Fidelity-Aware Inference Services in DT-Assisted Edge Computing via Service Model Retraining

Xuan Ai, Weifa Liang, *Senior Member, IEEE*, Yuncan Zhang, *Member, IEEE*, and Wenzheng Xu, *Member, IEEE*

***Abstract*—The Digital Twin (DT) technique enables seamless integrations between the physical and virtual worlds. By continuously synchronizing DTs with their physical counterparts, DTs can provide accurate reflections of physical objects and facilitate high-fidelity inference services based on service models. Orthogonal to the DT technology, Mobile Edge Computing (MEC) has been envisioning as a promising paradigm for providing intelligent services to users while meeting stringent delay and accuracy requirements. In this paper, we investigate fidelity-aware inference services in a DT-assisted MEC network where there are multiple source DTs providing new updated training data to service models often. We jointly schedule mobile devices to upload their update data to their DTs, and choose service models for retraining using their updated source DT data over a given time horizon. We further assume that the previous version of each service model can still serve its users during its retraining period, while a retrained service model can provide high-fidelity services to its users. To this end, we first formulate two novel optimization problems: the model instance placement problem that assigns model instances to cloudlets in an MEC network so that the total placement cost of all service models is minimized, and the cumulative utility maximization problem to maximize the cumulative fidelity of all service models over a given time horizon, by jointly scheduling mobile devices to upload their update data to their DTs and service models to be trained using their updated source DT data at each time slot. We then formulate an integer linear programming (ILP) solution for the model instance placement problem when the problem size is small; otherwise we develop an approximate solution to the problem, at the expense of moderate resource violations. We also devise an efficient online algorithm for the cumulative utility maximization problem. We finally evaluate the performance of the proposed algorithms via simulations, and the simulation results demonstrate that the proposed algorithms are promising.**

***Index Terms*—DT-empowered service models, mobile edge computing (MEC), model instance placement, choosing mobile devices for uploading, model retraining, inference service fidelity, algorithm design.**

Received 15 February 2025; revised 30 June 2025; accepted 2 July 2025. Date of publication 4 July 2025; date of current version 8 August 2025. The work of Weifa Liang was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant CityU 11202723, Grant CityU 11202824, Grant 7005845, Grant 8730094, and Grant 9380137, and in part by the CRF under Grant C1042-23GF. The work of Wenzheng Xu was supported in part by NSFC under Grant 62272328 and in part by Sichuan Science and Technology Program under Grant 2024NSFJQ0026. *(Corresponding author: Weifa Liang.)*

Xuan Ai and Weifa Liang are with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: xuanai2-c@my.cityu.edu.hk; weifa.liang@cityu.edu.hk).

Yuncan Zhang is with the School of Software Engineering, Sun Yat-Sen University, Zhuhai 519082, China (e-mail: zhangyc29@mail.sysu.edu.cn).

Wenzheng Xu is with the College of Computer Science, Sichuan University, Chengdu 610000, China (e-mail: wenzheng.xu@scu.edu.cn).

Digital Object Identifier 10.1109/TSC.2025.3586126

I. Introduction

EMERGING as a bridge connecting the physical and virtual spaces, Digital Twin (DT) technology enables seamless bidirectional communication by creating precise virtual representations of physical objects, and dynamically evolving with physical objects throughout their life cycles [19]. With explosive volume of data generated by IoT devices, lots of efforts have been dedicated to harnessing the immense business value of the data generated [16], [28], [31]. By applying Machine Learning (ML) and Artificial Intelligence (AI) approaches, insightful analysis on DTs can provide comprehensive interpretations and behavior predictions of their physical counterparts [19], thereby helping human decision-making. To support real-time virtue reflections, it is essential to maintain DTs through continuous synchronizations with their physical objects. Furthermore, the state freshness of DTs, in a certain extent, reflects the gap between the cyber-physical integration, which can be measured by the Age of Information (AoI) [8], [14] that is the duration of a piece of data from its generation to its first usage.

DT has attracted massive attention in boosting edge intelligence (EI), which is a convergence of AI and advanced data processing capabilities directly at the edge of core networks [2]. To implement high-fidelity intelligent services based on inference machine learning models, it is critical to retrain the machine learning models continuously, using real-time update data of their DT sources. The source data of an inference model comes from its physical objects, and each DT is usually regarded as a ”sandbox” that facilitates the encapsulation of newly generated information [11], [23]. Meanwhile, Mobile Edge Computing (MEC) has been emerging as a promising paradigm for the promotion of DT-assisted edge intelligence, by pushing communication and computing resources in the proximity of end devices [1], [9], [15]. Consequently, DT-assisted MEC is an ideal platform to provide users with intelligent services while meeting their stringent service delay requirements. An illustrative example of intelligent services in a DT-assisted MEC is smart transportation services based on ML models [2]: a smart transportation service model is built upon DTs of nearby vehicles and roadside sensors, where the DTs of each vehicle and each sensor continuously synchronize with their physical counterparts to monitor the traffic conditions in a real-time manner. Due to fast-changing traffic conditions, the smart transportation service model needs to be retrained, using the updated DT data, in order to provide users with traffic conditions and to mitigate traffic congestion. Furthermore, there are also studies dealing with edge server deployments in an intelligent Internet of Vehicles ecosystem, by jointly considering service delays, energy consumptions, and workload balancing [24].

In this paper, we study fidelity-aware inference services in a DT-assisted MEC network, where service model instances are placed and retrained. Assume that DTs of mobile devices that feed service models with training data have already been deployed, and these DTs are referred to as the data sources of the service models. To provide high-fidelity inference services, each service model needs to be continuously trained, using the update data from its source DTs.

Most existing studies on model retraining focused solely on improving service accuracy, and assumed that model retraining can be finished within a single time slot, while overlooking the impact on service provisioning caused by the multi-time slot model retraining process [9], [14], [25], [29]. In most real-world applications, a previous version of a service model can still serve its users while the model is under the updating or maintenance status, and the updated service model with a higher service fidelity will be available after finishing its retraining. However, both model training and user services consume computing resource in a resource-limited MEC network. This results in a dilemma between the amount of computing resource used for model retraining to achieve a higher inference accuracy in future and the amount of computing resource used for model-driven user services at this moment. To achieve the cumulative service fidelity of all service models, it poses the following challenges through exploring non-trivial trades-off between model service provisioning and model retraining. First, to enhance service fidelity of a service model, the model is required to be continuously retrained using the update data of its source DTs. It is crucial to have a novel metric that can capture the impact of the update data from its DT sources on the service fidelity of a service model. Second, since both model-driven inference services and model retraining consume computing and communication resources, the placement location of a model instance impacts its retaining cost as the update data from its DT sources requires to be routed to the model instance host for its retraining. How to place model instances to proper locations (cloudlets) so that the overall cost of model instance placements is minimized? Third, a service model retraining may stretch multiple time slots. Although the service fidelity of the model will be enhanced after the retraining, the computing and communication resource consumptions during its retraining period (time slots) cannot be ignored. Thus, choosing which service models for retraining at each time slot is important in order to maximize the overall service fidelity of all models. Finally, due to limited communication resource capacity on each access point (AP) in an MEC network, not all mobile devices under the coverage of each AP can upload their update data via the AP at each time slot to synchronize with their DTs. Then, which mobile devices should be chosen for uploading their update data at each time slot to maximize the cumulative service fidelity of all models? The rest of this paper will tackle the aforementioned challenges.

The novelties of this study lie in formulating a novel cumulative utility maximization problem in a DT-assisted MEC network, by exploring non-trivial trades-off between high-fidelity service provisioning and model retraining at each time slot, through scheduling mobile devices for DT synchronization and service models for retraining. Efficient algorithms for placing model instances and scheduling models for retraining are devised and thoroughly analyzed.

The main contributions of the paper are given as follows.

- We study fidelity-aware inference services in a DT-assisted MEC network, by introducing a novel metric to measure the impact of the DT update data on the inference accuracy of service models, where the training data of each service model comes from its source DTs, while the update data of each DT is obtained through synchronizations with its corresponding mobile device. We assume that each service model is required to be retrained continuously, using the update data from its multiple DT sources in order to maintain high-fidelity service.
- We consider the initial placement problem of service models by placing service model instances to proper cloudlets to minimize the total placement cost, for which we formulate an Integer Linear Programming (ILP) solution if the problem size is small; otherwise we propose a constant approximation algorithm.
- We deal with the novel cumulative utility maximization problem over a given time horizon, with the aim to maximize the cumulative utility (service accuracy) of all service models, subject to computing and communication resource capacities in the MEC network. We develop an online algorithm for the problem, by choosing mobile devices to synchronize with their DTs and scheduling service models for retraining using their updated source DT data at each time slot.
- We evaluate the performance of the proposed algorithms through simulations. Experimental results indicate that the proposed algorithms are promising and outperform their comparison baselines.

The remainder of the paper is organized as follows. Section II reviews related work on this topic. Section III introduces the system model, notions, notations, and problem definitions. Section IV provides an ILP solution for the model instance placement problem when the problem size is small; otherwise, an approximation algorithm is developed. Section V proposes an online algorithm for the cumulative utility maximization problem. Section VI evaluates the performance of the proposed algorithms for the defined problems through simulations, and Section VII concludes the paper.

II. Related Work

Digital twin, as an emerging technique, has attracted lots of attention across various fields, including manufacturing, personalized healthcare, autonomous driving, the Internet of Things, precision agriculture, smart education, Metaverse (AR/VR entertainment), and so on. Most existing studies explored diverse services driven by DTs [5], [7], [8], [16], [20]. For instance, Gong et al. [5] investigated the integration of space-air-ground and digital twin, with the aim to reduce the gap between data analysis and physical status through computation offloading. Li et al. [8] studied DT placements to improve user satisfaction on services in MEC environments by developing performance-guaranteed approximation algorithms for maximizing cumulative user satisfaction on services. Li et al. [7] investigated a DT-assisted MEC network, where users request SFC-enabled reliable services under both static and dynamic request arrival settings. They devised efficient algorithms to minimize the service cost in the static case and to maximize the number of service request admissions in the dynamic case, respectively, as DTs can be used to predict the reliability of VNF instances accurately by utilizing the historical traces of VNFs maintained in their DTs. Lin et al. [16] developed a congestion control policy based on incentives to meet the spatio-temporal dynamic DT

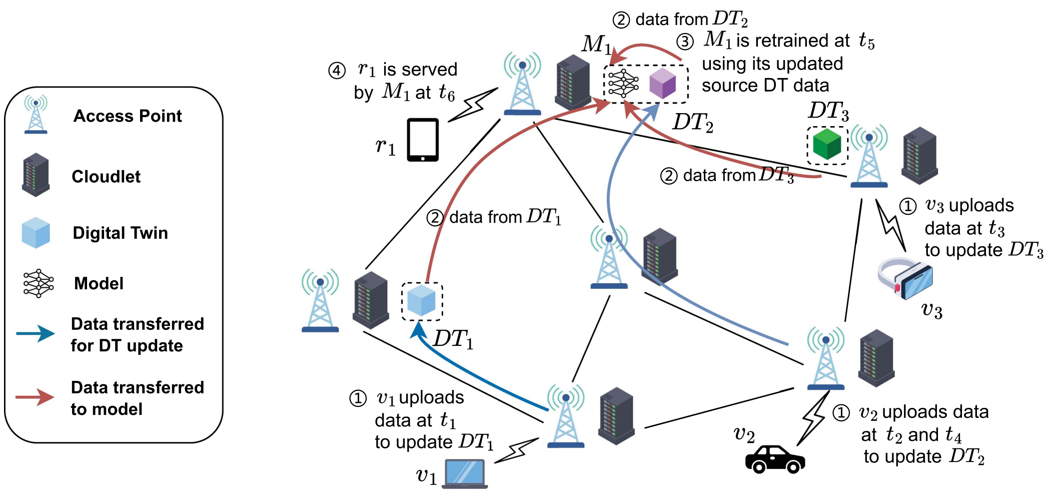

Access Point
Cloudlet
Digital Twin
Model
Data transferred for DT update
Data transferred to model
② data from DT_2
M_1
③ M_1 is retrained at t_5 using its updated source DT data
DT_2
④ r_1 is served by M_1 at t_6
r_1
DT_3
② data from DT_3
① v_3 uploads data at t_3 to update DT_3
v_3
② data from DT_1
DT_1
① v_1 uploads data at t_1 to update DT_1
v_1
v_2
① v_2 uploads data at t_2 and t_4 to update DT_2

Fig. 1. An illustrative example of model-driven services in a DT-assisted MEC network. There are three mobile devices v_1, v_2, and v_3 under the coverages of three different APs, and for each mobile device v_i with $1 \leq i \leq 3$, its digital twin, DT_i, has been deployed in a cloudlet in the network. We further assume that the source data of service model M_1 come from DT_1, DT_2, and DT_3, respectively. ①Assume that v_i uploads its update data to the AP at which it is located at time slot t_i, and the uploaded update data then is transferred to the cloudlet host $h(DT_i)$ of DT_i at time slot t_i with $1 \leq i \leq 3$. Notice that v_2 has another uploading at later time slot t_4 as well. ②Let $(l-1)$ be the last retraining of model M_1 ending at time slot t_0, and let the lth retraining of model M_1 start at time slot t_5 . ③ The updated source data at DT_1, DT_2, and DT_3 are transferred from their cloudlet hosts $h(DT_1)$, $h(DT_2)$, and $h(DT_3)$ to the cloudlet host $h(M_1)$ of model M_1 for its retraining that starts at time slot t_5 . ④After model M_1 finishes its retraining at time slot $t_5 + I_{m,l}$, user r_1 can request services of model M_1 at time slot t_6 with $t_6 > t_5 + I_{m,l}$, where $t_0 \leq t_1 \leq t_2 \leq t_3, t_2 < t_4$, and $t_0 \leq t_1 \leq t_3 \leq t_4 < t_5 < t_6$.

service demands of mobile users, and utilized the Lyapunov optimization technique to achieve the long-term stability of DT services. Wang et al. [20] designed a data collection scheme for the digital twin network, which can guarantee the timeliness and accuracy of collected data to build accurate DT mappings. Zhang et al. [26], [27], [28] considered DT-assisted service provisioning in an MEC network via DT replica placements and migrations, assuming that objects are highly movable in the network. However, the aforementioned studies focused solely on digital twin technology, neglecting its broader potential, such as its capability to promote the development of edge intelligence.

Several studies on DT-assisted intelligent services in MEC networks have been proposed, and DT has also been regarded as an enabling support technique for edge intelligence. For example, Li et al. [12] addressed the DT freshness issue, and proposed freshness metrics and one potential application of using Unmanned Aerial Vehicles (UAVs) for data collection. They proposed an efficient approximate solution for the problem through synchronizations between DTs and their objects under bandwidth and computing resource constraints. Chen et al. [3] utilized DT-assisted Deep Reinforcement Learning (DRL) to address an online mobility-aware dependent task offloading problem, with the aim of enhancing service quality. Yang et al. [23] studied a novel two-timescale accuracy-aware online optimization problem where models update by collecting personalized data from end devices, with the objective of maximizing the average accuracy of task execution assisted by human digital twins. Li et al. [9] utilized continual learning to retrain DT-assisted service models incrementally, to ensure that the service models can provide accurate services over time. Liang et al. [14] devised efficient algorithms to keep the states of both models and their source DTs as fresh as possible, and the state freshness is implemented through continuous synchronizations between DTs and their corresponding physical objects. The updated DT data is then utilized for the service model training. Zhang et al. [25] dealt with multi-Federated Learning (FL) service provisioning in an MEC network, by leveraging DTs to optimize resource allocation and mobile device scheduling. They proposed efficient algorithms and aimed at maximizing the cumulative utility of multiple FL services. However, none of the aforementioned studies considered the non-trivial relationship between the duration of model retraining and the accuracy improvement of DT-assisted inference models, not to mention the use of a previous version of a training model for user services.

Unlike the aforementioned studies, in this paper, we consider a realistic setting where the retraining duration of a service model may last multiple time slots, rather than within a single time slot, and limited resources in an MEC network for both model retraining and services need to be judiciously allocated at each time slot. We will investigate fidelity-aware inference services in a DT-assisted MEC network, striving for non-trivial trades-off of computing resource allocation between model retraining and inference services.

III. Preliminaries

In this section, we first introduce the system model, notions, and notations. Then, we define problems precisely.

A. System Model

Consider an MEC network $G = (N, E)$ operated by a service provider, where N is the set of Access Points (APs) with each having a co-located cloudlet, and E is the set of links between APs. Each AP and its co-located cloudlet are interconnected by an optic cable, and the communication delay between them is negligible [17]. Let C_j be the computing capacity on the cloudlet co-located with each AP j. Let d_e be the transmission delay on each link $e \in E$ per unit data [22]. Denote by L_j the number of sub-channels on each AP j, and these sub-channels follow the Orthogonal Frequency-Division Multiple Access (OFDMA) scheme.

Assume that there is a set V of mobile devices under the coverage of APs in the network. For each mobile device $v_i \in V$, there is a digital twin, DT_i, which has already been deployed in a cloudlet. Denote by $h(DT_i)$ the cloudlet hosting DT_i. As each mobile device v_i generates data continuously, to maintain the freshness of its DT, DT_i needs to synchronize with mobile device v_i quite often, which requires v_i to upload its update data to DT_i at some time slots. The state of DT_i is then refreshed after receiving and storing the update data. It is noticed that the synchronizations between DT_i and mobile device v_i will impact the inference accuracy of service models whose retraining data is fed by DT_i.

Assume that there is a set $\mathcal{M}$ of service models (e.g., DNNs) that need to be deployed in the network. For each model $M_m \in \mathcal{M}$, let μ_m be the amount of computing resource demanded for placing one instance of the model to provide inference services. Let $h(M_m)$ be the cloudlet hosting model M_m and V_m be the set of attributes of model M_m, with the source data of each attribute coming from an object $v \in V$. Let $comp_m$ be the amount of computing resource needed by model M_m for its retraining, and each model M_m is retrained using its updated DT source data, where objects of the DTs are in a subset V_m of mobile devices with $V_m \subseteq V$, and the DTs of mobile devices in V_m are referred to as *the source DTs* of model M_m. Fig. 1 gives an illustrative example, where five DTs and two DT-assisted service models are deployed in an MEC network.

In this paper, we consider a finite time horizon $\mathbb{T}$ that is divided into T equal time slots, with each lasting τ time units. Let $\mathbb{T} = \{1, \ldots, T\}$ be the set of time slots. We investigate high-fidelity service provisioning in a DT-assisted MEC network within time horizon $\mathbb{T}$.

B. Delays of Uploading Update Data for DT Updates

Given a mobile device $v_i \in V$, if it synchronizes with its digital twin, DT_i, by uploading its update data at some time slots, the data and state of DT_i will then be updated accordingly. Such an update on DT_i impacts the accuracy of all service models in which DT_i is their source DT.

The synchronization between mobile device v_i and its digital twin DT_i is detailed as follows. A mobile device is under the coverage of an AP if the AP is within the transmission range of the mobile device. Denote by $\mathcal{C}(j,t)$ the set of mobile devices covered by AP j at time slot t. Each mobile device v_i can be located at an overlapping area covered by multiple APs. We assume that the OFDMA scheme is adopted at each AP for its sub-channel allocation, and that each AP j has $L_j \geq 1$ sub-channels. If mobile device v_i is covered by AP j, it can choose AP j as a gateway for update data uploading, i.e., mobile device v_i is allocated with a sub-channel by AP j and its uploading rate $R_{i,j}$ is

$$R_{i,j} = \frac{B_j}{L_j} \log\left(1 + \frac{PX_i \cdot H_{i,j}}{\sigma^2}\right), \tag{1}$$

where B_j is the bandwidth capacity on AP j, PX_i is the transmission power of mobile device v_i, $H_{i,j}$ is the channel gain between mobile device v_i and AP j, and σ is the average noise power.

As the number of mobile devices under the coverage of each AP j is usually far larger than the number of sub-channels allocated to it, at each time slot, there are at most L_j mobile devices that can upload their update data via AP j by the assumption. Once a mobile device $v_i \in V$ (assuming that it is under the coverage of AP j) is chosen for uploading, its uploaded update data will be transferred to its DT hosting cloudlet $h(DT_i)$ for further processing, the processed data will be stored at DT_i, and the state of DT_i will be updated accordingly.

Let $vol(v_i, t)$ be the volume of the update data uploaded by mobile device v_i at time slot t, where the volume is proportional to the duration from the last uploading of mobile device v_i [14]. Denote by $V(t)$ the set of mobile devices chosen for uploading at time slot t. The cumulative volume $s_i(t)$ of the data stored in DT_i by time slot t is

$$s_i(t) = \begin{cases} s_i(t-1) + vol(v_i,t) & \text{if } v_i \in V(t), \\ s_i(t-1) & \text{otherwise,} \end{cases} \tag{2}$$

where $s_i(0) = 0$ for each DT_i initially with $v_i \in V$, and $s_i(t) \geq s_i(t-1)$.

If mobile device v_i is chosen to synchronize with its digital twin DT_i by uploading its update data to AP j at time slot t, the delay of updating DT_i consists of the uploading delay, the transmission delay, and the processing delay of the update data. Let $d^{DT}(v_i,t)$ be the delay of updating DT_i at time slot t, which is defined as follows.

$$d^{DT}(v_i,t) = \frac{vol(v_i,t)}{R_{i,j}} + \sum_{e \in P_{j,h(DT_i)}} d_e \cdot vol(v_i,t) + \frac{vol(v_i,t)}{\rho(DT_i)}, \tag{3}$$

where $R_{i,j}$ is the data uploading rate between mobile device v_i and AP j, $P_{j,h(DT_i)}$ is a shortest path in G between AP j and cloudlet $h(DT_i)$ in terms of delay metric, and $\rho(DT_i)$ is the processing rate of the update data by cloudlet $h(DT_i)$. We assume that the updating delay of DT_i of each mobile device v_i is no greater than the length τ of one time slot.

C. Retraining Duration of a Service Model Retraining

For each model $M_m \in \mathcal{M}$, let $\hat{t}_{m,l}$ be the start time slot of its lth retraining and the retraining last $I_{m,l}$ time slots, i.e., the retraining finishes at time slot $\hat{t}_{m,l} + I_{m,l} - 1$. $\hat{t}_{m,0} = 0$ and $I_{m,0} = 0$ for each model $M_m \in \mathcal{M}$ initially. Prior to the retraining of model M_m, the update data from its source DTs are required to be aggregated in cloudlet $h(M_m)$ that hosts its instance. The duration of the retraining of model M_m consists of the delay of its source DT updating, the transmission delay of its updated source DT data, and the delay of model retraining.

DT updating delay: Let $V_m(t)$ be the subset of mobile devices in V_m that are chosen to synchronize with their DTs at time slot t with $V_m(t) = V_m \cap V(t)$. When the lth retraining of model M_m starts at time slot $\hat{t}_{m,l}$, the mobile devices in $V_m(\hat{t}_{m,l})$ need to update their DTs first. The delay of the source DT updating of model M_m for its lth retraining is

$$d^{DT}_{m,l} = \max\left\{d^{DT}(v_i, \hat{t}_{m,l}) \mid v_i \in V_m\left(\hat{t}_{m,l}\right)\right\}, \tag{4}$$

which is the maximum updating delay of its source DTs at the start time slot of the lth retraining of model M_m.

Data transmission delay: As model retraining proceeds iteratively, the lth retraining of model M_m utilizes its updated source DT data that are uploaded between its $(l-1)$th and lth retraining, of which the volume is $s_i(\hat{t}_{m,l}) - s_i(\hat{t}_{m,l-1})$ for each $v_i \in V_m$. For each mobile device $v_i \in V_m$, the volume of the update data of DT_i for the lth retraining of model M_m is

$$\Delta s_{i,m,l} = \xi \cdot \left(s_i(\hat{t}_{m,l}) - s_i\left(\hat{t}_{m,l-1}\right)\right), \quad (5)$$

where ξ is the constant data compression rate with $0 < \xi \leq 1$. If there is not any data update on DT_i between time slot $\hat{t}_{m,l-1}$ and time slot $\hat{t}_{m,l}$, i.e., $\Delta s_{i,m,l} = 0$, there is no need to transmit the update data of DT_i for the lth retraining of model M_m.

For each source DT, DT_i, of model M_m with its mobile device $v_i \in \cup_{t=\hat{t}_{m,l-1}+1}^{\hat{t}_{m,l}} V_m(t)$ (there is updated data of DT_i since time slot $\hat{t}_{m,l-1}$), the transmission delay of transmitting the update data of DT_i to cloudlet $h(M_m)$ hosting an instance of model M_m for its lth retraining is

$$d_{i,m,l}^{trans} = \sum_{e \in P_{h(DT_i),h(M_m)}} d_e \cdot \Delta s_{i,m,l}, \quad (6)$$

where d_e is the transmission delay along link e per unit data, and $P_{h(DT_i),h(M_m)}$ is a shortest path in G between cloudlets $h(DT_i)$ and $h(M_m)$.

The transmission delay of transmitting the updated source DT data from their DT hosts to cloudlet $h(M_m)$ for the lth retraining of model M_m is

$$d_{m,l}^{trans} = \max\left\{d_{i,m,l}^{trans} \mid v_i \in \cup_{t=\hat{t}_{m,l-1}+1}^{\hat{t}_{m,l}} V_m(t)\right\}. \quad (7)$$

Model retraining delay: After its source DTs have transferred their update data to cloudlet $h(M_m)$, model M_m will be retrained using the updated DT data, and its lth retraining delay is

$$d_{m,l}^{train} = \sum_{v_i \in V_m} \frac{\Delta s_{i,m,l}}{\rho(M_m)}, \quad (8)$$

where $\rho(M_m)$ is the data processing rate of cloudlet $h(M_m)$ allocated for retraining model M_m.

Model retraining duration: The number $I_{m,l}$ of time slots needed for the lth retraining of model M_m is

$$I_{m,l} = \left\lceil \frac{d_{m,l}^{DT} + d_{m,l}^{trans} + d_{m,l}^{train}}{\tau} \right\rceil, \quad (9)$$

where τ is the length of one time slot.

D. Cost of the Initial Service Model Instance Placements

The cost of communication and computing resources consumed by a service model retraining is heavily determined by its placement location. Since the volume of update data of each mobile device varies over time, it is challenging to place service models without the update data information of their source DTs. For each mobile device $v_i \in V$, let $\widetilde{vol}(v_i)$ be the average volume of update data uploaded by device v_i to its DT_i, which can be estimated by leveraging machine learning prediction techniques on the historical data traces stored at DT_i. *The expected model placement cost* of model $M_m \in \mathcal{M}$ when deployed in cloudlet j is defined as follows.

$$\overline{cost}(M_m, j) = \sum_{v_i \in V_m} \sum_{e \in P_{h(DT_i),j}} \xi \cdot cost(e) \cdot \widetilde{vol}(v_i)$$
$$+ \eta_j \cdot comp_m \cdot \left\lceil \frac{\sum_{v_i \in V_m} \xi \cdot \widetilde{vol}(v_i)}{\rho(M_m) \cdot \tau} \right\rceil \quad (10)$$

where $cost(e)$ is the cost of transferring a unit data along link $e \in E$, ξ is the compression ratio of the uploaded data by a device that is merged with existing data of its DT, and η_j is the cost of unit computing resource consumed of cloudlet j per time slot. $P_{h(DT_i),j}$ is a shortest path in G between cloudlet $h(DT_i)$ and cloudlet $j \in N$. Note that the first term in the RHS of Eq. (10) is the total cost of transmitting the compressed update data from its source DTs to cloudlet $j \in \mathcal{N}$ for retraining model M_m, and the second term in the RHS of Eq. (10) is the cost of model retraining in cloudlet $j \in \mathcal{N}$ for model M_m, where $\left\lceil \frac{\sum_{v_i \in V_m} \xi \cdot \widetilde{vol}(v_i)}{\rho(M_m) \cdot \tau} \right\rceil$ is the number of time slots needed for the training of M_m, and processing rate $\rho(M_m)$ is a given function proportional to the amount $comp_m$ of computing resource allocated to model M_m, which is determined by the parameter size of the model and the amount $comp_m$ of computing resource allocated to the model.

Let $x_{m,j}$ be a binary variable, indicating whether model instance M_m is deployed to cloudlet j ($x_{m,j} = 1$) or not ($x_{m,j} = 0$), with $M_m \in \mathcal{M}$ and $j \in N$. The total cost of all model placements in cloudlets of the MEC network G is

$$\sum_{M_m \in \mathcal{M}} \sum_{j \in N} \overline{cost}(M_m, j) \cdot x_{m,j}. \quad (11)$$

E. Cost of Service Model Retraining

The model retraining in an MEC network consumes both computing and communication resources. Assuming that model $M_m \in \mathcal{M}$ is deployed in cloudlet j, i.e., $h(M_m) = j$, then the cost for its lth retraining consists of the following components.

The transmission cost of the update data transferred from the DTs of mobile devices in V_m to cloudlet $h(M_m)$ hosting model M_m for its lth retraining starting at time slot t ($\hat{t}_{m,l} = t$) is defined as follows. Recall that V_m is the set of attributes of model M_m. To train M_m in its lth retraining starting at time slot t, the cumulative update data from each of its source DTs needs to be transferred to the host $h(M_m)$ of model M_m, and the update data transmission cost of the retraining thus is

$$comm(M_m, t) = \sum_{v_i \in V_m} \sum_{e \in P_{h(DT_i),h(M_m)}} cost(e) \cdot \Delta s_{i,m,l}, \quad (12)$$

where $cost(e)$ is the cost of transferring a unit data along link $e \in E$, and $P_{h(DT_i),h(M_m)}$ is a shortest path in G between cloudlets $h(DT_i)$ and $h(M_m)$.

The retraining cost of the lth retraining of model M_m in cloudlet $h(M_m)$ starting at time slot t is defined as follows. During its lth retraining, the previous version of the model after its $(l-1)$th retraining still can serve its users, and the version after the lth retraining will be used for user services in the future time window $[\hat{t}_{m,l} + I_{m,l}, |\mathbb{T}|]$ if it will be no longer retrained. Since the lth retraining of model M_m lasts $I_{m,l}$ time slots, the cost $comp(M_m, t)$ of the total amount of computing resource consumed by the retraining is

$$comp(M_m, t) = \eta_{h(M_m)} \cdot comp_m \cdot I_{m,l}, \quad (13)$$

where $\eta_{h(M_m)}$ is the cost of unit resource of cloudlet $h(M_m)$ per time slot, and $comp_m$ is the amount of computing resource allocated to model M_m for its retraining.

The total cost for the lth retraining of model M_m in cloudlet $h(M_m)$ starting at time slot t thus is $comm(M_m, t) + comp((M_m, t)$.

F. Utility Gain of Service Models

The fidelity of a service model can be enhanced after its retraining using the update data from its source DTs. Here

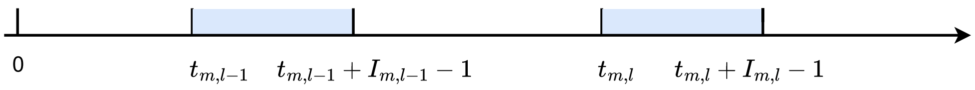


Fig. 2. An illustrative example, where the lth retraining of model M_m starts and ends at time slots $\hat{t}_{m,l}$ and $\hat{t}_{m,l} + I_{m,l} - 1$, respectively, and its previous $(l-1)$th retraining started and ended at time slots $\hat{t}_{m,l-1}$ and $\hat{t}_{m,l-1} + I_{m,l-1} - 1$, respectively, and $\hat{t}_{m,l} \geq \hat{t}_{m,l-1} + I_{m,l-1} - 1$.

we make use of the utility gain to capture the service fidelity enhancement on a service model. Specifically, let $u_m(t)$ be the utility of model M_m at time slot t, which is usually proportional to the inference accuracy of model M_m at time slot t. A larger utility indicates a higher inference accuracy, while a lower utility implies a lower inference accuracy. To maintain high inference accuracy of a service model M_m, it is essential to retrain model M_m continuously using its updated source DT data. For the lth retraining of model M_m, recall that its start time slot is $\hat{t}_{m,l}$ and the retraining lasts $I_{m,l}$ time slots, where the values of $\hat{t}_{m,l}$ and $I_{m,l}$ are determined by the retraining scheduling of model M_m. Fig. 2 shows the $(l-1)$th and lth retraining of model M_m, respectively, $(\hat{t}_{m,l-1} + I_{m,l-1} - 1)$ is the finish time slot of the $(l-1)$th retraining with $(\hat{t}_{m,l-1} + I_{m,l-1} - 1) \leq \hat{t}_{m,l}$.

The utility gain of model M_m after its lth retraining is defined as follows.

Assume that DT_i of mobile device $v_i \in V_m$ is one source DT of model M_m, and there is a fixed weight $\omega_{i,m}$ of DT_i that is its contribution to the accuracy of model M_m with $0 < \omega_{i,m} < 1$, indicating the importance of its update data to the accuracy of model M_m with $\sum_{v_i \in V_m} \omega_{i,m} = 1$ [11]. Furthermore, if DT_i is the source DT of multiple service models, then the update data of DT_i will impact the utilities of these models when they are retrained.

To model the impact of the update data of an object on service models in which it is one source data, we make use of a multi-dimensional submodular non-decreasing function $g_m(\cdot, \ldots, \cdot)$ with $|V_m|$ parameters to represent the utility (or accuracy) of model M_m, where function $g_m(\cdot, \ldots, \cdot)$ meets a diminishing marginal utility gain property. Assuming that the update data uploaded by different mobile devices in V_m impact the service accuracy of model M_m independently, then, the submodular function $g_m(\cdot, \ldots, \cdot)$ can be decomposed into a weighted sum of a one-dimensional submodular non-decreasing function $f(\cdot)$, i.e., $g_m(\cdot, \ldots, \cdot) = \sum_{v_i \in V_m} \omega_{i,m} \cdot f(\cdot)$. Since $f(\cdot)$ is a one-dimensional submodular non-decreasing function with $f(0) = 0$, we have $f(x + \delta) - f(x) \leq f(x' + \delta) - f(x')$ if $x > x' > 0$ and $\delta > 0$.

For each DT_i with $v_i \in V_m$, the update data of DT_i for the lth retraining of model M_m is the one generated and then uploaded by mobile device v_i from time slot $(\hat{t}_{m,l-1} + 1)$ to time slot $\hat{t}_{m,l}$. Denote by $\Delta u_{m,l}(v_i)$ the utility gain of device v_i to model M_m after the lth retraining of model M_m. Then,

$$\Delta u_{m,l}(v_i) = \omega_{i,m} \cdot \left(f\left(s_i\left(\hat{t}_{m,l}\right)\right) - f\left(s_i\left(\hat{t}_{m,l-1}\right)\right)\right), \quad (14)$$

where $s_i(t)$ is the cumulative volume of data stored in DT_i by time slot t. Notice that if there is no update on DT_i between time slots $(\hat{t}_{m,l-1} + 1)$ and $\hat{t}_{m,l}$, then $\Delta u_{m,l}(v_i) = 0$.

Let $u_{m,l}$ and $\Delta u_{m,l}$ be the utility and utility gain of model M_m after its lth retraining, respectively, then

$$\Delta u_{m,l} = u_{m,l} - u_{m,l-1} = \sum_{v_i \in V_m} \Delta u_{m,l}(v_i), \quad (15)$$

$$\begin{aligned} u_{m,l} &= u_{m,0} + \sum_{k=0}^{l-1} (u_{m,k+1} - u_{m,k}) \\ &= u_{m,0} + \sum_{k=1}^{l} \Delta u_{m,k} \\ &= u_{m,0} + \sum_{k=1}^{l} \sum_{v_i \in V_m} \Delta u_{m,k}(v_i), \text{ by Eq. (15)} \\ &= \sum_{v_i \in V_m} \sum_{k=1}^{l} \omega_{i,m} \cdot \left(f(s_i(\hat{t}_{m,k})) - f\left(s_i\left(\hat{t}_{m,k-1}\right)\right)\right), \\ & \qquad \text{by Eq. (14)} \\ &= \sum_{v_i \in V_m} \omega_{i,m} \cdot f\left(s_i(\hat{t}_{m,l})\right), \end{aligned} \quad (16)$$

where $u_{m,0} = 0$, $t_{m,0} = 0$, and $I_{m,0} = 0$ for all $M_m \in \mathcal{M}$ initially.

The utility $u_m(t)$ of model M_m at each time slot t is defined as follows. At any time slot $t \in \mathbb{T}$, each model M_m is either being retrained or providing inference services. If time slot t is within the lth retraining of model M_m, i.e., $\hat{t}_{m,l} \leq t < \hat{t}_{m,l} + I_{m,l}$, then the utility of model M_m at time slot t is $u_m(t) = u_{m,l-1}$. The updated version of model M_m after its lth retraining with no further retraining can be used for user services if $t \geq \hat{t}_{m,l} + I_{m,l}$, then $u_m(t) = u_{m,l}$ if it is not in retraining at time slot t, i.e.,

$$u_m(t) = \begin{cases} 0, & \text{if } t = 0, \\ u_{m,l-1}, & \exists\, l \geq 0, \hat{t}_{m,l} \leq t < \hat{t}_{m,l} + I_{m,l} \\ u_{m,l}, & t \geq \hat{t}_{m,l} + I_{m,l}, \end{cases} \quad (17)$$

where $u_{m,l-1}$ and $u_{m,l}$ are defined in Eq. (16).

G. Problem Definitions

In the following, we first deal with the model instance placement problem with the aim to minimize the placement cost of all service model instances. Then, we define a cumulative utility maximization problem to maximize the cumulative utility of all placed service models over a given time horizon $\mathbb{T}$, by jointly choosing mobile devices for uploading, and service models for retraining using their updated source DT data.

Definition 1: Consider an MEC network $G = (N, E)$, a set V of mobile devices with DT_i of each device v_i deployed in cloudlet $h(DT_i)$, and a set $\mathcal{M}$ of service models (e.g., DNNs) with each having multiple DT sources. Each service model $M_m \in \mathcal{M}$ is retrained continuously by leveraging its updated source DT data uploaded by devices in V_m ($V_m \subseteq V$). With the assumption that DT_i of each mobile device $v_i \in V$ is placed into the cloudlet $h(DT_i)$, *the model instance placement problem* in G is to place an instance for each model $M_m \in \mathcal{M}$ to a cloudlet such that the total cost (11) of resources consumed for model

TABLE I
TABLE OF NOTATIONS

Notation	Descriptions
$G = (N, E)$	An MEC network with a set N of APs and a set E of links
C_j, B_j	The computing resource capacity on cloudlet j and the bandwidth capacity of AP j
d_e, $cost(e)$	The transmission delay per unit data and cost of per unit data transfer along link e
L_j	The number of sub-channels on AP j
V, $V(t)$	A set of mobile devices and the set of mobile devices chosen for uploading at time slot t
DT_i, $h(DT_i)$	The DT of mobile device v_i and its hosting cloudlet
$\mathcal{M}$, $h(M_m)$	A set of inference service models and the cloudlet hosting model $M_m \in \mathcal{M}$
μ_m, $comp_m$	The amount of computing resource demanded by a model instance of model M_m and the amount of computing resource for its retraining
V_m, $V_m(t)$	The attribute set of model M_m and the subset of mobile devices in V_m that are chosen for uploading at time slot t
$\mathbb{T} = \{1, \ldots, T\}$, τ	A finite time horizon and the length of one time slot
$\mathcal{C}(j, t)$	The set of mobile devices covered by AP j at time slot t
$R_{i,j}$	The uploading rate of mobile device v_i to AP j
PX_i	The transmission power of mobile device v_i
$H_{i,j}$	The channel gain between mobile device v_i and AP j
$\omega_{i,m}$	The weight associated with the contribution of DT_i to the accuracy of model M_m
$vol(v_i, t)$	The volume of the update data uploaded by mobile device $v_i \in V(t)$ at time slot t
$s_i(t)$	The cumulative volume of the data stored in DT_i by time slot t
$d^{DT}(v_i, t)$	The delay of updating DT_i at time slot t
$P_{j,h(DT_i)}$	A shortest path in G between AP j and cloudlet $h(DT_i)$
$\rho(DT_i)$	The processing rate of the update data by cloudlet $h(DT_i)$
$\hat{t}_{m,l}$	The start time slot of the lth retraining of model M_m
$d^{DT}_{m,l}$	The delay of the source DT updating of model M_m for its lth retraining
$\Delta s_{i,m,l}$	The volume of the update data of DT_i for the lth retraining of model M_m
ξ	The constant data compression rate
$d^{trans}_{i,m,l}$	The transmission delay of transmitting the update data of DT_i to cloudlet $h(M_m)$ for the lth retraining of model M_m
$d^{trans}_{m,l}$	The transmission delay of transmitting all the updated source DT data to cloudlet $h(M_m)$ for the lth retraining of model M_m
$P_{h(DT_i),h(M_m)}$	A shortest path in G between cloudlets $h(DT_i)$ and $h(M_m)$
$d^{train}_{m,l}$	The retraining delay for the lth retraining of model M_m
$\rho(M_m)$	The data processing rate of cloudlet $h(M_m)$
$I_{m,l}$	The number of time slots experienced by the lth retraining of model M_m
$g_m(\cdot, \ldots, \cdot)$, $f(\cdot)$	Submodular functions for the model utility
$u_m(t)$	The utility of model M_m at time slot t
$\Delta u_{m,l}(v_i)$	The utility gain of device v_i to model M_m after the lth retraining of model M_m
$\Delta u_{m,l}$, $u_{m,l}$	The utility gain and utility of model M_m after its lth retraining
$\overline{cost}(M_m, j)$	The expected model placement cost of deploying model M_m in cloudlet j
η_j	The cost of unit computing resource consumed of cloudlet $j \in N$ per time slot
$\widetilde{vol}(v_i)$	The average volume of updated data uploaded by v_i

placement and retraining is minimized, subject to computing and communication capacities on cloudlets and APs in G.

Recall that DTs of all mobile devices have been deployed in cloudlets, i.e., DT_i is placed in cloudlet $h(DT_i)$. After the model instance of each model $M_m \in \mathcal{M}$ has been placed to cloudlet $h(M_m)$, we consider the cumulative utility maximization problem over a finite time horizon $\mathbb{T}$ as follows.

Definition 2: Given an MEC network $G = (N, E)$, a set V of mobile devices with DT_i of each device $v_i \in V$ deployed in cloudlet $h(DT_i) \in N$, a set $\mathcal{M}$ of service models, and a finite time horizon $\mathbb{T}$ that is divided into $|\mathbb{T}|$ equal time slots, assume that the model instance of each model $M_m \in \mathcal{M}$ has already been deployed in cloudlet $h(M_m)$. For each model $M_m \in \mathcal{M}$, its previous version can be used for user services during its retraining period, and its retrained version with a higher fidelity will be available when its retraining finishes. The *cumulative utility maximization problem* in G is to maximize the cumulative utility of all service models over time horizon $\mathbb{T}$, subject to the communication and computing capacities on APs and cloudlets in the network, where the utility of a service model at one time slot implies its service fidelity at that time slot.

The optimization objective of the problem is to

$$\text{maximize} \sum_{t=1}^{|\mathbb{T}|} \sum_{M_m \in \mathcal{M}} u_m(t), \tag{18}$$

where the utility $u_m(t)$ of model M_m at time slot t is defined by Eq. (17).

H. NP-Hardness of the Problem

Theorem 1: The model instance placement problem in an MEC network $G = (N, E)$ is NP-hard.

Proof: We show the NP-hardness of the model instance placement problem by reducing it from the minimum-cost generalized assignment problem (GAP) as follows.

There is a set $\mathcal{B}$ of bins and a set $\mathcal{I}$ of items. Each bin $b \in \mathcal{B}$ has capacity $C(b)$. Packing item $i \in \mathcal{I}$ to bin $b \in \mathcal{B}$ brings a cost $c(i, b)$ with size $s(i, b)$. The minimum-cost GAP is to minimize the total cost of packed items, subject to capacities on bins.

Each cloudlet $j \in N$ is treated as a bin with capacity C_j, i.e., the computing capacity of cloudlet j. Each model M_m is treated as an item with a weight μ_m, i.e., the amount of computing resource demanded by placing an instance of model M_m. Placing an instance of model M_m to cloudlet j will incur a cost $\overline{cost}(M_m, j)$ with weight μ_m. The model instance placement problem is to minimize the total cost by placing as many models as possible to cloudlets, subject to the computing capacity on each cloudlet. The model instance placement problem can be reduced from the minimum-cost GAP problem, while the latter is a classic NP-hard problem [18]. Hence, the model instance placement problem is NP-hard.

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

IV. Algorithms for the Model Instance Placement Problem

In this section, we first show that the average volume of updated DT data can be obtained by a prediction method. Then, we formulate an Integer Linear Programming (ILP) for the model instance placement problem when the problem size is small, otherwise we propose an approximation algorithm by reducing the problem from the minimum-cost Generalized Assignment Problem (GAP) [18]. An approximate solution to the latter in turn returns an approximate solution to the former.

A. Predicting the Average Volume of Update Data

We introduce a prediction mechanism to predict the average volume $\widetilde{vol}(v_i)$ of the update data by each mobile device $v_i \in V$ since its last uploading. Since the digital twin, DT_i, of each mobile device v_i contains all historical traces of uploading information of device v_i, by making use of an ML prediction method such as the auto-regression method, or the Long-Short-Term-Memory (LSTM) method, the average volume $\widetilde{vol}(v_i)$ of update data of mobile device v_i since its last uploading can be obtained. Assuming that the current time slot is t, we here adopt the auto-regression method for predicting the average volume $\widetilde{vol}(v_i)$ of the update data by mobile device v_i, using the uploading information of v_i in the previous p uploading rounds, and the value of parameter $p \geq 1$ is fixed. We have

$$\widetilde{vol}(v_i) = a_1 \cdot vol\left(v_i, t_p\right) + a_2 \cdot vol\left(v_i, t_{p-1}\right) + \dots + a_p \cdot vol\left(v_i, t_1\right), \tag{19}$$

where for each p' with $1 \leq p' \leq p$, $t_{p'}$ is the time slot of the previous uploading round p' of v_i, $a_{p'}$ is a positive constant with $\sum_{p'=1}^{p} a_{p'} = 1$, while the volume $vol(v_i, t_{p'})$ is the actual volume of the update data uploaded by v_i at previous round p' with $1 \leq p' \leq p$.

B. ILP and Approximation Algorithms

We consider the model instance placement problem by formulating an ILP solution when the problem size is small. Otherwise, we devise an approximation algorithm to it, at the expense of moderate resource violations. We formulate an ILP solution to the model instance placement problem. Assume that the average volume of the update data uploaded by each mobile device is given. The optimization objective (20) is the expected total cost of all model instance placements.

Let $x_{m,j}$ be a binary decision variable, where $x_{m,j} = 1$ indicates that model M_m is placed in cloudlet j, and $x_{m,j} = 0$ otherwise. The ILP formulation is given as follows.

$$\text{Minimize} \sum_{M_m \in \mathcal{M}} \sum_{j=1}^{|N|} \overline{cost}(M_m, j) \cdot x_{m,j} \tag{20}$$

subject to

$$\sum_{M_m \in \mathcal{M}} \mu_m \cdot x_{m,j} \leq C_j, \quad \forall j:\ 1 \leq j \leq |N| \tag{21}$$

$$\sum_{j=1}^{|N|} x_{m,j} = 1,\ \forall M_m \in \mathcal{M} \tag{22}$$

$$x_{m,j} \in \{0,1\},\ \forall M_m \in \mathcal{M}, \quad \forall j: 1 \leq j \leq |N| \tag{23}$$

where cost $\overline{cost}(M_m, j)$ is defined by Eq. (10). Constraint (21) ensures that the computing resource constraint on each cloudlet j must be met. Constraint (22) guarantees that the instance of each model is placed in one cloudlet only. It must be mentioned that the proposed ILP solution is applicable only when the problem size is small; otherwise, finding an exact solution takes a prohibitively long time. In the following, we devise an approximation algorithm for it when the problem size is large, through a reduction from the minimum-cost Generalized Assignment Problem (GAP) as follows. Each cloudlet $j \in N$ can be seen as a bin with capacity C_j, each model $M_m \in \mathcal{M}$ corresponds to an item with weight of μ_m, and placing model M_m to cloudlet j incurs a cost $\overline{cost}(M_m, j)$ by Eq. (10) with weight μ_m if $\mu_m \leq C_j$. There is an approximation algorithm for the minimum-cost GAP due to Shmoys and Tardos [18], which delivers an optimal solution for the problem at the expense of twice the computing capacity violation on any cloudlet.

The detailed approximation algorithm for the model instance placement problem is given in Algorithm 1.

Algorithm 1: Approximation Algorithm for the Model Instance Placement Problem.

```
Input: An MEC network $G = (N, E)$, a set $V$ of mobile
    devices whose DTs have been deployed in cloudlets
    already, the average volume $\widetilde{vol}(v_i)$ of the updated DT
    data from each mobile device $v_i$ (which can be derived
    by the historical traces in its DT), and a set $\mathcal{M}$ of models
    to be placed in cloudlets.
Output: Minimize the total cost of model placements.
1: Find a shortest path in MEC network $G$ between each
   pair of cloudlets;
2: for each $M_m \in \mathcal{M}$ do
3:   for each cloudlet $j \in N$ do
4:     Calculate $\overline{cost}(M_m, j)$ by Eq (10)
5:   end for ;
6: end for ;
7: The model instance placement problem is reduced
   from the minimum-cost GAP;
8: An approximate solution $\mathcal{A}$ is obtained to the
   minimum-cost GAP, by applying the approximation
   algorithm in [18];
9: return An approximate solution is derived from $\mathcal{A}$ for
   the model instance placement problem.
```

C. Algorithm Analysis

Theorem 2: Given an MEC network $G = (N, E)$, a set V of mobile devices with the given average volume $\widetilde{vol}(v_i)$ of update data by each mobile device $v_i \in V$, and a set $\mathcal{M}$ of models with each model M_m demanding amount μ_m of computing resource, assume that the DTs of all mobile devices have already been deployed in cloudlets, and each model M_m is retrained using its updated source DT data that come from mobile devices in V_m. Algorithm 1 delivers an optimal solution to the model instance placement problem, at the expense of twice the computing capacity violation on each cloudlet $j \in N$. The time complexity of Algorithm 1 is $O(|\mathcal{M}|^3 \cdot |N|^3)$.

Proof: As the model instance placement problem can be reduced from a minimum-cost GAP, it can be solved by adopting the approximation algorithm in [18], and Algorithm 1 delivers an optimal solution for the problem, at the expense of no more than twice the resource capacity violation. The running time of Algorithm 1 is analyzed as follows. To calculate the cost of placing the model instances into cloudlets, a shortest path between each pair of cloudlets in G needs to be found, which takes $O(|N|^3)$ time. Since the time complexity of the approximation algorithm in [18] for the minimum-cost GAP is $O(|\mathcal{M}|^3 \cdot |N|^3)$, Algorithm 1 takes $O(|\mathcal{M}|^3 \cdot |N|^3)$ time.

Remarks: Although the time complexity of Algorithm 1 is relatively high, the deployment of service models into cloudlets is performed at the initial deployment stage, and only once. It is also noticed that Algorithm 1 will deliver an optimal solution, at the expense of bounded resource violations on some cloudlets in N. In practice, the probability of such resource capacity violations is negligible when each cloudlet is not overloaded. This has been validated in the later experimental evaluation on the performance of Algorithm 1.

V. Online Algorithm for the Cumulative Utility Maximization Problem

In this section, we tackle the cumulative utility maximization problem by jointly choosing mobile devices for uploading and service models for retraining at each time slot, assuming that both DTs of mobile devices and service model instances have already been deployed. Since this joint optimization problem is intractable, we instead decompose the problem into two subproblems: the mobile device selection problem and the model retraining selection problem, respectively. In the following, we deal with these two sub-problems separately, by proposing a solution to each of the subproblems.

A. Choosing Mobile Devices to Update Their DTs

We schedule mobile devices to upload their update data to their DTs, subject to the bandwidth capacity on each AP. It can be seen that it is very unlikely that all mobile devices under the coverage of an AP can upload their update data via the AP at each time slot, due to the limited bandwidth capacity on the AP. Instead, we only choose these mobile devices to maximize the cumulative utility gain of service models retrained by their uploaded data.

Specifically, let $V(t) \subseteq V$ be a subset of mobile devices chosen at time slot t. If a mobile device $v_i \in V(t)$ is chosen to upload its update data at time slot t, the data volume $s_i(t)$ of DT_i in Eq. (2) increases by $vol(v_i, t)$, which potentially contributes to the cumulative utility gain $\Delta utility(v_i, t)$ of the service models that DT_i is their source DT, provided that each of the models is retrained. Then we have

$$\Delta utility\,(v_i, t) = \sum_{M_m \in \mathcal{M}_i} \omega_{i,m}\ \left(f\left(s_i(t)\right) - f\left(s_i(t-1)\right)\right), \tag{24}$$

where $\mathcal{M}_i$ is a set of models in which v_i is one of their DT sources, $\omega_{i,m}$ is the weight associated with the update data in DT_i to the utility gain of model M_m, and $\omega_{i,m}\ (f(s_i(t)) - f(s_i(t-1)))$ is the proportion of the potential utility gain of model $M_m \in \mathcal{M}_i$ after its retraining, using the update data in DT_i at time slot t.

Due to limited bandwidth capacity on each AP j, it is critical to choose which mobile devices via AP j for their DT updating to maximize the total utility. As a mobile device can be covered by multiple APs, choosing which AP for its uploading is challenging, too. To this end, we choose mobile devices through which APs for their uploading at each time slot t, by utilizing the maximum matching in an auxiliary bipartite graph as follows.

A bipartite graph $G_B(t) = (X, Y, E_{XY})$ at time slot t is constructed, where set $X = \{v_i \mid v_i \in V\}$, set $Y = \{ch_{j,k} \mid j \in N,\ 1 \le k \le L_j\}$, the node set of $G_B(t)$ is $X \cup Y$ and the edge set is E_{XY}. Recall that $\mathcal{C}(j, t)$ is the set of mobile devices covered by AP j at time slot t. For each mobile device $v_i \in \mathcal{C}(j, t)$, there is an edge $(v_i, ch_{j,k}) \in E_{XY}$ between nodes $v_i \in X$ and $ch_{j,k} \in Y$ with weight $\Delta utility(v_i, t)$ by Eq. (24) and $1 \le k \le L_j$. A weighted maximum matching $M_B(t)$ in $G_B(t)$ can then be found, corresponding to which mobile devices are chosen for uploading through which APs at time slot t.

Denote by $V(t)$ the set of chosen mobile devices at time slot t, i.e., $V(t) = \{v_i \mid v_i \in V,\ (v_i, ch_{j,k}) \in M_B(t)\}$. For each model $M_m \in \mathcal{M}$, let $V_m(t) = V(t) \cap V_m$ be the subset of chosen mobile devices in V_m at time slot t. Denote by $\mathcal{M}(t) = \{M_m \mid M_m \in \mathcal{M}, \bigcup_{t'=\hat{t}_{m,l-1}+1}^{t} V_m(t') \neq \emptyset\}$ the set of models in which at least one source DT of each model $M_m \in \mathcal{M}(t)$ has updated since time slot $\hat{t}_{m,l-1}$, where $\hat{t}_{m,l-1}$ is the starting time slot of the last retraining (or the $(l-1)$th retraining) of model M_m with $\hat{t}_{m,l-1} < t$. It can be seen that the upper bound on the utility gain of the objective function (18) by the updated DT data from mobile devices in $V(t)$ is $\sum_{v_i \in V(t)} \Delta utility(v_i, t)$ at time slot t, as some models may not be chosen for retraining at time slot t.

Assume that the last retraining of model $M_m \in \mathcal{M}(t)$ was its $(l-1)$th retraining and starting at time slot $\hat{t}_{m,l-1}$. If model M_m is chosen to be trained at time slot t with $t > \hat{t}_{m,l-1} + I_{m,l-1}$ (i.e., its lth retraining starts at time slot t with $\hat{t}_{m,l} = t$), then its utility gain $\Delta u_{m,l}$ after the retraining is obtained by Eq. (14) and Eq. (15). An algorithm for choosing mobile devices to update their DTs at time slot t is given in Algorithm 2.

B. Choosing Service Models for Retraining

Having chosen mobile devices for DT updating, to maximize the cumulative utility of service models, we now choose service models for retraining using their updated source DT data, by exploring a nontrivial trade-off of computing resource consumptions between model retraining and inference services, subject to resource capacities on cloudlets.

Specifically, at each time slot t, a subset of mobile devices is chosen to update their DTs by invoking Algorithm 2 first. Then, a control policy is developed to choose a subset of service models for retraining at time slot t. Since each chosen model will incur a certain amount of resource consumption during its retraining, a certain amount of utility gain will be brought by each retrained service model as well. Hence, only service models that can bring positive net gain to the objective function and utilize the resources efficiently can be chosen for retraining. To this end, the control policy of model choices makes use of the ratio of the net utility gain obtained by retraining a service model to its prolonged resource consumption for its retraining.

Algorithm 2: Algorithm for Scheduling Mobile Devices for Uploading Their Update Data at Time Slot t.

```
Input: An MEC network $G = (N, E)$, a given time slot $t$, $DT_i$ of each device $v_i \in V$ deployed in cloudlet $h(DT_i)$, and a set $\mathcal{M}$ of DT-assisted service models placed in cloudlets.
Output: Choose a subset of mobile devices in $V$ to upload their update data, subject to bandwidth capacities on APs.
1:  $X \leftarrow \{v_i \mid v_i \in V\}$;
2:  $Y \leftarrow \{ch_{j,k} \mid j \in N,\ 1 \le k \le L_j\}$;
3:  $E_{XY} \leftarrow \emptyset$;
4:  for AP $j \in N$ do
5:      for mobile device $v_i \in \mathcal{C}(j,t)$ do
6:          $E_{XY} \leftarrow E_{XY} \cup \{(v_i, ch_{j,k}) \mid 1 \le k \le L_j\}$
7:      end for ;
8:  end for ;
9:  Construct a bipartite graph $G_B(t) = (X, Y, E_{XY})$;
10: Find a weighted maximum matching $M_B(t)$ in $G_B(t)$ by invoking the Hungarian algorithm;
11: return A subset $V(t)$ of mobile devices is derived from the maximum matching $M_B(t)$ at time slot $t$.
```

The procedure of choosing which models for retraining at each time slot proceeds iteratively. At each time $t \in \mathbb{T}$, a model with the maximum ratio is chosen for retraining, and this procedure continues until either all models are chosen, or there is no adequate resource to host any further model retraining at the current time slot.

Let $\mathcal{M}^{train}(t)$ be the set of models in training by time slot t, let $\mathcal{M}^{pot}(t)$ be the set of potential models for retraining at time slot t, and let $C_j^{res}(t)$ be the remaining computing resource on cloudlet j at time slot t, excluding the resource that has been allocated to DTs, model instance placements, and the current model retraining. Given a model $M_m \in \mathcal{M}^{pot}(t)$ $(= \mathcal{M}(t) \backslash \mathcal{M}^{train}(t))$ with $h(M_m) = j$, assume that it has been chosen for retraining at time slot t. Let $(l-1)$ be the index of its last round retraining, i.e., model M_m starts its lth retraining with the start time slot t. Then, its retraining will last $I_{m,l}$ time slots, and the utility gain $\Delta u_{m,l}$ after its retraining can be calculated by Eq. (9) and Eq. (15), respectively. Let $net\_gain(M_m, t)$ be the net utility gain after the lth retraining of model M_m starting at time slot $t\ (= \hat{t}_{m,l})$. Then,

$$net\_gain(M_m, t) = \Delta u_{m,l} \cdot (|\mathbb{T}| - (t + I_{m,l} - 1)), \quad (25)$$

where $t + I_{m,l} - 1$ is the time slot at which the lth retraining of model M_m finishes. It can be seen that the value of $net\_gain(M_m, t)$ is not always positive. If the retraining of model M_m cannot finish in the end of time horizon $\mathbb{T}$, i.e., $t + I_{m,l} - 1 \ge |\mathbb{T}|$, then the retraining of model M_m will not lead to any improvement on its service accuracy.

The ratio $r(M_m, t)$ of the net utility gain of model $M_m \in \mathcal{M}^{pot}(t)$ to the cost of resources consumed for its lth retraining is

$$r(M_m, t) = \frac{net\_gain(M_m, t)}{comm(M_m, t) + comp(M_m, t)}, \quad (26)$$

where $comm(M_m, t)$ and $comp(M_m, t)$ are defined in Eq. (12) and Eq. (13), respectively.

C. Online Algorithm

The online algorithm proceeds iteratively. Each iteration corresponds to one time slot. Within each time slot $t \in \mathbb{T}$, it first chooses a subset $V(t) \subseteq V$ of mobile devices for uploading by invoking `Algorithm` 2. It then chooses a subset of service models in $\mathcal{M}^{pot}(t)$ for retraining, and a service model $M_m \in M^{pot}(t)$ is chosen when Model M_m meets the following condition:

$$m = \underset{M_m \in \mathcal{M}^{pot}(t)}{\operatorname{argmax}} \{r(M_m, t) \mid comp_m \le C_j^{res}(t), h(M_m) = j\}. \quad (27)$$

Recall that $C_j^{res}(t)$ is the residual resource of cloudlet $j \in N$ at time slot t. In other words, model M_m is assigned to cloudlet j if cloudlet j has adequate computing resource for the retraining of model M_m. The detailed online algorithm for the cumulative utility maximization problem is given in `Algorithm` 3.

D. Algorithm Analysis

The rest is to analyze an important property of the proposed online algorithm `Algorithm` 3, show that the solution delivered by `Algorithm` 3 is feasible, and analyze the time complexity of the algorithm as follows.

Lemma 1: For `Algorithm` 3, given a time slot $t \in \mathbb{T}$, if the residual computing resource in a cloudlet meets the amount of computing resource demanded by model $M_m \in \mathcal{M}(t) \backslash \mathcal{M}^{train}(t)$ for its retraining at time slot t and $net\_gain(M_m, t) > 0$, then model M_m can be chosen for retraining at the time slot; otherwise, model M_m will not be chosen for retraining in the rest of any time slot $t' \in \mathbb{T}$ with $t' \ge t$.

Proof: As $\Delta u_{m,l}$ is non-negative by Eq. (14) and Eq. (15), $net\_gain(M_m, t) \le 0$ implies that $I_{m,l} \ge |\mathbb{T}| - t + 1$. In other words, if model M_m is chosen for retraining at time slot t with $net\_gain(M_m, t) \le 0$, its retraining cannot finish in the end of time horizon $\mathbb{T}$. Consequently, there is no improvement on service accuracy of M_m by this retraining.

Theorem 3: Given an MEC network $G = (N, E)$, a given time horizon $\mathbb{T}$, a set V of mobile devices under the coverages of APs in N, and a set $\mathcal{M}$ of service models, assuming that both DTs and service models have been placed into cloudlets already, there is an online algorithm `Algorithm` 3 for the cumulative utility maximization problem that delivers a feasible solution, and the algorithm takes $O((|V| + |N| \cdot L_{\max})^3 + |\mathcal{M}| \cdot \log|\mathcal{M}|)$ time at each time slot, where $L_{\max} = \max\{L_j \mid j \in N\}$.

Proof: The set of mobile devices chosen for their DT updating at each time slot t by `Algorithm` 3 is feasible, due to the fact that at most one edge in $M_B(t)$ is incident to each mobile device $v_i \in V$, i.e., v_i is allocated to an AP within its transmission range via the construction of $G_B(t)$. Also, at most L_j mobile devices under the coverage of AP $j \in N$ are chosen to upload via AP j at each time slot t. Since there are exactly L_j corresponding nodes in $G_B(t)$, the endpoints of at most L_j matched edges in $M_B(t)$ are these nodes. Thus, no more than L_j devices under the coverage of AP j are chosen for uploading at time slot t. The total weight of all edges in $M_B(t)$ is the maximum one, i.e., the cumulative utility gain obtained by the chosen mobile devices at time slot t is the maximum one. Meanwhile, at each time slot t, choosing a service model for retraining by `Algorithm` 3 takes into account both the net utility gain and the amount of resource consumed by the

Algorithm 3: Online Algorithm for the Cumulative Utility Maximization Problem.

```
Input: An MEC network $G = (N, E)$, a time horizon $\mathbb{T}$, a set $V$ of mobile devices with $DT_i$ of each $v_i \in V$ having been placed in a cloudlet $h(DT_i)$, a set $\mathcal{M}$ of service models with each model $M_m$ having been deployed in the cloudlet $h(M_m)$ by invoking Algorithm 1.
Output: A subset $\mathcal{M}^{train}(t)$ of models in $\mathcal{M}$ for retraining at each time slot $t$, using their updated source DT data.
1:  $\mathcal{P} \leftarrow \emptyset$; /* the solution of the problem */
2:  for each model $M_m \in \mathcal{M}$ do
3:      $l_m \leftarrow 0$; /* index of the last retraining of $M_m$ */
4:      $\hat{t}_{m,l_m} \leftarrow 0$; $I_{m,l_m} \leftarrow 0$; $u_{m,l_m} \leftarrow 0$; $u_m(0) \leftarrow 0$;
5:  end for ;
6:  $\mathcal{M}(0) \leftarrow \emptyset$; $\mathcal{M}^{pot}(0) \leftarrow \emptyset$; $\mathcal{M}^{train}(0) \leftarrow \emptyset$;
7:  for each cloudlet $j \in N$ do
8:      Identify the amount $C_j^{res}$ of remaining resource on cloudlet $j$ after DT and model placements;
9:      $C_j^{res}(0) \leftarrow C_j^{res}$;
10: end for ;
11: $t \leftarrow 1$;
12: while $t \leq |\mathbb{T}|$ do
13:     Choose a subset $V(t)$ of mobile devices for DT updating at time slot $t$ by invoking Algorithm 2;
14:     Obtain a subset $\mathcal{M}(t)$ of models, where at least one source DT of each model $M_m \in \mathcal{M}(t)$ has been updated since its last retraining;
15:     for each model $M_m \in \mathcal{M}^{train}(t-1)$ do
16:         if $t-1$ is its last time slot for retraining then
17:             $M^{train}(t) \leftarrow M^{train}(t-1) \setminus \{M_m\}$;
18:             $j \leftarrow h(M_m)$; $C_j^{res}(t) \leftarrow C_j^{res}(t-1) + comp_m$;
19:             $u_m(t) \leftarrow u_m(t-1) + \Delta u_{m,l_m}$;
20:         else
21:             $u_m(t) \leftarrow u_m(t-1)$;
22:         end if ;
23:     end for ;
24:     for each model $M_m \in \mathcal{M} \backslash \mathcal{M}^{train}(t-1)$ do
25:         $u_m(t) \leftarrow u_m(t-1)$;
26:     end for
27:     $\mathcal{M}^{pot}(t) \leftarrow \mathcal{M}(t) \setminus \mathcal{M}^{train}(t)$;
28:     for each model $M_m \in \mathcal{M}^{pot}(t)$ do
29:         Calculate $r(M_m, t)$ of model $M_m$ by assuming that time slot $t$ is the start time slot of the $(l_m + 1)$th retraining of model $M_m$;
30:     end for ;
31:     Sort all potential retraining models in $\mathcal{M}^{pot}(t)$ in a decreasing order of $r(M_m, t)$ of each model $M_m$;
32:     for each model $M_m$ in the sorted sequence do
33:         if cloudlet $j = h(M_m)$ with residual computing resource $C_j^{res}(t) \geq comp_m$ then
34:             Choose model $M_m$ for retraining at time slot $t$;
35:             $l_m \leftarrow l_m + 1$; $\hat{t}_{m,l_m} \leftarrow t$;
36:             Calculate $I_{m,l_m}$, $\Delta u_{m,l_m}$, and $u_{m,l_m}$;
37:             $\mathcal{M}^{train}(t) \leftarrow \mathcal{M}^{train}(t) \cup \{M_m\}$;
38:             $\mathcal{M}^{pot}(t) \leftarrow \mathcal{M}^{pot}(t) \setminus \{M_m\}$;
39:             $C_j^{res}(t) \leftarrow C_j^{res}(t) - comp_m$;
40:         end if ;
41:     end for ;
42:     $\mathcal{P} \leftarrow \mathcal{P} \cup \{\langle V(t), \mathcal{M}^{train}(t), t \rangle\}$;
43:     $t \leftarrow t + 1$;
44: end while ;
45: returnSolution $\mathcal{P}$.
```

model retraining, provided that the computing resource capacity on each cloudlet will not be violated. Therefore, the solution delivered by Algorithm 3 is feasible.

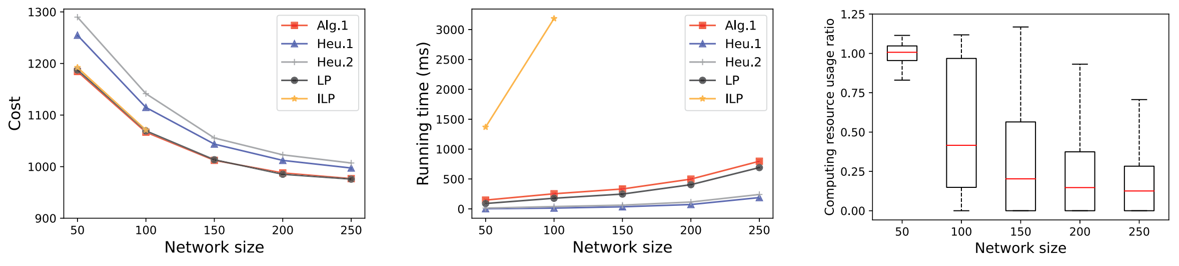
The time complexity of Algorithm 3 is analyzed as follows. Recall that all service models have already been deployed into cloudlets. At each time slot t, Algorithm 3 schedules mobile devices to upload by invoking Algorithm 2 through an auxiliary bipartite graph $G_B(t)$. The construction of $G_B(t)$ takes $O((L_{\max} \cdot |N| \cdot |V|)^3)$ time because $G_B(t)$ contains no more than $(L_{\max} \cdot |N| + |V|)$ nodes and $(L_{\max} \cdot |N| \cdot |V|)$ edges, and its edge weight assignment takes $O(|N|^3)$ time, due to finding all pairs of shortest paths in G. Finding a weighted maximum matching in G_B then takes $O((L_{\max} \cdot |N| + |V|)^3)$ time by the Hungarian algorithm. Algorithm 2 for mobile device uploading at each time slot thus takes $O((L_{\max} \cdot |N| + |V|)^3)$ time. Then, Algorithm 3 chooses service models in $\mathcal{M}$ for retraining at time slot t, which takes $O(|\mathcal{M}| \cdot \log |\mathcal{M}|)$ time for calculating $r(M_m, t)$ for each model M_m and sorting all models in a decreasing order of their value of $r(M_m, t)$. Algorithm 3 then takes $O(|\mathcal{M}|)$ time for choosing service models for retraining at each time slot t, subject to computing capacity on each cloudlet. Algorithm 3 thus takes $O((|V| + |N| \cdot L_{\max})^3 + |\mathcal{M}| \cdot \log |\mathcal{M}|)$ time for jointly choosing mobile devices for uploading and service models for retraining at each time slot $t \in \mathbb{T}$. The theorem then follows.

VI. Performance Evaluation

In this section, we evaluated the performance of the proposed algorithms for the two defined problems through simulations. Also, we studied impacts of important parameters on the performance of the proposed algorithms.

A. *Experimental Environment Settings*

Consider an MEC network $G = (N, E)$ generated by GT-ITM [4], where the number $|N|$ of APs is set as 50. The bandwidth capacity on each AP is randomly drawn in [5, 20] MHz [10], [21], and the bandwidth allocation to mobile devices under the coverage of an AP is performed by the OFDMA scheme, assuming that the number of sub-channels on each AP is randomly drawn between 5 and 10. The computing capacity on the cloudlet co-located with an AP is randomly drawn in [2,000, 4,000] MHz [28], and the cost of unit computing resource consumption on a cloudlet is randomly drawn within [0.015, 0.025]\$ [30]. The amount of computing resource demanded by deploying a model instance is randomly drawn in [100, 200] MHz [13]. The amount of computing resource needed by a service model for its retraining is randomly drawn in [400, 600] MHz. The transmission delay by transmitting 1 MB data along a link is within [0.2, 1] ms [8], and the transmission cost of transmitting 1 MB data along a link varies from 0.01\$to 0.04\$ randomly [28]. There are 2,000 mobile devices in the network. The transmission power of each mobile device is randomly drawn from 10 dBm to 20 dBm [14]. The volume of update data of a mobile device is randomly drawn in [1,5] MB, and the compression ratio ξ of the update data after merging with its DT data is 0.5. The data processing rate of a DT is randomly drawn in [10,20] MB per ms [28]. The retraining rate of a model is randomly drawn in [10,15] MB per millisecond. Assuming that there are 500 service models, the number of DT sources of a service model varies from 10 to 20 randomly.



(a) Performance (b) Running time (c) Resource violation ratio of `Alg. 1`

Fig. 3. Performance of different comparison algorithms for the model instance placement problem.

A sub-modular non-decreasing function $f(x) = \log_2(x+1)$ in Eq. (14) is adopted to measure the fidelity contribution of update data of a mobile device v_i to model M_m, and the weight $\omega_{i,m}$ of mobile device $v_i \in V_m$ to model M_m is set to $\omega_{i,m} = \frac{1}{|V_m|}$ in the default setting [11]. The time horizon contains 30 time slots with each lasting 50 ms. The ILP formulation is solved by the IBM ILOG CPLEX Optimizer [6].

We referred to `Algorithm` 1 for the model instance placement problem as `Algorithm 1`, and evaluated its performance against the following four benchmarks. (1) The ILP solution in (20) that will deliver an optimal solution to the problem when the problem size is small, otherwise its running time is prohibitively high. (2) The LP solution is the linear relaxation of the ILP, which delivers a lower bound on the optimal ILP solution (for a minimization problem) in polynomial time. (3) `Heu.1` deploys each model M_m into a cloudlet $j \in N$ with sufficient remaining computing resource and the least expected model placement cost $\overline{cost}(M_m, j)$ greedily. (4) `Heu.2` places service models one by one greedily. For each model, choose a cloudlet with sufficient computing resource for it so that the expected model placement cost is the minimum one. Among all pairs of yet-to-placed models and cloudlets, `Heu.2` chooses a pair with the minimum expected placement cost. This procedure continues until all models are deployed. As `Algorithm 1` delivers an optimal solution to the problem at the expense of moderate resource violations on cloudlets, we set the computing resource budget $\frac{1}{2}C_j$ on each cloudlet j for model instance placements in the default setting.

We referred to `Algorithm` 3 for the cumulative utility maximization problem as `Algorithm 3`, and evaluated its performance against the following two benchmarks. `Online.1`: At each time slot t, it always chooses a mobile device v_i with the largest $\Delta utility(v_i, t)$ for uploading greedily. For a mobile device that is covered by multiple APs, it is assigned to the AP with the largest number of unassigned sub-channels at that moment. This procedure continues until no more sub-channels can be allocated to mobile devices. It then greedily chooses models for retraining based on their utility gain after retraining, where for each chosen model, at least one of its source DT data has been updated since the last retraining. `Online.2`: At each time slot t, each AP j first chooses up to L_j mobile devices under its coverage randomly to upload their update data. It then chooses a subset of models for retraining the same way as `Online.2`.

The value in each figure is the average of the results delivered by each algorithm in 30 different MEC network topological instances with the same network size, and the running time of each algorithm is obtained by a desktop equipped with 64 GB RAM and an Intel 8 Core i7 CPU operating at 3.60 GHz. The above default settings will be adopted unless otherwise specified.

B. Performance Evaluation of Algorithms for the Model Instance Placement Problem

We first evaluated the performance of `Algorithm 1` for the model instance placement problem, against those of the ILP, the LP, `Heu.1`, and `Heu.2` by varying the number of cloudlets from 50 to 250. Fig. 3 plots the performance and running time curves of the mentioned comparison algorithms, and the computing resource violation ratio by `Algorithm` 1. It can be seen from Fig. 3(a) that the performance curves of the ILP, the LP, and `Algorithm 1` are almost identical when the problem size is no greater than 100. We conducted the ILP solution only when the network size is set as 50 and 100, because its running time becomes quite long with a large problem size. This indicates that the solution delivered by `Algorithm 1` is the optimal, however, this is achieved at the expense of resource capacity violations as shown in Fig. 3(c). Fig. 3(a) indicates that the cost delivered by `Algorithm 1` is 94.4% of that delivered by `Heu.1`, and 91.8% of that delivered by `Heu.2` when the network size is set at 50. While the network size is 250, the cost delivered by `Algorithm 1` is nearly 97.9% of that delivered by `Heu.1`, and 96.8% of that delivered by `Heu.2`. It shows that `Algorithm 1` can utilize resource efficiently when the resource is scarce. It can also be seen from Fig. 3(b) that the ILP takes the longest running time, and its running time is prohibitively high when the problem size is no less than 100, while the running times of `Algorithm 1` and the LP are comparable. Notice that the solution delivered by the LP is infeasible in most cases as it serves as an ideal benchmark for the performance of `Algorithm 1`. It is noted that `Algorithm 1` takes a longer running time than those of `Heu.1` and `Heu.2`. Fig. 3(c) plots resource violation ratios of by `Algorithm` 1. It can be seen from Fig. 3(c) that there are moderate resource violations on a small number of cloudlets. With the increase on the network size, the violation ratio decreases, because more resources are available. Furthermore, there is no resource violation when the network size $|N|$ reaches 200.

We studied the impact of the number $|\mathcal{M}|$ of service models on the performance of `Algorithm 1`, by varying its value from 400 to 600 when the network size is fixed at 50. It is observed from Fig. 4(a) that `Algorithm 1` with $|\mathcal{M}| = 400$ outperforms itself with $|\mathcal{M}| = 600$ by 34.1%, and its running

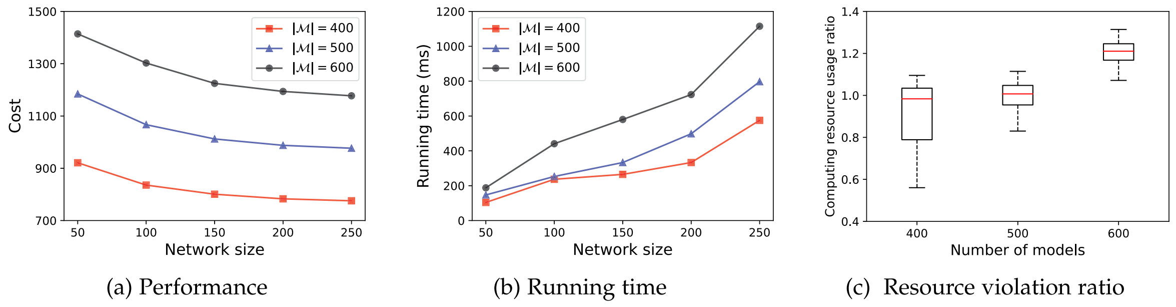

(a) Performance (b) Running time (c) Resource violation ratio

Fig. 4. Impact of the number $|\mathcal{M}|$ of service models on the performance of `Algorithm` 1.

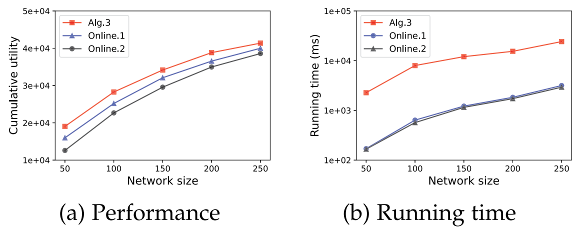

(a) Performance (b) Running time

Fig. 5. Performance of different algorithms for the cumulative utility maximization problem.

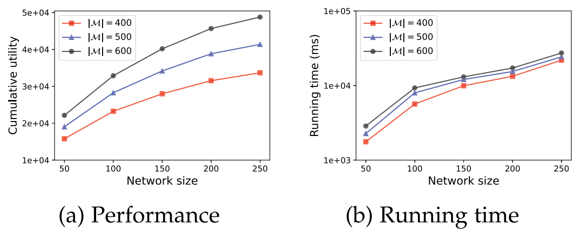

(a) Performance (b) Running time

Fig. 6. Impact of the number $|\mathcal{M}|$ of service models on the performance of `Algorithm` 3.

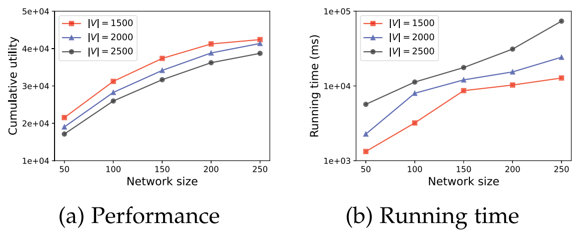

(a) Performance (b) Running time

Fig. 7. Impact of the number $|V|$ of mobile devices on the performance of `Algorithm` 3.

time is shown in Fig. 4(b). The violation ratios of computing resource are plotted in Fig. 4(c) when the network size is set at 50. With the growth of $|\mathcal{M}|$, the violation ratio also increases, as more resource is needed to accommodate more different model instances.

C. Performance Evaluation of Different Algorithms for the Cumulative Utility Maximization Problem

We then evaluated the performance of `Algorithm 3` for the cumulative utility maximization problem against `Online.1` and `Online.2`, by varying the network size from 50 to 250. Fig. 5 shows the performance and running time curves of the three comparison algorithms. It can be seen from Fig. 5(a) that `Algorithm 3` outperforms `Online.1` by 19.1%, and outperforms `Online.2` by 26.9% when the network size is set at 50. When the network size is set at 250, the cumulative utilities delivered by `Online.1` and `Online.2` are 95.8% and 92.5% of the one by `Algorithm 3`, respectively. The reason behind is that `Algorithm 3` can utilize communication resource efficiently when the resource is scarce. It can also be seen from Fig. 5(b) that `Algorithm 3` takes the longest running time among the comparison algorithms.

D. Impacts of Parameters on the Performance of the Proposed Online Algorithm

Finally, we studied impacts of important parameters on the performance of `Algorithm 3`. We considered the impact of the number $|\mathcal{M}|$ of models on the performance of `Algorithm 3` by varying $|\mathcal{M}|$ from 400 to 600. It is observed from Fig. 6(a) that the performance of `Algorithm 3` with $|\mathcal{M}| = 400$ is only 68.9% of itself with $|\mathcal{M}| = 600$, when the network size is set at 250. The running time of `Algorithm 3` is shown in Fig. 6(b).

We also studied the impact of the number $|V|$ of mobile devices on the performance of `Algorithm 3`, by varying it from 1,500 to 2,500. As shown in Fig. 7(a), the performance of `Algorithm 3` when $|V| = 1,500$ outperforms itself when $|V| = 2,500$ by 9.6%, when the network size is set at 250. It can be seen from Fig. 7(b) that the running time of `Algorithm 3` grows with the number of mobile devices.

VII. Conclusion

In this paper, we studied fidelity-aware inference service provisioning in a DT-assisted MEC network, through jointly choosing mobile devices to upload and scheduling service models to train using the updated data at each time slot over a given time horizon. To this end, we first considered the service model instance placement problem with the aim of minimizing the placement cost, and formulated an ILP solution to it when the problem size is small; otherwise we devised an approximation algorithm to the problem. We then investigated the cumulative utility maximization problem over a given time horizon, and developed an efficient online algorithm for it. We finally evaluated the performance of the proposed algorithms via simulations. Simulation results demonstrate that the proposed algorithms are promising and outperform the mentioned benchmarks.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[2] A. E. Azzaoui, S. R. Jeremiah, N. N. Xiong, and J. H. Park, “A digital twin-based edge intelligence framework for decentralized decision in IoV system,” *Inf. Sci.*, vol. 649, 2023, Art. no. 119595.

[3] 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 Trans. Mobile Comput.*, vol. 24, no. 4, pp. 2979–2994, Apr. 2025.

[4] GT-ITM, 2019. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[5] Y. Gong, H. Yao, Z. Xiong, C. L. P. Chen, and D. Niyato, “Blockchain-aided digital twin offloading mechanism in space-air-ground networks,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 183–197, Jan. 2025.

[6] IBM ILOG CPLEX Optimizer, 2024. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer

[7] J. Li et al., “Digital twin-assisted, SFC-enabled service provisioning in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 393–408, Jan. 2024.

[8] J. Li et al., “AoI-aware user service satisfaction enhancement in digital twin-empowered edge computing,” *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1677–1690, Apr. 2024.

[9] J. Li et al., “Digital twin-enabled service provisioning in edge computing via continual learning,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 7335–7350, Jun. 2024.

[10] 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 Trans. Mobile Comput.*, vol. 22, no. 5, pp. 3017–3030, May 2023.

[11] J. Li, W. Liang, J. Wang, and X. Jia, “Cumulative fidelity maximization of inference services in DT-assisted edge computing,” in *Proc. IEEE 1st Int. Conf. Meta Comput.*, 2024, pp. 1–10.

[12] Y. Li, W. Liang, Z. Xu, W. Xu, and X. Jia, “Budget-constrained digital twin synchronization and its application on fidelity-aware queries in edge computing,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 165–182, Jan. 2025.

[13] W. Liang, Y. Ma, W. Xu, Z. Xu, X. Jia, and W. Zhou, “Request reliability augmentation with service function chain requirements in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4541–4554, Dec. 2022.

[14] X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia, “Multiple service model refreshments in digital twin-empowered edge computing,” *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2672–2686, Sep./Oct. 2024.

[15] H. Liao, Y. Shu, J. Lu, Z. Zhou, M. Tariq, and S. Mumtaz, “Integration of 6G signal processing, communication, and computing based on information timeliness-aware digital twin,” *IEEE J. Sel. Topics Signal Process.*,vol. 18, no. 1, pp. 98–108, Jan. 2024.

[16] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, “Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach,” *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2402–2416, Apr. 2023.

[17] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, “Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.

[18] D. Shmoys and E. Tardos, “An approximation algorithm for the generalized assignment problem,” *Math. Program.*, vol. 62, pp. 461–474, 1993.

[19] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital twin in industry: State-of-the-art,” *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[20] Z. Wang, D. Jiang, and S. Mumtaz, “Network-wide data collection based on in-band network telemetry for digital twin networks,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 86–101, Jan. 2025.

[21] Z. Xu et al., “Energy-aware inference offloading for DNN-driven applications in mobile edge clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 4, pp. 799–814, Apr. 2021.

[22] Z. Xu et al., “Energy-aware collaborative service caching in a 5G-enabled MEC with uncertain payoffs,” *IEEE Trans. Commun.*, vol. 70, no. 2, pp. 1058–1071, Feb. 2022.

[23] Y. Yang et al., “Dynamic human digital twin deployment at the edge for task execution: A two-timescale accuracy-aware online optimization,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 12262–12279, Dec. 2024.

[24] Z. Zhou and J. Abawajy, “Reinforcement learning-based edge server placement in the intelligent Internet of Vehicles environment,” *IEEE Trans. Intell. Transp. Syst.*, to be published, 2025, doi: 10.1109/TITS.2025.3557259.

[25] R. Zhang, Z. Xie, D. Yu, W. Liang, and X. Cheng, “Digital twin-assisted federated learning service provisioning over mobile edge networks,” *IEEE Trans. Comput.*, vol. 73, no. 2, pp. 586–598, Feb. 2024.

[26] Y. Zhang and W. Liang, “Cost-aware digital twin migration in mobile edge computing via deep reinforcement learning,” in *Proc. 23rd IFIP/IEEE Netw. Conf.*, 2024, pp. 441–447.

[27] Y. Zhang, L. Wang, and W. Liang, “Deep reinforcement learning for mobility-aware digital twin migrations in edge computing,” *IEEE Trans. Serv. Comput.*, vol. 18, no. 2, pp. 704–717, Mar./Apr., 2025.

[28] Y. Zhang, W. Liang, Z. Xu, and X. Jia, “Mobility-aware service provisioning in edge computing via digital twin replica placements,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11295–11311, Dec. 2024.

[29] Y. Zhang, W. Liang, Z. Xu, X. Jia, and Y. Yang, “Profit maximization of delay-sensitive, differential accuracy inference services in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 24, no. 7, pp. 6209–6224, Jul. 2025.

[30] Y. Zhang, W. Liang, Z. Xu, W. Xu, and M. Chen, “AoI-aware inference services in edge computing via digital twin network slicing,” *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3154–3170, Nov./Dec. 2024.

[31] Y. Zhang, W. Liang, W. Xu, Z. Xu, and X. Jia, “Cost minimization of digital twin placements in mobile edge computing,” *ACM Trans. Sensor Netw.*, vol. 20, no. 3, pp. 1–26, 2024.

Xuan Ai received the BS degree in statistics from the University of Science and Technology of China in 2022. She is currently working toward the PhD degree with the City University of Hong Kong. Her research interests include cloud and edge computing, serverless computing, and edge intelligence.

Weifa Liang (Senior Member, IEEE) received the PhD degree in computer science from the Australian National University in 1998. He is a full professor with the Department of Computer Science, City University of Hong Kong (CityUHK). He was a full professor with the Australian National University, prior to joining CityUK in 2021. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Mobile Edge Computing (MEC), machine learning, Network Function Virtualization (NFV), Internet of Things and digital twins, design and analysis of parallel and distributed algorithms, approximation algorithms, and graph theory. He currently serves as an editor for *IEEE Transactions on Communications*.

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

Wenzheng Xu (Member, IEEE) received the BSc, ME, and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He currently is a full professor with Sichuan University, China. He was a visitor with both the Australian National University, Australia and the Chinese University of Hong Kong, Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.