Cost Minimization of Digital Twin Placements in Mobile Edge Computing

YUNCAN ZHANG, Department of Computer Science, City University of Hong Kong, Hong Kong, China
WEIFA LIANG, Department of Computer Science, City University of Hong Kong, Hong Kong, China
WENZHENG XU, Computer Science College, Sichuan University, Chengdu, China
ZICHUAN XU, School of Software Engineering, Dalian University of Technology, Dalian, China
XIAOHUA JIA, Department of Computer Science, City University of Hong Kong, Hong Kong, China

In the past decades, explosive numbers of Internet of Things (IoT) devices (objects) have been connected to the Internet, which enable users to access, control, and monitor their surrounding phenomenons at anytime and anywhere. To provide seamless interactions between the cyber world and the real world, Digital twins (DTs) of objects (IoT devices) are key enablers for real time monitoring, behaviour simulations, and predictive decisions on objects. Compared to centralized cloud computing, mobile edge computing (MEC) has been envisioning as a promising paradigm for low latency IoT applications. Accelerating the usage of DTs in MEC networks will bring unprecedented benefits to diverse services, through the co-evolution between physical objects and their virtual DTs, and DT-assisted service provisioning has attracted increasing attention recently.

In this article, we consider novel DT placement and migration problems in an MEC network with the mobility assumption of objects and users, by jointly considering the freshness of DT data and the service cost of users requesting for DT data. To this end, we first propose an algorithm for the DT placement problem with the aim to minimize the sum of the DT update cost of objects and the total service cost of users requesting for DT data, through efficient DT placements and resource allocation to process user requests. We then devise an approximation algorithm with a provable approximation ratio for a special case of the DT placement problem when each user requests the DT data of only one object. Meanwhile, considering the mobility of users and objects, we devise an online, two-layer scheduling algorithm for DT migrations to further reduce the total service cost of users within a given finite time horizon. We finally evaluate the performance of the proposed algorithms through experimental simulations. The simulation results show that the proposed algorithms are promising.

CCS Concepts: • **Networks** → **Network services**; • **Theory of computation** → *Design and analysis of algorithms;*

The work by Yuncan Zhang and Weifa Liang were partially supported by the Research Grants Council (RGC) of Hong Kong under CityU 7005845, 8730094, 9043510, and 9380137, respectively. The work by Zichuan Xu was funded by the National Natural Science Foundation of China (NSFC) with grant No: 62172068, 62172071, 61802048, 61802047, and the "Xinghai scholar" program, and the work done by Wenzheng Xu was supported in part by NSFC (Grant No. 61602330).

Authors' Contact Information: Yuncan Zhang, Department of Computer Science, City University of Hong Kong, Hong Kong, China; e-mails: yuncan.zhang@cityu.edu.hk; Weifa Liang, Department of Computer Science, City University of Hong Kong, Hong Kong, China; e-mail: weifa.liang@cityu.edu.hk; Wenzheng Xu, Computer Science College, Sichuan University, Chengdu, Sichuan, China; e-mail: wenzheng.xu@scu.edu.cn; Zichuan Xu, School of Software Engineering, Dalian University of Technology, Dalian, Liaoning, China; e-mail: z.xu@dlut.edu.cn; Xiaohua Jia, Department of Computer Science, City University of Hong Kong, Hong Kong, HK, China; e-mail: csjia@cityu.edu.hk.

ACM 1550-4859/2024/05-ART74
https://doi.org/10.1145/3658449

Additional Key Words and Phrases: Digital twin placement and migration, Digital twin synchronization, Cost modeling of digital twin placement, Digital twin service cost minimization, Mobile Edge Computing

ACM Reference Format:
Yuncan Zhang, Weifa Liang, Wenzheng Xu, Zichuan Xu, and Xiaohua Jia. 2024. Cost Minimization of Digital Twin Placements in Mobile Edge Computing. *ACM Trans. Sensor Netw.* 20, 3, Article 74 (May 2024), 26 pages. https://doi.org/10.1145/3658449

1 INTRODUCTION

As the Internet of Things (IoT) develops rapidly, the last decades have witnessed billions of devices connected to the Internet, and virtual representations of physical or abstract objects are increasingly accessible via Web technology, enabling seamless interactions between the cyber world and the real world to become a reality [1]. Digital twin (DT) technology, as a disruptive technique, enables to create virtual DTs of physical objects, and makes use of Artificial Intelligence (AI) and machine learning (ML) methods to simulate the behaviours or provide decisive predictions of objects. DTs have been applied to diverse fields including manufacturing, the IoT, personalized healthcare, autonomous driving, smart city, Metaverse, and so on [26].

With the proliferation of mobile and IoT applications, the volume of data generated at the network edge has been growing explosively. Mobile edge computing (MEC) that brings the computing and storage capabilities to the edge of core networks was conceived in a bid to fill the gap between centralized clouds and mobile devices [25]. Orthogonal to the DT technology, there is growing interest in DT-assisted service provisioning in MEC networks. In [4] and [20], DTs are integrated with edge networks, where DTs can exchange information with their objects, process the collected data by utilizing the computation resource of edge cloudlets, and generate results to improve the performance of objects and offer synthesized data for users. Instead of being a simple data relay, a DT offers various functionalities for users, e.g., providing information by analyzing the historical data received from its object, which usually cannot be obtained from the object itself [6, 8]. A DT can also interact with users on behalf of its object. Therefore, the communications between objects and users are decomposed into the state synchronization between objects and their DTs and service results delivery between DTs and users.

To enable DT-assisted service provisioning in an MEC network, there is a fundamental issue to place DTs of objects to cloudlets while meeting service demands of different users. A user usually requests DT services from multiple objects, and the requested DT data of these objects are then transmitted from their DTs to the home cloudlet of the user for request processing. For example, consider an anomaly detection service model in autonomous driving environments whose training source data comes from multiple objects, including autonomous vehicle sensing data, traffic lights, local weather conditions, and roadside cameras [6]. Timely and accurate response to anomalies can significantly improve driving safety and mitigate fatal accidents. The adoption of DTs of source objects in such scenario facilitates the historical traces analysis, various anomaly detection, and vehicle trajectory prediction. In this case, if the update data of objects are not uploaded to the system on time (or the objects do not synchronize with their DTs on time), the accuracy of the anomaly detection model will drastically deteriorate, and the quality of its service will be doubtful.

To ensure that each DT can represent its object truly, the object needs to continuously synchronize with its DT by uploading its update data to the DT and processing the update data at the DT. The DT placement thus affects not only the synchronization delay between each DT and its object but also the communication delay between the DT and its users. Furthermore, providing DT services consumes various network resources including computing, storage, and bandwidth

resources. The DT placement needs to consider the computing and storage resources for hosting DTs, and the communication resource for update data synchronizations between objects and their DTs, and DT data delivery from DTs to their users.

The novelty of the study in this article lies in formulating novel DT placement and migration problems in MEC networks with the mobility assumption of objects and users, with the aim to minimize the sum of the DT update cost and the total service cost of users requesting for DT data. Efficient algorithms for the defined problems are proposed, which explore non-trivial trades-off between the DT update (synchronization) cost of objects and the total service cost of users requesting for DT data, through intelligent DT placements and migrations.

The main contributions of the article are given as follows. We first formulate a novel DT placement problem in an MEC network with the aim to minimize the cost sum of the DT update cost of objects and the total service cost of users requesting for DT data. Due to the NP-hardness of the problem, we then propose an efficient algorithm for the DT placement problem through a reduction to the minimum-cost maximum matching problem. We also devise a performance-guaranteed approximation algorithm for a special case of the problem where each user requests only the DT data of a single object. We thirdly develop an online, two-layer scheduling algorithm for the DT migration problem under the mobility assumption of both objects and users to mitigate the impact of object mobility on the service cost of users for a given time horizon. We finally evaluate the performance of the proposed algorithms for DT placements and migrations through experimental simulations. Simulation results show that the proposed algorithms are promising.

The remainder of this article is organized as follows. Section 2 surveys related works. Section 3 introduces the system model and defines the problem formally. Section 4 devises an efficient algorithm for the DT placement problem. Section 5 proposes an approximation algorithm with a provable approximation ratio for a special case of the DT placement problem. Section 6 develops an online, two-layer scheduling algorithm for the DT migration problem, which deals with the mobility of objects to explore non-trivial trades-off between the DT update data cost and the total service cost of users requesting for DT data. Section 7 evaluates the proposed algorithms empirically, and Section 8 concludes the article.

2 RELATED WORK

Lot of efforts have been taken on service provisioning in MEC networks [16, 17, 23, 24, 37]. For example, Ma et al. [24] studied service provisioning through virtual network function (VNF) placement in MEC, by considering user mobility and service delay requirement with the aim to maximize the accumulative network utility. Xu et al. [37] investigated VNF service provisioning through sharing existing VNF instances or instantiating new VNF instances among users. They presented a prediction mechanism for new VNF instance creations and idle VNF instance releases to reduce the cost, considering dynamic changes of user request patterns over time. Ma et al. [23] considered different user requests need different services that are represented by VNFs instantiated in cloudlets in an MEC network. They investigated the joint VNF instance deployment and user requests assignment in MEC, by explicitly exploring a non-trivial usage tradeoff between different computing and communication resources. In the above works, the VNF instances for service provisioning do not change after the deployment. In contrast, the DT-assisted service provisioning needs continual synchronizations between objects and their DTs. The previous mentioned works did not handle continuously monitoring the state information of objects and thus cannot be directly applied for DT placements in MEC.

There are several recent studies of applying the DT technology for resource allocation and optimization in MEC networks. Tang et al. surveyed DT-assisted resource allocation for 6G wireless networks to reduce the cost of computation and communication [31]. Fan et al. [4] designed a DT

empowered MEC framework for achieving intelligent vehicular lane-changing, by leveraging DT techniques to build a virtual counterpart of the real MEC network. Lu et al. [21] integrated DTs with MEC to bridge the gap between physical edge networks and digital systems, and developed a deep reinforcement learning (DRL) algorithm for resource allocation and user request scheduling in DT edge networks. Dai et al. [3] introduced a DT network paradigm for industrial IoT to explore task offloading and resource allocation in the DT network. Ren et al. [28] studied a Digital Twin Network (DTN)-based SLA quality closed-loop management scheme to achieve low cost network service deployment. Tang et al. [32] designed a DTN-assisted network slicing framework by adopting a DRL approach, and made use of the DTN to simulate/predict the resource consumption of the MEC network and utilized the resources in the network efficiently. Zhao et al. [39] introduced a hierarchical routing strategy in a DT-assisted vehicular network to enable various function services for vehicular users. Zhang et al. [38] leveraged DT models to optimize device scheduling and MEC resource allocation, aiming to maximize the accumulative utility across FL services. They developed heuristic and constant-approximation algorithms for offline multi-FL services. These mentioned studies focused mainly on DT network modelling and did not consider and improve the network performance ultimately.

Mobility-aware service provisioning in a DT-empowered network has also been investigated in literature recently. Lei et al. [8] made use of DTs to monitor the life cycle of a Unmanned Aerial Vehicles (UAVs) swarm with high mobility. They proposed an ML algorithm to optimize the behaviours of the UAV swarm with the assistance of the DTs of UAVs. Li et al. [9] investigated trajectories of UAVs in a time-varying MEC environment. They established DTs in a base station to estimate the states of the MEC network, and devised a Double Deep Q-network (DDQN) algorithm to minimize the total energy consumption of the system. Li et al. [10] studied reliability-aware SFC placements in MEC by leveraging DTs for predicting the reliability of service function instances. Liang et al. [18] investigated the relationship between the freshness of service models and AoIs of the DT data for training of these models. They devised an efficient algorithm for minimizing the cost of various resource consumed for improving model freshness. Lin et al. [19] paid an attention on dynamic and stochastic DT service demands of users with mobility in MEC, by devising an incentive-based congestion control scheme to maximize the long-term profit of the network service provider. Liu et al. [20] addressed the problem of mobile users intelligently offloading their tasks to cooperative mobile-edge servers with the DT assistance. Wang et al. [35] developed a Mobility Digital Twin (MDT) framework for transportation in MEC, i.e., creating DTs for human, vehicles and traffic infrastructures to implement functionalities such as modelling, learning, and prediction, thereby providing efficient mobility services. However, none of the mentioned studies considered the user service augmentation based on the age of information (AoI) query results by deploying DTs in MEC networks.

Complement to this study, there were several efforts on optimizing the service delay or the freshness of request results in a DT-assisted MEC network. For example, Corneo et al. [2] investigated the one-to-many information dissemination from sensors to remote applications by means of DTs and presented an AoI-aware scheduling policy to reduce the energy consumption, while ensuring the freshness of information. Sun et al. [30] utilized DTs to predict the workload of edge servers under uncertain user mobility. They leveraged a DT counterpart of the entire MEC to obtain the training data and developed a DRL algorithm to minimize the offloading delay of users. Lu et al. [22] developed a DRL-based algorithm in MEC to decide the DT placement and cope with the mobility of users by DT migration, with the aim to minimize the average system latency. Li et al. [11, 12] devised an approximation algorithm for dynamic DT placements to improve user service satisfaction under the mobility of objects while ensuring low AoI services on DT data. Li et al. [13] also investigated the DT synchronization issue via continual learning in an MEC environment, with

the aim to maximize the total utility gain, i.e., the enhanced model accuracy. They studied two novel optimization problems: the static DT synchronization problem in a single time slot, and the dynamic DT synchronization problem for a finite time horizon, by developing a randomized approximation algorithm at the expense of bounded resource violations for the former, and an online algorithm for the latter. Vaezi et al. [33] developed algorithms for the DT placement problem to minimize the maximum response delay of requests while meeting AoI requirements of requests.

Unlike the aforementioned studies focusing on either object/user mobility or service delays in DT-assisted MEC networks, in this article we deal with DT-assisted service provisioning in MEC networks through efficient DT placements and migrations to minimize the sum of the DT update cost and the total service cost of user requests, by jointly considering the freshness of request results and the resource consumption cost of answering user requests. We address how to synchronize DTs with their objects continuously to keep the DT data as fresh as possible. Meanwhile, we also explore where to place DTs or migrate DTs to appropriate locations to better serve their users and further reduce the service cost.

3 PRELIMINARY

In this section, we first introduce the system model. We then introduce notions, notations, and cost measurements. We finally define problems precisely, and show the NP-hardness of the defined problems.

3.1 System Model

Consider an MEC network $G = (N, E)$, where N is the set of Access Points (APs) and E is the set of links between APs. Associated with each AP, there is a co-located cloudlet. Without loss of generality, we use the AP or its co-located cloudlet interchangeably if no confusion arises. Assume that each cloudlet $c_j \in N$ has storage capacity K and computing capacity C_j. The cost of per unit computing resource consumption of cloudlet c_j is μ_j. There are physical objects and users under the coverage of each AP (or its co-located cloudlet). Both objects and users are allowed to move around in the MEC network.

Let V be the set of physical objects. We assume that each object $v_i \in V$ has a DT, DT_i, that will be placed into a cloudlet in the MEC network G. Object v_i generates data continuously, and the generated data usually is stored in its local storage, and then is uploaded to its DT in the network. Specifically, the update data produced by object $v_i \in V$ is uploaded to its gateway $g(v_i) \in N$ first, then transmitted from gateway $g(v_i)$ to cloudlet $c(DT_i)$ at which DT_i is placed.

Let U be the set of users requesting DT services from a set of objects. The request of each user $u \in U$ is represented by a tuple $(b(u), S_u, comp_u)$ to request the update data of multiple objects by querying their DT data, where $S_u \subseteq V$ is the set of objects that user $u \in U$ requests, $comp_u$ is the amount of computing resource demanded by user u for its request processing, and $b(u)$ is the co-located cloudlet of the AP that user u is under its coverage. The DT data of objects in S_u are transmitted from their hosting cloudlets to the home cloudlet $h(u)$ of user u for processing and analysis. Unless otherwise specified, $h(u)$ usually is $b(u)$. Due to limited computing resource on each cloudlet, the home cloudlet $h(u)$ of user u can be a different cloudlet from $b(u)$.

By leveraging the DT technology and edge computing paradigm, the geographically distributed data that are generated by different objects can be synchronized with DTs and then aggregated to derive synthesized information with effectiveness. For example, Gupta et al. [6] utilized DT and Federated Learning (FL) technologies to detect anomalies in vehicular-IoT systems, such as traffic congestion, collision detection, vehicle breakdown, and traffic violations. The anomaly detection model is trained by incorporating data from various sources such as smart sensors, traffic light data, UAVs, vehicle data, and weather statistics. DTs of objects in the system enable

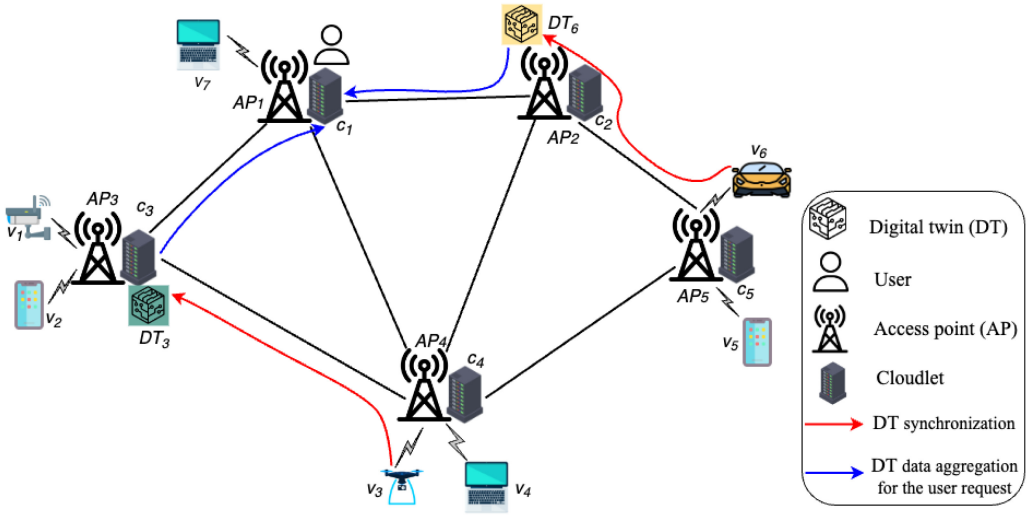


Fig. 1. An illustrative example of an MEC network with DT placements for user requests. A user requests update data of objects v_3 and v_6 by querying their DTs, DT_3 and DT_6. The gateways $g(v_3)$ and $g(v_6)$ of objects v_3 and v_6 are cloudlets c_4 and c_5, respectively. DT_3 of object v_3 is placed at cloudlet c_3, and DT_6 of object v_6 is placed at cloudlet c_2. The home cloudlet of the user is cloudlet c_1.

simulations and analysis of vital factors like traffic flow and traffic trajectories. The integration of DT data from multiple vendors enables more accurate and advanced service provisioning for the vehicular-IoT system.

Figure 1 is an illustrative example of DT-assisted service provisioning in an MEC network, where there are five APs, each of which is co-located with a cloudlet. There are seven objects distributed under the coverage of APs, including mobile phones, laptops, camera, UAV, and vehicle. A user located at AP_1 requests the update data of objects v_3 and v_6 by querying their DTs, DT_3 and DT_6. In this example, DT_3 of object v_3 is placed at cloudlet c_3, and DT_6 of object v_6 is placed at cloudlet c_2. The status information of object v_3 is uploaded from its gateway c_4 and then synchronized with DT_3 at cloudlet c_3. Similarly, object v_6 uploads its update data to its gateway c_5, and the update data then is transmitted to DT_6 at cloudlet c_2. The requested DT data by the user is aggregated at the home cloudlet of the user, cloudlet c_1, for further processing.

3.2 Service Cost Modelling

We deal with user service requests by requesting DT data of objects in an MEC environment. On one hand, each object has an update cost. The update data produced by object $v_i \in V$ is uploaded to its gateway $g(v_i)$ first, then transmitted from gateway $g(v_i)$ to cloudlet $c(DT_i)$ at which DT_i is placed. On the other hand, there is a service cost for each user $u \in U$ requesting for DT data. The requested DT data of objects in S_u are transmitted to the home cloudlet $h(u)$ of user u for processing, and the processing result then is transmitted from $h(u)$ to cloudlet $b(u)$ at which user u is located. Furthermore, DT service provisioning incurs service delays that impacts user experiences on DT services. Thus, there are non-trivial trades-off between the total resource consumption for answering user requests and service delays, which need to be considered in the service cost modelling as follows.

Let P_{c_i,c_j} be the shortest path in the MEC network G between cloudlets c_i and c_j. Then, the communication cost of a unit data transmitted between cloudlets c_i and c_j is $comm(c_i, c_j)$

($= \sum_{e \in P_{c_i,c_j}} \zeta_e$), where ζ_e is the cost of transmitting a unit data along link $e \in E$. The data transmission delay of a unit data in G between cloudlets c_i and c_j is $delay(c_i, c_j)$ ($= \sum_{e \in P_{c_i,c_j}} d_e$), where d_e is the delay of a unit of data transmitted along link $e \in E$.

DT updating cost of an object: The uploading cost of an object from the object itself to its gateway $g(v_i)$ is the product of its transmission power times the uploading duration of its update data. For each object $v_i \in V$, denote by $vol(DT_i)$ the amount of update data uploaded by v_i. Following Shannon's theory, the transmission rate B_{v_i} of object v_i to gateway $g(v_i)$ is defined as $B_{v_i} = W_{g(v_i)} \log\left(1 + \frac{P_{v_i}^{TX} \cdot H^2_{g(v_i),v_i}}{\sigma^2}\right)$, where $W_{g(v_i)}$ is the bandwidth capacity of cloudlet $g(v_i)$, $P_{v_i}^{TX}$ is the transmission power of object v_i, $H^2_{g(v_i),v_i}$ is the channel gain that follows the Rayleigh flat fading under the allocated channel bandwidth, and σ^2 is the white noise.

The updating cost of object v_i is the sum of its uploading cost and its update data transmission cost from gateway $g(v_i)$ to cloudlet $c(DT_i)$ hosting DT_i, which is given by

$$update\_cost(v_i) = \xi \cdot P_{v_i}^{TX} \cdot \frac{vol(DT_i)}{B_{v_i}} + vol(DT_i) \cdot comm(g(v_i), c(DT_i)), \tag{1}$$

where ξ is the cost of a unit transmission power consumption per second.

Service cost of a user request: To provide cost-effective and satisfied DT services for users, we should take into account both the resource consumption and service delay in the service cost modelling for each user request. The duration of an update data from its generation to its first usage is referred to as *the* AoI of the data. Users always want to acquire the up-to-date data from their requested objects as soon as possible, which requires the AoI of the requested data to be as short as possible. Each user $u \in U$ usually requests DT data from multiple objects, the maximum AoI among the received DT data in S_u is referred to as the *end-to-end service delay* of user u.

The end-to-end service delay $d_{e2e}(u)$ of user u includes the uploading delays of the update data from objects in S_u ($\subseteq V$) to their gateways, the data transmission delays between their gateways of objects in S_u and the cloudlets hosting these DTs, and between these DT cloudlets and the home cloudlet $h(u)$ of u, and the processing delay for service model training at home cloudlet $h(u)$ of u. Specifically, the uploading delay of object v_i is $upload\_delay(v_i)$($= \frac{vol(DT_i)}{B_{v_i}}$), and $d_{e2e}(u)$ is defined as follows:

$$\begin{aligned} d_{e2e}(u) = \max_{v_i \in S_u} \{ & upload\_delay(v_i) + vol(DT_i) \cdot (delay((g(v_i), c(DT_i)) \\ & + delay(c(DT_i), h(u)))\} + vol(b(u)) \cdot delay(h(u), b(u)) + \frac{\sum_{v_i \in S_u} vol(DT_i)}{f(comp_u)}, \end{aligned} \tag{2}$$

where $f(comp_u)$ is the data processing rate corresponding to the amounts $comp_u$ of computing resource demanded by user u. If the home cloudlet $h(u)$ of user u is cloudlet $b(u)$ at which the user is located, then $delay(h(u), b(u))$ in Equation (2) is zero. $vol(b(u))$ is the volume of the request result of user u. For the sake of convenience, we assume that the volume of the request result of each user u is the average of its requested DT data, i.e., $vol(b(u)) = \sum_{v \in S_u} vol(DT_v)/|S_u|$.

The service cost $service\_cost(u)$ of user $u \in U$ is the weighted sum of the end-to-end delay cost, the transmission cost of DT update data from their DT cloudlets, and the processing cost of the request by user u, which is defined as

$$\begin{aligned} service\_cost(u) = & \alpha \cdot d_{e2e}(u) + \beta \cdot \sum_{v \in S_u} vol(DT_v) \cdot comm(c(DT_v), h(u)) \\ & + \beta \cdot vol(b(u)) \cdot comm(h(u), b(u)) + \beta \cdot \mu_{h(u)} \cdot comp_u, \end{aligned} \tag{3}$$

where α and β are non-negative constants that strive for a fine balancing between the freshness of a request result and the resource consumption cost for obtaining the request result with $\alpha + \beta = 1$, $comm(h(u), b(u))$ is zero if home cloudlet $h(u)$ of user u is cloudlet $b(u)$, and $\mu_{h(u)}$ is the cost of per unit computing resource consumption of cloudlet $h(u)$.

Total service cost of user requests: Due to limited resource imposed on an MEC network, the network service provider usually allocates its various resources intelligently to its clients – objects and users, to meet their resource demands while providing up-to-date fresh data services for users. The total service cost of users requesting DT data in an MEC network G thus consists of the costs of these two components, i.e., the problem optimization objective is to minimize

$$\beta \sum_{v_i \in V} update\_cost(v_i) + \sum_{u \in U} service\_cost(u), \tag{4}$$

where the first term of Equation (4) is the total cost of the DT update data of all objects in V, and the second term is the total service cost of users in U requesting for DT data, which can be calculated by Equations (1) and (3), respectively. Notice that $service\_cost(u)$ defined by Equation (3) is a weighted sum of the end-to-end delay cost and the resource consumption cost of user u requesting for DT data with coefficients α and β. The larger the value of α, the closer the DTs of objects are placed to their users to reduce user service delays. Conversely, the larger the value of β, the solution will tend to reduce the resource consumption on user services. In extreme case, when $\alpha = 0$ and $\beta = 1$, the optimization objective only considers the resource consumption on DT placements and user service requests. When $\alpha = 1$ and $\beta = 0$, the optimization objective is to minimize the service delay regardless of the resource consumption on user services.

For the sake of convenience, the notations are summarized in Table 1.

3.3 Problem Definitions

In the following, we define the DT placement and migration problems in an MEC network. We first define a static version—the DT placement problem without considering the mobility of objects and users. We then define a dynamic version—the DT migration problem where both users and objects are mobile.

Definition 1. Given an MEC network $G = (N, E)$, there is a set N of APs with each having a co-located cloudlet. Each cloudlet $c_j \in N$ has computing capacity C_j, and storage capacity that can store up to K DTs. There are a set V of objects and a set U of users under the coverage of APs. Each user $u \in U$ issues a service request, represented by a tuple $(b(u), S_u, comp_u)$, requesting DT data from a set S_u of objects with the amount $comp_u$ of computing resource demanded by the request, where $b(u)$ is the location of user u and $S_u \subseteq V$. *The DT placement problem* in G is to place the DTs of all objects in V to cloudlets in N such that the total service cost of user requests defined in Equation (4) is minimized, subject to both computing and storage resource capacities on each cloudlet.

Since we assume that users and objects are mobile in the MEC network, the placed DTs of objects may need to migrate to different cloudlets overtime to minimize the total service cost of their user requests. The DT migration problem then is defined as follows.

Definition 2. Given an MEC network $G = (N, E)$ with initial DT placements of mobile objects in V, a finite time horizon T that is divided into equal time slots, we here consider dynamic user request admissions. For each time slot $t \in T$, there is a set $U'(t)$ of users requesting for DT information of objects in V, and there is a set $V'(t)$ of mobile objects moving to locations different from their locations at time slot $t-1$ with $V'(t) \subseteq V$. We assume that each cloudlet has sufficient computing resource to process all user requests under its coverage, we further assume that the home

Table 1. Table of Notations

Notation	Descriptions
$G = (N, E)$	An MEC network with a set N of APs and a set E of links.
K	The storage capacity of each cloudlet for hosting DTs.
C_j	The computing capacity of cloudlet $c_j \in N$.
ζ_e and d_e	The cost and the delay of transmitting a unit data along link $e \in E$, respectively.
μ_j	The cost of per unit computing resource consumption of cloudlet c_j.
V and U	A set of objects and a set of users.
$g(v_i)$	Gateway cloudlet of object v_i.
DT_i and $c(DT_i)$	DT of object v_i, and the cloudlet at which DT_i is placed.
$vol(DT_i)$	The amount of update data uploaded from v_i to its gateway $g(v_i)$.
B_{v_i}	The transmission rate of object v_i to its gateway $g(v_i)$.
W_{c_j}	The bandwidth of cloudlet $c_j \in N$.
$P_{v_i}^{TX}$	The transmission power of object v_i.
ξ	The cost of a unit transmission power consumption per second.
$(b(u), S_u, comp_u)$	Request of user $u \in U$, where $b(u)$ is the co-located cloudlet of the AP where user u is located, $S_u \subseteq V$ is the set of requested objects, and $comp_u$ is the amount of demanded computing resource for request processing.
$h(u)$	Home cloudlet of user u for request processing.
P_{c_i,c_j}	The shortest path between cloudlets c_i and c_j in the network.
$comm(c_i, c_j)$	The communication cost of a unit data transmitted between cloudlets c_i and c_j.
$delay(c_i, c_j)$	The delay of a unit data transmitted between cloudlets c_i and c_j.
$update\_cost(v_i)$	The updating cost of object v_i.
$upload\_delay(v_i)$	The uploading delay of object v_i.
$d_{e2e}(u)$	The end-to-end service delay of user $u \in U$.
$f(comp_u)$	The data processing rate corresponding to $comp_u$ amounts of computing resource demanded by user u.
$vol(b(u))$	The volume of the request result of user $u \in U$.
$service\_cost(u)$	The service cost of user $u \in U$.
α and β	Non-negative constants that strive for a fine balancing between the freshness of a request result and the resource consumption cost with $\alpha+\beta = 1$.
$G_{v_i} = (N, E; \omega'(\cdot,\cdot))$	A copy of the MEC network $G(N, E)$ for object v_i exclusively, where the cost $\omega'(e)$ of each edge $e \in E$ is defined as $\omega'(e) = \alpha \cdot vol(DT_i) \cdot d_e + \beta \cdot vol(DT_i) \cdot \zeta_e$.
T_{v_i,c_j}	A Steiner tree in G_{v_i} rooted at c_j and spanning home cloudlets of users in $U(v_i)$.
$\omega(v_i, c_j)$	The total service cost of all users in $U(v_i)$ in an approximate Steiner tree T_{v_i,c_j} of G_{v_i}.
$\Delta^t(o(DT_i), o'(DT_i))$	The migration cost gap of object v_i when its DT migrates from location $o(DT_i)$ to $o'(DT_i)$.

cloudlet of each user u is the cloudlet at which the user is located, i.e., $h(u) = b(u), \forall u \in U'(t)$, $t \in T$. *The DT migration problem* in G is to determine which DTs of mobile objects in $V'(t)$ to be migrated to new locations at each time slot t such that the total service cost of all user requests for the given time horizon T is minimized, subject to computing and storage capacities on each cloudlet.

3.4 NP-Hardness

In the following, we show that the NP-completeness of the decision version of the DT placement problem is NP-hard.

Theorem 1. *The DT placement problem in an MEC network $G = (N, E)$ is NP-complete.*

Proof. Given a solution to the decision version of the problem, it can be verified whether the solution is a feasible solution of the problem with the cost no greater than the given value in polynomial time. The decision version of DT placement problem thus is in NP.

We then reduce the 3DM problem in a graph $G' = (X \cup Y \cup Z, E \cap ((X \times Y) \cup (Y \times Z)))$ to a special case of the DT placement problem, where the 3DM problem is defined as follows.

Given an undirected graph $G'(X \cup Y \cup Z, E)$, its vertex set is $X \cup Y \cup Z$ with $X \cap Y = \emptyset$, $Y \cap Z = \emptyset$, $X \cap Z = \emptyset$, and $|X| = |Y| = |Z|$. There are edges between vertices in X and Y, and between vertices in Y and Z, respectively. The decision version of the 3DM problem is to determine whether there is a 3DM prefect matching in G' [5].

We reduce an instance of the 3DM problem to a special instance of the DT placement problem as follows. Assume that each $x \in X$ represents a user, and each user requests the DT of one physical object. Y is the set of cloudlets, and each cloudlet can accommodate one DT only. Z is the set of physical objects. An edge (y, z) represents that the DT of physical object z can be deployed into y and the update cost is no more than a given value L_2, while an edge (x, y) represents that the DT of user x demanded can be deployed at cloudlet y and its associated service cost is no more than L_1. We further assume that none of the users requests the same DT. In other words, different users request for different DTs. Notice that in this reduction, we ignore the end-to-end delay cost in the calculation of the total service cost of users by assuming that they are zeros.

Now, we claim that if there is a feasible solution to this special DT placement problem with the total service cost $|X| \cdot (L_1 + L_2)$, then there is a 3DM perfect matching in G'. We show the claim by contradiction.

It can be seen that there is a one-to-one relationship between a $y \in Y$ and a $z \in Z$ as each physical object has only one DT to be placed while each cloudlet can hold one DT only, the total update cost of the DTs of all physical objects is $L_2 \cdot |Z|$. It can also be seen that there is only a one-to-one relationship between $y \in Y$ and $x \in X$ as each DT is requested by one user only, which implies that no two users request the same DT, and each cloudlet can hold one DT only, the total service cost from DTs to their users is $|X| \cdot L_1$. If there is a solution for the DT placement decision problem with the total service cost of $|X| \cdot L_1 + |Z| \cdot L_2 = |X| \cdot (L_1 + L_2)$ due to $|X| = |Z|$, then there is a 3DM prefect matching in G'. Also, such a reduction can be done in polynomial time. Since the 3DM problem is NP-complete [5], the DT placement problem is NP-complete, too. □

4 ALGORITHMS FOR THE DT PLACEMENT PROBLEM

In this section, we deal with the DT placement problem, which is a joint optimization problem. It is noticed that DT placements impact not only the DT update data cost of all objects but also the total service cost of user requests for DT data. On one hand, we aim to find a scheduler of DT placements such that the update data of objects can be transmitted to their DTs as fast as possible, thereby reducing the DT update communication cost and shortening the update delays of DT states. On the other hand, we should place DTs of objects to cloudlets that are not far away from users to reduce the service cost of user requesting for DT data that consume communication, storage, and computing resources. It thus matter to strive for a non-trivial tradeoff between the end-to-end service delay and the resource consumption for each user request when performing DT placements to cloudlets.

4.1 Algorithm Overview

Due to high computational complexity, we decompose this joint optimization problem into two optimization sub-problems. We first place DTs of objects into cloudlets such that the total DT update data cost of all objects is minimized, subject to the storage capacity on each cloudlet. We then determine the home cloudlet of each user for implementing his or her request that are built upon placed DTs such that the total service cost of user requests is minimized, subject to computing capacity on each cloudlet.

4.2 Algorithm

The proposed algorithm consists of two stages: the DT placement in cloudlets, followed by finding home cloudlets for user requests such that the total service cost of users requesting DT data is minimized. In the following, we detail these two stages.

In the first stage, we determine the DT placement of each object through the construction of a bipartite graph $G_{XY} = (X, Y, E_{XY}; updating(\cdot,\cdot))$, where X $(= V)$ is the set of objects and $Y = \{c_{j,k} \mid 1 \le j \le |N|, 1 \le k \le K\}$ is the set of DT potential locations of objects in cloudlets, as the storage capacity of each cloudlet is upper bounded by K DTs. There is an edge $(v_i, c_{j,k}) \in E_{XY}$ between every vertex $v_i \in X$ and every vertex $c_{j,k} \in Y$. Associated with each edge $(v_i, c_{j,k}) \in E_{XY}$, there is a weight function $updating(\cdot,\cdot)$ if the DT of object v_i is deployed at cloudlet c_j, i.e., $c(DT_i) = c_j$.

Since the total service cost is determined by the end-to-end service delay cost and the resource consumption cost of user requests, the edge weight function $updating(\cdot,\cdot)$ combines both of these two costs into one. The DT update data cost of object v_i assigned to edge $(v_i, c_{j,k})$ can be obtained by Equation (1) if $c(DT_i) = c_j$. The update delay $d(v_i, c_{j,k})$ consists of the update data uploading delay and the data transmission delay between gateway $g(v_i)$ of object v_i and cloudlet $c(DT_i)$, which is defined as $d(v_i, c_{j,k}) = upload\_delay(v_i) + vol(DT_i) \cdot delay(g(v_i), c(DT_i))$. The weight $updating(v_i, c_{j,k})$ of edge $(v_i, c_{j,k})$ thus is $\alpha \cdot d(v_i, c_{j,k}) + \beta \cdot update\_cost(v_i)$.

Having constructed bipartite graph G_{XY}, let M be a minimum-cost maximum matching M in G_{XY}. For each matching edge $(v_i, m(v_i)) \in M$, DT_i of object v_i will be deployed in cloudlet $m(v_i)$, where $m(v_i)$ is another endpoint of the matching edge.

In the second stage, we assume that DTs of all objects in V have been deployed in cloudlets. We now implement user requests to minimize their total service cost. Recall that each user $u \in U$ requests the DT data of objects in set $S_u \subseteq V$ with computing resource demand $comp_u$. The DT update data of objects in S_u will be aggregated at the home cloudlet $h(u)$ of user u in the end.

To process the requested DT data of objects in S_u, the home cloudlet $h(u)$ that meets the computing resource demand $comp_u$ of user u needs to be identified. The rest is to identify the home cloudlet for each user $u \in U$. A weight function $sc(\cdot,\cdot)$ for edges in $G(N, E; sc(\cdot,\cdot))$ is defined as follows:

$$sc(e) = \alpha \cdot d_e + \beta \cdot \zeta_e, \ \forall e \in E, \tag{5}$$

where d_e and ζ_e are the delay and communication costs of a unit data transmission along edge e, respectively. Note that graph $G(N, E; sc(\cdot,\cdot))$ is the MEC network G by assigning each of its edges with the defined weight in Equation (5).

Given user $u \in U$, let $\mathbb{C}_u$ is the set of cloudlets containing the DTs of objects in S_u and cloudlet $b(u)$, i.e., $\mathbb{C}_u = \{c(DT_i) \mid v_i \in S_u\} \cup \{b(u)\}$. For each potential home cloudlet $h(u) \in N$ for user u, we find a shortest-path tree $T_{u,h(u)}$ in $G(N, E; sc(\cdot,\cdot))$ rooted at $h(u)$ and spanning cloudlets in $\mathbb{C}_u$, subject to computing capacity on cloudlet $h(u)$. Notice that cloudlet $b(u)$ hosting user u is included in the shortest path tree, as the request result at cloudlet $h(u)$ must be sent back to cloudlet $b(u)$ at which user u is located. *The home cloud choice problem* is to choose a home cloudlet $h(u)$ for

each user $u \in U$ in the MEC network $G(N, E)$ such that the total service cost of all users in U is minimized.

In the following, the home cloudlet choice problem is reduced to the minimum-cost **generalized assignment problem** (**GAP**) [29], and an approximate solution to the latter in turn returns an approximate solution to the former.

Given $|N|$ bins (cloudlets) with computing capacity C_j of bin $c_j \in N$, there are $|U|$ items with each corresponding to a user request. Each item u has a non-negative weight $comp_u$, which is the amount of computing resource demanded by user u for its request processing. It will incur a cost $SC(u, c_j)$ if cloudlet c_j is chosen as the home cloudlet $h(u)$ of user u, i.e., $h(u) = c_j$. It can be seen that choosing a different home cloudlet for user u results in a different service cost. Recall that T_{u,c_j} is a shortest path tree in $G(N, E; sc(\cdot, \cdot))$ rooted at c_j and spanning cloudlets in $\mathbb{C}_u$ containing the DTs of objects in S_u and cloudlet $b(u)$. The service cost of user u is the sum of the communication cost of the DT update data between each requested DT's cloudlet and home cloudlet $h(u)$, the maximum end-to-end service delay cost among all routing paths for user u, and the request processing cost at cloudlet $h(u)$.

The cost of tree T_{u,c_j} serves as an approximation of the service cost of user u when $c_j = h(u)$. Let $delay(T_{u,c_j})$ be the end-to-end service delay of user u if cloudlet c_j is its home cloudlet, and tree T_{u,c_j} is used for DT data aggregation of user u. Then, the service cost $SC(u, c_j)$ of user u is

$$SC(u, c_j) = \beta \cdot \left(\sum_{v_i \in S_u} \sum_{e \in P^T_{c(DT_i), c_j}} vol(DT_i) \cdot \zeta_e + \mu_j \cdot comp_u + \sum_{e \in P^T_{c_j, b(u)}} vol(b(u)) \cdot \zeta_e \right) + \alpha \cdot delay(T_{u,c_j}), \tag{6}$$

where $P^T_{x,y}$ is the tree path in T_{u,c_i} between cloudlets x and y. In other words, packing item $u \in U$ to bin $c_j \in N$ incurs the service cost $SC(u, c_j)$, subject to capacity C_j on bin c_j.

The minimum-cost GAP is to pack as many items in U as possible to the $|N|$ bins such that the total service cost of the packed items is minimized, subject to computing capacity C_j on each cloudlet c_j with $1 \le j \le |N|$. There is an approximation algorithm for the minimum-cost GAP due to Shmoys and Tardos [29], which delivers an optimal solution at the expense of moderate computing resource violations, i.e., twice the computing resource capacity violations.

The detailed algorithm for the DT placement problem is given in Algorithm 1.

4.3 Algorithm Analysis

The rest is to analyze the time complexity of the proposed algorithm, Algorithm 1.

Theorem 2. *Given an MEC network $G = (N, E)$, the DTs of a set V of objects are to be placed into cloudlets in N, assume that each cloudlet $c_j \in N$ has storage capacity up to K DTs with computing capacity C_j, and a set U of user requests, each user $u \in U$ requests for the DT data from a subset $S_u \subseteq V$ of objects with the amounts $comp_u$ of computing resource demand. There is an efficient algorithm, Algorithm 1, for the DT placement problem in G, which takes $O((|V| \cdot K \cdot |N|)^3 + |U| \cdot |E| \log |N| + (|U| \cdot |N|)^3)$ time.*

Proof. We show that the solution delivered by Algorithm 1 is feasible. In the constructed bipartite graph, each object $v_i \in V$ is only matched with one cloudlet vertex $c_{j,k}$ with $c_j \in N$ and $1 \le k \le K$. We assume that the accumulative storage capacity of all cloudlets suffices to accommodate the DTs of all objects in V. Then, it must exist such a $c_{j,k}$ hosting the DT, DT_i, of object v_i. On the other hand, there are at most K vertices in G_{XY} derived from each cloudlet $c_j \in N$, which implies that there are at most K matching edges associated with these K vertices, i.e., the storage

ALGORITHM 1: An Algorithm for the DT Placement Problem.

Input: An MEC network $G = (N, E)$, a set V of objects, a set U of users with each user u requesting the DT update data from a set $S_u \subset V$ of objects with computing resource demand $comp_u$.

Output: Minimize the total service cost of all user requests through deploying the DTs of all objects to cloudlets and determine the home cloudlet $h(u)$ for each user $u \in U$.

/* Stage one */
$S \leftarrow (\emptyset, \emptyset)$; /* the first component is DT placement and the second component is the service routing */;
$COST \leftarrow 0$; /* the cost of the solution */
Construct a bipartite graph $G_{XY} = (X, Y; E_{XY})$, where X (= V) is the set of objects and Y is the set of potential DT locations of objects in X. There is an edge in E_{XY} between every $x \in X$ and every $y \in Y$ with weight $updating(x, y)$;
Find a minimum cost maximum matching M in G_{XY};
$COST \leftarrow COST + \sum_{e \in M} updating(e)$;
$S \leftarrow (M, \emptyset)$; /* the DT placement in cloudlets */
/* Stage two */
for each user $u \in U$ **do**
  for all $c_i \in N$ **do**
    Construct an auxiliary graph $G(N, E; sc(\cdot, \cdot))$;
    Construct a shortest-path tree T_{u,c_i} in $G(N, E; sc(\cdot, \cdot))$ rooted at c_i and spanning all cloudlets in $\mathbb{C}_u$ in terms of edge weight $sc(x, y)$;
    Calculate the cost $SC(u, c_i)$ of shortest-path tree T_{u,c_i};
Find an approximate solution for the minimum-cost GAP by applying an approximation algorithm due to Shmoys and Tardos [29];
for each user $u \in U$ **do**
  for each cloudlet $c_j \in N$ **do**
    if item u is packed into bin c_j **then**
      $h(u) \leftarrow c_j$;
      $COST \leftarrow COST + SC(u, c_j)$;
      $S \leftarrow S \cup (\emptyset, T_{u,c_j})$;
return Solution S with service cost $COST$.

capacity on each cloudlet will not be violated. The solution delivered by the proposed algorithm thus is feasible.

The proposed algorithm, Algorithm 1, considers both the DT update data cost of objects and the user service cost in the weighted edge functions of auxiliary graphs G_{XY} and $G(N, E; sc(\cdot, \cdot))$ for each user $u \in U$. Finding a minimum-cost maximum matching M in G_{XY} and a single source shortest path tree in $G(N, E; sc(\cdot, \cdot))$ will make use of the edge weights, and the total service cost of all users requesting for DT data in the solution delivered by the proposed algorithm is minimized.

The time complexity analysis of the proposed algorithm is given as follows. It take $O((|X| \cdot |Y|)^3) = O((|V| \cdot K \cdot |N|)^3)$ time to construct graph G_{XY} and find a minimum-cost maximum matching in G_{XY}, while it takes $O(|E| \cdot \log |N|)$ time to find a single source shortest tree rooted at each user $u \in U$ in the MEC network G. Therefore, the proposed algorithm takes $O((|V| \cdot K \cdot |N|)^3 + |U| \cdot |E| \cdot \log |N|)$ time for the DT placement problem. The construction of the minimum-cost GAP takes $O((|U| \cdot |N|)^3)$ time, while finding an approximate solution for the minimum-cost GAP takes $O((|U| \cdot |N|)^3)$ time. Thus, the proposed algorithm for the DT placement problem takes $O((|V| \cdot K \cdot |N|)^3 + |U| \cdot |E| \cdot \log |N| + (|U| \cdot |N|)^3)$ time. □

5 APPROXIMATION ALGORITHM FOR A SPECIAL DT PLACEMENT PROBLEM

In this section, we consider a special DT placement problem where each user $u \in U$ requests the DT data of one object only, i.e., $|S_u| = 1$, for which we propose an approximation algorithm. We assume that each cloudlet has sufficient computing resource to process the DT data requested by users under the coverage of its co-located AP. This implies that the home cloudlet $h(u)$ of each user $u \in U$ is the one at which the user is located, i.e., $h(u) = b(u)$.

5.1 Algorithm Overview

Let $U(v_i)$ be the set of users requesting object v_i with $|U(v_i)| \geq 1$. For the sake of convenience, we assume that DT_i of object $v_i \in V$ is placed in cloudlet $c_j \in N$, i.e., $c(DT_i) = c_j$. The total service cost of users requesting the DT data of object v_i is the sum of its DT update data cost, the data transmission cost of requested DT data between cloudlet c_j and the home cloudlet of each user in $U(v_i)$, and the total end-to-end service delay of all users in $U(v_i)$, which can be calculated by finding a Steiner tree in an auxiliary graph [7]. Note that a Steiner tree in an undirected weighted graph with a given subset of vertices is defined as follows. A Steiner tree in the graph is a tree interconnecting all vertices in the given subset of vertices such that the total weight of the tree is minimized [7].

To obtain the total service cost of users requesting the DT data of object v_i with its DT placed in cloudlet c_j, we construct a Steiner tree T_{v_i,c_j} in an auxiliary graph $G_{v_i} = (N, E; \omega'(\cdot,\cdot))$ rooted at c_j and spanning home cloudlets of users in $U(v_i)$, where graph $G_{v_i} = (N, E; \omega'(\cdot,\cdot))$ is a copy of the MEC network $G(N, E)$ for object v_i exclusively, and the cost $\omega'(e)$ of each edge $e \in E$ is defined as $\omega'(e) = \alpha \cdot vol(DT_i) \cdot d_e + \beta \cdot vol(DT_i) \cdot \zeta_e$.

Since the Steiner tree problem is NP-hard, we instead find an approximate Steiner tree in G_{v_i} for the service cost of all users in $U(v_i)$, because the amount of update data at DT_i transferred between the DT location c_j and the home cloudlets of users in $U(v_i)$ is identical. The end-to-end service delay $d'_{e2e}(u)$ of user $u \in U(v_i)$ is the sum of the uploading delay of the update data of object v_i to cloudlet $g(v_i)$, the transmission delay of the update data between cloudlet $g(v_i)$ and cloudlet $c(DT_i)$, the transmission delay of the request result between cloudlet $c(DT_i)$ and cloudlet $b(u)$, and the data processing delay at cloudlet $b(u)$, i.e.,

$$d'_{e2e}(u) = upload\_delay(v_i) + vol(DT_i) \cdot delay(g(v_i), c(DT_i)) + \sum_{e \in P^T_{c(DT_i),b(u)}} vol(DT_i) \cdot d_e + \frac{vol(DT_i)}{f(comp_u)}, \tag{7}$$

where $P^T_{c(DT_i),b(u)}$ is a routing path in tree T_{v_i,c_j} between cloudlets $c(DT_i)$ and $b(u)$. The sum of terms $upload\_delay(v_i, g(v_i))$ and $vol(DT_i) \cdot delay(g(v_i), c(DT_i))$ is the update delay of DT_i, which is determined by finding a routing path in G_{v_i} between cloudlets $g(v_i)$ and $c(DT_i)$ such that the sum of the delay cost and the communication cost between them is minimized. Finding such a path is NP-hard, as the well-known delay-constrained shortest path problem, which is NP-hard, is a special case of this problem. We here instead consider a simplified case by assuming that the communication cost of each link is proportional to the delay on it. Thus, we find a shortest path $P(g(v_i), c(DT_i))$ in G_{v_i} between $g(v_i)$ and $c(DT_i)$ in terms of link delays, which has the minimum data transmission cost.

For the sake of convenience, we assume that users in $U(v_i)$ come from different cloudlets. The total service cost of users for object v_i is the weighted sum of edge costs in an approximate Steiner tree T_{v_i,c_j} of G_{v_i} due to the NP-hardness of the Steiner tree problem, assuming that DT_i of object

v_i is deployed in cloudlet c_j, i.e., $c(DT_i) = c_j$. The total service cost of all users in $U(v_i)$ thus is

$$\begin{aligned}\omega(v_i, c_j) &= \beta \cdot update\_cost(v_i) + \sum_{u \in U(v_i)} service\_cost(u) \\ &= \beta \cdot \xi \cdot P^{TX}_{v_i} \cdot \frac{vol(DT_i)}{B_{v_i}} + \beta \cdot vol(DT_i) \sum_{e \in P(g(v_i), c_j)} \zeta_e + \alpha \cdot |U(v_i)| \cdot upload\_delay(v_i) \\ &\quad + \alpha \cdot |U(v_i)| \cdot \sum_{e \in P(g(v_i), c_j)} d_e \cdot vol(DT_i) + \alpha \cdot \sum_{u \in U(v_i)} \frac{vol(DT_i)}{f(comp_u)} \\ &\quad + \sum_{e \in T_{v_i, c_j}} \omega'(e) + \beta \cdot \sum_{u \in U(v_i)} \mu_{h(u)} \cdot comp_u, \end{aligned} \tag{8}$$

where T_{v_i, c_j} is an approximate Steiner tree in $G_{v_i}(N, E; \omega'(\cdot, \cdot))$ and $P(g(v_i), c_j)$ is the routing path between gateway $g(v_i)$ and cloudlet c_j hosting DT_i of object v_i. Consider two different objects v and v' with $v \neq v'$, it can be shown that $U(v) \cap U(v') = \emptyset$, implying that there are no users requesting for both objects v and v' simultaneously. The service costs of users requesting for the DT data of different objects thus can be calculated independently.

The special DT placement problem is to find the DT placement of each object $v_i \in V$ such that the total service cost of all users in U is minimized, subject to the storage capacity on each cloudlet.

5.2 Approximation Algorithm

We design an approximation algorithm for the special case of the DT placement problem as follows. The algorithm first constructs a bipartite graph $G_{XY} = (X, Y, E_{XY}; \omega)$, where the vertex sets are $X = V$ and $Y = \{c_{j,k} \mid 1 \leq j \leq |N|, 1 \leq k \leq K\}$. Each vertex in set X corresponds to an object in set V and thus set X contains $|V|$ vertices. Each vertex in set Y represents a potential location to host the DT of an object. Since each cloudlet can accommodate up to K DTs, set Y contains $|N| \cdot K$ vertices. When there is an edge between a vertex in X and a vertex in Y, it implies that the DT of the object in X can be placed in the cloudlet in Y, i.e., if $(v_i, c_{j,k}) \in E_{XY}$, it is feasible that DT_i of object v_i is placed to cloudlet c_j with $1 \leq k \leq K$, and the weight of edge $(v_i, c_{j,k})$ is the total service cost $\omega(v_i, c_j)$ of users in $U(v_i)$, where function $\omega(\cdot, \cdot)$ is a weight function on edges in E_{XY} defined by Equation (8), and its value can be estimated with the help of Steiner tree T_{v_i, c_j}.

The algorithm finds a minimum-cost maximum matching M in G_{XY}, and a solution for the special DT placement problem is derived from M. That is, for each matching edge $(v_i, c_{j,k}) \in M$, DT_i of object v_i will be deployed to cloudlet c_j. Both object v_i and its users in $U(v_i)$ will make use of the approximate Steiner tree T_{v_i, c_j} for the data transmission of DT_i. For each user $u \in U(v_i)$ requesting data in DT_i, the update data of DT_i is transmitted from cloudlet c_j to home cloudlet $h(u)$ $(= b(u))$ of user u along the tree path $P^T_{c_j, b(u)}$.

The detailed algorithm for the special DT placement problem is given in Algorithm 2.

5.3 Algorithm Analysis

Theorem 3. *Given an MEC network $G = (N, E)$, a set V of objects whose DTs are to be placed to cloudlets in N, a set U of users with each user u requesting the DT data of one object only, i.e., $|S_u| = 1$, assume that the cloudlet at which user u is located is its home cloudlet, i.e., $h(u) = b(u)$, and that each cloudlet can host up to K DTs. There is an approximation algorithm, Algorithm 2, with the approximation ratio of 2 for the special DT placement problem, and the time complexity of Algorithm 2 is $O(|V| \cdot |N|^4 + (|V| \cdot K \cdot |N|)^3)$.*

ALGORITHM 2: An Approximation Algorithm for the Special DT Placement Problem.

Input: An MEC network $G = (N, E)$, a set V of objects, and a set U of users with each user $u \in U$ requesting DT data from a subset $S_u \subset V$ of objects with $|S_u| = 1$, assuming the home cloudlet $h(u)$ of each user $u \in U$ is the location it is located.

Output: Minimize the total service cost of user requests through deploying the DTs of objects in V to cloudlets under the storage capacity constraint on each cloudlet.

$S \leftarrow \emptyset$; $COST \leftarrow 0$; /* the solution */
for each $v_i \in V$ **do**
  Calculate $U(v_i)$;
for each $v_i \in V$ **do**
  for each $c_j \in N$ **do**
    Construct an approximate Steiner tree T_{v_i,c_j} in $G(N, E; \omega'(\cdot,\cdot))$ rooted at cloudlet c_j and spanning all home cloudlets of users in $U(v_i)$;
Construct a bipartite graph $G_{XY} = (X, Y, E_{XY}; \omega(\cdot,\cdot))$. There is an edge in E_{XY} between every $v_i \in X$ and every $c_{j,k} \in Y$ with weight $\omega(v_i, c_j)$;
Find a minimum cost maximum matching M in G_{XY};
$COST \leftarrow COST + \sum_{e \in M} \omega(e)$;
for each user $u \in U$ with $S_u = \{v_i\}$ **do**
  Tree $T_{v_i,m(v_i)}$ is used for the DT data transfer from cloudlet $m(v_i)$ to home cloudlet $h(u)$ of user u if edge $(v_i, m(v_i)) \in M$, where cloudlet $m(v_i)$ is the cloudlet in which DT_i of object v_i is placed and $h(u)$ is the home cloudlet of user u to process the DT data;
  $S \leftarrow S \cup \{T_{v_i,m(v_i)} \mid (v_i, m(v_i)) \in M \ \&\ v_i \in V\}$;
return Solution S with the service cost $COST$.

Proof. We first show that the solution delivered by Algorithm 2 is feasible. As the DT of each object is allocated to one cloudlet through the minimum-cost maximum matching M, there is only one DT in the MEC network for the object. We also assume that the accumulative storage capacity of all cloudlets is no less than the total storage demand of the DTs of all objects in V, thus there must have a prefect matching M in graph G_{XY} containing all objects as they are endpoints of the matching edges in the found maximum matching. Meanwhile, each cloudlet contains at most K DTs due to the fact that there are at most K matching edges derived from each cloudlet. The solution delivered by Algorithm 2 thus is feasible.

Since each user requests the DT data of one object only, the total service cost of all users for object v_i is $\omega(v_i, m(v_i))$ if edge $(v_i, m(v_i)) \in M$, where cloudlet $m(v_i) \in N$ is another endpoint of matching edge $(v_i, m(v_i)) \in M$. The total service cost $COST$ of all users in U is the sum of the service costs of users requesting for the DT data of different objects in V, i.e., $COST = \sum_{v_i \in V,\ c_j = m(v_i)} \omega(v_i, c_j)$, where $\omega(v_i, c_j)$ defined in Equation (8) is the total service cost of users in $U(v_i)$. Note that the routing path $P_{delay}(g(v_i), c_j)$ has the shortest delay, and incurs the minimum communication cost of data transmissions between gateway $g(v_i)$ and cloudlet $c(DT_i)$ of object v_i, assuming that the delay along a path is proportional to the length of the path.

Given each user $u \in U(v_i)$, as we assume that cloudlet $b(u)$ at which user u is located is its home cloudlet $h(u)$, then the service cost of all users in $U(v_i)$ is calculated by Equation (8). Regarding term $\sum_{e \in T_{v_i,c_j}} \omega'(e)$ in the right hand side of Equation (8), it is well known that an approximate Steiner tree in an weighted undirected graph can be obtained by applying the approximation algorithm in [7], and the cost of the approximate Steiner tree is twice the optimal cost of the Steiner tree, i.e., $\sum_{e \in T_{v_i,c_j}} \omega'(e) \le 2 \cdot OPT_{v_i,m(v_i)}$, where $OPT_{v_i,m(v_i)}$ is the optimal cost of a Steiner tree in

$G_{v_i}(N, E; \omega'(\cdot,\cdot))$ rooted at $m(v_i)$ and spanning cloudlets at which users in $U(v_i)$ are located, and $m(v_i)$ is another endpoint of matching edge $(v_i, m(v_i))$ in the minimum cost maximum matching M of graph G_{XY} for DT placements. Therefore, the total service cost of users in $U(v_i)$ is no more than twice the optimal one.

We finally analyze the time complexity of the proposed algorithm, Algorithm 2. The construction of a Steiner tree T_{v_i,c_j} in $G(N, E)$ for potential location of DT_i of each object $v_i \in V$ rooted at $c_j \in N$ spanning home cloudlets of users in $U(v_i)$ takes $O(|N|^3)$ time [34]. The construction of G_{XY} takes $O(|V| \cdot K \cdot |N| + |V| \cdot |N| \cdot |N|^3)$ time as there are $|N|$ choices for the DT placement of each object $v_i \in V$, and the construction of a Steiner tree for each potential DT location $c_j \in N$ takes $O(|N|^3)$ time. On the other hand, finding a minimum cost maximum matching in G_{XY} takes $((|V| \cdot K \cdot |N|)^3)$ time. The time complexity of Algorithm 2 thus takes $O(|V| \cdot |N|^4 + (|V| \cdot K \cdot |N|)^3)$ time. □

6 ONLINE ALGORITHM FOR THE DT MIGRATION PROBLEM WITH THE MOBILITY OF USERS AND OBJECTS

In this section, we deal with the DT migration problem under the mobility assumption of users and objects for a finite time horizon. If an object moves out from the coverage range of its original gateway, then its DT data update delay between its new gateway location and its DT location will change due to that a different routing path will be used. The DT update cost of the object will change, too. Consequently, the total service cost of users requesting for the DT data of the object will change.

To mitigate the impact of object movement on the total service cost of users, one promising approach is to migrate the DT of a mobile object from its current location to another location nearby its new gateway to reduce the DT update delay and the update cost of its DT data. However, performing DT migrations poses several challenges. One is to identify an appropriate new DT location for each mobile object to its DT migration, and the new DT location however may not have adequate storage for the DT deployment. Another is that the overhead on DT migrations cannot be ignored as the DT migration may lead to the increase on the DT update cost and the total service cost of users requesting for the DT data. Furthermore, DT migrations also result in changes of routing paths of DT update data between the current locations of objects and new DT locations, and of routing paths of the update data between the new DT locations and home cloudlets of users for requesting the DT data. We thus impose a migration cost threshold θ to determine whether a DT migration should take place. In other words, only if a DT migration can reduce the total service cost of users no less than a given threshold θ, the DT migration then can proceed.

6.1 Online Algorithm

Given a finite horizon T that is slotted into equal time slots, we assume that each mobile user $u \in U(t)$ at a different location can issue a different request for requesting DT data from a set $S_u(t)$ of objects at time slot t. In other words, users are allowed to move around in the network and can issue different queries at different time slots. They also can join in or depart from the system at any time.

Consider a subset $V'(t) \subseteq V$ of mobile objects at time slot t, the DT migration problem is to find a scheduling for DT migrations of mobile objects in $V'(t)$ at each time slot t such that the total service cost of all user requests for the given time horizon is minimized, subject to the storage capacity on each cloudlet, assuming that the home cloudlet of each user is the one at which it is located and has sufficient computing resource for processing all user requests under its coverage.

We consider the migration cost gap of mobile objects with and without DT migrations. Let $o(DT_i)$ and $o'(DT_i)$ be the current location (at the end of time slot $t-1$) and the next locations (at the end of time slot t) of DT_i of mobile object v_i. Recall that $g(v_i)$ and $g'(v_i)$ are the gateways (clouldlets) of object v_i at its current and next locations, respectively. Let $Delay(u, v_i)$ be the end-to-end service delay of user u requesting the DT data of object v_i. Then, we have

$$Delay(u, v_i) = upload\_delay(v_i) + vol(DT_i) \cdot delay(g(v_i), o(DT_i)) + vol(DT_i) \cdot delay(o(DT_i), h(u)) + \frac{\sum_{v \in S_u(t)} vol(DT_v)}{f(comp_u)}. \quad (9)$$

Let $U_t(v_i)$ be the set of users requesting for the DT data of object v_i at time slot t. When DT_i of mobile object v_i is migrated from location $o(DT_i)$ to location $o'(DT_i)$), *the migration cost gap* $\Delta^t(o(DT_i), o'(DT_i))$ of the total service cost of users requesting the DT data of object v_i is defined as follows:

$$\begin{aligned}
&\Delta^t(o(DT_i), o'(DT_i)) \\
&= \beta \cdot upload_cost(v_i, g(v_i)) + \beta \cdot vol(DT_i) \cdot comm(g(v_i), o(DT_i)) \\
&\quad + \beta \sum_{u \in U_t(v_i)} vol(DT_i) \cdot comm(o(DT_i), h(u)) \\
&\quad + \alpha \sum_{u \in U_t(v_i)} \{d_{e2e}(u) \mid \text{if } d_{e2e}(u) = Delay(u, v_i)\} - \beta \cdot upload_cost(v_i, g'(v_i)) \\
&\quad - \beta \cdot vol(DT_i) \cdot comm(g'(v_i), o'(DT_i)) - \beta \sum_{u \in U_t(v_i)} vol(DT_i) \cdot comm(o'(DT_i), h(u)) \\
&\quad - \alpha \sum_{u \in U_t(v_i)} \{d'_{e2e}(u) \mid \text{if } d'_{e2e}(u) = Delay'(u, v_i)\}, \quad (10)
\end{aligned}$$

where $d'_{e2e}(u)$ is the end-to-end service delay of user u when DT_i is migrated from its current location $o(DT_i)$ to a new location $o'(DT_i)$, and $Delay'(u, v_i)$ can be defined similarly as the definition of $Delay(u, v_i)$. The first four terms in the right hand side of Equation (10) are the service cost of users in $U_t(v_i)$ when DT_i of object v_i is located at cloudlet $o(DT_i)$, $upload\_cost(v_i, g(v_i))$ is the uploading cost of object v_i to its gateway $g(v_i)$, $vol(DT_i) \cdot comm(g(v_i), o(DT_i))$ is the communication cost between gateway $g(v_i)$ and the DT location $o(DT_i)$, $\sum_{u \in U_t(v_i)} vol(DT_i) \cdot comm(o(DT_i), h(u))$ is the communication cost between the location of DT_i and home cloudlets of users in $U_t(v_i)$, and $\sum_{u \in U_t(v_i)} \{d_{e2e}(u) \mid \text{if } d_{e2e}(u) = Delay(u, v_i)\}$ is the end-to-end delay cost. If DT_i of object v_i is located at cloudlet $o'(DT_i)$, then the total service cost of users in $U_t(v_i)$ is the sum of the rest of four terms in the right hand side of Equation (10), which can be defined similarly.

It can be seen from Equation (10) that the DT migration of each mobile object $v_i \in V'(t)$ is determined by (i) whether its migration location $o'(DT_i)$ has sufficient storage for the deployment of DT_i, and (ii) whether the migration can reduce the total service cost of users requesting for DT_i data by at least θ in comparison without the migration, i.e., $\Delta^t(o(DT_i), o'(DT_i)) > \theta$. Identifying a migration location $o'(DT_i)$ for hosting DT_i of object v_i is given as follows.

It first calculates $\Delta^t(o(DT_i), c_j)$ for every potential DT migration location $c_j \in N \setminus \{o(DT_i)\}$ for DT_i. It then identifies a location c_{j_0} with the maximum migration cost gap $\Delta^t(o(DT_i), c_{j_0})$ as the migration location of DT_i, i.e., $o'(DT_i) = c_{j_0}$ if $\Delta^t(o(DT_i), c_{j_0}) > \theta$, where

$$\Delta^t(o(DT_i), c_{j_0}) = \max_{c_j \in N \setminus \{o(DT_i)\}} \{\Delta^t(o(DT_i), c_j) \mid \Delta^t(o(DT_i), c_j) > \theta, \text{and } c_j \text{ can host } DT_i\}. \quad (11)$$

The online algorithm for the DT migration problem proceeds iteratively. It first determines which DTs of mobile objects in $V'(t)$ to be migrated by examining DTs of the objects one by one.

Within each iteration, the DT of a mobile object is chosen to migrate from its current location to a new location if the migration results in the maximum migration cost gap among the remaining mobile objects, and the object is removed from $V'(t)$ for further consideration. This procedure continues until none of the remaining mobile objects exists or the maximum migration cost gap among the remaining mobile objects is less than the given threshold θ.

The proposed online algorithm may lead to frequent migration failures due to the unbalancing workload among cloudlets. For example, if a cloudlet is the destination of many DT migrations, then most of these DT migrations are very likely to fail due to limited storage capacity on the cloudlet. To mitigate workload unbalancing among cloudlets, it is desirable to re-balance the cloudlet workload through the re-deployments of DTs of objects in the network. An improved online algorithm for the DT migration problem is then developed, by considering workload balancing. Specifically, the improved online algorithm consists of a two-layer granularity scheduling on DT migrations: the low granularity layer DT migration, and the high granularity layer DT migration. The low layer scheduling is to migrate DTs at each time slot t. However, if the set of mobile objects at time slot t is not empty and the maximum migration cost gap of the remaining mobile objects is no greater than the given threshold θ, then the high layer scheduling is activated to re-balance the workload among cloudlets through invoking Algorithm 1.

The detailed online, two-layer scheduling algorithm for the DT migration problem is given in Algorithm 3. where $service\_cost(u, o'(DT), t)$ is the service cost of user u at time slot t, i.e., $service\_cost(u, o'(DT), t) = service\_cost(u)$, and $service\_cost(u)$ is calculated by Equation (3).

6.2 Algorithm Analysis

Theorem 4. *Given an MEC network $G = (N, E)$ and a finite horizon T that is slotted into equal time slots, a set $U(t)$ of users with each user u requesting the DT data of a subset $S_u(t)$ objects, and a subset $V'(t)$ of mobile objects at time slot $t \in T$, assume that each cloudlet can contain up to K DTs, and the home cloudlet $h(u)$ of each user $u \in U$ is the cloudlet at which it is located with sufficient computing resource for its request processing. There is an online, two-layer scheduling algorithm, Algorithm 3, for the DT migration problem, and its time complexity is $O((|V| \cdot K \cdot |N|)^3 + |U(t)| \cdot |E| \log |N| + (|U(t)| \cdot |N|)^3)$ per time slot t.*

Proof. We first show that the solution delivered by Algorithm 3 is feasible. For each mobile object $v_i \in V'(t)$, we conduct the DT migration only when there is sufficient storage resource at the migration destination location $o'(DT_i)$. Recall that we assumed that each cloudlet has sufficient computing resource for processing all user requests under its coverage. Thus, the storage capacity and computing capacity on any cloudlet will not be violated, and the solution delivered by Algorithm 3 is feasible.

We then analyze the time complexity of Algorithm 3 as follows. Since the calculation of the migration cost gap at each time slot takes $O(1)$ time, the number of iteration for the loop from line 9 to line 19 is $O(|V| \cdot |N|)$. If Algorithm 1 is not invoked at this time slot, the number of iteration in the while loop in line 7 is $O(|V|^2 \cdot |N|)$; otherwise (Algorithm 1 is invoked), the DTs of all objects will be re-deployed by considering potential locations of all mobile objects at this time slot to balance the workload among cloudlets. The most time-consuming component in the while loop is the invoking of Algorithm 1 in line 21. The time complexity of Algorithm 3 at each time slot t thus is the same as the time complexity of Algorithm 1. □

7 PERFORMANCE EVALUATION

In this section, we evaluated the performance of the proposed algorithms for DT placement and migration problems in an MEC network through experimental simulations. We also investigated the impacts of important parameters on the performance of the proposed algorithms.

ALGORITHM 3: An Online Algorithm for the DT Migration Problem with Object Mobility.

Input: An MEC network $G = (N, E)$, a given time horizon T. At each time slot t, a set $U(t)$ of users, and each user $u \in U(t)$ requests the DT data from a subset $S_u(t) \subset V$ of objects, each object v in a subset $V'(t) \subset V$ of objects will move from its current location $g(v) \in N$ to a new location $g'(v) \in N$, and a given migration cost gap threshold θ.

Output: Minimize the total service cost of all users through migrating the DTs of objects in $V'(t)$ between different cloudlets under the storage capacity constraint.

$S \leftarrow \emptyset$; $COST \leftarrow 0$; /* the solution */
for each time slot $t \in \{1, 2, \ldots, T\}$ **do**
  $MO \leftarrow V'(t)$; /* the set of mobile objects at time slot t to be migrated or not */
  $temp \leftarrow 0$; /* temporal value of migration cost gap */
  for each $v_i \in V'(t)$ **do**
    Calculate $U_t(v_i)$; /* user set of object v_i at t */
  while $MO \neq \emptyset$ **do**
    $flag \leftarrow false$; /* mark whether there exists an object in MO satisfying the migration threshold */
    for each $v_i \in MO$ **do**
      $\Delta_i \leftarrow 0$; /* the initial value of the migration cost gap of v_i */
      for each $c_j \in N$ **do**
        if c_j has adequate storage capacity to host DT_i **then**
          $\Delta^t(o(DT_i), c_j) \leftarrow 0$;
          Calculate $\Delta^t(o(DT_i), c_j)$ by Eq (10);
          if $\Delta^t(o(DT_i), c_j) > \max\{\theta, \Delta_i\}$ **then**
            $\Delta_i \leftarrow \Delta^t(o(DT_i), c_j)$; $j' \leftarrow j$;
            $flag \leftarrow true$;
      if $\Delta_i > temp$ **then**
        $temp \leftarrow \Delta_i$; $i_0 \leftarrow i$; $j_0 \leftarrow j'$;
    if $flag \neq true$ **then**
      Redeploy the DTs of all objects by invoking Algorithm 1;
      Break;
    $MO \leftarrow MO \setminus \{v_{i_0}\}$; /* choose an object with the maximum migration cost gap */
    $S \leftarrow S \cup \{(v_{i_0}, c_{j_0}, t) \mid DT_{i_0}$ of object v_{i_0} ismigrated to cloudlet c_{j_0} at time slot $t\}$;
    Migrate DT_{i_0} to cloudlet c_{j_0}, and remove DT_{i_0} from cloudlet $o(DT_{i_0})$;
  $COST \leftarrow COST + \sum_{u \in U(t)} service\_cost(u, o'(DT), t)$;
return S with cost $COST$,

7.1 Experimental Settings

We considered an MEC network $G(N, E)$ with 25 APs, where there is a co-located cloudlet with each AP. The topology of the MEC network is generated via a network topological tool NetworkX [27]. The computing capacity C_j of each cloudlet $c_j \in N$ is randomly selected from the range of [500, 1,000] MHz. The cost of per unit computing resource μ_j is randomly drawn from \$0.1 to \$0.5 per MHz [15], and the bandwidth capacity W_j on the AP co-located with cloudlet c_j is randomly selected from the range of [20, 40] MHz [14]. The data processing rate f of a unit computing resource on each cloudlet is 0.1 MB per millisecond (ms). The communication delay d_e of a unit data transfer on any link $e \in E$ is uniformly distributed over the range of [0.2, 0.5] ms

per MB, and the transmission cost ζ_e of a unit data on link e varies from \$0.01 to \$0.05 per MB randomly [36].

Objects and users are randomly distributed under the coverage of APs. The volume $vol(DT_i)$ of the update data from object $v_i \in V$ to its DT is randomly selected in the range of [10, 20] MB. The transmission power $P_{v_i}^{TX}$ of object v_i is uniformly distributed over the range of [30, 40] units, and the cost of a unit data transmission power consumption of any object ξ is \$0.4. The amount $comp_u$ of computing resource demanded by each user request $u \in U$ is uniformly distributed over the range of [30, 50] MHz [14]. The values of coefficients α and β in the optimization objective are set to 0.5 in the default setting. The storage capacity K (in terms of numbers of DTs) on a cloudlet is randomly selected from the range of [100, 200]. We refer to Algorithms 1–3 as Algorithms 1–3 for short, respectively.

To evaluate the performance of Algorithms 1 and 2 for the DT placement problem, a greedy algorithm `Greedy` is proposed. Specifically, it proceeds as follows. For the DT placement of each object $v_i \in V$, the cloudlet with sufficient storage capacity and the minimum routing cost between gateway $g(v_i)$ of v_i and the cloudlet in terms of edge weight function $sc(\cdot, \cdot)$ defined in Equation (5) is selected as its DT cloudlet $c(DT_i)$. The home cloudlet of each user $u \in U$ is the cloudlet at which the user is located by default, i.e., $h(u) = b(u)$. If the available computing capacity of cloudlet $b(u)$ is less than the demanded computing resource $comp_u$ by user u, another cloudlet with sufficient available computing capacity and the minimum routing cost from cloudlet $b(u)$ to it is selected as the home cloudlet $h(u)$ of user u.

To evaluate the performance of Algorithm 3 for the DT migration problem, we considered one benchmark without DT migrations, referred to as `Static`. The DT placement of each object in `Static` remains unchanged throughout all time slots, regardless of the mobility of objects and users.

The value in each figure is the mean of the results out of 100 MEC instances with the same network size. The actual running time of each algorithm is obtained on a desktop equipped with an Intel(R) Xeon(R) Platinum 8280 2.70 GHz CPU, with 10 GB RAM. These parameters are adopted as the default settings unless otherwise specified.

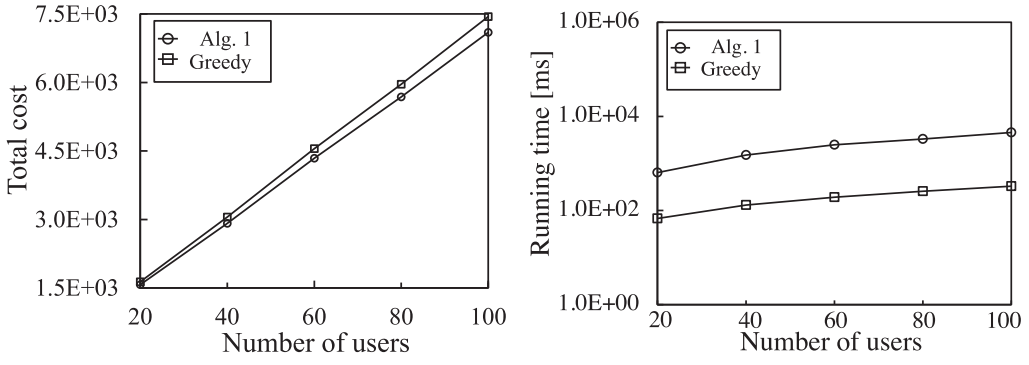
7.2 Performance of Different Algorithms for the DT Placement Problem

In the following, we studied the performance of Algorithms 1 and 2, and `Greedy` for the DT placement problem, by varying the number of objects while fixing $|U| = 50$ and the number of users while fixing $|V| = 50$, respectively. We also investigated impacts of different parameters on the performance of the proposed algorithms.

We first studied the performance of algorithms Algorithm 1 and `Greedy` for the DT placement problem, where the results are shown in Figures 2 and 3, respectively.

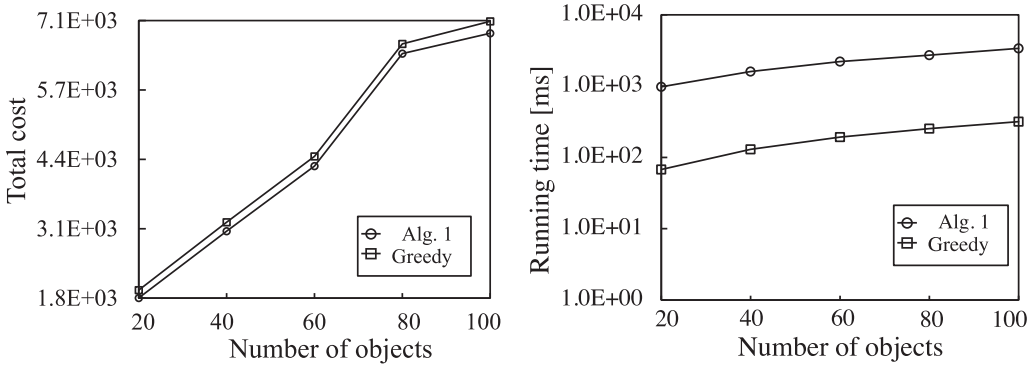
It can be seen from Figures 2 and 3 that Algorithm 1 has a less total service cost than that of `Greedy` at the expense of a longer running time. This is because Algorithm 1 considers both DT and home cloudlet of users placements to reduce the total service cost compared with `Greedy`.

We then evaluated the performance of the proposed approximation algorithm, Algorithm 2, for the special DT placement problem in which each user requests the DT data of one object only. Figures 4 and 5 plot the performance curves of Algorithms 1 and 2, and `Greedy`, respectively. It can be seen from Figures 4(a) and 5(a) that Algorithm 1 has the lowest total service cost among the three comparison algorithms. The home cloudlet of each user in Algorithm 2 or `Greedy` is the cloudlet at which the user is located by default. In Algorithm 1, the home cloudlet selection of each user is related to the DT placement of objects in the first stage, which is more flexible and enables Algorithm 1 to achieve a lower total service cost than those of the other two algorithms.



(a) Total service cost (b) Running time

Fig. 2. Performance of different algorithms for the DT placement problem, by varying the number $|U|$ of users while fixing $|V| = 50$.

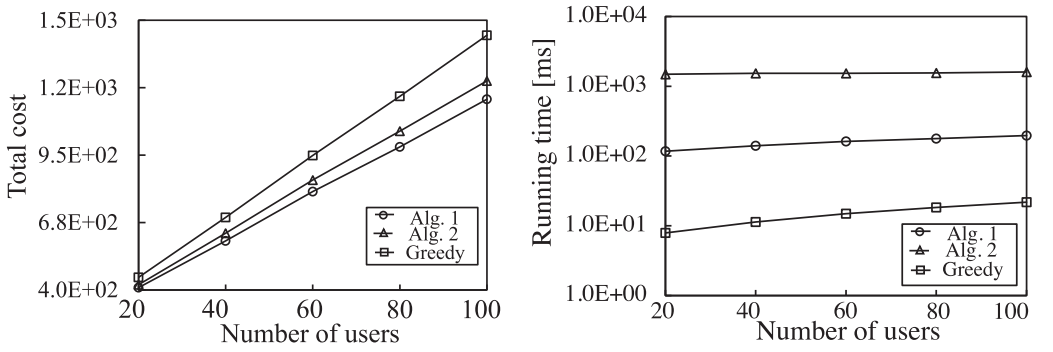


(a) Total service cost (b) Running time

Fig. 3. Performance of different algorithms for the DT placement problem, by varying the number $|V|$ of objects while fixing $|U| = 50$.

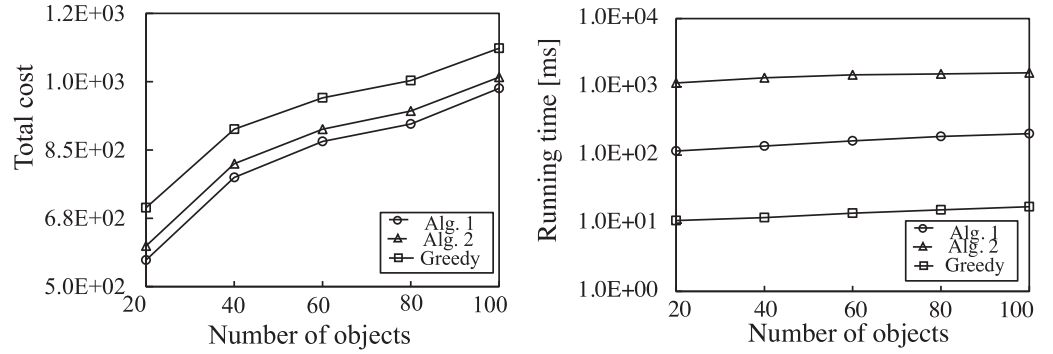
We studied the impact of network size $|N|$ on the scalability of the mentioned three algorithms. We vary the network size $|N|$ from 50 to 250 while fixing the numbers of users and objects at 100, i.e., $|U| = 100$ and $|V| = 100$. Figure 6 plots the performance curves of the three comparison algorithms. It is noticed that the total service cost of each of the algorithms in the network with size $|N| = 250$ is the largest one due to the fact that a larger network size leads to a longer routing delay.

We finally examined the impact of weighting coefficients α and β on the performance of the three comparison algorithms, by varying their values while fixing the numbers of objects and users at 50 and 50, respectively. Figure 7 plots the total service cost curves of the three algorithms, from which it can be seen that the total service cost of each of the algorithms decreases with the growth of the value of α. This indicates that the cost of resource consumption is dominant of the total service cost in comparison with the end-to-end delay cost. A larger α implies that the solution more favours the freshness of update data, and conversely it is conducive to control the resource consumption on user service requests. The service provider can adjust the values of α and β according to its service demands.



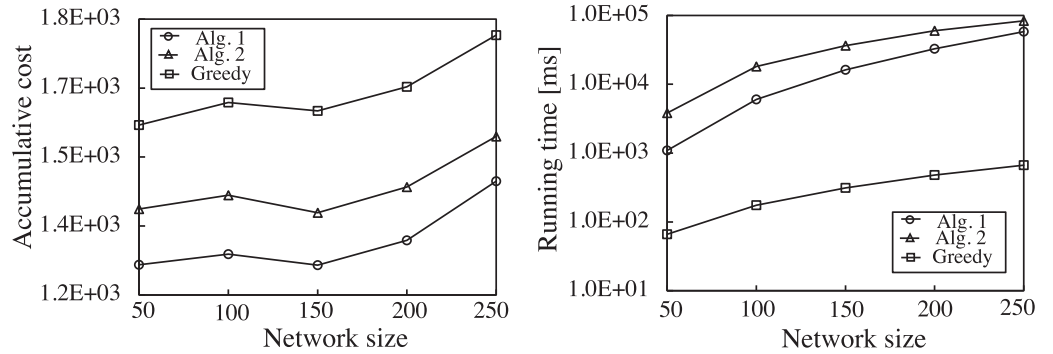
(a) Total service cost (b) Running time

Fig. 4. Performance of the approximation algorithm for the special DT placement problem, by varying the number $|U|$ of users while fixing $|V| = 50$.



(a) Total service cost (b) Running time

Fig. 5. Performance of the approximation algorithm for the special DT placement problem, by varying the number $|V|$ of objects while fixing $|U| = 50$.



(a) Total service cost (b) Running time

Fig. 6. Impact of parameter $|N|$ on the performance of different algorithms for the DT placement problem.

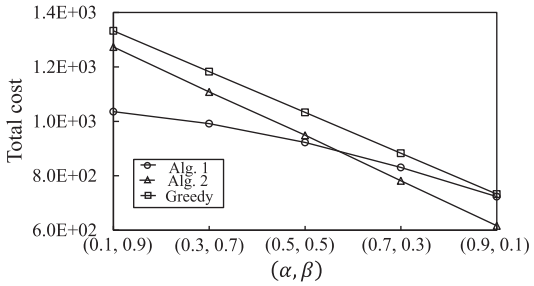
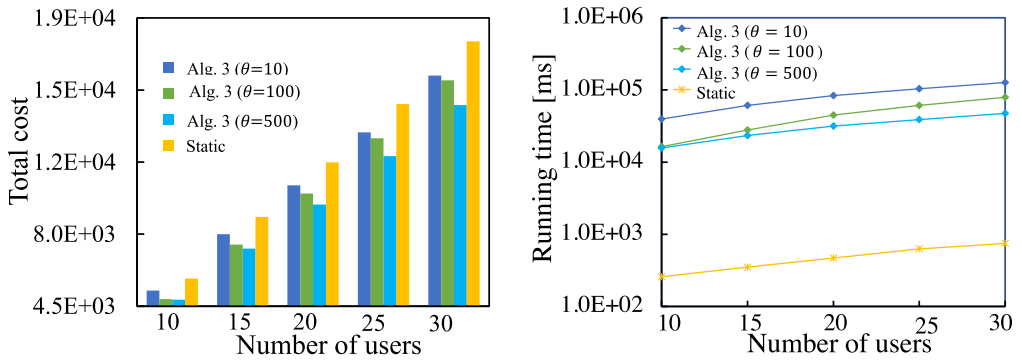


Fig. 7. Impact of α and β on the performance of Algorithms 1 and 2, and Greedy for the DT placement problem.



(a) Total service cost (b) Running time

Fig. 8. Performance of different algorithms for the DT migration problem.

7.3 Performance of Different Algorithms for the DT Migration Problem

The rest is to investigate the performance of Algorithm 3 and the benchmark `Static` for the DT migration problem, by varying the value of threshold θ and the number $|U|$ of users while fixing the number of objects at 10. We assumed that the monitoring period consists of T (=20) equal time slots [15], and objects move around within the MEC network randomly at different time slots.

Figure 8 illustrates the results of Algorithm 3 and `Static`. It can be seen from Figure 8(a) that the reduction on the total service cost by Algorithm 3 compared to `Static` becomes larger and larger with the increase on the number of users, since proper DT migrations can be beneficial to most users when most of the users request for moving objects. It is observed that an increasing value of θ leads to a lower total service cost by Algorithm 3, in addition to reducing the running time of the algorithm (see Figure 8(b)). The rationale behind is that when the value of θ becomes larger, less numbers of DT migrations of objects can meet the cost threshold θ, this triggers the high layer scheduling, thereby reducing the algorithm running time on the low layer scheduling.

8 CONCLUSION

In this article, we studied novel DT placement and migration problems in an MEC network with the mobility assumption of both objects and users, subject to computing and storage resource capacities on each cloudlet. We aim to minimize the sum of the DT update cost and the total service cost of users requesting for DT data. To this end, we first proposed an efficient algorithm for the DT placement problem, through a reduction to the weighted maximum matching and

minimum-cost GAPs. We then devised an approximation algorithm for a special DT placement problem where each user requests only the DT data of one object. meanwhile, we also investigated the DT migration problem with the mobility assumption of both objects and users within a finite time horizon, for which we devised a two-layer scheduling algorithm for DT migrations to reduce the total service cost of user requests within the given time horizon. We finally evaluated the performance of the proposed algorithms through simulations. Simulation results demonstrate that the proposed algorithms are promising.

ACKNOWLEDGEMENT

We appreciate the editor and three anonymous referees for their constructive comments and valuable suggestions, which helped us to improve the quality and presentation of the article significantly. .

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805.
[2] L. Corneo, C. Rohner, and P. Gunningberg. 2019. Age of information-aware scheduling for timely and scalable internet of things applications. In *Proceedings of the INFOCOM*. IEEE, 2476–2484.
[3] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang. 2021. Deep reinforcement learning for stochastic computation offloading in digital twin networks. *IEEE Transactions on Industrial Informatics* 8, 4 (2021), 2276–2288.
[4] B. Fan, Y. Wu, Z. He, Y. Chen, T. Q. S. Quek, and C. Z. Xu. 2022. Digital twin empowered mobile edge computing for intelligent vehicular lane-changing. *IEEE Network* 35, 6 (2022), 194–201.
[5] M. R. Garey and D. S. Johnson. 1990. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co..
[6] D. Gupta, S. S. Moni, and A. S. Tosun. 2023. Integration of digital twin and federated learning for securing vehicular Internet of Things. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems* (RACS'23), ACM.
[7] L. Kou, G. Markowsky, and L. Berman. 1981. A fast algorithm for Steiner trees. *Acta Informatica* 15, (1981), 141–145.
[8] L. Lei, G. Shen, L. Zhang, and Z. Li. 2021. Toward intelligent cooperation of UAV swarms: When machine learning meets digital twin. *IEEE Network* 35, 1 (2021), 386–392.
[9] B. Li, Y. Liu, L. Tan, H. Pan, and Y. Zhang. 2022. Digital twin assisted task offloading for aerial edge computing and networks. *IEEE Transactions on Vehicular Technology* 71, 10 (2022), 10863–10877.
[10] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, W. Xu, and A. Y. Zomaya. 2024. Digital twin-assisted, SFC-enabled service provisioning in mobile edge computing. *IEEE Transactions on Mobile Computing* 23, 1 (2024), 393–408.
[11] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Z. Xu, and W. Xu. 2024. AoI-aware user service satisfaction enhancement in digital twin-empowered edge computing. *IEEE/ACM Transactions on Networking*, 32, 2 (2024), 1556–1572. DOI: 10.1109/TNET.2023.3324704
[12] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya. 2023. Digital twin-enabled service satisfaction enhancement in edge computing. In *Proceedings of the INFOCOM'23*. IEEE, 1–10.
[13] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Y. Zeng, B. Ye, and X. Jia. 2023. Digital twin-enabled service provisioning in edge computing via continual learning. *IEEE Transactions on Mobile Computing*. DOI: 10.1109/TMC.2023.3332668
[14] J. Li, W. Liang, W. Xu, Z. Xu, and J. Zhao. 2020. Maximizing the quality of user experience of using services in edge computing for delay-sensitive IoT applications. In *Proceedings of the MSWiM'20*. ACM, 113–121.
[15] J. Li, W. Liang, M. Chen, and Z. Xu. 2021. Mobility-aware dynamic service placement in D2D-Assisted MEC environments. In *Proceedings of the WCNC'21*. IEEE, 1–6.
[16] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, A. Zomaya, and S. Guo. 2023. Budget-aware user satisfaction maximization on service provisioning in mobile edge computing. *IEEE Transactions on Mobile Computing* 22, 12 (2023), 7057–7069.
[17] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, and J. Zhao. 2022. Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 5 (2022), 1199–1212.
[18] X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia. 2023. Multiple service model refreshments in digital twin-empowered edge computing. *IEEE Transactions on Services Computing*. DOI: 10.1109/TSC.2023.3341988
[19] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani. 2023. Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach. *IEEE Transactions on Mobile Computing* 22, 4 (2023), 2402–2416.
[20] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng. 2022. Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network. *IEEE Internet of Things Journal* 9, 2 (2022), 1427–1444.

[21] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang. 2021. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet of Things Journal* 8, 4 (2021), 2276–2288.
[22] Y. Lu, S. Maharjan, and Y. Zhang. 2021. Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet of Things Journal* 8, 22 (2021), 16219–16230.
[23] Y. Ma, W. Liang, M. Huang, W. Xu, and S. Guo. 2022. Virtual network function service provisioning in MEC via trading off the usages between computing and communication resources. *IEEE Transactions on Cloud Computing* 10, 4 (2022), 2949–2963.
[24] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo. 2022. Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks. *IEEE Transactions on Mobile Computing* 21, 1 (2022), 196–210.
[25] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2322–2358.
[26] R. Minerva, G. M. Lee, and N. Crespi. 2020. Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE* 108, 10 (2020), 1785–1824.
[27] NetworkX. Retrieved from July 2022 https://networkx.org/
[28] Y. Ren, S. Guo, B. Cao, and X. Qiu. 2024. 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* 23, 5 (2024), 4405–4422.
[29] D. Shomys and E. Tardos. 1993. An approximation algorithm for the generalized assignment problem. *Mathematical Programming* 62, (1993), 461–474.
[30] W. Sun, H. Zhang, R. Wang, and Y. Zhang. 2020. Reducing offloading latency for digital twin edge networks in 6G. *IEEE Transactions on Vehicular Technology* 69, 10 (2020), 12240–12251.
[31] F. Tang, X. Chen, T. K. Rodrigues, M. Zhao, and N. Kato. 2022. Survey on digital twin edge networks (DITEN) towards 6G. *IEEE Open Journal of the Communications Society* 3, (2022), 1360–1381.
[32] L. Tang, Y. Du, Q. Liu, J. Li, S. Li, and Q. Chen. 2023. Digital twin assisted resource allocation for network slicing in industry 4.0 and beyond using distributed deep reinforcement learning. *IEEE Internet of Things Journal* 10, 19 (2023), 16989–17006.
[33] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, and G. Karakostas. 2023. Digital twin placement for minimum application request delay with data age targets. *IEEE Internet of Things Journal* 10, 13 (2023), 11547–11557.
[34] V. Vazirani. 2001. *Approximation Algorithms*. Springer.
[35] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari. 2022. Mobility digital twin: Concept, architecture, case study, and future challenges. *IEEE Internet of Things Journal* 9, 18 (2022), 17452–17467.
[36] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. 2019. Efficient NFV-enabled multicasting in SDNs. *IEEE Transactions on Communications*, 67, 3 (2019), 2052–2070.
[37] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. 2019. Task offloading with network function requirements in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing* 18, 11 (2019), 2673–2685.
[38] R. Zhang, Z. Xie, D. Yu, W. Liang, and X. Chang. 2024. Digital twin-assisted federated learning service provisioning over mobile edge networks. *IEEE Transactions on Computers* 73, 2 (2024), 586–598.
[39] L. Zhao, Z. Bi, A. Hawbani, K. Yu, Y. Zhang, and M. Guizani. 2023. ELITE: An intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks. *IEEE Transactions on Mobile Computing* 22, 9 (2023), 5231–5247.

Received 10 October 2023; revised 24 March 2024; accepted 9 April 2024