Digital Twin Freshness Maximization in Edge Computing

Jing Li, Jianping Wang, *Fellow, IEEE*, Weifa Liang, *Fellow, IEEE*, Quan Chen, *Member, IEEE*, Sajal K. Das, *Fellow, IEEE*, and Xiaohua Jia, *Fellow, IEEE*

***Abstract*—Mobile Edge Computing (MEC) shifts powerful computing resource provisioning from remote powerful data centers to the edge of core networks. Meanwhile, Digital Twin (DT) has surfaced as a promising technology to provide comprehensive and dynamic descriptions of physical objects in cyberspace with bidirectional and real-time interactions. Moreover, Internet of Things (IoT) devices have contributed abundant, heterogeneous and continuous data from interconnected devices to the explosion of DTs. With technologies evolution, there is an increasing necessity to address the freshness of both DT states and DT data, through timely synchronizations between DTs and their objects in a highly dynamic IoT environment. In this paper, we develop innovative methodologies to improve the DT freshness while minimizing the cost of various resources consumed on the DT freshness improvement. Specifically, we first formulate two novel optimization problems of DT freshness: (i) The static DT freshness optimization problem, where all DT synchronization tasks are given in advance; and (ii) the dynamic DT freshness optimization problem without any knowledge of future DT synchronization tasks over a given finite time. We then devise an approximation algorithm with a provable approximation ratio for the static DT freshness optimization problem. Also, we develop an online algorithm with a provable competitive ratio for the dynamic DT freshness optimization problem. Finally, we evaluate the performance of the proposed algorithms through simulations. Simulation results show that the proposed algorithms outperform their comparison baselines by no less than 13.2%.**

***Index Terms*—Digital twin (DT), DT synchronization, mobile edge computing, cost modelling, online and approximation algorithms, and resource allocation.**

Received 12 June 2025; revised 7 December 2025; accepted 25 December 2025. Date of publication 12 January 2026; date of current version 10 April 2026. The work of Sajal K. Das was supported by the U.S. National Science Foundation under Grant EPCN-2319995 (SOTERIA: Satisfaction and Risk-aware Dynamic Resource Orchestration in Public Safety Systems), Grant PFI-2431990 (Smart Connected Farms: Pest Management in Agriculture through AI and IoT), and Grant AI-ENGAGE-2520346 (HARVEST: Holistic AI-powered Agricultural Response Validation and Early Prediction System across Territories). *(Corresponding author: Jianping Wang.)*

Jing Li, Jianping Wang, Weifa Liang, and Xiaohua Jia are with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China (e-mail: jing.li@cityu.edu.hk; jianwang@cityu.edu.hk; weifa.liang@cityu.edu.hk; csjia@cityu.edu.hk).

Quan Chen is with the School of Computers, Guangdong University of Technology, Guangzhou 510006, China (e-mail: quan.c@gdut.edu.cn).

Sajal K. Das is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409 USA (e-mail: sdas@mst.edu).

Digital Object Identifier 10.1109/TSC.2026.3651602

I. Introduction

USHERING into the era of the Internet of Things (IoT), the physical world is witnessing an unprecedented proliferation of Big Data, primarily driven by the exponential growth of IoT devices and services [38]. It is projected that the number of IoT devices worldwide will surpass 30 billion by 2027 [44]. Digital Twins (DTs) as virtual representations of real-world entities are poised to offer a compelling technology capable of integrating enormous volumes of IoT data, including historical and real-time sensor data, for achieving effective digital visualization and comprehensive analysis [10], while the insights and predictions derived from DTs can be used for emulating behaviors of physical entities and optimizing their operations as well as enhancing their efficiency [5], [7]. Indeed, DTs bridge the physical and cyber dimensions by establishing high-fidelity virtual representations of physical objects in virtual worlds, which rely on processing the real-time data from objects in a continuous and dynamic way [9], [33].

Conventional remote cloud platforms usually fall short in supporting applications having real-time requirements in the continuous evolution of DTs, because clouds being typically located at considerable distances from physical objects [43], incur significant communication delays in addition to making the undesirable staleness of DTs [12]. To this end, Mobile Edge Computing (MEC) has surfaced through a highly viable solution for shifting cloud-like computing resource to the proximity of physical objects at network edge, thereby reducing latency and enhancing the overall responsiveness of the system [4], [16], [30]. The deployment of DTs in MEC environments holds a great promise for improving the freshness and accuracy of DTs through timely synchronization and real-time feedback, facilitating optimized physical operations and decision-making across diverse sectors such as autonomous vehicles and smart manufacturing [35].

This paper delves into investigating efficient strategies for improving the freshness of DTs in MEC networks, while minimizing the resource cost associated with such improvements. As illustrated in Fig. 1, each physical object is paired with a DT placed in a cloudlet within an MEC network, and each object will periodically upload new data to its corresponding DT for the continuous synchronization by issuing synchronization tasks at some time slots during a monitoring time horizon. For example, consider a scenario where there exists a set of vehicles in an MEC network with their DTs established in cloudlets,

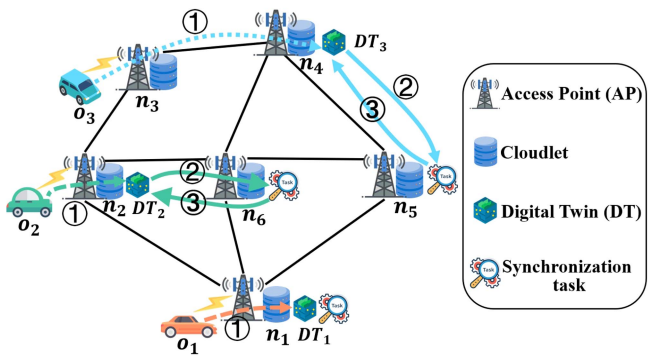


Fig. 1. A DT-assisted MEC network, where each Access Point (AP) has a co-located cloudlet. IoT devices (physical objects) o_1, o_2 and o_3 have their DTs (DT_1, DT_2 and DT_3) deployed in cloudlets n_1, n_2 and n_4, respectively. ① An object may generate and upload new data to the cloudlet holding its DT in a time slot, while issuing a synchronization task for updating its DT. For example, DT_1 processes its synchronization task in its local cloudlet n_1. Meanwhile, DT_2 and DT_3 offload their synchronization tasks to cloudlets n_6 and n_5, respectively, i.e., ② they transmit the new data to the chosen cloudlets for processing, and ③ the processing results are sent back to the cloudlets hosting the DTs.

and it is crucial to maintain the freshness of these DTs for a variety of applications, such as traffic management and vehicle maintenance. However, optimizing the freshness of DTs in an MEC network through DT synchronization scheduling poses the following significant challenges.

1) *How to determine where to perform a DT synchronization?* A cloudlet, while providing low-latency computing capabilities, typically possesses limited computing resources, compared to centralized clouds. This resource limitation creates a fundamental tension: when multiple DTs hosted on the same cloudlet require simultaneous synchronizations (particularly during peak demand periods), the cloudlet may lack sufficient computing capacity to process all synchronization tasks in a timely way. To resolve this issue, a DT in the cloudlet can offload its synchronization to another cloudlet with sufficient computing resource by transmitting its new update data to the chosen cloudlet for processing, and then sending the result back to the original cloudlet [13]. However, such a synchronization offloading will inevitably lead to non-negligible overheads, thus necessitating to determine whether to conduct a synchronization of a DT in its hosting cloudlet or another cloudlet in the MEC network possessing sufficient computing resources.

2) *How to determine when to conduct a DT synchronization?* Considering the MEC system running in discrete time slots, we need to determine whether to conduct a synchronization of a DT in the current or a future time slot. Choosing to conduct the DT synchronizations in future time slots can be due to several reasons, e.g., there may be a large amount of new data generated from objects in the current time slot, whereas there exists insufficient computing resource within the MEC network to process all such new data. Therefore, the MEC network service provider may intend to postpone the synchronizations of some DTs. However, this may be at the expense of freshness loss of such DT synchronizations.

3) *How to jointly optimize the freshness and cost of DT synchronizations in an online manner?* Synchronization tasks are often issued dynamically, owing to the uncertain dynamics in IoT environments. This means there usually exists no knowledge of future synchronization tasks, and it is important to dynamically optimize the freshness of DT synchronizations to facilitate the quality enhancement of DT-enabled services, while optimizing the incurred cost.

The novelty of the work lies in jointly optimizing the freshness and cost of performing DT synchronizations for achieving efficient DT service provisioning, which involves determining where and when to conduct each DT synchronization task in an MEC network. To achieve this objective, we formulate two novel optimization problems for DT synchronizations, and solve them by developing an approximation algorithm and an online algorithm with guaranteed performance, considering static and dynamic synchronization task arrivals, respectively. We also provide theoretical analysis on the approximation ratio of the proposed approximation algorithm and the competitive ratio of the online algorithm, respectively.

The main contributions of this paper are summarized as follows.

- We formulate a static DT freshness optimization problem when all synchronization tasks are given and a dynamic DT freshness optimization problem without any knowledge of future DT synchronization tasks for a finite time horizon. We also show NP-hardness of the defined problems.
- For the static DT freshness optimization problem, we develop a constant approximation algorithm.
- For the dynamic DT freshness optimization problem, we devise an online algorithm with a provable competitive ratio.
- We evaluate the performance of the proposed algorithms for both static and dynamic DT freshness optimization problems by simulations. The simulation results demonstrate the two proposed algorithms are promising, improving the performance of the problems by no less than 13.2%, compared to the other comparison benchmarks.

The organization of the paper is organized as follows. Section II offers a comprehensive review of existing literature concerning DT synchronizations within MEC environments. Section III provides the system and cost models, while Section III-E outlines the problem definitions and proves their NP-hardness. Section IV designs an approximation algorithm for static DT freshness optimization. Section V designs an online algorithm for dynamic DT freshness optimization. The performance of these algorithms is evaluated in Section VI, while the conclusion is provided in Section VII.

II. Related Work

DT-assisted MEC: Empowered by the powerful computation, communication and storage in cyberspace, the DT technology offers to portray the physical world in a digital way, and has attracted much attention recently [10], [14], [19], [20], [26], [30], [30], [36], [44], [46], [48], [49], [50]. The authors in [10] designed a satellite-terrestrial scheme for cooperative edge computing. They leveraged a DT-based cloud to make decisions for resource sharing, while offering a Deep Reinforcement

Learning (DRL) method to reduce the experienced delay and the amount of satellite energy consumed. Approximation and online algorithms were proposed in [14] for DT replica placements in cloudlets to provide delay-sensitive DT-assisted services in serverless edge computing, with the assumption of high user mobility. The authors in [47] dealt with mobility-aware service provisioning through DT replica deployments and migrations in MEC. They developed a randomized algorithm with high probability for given service requests, together with a DRL algorithm for dynamic service requests within a given time horizon. The authors in [20] proposed a mechanism, AutoOPT, to automatically generate the data provided for building real-time and accurate DT networks, while optimizing the data quality, based on generative data-centric AI models. The authors in [26] explored the incentive mechanism to facilitate the trading between the DT service providers and network resource providers. They proposed a DRL-based methodology to maximize the revenue of DT service providers, while designing a semantic communication method to mitigate the data collection cost. Long-term performance optimization in DT-empowered MEC networks was suggested in [30], which adopted a Lyapunov optimization method, and proposed a long-term queue-aware scheme to minimize the energy consumption, by managing computing and communication resources. The authors in [3] proposed a blended deep learning framework to detect different attacks throughout data communication at the network edge with high accuracy and efficiency. The authors in [29] proposed a digital network twin-based framework to achieve the joint synchronization and localization in 5G networks.

The authors in [31] presented a dual-asynchronous federated learning approach tailored for Edge-AI-as-a-Service (EAaaS) within an edge-cloud continuum and leveraged the blockchain technique to achieve decentralized consensus and data integrity, where DTs were utilized to conduct global aggregation to enhance model accuracy and efficiency. The authors in [36] proposed an in-band network telemetry-based mechanism to collect data for building a DT network to mirror a physical network, and developed algorithms to plan paths and synchronize multi-path telemetry. A cooperative collection algorithm was proposed in [49] to build a federated framework for enabling DTs to precisely imitate the real-world systems in real time with the assistance of Unmanned Aerial Vehicles (UAVs) to track physical objects with high dynamics. Partial offloading in DT-assisted vehicle edge computing was investigated in [48], which proposed an efficient scheme to facilitate vehicle cooperation and improve offloading effectiveness, based on a designed DRL approach. The adaptive resource allocation for DT synchronization was studied in [34], which proposed a continual reinforcement learning algorithm to minimize the long-term mismatch. The authors in [25] incorporated DT synchronization and migration to devise a two-timescale framework, which considers the DT synchronization failure during the migration process. They also designed a DRL algorithm to minimize the long-term average energy consumption. These studies however did not consider the freshness improvement of DTs in MEC environments.

Freshness-aware DT service provisioning: There exist several studies on improving the freshness of DTs [1], [2], [9], [11], [12], [13], [18], [22], [23], [32], [35], [45]. Movable IoT devices were leveraged in [9] to collect the state data from physical objects via DT synchronizations. It modelled the freshness of a DT using a value metric, and provided an evolutionary game approach to optimize the cumulative value of all DTs. In [13], the data drift issue due to the system dynamics in edge environments was examined, by devising a continual learning-based scheme for continuous training of DT models to ensure that all DTs are as fresh as possible. The DT synchronization issue was considered in [21] by introducing the DT freshness concept, and then defining the freshness of inference models built on DTs, which facilitates providing high-fidelity query services. Efficient algorithms were proposed in [22] to enhance the freshness of inference models in MEC environments, which is impacted by the freshness of source data from DTs. The authors in [1] considered the joint optimization of service model retraining and inference service provisioning in edge computing environments, by leveraging the DTs of historical traces of past requests for different service models and the multi-armed bandit optimization technology. They developed an online learning algorithm for maximizing the cumulative fidelity of all admitted requests over a finite time horizon through exploring trade-offs on limited resource allocations to both model retraining and admitting requests simultaneously. Meanwhile, the authors in [32] focused on the semantic communication during the synchronization of DTs in a UAV-assisted MEC environment. They introduced a semantic extraction factor, and proposed a DRL-based method to achieve a good balance between synchronization delay and accuracy of DTs.

The authors in [23] introduced a new metric, Ultra-Low Age of Information (ULAoI), with the aim to capture the timeliness of the received DT information by extreme value theory. They proposed a learning-based approach to achieve DT-assisted resource allocation, under the ULAoI constraint on DTs and the power constraint. The problem of placing DTs was investigated in [35], aiming to reduce the response time of service requests, while optimizing the synchronization delay between DTs and their objects to meet the data age targets. A communication algorithm was developed in [40] to reduce the average peak Age of Information (AoI) within DT edge networks to keep DTs as fresh as possible, considering the communication failure. The authors in [42] devised a two-timescale approach to optimize the collected utility of service providers by caching fresh and popular contents, and leveraged a DT-based prediction strategy to predict the content popularity in the future. However, these studies do not include the joint optimization of freshness and cost of DT synchronization, nor determine where and when to conduct each DT synchronization task in the MEC environments.

Different from the aforementioned works, we study how to improve the freshness of DTs in MEC, considering static and dynamic cases of DT synchronization task arrivals. We tackle the static and dynamic DT freshness optimization problems via developing efficient approximation and online algorithms. It must be mentioned that this is the journal extension of a conference paper [17]. Compared with the conference version of the paper that a randomized algorithm was developed for the static DT freshness optimization problem with bounded resource

TABLE I
COMPARISON WITH RELATED WORKS

Feature Work	[3]	[9]	[15]	[17]	[23]	[25]	[32]	[35]	[40]	[42]	Ours
DT synchronization	×	✓	✓	✓	✓	✓	✓	✓	×	×	✓
Freshness optimization	×	×	✓	✓	✓	×	×	✓	✓	✓	✓
Cost modeling	✓	✓	×	✓	✓	×	✓	×	×	✓	✓
Approximation algorithm	×	×	✓	✓	×	×	×	✓	×	×	✓
Online algorithm	×	×	×	×	×	✓	✓	×	✓	✓	✓

violations, a constant approximation algorithm in this journal version is devised, which improves the previous result significantly. Furthermore, the dynamic DT freshness optimization problem that has not been dealt in the conference version has also been considered, and an online algorithm with a provable competitive ratio for it is developed.

For the sake of convenience, Table I highlights the work of this paper that is different from the aforementioned existing studies.

III. PRELIMINARIES

In this section, we present the system model, and specify the notations, cost model and utility model. We also outline the problem definitions.

A. System Model

Consider an MEC network $G = (N, E)$, where there are a set N of Access Points (APs) and a set E of optic links interconnecting pairs of APs, as illustrated by Fig. 1, where each AP is connected to a co-located cloudlet through an optic link with negligible communication delay [11], and there is a co-located cloudlet with each AP, and the AP and its co-located cloudlet are interchangeable if no confusion arises. Denote by C_n the amount of available computing resource in cloudlet $n \in N$.

Supposing the network has a set O of physical objects, each object $o_i \in O$ has a DT denoted by DT_i and placed in a cloudlet $loc(DT_i) \in N$. During a time horizon $\mathbb{T} = \{1, 2, \ldots, T\}$ with T equal time slots, an object o_i may generate and upload new data of size $a_{i,t}$ to cloudlet $loc(DT_i)$ for storage at a time slot t, while issuing a synchronization task to update its DT with newly generated data. There is a set Q_t of issued synchronization tasks from physical objects at time slot t.

B. Cost Model

At each time slot t, a physical object o_i may issue a synchronization task $q_{i,t} \in Q_t$ by uploading new data of size $a_{i,t}$ to its DT_i. As mentioned, the service provider may choose to offload such a synchronization task to another cloudlet n for execution. Therefore, the cost for executing a synchronization task consists of *the processing cost* and *the communication cost*, defined as follows.

The processing cost: Denote by $\kappa(i, t, n)$ the cost for cloudlet n to process a data unit for the synchronization of DT_i with its data generated at time t. Suppose a synchronization task $q_{i,t}$ with data size $a_{i,t}$ is processed in cloudlet n. Then the processing cost for the synchronization task $q_{i,t}$ is

$$c_{proc}(i, t, n) = \kappa(i, t, n) \cdot a_{i,t}. \tag{1}$$

The communication cost: Denote by $\xi(e)$ the cost for sending a data unit through network edge e, and $Path_{n,n'}$ is the shortest routing path between cloudlets n and n'. Recall that DT_i is hosted by cloudlet $loc(DT_i)$. Suppose a synchronization task $q_{i,t}$ is offloaded to cloudlet n, i.e., its new data with size $a_{i,t}$ is first sent to cloudlet n for processing, while the processed data with size $b_{i,t}$ is sent back to cloudlet $loc(DT_i)$ for aggregation with DT_i. Therefore, the communication cost for offloading the synchronization task $q_{i,t}$ is

$$c_{comm}(i, t, n) = \sum_{e \in Path_{loc(DT_i),n}} \xi(e) \cdot (a_{i,t} + b_{i,t}). \tag{2}$$

C. Utility of Each DT Synchronization

Considering a synchronization task $q_{i,t}$ that was issued at time slot t but processed at time slot $\tau \geq t$, we can see $(\tau - t)$ is the duration from the issuing time to the processing time of $q_{i,t}$, similar to the definition of the Age of Information (AoI) [41]. Intuitively, the freshness of a DT synchronization decreases with the increase on $(\tau - t)$, while users usually have expected DT freshness requirements [35]. We thus define the freshness of the DT synchronization as follows.

$$fresh_{i,t} = \begin{cases} \lambda_{i,t}^{\tau - t} & \text{if } \tau \leq \theta_{i,t} \\ \lambda_{i,t}^{\theta_{i,t} - t} & \text{otherwise,} \end{cases} \tag{3}$$

where $\lambda_{i,t}$ is constant with $0 < \lambda_{i,t} < 1$. The term $\lambda_{i,t}^{\tau - t}$ measures the freshness of DT synchronization by $q_{i,t}$, which decreases with the increase of the value of τ. To meet the DT freshness requirements of users, the constant $\theta_{i,t}$ $(\geq t)$ is the time threshold for processing DT synchronization task $q_{i,t}$, i.e., the freshness of DT synchronization of $q_{i,t}$ drops to the minimum value $\lambda_{i,t}^{\theta_{i,t} - t}$ after time slot $\theta_{i,t}$.

The utility of synchronization task $q_{i,t}$ then is defined as

$$u(q_{i,t}, n, \tau) = w \cdot fresh_{i,t} - c_{proc}(i, t, n) - c_{comm}(i, t, n)), \tag{4}$$

where $c_{proc}(i, t, n)$ and $c_{comm}(i, t, n)$ are the processing and communication costs by (1) and (2), respectively, and coefficient w is a constant related to the DT freshness, which ensures the utility to be non-negative, and its value can be set by analyzing the historical traces.

D. Computing Resource Capacity Constraints of Cloudlets

We assume the consumed computing resource of a DT synchronization is proportional to the size of the uploaded data of its object [13], [37], [38], [46], and the computing resource consumption of DT_i for processing the synchronization task $q_{i,t}$ with the uploaded update data of size $a_{i,t}$ at time slot t is $\sigma_i \cdot a_{i,t}$, where σ_i is a constant.

Recalling that each cloudlet n is associated with a computing resource capacity C_n, it can be seen that the resource consumed by each cloudlet must not exceed its capacity in any time slot.

E. Problem Definitions

In the following, we introduce the definitions of static and dynamic DT freshness optimization problems, and then demonstrate NP-hardness of the two defined problems.

In practice, objects may periodically send their update data to DTs in fixed intervals for synchronizations [12], [35]. In this way, a static case assumes that each DT synchronization task is given in advance with its issuing time and data size, which can be analyzed and predicted using their historical traces. Thus, the *static DT freshness optimization problem* is defined as follows.

Definition 1: Consider an MEC network $G = (N, E)$, a set O of physical objects with DTs placed in cloudlets, and a set Q_t of synchronization tasks issued in each time slot t within a time horizon $\mathbb{T}$ given in advance. The *static DT freshness optimization problem* is to maximize the total utility gain for improving DT freshness while minimizing the total cost of various resource consumption, by determining the processing cloudlet and the processing time slot of each synchronization task, subject to the computing capacities on cloudlets in N.

Let $x_{i,t,n,\tau}$ denote a binary decision variable, where $x_{i,t,n,\tau} = 1$ implies that synchronization task $q_{i,t}$ of DT_i is offloaded to the cloudlet n for processing at time slot $\tau \in \mathbb{T}$; otherwise $x_{i,t,n,\tau} = 0$.

We here present the Integer Linear Program (ILP) formulation for the static DT freshness optimization problem as follows.

$$\text{maximize} \sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} u(q_{i,t}, n, \tau) \cdot x_{i,t,n,\tau}, \tag{5}$$

subject to:

$$\sum_{\tau=t}^{T} \sum_{n \in N} x_{i,t,n,\tau} \leq 1, \qquad \forall t \in [1, T],\ \forall q_{i,t} \in Q_t \tag{6}$$

$$x_{i,t,n,\tau} = 0, \quad \forall t \in [1, T],\ \forall q_{i,t} \in Q_t,\ \forall \tau \in [1, t-1],\ \forall n \in N \tag{7}$$

$$\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sigma_i \cdot a_{i,t} \cdot x_{i,t,n,\tau} \leq C_n,\ \forall n \in N,\ \forall \tau \in [1, T] \tag{8}$$

$$x_{i,t,n,\tau} \in \{0, 1\}, \quad \forall t \in [1, T],\ \forall q_{i,t} \in Q_t,\ \forall \tau \in [1, T],\ \forall n \in N, \tag{9}$$

where the objective (5) provides the accumulative collected utility from all synchronization tasks defined by (4). Constraint (6) ensures that each synchronization task is processed in at most one cloudlet; according to Constraint (7), each synchronization task of DTs is processed no earlier than its issuing time; and Constraint (8) further ensures that the computing resource consumed at a cloudlet does not exceed its capacity.

Given the uncertain dynamics in IoT environments, the DT synchronization tasks are often issued dynamically. Hence, we further consider a dynamic case, where DTs issue the synchronization tasks in each time slot dynamically, and there is no provided information about future synchronization task arrivals. We define the dynamic DT freshness optimization problem as follows.

Definition 2: Consider an MEC network $G = (N, E)$, a set O of physical objects with DTs placed in cloudlets, a time horizon $\mathbb{T}$, and a set Q_t of synchronization tasks issued in each time slot t without any given future information. The *dynamic DT freshness optimization problem* is to maximize the accumulative utility gain for improving DT freshness while minimizing the incurred cost over the given time horizon, by determining the processing cloudlet and the processing time slot of each synchronization task, subject to the computing capacities on cloudlets in N.

All symbols adopted are listed in Table II.

F. NP-Hardness of the Defined Problems

Theorem 1: The static DT freshness optimization problem for improving DT freshness in an MEC network $G = (N, E)$ is NP-hard.

Proof: The NP-hardness of the static DT freshness optimization problem is proven via a reduction from a known NP-hard problem – the Generalized Assignment Problem (GAP) [28]. In GAP, there are a set of bins and a set of items, where placing an item $item_i$ into a bin bin_j leads to a profit $profit_{i,j}$. Each item has a weight, while each bin has a capacity. The objective of the GAP is to collect as much profit as possible, respecting capacities on bins.

We now consider a special case of the static DT freshness optimization problem, by assuming the time horizon has only one time slot. Namely, synchronization tasks are issued by DTs at time slot 1 and can only be processed at time slot 1. Each cloudlet n is treated as a bin with a capacity C_n, and each synchronization task $q_{i,1}$ of DTs is treated as an item with the weight $\sigma_i \cdot a_{i,1}$ (its consumed computing resource). If $q_{i,1}$ is executed in a cloudlet $n \in N$, then a profit $u(i, 1, n, 1)$ is collected according to (4). The special case of the static DT freshness optimization problem aims to collect as much profit as possible via an efficient assignment of synchronization tasks to cloudlets to improve DT freshness.

It is observed that the above-mentioned special case of the static DT freshness optimization problem is equivalent to GAP [28], implying that the static DT freshness optimization problem is also NP-hard. ■

Corollary 1: The dynamic DT freshness optimization problem for improving DT freshness in an MEC network $G = (N, E)$ is NP-hard.

Proof: The static DT freshness optimization problem is considered as a special case of the dynamic DT freshness optimization problem, through assuming all synchronization tasks are given in advance. Hence, we can achieve the corollary that the dynamic DT freshness optimization problem is also NP-hard. ■

IV. Approximation Algorithm for the Static DT Freshness Optimization Problem

The ILP (5) presents an optimal solution for the static DT freshness optimization problem, which, however, usually leads to unacceptable running time for solving the ILP. Therefore, we further devise a constant approximation algorithm for the static DT freshness optimization problem in this section, which

TABLE II
TABLE OF SYMBOLS

Notations	Descriptions
$G = (N, E)$	The MEC network G with a set N of APs (co-located with cloudlets) and a set E of links connecting pairs of APs.
C_n	The amount of available computing resource in cloudlet $n \in N$.
O and o_i	The set of all physical objects within the MEC network and a physical object.
DT_i and $loc(DT_i)$	The DT of object o_i and the cloudlet hosting DT_i.
$\mathbb{T}$	The time horizon with T equal time slots.
Q_t	The set of issued synchronization tasks from physical objects at time slot t.
$q_{i,t}$	The issued synchronization task by object o_i at time slot t.
$\kappa(i,t,n)$	The cost for cloudlet n to process a unit of data for the synchronization of DT_i with its data generated at time t.
$a_{i,t}$	The update data size associated with synchronization task $q_{i,t}$.
$b_{i,t}$	The size of the processed data associated with synchronization task $q_{i,t}$.
$\xi(e)$	The cost for transmitting a unit of data through network edge e.
$Path_{n,n'}$	The shortest routing path between cloudlets n and n'.
$c_{proc}(i,t,n)$	The processing cost of synchronization task $q_{i,t}$ in cloudlet n.
$c_{comm}(i,t,n)$	The communication cost caused by offloading synchronization task $q_{i,t}$ to cloudlet n.
$\lambda_{i,t}$ and $\theta_{i,t}$	A coefficient with $0 < \lambda_{i,t} < 1$, and the time threshold for the processing of synchronization task $q_{i,t}$.
$fresh_{i,t}$	The freshness of synchronization task $q_{i,t}$ issued at time slot t and processed at time slot $\tau \geq t$, by Eq. (3).
w	The constant weight associated with the freshness.
$u(q_{i,t}, n, \tau)$	The utility of synchronization task $q_{i,t}$ issued at time slot t and processed at time slot $\tau \geq t$, by Eq. (4).
σ_i	The consumed computing resource for processing a unit data of the synchronization tasks from object o_i.
$x_{i,t,n,\tau}$	The decision variable indicating whether $q_{i,t}$ is offloaded to the cloudlet n for processing at time slot τ.
$\mathcal{Q}(\tau)$	The set of synchronization tasks, which are taken into consideration for processing at current time slot τ.
$\mathbb{A}(\tau)$	The potential solution $\mathbb{A}(\tau)$ consisting of the synchronization tasks assigned to cloudlets at time slot t.
$A_1(\tau)$ and $A_2(\tau)$	Two disjoint subsets derived by partitioning $\mathbb{A}(\tau)$.
$\zeta(q_{i,t}, n, \tau)$	The ratio $\zeta(q_{i,t}, n, \tau)$ of assigning synchronization task $q_{i,t}$ to cloudlet n at time slot $\tau \geq t$, by Eq. (10).
q^l	The lth assigned synchronization task.
$\mathbb{A}^{l-1}(\tau)$	The first $l-1$ assigned synchronization tasks before the assignment of the lth synchronization task at time slot τ.
$\mathcal{C}_{n,\tau}(\mathbb{A}^{l-1}(\tau))$	The total computing resource utilized on cloudlet n at time slot τ after assigning synchronization tasks in $\mathbb{A}^l(\tau)$.
$\mathbb{A}(\tau)^{opt}$	The set of assigned synchronization tasks at time slot τ in optimal solution to the dynamic DT freshness optimization problem.
ζ_{min} and ζ_{max}	The minimum and the maximum values of ratio $\zeta(q_{i,t}, n, \tau)$ by Eq. (10), respectively.

achieves a guaranteed performance with a constant approximation ratio and runs in polynomial time. We also analyze the theoretical performance of the proposed approximation algorithm.

A. Approximation Algorithm

In the following, we detail the proposed approximation algorithm. The core idea behind the proposed algorithm is to reduce the static DT freshness optimization problem to a Generalized Assignment Problem (GAP). Then, we approach the latter by applying the approximation algorithm in [6].

We first treat each cloudlet $n \in N$ in each time slot $\tau \in \mathbb{T}$ as a bin $bin_{n,\tau}$ associated with a capacity C_n, which is the computing capacity of the cloudlet n. We then treat each synchronization task $q_{i,t}$ as an item $item_{i,t}$ associated with a weight $\sigma_i \cdot a_{i,t}$, which is its computing resource consumption. The assignment of an item $item_{i,t}$ to a bin $bin_{n,\tau}$ with $\tau \geq t$ leads to a profit $u(q_{i,t}, n, \tau)$, which is the obtained utility calculated by (4). Notice that when $\tau < t$, the obtained profit of such assignment is 0, because a synchronization task can only be processed no earlier than its issuing time. It can be observed that an approximate solution for the GAP can be derived from the approach in [6], which provides an approximate solution for the static DT freshness optimization problem.

The proposed approximation algorithm is illustrated in Algorithm 1.

B. Algorithm Analysis

We now conduct a theoretical analysis on Algorithm 1 for the static DT freshness optimization problem as follows.

Theorem 2: Given an MEC network $G = (N, E)$, a set O of physical objects with DTs placed in cloudlets, and a set Q_t of synchronization tasks issued in each time slot t given in

Algorithm 1: Approximation Algorithm for the Static DT Freshness Optimization Problem.

Input: An MEC network $G = (N, E)$, a set O of physical objects with their DTs placed in cloudlets, and a set Q_t of synchronization tasks at each time slot t for a given time horizon $\mathbb{T}$.

Output: Improve the DT freshness.

1: Reduce the problem instance to a GAP instance, via generating a bin $bin_{n,\tau}$ associated with the capacity C_n for each cloudlet $n \in N$ in each time slot $\tau \in \mathbb{T}$, and an item $item_{i,t}$ associated with a weight $\sigma_i \cdot a_{i,t}$ for each synchronization task $q_{i,t}$.
2: **for** each $item_{i,t}$ **do**
3: **for** each $bin_{n,\tau}$ **do**
4: **if** $\tau \geq t$ **then**
5: The profit of putting $item_{i,t}$ into $bin_{n,\tau}$ is $u(q_{i,t}, n, \tau)$ by (4);
6: **else**
7: The profit of putting $item_{i,t}$ into $bin_{n,\tau}$ is 0;
8: **end if**
9: **end for**
10: **end for**
11: Apply the approximation algorithm from [6] to solve the GAP;
12: **return**The approximate solution to the static DT freshness optimization problem.

advance, there is an approximation algorithm, Algorithm 1, for the static DT freshness optimization problem, which delivers an approximate solution with a $\frac{1}{2+\epsilon}$ approximation ratio. The time complexity is $O(|N| \cdot \sum_{t \in \mathbb{T}} |Q_t| \cdot \log \frac{1}{\epsilon} + \frac{|N|}{\epsilon^4})$, where ϵ is a constant with $0 < \epsilon < 1$.

Proof: The approximation ratio of Algorithm 1 can be derived from the detailed analysis of the approximation algorithm in [6], i.e., the solution by Algorithm 1 is no less than $\frac{1}{2+\epsilon}$ times the optimal solution value, and ϵ is a constant with $0 < \epsilon < 1$.

The time complexity analysis of Algorithm 1 is as follows. The main time complexity of Algorithm 1 is to invoke the approximation algorithm in [6]. Therefore, its time complexity primarily is determined by the time complexity of the invoking approximation algorithm, which is $O(|N| \cdot \sum_{t \in \mathbb{T}} |Q_t| \cdot \log \frac{1}{\epsilon} + \frac{|N|}{\epsilon^4})$ time.

V. Online Algorithm for the Dynamic DT Freshness Optimization Problem

Having studied the static DT freshness optimization problem, in this section we study the dynamic DT freshness optimization problem by taking into account the uncertainty and dynamics of DT synchronization tasks over a finite time horizon. Under this dynamic setting, a set of synchronization tasks from DTs are issued in the beginning of each time slot, and there is no future information of synchronization task arrivals.

We here design an online algorithm for the dynamic DT freshness optimization problem, and its core idea behind is demonstrated as follows. In each time slot τ, we first propose a potential solution $\mathbb{A}(\tau)$ consisting of the synchronization tasks assigned to cloudlets to maximize the total utility collected at the time slot, which however is likely to cause violations of computing capacities on cloudlets. We then partition the solution set $\mathbb{A}(\tau)$ into two subsets $A_1(\tau)$ and $A_2(\tau)$, ensuring that neither of them has any violation on the computing capacities of cloudlets. We finally choose the one between $A_1(\tau)$ and $A_2(\tau)$ with the greater utility gain as the final solution at time slot t for the dynamic DT freshness optimization problem.

Denote by $\mathcal{Q}(\tau)$ the set of synchronization tasks, which are taken into consideration for processing at the current time slot τ, i.e., $\mathcal{Q}(\tau)$ consists of the synchronization tasks issued at the current time slot τ, and those issued before time slot τ but have not been processed.

For each synchronization task $q_{i,t} \in \mathcal{Q}(\tau)$, it can be seen that its computing resource consumption is $\sigma_i \cdot a_{i,t}$, as shown in Section III-D. Its utility gain $u(q_{i,t}, n, \tau)$ is defined in (4) when the synchronization is processed in cloudlet n at time slot $\tau \geq t$.

To guide assigning synchronization tasks, we define the ratio $\zeta(q_{i,t}, n, \tau)$ of assigning synchronization task $q_{i,t}$ to cloudlet n at time slot $\tau \geq t$ as follows.

$$\zeta(q_{i,t}, n, \tau) = \frac{u(q_{i,t}, n, \tau)}{\sigma_i \cdot a_{i,t}} \tag{10}$$

The online algorithm proceeds in each time slot $\tau \in \mathbb{T}$ by assigning synchronization tasks iteratively as follows. Denote by q^l the lth assigned synchronization task. Let $\mathbb{A}^{l-1}(\tau)$ be the set of the first $l-1$ synchronization tasks assigned before the assignment of lth synchronization task at time slot τ, where $\mathbb{A}^l(\tau) = \mathbb{A}^{l-1}(\tau) \cup \{q^l\}$, and the set $\mathbb{A}^0(\tau)$ of assigned synchronization tasks is empty initially. We also adopt $\mathcal{C}_{n,\tau}(\mathbb{A}^l(\tau))$ to denote the total computing resource consumption of cloudlet n at time slot τ after assigning the synchronization tasks in $\mathbb{A}^l(\tau)$.

In each iteration l, we identify a synchronization task $q^l \in \mathcal{Q}(\tau) \backslash \mathbb{A}^{l-1}(\tau)$ with the assignment to cloudlet n at time slot τ (no earlier than the issuing time of q^l), such that its ratio $\zeta(q^l, n, \tau)$ by (10) is maximized, and the computing resource utilized of cloudlet n is smaller than its capacity, i.e., $\mathcal{C}_{n,\tau}(\mathbb{A}^{l-1}(\tau)) < C_n$.

If the assignment of synchronization task q^l causes the capacity violation of cloudlet n at time slot τ, i.e., $\mathcal{C}_{n,\tau}(\mathbb{A}^l(\tau)) > C_n$, we choose to put q^l into set $A_1(\tau)$; otherwise (the capacity of cloudlet n at time slot τ is not violated with $\mathcal{C}_{n,\tau}(\mathbb{A}^l(\tau)) \leq C_n$), we put q^l into set $A_2(\tau)$. Notice that if $\mathcal{C}_{n,\tau}(\mathbb{A}^l(\tau)) \geq C_n$, we no longer consider assigning any more synchronization task to cloudlet n at time slot τ.

It can be seen $A_1(\tau)$ and $A_2(\tau)$ are disjoint with $\mathbb{A}(\tau) = A_1(\tau) \cup A_2(\tau)$. We claim that the assignment of synchronization tasks in either $A_1(\tau)$ and $A_2(\tau)$ will not result in any resource violation, which will be demonstrated later in Lemma 2.

The proposed online algorithm finally selects the set with the greater utility gain between $A_1(\tau)$ and $A_2(\tau)$ as the solution at time slot τ for the dynamic DT freshness optimization problem. The proposed online algorithm is shown in Algorithm 2.

Algorithm 2: Online Algorithm for the Dynamic DT Freshness Optimization Problem.

Input: An MEC network $G = (N, E)$, a set O of physical objects with their DTs placed in cloudlets, and a set Q_t of synchronization tasks at each time slot t for a given time horizon $\mathbb{T}$.

Output: Improve the DT freshness in an online manner.

for each time slot $\tau \in \mathbb{T}$ **do**
  $\mathbb{A} \leftarrow \emptyset$; /* the solution to the problem */
  $\mathbb{A}^0(\tau) \leftarrow \emptyset$; $A_1(\tau) \leftarrow \emptyset$; $A_2(\tau) \leftarrow \emptyset$;
  $\mathcal{C}_{n,\tau}(\mathbb{A}^0(\tau)) \leftarrow 0, \forall n \in N$;
  $l \leftarrow 1$;
  while $\mathcal{Q}(\tau) \backslash \mathbb{A}^{l-1}(\tau) \neq \emptyset$ and it exists a cloudlet n at a time slot τ with $\mathcal{C}_{n,\tau}(\mathcal{A}^{l-1}) < C_n$ **do**
    Identify a synchronization task $q^l \in \mathcal{Q}(\tau) \backslash \mathbb{A}^{l-1}(\tau)$ with the assignment to cloudlet n at time slot τ, such that its ratio $\zeta(q^l, n, \tau)$ by (10) is maximized, with $\mathcal{C}_{n,\tau}(\mathbb{A}^{l-1}(\tau)) < C_n$;
    $\mathbb{A}^l(\tau) \leftarrow \mathbb{A}^{l-1}(\tau) \cup \{q^l\}$;
    Update $\mathcal{C}_{n,\tau}(\mathbb{A}^l(\tau))$;
    if $\mathcal{C}_{n,\tau}(\mathbb{A}(\tau)^l) > C_n$ **then**
      $A_2(\tau) \leftarrow A_2(\tau) \cup \{q^l\}$;
    else
      $A_1(\tau) \leftarrow A_1(\tau) \cup \{q^l\}$;
    end if ;
    $l \leftarrow l + 1$;
  end while ;
  if the total utility gain by $A_1(\tau)$ is greater than that by $A_2(\tau)$ **then**
    $\mathbb{A}(\tau) \leftarrow A_1(\tau)$;
  else
    $\mathbb{A}(\tau) \leftarrow A_2(\tau)$;
  end if ;
  $\mathbb{A} \leftarrow \mathbb{A} \cup \mathbb{A}(\tau)$;
end for ;
return $\mathbb{A}$;

A. Algorithm Analysis

In the following, we analyze the competitive ratio and time complexity of Algorithm 2 for the dynamic DT freshness optimization problem.

For simplicity, we adopt the notations $u(q^l)$, $c(q^l)$, and $\zeta(q^l)$ to denote the utility gain, the computing resource consumption, and the ratio by (10) of each assigned synchronization task q^l at each time slot τ by Algorithm 2, respectively.

Lemma 1: Given a solution $\mathbb{A}(\tau) = A_1(\tau) \cup A_2(\tau)$ at time slot τ delivered by Algorithm 2, denote by $\mathcal{A}_{n,\tau}$ the set of synchronization tasks allocated to cloudlet n at time slot τ based on $\mathbb{A}(\tau)$, i.e., $\mathbb{A}(\tau) = \cup_{n \in N} \mathcal{A}_{n,\tau}$. Let $\mathbb{A}(\tau)^{opt}$ be the set of assigned synchronization tasks at time slot τ in optimal solution to the dynamic DT freshness optimization problem. Similarly, let $\mathcal{A}^{opt}_{n,\tau}$ be the set of synchronization tasks allocated to cloudlet n at time slot τ based on $\mathbb{A}(\tau)^{opt}$ with $\mathbb{A}(\tau)^{opt} = \cup_{n \in N} \mathcal{A}^{opt}_{n,\tau}$. Denote by $\zeta_{\min}$ and $\zeta_{\max}$ the minimum and the maximum values of the ratio ζ by (10), respectively. We have

(i) $\zeta(q^l) \geq \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \zeta(q^*), \forall q^l \in \mathcal{A}_{n,\tau}, \quad \forall q^* \in \mathcal{A}^{opt}_{n,\tau} \setminus \mathcal{A}_{n,\tau}, \forall \tau \in \mathbb{T}, \forall n \in N$; and

(ii) $\sum_{q^l \in \mathbb{A}(\tau)} u(q^l) \geq \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{q^* \in \mathbb{A}(\tau)^{opt} \setminus \mathbb{A}(\tau)} u(q^*), \forall \tau \in \mathbb{T}$.

Proof: We show the claims as follows.

Claim (i) follows the definition of $\zeta_{\min}$ and $\zeta_{\max}$ directly, and omitted.

To prove Claim (ii), let $q^*_{n,\tau,max} = \arg\max_{q^* \in \mathcal{A}^{opt}_{n,\tau} \setminus \mathcal{A}_{n,\tau}} \zeta(q^*), \forall \tau \in \mathbb{T}, \forall n \in N$, then

$$\sum_{q^l \in \mathbb{A}(\tau)} u(q^l) = \sum_{n \in N} \sum_{q^l \in \mathcal{A}_{n,\tau}} u(q^l)$$

$$= \sum_{n \in N} \sum_{q^l \in \mathcal{A}_{n,\tau}} \zeta(q^l) \cdot c(q^l) \tag{11}$$

$$\geq \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{n \in N} \sum_{q^l \in \mathcal{A}_{n,\tau}} \zeta(q^*_{n,\tau,max}) \cdot c(q^l) \tag{12}$$

$$= \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{n \in N} \zeta(q^*_{n,\tau,max}) \cdot \sum_{q^l \in \mathcal{A}_{n,\tau}} c(q^l)$$

$$\geq \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{n \in N} \zeta(q^*_{n,\tau,max}) \cdot \sum_{q^* \in \mathcal{A}^{opt}_{n,\tau} \setminus \mathcal{A}_{n,\tau}} c(q^*) \tag{13}$$

$$\geq \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{n \in N} \sum_{q^* \in \mathcal{A}^{opt}_{n,\tau} \setminus \mathcal{A}_{n,\tau}} \zeta(q^*) \cdot c(q^*)$$

$$= \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{n \in N} \sum_{q^* \in \mathcal{A}^{opt}_{n,\tau} \setminus \mathcal{A}_{n,\tau}} u(q^*)$$

$$= \frac{\zeta_{\min}}{\zeta_{\max}} \cdot \sum_{q^* \in \mathbb{A}(\tau)^{opt} \setminus \mathbb{A}(\tau)} u(q^*), \tag{14}$$

where (11) holds by the definition of $\zeta(q^l)$, i.e., (10), with $c(q^l)$ the computing resource consumption of the assigned synchronization task q^l. In (12) holds by (i) and the definition of $q^*_{n,\tau,\max}$. It can be seen Ineq. (13) holds since the computing resource consumed of each cloudlet n based on the solution $\mathbb{A}(\tau)$ by Algorithm 2 is no less than its capacity C_n, while no cloudlet experiences capacity violations by the optimal solution. Lastly, Ineq. (14) is valid due to the definition of $q^*_{n,\tau,\max}$. ■

Lemma 2: Given a solution $\mathbb{A}(\tau) = A_1(\tau) \cup A_2(\tau)$ at time slot τ delivered by Algorithm 2, the assignment of synchronization tasks in $A_1(\tau)$ or $A_2(\tau)$ does not lead to any resource violations on cloudlets.

Proof: By Algorithm 2, a synchronization task is added to $A_1(\tau)$ if no resource violation will occur.

When a synchronization task is added to $A_2(\tau)$, this assignment leads to the capacity violation on a cloudlet. Because the cloudlet with the violated capacity is excluded from the assignment of the rest synchronization tasks, each cloudlet is assigned with at most one synchronization task in $A_2(\tau)$. Supposing that each cloudlet can hold the processing of any single synchronization task, it can be seen that the assignment

of synchronization tasks $A_2(\tau)$ will not cause any capacity violation on cloudlets. Hence, we conclude that the assignment of synchronization tasks in $A_1(\tau)$ or $A_2(\tau)$ will not lead to any resource violation on cloudlets. ■

Theorem 3: Given an MEC network $G = (N, E)$, a set O of physical objects with DTs placed in cloudlets, and a set Q_t of synchronization tasks issued in each time slot t without any future information, there is an online algorithm, Algorithm 2 with a competitive ratio of $\frac{1}{2} \cdot \frac{\zeta_{\min}}{\zeta_{\max}+\zeta_{\min}}$, for the dynamic DT freshness optimization problem in G, which takes $O((\sum_{t\in\mathbb{T}} |Q_t|)^2 \cdot |N| \cdot T)$ time.

Proof: $\mathbb{A}(\tau)^{opt}$ is the set of assigned synchronization tasks at time slot τ in the optimal solution, and its utility gain is $\sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*)$. $\mathbb{A}(\tau) = A_1(\tau) \cup A_2(\tau)$ is the solution at time slot τ delivered by Algorithm 2. Then

$$\begin{aligned}
&\sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*) \\
&\leq \sum_{\lambda^l\in\mathbb{A}(\tau)} u(q^l) + \sum_{q^*\in\mathbb{A}(\tau)^{opt}\setminus\mathbb{A}(\tau)} u(q^*) \\
&\leq \left(1 + \frac{\zeta_{\max}}{\zeta_{\min}}\right) \cdot \sum_{\lambda^l\in\mathbb{A}(\tau)} u(\lambda^l), \text{ by (ii) in Lemma 1} \qquad (15)
\end{aligned}$$

Furthermore, we have

$$\begin{aligned}
&\sum_{\lambda^l\in A_1(\tau)} u(q^l) + \sum_{\lambda^l\in A_2(\tau)} u(q^l) \\
&= \sum_{\lambda^l\in\mathbb{A}(\tau)} u(q^l) \\
&\geq \frac{\zeta_{\min}}{\zeta_{\max}+\zeta_{\min}} \sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*), \text{ by Ineq. (15)} \qquad (16)
\end{aligned}$$

We select the set with a larger utility gain between $A_1(\tau)$ and $A_2(\tau)$ as the final solution at time slot τ, and the final solution value by Algorithm 2 is

$$\begin{aligned}
&\sum_{\tau\in\mathbb{T}} \max\left\{\sum_{\lambda^l\in A_1(\tau)} u(q^l), \sum_{\lambda^l\in A_2(\tau)} u(q^l)\right\} \\
&\geq \sum_{\tau\in\mathbb{T}} \frac{1}{2} \cdot \frac{\zeta_{\min}}{\zeta_{\max}+\zeta_{\min}} \sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*) \\
&= \frac{1}{2} \cdot \frac{\zeta_{\min}}{\zeta_{\max}+\zeta_{\min}} \sum_{\tau\in\mathbb{T}} \sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*), \qquad (17)
\end{aligned}$$

where $\sum_{\tau\in\mathbb{T}} \sum_{q^*\in\mathbb{A}(\tau)^{opt}} u(q^*)$ represents the utility by the optimal solution. It can be seen that assigning synchronization tasks in either $A_1(\tau)$ or $A_2(\tau)$ does not lead to any computing resource violations for the cloudlets, by Lemma (2).

The time complexity of Algorithm 2 is analyzed as follows. There are $\sum_{t\in\mathbb{T}} |Q_t|$ synchronization tasks over time horizon $\mathbb{T}$, which means that there are at most $\sum_{t\in\mathbb{T}} |Q_t|$ iterations to examine the assignment of synchronization tasks at each time slot τ. During each iteration, we need to find a synchronization task q^l with the largest $\zeta(q^l)$ defined by (10), considering its possible assignment with all $|N|$ cloudlets in the MEC network, which takes the time $O((\sum_{t\in\mathbb{T}} |Q_t|) \cdot |N|)$. Because there are T time slots within the time horizon $\mathbb{T}$, the running time of Algorithm 2 is $O((\sum_{t\in\mathbb{T}} |Q_t|)^2 \cdot |N| \cdot T))$. ■

VI. Performance Evaluation

In this section, we evaluate the performance of our algorithms, Algorithm 1 and Algorithm 2, for improving the freshness of DTs in an MEC network via DT synchronizations with their objects. We also investigate the impacts of important parameters on algorithm performance.

A. Experimental Settings

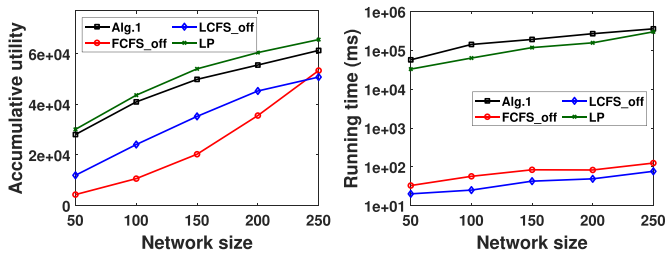
Consider an MEC network with the number of APs (and their co-located cloudlets) ranging from 50 to 250. Each network topology is generated by the GT-ITM tool [8]. There exist 1,000 physical objects (IoT devices) with their DTs placed in cloudlets. The computing capacity on a cloudlet is drawn between 3,000 MHz to 6,000 MHz [27]. The monitoring period consists of 10 time slots. The number of synchronization tasks at each time slot is randomly drawn from 600 and 1,000. Each synchronization task is issued by a randomly chosen object, and the size of the update data is drawn from $[10, 50]$ MB [35]. The amount of computing resource consumed for processing 1MB data is drawn from 30 MHz to 60 MHz, and the size of the processed data is within $[2, 10]$MB. The cost of data processing in a cloudlet is set within $[0.01, 0.1]$\$per MB, and the cost of data transferring along a link is set within $[0.01, 0.1]$\$per MB [27]. We set the parameter ϵ in Theorem 3 as 0.5. With regard to the utility definition (4), we set the value of parameter $\lambda_{i,t}$ as 0.5, the value of parameter $\theta_{i,t}$ within [2,4], and the value of weight w as 10.

The values presented in each figure are the mean of results via running the algorithms using 30 different network instances with the identical size. The algorithm running time is obtained by a desktop with an Octa-Core Intel(R) Xeon(R) CPU @ 2.20 GHz, 32G RAM. These parameters are used by default unless stated otherwise.

B. Benchmarks

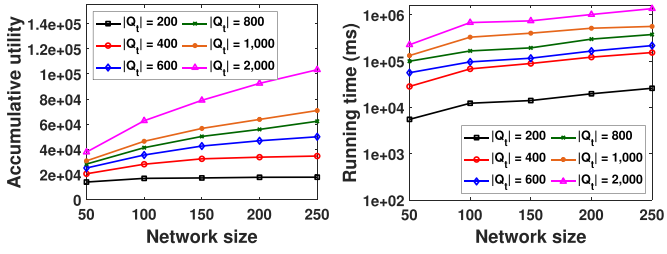
We evaluated the performance of Algorithm 1 (refered to as Algorithm 1) for the static DT freshness optimization problem against the following baselines.

- FCFS_off: It follows the spirit of [24] by adopting a first-come-first-served method to assign synchronization tasks from DTs one by one to cloudlets, and it assigns each synchronization task $q_{i,t}$ to the cloudlet n in a time slot τ ($\geq t$) with sufficient computing resource, such as its utility defined by (4) is maximized.
- LCFS_off: Similar to FCFS_off, it instead follows the spirit of [39] by adopting a last-come-first-served method to assign synchronization tasks from DTs one by one to cloudlets, and maximizes the utility of each assigned synchronization task.



(a) Accumulative utility (b) Running time

Fig. 2. Algorithm performance for the static DT freshness optimization problem.



(a) Accumulative utility (b) Running time

Fig. 3. Impact of the number $|Q_t|$ of synchronization tasks in each time slot on Algorithm 1.

- LP: It is a solution by Linear Program (5) with the relaxed variables in real numbers. It provides an upper bound on the optimal solution for the static DT freshness optimization problems.

We evaluated the performance of Algorithm 2 (refered to as Algorithm 2) for the dynamic DT freshness optimization problem against the following benchmarks.

- FCFS_on: It is the online version of FCFS_off, i.e., at each time slot, it considers the synchronization tasks from DTs that have not been processed one by one, and it adopts a first-come-first-served method to iteratively assign the synchronization task to the cloudlet with the largest utility.
- LCFS_on: It is the online version of LCFS_off. It adopts a last-come-first-served method to assign yet-to-be-processed synchronization tasks at the current time slot one by one to cloudlets, and maximizes the utility of each assigned synchronization task.
- LP: It can be seen LP also serves as an upper bound on the optimal solution for the dynamic DT freshness optimization problems.

C. Algorithmic Performance for the Static DT Freshness Optimization Problem

We first investigated the performance of Algorithm 1 for the static DT freshness optimization problem against benchmarks FCFS_off, LCFS_off and LP, with the varying number of cloudlets (APs) from 50 to 250. Fig. 2 depicts their algorithm performance. Fig. 2(a) demonstrates that the collected utility of Algorithm 1 is 93.5% of that by LP with 250 cloudlets, achieving a good performance. Also, Algorithm 1 outperforms FCFS_off and LCFS_off by 14.9% and 20.7% in terms of the collected utility, respectively, for the network size of 250. This is because Algorithm 1 reduces the instance of the static DT freshness optimization problem to a GAP instance, and leverages the approximate solution from [6] to deliver an efficient solution for the original problem. The algorithm running time is plotted in Fig. 2(b), where the running time of Algorithm 1 is longer than that of FCFS_off, LCFS_off, due to adopting the approximate solution in [6].

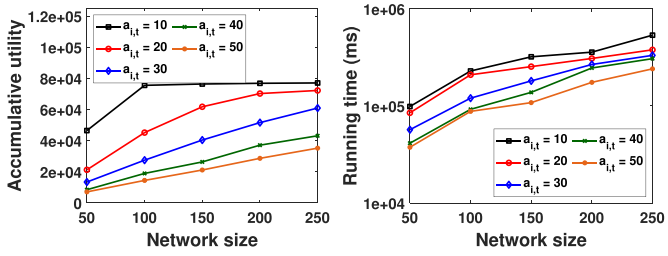
We then studied the impact of the number $|Q_t|$ of synchronization tasks issued at each time slot on the performance of Algorithm 1 when $|Q_t| = 200$, 400, 600, 800, 1,000 and 2,000, respectively. We observe from Fig. 3(a) that the utility delivered by Algorithm 1 when $|Q_t| = 200$ is 17.2% of that by itself when $|Q_t| = 2,000$, if 250 cloudlets are deployed in the MEC network. This justifies that Algorithm 1 can deliver a solution with a larger utility gain when more synchronization tasks are issued over time horizon $\mathbb{T}$. Fig. 3(b) illustrates the time complexity of Algorithm 1. When $|Q_t| = 2,000$, its running time is the longest one due to the time complexity of the approximation algorithm in [6]. Fig. 3 also demonstrates that Algorithm 1 exhibits nice scalability when the problem size is large, e.g., the running time of Algorithm 1 is moderate with 250 cloudlets and 2,000 synchronization tasks issued at each time slot.

We finally investigated the impact of the size $a_{i,t}$ (in MB) of each data update from an object on the performance of Algorithm 1 when $a_{i,t} = 10$, 20, 30, 40 and 50, respectively. It can be observed by Fig. 4(a) that the utility of Algorithm 1 with $a_{i,t} = 50$ is 45.6% of that by itself with $a_{i,t} = 10$, when the number of cloudlets is 250. The rationale behind is that a small amount of update data from objects means the DT synchronization consumes less computing resource, and more synchronization tasks can be processed to obtain more utility gain. Fig. 4(b) illustrates that Algorithm 1 with $a_{i,t} = 10$ takes the longest running time, due to dealing with the processing of the most synchronization tasks.

D. Algorithm Performance for the Dynamic DT Freshness Optimization Problem

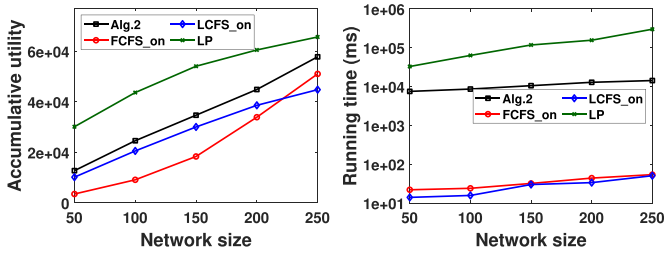
We first evaluated the performance of Algorithm 2 for the dynamic DT freshness optimization problem against benchmarks FCFS_on, LCFS_on and LP, by varying the number of cloudlets. Fig. 5(a) illustrates the utility gain of Algorithm 2, which is 88.1% of that by LP with 250 cloudlets. We can see Algorithm 2 delivers a solution with high utility gain even in a dynamic case with no knowledge of future synchronization tasks, which is 13.2% and 28.9% higher than that collected by FCFS_on and LCFS_on, respectively. This is due to the fact that Algorithm 2 carefully selects each assignment of synchronization task to maximize the collected utility in an online manner, through jointly considering its utility gain and its resource consumption.

We then considered the impact of the number $|Q_t|$ of synchronization tasks issued in each time slot on the performance of Algorithm 2 when $|Q_t| = 200$, 400, 600, 800, and 1,000, respectively. Fig. 6 demonstrates the similar performance behaviors to those in Fig. 3. As evidenced by Fig. 6(a), the collected



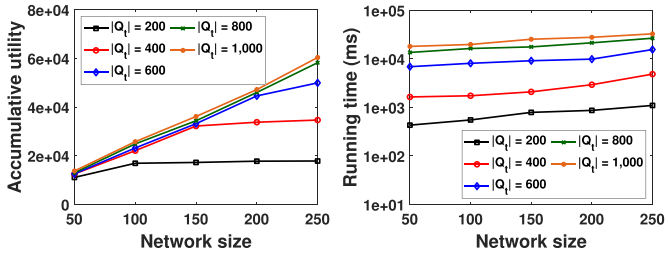
(a) Accumulative utility (b) Running time

Fig. 4. Impact of the update data size $a_{i,t}$ on Algorithm 1.



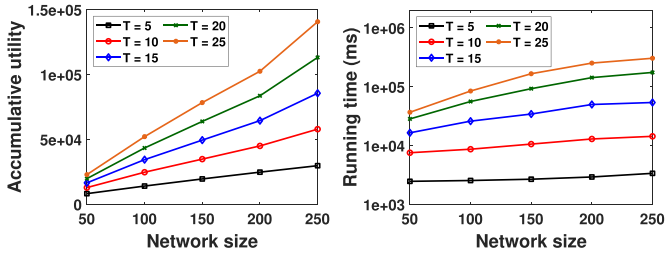
(a) Accumulative utility (b) Running time

Fig. 5. Algorithm performance for the dynamic DT freshness optimization problem.



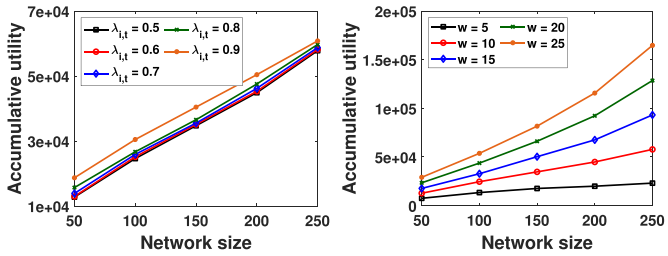
(a) Accumulative utility (b) Running time

Fig. 6. Impact of the number $|Q_t|$ of synchronization tasks in each time slot on Algorithm 2.



(a) Accumulative utility (b) Running time

Fig. 7. Impact of the length T of the time horizon on Algorithm 2.



(a) Accumulative utility (b) Accumulative utility

Fig. 8. Impact of parameters $\lambda_{i,t}$ and w on Algorithm 2.

utility of Algorithm 1 with $|Q_t| = 200$ is 29.5% of that by itself with $|Q_t| = 1,000$ when the number of cloudlets is 250. This is because Algorithm 2 is capable of collecting more utility gain when dealing with a larger number of synchronization tasks.

We further studied the impact of the length T of the time horizon on the performance of Algorithm 2 when there are 10, 20, 30, 40 and 50 time slots, respectively. Considering the network size of 250, Fig. 7(a) shows that the utility of Algorithm 2 with $T = 10$ is 20.9% of that by itself with $T = 50$. The reason is that Algorithm 2 dynamically determines where and when to process each synchronization task during the time horizon in an efficient way, and more utility gains are earned with a longer time horizon. It is observed from Fig. 7(b) that the more time slots lead to a longer running time of Algorithm 2, due to examining more synchronization tasks issued during a longer time horizon with more time slots.

We finally investigated the impacts of parameters $\lambda_{i,t}$ and w on the performance of Algorithm 2 when the network size is set at 50. Fig. 8(a) shows that the utility of Algorithm 2 with $\lambda_{i,t} = 0.5$ is 68.3% of that by itself with $\lambda_{i,t} = 0.9$. Meanwhile, when the network size is 250, Fig. 8(b) indicates that the utility of Algorithm 2 with $w = 5$ is 14.1% of that by itself with $w = 25$. It can be seen that a larger value of $\lambda_{i,t}$ or w leads to a larger utility, due to the definitions of DT freshness and utility by (3) and (4), respectively.

VII. Conclusion and Future Work

In this paper, we addressed the freshness issue of DT based services in MEC networks, by jointly synchronizing DTs to enhance the freshness of all DTs while minimizing various resource consumptions on DT synchronizations. To this end, we formulated both static and dynamic DT freshness optimization problems, respectively. We developed a constant approximation algorithm for the static DT freshness optimization problem, and an online algorithm with a provable competitive ratio for the dynamic DT freshness optimization problem, respectively. Finally, we evaluated the performance of the proposed algorithms by simulations. Simulation results show that the proposed algorithms outperform their baseline counterparts by no less than 13.2%.

Although the proposed algorithms are applicable to most realistic application settings, they may have limitations too, e.g., the accuracy of the solutions delivered by the proposed algorithms depends on the precise estimation on the amounts of demanded resources, which may be inaccurate in practice. In our future work, we will mitigate this limitation by developing robust yet accurate estimation on resource demands, and task prediction methods. We will also consider fairness issues among the requested DT services to improve the overall system performance.

Acknowledgments

The authors appreciate for the Associate Editor and five anonymous referees for their constructive comments and invaluable suggestions, which help us to improve the quality and presentation of the paper greatly.

REFERENCES

[1] X. Ai, W. Liang, and C. Liu, "Joint optimization of model retraining and inference services in DT-assisted edge computing," *IEEE Trans. Netw.*, vol. 34, pp. 1804–1819, 2026, doi: 10.1109/TON.2025.3632228.

[2] X. Ai, W. Liang, Y. Zhang, and W. Xu, "Fidelity-aware inference services in DT-assisted edge computing via service model retraining," *IEEE Trans. Serv. Comput.*, vol. 18, no. 4, pp. 2089–2102, Jul./Aug. 2025.

[3] J. A. Alzubi, O. A. Alzubi, I. Qiqieh, and A. Singh, "A blended deep learning intrusion detection framework for consumable edge-centric IoMT industry," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2049–2057, Feb. 2024.

[4] O. A. Alzubi, J. A. Alzubi, M. Alazab, A. Alrabea, A. Awajan, and I. Qiqieh, "Optimized machine learning-based intrusion detection system for fog and edge computing environment," *Electronics*, vol. 11, no. 19, 2022, Art. no. 3007.

[5] B. Bera, A. K. Das, and B. Sikdar, "Digital twins-empowered secure network slice access and isolation for consumer healthcare applications," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3429–3444, Nov./Dec. 2024.

[6] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Inf. Process. Lett.*, vol. 100, pp. 162–166, 2006.

[7] K. Gai, Y. Zhang, M. Qiu, and B. Thuraisingham, "Blockchain-enabled service optimizations in supply chain digital twin," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 1673–1685, May/Jun. 2023.

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

[9] Y. Han et al., "A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 268–284, Jan. 2023.

[10] Z. Ji, S. Wu, and C. Jiang, "Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3414–3429, Nov. 2023.

[11] J. Li et al., "AoI-aware service provisioning in edge computing for digital twin network slicing requests," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14607–14621, Dec. 2024.

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

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

[14] J. Li et al., "Mobility-aware utility maximization in digital twin-enabled serverless edge computing," *IEEE Trans. Comput.*, vol. 73, no. 7, pp. 1837–1851, Jul. 2024.

[15] J. Li et al., "AoI-aware, digital twin-empowered IoT query Serv. in mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 4, pp. 3636–3650, Aug. 2024.

[16] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 3017–3030, May 2023.

[17] J. Li, J. Wang, W. Liang, S. Das, and Q. Chen, "Improving the freshness of digital twins in edge computing," in *Proc. Int. Conf. Wireless Artif. Intell. Comput. Syst. Appl.*, 2025, pp. 74–86.

[18] J. Li, J. Wang, W. Liang, X. Jia, and A. Y. Zomaya, "Inference service fidelity maximization in DT-assisted edge computing," *IEEE Trans. Mobile Comput.*, vol. 25, no. 1, pp. 1352–1366, Jan. 2026.

[19] J. Li, J. Wang, W. Liang, J. Wu, Q. Chen, and Z. Xu, "Social-aware DT-assisted service provisioning in serverless edge computing," in *Proc 20th Int. Conf. Mobility, Sens. Netw.*, 2024, pp. 374–381.

[20] M. Li et al., "Automatic data generation and optimization for digital twin network," *IEEE Trans. Serv. Comput.*, vol. 18, no. 1, pp. 85–97, Jan./Feb. 2025.

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

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

[23] H. Liao et al., "Ultra-low AoI digital twin-assisted resource allocation for multi-mode power IoT in distribution grid energy management," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3122–3132, Oct. 2023.

[24] W. Lin, L. Li, J. Yuan, Z. Han, M. Juntti, and T. Matsumoto, "Age-of-information in first-come-first-served wireless communications: Upper bound and performance optimization," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9501–9515, Sep. 2022.

[25] W. Liu, Y. Fu, Y. Guo, F. Lee Wang, W. Sun, and Y. Zhang, "Two-timescale synchronization and migration for digital twin networks: A multi-agent deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 23, no. 11, pp. 17294–17309, Nov. 2024.

[26] N. C. Luong, T. L. Van, S. Feng, H. Du, D. Niyato, and D. I. Kim, "Edge computing for metaverse: Incentive mechanism versus semantic communication," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 6196–6211, May 2024.

[27] Y. Ma, W. Liang, J. Wu, and Z. Xu, "Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 2, pp. 393–407, Feb. 2020.

[28] T. Öncan, "A survey of the generalized assignment problem and its applications," *Inf. Syst. Oper. Res.*, vol. 45, no. 3, pp. 123–142, 2007.

[29] N. Paglierani, F. Linsalata, O. A. Sevimay, L. Cazzella, D. Badini, and M. Magarini, "Digital network twin-enabled synchronization and localization," *IEEE J. Sel. Areas Commun.*, early access, Sep. 16, 2025, doi: 10.1109/JSAC.2025.3610461.

[30] Y. Peng, J. Duan, J. Zhang, W. Li, Y. Liu, and F. Jiang, "Stochastic long-term energy optimization in digital twin-assisted heterogeneous edge networks," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 11, pp. 3157–3171, Nov. 2024.

[31] Y. Qu, S. Yu, L. Gao, K. Sood, and Y. Xiang, "Blockchained dual-asynchronous federated learning Serv. for digital twin empowered edge-cloud continuum," *IEEE Trans. Serv. Comput.*, vol. 17, no. 3, pp. 836–849, May/Jun. 2024.

[32] J. Tang, J. Nie, J. Bai, J. Xu, S. Li, and Y. Zhang, "UAV-assisted digital-twin synchronization with tiny-machine-learning-based semantic communications," *IEEE Internet Things J.*, vol. 11, no. 17, pp. 28437–28451, Sep. 2024.

[33] F. Tang, L. Luo, Z. Guo, Y. Li, M. Zhao, and N. Kato, "Semi-distributed network fault diagnosis based on digital twin network in highly dynamic heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 24, no. 5, pp. 3979–3992, May 2025.

[34] H. Tong et al., "Continual reinforcement learning for digital twin synchronization optimization," *IEEE Trans. Mobile Comput.*, vol. 24, no. 8, pp. 6843–6857, Aug. 2025.

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

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

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

[38] Z. Xu et al., "Schedule or wait: Age-minimization for IoT Big Data processing in MEC via online learning," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1809–1818.

[39] H. H. Yang, A. Arafa, T. Q. S. Quek, and H. V. Poor, "Spatiotemporal analysis for age of information in random access networks under last-come first-serve with replacement protocol," *IEEE Trans. Wireless Commun.*, vol. 21, no. 4, pp. 2813–2829, Apr. 2022.

[40] L. Yang, Y. Zou, S. Shen, P. Wang, D. Yu, and X. Cheng, "A fault-tolerant communication algorithm for age-of-information optimization in DITENs," *IEEE Trans. Commun.*, vol. 72, no. 5, pp. 2851–2864, May 2024.

[41] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[42] Y. Yi, G. Zhang, and H. Jiang, "Online digital twin-empowered content resale mechanism in age of information-aware edge caching networks," *IEEE Trans. Commun.*, vol. 73, no. 7, pp. 4990–5004, Jul. 2025.

[43] J. Yu, A. Alhilal, P. Hui, and D. H. K. Tsang, "Bi-directional digital twin and edge computing in the metaverse," *IEEE Internet Things Mag.*, vol. 7, no. 3, pp. 106–112, Mar. 2024.

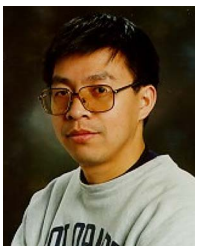
[44] Y. Zhang, J. Hu, and G. Min, "Digital twin-driven intelligent task offloading for collaborative mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 10, pp. 3034–3045, Oct. 2023.

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

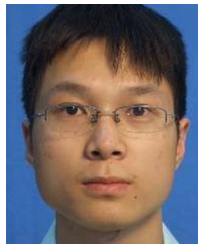
[46] Y. Zhang, W. Liang, and Y. Yang, "QoE-aware task executions on service models in DT-assisted edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 7, pp. 6209–6224, Oct. 20, 2025, doi: 10.1109/TMC.2025.3623582.
[47] Y. Zhang, L. Wang, and W. Liang, "Deep reinforcement learning for mobility-aware digital twin migrations in edge computing," *IEEE Trans. Serv. Comput.*, vol. 18, no. 7, pp. 704–717, Mar./Apr. 2025.
[48] L. Zhao et al., "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3386–3400, Nov. 2023.
[49] L. Zhou, S. Leng, and Q. Wang, "A federated digital twin framework for UAVs-based mobile scenarios," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 7377–7393, Jun. 2024.
[50] L. Zhou, S. Leng, and T. Q. S. Quek, "HaDT: Hardening digital twins for UAVs-based industrial logistics distribution systems," in *Proc. IEEE Conf. Comput. Commun.*, 2025, pp. 1–8.

Jing Li received the BSc (with first class Hons.) degree and the PhD degree from The Australian National University, in 2018 and 2022, respectively. He is currently a postdoctoral fellow with the City University of Hong Kong. His research interests include edge computing, Internet of Things, digital twin, network function virtualization, and combinatorial optimization.

Jianping Wang (Fellow, IEEE) received the BS and MS degrees in computer science from Nankai University, Tianjin, China, in 1996 and 1999, respectively, and the PhD degree in computer science from The University of Texas at Dallas, Richardson, TX, USA, in 2003. She is currently the chair professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. Her research interests include security, autonomous driving, cloud computing, edge computing, and machine learning.

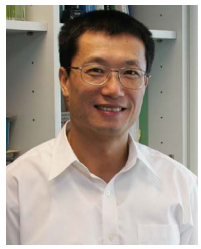


Weifa Liang (Fellow, IEEE) received the BSc degree in computer science from Wuhan University, China, in 1984, the ME degree in computer science from the University of Science and Technology of China, in 1989, and the PhD degree in computer science from Australian National University, in 1998. He is currently a full professor with the Department of Computer Science, City University of Hong Kong. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Internet of Things and digital twins, machine learning, mobile edge computing (MEC), network function virtualization (NFV), design and analysis of parallel and distributed algorithms, approximation and online algorithms, combinatorial optimization and graph theory. He is the editor of *IEEE Transactions on Communications* and distinguished contributor to the IEEE Computer Society.



Quan Chen (Member, IEEE) received the BS, master's, and PhD degrees from the School of Computer Science and Technology, Harbin Institute of Technology, China. He was a postdoctoral research fellow with the Department of Computer Science, Georgia State University. He is currently an associate professor with the School of Computers, Guangdong University of Technology. His research interests include routing and scheduling algorithms in wireless networks and sensor networks.

Sajal K. Das (Fellow, IEEE) is currently a Curators' distinguished professor of computer science and Daniel St. Clair endowed chair with the Missouri University of Science and Technology. He has authored or coauthored extensively with more than 600 research articles in high quality journals and refereed conference proceedings. His research interests include cyber-physical systems, IoT and sensor networks, mobile and pervasive computing, smart environments, HPC and edge-cloud computing, machine learning, cybersecurity, applied graph theory and game theory. He directed numerous funded projects in these areas totaling more than $25M USD. He is the founding editor-inchief of the Pervasive and Mobile Computing Journal and associate editor for *IEEE Transactions of Sustainable Computing*, IEEE Transactions on Dependable and Secure Computing, *IEEE Transactions of Networking*, and *ACM Transactions on Sensor Networks*. He is the co-founder of the IEEE PerCom, WoWMoM, SMARTCOMP, and ICDCN Conferences and served on numerous conference committees as general chair, technical program chair, or a program committee member.



Xiaohua Jia (Fellow, IEEE) received the BSc and ME degrees from the University of Science and Technology of China, in 1984 and 1987, respectively, and the DSc degree in information science from the University of Tokyo, in 1991. He is currently the chair professor with the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks, wireless sensor networks and mobile wireless networks. He is the editor of *IEEE Transactions on Parallel and Distributed Systems* (2006-2009), *Journal of World Wide Web*, *Wireless Networks*, and *Journal of Combinatorial Optimization*. He is the general chair of ACM MobiHoc 2008, TPC co-chair of IEEE MASS 2009, area-chair of IEEE INFOCOM 2010, TPC co-chair of IEEE GlobeCom 2010, panel co-chair of IEEE INFOCOM 2011, and general co-chair of IEEE ICDCS 2023.