AoI-Aware, Digital Twin-Empowered IoT Query Services in Mobile Edge Computing

Jing Li, Song Guo, Fellow, IEEE, Weifa Liang, Senior Member, IEEE, Jie Wu, Fellow, IEEE, Quan Chen, Member, IEEE, Zichuan Xu, Member, IEEE, Wenzheng Xu, Member, IEEE, and Jianping Wang, Fellow, IEEE

Abstract—The Mobile Edge Computing (MEC) paradigm gives impetus to the vigorous advancement of the Internet of Things (IoT), through provisioning of low-latency computing services at network edges. The emerging digital twin technique has been explosively growing in the IoT community, which bridges the gap between physical objects and their digital representations in an MEC network, enabling real-time monitoring and analysis, simulations on the dynamics of systems, accurate predictions on behaviours of objects, and optimization on network resource allocation. In this paper, we consider AoI-aware query services in an MEC network empowered by digital twin technology for diverse IoT applications. We aim to maximize the weighted sum of the accumulative freshness of query results measured by the Age of Information (AoI) and the total query service delay of admitted requests. To this end, we first formulate a novel minimization problem that explores nontrivial trade-offs between the two conflicting optimization objectives: the freshness of query results and service delays, and we show the NP-hardness of the problem. Then, we propose an approximation algorithm with a provable approximation ratio for the problem, at the expense of bounded computing capacity violations. We also develop a heuristic for the problem without any capacity violations. We finally evaluate the performance of the proposed algorithms via simulations. The simulation results demonstrate that the proposed algorithms are promising, and outperform the comparison benchmarks.

Index Terms—Digital twin, mobile edge computing, query services, age of information, IoT applications, service delays, approximation algorithms, resource allocation, optimization.

I. INTRODUCTION

The last decade witnessed the unprecedented explosion of Internet of Things (IoT) applications by culminating the proliferation of IoT devices connected with the Internet, thereby permeating the modern-day world and flourishing the potential for brilliant living [27]. However, most IoT devices have limited resources and energy for running IoT applications on themselves, instead, they usually offload computing-intensive tasks to remote clouds that may cause high service costs and prolonged service delays [8]. Such a computing paradigm is inappropriate for many delay-sensitive IoT applications due to stringent end-to-end delay requirements [32]. Moreover, in traditional IoT architectures, IoT devices store data in their backlogs for future diagnosis and improvement, which however may lead to stale feedback, unverified updates, and severe malfunctions [34].

Mobile Edge Computing (MEC) has been envisaged as a promising paradigm to provide computing resource (cloudlets) at the edge of the core network in the proximity of users to mitigate user service delays [4], [16], [17], [18]. The emerging digital twin technique creates digital avatars for IoT devices in MEC networks catering to massive data continuously generated from IoT devices, through leveraging integrated data and simulations to provide timely data analysis and modeling [23]. The marriage of MEC and digital twin techniques drives new opportunities and challenges for IoT service provisioning, accurate prediction, emulations, optimization, and decision-making.

1558-2566 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
facilitating efficient network management and resource allocations [15], [24].

Motivated by the recent studies in digital twin-empowered service provisioning in MEC [21], [24], [28], [31], [33], [34], the accuracy and fidelity of a service model are determined by how often it is retrained by the update data from its source objects. However, objects are not always accessible by the service model, due to their mobility and intermittent contact with the network. To overcome the unavailability of objects, there is a digital twin for each object in the MEC network, which is a virtual mirror of the object to reflect the status of the object. Thus, the service model always can access its DT status and data instead of the object itself. To maintain the freshness of the DT data of an object, the object needs to upload its update data to its DT quite often. In this study, we consider how to maintain the freshness of query results while minimizing query response delays of IoT applications in an MEC network empowered by digital twins of sensors, where sensors are scattered at diverse geographical locations to provide continuous sensory readings of time-varying environmental parameters (e.g., humidity or temperature). On account of the system dynamics, IoT applications usually rely on real-time and predictive data [9], however, portable sensors have limited computing, storage and communication resources. As a promising cure, the network service provider can create a digital twin for each sensor in a cloudlet to process its generated data and simulate its behaviours in the network [28]. For instance, digital twins facilitate an IoT application for forecasting weather and climate patterns, which is beneficial for smart farming.

The digital twin of each sensor contains not only its historical update locations, uploading times, the status of the object but also its update data at each time. We refer to these data as the DT data of the sensor, which can be used for service model training and retraining with the ultimate objective of providing intelligent services by Artificial Intelligence (AI) or Machine Learning (ML) algorithms. To maintain the state freshness of digital twins, it is desirable that each sensor can upload newly collected data to its digital twin on time. However, the volume of data uploaded by each sensor usually is constrained because of the limited energy and cost imposed on the sensor [2]. It thus poses a great challenge for each digital twin to instruct its sensor when performing data uploading, in order to provide the Age of Information (AoI)-aware IoT query services, subject to energy and cost capacities on the sensor, where a common metric to measure the data freshness is the AoI [1], [40], i.e., the duration of the data from its generation to its usage.

Quality of Services (QoS) of user queries on digital twin data usually is measured by two metrics: the freshness of query results and query service delays [2], [31], [34]. Meeting QoS requirements of different users for their IoT applications in MEC poses great challenges. First, the placement of IoT application instances to cloudlets impacts the freshness of query results and query service delays, e.g., a long distance between a cloudlet hosting an IoT application instance and its digital twins leads to a stale query result and a high service delay. How to deploy IoT application instances of users to the MEC network to optimize these two metrics is challenging. Second, it is very difficult to determine whether to utilize the current data at a digital twin for a query to have a lower query service delay, or to wait for the next update data of the digital twin to obtain a lower AoI on the query result. Finally, to provide fresh data for as many queries as possible, it is challenging to determine when scheduling sensors to upload their update data and synchronize with their digital twins over a finite time horizon, considering limited energy and cost budgets on sensors. In this paper, we will address the aforementioned challenges and explore nontrivial trade-offs between the freshness of query results and query service delays.

The novelty of this study lies in IoT service provisioning in a DT-empowered MEC network, by jointly considering the freshness of query results and query service delays, through DT update scheduling and IoT application instance placements. With the aim to minimize the weighted sum of AoIs of query results and service delays of admitted queries for a given time horizon, efficient approximation and heuristic algorithms for the problem are devised.

The main contributions of this paper are as follows.

- We formulate a novel minmization problem that jointly considers the accumulative freshness of query results and the total query service delay of admitted IoT service requests in an MEC network empowered by digital twins, and show the NP-hardness of the problem.
- We devise an approximation algorithm with a provable approximation ratio for the problem at the expense of bounded computing capacity violations. Meanwhile, we also develop an efficient heuristic algorithm for the problem without any capacity violations.
- We evaluate the algorithm performance via simulations. The simulation results demonstrate that the proposed approximation algorithm is promising.

The rest of the paper is arranged as follows. Section II surveyed related works of digital twins in MEC. Section III includes the system model and the problem definition. Section IV proposes an approximation algorithm for the problem with bounded capacity violations. Section V proposes a heuristic algorithm for the problem without any capacity violation. Section VI evaluates the performance of the proposed algorithms, and Section VII concludes the paper and points out a potential research topic from the research in this paper.

II. RELATED WORK

Lots of efforts have been taken in recent years to facilitate delay-sensitive IoT service provisioning in MEC platforms [6], [8], [17], [27], [32], [36], [38]. Gedawy et al. [6] designed heuristic algorithms to optimize the network throughput, as well as the energy consumed by IoT applications. Goudarzi et al. [8] developed an IoT application placement technique in MEC by the Memetic Algorithm (MA) to mitigate the execution time and energy consumption. Li et al. [17] offered approximate and online solutions to handle admitting multi-source IoT applications in MEC environments. Ma et al. [27] considered truthfulness and budget-balance of IoT services, by designing a truthful combinatorial double auction mechanism. Tuli et al. [32] explored the complicated workload dynamics of IoT devices and heterogeneous resources in edge-cloud networks. They
outlined a policy gradient-based learning method and captured the temporal patterns to build a framework for resource management in stochastic environments. Wu et al. [36] developed a distributed SDN controller for coordinating mobile IoT devices, enabling efficient location authentication and assignment between IoT devices and access points. These studies however did not consider data freshness of IoT service provisioning.

There are extensive studies on supplying fresh data of the provided IoT services within MEC environments, through minimizing the AoI [1], [2], [3], [11], [22], [35], [37], [43], [44]. Corneo et al. [2] investigated the problem of prompt dissemination of sensor updates to optimize the AoI of IoT services. Li et al. [11] proposed a heterogeneous federated multi-agent reinforcement learning algorithm to minimize the AoI in an Unmanned Aerial Vehicle (UAV)-assisted MEC system. Liu et al. [22] studied the batch generation and multi-path communication in MEC, and proposed approximation algorithms to minimize the peak and average AoI, while meeting the throughput requirements. Wang et al. [35] devised an offline scheduling algorithm and an online learning-based algorithm for minimizing the average age of critical information. Xu et al. [37] modelled the quality of big data analytic services based on the Age of Data (AoD) for IoT applications, and developed an online learning method to minimize the AoD in MEC, through leveraging the multi-armed bandit approach. Zhang et al. [43] explored a nontrivial trade-off between the AoI and the service delay, and proposed an efficient algorithm to mitigate the average service delay whilst meeting AoI requirements of user requests. Zhou et al. [44] addressed the dynamic demands of users with spectral efficiency features, through designing a congestion-based paradigm and an AoI-based paradigm to achieve network stability and maximize the throughput. However, none of these mentioned works incorporated the digital twin technique into their problem formulations.

MEC platforms empowered by the emerging digital twin technique enable the platforms to provide efficient services for various IoT applications [12], [14], [15], [20], [23], [24], [28], [31], [33], [34]. Li et al. [14], [15] estimated the reliability of virtual network functions by digital twins, and devised efficient algorithms to provide IoT services enabled by service function chains. Li et al. [12], [20] also incorporated the mobility of objects in edge-cloud environments, and proposed approximate and online solutions to provide Aoi-aware query services to users built upon the digital twin data. Lin et al. [23] proposed a congestion control scheme to meet the dynamic demands of digital twin services by Lyapunov optimization. Lu et al. [24] designed a federated learning algorithm built on the blockchain technique to improve security and data privacy in digital twin-assisted MEC networks. Sun et al. [31] utilized digital twins to minimize the offloading latency by the Lyapunov optimization, considering user mobility and service migration. Vaezi et al. [33] devised approximation algorithms to assign digital twins to execution servers at network edge to minimize the longest request-response delay, considering the data age constraints. Wang et al. [34] formulated two problems for managing digital twins to optimize the data fidelity and the reveal delay, respectively, where the data fidelity is the expected time from the production of the data to its delivery, and the reveal delay is defined as the time duration after the production of the latest collected log. They adopted the blockchain technique to design a sustainable digital twin management framework. These mentioned studies however did not investigate the joint optimization of AoIs of query results and query service delays for digital twin-empowered IoT services. Liang et al. [21] investigated the relationship between the freshness of a service model and the AoIs of the DT data for the model training. They devised an efficient algorithm for minimizing the cost of various resources consumed to achieve the model freshness. Zhang et al. [42] leveraged DT models to optimize device scheduling and MEC resource allocation, aiming to maximize utility across FL services. They developed heuristic and constant approximation algorithms for offline multi-FL services, and a DRL algorithm with dynamic bandwidth and moving client conditions settings.

In contrast to the aforementioned studies, in this paper we study IoT service provisioning in an MEC network empowered by the digital twin technique. We deal with AoI-aware query services with the aim to minimize the weighted sum of the accumulative AoI of query results and the total query delay of admitted queries, by striving for fine trade-offs between these two conflicting optimization objectives. It must be mentioned that part of this paper also appeared in a conference paper [13].

III. PRELIMINARIES

In this section, we introduce the system model and define the problem.

A. System Model
An MEC network can be modelled by an undirected graph $G = (V, E)$, where $V$ is the set of Access Points (APs), and $E$ is the set of links connecting APs. Each AP is co-located with a cloudlet that is connected through an optical cable, and the transmission delay between them is negligible [26]. Denote by $v \in V$ a cloudlet or its co-located AP for simplicity, and let $C_v$ be the computing capacity of cloudlet $v$. Each link $e \in E$ is associated with a transmission delay $d_e$ to transmit one unit data along the link [39].

Let $S$ be a set of sensors deployed across diverse geographical locations under the coverage of APs. We assume that each sensor $s \in S$ has a digital twin $DT(s)$ deployed in a cloudlet. Each digital twin $DT(s)$ needs to synchronize with its sensor $s$ often to maintain its state consistency as follows. Sensor $s$ sends its collected data to its nearest AP $v_s$, assuming that its $DT(s)$ has been placed in cloudlet $v_s$. Figure 1 illustrates an example of an MEC network with digital twins deployed in its cloudlets.

B. User Queries on Digital Twin Data of Sensors
We assume that MEC network $G$ runs in a discrete-time manner, and the monitoring time horizon $T$ is composed of $T$ equal time slots. Denote by $U$ the set of users with different IoT applications requesting data from digital twins of sensors. Assume that each user $u \in U$ deploys an instance for his IoT application in a cloudlet that requires the amount $c_u$ of computing resource in the beginning of time horizon $T$, and user $u$ issues queries (as his IoT application) for processing
Fig. 1. An MEC network consists of 5 Access Points (APs) and their co-located cloudlets. There are 4 sensors with each having a deployed digital twin, and 4 users requesting query services.

data from digital twins of different sensors in the beginning of different time slots.

Denote by $T_u \subseteq T$ the set of time slots in which user $u$ will issue his queries of data retrieving and processing, i.e., the deployed IoT application of user $u$ requests the data from the digital twin of a sensor $s_u$ at each time slot $t \in T_u \subseteq T$. For example, given $T = \{1, 2, 3\}$, the IoT application of user $u$ requests data from digital twins of sensors $s_1$ and $s_2$ in the beginning of time slots 1 and 3, respectively, with $T_u = \{1, 3\}$.

C. Updating Digital Twins of Sensors

Given a limited energy budget on each sensor $s \in \mathbb{S}$, we assume that sensor $s$ can deliver at most $K_s$ updates to its digital twin $DT(s)$ within time horizon $T$. The number of updates of any sensor is no larger than the number of time slots for the given monitoring time horizon, i.e., $K_s \leq |T|$, $\forall s \in \mathbb{S}$. For example, given $T = \{1, 2, 3\}$ and $K_s = 2$, sensor $s$ can send its updates at the beginning of time slots 1 and 3, respectively.

The data uploading rate $\mu_s$ from sensor $s$ to its allocated AP $v_s$ can be calculated by the Shannon-Hartley theorem [7], where $\mu_s = B_s \log_2(1 + P_s / (dist_s^2 \cdot \eta_s^2))$, $B_s$ is the bandwidth capacity of AP $v_s$, $P_s$ is the transmission power of sensor $s$, $dist_s$ is the distance between sensor $s$ and AP $v_s$, $\eta_s$ is the noise power, and $\alpha$ is the path loss factor, i.e., $\alpha = 2$ or 4 with regard to a short or large distance [4].

Denote by $\rho_s$ the volume of data per update of sensor $s$. Let $\rho_s / \mu_s$ be the data uploading delay from $s$ to cloudlet $v_s$ in which its digital twin $DT(s)$ is located. Denote by $f_s$ the processing rate of $DT(s)$ in cloudlet $v_s$. The processing delay of $DT(s)$ thus is $\rho_s / f_s$.

The update delay $t_s^{update}$ of digital twin $DT(s)$ consists of the data uploading delay from sensor $s$ to cloudlet $v_s$ and the processing delay of $DT(s)$ in cloudlet $v_s$, i.e.,

$$t_s^{update} = \rho_s / \mu_s + \rho_s / f_s.$$  \hspace{1cm} (1)

Suppose the current time slot is $t$, and the current data of digital twin $DT(s)$ (or the received result of a user) is the last updated one at time slot $t_0$ with $t_0 \leq t$, the Age of Information (AoI) of this generated data is defined as $(t - t_0)$ [40]. It can be seen that $t_s^{update}$ is the minimum AoI of data at $DT(s)$.

We assume that each $DT(s)$ of sensor $s \in \mathbb{S}$ has generated the initial data with the AoI of $t_s^{update}$ in the beginning of time horizon $T$. In the sequel, the AoI of the data at $DT(s)$ will linearly increase until the next update from sensor $s$. Assuming that sensor $s$ sends its first update to $DT(s)$ at time slot $t$, then $DT(s)$ will generate the data that will be available for IoT applications at time $(t + t_s^{update})$, and the AoI of the data at $DT(s)$ decreases to $t_s^{update}$ at time $(t + t_s^{update})$. The AoI of the data of $DT(s)$ then linearly increases again until receiving the next update from $s$. This procedure continues until the time horizon $T$ finishes.

D. QoS Model

We introduce a novel metric to measure the Quality of Service (QoS) of query services built upon digital twin data of sensors, which is the weighted sum of AoIs of query results, and query service delays (i.e., the duration between the query issuing time and query result receiving time). Recall that the IoT application of user $u \in U$ requests data from $DT(s,u,t)$ of sensor $s_u,t$ at each time slot $t \in T_u \subseteq T$. In the following, we omit subscripts of $s_u,t$ for notation simplicity, i.e., replace $s_u,t$ with $s$. Cloudlet $v_u$ hosts $DT(s)$ of sensor $s$. We assume cloudlet $v_u$ hosts the IoT application instance of user $u$. Denote by $d_{s,u}$ the transmission delay of transmitting a unit of data through the shortest path from cloudlet $v_u$ to cloudlet $v_w$ [39].

Let $\lambda_s$ be the volume of data at $DT(s)$ to be transferred, its transmission delay from $DT(s)$ to the IoT application instance of user $u$ is $\lambda_s \cdot d_{s,u}$. With the processing rate $f_u$ of the IoT application instance of user $u$, the processing delay of the IoT application of user $u$ in $v_u$ is $\lambda_s / f_u$.

At each time slot $t \in T_u$, a user $u$ needs to determine whether to retrieve the current data at $DT(s)$ or wait for its next update. If the user prefers a lower query service delay to a fresher AoI of the data, the user can retrieve the data of $DT(s)$ immediately; otherwise the user can wait for the next update of $DT(s)$ to obtain fresher AoI, at the expense of more query service delays. Note that the volume of a query result usually is small compared with query data, and the transmission delay of the query result between the cloudlet processing the IoT application and the user thus can be negligible [31].

Assume that the IoT application of user $u$ has been deployed in cloudlet $v_u$ and user $u$ issues a query for the data of digital twin $DT(s)$ of sensor $s$ at time slot $t$. We analyze the freshness of the query result and query service delay of this query by distinguishing the following two cases.

Case 1: User $u$ retrieves the current data at $DT(s)$. Let $t_0$ be the updating time of sensor $s$ to generate the current data of $DT(s)$. The AoI of query result of user $u$ is $\lambda_s \cdot d_{s,u} + \lambda_s / f_u + (t - t_0)$, where $\lambda_s \cdot d_{s,u}$ is the transmission delay of transmitting the data from $DT(s)$ in cloudlet $v_u$ to cloudlet $v_w$ hosting the IoT application of user $u$, $\lambda_s / f_u$ is the processing delay of the IoT application in cloudlet $v_u$, and $(t - t_0)$ is the AoI of the generated data at $DT(s)$. The query service delay of user $u$ is $\lambda_s \cdot d_{s,u} + \lambda_s / f_u$.

Case 2: User $u$ will retrieve the newly generated data of $DT(s)$ through waiting for its next update. Because the latest AoI of the generated data at $DT(s)$ is $t_s^{update}$ by Eq. (1), the AoI of the query result is $\lambda_s \cdot d_{s,u} + \lambda_s / f_u + t_s^{update}$. Let $t$ and $t'$ be the current time slot and the next update time slot of $DT(s)$. It can be seen that the data generated by the next
update of \( DT(s) \) will be available for IoT applications at time \( (t' + t_{\text{update}}^s) \). Therefore, the IoT application of user \( u \) needs to wait for \( (t' + t_{\text{update}}^s - t) \) time for the next update of \( DT(s) \). The query service delay of user \( u \) thus is \( \lambda_s \cdot d_{s,u} + \lambda_s/f_u + t' + t_{\text{update}}^s - t \).

In summary, if user \( u \in U \) issues a query at time slot \( t \in T_u \subseteq T \), then the AoI \( W_{\text{AoI}}(u,t) \) of the query result is

\[
W_{\text{AoI}}(u,t) = \begin{cases} 
\lambda_s \cdot d_{s,u} + \lambda_s/f_u + t - t_0, & \text{Case 1} \\
\lambda_s \cdot d_{s,u} + \lambda_s/f_u + t_{\text{update}}^s, & \text{Case 2}
\end{cases}
\]

and the query service delay \( W_{\text{delay}}(u,t) \) is

\[
W_{\text{delay}}(u,t) = \begin{cases} 
\lambda_s \cdot d_{s,u} + \lambda_s/f_u, & \text{Case 1} \\
\lambda_s \cdot d_{s,u} + \lambda_s/f_u + t' + t_{\text{update}}^s - t, & \text{Case 2}
\end{cases}
\]

Let \( \beta \) be a constant with \( 0 \leq \beta \leq 1 \). We define the weighted sum \( W(u,t) \) of the AoI of the query result and query service delay of a query issued by user \( u \) at time \( t \) as follows.

\[
W(u,t) = \beta \cdot W_{\text{AoI}}(u,t) + (1 - \beta) \cdot W_{\text{delay}}(u,t).
\]

### E. Problem Definition

**Definition 1:** Given an MEC network \( G = (V,E) \), a set \( S \) of sensors with limited energy capacity on each of them, a positive integer \( K_s \) for each sensor \( s \in S \), a set \( U \) of users issuing queries for IoT applications on sensors, and a finite time horizon \( T \). Assuming digital twins of sensors have already been deployed in cloudlets of \( V \) of \( G \). Also, assume that the query profile of each user \( u \in U \) for time horizon \( T \) is given (i.e., it is given that user \( u \) will retrieve which sensors at which time slots) too, and user \( u \) may retrieve data from digital twins of sensors in the beginning of time slots in \( T_u \subseteq T \). The minimization problem of jointly considering the accumulative freshness of query results and the total service delay of admitted queries, by placing IoT application instances of users to cloudlets, such that the average weighted sum of the accumulative AoI of query results and the total query delay of admitted queries for time horizon \( T \) is minimized, i.e.,

\[
\min \sum_{u \in U} \sum_{t \in T_u} W(u,t)/\sum_{u \in U} |T_u|,
\]

subject to computing capacities on cloudlets.

All symbols adopted are listed in Table 1.

### F. NP-Hardness of the Defined Problem

**Theorem 1:** The minimization problem is NP-hard.

**Proof:** The NP-hardness of the minimization problem is shown through a reduction from the minimum-cost Generalized Assignment Problem (GAP).

Consider a special case of the problem with a given update scheduling of each sensor over the time horizon, and users can only retrieve the current data at digital twins. Then, we only need to determine the IoT application instance placements of users. Let \( W'(u,t,v) \) be the value of \( W(u,t) \) by Eq. (4) for the query of user \( u \in U \) issuing at time \( t \) if its IoT application instance is deployed in cloudlet \( v \in V \). We reduce the problem to the minimum-cost GAP as follows.

In a minimum-cost GAP instance, there are \(|V|\) bins, and each bin \( v \in V \) possesses a capacity \( C_v \), i.e., computing capacity of cloudlet \( v \). There are \(|U|\) items and each item \( u \in U \) has a size \( c_u \), i.e., the computing resource demand of IoT application instance of user \( u \). Assigning item \( u \) to bin \( v \) will introduce a cost \( \sum_{t \in T_u} W'(u,t,v) \). The minimum-cost GAP is to minimize the total cost via assigning items to bins.

This minimum-cost GAP is equivalent to a special case of the minimization problem. The minimization problem is NP-hard as the minimum-cost GAP is NP-hard [29].

### IV. APPROXIMATION ALGORITHM WITH BOUNDED CAPACITY VIOLATIONS

In this section, we deal with the minimization problem. Specifically, we decompose the problem into two sub-problems: the update scheduling problem, and the IoT application placement problem. We propose an optimal solution to the former and an approximate solution for the latter, thereby proposing an approximation algorithm for the minimization problem.

We observe that optimization objective (5) is equivalent to minimizing the total weighted sum of Aois of query results and the total query service delay of all queries over a given time horizon, i.e.,

\[
\text{Minimize } \sum_{u \in U} \sum_{t \in T_u} W(u,t) .
\]

Considering a query issued by user \( u \) at time slot \( t \) for the data at \( DT(s) \). Let \( t_0 \) be the updating time of sensor \( s \) to generate the data at \( DT(s) \) at time slot \( t \) with \( t_0 \leq t \). We distinguish it into two cases: Case 1. There is no further update of sensor \( s \) before the time horizon ends. Case 2. The next update of sensor \( s \) is sent at time \( t' \) with \( t_0 < t' \) and \( t < t' + t_{\text{update}}^s \).

To minimize the optimization objective (6), two functions \( W_1(u,t) \) and \( W_2(u,t) \) are defined as follows.

\[
W_1(u,t) = \begin{cases} 
\beta \cdot (t - t_0), & \text{if Case 1} \\
\min\{\beta \cdot (t - t_0), t_{\text{update}}^s + (1 - \beta) \cdot (t' - t)\}, & \text{if Case 2}
\end{cases}
\]

and

\[
W_2(u,t) = \lambda_s \cdot d_{s,u} + \lambda_s/f_u .
\]

Especially, if there is no further update, \( W_1(u,t) = \beta \cdot (t - t_0) \), with \( (t - t_0) \) the AoI of the data at \( DT(s) \) when issuing the query. Otherwise, \( W_1(u,t) \) is the smaller one between \( \beta \cdot (t - t_0) \) and \( t_{\text{update}}^s + (1 - \beta) \cdot (t' - t) \), and \( (t' - t) \) is the duration between the query issuing time and the sending time of the next update. \( W_2(u,t) \) is the query service delay of user \( u \) that consists of the transmission delay of data transfer from \( DT(s) \) in cloudlet \( v_s \) to cloudlet \( v_u \) hosting the IoT application and the processing delay of the IoT application in cloudlet \( v_u \).

We claim that (i) the value of \( W_1(u,t) \) is determined by the update scheduling of sensor \( s \), that is, whether to retrieve current data at \( DT(s) \) or wait for its next update; and (ii) the value of \( W_2(u,t) \) is determined by the IoT application placement of user \( u \), as shown in Lemma 2 later.

We now define two sub-problems: the update scheduling problem and the IoT application placement problem, which...
TABLE I

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = (V, E)$</td>
<td>$G$ is an MEC network, $V$ is the set of APs (cloudlets) and $E$ is the set of links between APs.</td>
</tr>
<tr>
<td>$C_v$</td>
<td>The computing capacity of cloudlet $v$.</td>
</tr>
<tr>
<td>$d_e$</td>
<td>The transmission delay $d_e$ to transmit one unit data along the link $e \in E$.</td>
</tr>
<tr>
<td>$S, DT(s), v_s$</td>
<td>A set of sensors, a digital twin of sensor $s \in S$, the cloudlet holding $DT(s)$, and the cloudlet holding the IoT application of user $u$.</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>The monitoring time horizon.</td>
</tr>
<tr>
<td>$U$ and $c_u$</td>
<td>A set of users and the amount of computing resource of the IoT application of user $u$.</td>
</tr>
<tr>
<td>$T_u$</td>
<td>The set of time slots in which user $u$ will issue his queries of data retrieving and processing.</td>
</tr>
<tr>
<td>$K_s$</td>
<td>The number of the updates of sensor $s$ delivered to its digital twin within the time horizon.</td>
</tr>
<tr>
<td>$\mu_s$</td>
<td>The data uploading rate from sensor $s$ to its allocated AP $v_s$.</td>
</tr>
<tr>
<td>$B_s$, $P_s$, and $dist_s$</td>
<td>The bandwidth of AP $v_s$, the transmission power $P_s$ of sensor $s$, the distance between sensor $s$ and AP $v_s$.</td>
</tr>
<tr>
<td>$\eta^2$ and $\alpha$</td>
<td>The noise power and the path loss factor.</td>
</tr>
<tr>
<td>$f_s$</td>
<td>The processing rate of $DT(s)$ in cloudlet $v_s$.</td>
</tr>
<tr>
<td>$t^\text{update}_v$</td>
<td>The update delay of digital twin $DT(s)$ defined by Eq. (1).</td>
</tr>
<tr>
<td>$d_{u,v}$</td>
<td>The transmission delay of transmitting a unit of data through the shortest path from cloudlet $v_u$ to cloudlet $v_v$.</td>
</tr>
<tr>
<td>$\lambda_s$</td>
<td>The size of provided data at $DT(s)$.</td>
</tr>
<tr>
<td>$t_0$</td>
<td>The update sending time of sensor $s$ to generate the current data at $DT(s)$.</td>
</tr>
<tr>
<td>$t'$</td>
<td>The next update sending time of sensor $s$.</td>
</tr>
<tr>
<td>$W_{\text{AoI}}(u, t)$ and $W_{\text{delay}}(u, t)$</td>
<td>The AoI of the query result by Eq. (2) and the query service delay by Eq. (3).</td>
</tr>
<tr>
<td>$\beta$</td>
<td>A constant with $0 \leq \beta \leq 1$ associated with the AoI of the query result.</td>
</tr>
<tr>
<td>$W_1(u, t)$ and $W_2(u, t)$</td>
<td>The functions defined by Eq. (7) and Eq. (8), respectively.</td>
</tr>
<tr>
<td>$x_{u,v}$</td>
<td>A binary variable, indicating whether the IoT application instance of user $u \in U$ is deployed in cloudlet $v \in V$.</td>
</tr>
<tr>
<td>$W'_2(u, t, v)$</td>
<td>The value of $W_2(u, t)$ for the query of user $u$ issuing at time slot $t$ if the IoT application instance of user $u$ is deployed in cloudlet $v$.</td>
</tr>
<tr>
<td>$Q_s$</td>
<td>The set of queries of users in $U$ to request data of digital twin $DT(s)$ of sensor $s$ over time horizon $\mathcal{T}$.</td>
</tr>
</tbody>
</table>

Alternatively, there is an Integer Linear Programming (ILP) formulation for the IoT application placement problem as follows.

Let $x_{u,v}$ be a binary variable, where $x_{u,v} = 1$ means the IoT application instance of user $u \in U$ is deployed in cloudlet $v \in V$, and $x_{u,v} = 0$ otherwise. Let $W'_2(u, t, v)$ be the value of $W_2(u, t)$ by Eq. (8) for the query of user $u$ issuing at time slot $t$ if its IoT application instance is deployed in cloudlet $v$. An ILP solution for the IoT application placement problem is provided as follows.

Minimize $\sum_{u \in U} \sum_{t \in T_u} W_1(u, t)$, \hspace{1cm} (9)  

where $W_1(u, t)$ is defined in Eq. (7), by scheduling the $K_s$ updates for each sensor $s \in S$ over the time horizon $\mathcal{T}$.

**Definition 3:** Given an MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K_s$ for each sensor $s \in S$, a set $U$ of users with queries for IoT applications on sensors, and time horizon $\mathcal{T}$, assuming digital twins of sensors have already been deployed in cloudlets of $V$, each user $u \in U$ may retrieve data from digital twins of sensors at the beginning of time slots in $T_u \subseteq \mathcal{T}$, and we further assume that the query profile of each user $u$ is given (i.e., at which time slot, it will retrieve the data of digital twins of which sensors). The **update scheduling problem** in $G$ is to

Minimize $\sum_{u \in U} \sum_{t \in T_u} W_2(u, t)$, \hspace{1cm} (10)  

where $W_2(u, t)$ is defined in Eq. (8), by deploying IoT applications of users in $U$ on cloudlets, subject to computing capacities on cloudlets.

correspond to the two defined functions $W_1(u, t)$ in Eq. (7) and $W_2(u, t)$ in Eq. (8), respectively.

**Definition 2:** Given an MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K_s$ for each sensor $s \in S$, a set $U$ of users with queries for IoT applications on sensors, and time horizon $\mathcal{T}$, assuming digital twins of sensors have already been deployed in cloudlets of $V$, each user $u \in U$ may retrieve data from digital twins of sensors at the beginning of time slots in $T_u \subseteq \mathcal{T}$, and we further assume that the query profile of each user $u$ is given (i.e., at which time slot, it will retrieve the data of digital twins of which sensors). The **update scheduling problem** in $G$ is to

Minimize $\sum_{u \in U} \sum_{t \in T_u} W_1(u, t)$, \hspace{1cm} (9)  

where $W_1(u, t)$ is defined in Eq. (7), by scheduling the $K_s$ updates for each sensor $s \in S$ over the time horizon $\mathcal{T}$.

**Definition 3:** Given an MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K_s$ for each sensor $s \in S$, a set $U$ of users with IoT application queries, and time horizon $\mathcal{T}$, assuming digital twins of sensors have already been deployed in cloudlets of $V$, each user $u \in U$ may retrieve data from digital twins of sensors at the beginning of time slots in $T_u \subseteq \mathcal{T}$, and we further assume that the query profile of each user $u$ is given (i.e., at which time slot, it will retrieve the data of digital twins of which sensors). The **IoT application placement problem** in $G$ is to

Minimize $\sum_{u \in U} \sum_{t \in T_u} W_2(u, t)$, \hspace{1cm} (10)  

where $W_2(u, t)$ is defined in Eq. (8), by deploying IoT applications of users in $U$ on cloudlets, subject to computing capacities on cloudlets.

We claim that the optimal value of the optimization objective (6) is the sum of the optimal values of optimization objectives of the update scheduling problem and the IoT application placement problem, which will be shown in Lemma 4. Thus, minimizing the optimization objective (5) of the original problem is equivalent to minimizing the optimization objectives of the two sub-problems independently.

Authorized licensed use limited to: CITY UNIV OF HONG KONG. Downloaded on May 08,2024 at 00:29:52 UTC from IEEE Xplore. Restrictions apply.
For each edge \((s, n_{i,j})\) with \(s \leq n_{i,j} \leq s′\), which are scheduled in the first layer (the last layer). We add edges \((n_{i,j}, n_{i,j+1})\) for node \(n_{i,j}\) with \(j \in \mathbb{T}\), and edges \((n_{i,j+1}, D)\) for node \(n_{i,j+1}\) with \(j \in \mathbb{T}\) in the \(i\)th layer (the last layer). We add edges \((n_{i,j}, n_{i,j+1})\) with \(n_{i,j+1} \leq K_s - 1\) and \(1 \leq j < j′ \leq |\mathcal{T}|\), i.e., for each \(n_{i,j}\) in the \(i\)th layer, we add an edge \((n_{i,j}, n_{i,j+1})\) for each \(n_{i,j+1} \in \mathcal{T}\) in the \((i+1)\)th layer. A shortest path in \(G_s^o\) from source \(S\) to destination \(D\) will deliver an optimal update scheduling of sensor \(s\), which will be shown later.

The weight assignment of edges in \(E_s^o\) is given as follows. For each edge \((S, n_{i,j})\) in \(E_s^o\) with \(j \in \mathbb{T}\), its weight is \(w(S, n_{i,j}) = \sum_{q \in Q_i, (0,j)} W_1(u_q, t_q)\), with \(W_1(u_q, t_q)\) defined in Eq. (7). If edge \((S, n_{i,j})\) is included in a shortest path in \(G_s^o\) from \(S\) to \(D\), it implies that the first update of sensor \(s\) is sent at the beginning of time slot \(j\). For each edge \((n_{i,j}, n_{i+1,j′})\) in \(E_s^o\) with \(1 \leq i \leq K_s - 1\) and \(1 \leq j < j′ \leq |\mathcal{T}|\), its weight is \(w(n_{i,j}, n_{i+1,j′}) = \sum_{q \in Q_i, (j,j′)} W_1(u_q, t_q)\). Similarly, if edge \((n_{i,j}, n_{i+1,j′})\) is included in a shortest path in \(G_s^o\) from \(S\) to \(D\), the \(i\)th and \((i+1)\)th updates are sent by sensor \(s\) at the beginning of time slots \(j\) and \(j′\), respectively. For each edge \((n_{K_s,j}, D)\) in \(E_s^o\) with \(j \in \mathbb{T}\), its weight is \(w(n_{K_s,j}, D) = \sum_{q \in Q_i, (j+1)} W_1(u_q, t_q)\). If edge \((n_{K_s,j}, D)\) is included in a shortest path in \(G_s^o\) from \(S\) to \(D\), the \(K_s\)th update is sent by sensor \(s\) at the beginning of time slot \(j\).

We claim that a shortest path \(P_s^*\) in \(G_s^o\) from \(S\) to \(D\) corresponds to an optimal solution to the update scheduling of sensor \(s\), i.e., \(\sum_{e \in P_s^*} w(e)\) is the minimum value of \(\sum_{q \in Q_i} W_1(u_q, t_q)\), which will be shown later in Theorem 2.

To this end, we first construct an auxiliary graph \(G_s^o\) for each sensor \(s \in \mathbb{S}\), and find a shortest path \(P_s^*\) in \(G_s^o\), and \(\sum_{e \in E_s} w(e)\) is the minimum value of \(\sum_{e \in E_s} w(e)\) with the solution of \(P_s^*\). The found \(P_s^*\) is an optimal solution to the update scheduling problem.

An illustrative example of the construction of auxiliary graph \(G_s^o\) is given in Fig. 2, where sensor \(s\) can have \(K_s\) updates over time horizon \(T = \{1, 2, 3, 4\}\). \(P_s^* = \{(S, n_{i,1,1}), (n_{i,2,1}), (n_{i,3,3}), (n_{i,4,4}, D)\}\) is the shortest path in \(G_s^o\) from \(S\) to \(D\), which implies that \(K_s = 3\) updates of sensor \(s\) will be sent to its \(DT(s)\) in the beginning of time slot 1, 3, and 4, respectively.

The algorithm for the update scheduling problem is detailed in Algorithm 1.

**Algorithm 1 An Optimal Algorithm for the Update Scheduling Problem**

**Require**: An MEC network \(G = (V, E)\), a set \(\mathbb{S}\) of sensors, a positive integer \(K_s\) for each \(s \in \mathbb{S}\), a set \(U\) of users, and time horizon \(T\).

**Ensure**: Minimize \(\sum_{e \in E_s} \sum_{t \in \mathbb{T}} W_1(u_t, t)\) by scheduling the updates of sensors in \(\mathbb{S}\) over the time horizon \(|\mathbb{T}|\).

1. \(A_1 \leftarrow \emptyset\); \(P^*\) the solution */
2. for each sensor \(s \in \mathbb{S}\) do
3. Construct an auxiliary graph \(G_s^o = (N_s^o, E_s^o)\) for sensor \(s\), and find the shortest path \(P_s^*\) in \(G_s^o\) from source \(S\) to destination \(D\), which delivers the optimal solution \(A_{1,s}\) for updating scheduling of sensor \(s\);
4. \(A_1 \leftarrow A_1 \cup \{A_{1,s}\}\);
5. end for
6. return Solution \(A_1\).

**B. Approximation Algorithm for the IoT Application Placement Problem**

We propose an approximation algorithm with a provable approximation ratio for the IoT application placement problem, through a reduction to the minimum-cost Generalized Assignment Problem (GAP) [30]. An approximate solution to the minimum-cost GAP will in turn return an approximate solution to the IoT application placement problem.
Algorithm 2 An Approximation Algorithm for the IoT Application Placement Problem

Require: An MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K$, for each $s \in S$, a set $U$ of users, and time horizon $T$.

Ensure: Minimize $\sum_{u \in U} \sum_{t \in T_u} W_2(u, t)$ by deploying the IoT application instances of users in $U$ on cloudlets in $V$.

1: $A_2 \leftarrow \emptyset$ /* the solution */
2: Find a shortest path between each pair of cloudlets;
3: Solve the relaxed LP of the ILP (11);
4: Obtain the optimal solution $OPT$ of the LP, where $\hat{x}_{u,v} \in [0, 1]$ is the value of each $x_{u,v}$;
5: Construct a bipartite graph $B = (R, V; E)$ by [30].
6: Identify a minimum-cost matching $M$ in $B$, where all nodes in $R$ are matched, via Hungarian algorithm;
7: for each edge $(R_u, \gamma_{v,i}) \in M$ do
8: A solution $A_{2,u}$ for user $u$ is obtained, by deploying the IoT application instance of user $u$ in cloudlet $v$;
9: $A_2 \leftarrow A_2 \cup \{A_{2,u}\}$;
10: end for
11: return Solution $A_2$.

We reduce the IoT application placement problem to a minimum-cost GAP as follows.

Each IoT application instance of user $u \in U$ is an item $u$ with size $c_u$, while each cloudlet $v \in V$ is a bin with capacity $C_v$. Packing item $u$ to bin $v$ incurs a cost $\sum_{t \in T_u} W_2(u, t, v)$. By applying the approximation algorithm [30] for the minimum-cost GAP, an approximation algorithm for the IoT application placement problem then can be derived. Specifically, let $OPT$ be an optimal fractional solution of the linear relaxation (LP) of the ILP (11) and $\hat{x}_{u,v} \in [0, 1]$ the fractional value of $x_{u,v}$ by $OPT$.

A bipartite graph $B = (R, V; E)$ is then constructed, by adopting the method in [30], where $R$ and $V$ are two disjoint node sets, $R = \{R_u \mid u \in U\}$, $R_u$ is a node corresponding to the IoT application instance of user $u \in U$, $V = \{\gamma_{v,i} \mid v \in V, 1 \leq i \leq \eta_v\}$, and $\eta_v = |\sum_{R_u \in R} x_{u,v}|$ for each cloudlet $v \in V$. The edge set $E$ of $B$ consists of edges connecting nodes in $R$ and $V$, which is constructed as follows.

Considering each cloudlet $v \in V$, each node in $\{\gamma_{v,i} \mid 1 \leq i \leq \eta_v\}$ is regarded as a bin with capacity 1. Each node $R_u \in R$ with $\hat{x}_{u,v} > 0$ is regarded as an item with size of $\hat{x}_{u,v}$. Sort nodes (items) in $R$ in non-increasing order of $c_u$, and let $R_1 \geq R_2 \geq \ldots \geq R_{|U|}$ be the sorted items for notation simplicity. The sorted items are then packed into bins. Specifically, the sorted items are packed to the first bin $\gamma_{v,1}$ one by one until it causes the capacity violation of $\gamma_{v,1}$ by packing an item (e.g., $R_1$). Then, item $R_u$ is partitioned into two disjoint parts: $R_{u,1}$ and $R_{u,2}$, such that part $R_{u,1}$ packing into bin $\gamma_{v,1}$ makes $\gamma_{v,1}$ have no residual capacity. Part $R_{u,2}^{\prime}$ is then packed into the next bin $\gamma_{v,2}$. This procedure repeats until all items in $R$ are packed. If partial $R_u$ is packed into bin $\gamma_{v,i}$, an edge $(R_u, \gamma_{v,i})$ with cost $\sum_{t \in T_u} W_2(u, t, v)$ is added to set $E$.

Having constructed the bipartite graph $B$, find a minimum-cost maximum matching $M$ in $B$, where all nodes in $R$ are matched. For each edge $(R_u, \gamma_{v,i}) \in M$, the IoT application instance of user $u$ will be deployed in cloudlet $v$.

The algorithm is detailed in Algorithm 2.

Algorithm 3 An Approximation Algorithm for the Minimization Problem

Require: An MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K$, for each $s \in S$, a set $U$ of users, and time horizon $T$.

Ensure: Minimize $\sum_{u \in U} \sum_{t \in T_u} W(u, t)/|T_u|$, i.e., the average weighted sum of AoIs of query results and query service delays of all queries over $T$.

1: Formulate the updating scheduling problem, and its solution $A_1$ is obtained by Algorithm 1;
2: Formulate the IoT application placement problem, and its solution $A_2$ is obtained by Algorithm 2;
3: A solution $A$ is obtained to the minimization problem, by adopting the update scheduling of sensors in $A_1$ and the IoT application placement of users in $A_2$;
4: return Solution $A$.

D. Algorithm Analysis

**Lemma 1:** Let $W^*(u, t)$ be the value of $W(u, t)$ by Eq. (4) in an optimal solution of the minimization problem. Given a query of user $u$ for $DT(s)$ at time slot $t$, and the update sending time $t_0$ of sensor $s$ to generate the current data at $DT(s)$, we consider two cases: Case 1: There is no further update of sensor $s$ before time horizon $T$ ends. User $u$ then can only retrieve the current data of $DT(s)$, and we have $W^*(u, t) = \lambda_s \cdot d_s/a + \lambda_s/f_u + \beta \cdot (t - t_0)$. Case 2: Assuming the next update of $s$ is scheduled at time slot $t'$, we have $W^*(u, t) = \lambda_s \cdot d_s + \lambda_s/f_u + \min\{\beta \cdot (t - t_0), \; \gamma_s^{update} + (1 - \beta) \cdot (t' - t)\}$, and user $u$ prefers to retrieve the current data of $DT(s)$ to minimize $W(u, t)$, if $\beta \cdot (t - t_0) < \gamma_s^{update} + (1 - \beta) \cdot (t' - t)$; otherwise, user $u$ prefers to wait for the next update of $DT(s)$.

**Proof:** We show the claim by distinguishing the following two cases.

**Case 1:** Due to no further update on $DT(s)$ from now on until the end of the time horizon, user $u$ cannot wait for the next update of $DT(s)$ anymore. The user can only retrieve the current data of $DT(s)$. From Eq. (2), (3) and (4), we have

$$W^*(u, t) = \lambda_s \cdot d_s + \lambda_s/f_u + \beta \cdot (t - t_0).$$

(15)
Case 2: From Eq. (2), (3) and (4), if user \( u \) retrieves the data at \( DT(s) \) at time slot \( t \), we have
\[ W(u,t) = \lambda_s \cdot d_{s,u} + \lambda_u / f_u + \beta \cdot (t - t_0). \] (16)
Otherwise (i.e., user \( u \) waits for the next update of \( DT(s) \)),
\[ W(u,t) = \lambda_s \cdot d_{s,u} + \lambda_u / f_u + t_{update}^s + (1 - \beta) \cdot (t' - t). \] (17)
Therefore, if user \( u \) prefers retrieving the current data of \( DT(s) \), we have
\[ \beta \cdot (t - t_0) < t_{update}^s + (1 - \beta) \cdot (t' - t). \] (18)
Also, \( W^*(u,t) \) is the smaller one between Eq. (16) and (17), i.e.,
\[ W^*(u,t) = \lambda_s \cdot d_{s,u} + \lambda_u / f_u + \min \{ \beta \cdot (t - t_0), t_{update}^s + (1 - \beta) \cdot (t' - t) \}. \] (19)

**Lemma 2:** Given a query of user \( u \) at time slot \( t \) for data at \( DT(s) \), the update sending time \( t_0 \) of sensor \( s \) to generate the current data at \( DT(s) \), and the next update sending time \( t' \) of sensor \( s \) if available, (i) the value of \( W_1(u,t) \) by Eq. (7) is determined by the update scheduling of sensor \( s \), and shows whether to retrieve current data at \( DT(s) \) or wait for its next update; and (ii) the value of \( W_2(u,t) \) by Eq. (8) is determined by the IoT application instance placement of user \( u \).

**Proof:** (i) Recall that \( t_{update}^s \) is a constant, i.e., the update delay of \( DT(s) \) by Eq. (1), and \( \beta \) is a constant in Eq. (4). Then \( W_1(u,t) \) is determined by the update scheduling of sensor \( s \) (i.e., determining values of \( t_0 \) and \( t' \)). By Lemma 1, the \( \min \{ \cdot \} \) operation in Eq. (7) for \( W_1(u,t) \) shows whether to retrieve the current data at \( DT(s) \) or wait for its next update.

(ii) Recall that \( d_{s,u} \) is the transmission delay of transmitting a unit of data through the shortest path between cloudlets \( v_s \) and \( v_u \), where \( v_s \) holds \( DT(s) \), and \( v_u \) holds the IoT application instance of user \( u \), while \( \lambda_s \) is the size of the generated data at \( DT(s) \), and \( f_u \) is the processing rate of the IoT application instance of user \( u \). As \( f_u \) and \( \lambda_u \) are constants, the value of \( W_2(u,t) \) is only determined by the IoT application placement of user \( u \) (i.e., determining the value of \( d_{s,u} \)).

**Lemma 3:** Given a sensor \( s \) with \( K_s \) updates to be sent over the time horizon \( T \), and the constructed auxiliary graph \( G'_s = (N'_s, E'_s) \), (i) each potential update scheduling of \( s \) over time horizon \( T \) corresponds to a potential path in \( G'_s \) from source \( S \) to destination \( D \); and (ii) a potential path in \( G'_s \) from source \( S \) to destination \( D \) corresponds to a feasible update scheduling of \( s \) over the time horizon \( T \).

**Proof:** (i) Assume that a potential update scheduling of sensor \( s \) is at time slots \( t_1, t_2, \ldots, t_K \), respectively, with \( 1 \leq t_1 < t_2 \leq \ldots < t_K \leq |T| \). Then we can construct a path \( P_s \) in \( G'_s \) from \( S \) to \( D \) with \( P_s = \{ (S, n_{1,t_1}), (n_{1,t_1}, n_{2,t_2}), \ldots, (n_{K_s - 1,t_{K_s - 1}}, n_{K_s,t_K}) \}, (n_{K_s,t_K}, D) \} \). Now, we show that path \( P_s \) is feasible, i.e., all edges in \( P_s \) exist in \( G'_s \). We add edges \( \{ (S, n_{1,t_1}), (n_{1,t_1}, n_{2,t_2}), \ldots, (n_{K_s - 1,t_{K_s - 1}}, n_{K_s,t_K}) \} \) in \( G'_s \), because we also add edges \( \{ (n_{i,j+1}, n_{i,j+1+1}) \} \) with \( 1 \leq i \leq K_s - 1 \) and \( 1 \leq j < \lfloor |T| \rfloor \). Edge \( (n_{K_s,t_K}, D) \) exists in \( G'_s \), as we add edges \( \{ (n_{K-s}, D) \} \) for each \( n_{K-s} \) with \( j \in \mathbb{T} \). Thus, path \( P_s \) is feasible because all edges in \( P_s \) exist in \( G'_s \).

(ii) Let \( P_s \) be a potential path in \( G'_s \) from source \( S \) to destination \( D \). Because \( P_s \) starts from \( S \) and only edges \( (S, n_{1,t_1}), \ldots, (n_{K_s,t_K}, D) \) start at \( S \), we assume that the first edge of path \( P_s \) is \( (S, n_{1,t_1}) \) with \( 1 \leq t_1 \leq |T| \). Also, only edges \( (n_{1,t_1}, n_{2,t_2}) \) start from node \( n_{1,t_1} \) with \( 1 \leq t_1 < j' \leq |T| \), because we add edges \( \{ (n_{i,j}, n_{i,j+1}) \} \) with \( 1 \leq i \leq K_s - 1 \) and \( 1 \leq j < j' \leq |T| \). Then we assume that the second edge of \( P_s \) is \( (n_{1,t_1}, n_{2,t_2}) \) with \( 1 \leq t_1 < t_2 \leq |T| \). Similarly, the first \( K_s \) edges of \( P_s \) are \( (S, n_{1,t_1}), (n_{1,t_1}, n_{2,t_2}), \ldots, (n_{K_s - 1,t_{K_s - 1}}, n_{K_s,t_K}) \) with \( 1 \leq t_1 < t_2 < \ldots < t_{K_s} \leq |T| \). The last edge of \( P_s \) is \( (n_{K_s,t_K}, D) \) as only this edge starts from \( n_{K_s,t_K} \) and reaches \( D \). In such a potential path \( P_s = \{ (S, n_{1,t_1}), (n_{1,t_1}, n_{2,t_2}), \ldots, (n_{K_s - 1,t_{K_s - 1}}, n_{K_s,t_K}), (n_{K_s,t_K}, D) \} \), the derived update scheduling (i.e., send updates at time \( t_1, t_2, \ldots, t_{K_s} \)) is feasible, because there are \( K_s \) updates and \( 1 \leq t_1 < t_2 < \ldots < t_{K_s} \leq |T| \).

**Theorem 2:** Given an MEC network \( G = (V, E) \), a set \( S \) of sensors, a positive integer \( K_s \) for each sensor \( s \in S \), a set \( U \) of users, and time horizon \( T \), assuming digital twins of sensors have already been deployed in cloudlets of \( V \) in \( G \), a user...
$u \in U$ may retrieve data from digital twins of sensors at the beginning of time slot $t \in T_u \subseteq T$. There is an approximation algorithm, **Algorithm 2**, for the IoT application placement problem. The solution value by **Algorithm 2** is no greater than that of the optimal solution, and the computing resource consumed in each cloudlet $v \in V$ is no greater than twice its computing capacity. **Algorithm 2** takes $O(|U| \cdot |T| + |S| \cdot K_{\text{max}}^2 \cdot |T|^2 + |U|^3 \cdot |V|^3)$ time.

**Proof:** Because we reduce the IoT application placement problem to the minimum-cost GAP, and devise **Algorithm 2** for the problem by adopting the approximation technique in [30], readers can refer to [30] for more details, which are omitted due to space limitation.

**Lemma 4:** Let OPT be the optimal solution to the problem of concern, and let OPT$_1$ and OPT$_2$ be the optimal solutions to the update scheduling problem and the IoT application placement problem, respectively. Then, $OPT = \frac{OPT_1 + OPT_2}{\sum_{v \in V} |T_v|}$. 

**Proof:** Let $W^*(u, t)$ be the value of $W(u, t)$ by Eq. (4) in OPT. Let $W_1^*(u, t)$ be the value of $W_1(u, t)$ by Eq. (7) in OPT$_1$ and $W_2^*(u, t)$ the value of $W_2(u, t)$ by Eq. (8) in OPT$_2$, respectively. Then, $OPT = \sum_{u \in U} \sum_{t \in T_u} W^*(u, t)$, $OPT_1 = \sum_{u \in U} \sum_{t \in T_u} W_1^*(u, t)$, and $OPT_2 = \sum_{u \in U} \sum_{t \in T_u} W_2^*(u, t)$. We show that $\sum_{u \in U} \sum_{t \in T_u} W^*(u, t) = \sum_{u \in U} \sum_{t \in T_u} W_1^*(u, t) + \sum_{u \in U} \sum_{t \in T_u} W_2^*(u, t)$ as follows.

By Lemma 1, Lemma 2, Eq. (7) and Eq. (8), it can be seen that a feasible solution $W^*(u, t)$ can be obtained, by combining the optimal solutions $W_1^*(u, t)$ and $W_2^*(u, t)$ for the updating scheduling problem and IoT application placement problem, respectively, i.e.,

$$W^*(u, t) = W_1^*(u, t) + W_2^*(u, t).$$

Because of $\sum_{u \in U} \sum_{t \in T_u} W^*(u, t)$ indicating a feasible solution $OPT^*$ to the problem, we have $\sum_{u \in U} \sum_{t \in T_u} W^*(u, t) \geq \sum_{u \in U} \sum_{t \in T_u} W_1^*(u, t)$ by contradiction. We assume $OPT^* = \sum_{u \in U} \sum_{t \in T_u} W^*(u, t)$ and $OPT = \sum_{u \in U} \sum_{t \in T_u} W^*(u, t) = OPT$. Because OPT outperforms OPT$, OPT$ must adopt either another update scheduling of sensors (different from OPT$_1$) or another IoT application placement (different from OPT$_2$), such that OPT $<$ OPT*. However, such an update scheduling of sensors or IoT application placement will lead to a larger objective value of either the update scheduling problem or the IoT application placement problem than OPT$_1$ and OPT$_2$, respectively. This contradicts that OPT$_1$ and OPT$_2$ are the optimal solutions of the two problems. Therefore, we have $\sum_{u \in U} \sum_{t \in T_u} W^*(u, t) = \sum_{u \in U} \sum_{t \in T_u} W_1^*(u, t)$. By Eq. (20), we have $\sum_{u \in U} \sum_{t \in T_u} W_1^*(u, t) = \sum_{u \in U} \sum_{t \in T_u} W_2^*(u, t)$. Thus, we have $OPT = \frac{OPT_1 + OPT_2}{\sum_{v \in V} |T_v|}$.

**Theorem 4:** Given a MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K_s$ for each sensor $s \in S$, a set $U$ of users with IoT application queries, and time horizon $T$, assuming that digital twins of sensors have already been deployed in cloudlets of $V$ in $G$, a user $u \in U$ may retrieve data from digital twins of sensors at the beginning of time slot $t \in T_u \subseteq T$. There is an approximation algorithm, **Algorithm 3**, for the minimization problem. The solution value by **Algorithm 3** is no greater than that of the optimal solution, and the computing resource consumed in each cloudlet $v \in V$ is no greater than twice its computing capacity. **Algorithm 3** takes $O(|U| \cdot |T| + |S| \cdot K_{\text{max}}^2 \cdot |T|^2 + |U|^3 \cdot |V|^3)$ time with $K_{\text{max}} = \max\{K_s | s \in S\}$.

**Proof:** Let OPT, OPT$_1$, and OPT$_2$ be the optimal solutions of the minimization problem, the update scheduling problem, and the IoT application placement problem, respectively. Let $A_1, A_2$, and $A_2$ be the values of the solutions by **Algorithm 3**, **Algorithm 1**, and **Algorithm 2**, respectively. We have $A_1 = OPT_2$ by Theorem 2, $A_2 \leq OPT_2$ by Theorem 3, and $OPT = OPT_1 + OPT_2 \leq \sum_{v \in V} |T_v|$ by Lemma 4.

From Lemma 1, Eq. (7) and (8), we have $A_1 \leq A_2 \leq A_1 + A_2 \leq OPT_1 + OPT_2 \leq \sum_{v \in V} |T_v| = OPT$ with the computing resource consumed of each cloudlet no greater than twice its computing capacity, by Theorem 3.

**Algorithm 3** takes the time $O(|U| \cdot |T| + |S| \cdot K_{\text{max}}^2 \cdot |T|^2 + |U|^3 \cdot |V|^3)$, by time complexity of **Algorithm 1** and **Algorithm 2** in Theorem 2 and 3, respectively.

**V. HEURISTIC ALGORITHM WITHOUT COMPUTING CAPACITY VIOLATIONS**

In the previous section, we developed an approximate solution for the minimization problem at the expense of bounded computing capacity violations. In this section we devise an efficient heuristic algorithm for the problem without computing capacity violation as follows.

**Algorithm 4** An Algorithm for the IoT Application Placement Problem Without Resource Violations

**Require:** An MEC network $G = (V, E)$, a set $S$ of sensors, a positive integer $K_s$ for each sensor $s \in S$, a set $U$ of users, and time horizon $T$.

**Ensure:** Minimize $\sum_{u \in U} \sum_{t \in T_u} W_2(u, t)$ by deploying IoT application instances of queries in $U$ on cloudlets.

1: $A_2' \leftarrow \emptyset$; */ the solution */
2: $l \leftarrow 1$;
3: $U(l) \leftarrow U$, $V(l) \leftarrow V$, and identify $E(l)$;
4: while $E(l) \neq \emptyset$ do
5: Construct bipartite graph $B(l) = (U(l), V(l); E(l))$;
6: Find a minimum-cost maximum matching $M_l$ in $B(l)$, via the Hungarian algorithm;
7: for each matching edge $e(u, v)$ in $M_l$ do
8: A solution $A_{2, u}$ for user $u$ is obtained, by deploying the IoT application instance of user $u$ in cloudlet $v$;
9: end for
10: $A_2' \leftarrow A_2' \cup \{A_{2, u}\}$;
11: $l \leftarrow l + 1$;
12: Update $U(l)$, $V(l)$ and $E(l)$;
13: end while
14: return Solution $A_2'.$

**A. Algorithm for the IoT Application Placement Problem**

The proposed algorithm is based on a series of minimum-cost maximum matchings on their corresponding auxiliary graphs. Specifically, the algorithm proceeds iteratively. Within iteration $l$, we build a bipartite graph $B(l) = (U(l), V(l); E(l)),$
where \( U^{(l)} \) is the set of users, and their IoT applications have not been placed before iteration \( l \). \( V^{(l)} \) is the set of cloudlets possessing residual computing resource in iteration \( l \). We have \( U^{(l)} = U \), and \( V^{(l)} = V \) initially. \( E^{(l)} \) is a set of edges in iteration \( l \), and there exists an edge \( e(u,v) \) in \( E^{(l)} \) between a user \( u \in U^{(l)} \) and a cloudlet \( v \in V^{(l)} \) with weight \( \sum_{t \in T_u} W_2(u,t,v) \) if cloudlet \( v \) has sufficient residual resource to establish an IoT application instance for user \( u \).

Denote by \( M_l \) a minimum-cost maximum matching in \( B^{(l)} \). For each matching edge \( e(u,v) \in M_l \), we deploy the IoT application of user \( u \) in cloudlet \( v \), and the computing resource \( c_v \) is subtracted from cloudlet \( v \). Then, the subsequent auxiliary bipartite graph is built. This process continues until the instances of IoT applications of all users are deployed. Let \( L \) be the number of iterations of the proposed algorithm. Then, the solution delivered by the algorithm is \( \bigcup_{l=1}^{L} M_l \). Let \( c(M_l) \) be the weighted sum of edges in \( M_l \). Then, \( \sum_{l=1}^{L} c(M_l) \).

The heuristic algorithm is detailed in Algorithm 4.

**B. Heuristic Algorithm for the Minimization Problem**

In the following we devise an algorithm for the minimization problem. Following the spirit of Algorithm 3, we decompose the problem into two sub-problems: the update scheduling problem, and the IoT application placement problem. We obtain an optimal solution \( A_1 \) for the update scheduling problem by invoking Algorithm 1, and a feasible solution \( A'_2 \) to the IoT application placement problem by Algorithm 4, via adopting the update scheduling of sensors in \( A_1 \) and the IoT application placement of users in \( A'_2 \). The detailed algorithm is presented in Algorithm 5.

**Algorithm 5** An Algorithm for the Minimization Problem

**Require:** An MEC network \( G = (V,E) \), a set \( S \) of sensors, a positive integer \( K_s \) for each sensor \( s \in S \), a set \( U \) of users, and time horizon \( T \).

**Ensure:** Minimize \( \sum_{u \in U} \sum_{t \in T} W(u,t)/\sum_{u \in U} |T_u| \), i.e., the average weighted sum of AoIs of query results and query service delays of all queries over \( T \).

1. Formulate an updating scheduling problem and obtain its solution \( A_1 \) by Algorithm 1;
2. Formulate an IoT application placement problem and obtain its solution \( A'_2 \) by Algorithm 4;
3. A solution \( A' \) to the problem is obtained by adopting the update scheduling of sensors in \( A_1 \) and the IoT application placement of users in \( A'_2 \);
4. return Solution \( A' \).

**C. Algorithm Analysis**

**Theorem 5:** Given an MEC network \( G = (V,E) \), a set \( S \) of sensors, a positive integer \( K_s \) for each sensor \( s \in S \), a set \( U \) of users, and a finite time horizon \( T \), assuming that digital twins of sensors have already been deployed in cloudlets of \( V \), each user \( u \in U \) may retrieve data from digital twins of sensors in the beginning of time slot \( t \in T_u \). There is an algorithm, Algorithm 4, for the IoT application placement problem, which takes \( O(|U| \cdot (|U| + |V|)^3) \) time and delivers a feasible solution.

**Proof:** Algorithm 4 delivers a feasible solution for the IoT application placement problem, because there are no violations on any constraints. The time complexity of Algorithm 4 is analyzed as follows. It takes \( O(|U| + |V|)^3 \) time to find the minimum-cost maximum matching in the bipartite graph \( B^{(l)} \), and there are at most \( |U| \) iterations in the proposed algorithm, because at least an IoT application of one user is deployed in each iteration. Thus, the time complexity of Algorithm 4 is \( O(|U| \cdot (|U| + |V|)^3) \).

**Theorem 6:** Given an MEC network \( G = (V,E) \), a set \( S \) of sensors, a positive integer \( K_s \) for each sensor \( s \in S \), a set \( U \) of users with IoT application queries, and a finite time horizon \( T \), assuming that digital twins of sensors have already been deployed in cloudlets of \( V \), each user \( u \in U \) may retrieve data from digital twins of sensors at the beginning of time slot \( t \in T_u \). There is an algorithm, Algorithm 5, for the minimization problem, which takes \( O(|U| \cdot |T| + |S| \cdot K_s^{2 \max} \cdot |T|^2 + |U| \cdot (|U| + |V|)^3) \), due to the time complexities of Algorithm 1 and Algorithm 4 by Theorem 2 and Theorem 5, respectively.

**VI. PERFORMANCE EVALUATION**

In this section, we evaluated the algorithm performance via simulations.

**A. Experimental Environment Settings**

Consider an MEC network, in which the number of APs ranges from 50 to 250, and there is a co-located cloudlet with each AP. We leverage GT-ITM [5] to construct different topologies of MEC networks. The capacity on a cloudlet ranges from 10,000 MHz to 20,000 MHz [39], and bandwidth capacity on each AP is set within [5, 20] MHz [31]. The transmission delay of one unit data (one MB) along a link ranges from 0.2 ms to 1 ms [39]. There are 500 sensors, and each of them is in the proximity of an AP randomly with a distance from 10 meters to 50 meters. The number of updates \( K_s \) of each sensor \( s \) ranges from 10 to 30 for a given finite time horizon. The transmission power \( P_s \) of each sensor \( s \) ranges from 0.1 Watt to 0.5 Watt [4], while the path loss factor \( \alpha \) and the noise power \( \eta^2 \) are set as 4 [4] and \( 1 \times 10^{-10} \) Watt [19], respectively. There are 1,000 users, while the computing resource demanded by an IoT application instance or a digital twin is set within [300, 600] MHz [25]. The number of CPU cycles required for 1-bit task calculation is set within [200, 400] cycles/bit [23]. There are 100 time slots with each lasting 50 ms. Each user issues a query for a sensor randomly at each time slot. The volume of the update data of a sensor ranges from 1 MB to 5 MB [35], and the volume of data generated at a digital twin per update is set within [2, 10] MB. Parameter \( \beta \) in Eq. (4) is set as 0.5. The value in each figure represents the mean based on 30 MEC instances of the same size. The actual running time of each algorithm is obtained by a desktop with 3.60GHz Intel 8-Core i7 CPU and...
16GB RAM. Unless otherwise specified, these parameters are adopted by default.

We refer to Algorithm 3 and Algorithm 5 for the minimization problem as \text{Appr} and \text{Heu}, respectively, and we evaluated their performance against the following three benchmarks.

- Algorithm \text{Round} \_ \text{EUS}: It invokes a randomized rounding algorithm from [41] for IoT application placements, through the LP relaxation of the ILP (11). Each sensor adopts an Exact Uniform Scheduler (EUS) [10], i.e., each sensor delivers its updates evenly over the given time horizon. Each query is admitted through minimizing its weighted sum of the AoI of its query result and its query service delay, by determining whether to utilize the current data at $DT(s)$ or to wait for the next update of sensor $s$.

- Algorithm \text{Wait}: each user deploys its IoT application instance in a cloudlet such that the average transmission delay from the requested digital twin to the IoT application instance per query is minimized. Each sensor delivers its updates evenly over the given time horizon by adopting EUSs [10], and each query waits for the next update of its requested digital twin.

- Algorithm \text{NoWait}: similar to algorithm \text{Wait}, but queries retrieve the current data at their digital twins immediately.

- Algorithm \text{Random}: IoT application instances are deployed in cloudlets randomly. Sensors randomly send their updates to their digital twins, while queries can either retrieve the current data of digital twins or wait for their updates, and such actions are randomly chosen.

### B. Performance Evaluation of Different Algorithms

We first studied the performance of Algorithm \text{Appr} and Algorithm \text{Heu} against benchmarks \text{Round} \_ \text{EUS}, \text{Wait}, \text{NoWait} and \text{Random}, by varying the network size from 50 to 250, where the utilization ratio of cloudlet $v$ is the ratio of the amount of computing resource consumed to its capacity. Fig. 3 plots the performance (the optimization objective (5)) and running times of different algorithms, and the utilization ratios of cloudlets by algorithm \text{Appr}. Shown in Fig. 3(a), with the network size of 250, \text{Appr} outperforms \text{Heu} by 6.8%, while \text{Heu} outperforms \text{Round} \_ \text{EUS}, \text{Wait}, \text{NoWait} and \text{Random} by 8.4%, 12.9%, 18.2% and 32.4%, respectively. It can be seen from Fig. 3(b) that algorithm \text{Random} takes the least running time, because it makes decisions randomly. Meanwhile, algorithm \text{Heu} takes the longest running time, because it takes time to find a shortest path in the auxiliary graph for each sensor to schedule the updates of the sensor, and then to find a series of minimum-cost maximum matchings in corresponding bipartite graphs for IoT application instance placements. Fig. 3(c) shows that the computing capacity violation on any cloudlet by algorithm \text{Appr} is no greater than 4.1%. Fig. 3 demonstrates algorithms \text{Appr} and \text{Heu} outperform algorithms \text{Round} \_ \text{EUS}, \text{Wait}, \text{NoWait} and \text{Random}. The rationale behind is that both \text{Appr} and \text{Heu} can jointly optimize the freshness of query results and query service delays of all queries efficiently.

### C. Impact of Important Parameters on the Performance of Approximation Algorithm \text{Appr}

We then investigated the impact of the number of updates of each sensor on the performance of \text{Appr}, by varying the number of updates per sensor from 10 to 30. It can be seen from Fig. 4(a) that the performance of \text{Appr} with 30 updates within the given time horizon is 40.5% higher than that of itself with 10 updates in the same period when network size is set at 250. This indicates that the use of digital twins can deliver much fresher data through frequent updates of sensors. In Fig. 4(b), algorithm \text{Appr} with 30 updates takes the longest time because a large number of updates from a sensor leads to a large auxiliary graph for the updating scheduling, as shown in Section IV-A. We also evaluated the impact of parameter $\beta$ on the performance of \text{Appr}, where $\beta$ is a coefficient of the AoI of query results in Eq. (4). Fig. 5 plots the performance curves of \text{Appr}. When the network size is fixed at 250, the performance of \text{Appr} with $\beta = 0.9$ is 28.2% higher than...
When the network size reaches 250, Fig. 7(a) indicates that the curve of the performance of algorithm Heu shows that algorithm Heu with $|T| = 100$ outperforms itself with $|T| = 300$ by 53.2%, since sensors can provide fresher digital twin data for queries, through more frequent updates. Fig. 7(b) indicates that Heu with $|T| = 300$ takes the longest running time, due to that it takes much more time to find a shortest path in a larger auxiliary graph for each sensor.

**VII. Conclusion and Future Work**

In this paper, we investigated AoI-aware IoT query service provisioning in an MEC network empowered by the digital twin technology. We first formulated a novel minimization problem of jointly considering the accumulative freshness of query results and the total query service delay of admitted queries of IoT services, with the aim to minimize the weighted sum of Aos of query results and query service delays of admitted queries, and we showed the NP-hardness of the defined problem. We then devised a performance-guaranteed approximation algorithm for the problem, at the expense of bounded computing capacity violations. Also, We proposed an efficient heuristic for the problem without computing capacity violations. We finally evaluated the performance of the proposed algorithms via simulations. The simulation results indicated that the proposed algorithms are promising, and outperform the comparison baseline algorithms.

In our future work, we intend to establish an optimization framework for AoI-aware IoT query service provisioning in an MEC network empowered by digital twin technology, which enables to accurately predict network dynamics and provide real-time system information such as user query pattern changes and dynamic resource utilization, and so on. An efficient prediction mechanism based on deep reinforcement learning will be developed that will help to pre-allocate network resources for service provisions and improve the quality of services.

**ACKNOWLEDGMENT**

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

**REFERENCES**


IEEE/ACM TRANSACTIONS ON NETWORKING

X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia, “Multiple service model

R. Minerva, G. M. Lee, and N. Crespi, “Digital twin in the IoT context:

L. Ma, X. Wang, Z. Ning, S. Guo, M. Wen, and H. V. Poor, “Minimizing

J. Li et al., “AoI-aware user service satisfaction enhancement in digital

J. Li et al., “A الدول-aware user service satisfaction enhancement in digital


J. Li et al., “Wait for fresh data? Digital twin empowered IoT services in
edge computing,” in Proc. 20th Int. Conf. Mobile Ad Hoc Smart Syst. (MASS), Sep. 2023, pp. 397–405.

identification of multiple-source IoT applications in edge computing,”

J. Li et al., “Maximizing user service satisfaction for delay-sensitive IoT
applications in edge computing,” IEEE Trans. Parallel Distrib. Syst.,

maximization of delay-aware DNN inference in edge computing by
exploring DNN model partitioning and inference parallelism,” IEEE

J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya, “Digital twin-enabled
service satisfaction enhancement in edge computing,” in Proc. IEEE

J. Li, W. Lian, Z. Xu, Y. Zhang, and X. Jia, “Multiple service model
Services Comput., to be published, doi: 10.1109/TSC.2023.3341988.

Q. Liu, H. Zeng, and M. Chen, “Minimizing AoI with throughput
requirements in multi-path network communication,” IEEE/ACM Trans.

X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, “Stochastic digital-
twin service demand with edge response: An incentive-based congestion
control approach,” IEEE Trans. Mobile Comput., vol. 22, no. 4,

Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang,
“Communication-efficient federated learning and permissioned blockchain for
digital twin edge networks,” IEEE Internet Things J., vol. 8, no. 4,

Y. Ma, W. Lian, M. Huang, W. Xu, and S. Guo, “Virtual network
function service provisioning in MEC via trading off the usages between
computing and communication resources,” IEEE Trans. Cloud Comput.,

Y. Ma, W. Lian, J. Li, X. Jia, and S. Guo, “Mobility-aware and delay-
sensitive service provisioning in mobile edge-cloud networks,”

L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, and M. Huang,
“TCDA: Truthful combinatorial double auctions for mobile edge computing in
industrial Internet of Things,” IEEE Trans. Mobile Comput., vol. 21,
no. 11, pp. 4125–4138, Nov. 2022.

R. Minerva, G. M. Lee, and N. Crespi, “Digital twin in the IoT context:
A survey on technical features, scenarios, and architectural models,”

R. M. Nauss, “Solving the generalized assignment problem: An
optimizing and heuristic approach,” INFORMS J. Comput., vol. 15, no. 3,

D. B. Shmoys and É. Tardos, “An approximation algorithm for the
generalized assignment problem,” Math. Program., vol. 62, nos. 1–3,

W. Sun, H. Zhang, R. Wang, and Y. Zhang, “Reducing oflooding latency

S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, “Dynamic
scheduling for stochastic edge-cloud computing environments using

twin placement for minimum application request delay with data age
targets,” IEEE Internet Things J., vol. 10, no. 13, pp. 11547–11557,

management architecture for IoT devices,” IEEE Internet Things J.,

X. Wang, Z. Ning, S. Guo, M. Wen, and H. V. Poor, “Minimizing
the age-of-critical-information: An imitation learning-based scheduling
approach under partial observations,” IEEE Trans. Mobile Comput.,
vol. 21, no. 9, pp. 3225–3238, Sep. 2022.

D. Wu, X. Huang, X. Xie, X. Nie, L. Bao, and Z. Qin, “LEDGE:
Leveraging edge computing for resilient access management of mobile

Z. Xu et al., “Schedule or wait: Age-minimization for IoT big data

Z. Xu et al., “Learning-driven algorithms for responsive AR offloading with
non-deterministic rewards in metaverse-enabled MEC,” IEEE/ACM

Z. Xu et al., “Energy-aware collaborative service caching in a 5G-
enabled MEC with uncertain payoffs,” IEEE Trans. Commun.,

R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano,
and S. Ulukus, “Age of information: An introduction and survey,”
May 2021.

D. Zeng, H. Geng, L. Gu, and Z. Li, “Layered structure aware dependent
microservice placement toward cost efficient edge clouds,” in Proc. IEEE

federated learning service provisioning over mobile edge networks,”

mobile edge caching with freshness-aware content refreshing,”
IEEE Trans. Wireless Commun., vol. 20, no. 8, pp. 5329–5342,

X. Zhou, I. Koprulu, A. Erilimaz, and J. M. Neely, “Efficient distributed
MAC for dynamic demands: Congestion and age based designs,”
from IEEE/ACM conferences, journals, and technical committees. He has also served as the Chair of organizing and technical committees of many international conferences. He is the Editor-in-Chief of IEEE OPEN JOURNAL OF THE COMPUTER SOCIETY. He has served on the IEEE Communications Society Board of Governors; the IEEE Computer Society Fellow Evaluation Committee; and the Editorial Board of a number of prestigious international journals, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE INTERNET OF THINGS JOURNAL. He is an ACM Distinguished Member.

Weifa Liang (Senior Member, IEEE) received the B.Sc. degree from Wuhan University, China, in 1984, the M.E. degree from the University of Science and Technology of China in 1989, and the Ph.D. degree from The Australian National University in 1998, all in computer science. He is a Full Professor with the Department of Computer Science, City University of Hong Kong. Prior to joining City University of Hong Kong, he was a Full Professor with The Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, mobile edge computing (MEC), network function virtualization (NFV), the Internet of Things and digital twins, design and analysis of parallel and distributed algorithms, approximation algorithms, and graph theory. He currently serves as an Editor for IEEE TRANSACTIONS ON COMMUNICATIONS.

Jie Wu (Fellow, IEEE) received the Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, FL, USA, in 1989. He is the Director of the Center for Networked Computing; and a Laura H. Carnell Professor with Temple University. He is also the Director of International Affairs with the College of Science and Technology. He was the Chair of the Department of Computer and Information Sciences, from Summer 2009 to Summer 2016; and an Associate Vice Provost for International Affairs from Fall 2015 to Summer 2017. Prior to joining Temple University, he was the Program Director with the National Science Foundation and a Distinguished Professor with Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He is a Fellow of AAAS. He was a recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award. He was the General Co-Chair of IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016; and the Program Co-Chair of IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, an ACM Distinguished Speaker, and the Chair of the IEEE Technical Committee on Distributed Processing (TCDP). He serves on several editorial boards, including IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICE COMPUTING, Journal of Parallel and Distributed Computing, and Journal of Computer Science and Technology.

Quan Chen (Member, IEEE) received the B.S., master’s, and Ph.D. degrees from the School of Computer Science and Technology, Harbin Institute of Technology, China. He is currently an Associate Professor with the School of Computers, Guangdong University of Technology. He was a Post-Doctoral Research Fellow with the Department of Computer Science, Georgia State University. His research interests include routing and scheduling algorithms in wireless networks and sensor networks.

Zichuan Xu (Member, IEEE) received the B.Sc. and M.E. degrees from Dalian University of Technology, China, in 2008 and 2011, respectively, and the Ph.D. degree from The Australian National University in 2016, all in computer science. From 2016 to 2017, he was a Research Associate with the Department of Electronic and Electrical Engineering, University College London, U.K. He is currently a Full Professor and the Ph.D. Advisor with the School of Software, Dalian University of Technology. His research interests include mobile edge computing, serverless computing, network function virtualization, algorithmic game theory, and optimization problems.

Wenzheng Xu (Member, IEEE) received the B.Sc., M.E., and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2008, 2010, and 2015, respectively. He is currently an Associate Professor with Sichuan University, China. Also, he was a Visitor with The Australian National University, Australia; and The Chinese University of Hong Kong, Hong Kong. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory.

Jianping Wang (Fellow, IEEE) received the B.S. and M.S. degrees in computer science from Nankai University, Tianjin, China, in 1996 and 1999, respectively, and the Ph.D. degree in computer science from The University of Texas at Dallas in 2003. She is currently a Professor with the Department of Computer Science, City University of Hong Kong. Her research interests include cloud computing, service oriented networking, edge computing, and network performance analysis.