Accumulative Fidelity Maximization of Inference Services in DT-Assisted Edge Computing

Jing Li, Weifa Liang, Jianping Wang, and Xiaohua Jia
Department of Computer Science, City University of Hong Kong, Hong Kong, P. R. China

***Abstract*—Digital twin (DT) technology illuminates the smooth integration of cyber and physical worlds in alignment with the Industry 4.0 initiative. Through synchronizations with physical objects, DTs of objects can reflect the states of the objects with high fidelity. Machine learning-based service models through continual training by the DT update data can provide users with accurate inference services. Orthogonal to the DT technology, mobile edge computing (MEC) is a promising computing paradigm that shifts computing power to the edge network, which is particularly appropriate for delay-sensitive intelligent services. In this paper, we study the fidelity enhancement of service models in a DT-assisted edge computing environment empowered by 6G communication, through continuously training service models, using DT update data obtained from their mobile IoT devices (objects) in a real-time manner. To this end, we first formulate an accumulative fidelity maximization problem that jointly considers the placement of DTs and models, with the aim to maximize the accumulative fidelity gain of all models while minimizing the total cost of resource consumed due to DT updating and service model training. We then develop an efficient algorithm for a sub-optimization problem – the placement problem of DTs and models, under the assumption that the mobility profile of each mobile object is given. When both DTs and models have already been placed, we devise an algorithm for the accumulative fidelity maximization problem that schedules each object to choose an access point (AP) to upload its update data at each time slot for a given time horizon to maximize the accumulative fidelity of all service models while minimizing the total cost of various resources consumed. We finally evaluate the performance of the proposed algorithms through simulations. Simulation results indicate that the proposed algorithms are promising, and outperform their baselines nearly by 20%.**

I. Introduction

Accompanied by the widespread deployment and rapid adoption of 5G network, significant research endeavours are underway to drive the development of the next-generation network via the Internet of Things (IoT) towards a fully intelligent future world [3]. Holding immense potential in realizing the digitization goals of 6G, the concept of digital twins (DTs) is transitioning from a conceptual idea to a tangible simulation technology recently, and continues to garner significant interest across various fields, including healthcare, autonomous driving, and manufacturing [19]. A DT is a digital portrayal of a physical object, which enables diverse functionalities such as simulation, analysis, prediction and optimization, due to the advancements in technologies, such as artificial intelligence and big data analytics [12]. DTs offer an opportunity to portray and model the physical world in a virtualized manner, serving as a connection between the physical and virtual realms [9].

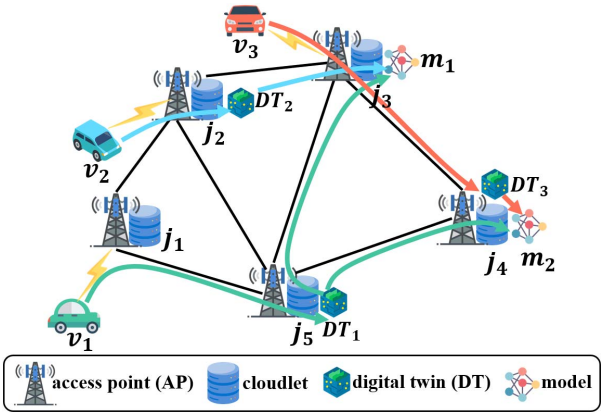


Fig. 1. An illustrative example of a DT-assisted edge computing network, where each access point (AP) has a co-located cloudlet. Objects v_1, v_2 and v_3 have their DTs, DT_1, DT_2 and DT_3 deployed in cloudlets j_5, j_2, and j_4, respectively. There are two service models m_1 and m_2, where m_1 consists of two attributes (DT_1 and DT_2), and m_2 consists of two attributes (DT_1 and DT_3), respectively.

Mobile edge computing (MEC) presents an excellent solution for building DTs to enhance intelligent service in future 6G networks, through bringing data processing and service model training to the network edge, in the proximity of users and IoT devices [11]. DTs are able to record the dynamic state information of their physical counterparts in real-time, while MEC facilitates ubiquitous low-latency intelligent services to handle the high update frequency of DTs with large amounts of update data, and boost data processing efficiency.

Model-driven inference service provisioning in MEC environments has emerged as a recent prominent research focus [13]. In this work, we consider an IoT application scenario as illustrated in Fig. 1, where there are multiple service models providing different inference services, and each service model consists of multiple attributes. To maintain their accuracy and high fidelity, service models are required to be continuously trained, using the DT update data of objects as their attributes. One such example is an autonomous driving model. Vehicle movement behaviours and speed are purely determined by the service model that consists of multiple sensors sensing the surroundings of the vehicle, roadside information and installed video.

Most service models, including the above example of autonomous driving, are required to be trained continuously, using the update data from their sources to maintain their service fidelity. Then how to maintain high fidelity of ser-

2024 International Conference on Meta Computing (ICMC) | 979-8-3503-5599-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICMC60390.2024.00014

979-8-3503-5599-4/24/$31.00 ©2024 IEEE
DOI 10.1109/ICMC60390.2024.00014

vice models while utilizing limited computing and storage resources in edge computing is challenging.

First, as the DTs of different objects provide the input data of service models for model training, what is the impact of the DT update data on the fidelity of service models? and how to measure the fidelity contribution of the update data of each object to its service models?

Second, since objects move around in the network, both DT locations and model locations determine the DT and model updating costs which include uploading the update data from objects, transmitting the update data to DTs and models, and training the models using the update data. How to place DT and service models in cloudlets without violating their computing capacities is challenging.

Thirdly, due to the uncertainty on the volume of the update data generated by each IoT device since its last uploading, accurately predicting the volume of the update data of the IoT device is challenging.

Finally, because of the limited bandwidth of APs, usually only a subset of IoT devices are able to upload their update data through APs at each time slot. Then, how to determine which IoT devices can upload their update data at each time slot to jointly maximize the accumulative fidelity gain and minimize the updating cost? The aforementioned challenges will be addressed in the rest of this paper.

The novelty of this study lies in formulating a novel accumulative fidelity maximization problem in DT-assisted edge computing environments. We introduce a novel metric to measure the fidelity of service models with the aim to strive for non-trivial trade-offs between the accumulative fidelity gain of all models and the total cost of various resource consumed to achieve the model fidelity. We develop efficient algorithms for DT and model deployments, as well as the scheduling of uploading the update data of IoT devices for model training.

The main contributions of this paper are summarized as follows.

- We investigate the fidelity enhancement of service models in a DT-assisted edge computing environment, through continuously training service models, by leveraging DT update data. We develop novel metrics to measure the model fidelity and conduct cost modelling on various resource consumptions.
- We formulate an accumulative fidelity maximization problem by placing DTs of objects and service models to cloudlets, with the aim to maximize the accumulative fidelity gain of all models while minimizing the total cost of various resources consumed on the improvement of model fidelities.
- We develop an efficient algorithm to place DT and service models, by reducing to a series of minimum-cost maximum matchings in auxiliary bipartite graphs.
- Under the assumption that all DTs and models have been placed, we devise an algorithm to schedule mobile objects to upload their update data at each time slot to maximize the accumulative fidelity gain of all models.
- We evaluate the algorithm performance through simulations. Simulation results demonstrate that the proposed algorithms are promising, and outperform the baselines nearly by 20%.

The remainder of the paper is as follows. Section II surveys the related work. Section III introduces the system model and problem definition. Section IV focuses on the development of an efficient algorithm for DT and model placements. Section V devises an online algorithm for the accumulative fidelity maximization problem, by scheduling IoT devices to upload their update data, based on the assumption that both DTs and service models have been placed. Section VI provides simulation results, and Section VII concludes the paper.

II. Related work

DT technology has been envisioned as a promising technique for intelligent services. For example, Hui *et al.* [4] established the DT for each vehicle and designed a DT-enabled collaborative and distributed autonomous driving scheme. Li *et al.* [8], [10] devised approximation and online algorithms for dynamic DT placements in an MEC network with the mobility assumption of objects while ensuring services on DT data with low age of information (AoI). Li *et al.* [6] also studied reliability-aware SFC placements in MEC with leveraging DTs to predict the reliability of service function instances. Liang *et al.* [13] investigated the correlation between the freshness of a service model and the AoIs of DT data for the model training. They devised an efficient algorithm for minimizing the resource consumption cost to achieve the model freshness. Lin *et al.* [15] developed a congestion control scheme to ensure the stability of long-term DT services, by using the Lyapunov optimization in order to maximize the profit. Ren *et al.* [20] presented a congestion control scheme for DT edge networks and developed a deep reinforcement learning (DRL) method for performance prediction of the physical network. Zhang *et al.* [25] leveraged DTs to improve device scheduling, intending to maximize the utility of federated learning (FL) services. They developed efficient approaches for offline multi-FL services, as well as a DRL algorithm under dynamic conditions settings. Liao *et al.* [14] investigated DT-empowered resource management in 6G networks, and formulated a joint optimization problem for energy management of electric vehicles, and developed an AoI-aware DRL joint optimization algorithm for signal, communication and computing resources that strives for balance between the fidelity of service models and differential level consistency of DTs with their objects. Zhang *et al.* [26] considered mobility-aware, delay- sensitive inference service provisioning in a DT-assisted MEC network under the mobility of both objects and users via DT replica placements. They developed efficient algorithms for minimizing service costs while meeting service delay requirements. Zhao *et al.* [27] developed a hierarchical routing strategy in a DT-assisted vehicular edge network for providing services to vehicle users. Vaezi *et al.* [23] proposed algorithms for the DT deployment problem to reduce the maximum response delay while meeting user AoI requirements. Shu *et al.* [22] dealt with

DT-assisted resource management through energy dispatching and control model training under the constraint on long-term AoI. They proposed a DT-assisted federated learning scheme for the problem, by adopting the Lyapunov optimization.

However, none of the aforementioned work considered non-trivial trade-offs between the fidelity gain of models and the cost of resource consumption on the continuous training of service models, using the update data of objects in a real-time manner. This paper focuses on placing DTs and service models to cloudlets effectively to enhance the accumulative fidelity gain of all service models for a given monitoring period while minimizing the total cost of various resource consumed for DT and model updatings.

III. Preliminaries

A. System model

Consider an MEC network $G = (N, E)$ that consists of a set N of access points (APs) and a set E of optic links between APs. Each AP has a co-located cloudlet and the communication delay between them is negligible. Each AP $j \in N$ has bandwidth capacity B_j and its co-located cloudlet has computing capacity C_j. In this paper, we assume that an AP and its co-located cloudlet are interchangeable if no confusion arises.

We consider a finite time horizon $\mathbb{T}$ with T equal time slots. There is a set V of mobile IoT devices (objects) under the coverage of APs. Associated with each object $v_i \in V$, there is a DT, DT_i, to be placed in a cloudlet, which is a virtual representation of the object. The data generated by object v_i will be uploaded at some time slots to DT_i, and the object will be synchronized with DT_i, by transmitting the update data from its uploading AP to the cloudlet hosting DT_i.

We further assume that the total amount of time needed for uploading the update data, processing the update data at their DTs, and continual training on service models is no greater than the duration of a time slot, i.e., the DT synchronization and model training can be achieved within each time slot.

B. Bandwidth allocation of IoT devices

Each object can be covered by a set of APs. We here adopt the OFDMA scheme at each AP, and suppose that each AP $j \in N$ has L_j orthogonal subchannels, i.e., at most L_j IoT devices can upload update data by AP j at each time slot due to its bandwidth capacity constraint [5].

Considering an object v_i accessing AP j at time slot t, then the data uploading rate of v_i at time slot t is

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

where B_j is the bandwidth of AP j, PX_i is the transmission power of object v_i, $H_{i,j}$ is the channel gain between object v_i and AP j, and σ^2 is the noise power.

If object v_i at time slot t has a volume $vol(v_i, t)$ of update data to be uploaded to AP j, then its uploading time is

$$time_{upload}(v_i, j, t) = \frac{vol(v_i, t)}{b_{i,j}(t)}. \quad (2)$$

Because the number of objects (IoT devices) covered by each AP j usually is larger than its L_j subchannels, only some of the objects are chosen to upload their update data to their DTs, and the DT update data then are forwarded to the models in which they serve as the source data.

C. Fidelity gain of service models

Given a service model $m_k = (attr_1, \ldots, attr_{l_k})$ that consists of l_k attributes, where the data of each attribute is based on the DT data of its corresponding object, and each attribute $attr_l$ is associated with a given weight $w_{k,l}$ (≥ 0) that is its importance to the model, and $\sum_{l=1}^{l_k} w_{k,l} = 1$, which implies that a larger weight of an attribute plays a heavier impact on the fidelity of the model.

When an object v_i is the data source of multiple service models, if its update data is uploaded at time slot t, then it will impact the service fidelity (or service accuracy) of each of the models. In the following, we quantify the fidelity gain on each of the models by uploading the update data of object v_i at time slot t.

Denote by $\mathcal{V}(m_k)$ the set of l_k objects contributing to the attributes of model m_k, and $|\mathcal{V}(m_k)| = l_k$. Let $V(m_k, t) \subseteq \mathcal{V}(m_k)$ be a subset of objects whose DT data are updated at time slot t for model m_k. Supposing that object v_i is the lth attribute of model m_k, denote by $\rho_{k,l}(t)$ the accumulative data volume of DT_i generated by object v_i until time slot t that contributes to the lth attribute of model m_k, which is calculated as follows.

$$\rho_{k,l}(t) = \begin{cases} \rho_{k,l}(t-1) + vol(v_i, t), & \text{if } v_i \in V(m_k, t) \\ \rho_{k,l}(t-1), & \text{otherwise,} \end{cases}$$

where $v_i \in V(m_k, t)$ corresponds to an attribute of model m_k, and its update data is uploaded at time slot t. It can be seen that $\rho_{k,l}(t) \geq \rho_{k,l}(t-1)$, and $\rho_{k,l}(0) = 0$ initially for all l with $1 \leq l \leq l_k$.

In the following, we delve to measure the impact of the update data of object v_i on the models it serves. Let $Fid(m_k, t)$ be the fidelity of model m_k at time slot t with $Fid(m_k, 0) = 0$ initially. We make use of a submodular non-decreasing function $g(\cdot, \cdot, \ldots, \cdot)$ with l_k parameters to measure the fidelity of service model $m_k \in M$. Function $g(\cdot, \cdot, \ldots, \cdot)$ maintains the diminishing fidelity gain property, i.e., if $g(\cdot)$ is a 1-dimension function, we have $g(x + \delta x) - g(x) \leq g(x' + \delta x) - g(x')$ with $x' < x$ and $\delta x \geq 0$, e.g., $g(x) = \log_2(x+1)$.

Assuming that the multi-dimensional submodular function $g(\cdot, \cdot, \ldots, \cdot)$ can be expressed by a weighted sum of 1-dimensional submodular functions $f(\cdot)$, then the fidelity gain $\Delta Fid(m_k, v_i, t)$ of the lth attribute of model m_k induced by object $v_i \in \mathcal{V}(m_k)$ at time slot t is defined as follows.

$$\Delta Fid(m_k, v_i, t) = w_{k,l} \cdot (f(\rho_{k,l}(t)) - f(\rho_{k,l}(t-1))) \quad (3)$$

where $w_{k,l}$ is the weight of the lth attribute contributing to the fidelity of model m_k and $\sum_{l=1}^{l_k} w_{k,l} = 1$. The values of $\rho_{k,l}(t)$ and $\rho_{k,l}(t-1)$ are the accumulative volumes of the lth attribute of model m_k at time slots t and $t-1$, respectively.

The fidelity gain $\Delta Fid(m_k, t)$ of model m_k at time slot t is defined as follows.

$$\Delta Fid(m_k, t) = \sum_{v_i \in \mathcal{V}(m_k)} \Delta Fid(m_k, v_i, t) \tag{4}$$

D. Computing capacity constraints on cloudlets

The deployments of DTs or service models to cloudlets consume computing resource. let $comp(DT_i)$ and $comp(m_k)$ be the amounts of computing resource consumed by DT_i and model m_k, respectively. Each cloudlet n_j possesses a computing capacity C_j. We can see there is a capacity constraint posed on each cloudlet, i.e., the computing resource consumption of deployed DTs and models in a cloudlet cannot exceed its computing capacity.

E. Update cost of DTs and service models

Let $AP(v_i, t)$ be the set of APs covering object v_i at time slot t. *The uploading cost* of the update data of v_i located at AP $j \in AP(v_i, t)$ at time slot t is

$$cost_{upload}(v_i, j, t) = \xi_1 \cdot PX_i \cdot time_{upload}(v_i, j, t) \tag{5}$$

where ξ_1 is the cost for a unit power consumption, PX_i is the transmission power of device v_i, and $time_{upload}(v_i, j, t)$ is its data uploading duration that is defined in Eq. (2).

The data transmission cost of object v_i from AP j to its DT_i located at cloudlet $h(DT_i)$, is

$$cost_{trans}(v_i, j, h(DT_i), t) = \xi_2 \cdot vol(v_i, t) \cdot \sum_{e \in P_{j,h(DT_i)}} length(e) \tag{6}$$

where ξ_2 is the transferring cost of a unit data via a routing path with length 1, $P_{j,h(DT_i)}$ is the shortest path in the network G between AP j and AP $h(DT_i)$, and $length(e)$ is the length of link $e \in E$.

The DT processing cost of DT_i in cloudlet $h(DT_i)$ by adding the update data is

$$cost_{proc}(v_i, t) = \xi_3 \cdot vol(v_i, t) \tag{7}$$

where ξ_3 is the processing cost of a unit data by DT_i in $h(DT_i)$.

We define *the DT updating cost* $cost_{DT}(v_i, t)$ of DT_i at time slot t, which consists of the uploading cost, the data transmission cost, and the DT processing cost, i.e.,

$$cost_{DT}(v_i, t) = cost_{upload}(v_i, j, t) + cost_{trans}(v_i, j, h(DT_i), t) + cost_{proc}(v_i, t) \tag{8}$$

To train model m_k using the update data of its attributes, all update data from their DTs of m_k must be forwarded to the hosting cloudlet $h(m_k)$ of model m_k. Then *the data aggregation cost* of m_k in $h(m_k)$ for training through DT data aggregation is

$$cost_{agg}(m_k, h(m_k), t) = \xi_2 \cdot \sum_{v_i \in V(m_k, t)} vol'(v_i, t) \cdot \sum_{e \in P_{h(DT_i), h(m_k)}} length(e) \tag{9}$$

where $vol'(v_i, t)$ is the volume of the processed data from DT_i at time slot t, which will be sent to the models requiring the DT data of DT_i.

The continual training cost of m_k located at cloudlet $h(m_k)$ at time slot t is

$$cost_{train}(m_k, t) = \xi_3 \cdot \sum_{v_i \in V(m_k, t)} vol'(v_i, t) \tag{10}$$

where ξ_3 is the processing cost of a unit data by m_k in $h(m_k)$.

Suppose that object $v_i \in V(m_k, t)$ uploads its data through AP j, while the DT_i of object $v_i \in V(m_k, t)$ is located at cloudlet $h(DT_i)$, and model m_k is located at cloudlet $h(m_k)$. *The model updating cost* of m_k at time slot t for enhancing its fidelity thus is given as follows.

$$cost_{model}(m_k, t) = cost_{agg}(m_k, h(m_k), t) + cost_{train}(m_k, t). \tag{11}$$

F. Problem definition

The objective of the accumulative fidelity maximization problem in this paper is to maximize the accumulative fidelity gain of all models for a given monitoring period $\mathbb{T}$ while minimizing the total cost of various resources consumed, i.e.,

$$\text{Maximize} \sum_{t=1}^{T} \sum_{m_k \in M} \Delta Fid(m_k, t) - \omega \cdot \sum_{t=1}^{T} \Big(\sum_{v_i \in V} cost_{DT}(v_i, t) + \sum_{m_k \in M} cost_{model}(m_k, t) \Big) \tag{12}$$

where ω is a coefficient and $\Delta Fid(m_k, t)$ is the fidelity gain of model m_k at time slot t due to some of its attribute data updates. $cost_{DT}(v_i, t)$ and $cost_{model}(m_k, t)$ are the updating costs of DT v_i and model m_k at time slot t defined in Eq. (8) and (11), respectively.

Definition 1: Given an MEC network $G = (N, E)$ with $|N|$ APs (cloudlets), a time horizon $\mathbb{T}$, a set V of mobile objects, and a set M of service models to be trained continuously using the update data from objects, *the accumulative fidelity maximization problem* in G is to maximize the accumulative fidelity of service models while minimizing the total cost of various resource consumed for achieving the model fidelity during the given monitoring period $\mathbb{T}$, i.e., objective (12), through efficient placements of DTs and models, subject to computing capacity on each cloudlet and bandwidth capacity on each AP.

G. NP-hardness

Theorem 1: The accumulative fidelity maximization problem in a DT-assisted MEC network $G = (N, E)$ is NP-hard.

Proof The NP-hardness of the accumulative fidelity maximization problem is proven via reducing from the maximum-profit Generalized Assignment Problem (GAP), which is NP-hard [1]. In the GAP, there are a set of bins with each having a capacity, and a set of items with each having a weight and a profit. By packing items into bins, the maximum-profit GAP is to maximize the collected profit while respecting the bin capacities.

We consider a special case of the accumulative fidelity maximization problem with a single time slot, and both DTs and service models have been placed in cloudlets. Also, each object is covered by all APs and various resource consumption costs are ignored. We treat each AP j as a bin b_j with capacity L_j, i.e., the number of subchannels provided by the AP. We also treat each object $v_i \in V$ as an item with weight 1, i.e., the update data uploading of object v_i occupies one subchannel exclusively. The profit of assigning item $item_i$ to bin b_j is $\sum_{m_k \in \mathcal{M}(v_i)} \Delta Fid(m_k, v_i, 1)$, where $\mathcal{M}(v_i)$ is the set of models requiring the DT data of v_i. We observe that the special accumulative fidelity gain maximization problem is equivalent to the maximum-profit GAP. Thus, the accumulative fidelity maximization problem is NP-hard. ■

H. A sub-optimization problem: the DT and model placement problem

The accumulative fidelity maximization problem is a joint optimization problem, which jointly considers the bandwidth capacity on each AP, computing capacity on each cloudlet, computing resource demands by each DT and each model, and the costs of routing the update data from their uploading APs to their DTs and from their DTs to service models.

Considering the mobility of objects, the placements of both DTs and service models significantly impact the problem optimization objective, which makes the choices of cloudlets for DT and model placements become challenging. To tackle this joint optimization problem, in the following, we first study a sub-optimization – the DT and model placement problem. With the given placed DTs and models, we then can solve the problem of concern, the accumulative fidelity maximization problem, through choosing IoT devices to upload their update data at each time slot for model training, in order to maximize the optimization objective (12).

Since objects are movable in the network, it is challenging to place the DT of each object in an appropriate location to mitigate its subsequent updating cost when the object moves to other locations. However, it is observed that most objects do not move around all locations in the network evenly, and they instead usually visit only a very few locations. Let K be the number of frequent visiting locations of an object. We further assume that $K \ll |N|$. In the following, we make use of the top-K frequent visiting locations by each object as an approximate representation of its movement trajectory in the network. Notice that such top-K locations of each object can be found through mining the historical moving traces of the object from the data stored at its DT. Now, let $N(v_i)$ be the top-K visiting locations of object $v_i \in V$ and p_{ij} is the probability of object v_i visiting location $j \in N(v_i) \subset N$, i.e., $0 \leq p_{ij} \leq 1$ and $|N(v_i)| = K$. Suppose that $\sum_{j \in N(v_i)} p_{ij} \approx 1$, in other words, the total probability of object v_i visiting other locations $j \in N \setminus N(v_i)$ is negligible.

Having this approximate movement assumption for each object v_i, referring to Eq. (5), (6), (7), we define *the expected DT updating cost* of DT_i in a single time slot when DT_i is deployed in cloudlet $h(DT_i)$ as follows.

$$\overline{cost}_{DT}(v_i, h(DT_i)) = \sum_{j \in N(v_i)} p_{ij} \cdot (\xi_1 \cdot PX_i \cdot \frac{\widetilde{vol}(v_i, j)}{\widetilde{b}_{i,j}} + \xi_2 \cdot \widetilde{vol}(v_i, j) \cdot \sum_{e \in P_{j,h(DT_i)}} length(e) + \xi_3 \cdot \widetilde{vol}(v_i, j)), \quad (13)$$

where $\widetilde{vol}(v_i, j)$ is the average volume of the update data by object v_i at location $j \in N(v_i)$ with the average data uploading rate $\widetilde{b}_{i,j}$.

Similarly, assuming that each data source DT_i of model m_k, corresponding to an attribute of m_k, has been deployed in cloudlet $h(DT_i)$ already, we define *the expected model updating cost* of service model m_k when model m_k is deployed in cloudlet $h(m_k)$ as follows.

$$\overline{cost}_{model}(m_k, h(m_k)) = \sum_{v_i \in \mathcal{V}(m_k)} (\xi_2 \cdot \sum_{j \in N(v_i)} p_{ij} \cdot \widetilde{vol'}(v_i, j) \cdot \sum_{e \in P_{h(DT_i), h(m_k)}} length(e) + \xi_3 \cdot \sum_{j \in N(v_i)} p_{ij} \cdot \widetilde{vol'}(v_i, j) \quad (14)$$

where $\widetilde{vol'}(v_i, j)$ is the average volume of the processed data from DT_i based on the update data by object v_i located at location AP j.

The objective of the DT and model placement problem is to minimize the total expected cost of resources consumed in a single time slot by deploying DTs and models, i.e.,

$$\text{Minimize} \quad \sum_{v_i \in V} \overline{cost}_{DT}(v_i, h(DT_i)) + \sum_{m \in M} \overline{cost}_{model}(m_k, h(m_k)) \quad (15)$$

Definition 2: Given an MEC network $G = (N, E)$ with $|N|$ APs (cloudlets), a set V of mobile objects with given mobility profiles (i.e., the average volume of the update data $\widetilde{vol}(v_i, j)$ and the average volume of the processed data $\widetilde{vol'}(v_i, j)$ at one location j of the top-K locations with probability p_{ij}, the average uploading rate $\widetilde{b}_{i,j}$, and the amount $comp(DT_i)$ of computing resource demanded), there are a set M of service models to be placed and then trained continuously with the amount $comp(m_k)$ of computing resource demanded by model $m_k \in M$, using the update data from mobile objects for a monitoring period. *The placement problem of DTs and models* is to minimize the expected cost of resources consumed by deploying DTs and models, i.e., objective(15), through efficient placements of DTs of mobile objects and service models, subject to computing capacity on each cloudlet.

IV. Algorithm for the placement problem of DTs and models

A. Algorithm

The placement problem of DTs and models is a joint optimization problem that places both DTs and models to cloudlets. Since tackling this problem is very intriguing, we instead decompose it into two sub-problems: the DT placement problem, and the model placement problem based on the placed DTs. We approach both sub-problems through finding minimum-cost maximum matchings in a series of auxiliary bipartite graphs. Specifically, to determine the placements of DTs, we construct auxiliary bipartite graphs iteratively. Within iteration μ with $\mu \geq 1$, the DTs of some objects will be placed. To this end, we construct a bipartite graph $\mathbb{B}_{DT}(\mu) = (\mathbb{V}(\mu), \mathbb{N}(\mu), \mathbb{E}(\mu))$, where $\mathbb{V}(\mu)$ is the virtual node set of objects whose DTs have not been placed yet prior to iteration μ with $\mathbb{V}(1) = V$ initially, and $\mathbb{N}(\mu)$ is the virtual node set of cloudlets with residual computing resource at iteration μ. $\mathbb{E}(\mu)$ is a virtual edge set of $\mathbb{B}_{DT}(\mu)$ at iteration μ, where there is an edge $(v_i, j) \in \mathbb{E}(\mu)$ in the bipartite graph $B_{DT}(t)$ between an object $v_i \in \mathbb{V}(\mu)$ and a cloudlet $j \in \mathbb{N}(\mu)$ with weight $\overline{cost}_{DT}(v_i, j)$ by Eq. (13) if cloudlet j has enough residual computing resource to host DT_i.

Denote by $Match_{DT}(\mu)$ a minimum-cost maximum matching in bipartite graph $\mathbb{B}_{DT}(\mu)$, which can be found by applying the Hungarian algorithm. Then, for each edge $(v_i, j) \in Match_{DT}(\mu)$, DT_i of object v_i is placed to cloudlet j with the amount $comp(DT_i)$ of computing resource consumed, and the residual computing resource of cloudlet j then is reduced by $comp(DT_i)$. This procedure continues until the DTs of all objects are deployed in cloudlets.

Having placed all DTs, we construct a series of auxiliary bipartite graphs to place service models in M similarly. That is, within each iteration $\mu \geq 1$, we construct a bipartite graph $\mathbb{B}_{model}(\mu) = (\mathbb{M}(\mu), \mathbb{N}'(\mu), \mathbb{E}'(\mu))$. Let $\mathbb{M}(\mu)$ be the virtual node set of models that have not been deployed prior to iteration μ, and $\mathbb{M}(1) = M$ initially. $\mathbb{N}'(\mu)$ is the virtual node set of cloudlets with residual computing resource at iteration μ, and $\mathbb{E}'(\mu)$ is the virtual edge set of $\mathbb{B}_{model}(\mu)$, where there is an edge $(m_k, j) \in \mathbb{E}'(\mu)$ in $\mathbb{B}_{model}(\mu)$ between a service model $m_k \in \mathbb{M}(\mu)$ and a cloudlet $j \in \mathbb{N}'(\mu)$ if cloudlet j has enough residual computing resource to accommodate service model m_k. Given the hosting cloudlet $h(DT_i)$ of each DT_i, the weight of each $(m_k, j) \in \mathbb{E}'(\mu)$ is $\overline{cost}_{model}(m_k, j)$, referring to Eq. (14).

Let $Match_{model}(\mu)$ be a minimum-cost maximum matching in $\mathbb{B}_{model}(\mu)$. For each edge $e(m_k, j) \in Match_{model}(\mu)$, model m_k is placed to cloudlet j with the amount $comp(m_k)$ of computing resource consumed, and the residual computing resource in cloudlet j is then reduced by $comp(m_k)$. This procedure continues until all models in M are deployed.

The proposed algorithm for the placement problem for DTs and models is detailed in `Algorithm` 1.

Algorithm 1 Algorithm for the placement problem of DTs and models

Input: An MEC network $G = (N, E)$, a set V of mobile objects, with each object having its top-K visiting location profile, and demanding the amount $comp(DT_i)$ of computing resource for its DT, a set N of cloudlets with computing capacities, a set m_k of service models that are retrained by the update data from mobile objects in V.

Output: Find a DT and model placement scheduling for each object in V to minimize $\sum_{m_k \in M} \overline{cost}_{model}(m_k, h(m_k))$.

$\mu \leftarrow 1$, $\mathbb{V}(\mu) \leftarrow V$, $\mathbb{N}(\mu) \leftarrow N$,
Identify $\mathbb{E}(\mu)$, and each edge $e(v_i, j) \in \mathbb{E}(\mu)$ has a weight $\overline{cost}_{DT}(v_i, j)$ by Eq. (13) ;
while $\mathbb{E}(\mu) \neq \emptyset$ **do**
  Construct the bipartite graph $\mathbb{B}_{DT}(\mu) = (\mathbb{V}(\mu), \mathbb{N}(\mu), \mathbb{E}(\mu))$;
  Identify a minimum-cost maximum matching $Match_{DT}(\mu)$ in $\mathbb{B}_{DT}(\mu)$ by the Hungarian algorithm;
  for each edge $e(v_i, j)$ in $Match_{DT}(\mu)$ **do**
    Deploy DT_i in cloudlet j;
  end for
  $\mu \leftarrow \mu + 1$;
  Update $\mathbb{V}(\mu)$, $\mathbb{N}(\mu)$ and $\mathbb{E}(\mu)$;
end while
$\mu \leftarrow 1$, $\mathbb{M}(\mu) \leftarrow M$;
Identify $\mathbb{N}'(\mu)$ and $\mathbb{E}'(\mu)$, and each edge $e(m_k, j) \in \mathbb{E}'(\mu)$ has a weight $\overline{cost}_{model}(m_k, j)$;
while $\mathbb{E}'(\mu) \neq \emptyset$ **do**
  Construct bipartite graph $\mathbb{B}_{model}(\mu) = (\mathbb{M}(\mu), \mathbb{N}'(\mu), \mathbb{E}'(\mu))$;
  Identify a minimum-cost maximum matching $Match_{model}(\mu)$ in $\mathbb{B}_{model}(\mu)$ by the Hungarian algorithm;
  for each edge $e(m_k, j)$ in $Match_{model}(\mu)$ **do**
    Deploy model m_k in cloudlet j;
  end for
  $\mu \leftarrow \mu + 1$;
  Update $\mathbb{M}(\mu)$, $\mathbb{N}'(\mu)$ and $\mathbb{E}'(\mu)$);
end while

B. Algorithm analysis

Theorem 2: Given an MEC network $G = (N, E)$, a set V of mobile objects, with each object having its top-K visiting location profile, and demanding the amount $comp(DT_i)$ of computing resource for its DT, a set N of cloudlets with computing capacities, a set m_k of service models that are retrained by the update data from mobile objects in V, there is an algorithm, `Algorithm` 1 for the placement problem of DTs and models, which delivers a feasible solution to the problem. The time complexity of `Algorithm` 1 is $O(|V| \cdot (|V| + |N|)^3 + |M| \cdot (|M| + |N|)^3)$.

Proof We first show the feasibility of the delivered solution by `Algorithm` 1 to the placement problem of DTs and models as follows. For placing DTs in cloudlets, recall that

$Match_{DT}(\mu)$ is a minimum-cost maximum matching in bipartite graph $\mathbb{B}_{DT}(\mu)$ in iteration μ. It can be seen that if there is one edge $(v_i, j) \in Match_{DT}(\mu)$ between an object $v_i \in \mathbb{V}(\mu)$ and a cloudlet $j \in \mathbb{N}(\mu)$, it means cloudlet j has enough computing resource to accommodate DT_i in iteration μ, and the DT_i of object v_i will be placed in cloudlet j. Otherwise, the DT_i of object v_i will not be placed in cloudlet j in iteration μ. Thus, the solution for deploying DTs is feasible. Similarly, we can see it is feasible to deploy service models in cloudlets by `Algorithm` 1. The solution thus is a feasible solution to the placement problem of DTs and models.

We then analyze the time complexity of `Algorithm` 1. For placing DTs, in each iteration, the dominant time complexity of the algorithm is to construct a bipartite graph and find a minimum-cost maximum matching in the graph, which takes $O((|V|+|N|)^3)$ time. Therefore, it takes $O(|V|\cdot(|V|+|N|)^3)$ time to place DTs in cloudlets, because there are at most $|V|$ iterations. Similarly, we can see it takes $O(|M|\cdot(|M|+|N|)^3)$ time to place service models in cloudlets. `Algorithm` 1 thus takes $O(|V|\cdot(|V|+|N|)^3+|M|\cdot(|M|+|N|)^3)$ time.

V. Algorithm for the Accumulative Fidelity Maximization Problem

In this section, assuming that both DTs of all objects and all service models have been placed by `Algorithm` 1, we propose an algorithm for the accumulative fidelity maximization problem for a given time horizon. We start by predicting the update data volume of each object since its last uploading. We then choose which objects to upload update data at each time slot, such that the optimization objective (12) is maximized for a given time horizon.

A. Volume prediction of the update data of each object

So far, we assumed that the volume of the update data $vol(v_i, t)$ of device v_i at time slot t is given. In reality, the volume usually is unknown in advance. Thus, accurate prediction on the volume of each object at each time slot becomes crucial, as the volume of the update data of an object determines not only the fidelity gains of service models in which the object participates but also the resource cost for uploading, transferring, and processing of the update data. In the following, we make use of the data stored at its DT of each object to predict the volume of its update data at the current time slot since its last uploading.

For a given time slot t, we first predict the volume of the update data of each IoT device (object) since its last uploading at time slot t' with $t' < t$. Note that each object v_i can move to different locations at different time slots.

There are many prediction methods for the prediction on the volume of update data of each object, such as the Long Short Term Memory (LSTM) method and auto-regression method. For the sake of convenience, we here adopt the auto-regression method [24] to predict the volume $vol(v_i, t)$ of the update data by object v_i at time slot t as follows.

$$vol(v_i,t) = \lambda_1 \cdot vol(v_i, t-1) + \lambda_2 \cdot vol(v_i, t-2) + \ldots \lambda_q \cdot vol(v_i, t-q), \quad (16)$$

where $\lambda_1, \lambda_2, \ldots, \lambda_q$ are constant weights with $\lambda_1 + \lambda_2 + \cdots + \lambda_q = 1$. Also, $\lambda_q \geq \lambda'_q$ with $q < q'$, because the fresher information is more important.

B. Algorithm

Having the predicted data volume at time slot t, we can reduce the accumulative fidelity maximization problem in a time slot into a maximum-profit Generalized Assignment Problem (GAP). We then adopt the approximation algorithm from [1] for the maximum-profit GAP, which returns an approximate solution to the original problem at each time slot.

Recall that object v_i is covered by a set $AP(v_i, t)$ of APs. Denote by $\mathcal{M}(v_i)$ the set of models requiring the DT data of v_i. Having deployed DT_i of object v_i into cloudlet $h(DT_i)$ and model $m_k \in M$ into cloudlet $h(m_k)$, when object v_i uploads its update data of volume $vol(v_i, t)$ through AP $j \in AP(v_i, t)$ at time slot t, its contribution $obj\_contri(v_i, j, t)$ to the optimization objective function (12) is

$$\begin{aligned} &obj\_contri(v_i,j,t) \\ &= \sum_{m_k \in \mathcal{M}(v_i)} \Delta Fid(m_k, v_i, t) - \omega \cdot (\xi_1 \cdot PX_i \cdot time_{upload}(v_i, j, t) \\ &+ \sum_{e \in P_{j,h(DT_i)}} \xi_2 \cdot vol(v_i, t) \cdot length(e) + \xi_3 \cdot vol(v_i, t) \\ &+ \sum_{m \in \mathcal{M}(v_i)} \Big(\sum_{e \in P_{h(DT_i),h(m_k)}} \xi_2 \cdot vol'(v_i, t) \cdot length(e) \\ &+ \xi_3 \cdot vol'(v_i, t) \Big) \Big) \end{aligned} \quad (17)$$

where the first term in Eq. (17) is the positive contribution on the accumulative fidelity gain of all models by the data uploading of object v_i with $\Delta Fid(m_k, v_i, t)$ defined by Eq. (3). The rest of the five terms are the uploading cost, the transmission cost of the update data from the uploading location of v_i to its DT location $h(DT_i)$, the processing cost of the update data at its DT location, the transmission cost of the update data from its DT location to the location of all models in $\mathcal{M}(v_i)$, and the training cost of each model.

We now schedule a subset of IoT devices covered by APs to upload update data such that the sum $\sum_{v_i \in V} obj\_contri(v_i, j, t)$ is maximized at each time slot.

The maximum-profit GAP is given as follows. Given each AP $j \in N$ with L_j subchannels, there is a bin b_j with capacity L_j. For each object $v_i \in V$, if it is covered by AP j at time slot t, with $j \in AP(v_i, t)$ and $obj\_contri(v_i, j, t) > 0$, there is an item $item_i$ with weight 1, i.e., object v_i occupies one of the L_j subchannels of AP j for data uploading. The profit of assigning item $item_i$ to bin b_j with $j \in AP(v_i, t)$ is $obj\_contri(v_i, j, t)$ by Eq. (17). Otherwise, the profit of assigning $item_i$ to bin b_j is 0, i.e., v_i cannot upload its update data via AP $j \in N \backslash AP(v_i, t)$ as either it is not covered by AP j at time slot t or its contribution to the objective function with a negative value. The proposed algorithm is detailed in `Algorithm` 2.

Algorithm 2 Algorithm for the accumulative fidelity maximization problem

Input: An MEC network $G = (N, E)$, a set N of cloudlets (APs), a given time horizon $\mathbb{T}$, a set V of mobile objects, a set M of service models that are retrained by the update data from mobile objects in V.

Output: Deploy DTs and service models in cloudlets, and schedule each object to which AP to upload its update data at each time slot t for a given time horizon such that the objective (12) is maximized.

1: Deploy all DTs and models in cloudlets by invoking `Algorithm` 1;
2: **for** each time slot $t \in T$ **do**
3: **for** each $v_i \in V$ **do**
4: Predict the volume $vol(v_i, t)$ of object v_i at time slot t, by applying the auto-regression prediction mechanism;
5: **end for**;
6: **for** each $v_i \in V$ **do**
7: **for** each AP $j \in N$ **do**
8: **if** AP $j \in AP(v_i, t)$ **then**
9: Calculate $obj\_contri(v_i, j, t)$ by Eq. (17);
10: **else**
11: $obj\_contri(v_i, j, t) \leftarrow 0$;
12: **end if**
13: **end for**
14: **end for**;
15: Construct an instance of the maximum-profit GAP, where each AP $j \in N$ corresponds to a bin b_j with capacity L_j, i.e., it has L_j subchannels. Each IoT device $v_i \in V$ corresponds to an item $item_i$ with weight 1, i.e., the IoT device is assigned to one subchannel, and the profit of assigning $item_i$ to bin b_j with $j \in AP(v_i, t)$ is $obj\_contri(v_i, j, t)$ if $obj\_contri(v_i, j, t) > 0$; otherwise, the profit of assigning $item_i$ to bin b_j is 0;
16: Schedule the update data uploading of objects by invoking the approximation algorithm in [1] to find an approximate solution to the maximum-profit GAP at time slot t
17: **end for**.

C. Algorithm analysis

Lemma 1: Given an MEC network $G = (N, E)$, a set N of cloudlets (APs), a time horizon $\mathbb{T}$, a set V of mobile objects and a set M of service models, consider a special case of the accumulative fidelity maximization problem with the assumption that DTs and models have been deployed, and the volumes of the update data of objects in the current time slot are given. `Algorithm` 2 delivers an approximate solution for this special accumulative fidelity maximization problem in a single time slot with an approximation ratio of $\frac{1}{2+\epsilon}$, where ϵ is a constant with $0 < \epsilon \leq 1$.

Proof Because we reduce this special accumulative fidelity maximization problem in a single time slot into a maximum-profit GAP, we adopt the approximation algorithm from [1] for the maximum-profit GAP, which returns an approximate solution to the special problem at each time slot.

Therefore, the analysis of the approximation ratio can be referred to the detailed analysis in [1], i.e., the solution value by `Algorithm` 2 for the special accumulative fidelity maximization problem in a single time slot is no less than $\frac{1}{2+\epsilon}$ times the optimal one. ■

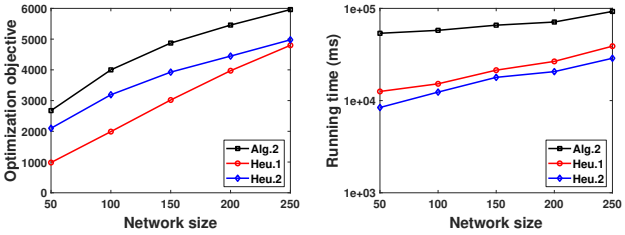
Theorem 3: Given an MEC network $G = (N, E)$, a set N of cloudlets (APs), a time horizon $\mathbb{T}$, a set V of mobile objects, a set M of service models continuously retrained by the update data from mobile objects in V, there is an online algorithm, `Algorithm` 2 for the accumulative fidelity maximization problem, which delivers a feasible solution to the problem. The time complexity of `Algorithm` 1 is $O(|V| \cdot (|V|+|N|)^3 + |M| \cdot (|M|+|N|)^3 + T \cdot (|N| \cdot |V| \cdot \log \frac{1}{\epsilon} + \frac{|N|}{\epsilon^4}))$.

Proof Theorem 2 shows invoking `Algorithm` 1 for placing DTs and models causes no violations on computing capacities of cloudlets. We can also observe that the number of objects assigned to each AP is no greater than its number of subchannels at each time slot by `Algorithm` 2. Therefore, `Algorithm` 2 delivers a feasible solution to the accumulative fidelity maximization problem.

The rest is to analyze the time complexity of `Algorithm` 2. Theorem 2 shows invoking `Algorithm` 1 takes $O(|V| \cdot (|V|+|N|)^3 + |M| \cdot (|M|+|N|)^3)$ time. Invoking the approximation algorithm in [1] at each time slot takes $(|N| \cdot |V| \cdot \log \frac{1}{\epsilon} + \frac{|N|}{\epsilon^4})$ time. Therefore, the time complexity of `Algorithm` 2 is $O(|V| \cdot (|V|+|N|)^3 + |M| \cdot (|M|+|N|)^3 + T \cdot (|N| \cdot |V| \cdot \log \frac{1}{\epsilon} + \frac{|N|}{\epsilon^4}))$. ■

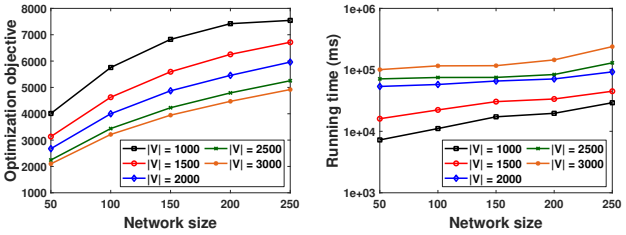
VI. Performance evaluation

We considered MEC network instances, where the number of APs (and their co-located cloudlets) ranges from 50 to 250, and the MEC network instances are generated by the GT-ITM tool [2]. The computing capacity on a cloudlet is drawn from $4,000$ MHz to $8,000$ MHz [7]. The amount of computing resource demanded by a DT or a service model is within $[20, 100]$ MHz [8], [10]. The bandwidth capacity on an AP ranges from 5 MHz to 20 MHz, and the number of subchannels of an AP ranges from 3 to 6 by adopting the OFDMA scheme. There are $2,000$ IoT devices (objects), and the transmission power of an IoT device is within $[0.1, 0.5]$ Watt [7]. The cost ξ_1 of unit power is within $[0.1, 0.01]$, and the noise power is set as 1×10^{-10} Watt [7]. Following [18], we set the channel gain between an IoT device v_i and an AP j as $dist_{i,j}^{-\alpha}$, where $dist_{i,j}$ is the Euclidean distance between locations of IoT device v_i and AP j, and α is the path loss coefficient (we set α = 4). The volume of the update data of an IoT device is within $[1, 5]$ MB [23], and the volume of the processed update data is half of that of its raw data. Assuming that there are 500 service models, the number of attributes in a service model ranges from 5 to 15, respectively. The cost ξ_3 of processing a unit data (1 MB) by a DT or model at a cloudlet is set within $[0.1, 0.01]$, while the cost ξ_3 of transferring a unit data along a link is set within $[0.1, 0.01]$ [17]. We adopt the submodular



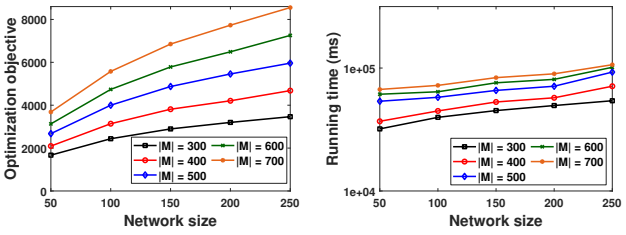
(a) Performance (b) Running time

Fig. 2. Algorithm performance for the accumulative fidelity maximization problem.



(a) Performance (b) Running time

Fig. 3. Impact of the number $|V|$ of the IoT devices on the performance of Alg.2.



(a) Performance (b) Running time

Fig. 4. Impact of $|M|$ of service models on the performance of Alg.2.

function $f(x) = \log_{1.1}(\frac{x}{10} + 1)$ in Eq. (3), and the weight $w_{k,l}$ of the lth attribute of model m_k is set as $\frac{1}{l_k}$, where l_k is the number of attributes of model m_k. The time horizon consists of 20 time slots. The maximum number of potential mobility locations for an IoT device is 10% of the number of APs [16]. We adopt the auto-regression method [24] to predict the volume $vol(v_i, t)$ of the update data by object v_i at time slot t, with $vol(v_i, t) = 0.4 \cdot vol(v_i, t-1) + 0.3 \cdot vol(v_i, t-2) + 0.2 \cdot vol(v_i, t-3) + 0.1 \cdot vol(v_i, t-4)$. We set ω as 0.1 and ϵ as 0.5. The value in each figure is the mean of 30 different network instances with the same size. The running time of each algorithm is obtained by a desktop with an Octa-Core Intel(t) Xeon(t) CPU @ 2.20 GHz, 32G RAM. Unless otherwise specified, we adopt the above parameters by default.

We evaluated the proposed Algorithm 2, referred to as Alg.2, for the accumulative fidelity maximization problem against the following benchmarks.

(1) Heu.1: It first deploys DT_i of each object $v_i \in V$ in a cloudlet $h(DT_i)$ with the least expected DT updating cost $\overline{cost}_{DT}(v_i, h(DT_i))$ greedily, and then deploys each service model m_k in a cloudlet with the least expected model updating cost $\overline{cost}_{model}(m_k, h(m_k))$ greedily too. Within each time slot, Heu.1 considers IoT devices one by one, and predicts the volume of its update data at the current time slot, through adopting that in the previous time slot (Heu.1 predicts the volume of its update data in the first time slot, through adopting the expected value of the data volume). Heu.1 assigns each IoT device to an AP with the maximum $obj\_contri(v_i, j, t)$ by Eq. (12).

(2) Heu.2: similar to Heu.1, however, it considers APs one by one at each time slot, i.e., each AP is iteratively assigned with an IoT device which can achieve the largest $obj\_contri(v_i, j, t)$ until the AP cannot accommodate any more IoT devices. This procedure continues until all APs have been examined.

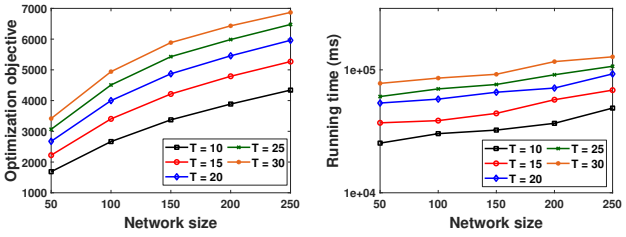
A. Algorithm performance evaluation

We first investigated the performance of Alg.2 against Heu.1 and Heu.2 for the accumulative fidelity maximization problem by varying network size from 50 to 250. Fig. 2 plots the performance (the optimization objective (12)) and running time of different algorithms. From Fig. 2(a), we see Alg.2 achieves the maximum objective value, which outperforms Heu.1 and Heu.2 by 24.2% and 19.8% in terms of performance, respectively, when the network size is 250. The rationale behind is that Alg.2 can determine the placement of DTs and models efficiently, while establishing an efficient strategy for predicting the volume of the update data of each IoT device and determining its uploading AP at each time slot. We can see from Fig. 2(b) that Alg.2 takes the longest running time, because of identifying minimum-cost maximum matchings in bipartite graphs for the placement of DTs and models, and invoking the algorithm from [1] at each time slot to determine the assignment of APs to IoT devices.

We then studied the impact of the number $|V|$ of IoT devices on the performance of Alg.2 for the accumulative fidelity maximization problem, with $|V|$ ranging from $1,000$ to $5,000$. Observed from Fig. 3(a), the performance by Alg.2 with $|V| = 3,000$ is 65.2% of that by itself with $|V| = 1,000$, when the network size is 250. This indicates that DTs of IoT devices can be updated more frequently with less numbers of IoT devices in the MEC network. Fig. 3(b) shows Alg.2 with $|V| = 3,000$ takes the most running time due to examining the most number of IoT devices.

We further evaluated the impact of the number $|M|$ of the service models on the performance of Alg.2 when $|M| = 300, 400, 500, 600$ and 700. As evidenced by Fig. 4(a), the performance of Alg.2 with 300 models is 40.5% of that by itself with 700 models when the network size is 250. This is because more fidelity gains of service models can be obtained with more service models, which will also lead to more running time shown in Fig. 4(b).

We finally studied the impact of the number T of time slots on the performance of Alg.2 in Fig. 5, by varying the number of time slots from 10 to 30. Fig. 5(a) depicts that when the network size is 250, the performance by Alg.2 with 30 time slots is 58.2% higher than that by itself with 10 time



(a) Performance (b) Running time

Fig. 5. Impact of the number T of time slots on the performance of `Alg.2`.

slots. Also, because the fidelity of a service model is measured by a submodular non-decreasing function, the performance increment of `Alg.2` in a time slot decreases with the increase on the number of time slots.

VII. Conclusion

In this paper, we investigated the accumulative fidelity maximization problem of service models in a DT-assisted edge computing network, through DT and model placements and continuous training on service models using the DT update data of IoT devices in a real-time manner. We first formulated a sub-optimization problem – the DT and model placement problem to minimize the total cost of various resources consumed under the assumption that the profile of each mobile object is given. We then developed an algorithm for the DT and model placement problem, by decomposing the problem into two sub-problems and tackled them through reducing them to a series of minimum-cost maximum matchings in auxiliary bipartite graphs. Having all DTs and service models been placed, we thirdly investigated the accumulative fidelity maximization problem, and proposed an online algorithm to schedule each mobile object to an AP to upload its update data at each time slot. We finally evaluated the performance of the proposed algorithm via simulations. Simulation results indicated that the proposed algorithms are promising, and outperform their comparison counterparts nearly by 20%.

References

[1] R. Cohen, L. Katzir, and D. Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, vol. 100, pp. 162 – 166, 2006.

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

[3] Q. Guo, F. Tang, and N. Kato. Resource allocation for aerial assisted digital twin edge mobile network. *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3070 – 3079, 2023,

[4] Y. Hui, X. Ma, Z. Su, N. Cheng, Z. Yin, T. H. Luan, and Y. Chen. Collaboration as a service: digital-twin-enabled collaborative and distributed autonomous driving. *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18607 – 18619, 2022.

[5] Z. Kuang, Y. Shi, S. Guo, J. Dan, and B. Xiao. Multi-user offloading game strategy in OFDMA mobile cloud computing system. *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12190 – 12201, 2019.

[6] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, W. Xu, A. Y. Zomaya. Digital twin-assisted, SFC-enabled service provisioning in mobile edge computing. *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 393 – 408, 2024.

[7] 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.

[8] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya. Digital twin-enabled service satisfaction enhancement in edge computing. *Proc of INFOCOM'23*, IEEE, 2023.

[9] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, W. Xu, K. Wei, and X. Jia. Mobility-aware utility maximization in digital twin-enabled serverless edge computing. *IEEE Transactions on Computers*, to be published, 2024, doi: 10.1109/TC.2024.3388897.

[10] 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.

[11] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Y. Zeng, B. Ye, and X. Jia. Digital twin-enabled service provisioning in edge computing via continual learning. *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7335 – 7350, 2024.

[12] 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*, to be published, 2024, doi: 10.1109/TNET.2024.3395709.

[13] 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*, to be published, 2023, doi: 10.1109/TSC.2023.3341988.

[14] 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 Journal of Selected Topics in Signal Processing*, vol. 18, no. 1, pp. 98 – 108, 2024.

[15] 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 Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2402 – 2416, 2023.

[16] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo. Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks. *IEEE Transactions on Mobile Computing*, vol. 21, no, 1, pp. 196 – 210, 2022.

[17] Y. Ma, W. Liang, J. Wu, and Z. Xu. Throughput maximization of nfv-enabled multicasting in mobile edge cloud networks. *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 2, pp. 393 – 407, 2020.

[18] S. Nath, Y. Li, J. Wu, and P. Fan. Multi-user multi-channel computation offloading and resource allocation for mobile edge computing. *Proc of ICC'20*, 2020

[19] S. D. Okegbile, J. Cai, H. Zheng, J. Chen, and C. Yi, Differentially private federated multi-task learning framework for enhancing human-to-virtual connectivity in human digital twin. *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3533 – 3547, 2023.

[20] Y. Ren, S. Guo, B. Cao, and X. Qiu. End-to-end network SLA quality assurance for C-RAN: a closed-loop management method based on digital twin network. *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4405 – 4422, 2024.

[21] D. Shomys and E. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, vol. 62, pp. 461 – 474, 1993.

[22] Y. Shu, Z. Wang, H. Liao, Z. Zhou, N. Nasser, and M. Imran. Age-of-information-aware digital twin assisted resource management for distributed energy scheduling. *Proc of GLOBECOM'22*, IEEE, pp. 5705 – 5710, 2022.

[23] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, and G. Karakostas. Digital twin placement for minimum application request delay with data age targets. *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11547 – 11557, 2023.

[24] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, vol. 18, no. 11, pp. 2672 – 2685, 2019.

[25] R. Zhang, Z. Xie, D. Yu, W. Liang, and X. Chang. Digital twin-assisted federated learning service provisioning over mobile edge networks. *IEEE Transactions on Computers*, vol. 73, no. 2, pp. 586 – 598, 2024.

[26] 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*, to be published, 2024, doi: 10.1109/TMC.2024.3394839.

[27] L. Zhao, Z. Bi, A. Hawbani, K. Yu, Y. Zhang, and M. Guizani. ELITE: an intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks. *IEEE Transactions on Mobile Computing*, vol. 22, no. 9, pp. 5231 – 5247, 2023.