Kyoto, Japan, in 2021. She now is an associate professor at Sun Yat-Sen University, China. She was a Post-Doc in the Department of Computer Science at City University of Hong Kong. Her research interests include edge computing, digital twin, network function virtualization, and optimization problems.



Weifa Liang (Senior Member, IEEE) received his PhD degree in computer science from the Australian National University in 1998. He is a Full Professor in the Department of Computer Science at City University of Hong Kong. Prior to that, he was a Full Professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Mobile Edge Computing (MEC), Internet of Things and digital twins, machine learning, design and analysis of parallel and distributed algorithms, approximation algorithms, and graph theory. He currently serves as an Editor of IEEE Transactions on Communications.



Yuanyuan Yang (Life Fellow, IEEE) received the B.Eng. and M.S. degrees in computer science and engineering from Tsinghua University, Beijing, China, and the M.S.E. and Ph.D. degrees in computer science from Johns Hopkins University, Baltimore, MD, USA. She is currently a SUNY Distinguished Professor of computer engineering and computer science with Stony Brook University, Stony Brook, NY, USA. She was on leave with the National Science Foundation as the Program Director from 2018-2022. Her research interests include quantum computing, edge computing, data center networks, cloud computing, and wireless networks. She has published more than 530 papers in major journals and refereed conference proceedings and holds seven U.S. patents in these areas. She is currently an Associate Editor for *IEEE Transactions on Parallel and Distributed Systems* and *ACM Computing Surveys*. She has served as the Editor-in-Chief for *IEEE Transactions on Cloud Computing*, the Associate Editor-in-Chief and an Associate Editor for *IEEE Transactions on Computers*, and an Associate Editor for *IEEE Transactions on Parallel and Distributed Systems*. She has also served as the general chair, program chair, or vice chair for several major conferences, and a program committee member for numerous conferences.