# Wait for Fresh Data? Digital Twin Empowered IoT Services in Edge Computing

Jing Li[†], Song Guo[†], Weifa Liang[‡], Jie Wu[¶], Quan Chen[§], Zichuan Xu[$], Wenzheng Xu[%], and Jianping Wang[‡]
[†] Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P. R. China
[‡] Department of Computer Science, City University of Hong Kong, Hong Kong, P. R. China
[¶] Department of Computer and Information Sciences, Temple University, Philadelphia, USA
[§] Guangdong University of Technology, Guangzhou, 510006, P. R. China
[$] School of Software, Dalian University of Technology, Dalian, 116621, P. R. China
[%] College of Computer Science, Sichuan University, Chengdu, Sichuan, 610065, P. R. China.

***Abstract*—The Mobile Edge Computing (MEC) paradigm gives impetus to the vigorous advancement of Internet of Things (IoT), through provisioning low-latency computing services at network edges. The emerging digital twin technique has grown in the community of IoT, and bridges the gap between physical objects and their digital representations in MEC, thereby enabling real-time data analysis, simulating the dynamics of systems, and optimizing network resource allocation. In this paper, we consider query services for various IoT applications in an MEC network, built upon digital twin data in the network, with the aim to optimize the freshness of query results, measured by the Age of Information (AoI) and query service delays simultaneously. We first formulate a novel minimization problem that explores a nontrivial trade-off between these two critical yet conflicted optimization objectives, and show the NP-hardness of the problem. We then propose an approximation algorithm for the problem with a provable approximation ratio, at the expense of a moderate computing resource violation. We finally evaluate the performance of the proposed algorithm via simulations. Simulation results demonstrate that the proposed algorithm is promising, and outperforms the benchmarks, improving by no less than $18.9\%$ of the performance in comparison with that of the baseline algorithms.**

## I. Introduction

The last decade has witnessed the unprecedented explosion of Internet of Things (IoT) applications by culminating the proliferation of IoT devices around us connected with the Internet, thereby permeating the modern-day world and flourishing the potential for brilliant living [17]. However, most IoT devices have limited resources and energy for running IoT applications on themselves, instead, they usually offload their computing tasks to remote clouds for processing which causes high service costs and service delays [6]. Moreover, in traditional IoT architectures, IoT devices store data in their backlogs for future diagnosis and improvements, which however may lead to stale feedback, unverified updates, and severe malfunctions [22].

Mobile Edge Computing (MEC) has been anticipated as a promising paradigm of paramount significance to supply computing resource (cloudlets) at network edges within the proximity of users to mitigate their service delays [2], [10], [11]. The emerging digital twin technique creates digital avatars for IoT devices in MEC networks catering to the massive data continuously generated from IoT devices, through leveraging integrated data and simulations to provide timely data analysis and modeling [13]. The marriage of the MEC and digital twin techniques drives new opportunities and challenges for IoT service provisioning by real-time monitoring, accurate prediction, and optimized decision-making, facilitating efficient network management and resource allocations [8], [14].

In this study, we consider query services of IoT applications in an MEC network, built upon the digital twin data of sensors for a finite time horizon, where sensors are scattered at diverse geographical locations to provide continuous sensory readings for time-varying environmental parameters. Because physic sensors have limited resources, the network service provider creates a digital twin for each sensor in a cloudlet for processing its generated data and simulating its behaviours [18]. To maintain the freshness of digital twin states, it is desirable that a sensor can upload newly collected data to its digital twin on time. However, the number of data uploading of each sensor usually is constrained because of the constrained energy and cost imposed on the sensor [1]. It thus poses a great challenge for a digital twin to instruct its sensor when performing data uploading to the digital twin, in order to provide the Age of Information (AoI)-aware IoT query services, where a common metric to measure the freshness of data is AoI [26], i.e., the amount of time between the current time and the data generation time.

The Quality of Services (QoS) of user queries based on digital twin data usually are measured by two metrics: the freshness of query results and query service delays [1], [20], [22]. Meeting QoS requirements of user queries for IoT applications in MEC poses the following challenges. On one hand, the placement of IoT application instances to cloudlets impacts both the freshness of query results and query service delays, e.g., a long distance between the cloudlet of an IoT application instance and a digital twin leads to a stale query result and a high query service delay. How to deploy IoT application instances of users to the MEC network to optimize these two metrics subject to computing capacities on cloudlets is challenging. On the other hand, it is challenging to determine whether to use the current data of a digital twin

with a lower query service delay or wait for its next update with a lower AoI. Also, to provide fresh data for queries, it is challenging to determine when scheduling data uploading of sensors and updating their digital twins over the finite time horizon, considering limited energy and cost budgets on sensors. In the rest of this paper, we will deal with AoI-aware query services for IoT applications built upon digital twin data in MEC, by addressing the aforementioned challenges.

The novelty of this study lies in exploring the power of digital twin technology, and its application for a minimization problem of jointly considering the freshness of query results and query service delays for IoT applications in an MEC network, with the aim to minimize the average weighted sum of the AoI of query results and query service delays of all queries for a given time horizon. An efficient approximation algorithm for the problem of concern is devised. To the best of our knowledge, we are the first to deal with digital twin state updating and AoI-aware query services built upon digital twin data in MEC networks.

The main contributions of this paper are as follows. We formulate a novel minimization problem of jointly considering the freshness of query results and query service delays for IoT service queries in an MEC network, and show the NP-hardness of the problem. We then develop an approximation algorithm with a provable approximation ratio for the problem with moderate resource violation. We finally evaluate the algorithm performance via simulations. Simulation results demonstrate that the proposed algorithm is promising, and outperforms the comparison baseline algorithms, improving the algorithm performance by no less than $18.9\%$ in comparison with that of baseline algorithms.

The rest of the paper is arranged as follows. The related work on digital twins in MEC is surveyed in Section II. Section III includes the system model and the problem definition. Section IV proposes an approximation algorithm for the problem of concern. The algorithm performance is evaluated in Section V. The conclusion is presented in Section VI.

## II. Related work

Plenty of works have been conducted in recent years to facilitate delay-sensitive IoT service provisioning in MEC platforms [4], [6], [9], [17], [21]. For example, Gedawy *et al.* [4] proposed heuristic algorithms to optimize the network throughput, as well as the energy consumption of IoT applications. Goudarzi *et al.* [6] developed an IoT application placement technique in MEC by the Memetic Algorithm (MA) to minimize the execution time and energy consumption. Ma *et al.* [17] considered truthfulness and budget-balance of IoT services, by designing a truthful combinatorial double auction mechanism. There are also extensive studies on optimizing the Age of Information (AoI) of IoT services in MEC [1], [23], [24], [27]. Corneo *et al.* [1] investigated the problem of efficient dissemination of sensor updates to optimize the AoI of IoT services. Wang *et al.* [23] devised an offline scheduling algorithm and an online learning algorithm for minimizing the average age of critical information. Zhang *et al.* [27] explored the trade-off between the AoI and service delay, and proposed an efficient algorithm to minimize the average service delay while meeting AoI requirements.

Recent emerging digital twin techniques empower MEC platforms to enable efficient services for various IoT applications [7], [8], [12]–[14], [18]. Li *et al.* [7], [8] estimated the reliability of virtual network functions by digital twins, and devised efficient algorithms to provide IoT services enabled by service function chains. Lin *et al.* [13] proposed an incentive-based congestion control scheme to meet the dynamic demands of digital twin services by Lyapunov optimization. Lu *et al.* [14] designed a federated learning algorithm based on the blockchain technique to improve security and data privacy in digital twin-assisted MEC networks. Sun *et al.* [20] utilized digital twins to minimize the offloading latency by Lyapunov optimization, considering user mobility and service migration.

In contrast to the aforementioned works, in this paper we study IoT service provisioning in MEC, empowered by digital twin technology. We focus on AoI-aware query services with the aim to minimize the average weighted sum of AoI of query results and query service delays for a given time horizon, by exploring the finest trade-off between the freshness of query results and query service delays.

## III. Preliminaries

### A. System model

Consider an MEC network modelled by an undirected graph $G = (V, E)$, with $V$ the set of Access Points (APs) and $E$ the set of links connecting APs. Each AP is co-located with a cloudlet by an optical fiber cable, and the communication delay between them is negligible [16]. We adopt notation $v \in V$ to represent either an AP or its co-located cloudlet for simplicity. Let $C_v$ be the computing capacity of cloudlet $v \in V$. Each link $e \in E$ is associated with a transmission delay $d_e$ to transmit a unit of data along the link [25].

Let $\mathbb{S}$ be a set of sensors deployed across diverse geographical locations. We assume that each sensor $s \in \mathbb{S}$ 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 the cloudlet co-located with AP $v_s$.

### B. User queries on digital twin data of sensors

We assume the MEC network runs in a discrete-time fashion, and a given monitoring time horizon is slotted into equal *time slots*. Denote by $\mathbb{T} = \{1, 2, \ldots, |\mathbb{T}|\}$ the set of time slots. There is a set $U$ of users with different IoT applications requesting data from digital twins of sensors. Assume each user $u \in U$ deploys an instance for his IoT application in a cloudlet that demands the amount $c_u$ of computing resource at the beginning of time horizon $\mathbb{T}$, and user $u$ issues queries (as his IoT application) for processing data from digital twins of different sensors at the beginning of different time slots. Denote by $\mathcal{T}_u \subseteq \mathbb{T}$ the set of time slots in which user $u$ will issue his queries of data retrieving and processing, i.e.,

at the beginning of each time slot $t \in \mathcal{T}_u \subseteq \mathbb{T}$, the deployed IoT application of user $u$ requests the data from the digital twin of a sensor $s_{u,t}$. For example, given $\mathbb{T} = \{1, 2, 3\}$, the IoT application of user $u$ requests data from digital twins of sensors $s_1$ and $s_2$ at the beginning of time slots 1 and 3, respectively, with $\mathcal{T}_u = \{1, 3\}$.

### C. Updating digital twins of sensors

Given the 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 the time horizon $\mathbb{T}$, and assume the number of updates of any sensor is not greater than the number of time slots of the given monitoring time horizon, i.e., $K_s \leq |\mathbb{T}|$, $\forall s \in \mathbb{S}$. For example, given $\mathbb{T} = \{1, 2, 3\}$ 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 [5], i.e., $\mu_s = B_s \cdot \log_2(1 + P_s/(dist_s^{\alpha} \cdot \eta^2))$, where $B_s$ is the bandwidth of AP $v_s$, $P_s$ is the transmission power of sensor $s$, $dist_s$ is the distance between sensor $s$ and AP $v_s$, $\eta^2$ is the noise power, and $\alpha$ is the path loss factor with $\alpha = 2$ or 4 for a short or long distance [2]. 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 $v_s$, therefore, the processing delay of $DT(s)$ is $\rho_s/f_s$.

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

$$t_s^{update} = \rho_s/\mu_s + \rho_s/f_s. \tag{1}$$

Suppose the current time slot is $t$. The current data of digital twin $DT(s)$ (or the received result of a user) is based on the received update from sensor $s$ generated at time slot $t_0$ with $t_0 \leq t$, *the Age of Information* (AoI) of this generated data is defined as $(t - t_0)$ [26]. It can be seen that $t_s^{update}$ is the minimum AoI of data at $DT(s)$. We assume that each $DT(s)$ with $s \in \mathbb{S}$ has generated initial data with AoI of $t_s^{update}$ at the beginning of time horizon $\mathbb{T}$. The AoI of data at $DT(s)$ will linearly increase until receiving an update from sensor $s$. Assuming sensor $s$ sends its first update to $DT(s)$ at time slot $t$, $DT(s)$ will generate the data that will be available for IoT applications at time $(t + t_s^{update})$, and the AoI of data at $DT(s)$ decreases to $t_s^{update}$ at time $(t + t_s^{update})$. The AoI of data of $DT(s)$ then linearly increases again until receiving the next update from $s$. This procedure continues until $\mathbb{T}$ ends.

### 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 the AoI 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 the beginning of each time slot $t \in \mathcal{T}_u \subseteq \mathbb{T}$. In the following, we omit the subscripts of $s_{u,t}$ for notation simplicity, i.e., replace $s_{u,t}$ with $s$. Recall that cloudlet $v_s$ 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 along the shortest path from cloudlet $v_s$ to cloudlet $v_u$ [25]. Denote by $\lambda_s$ the size of provided data at $DT(s)$, and the transmission delay of transmitting the data from $DT(s)$ to the IoT application instance of $u$ is $\lambda_s \cdot d_{s,u}$. Let $f_u$ be the processing rate of the IoT application instance of $u$. Then the processing delay of the IoT application of $u$ in $v_u$ is $\lambda_s/f_u$.

At the beginning of time slot $t \in \mathcal{T}_u$, the 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 fresh AoI, the user can retrieve the data of $DT(s)$ immediately; otherwise, the user can wait for a fresher AoI until the next update of $DT(s)$, 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 [20].

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 two cases as follows.

Case (i): 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 $v_s$ to cloudlet $v_u$ 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 (ii): User $u$ will retrieve the newly generated data of $DT(s)$ through waiting for its next update. Because the least AoI of the generated data at $DT(s)$ is $t_s^{update}$ by Eq. (1), the AoI of query result is $\lambda_s \cdot d_{s,u} + \lambda_s/f_u + t_s^{update}$. Suppose $t$ is the current time slot. Let $t'$ be the time slot of sending the next update of $DT(s)$. The IoT application of $u$ needs to wait for $(t' + t_s^{update} - t)$ time for the next update of $DT(s)$. The query service delay of user $u$ then is $\lambda_s \cdot d_{s,u} + \lambda_s/f_u + t' + t_s^{update} - t$.

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

$$W_{AoI}(u,t) = \begin{cases} \lambda_s \cdot d_{s,u} + \lambda_s/f_u + t - t_0, & \text{Case (i)} \\ \lambda_s \cdot d_{s,u} + \lambda_s/f_u + t_s^{update}, & \text{Case (ii)} \end{cases} \tag{2}$$

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

$$W_{delay}(u,t) = \begin{cases} \lambda_s \cdot d_{s,u} + \lambda_s/f_u, & \text{Case (i)} \\ \lambda_s \cdot d_{s,u} + \lambda_s/f_u + t' + t_s^{update} - t, & \text{Case (ii)} \end{cases} \tag{3}$$

Let $\beta$ be a constant with $0 \leq \beta \leq 1$, and we define the weighted sum $W(u, t)$ of the AoI of query result and query

service delay of a query issued by user $u$ at time $t$ as follows.

$$W(u,t) = \beta \cdot W_{AoI}(u,t) + (1-\beta) \cdot W_{delay}(u,t). \quad (4)$$

### E. Problem definition

*Definition 1:* Given an MEC network $G = (V, E)$, a set $\mathbb{S}$ of sensors, a positive integer $K_s$ for each sensor $s \in \mathbb{S}$, a set $U$ of users issuing queries for IoT applications on sensors, and a finite time horizon $\mathbb{T}$, assuming the digital twins of sensors have already been deployed in cloudlets $V$ of $G$, a user $u \in U$ may retrieve data from digital twins of sensors in $\mathbb{S}$ at the beginning of time slots in $\mathcal{T}_u \subseteq \mathbb{T}$. *The minimization problem of joint freshness of query results and query service delays of all queries* is to minimize the average weighted sum of the AoI of query results and query service delays of all queries for the given time horizon $\mathbb{T}$, i.e.,

$$\text{Minimize} \quad \sum\nolimits_{u \in U} \sum\nolimits_{t \in \mathcal{T}_u} W(u,t) / \sum\nolimits_{u \in U} |\mathcal{T}_u|, \quad (5)$$

by deploying IoT application instances of users in $U$ to cloudlets in $G$, subject to computing capacities on cloudlets.

### F. NP-hardness of the defined problem

*Theorem 1:* The minimization problem of joint freshness of query results and query service delays of all queries is NP-hard.

The detailed proof is omitted, due to space limitation.

## IV. Approximation Algorithm

In this section, we deal with the minimization problem of joint freshness of query results and query service delays of all queries in an MEC network, via devising an approximate solution for it as follows. We first decompose the problem into two sub-problems: the update scheduling problem and the IoT application placement problem. We then devise an approximation algorithm for the problem of concern by proposing an optimal solution to the first sub-problem and an approximation algorithm for the second sub-problem, respectively.

The optimization objective (5) is equivalent to minimizing the total weighted sum of the AoI of query results and query service delays of all queries over the time horizon, i.e.,

$$\text{Minimize} \quad \sum\nolimits_{u \in U} \sum\nolimits_{t \in \mathcal{T}_u} W(u,t). \quad (6)$$

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 $t_0 \leq t$, and we distinguish it into two cases: Case (i): There is no further update of sensor $s$ before the time horizon ends; and Case (ii): The next update of sensor $s$ is sent at time $t'$ with $t_0 < t'$ and $t < t' + t_s^{update}$.

To approach optimization objective (6), we define two functions $W_1(u,t)$ and $W_2(u,t)$ as follows.

$$W_1(u,t) = \begin{cases} \beta \cdot (t - t_0), & \text{if there is no further update} \\ \min\{\beta \cdot (t - t_0),\ t_s^{update} + (1-\beta) \cdot (t' - t)\}, & \\ & \text{if the next update is at time } t' \end{cases} \quad (7)$$

and

$$W_2(u,t) = \lambda_s \cdot d_{s,u} + \lambda_s / f_u. \quad (8)$$

We claim that (1) the value of $W_1(u,t)$ 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 (2) the value of $W_2(u,t)$ is determined by the IoT application placement of user $u$. We will rigorously show this claim in Lemma 2. We now define two sub-problems: the update scheduling problem and the IoT application placement problem, which correspond to the two aforementioned 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 $\mathbb{S}$ of sensors, a positive integer $K_s$ for each sensor $s \in \mathbb{S}$, a set $U$ of users with queries for IoT applications on sensors, and a finite time horizon $\mathbb{T}$, assuming digital twins of sensors have already been deployed in cloudlets $V$ of $G$, a user $u \in U$ may retrieve data from digital twins of sensors in $\mathbb{S}$ at the beginning of time slots in $\mathcal{T}_u \subseteq \mathbb{T}$. *The update scheduling problem* in $G$ is to

$$\text{Minimize} \quad \sum\nolimits_{u \in U} \sum\nolimits_{t \in \mathcal{T}_u} W_1(u,t), \quad (9)$$

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

*Definition 3:* Given 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 with IoT application queries, and a finite time horizon $\mathbb{T}$, assuming digital twins of sensors have already been deployed in cloudlets $V$ of $G$, a user $u \in U$ may retrieve data from digital twins of sensors at the beginning of time slots in $\mathcal{T}_u \subseteq \mathbb{T}$. *The IoT application placement problem* in $G$ is to

$$\text{Minimize} \quad \sum\nolimits_{u \in U} \sum\nolimits_{t \in \mathcal{T}_u} W_2(u,t), \quad (10)$$

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 in $G$.

We claim that the optimal value of 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.

### A. An optimal algorithm for the update scheduling problem

We here propose an optimal solution to the update scheduling problem. Because the update scheduling of sensor $s$ does not affect that of another sensor $s'$, we first focus on the update scheduling of sensor $s$ by proposing an optimal solution for it. We then extend this result to propose an optimal solution to the update scheduling problem for all sensors.

Denote by $Q_s$ the set of queries of users in $U$ to request data of digital twin $DT(s)$ of sensor $s$ over time horizon $\mathbb{T}$. Let $u_q \in U$ be the user who issues a query $q \in Q_s$ at time slot $t_q$ with $1 \leq t_q \leq |\mathbb{T}|$. We define four subsets of queries

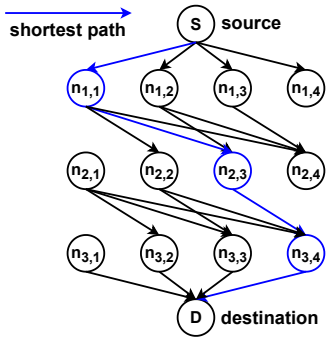



Fig. 1. An illustrative example of the auxiliary network $G'_s = (N'_s, E'_s)$ for sensor $s$, which can send $K_s$ (=3) updates over time horizon $\mathbb{T} = \{1, 2, 3, 4\}$.

of $Q_s$ as follows. (1) $Q_s(0, t_1) \subseteq Q_s$ is the set of queries issued earlier than $(t_1 + t_s^{update})$, where $t_s^{update}$ is the update delay of $DT(s)$ by Eq. (1), i.e., $1 \leq t_q < t_1 + t_s^{update}$ with $1 \leq t_1 \leq |\mathbb{T}|$; (2) $Q_s(t_1, t_2) \subseteq Q_s$ is the set of queries issued no earlier than $(t_1 + t_s^{update})$ but earlier than $(t_2 + t_s^{update})$, i.e., $t_1 + t_s^{update} \leq t_q < t_2 + t_s^{update}$ with $1 \leq t_1 < t_2 \leq |\mathbb{T}|$; (3) $Q_s(t_2, |\mathbb{T}| + 1) \subseteq Q_s$ is the set of queries issued no earlier than $(t_2 + t_s^{update})$, i.e., $t_2 + t_s^{update} \leq t_q \leq |\mathbb{T}|$ with $1 \leq t_2 \leq |\mathbb{T}|$; (4) Let $Q_s(0, |\mathbb{T}| + 1) = Q_s$.

Because the total queries of all users over the time horizon are the total queries requesting data of the digital twins of all sensors over the time horizon, we have $\sum_{s \in \mathbb{S}} \sum_{q \in Q_s} W_1(u_q, t_q) = \sum_{u \in U} \sum_{t \in \mathcal{T}_u} W_1(u, t)$.

To determine whether to use the current data at $DT(s)$ or wait for its next update, we construct an auxiliary graph $G'_s = (N'_s, E'_s)$ with edge weight $w : E'_s \mapsto \mathbb{R}^{\geq 0}$ for sensor $s \in \mathbb{S}$ as follows. The set $N'_s = \{S, D\} \cup \{n_{i,j} \mid 1 \leq i \leq K_s,\ 1 \leq j \leq |\mathbb{T}|\}$ of nodes and the set $E'_s = \{(S, n_{1,j}) \mid 1 \leq j \leq |\mathbb{T}|\} \cup \{(n_{K_s,j}, D) \mid 1 \leq j \leq |\mathbb{T}|\} \cup \{(n_{i,j}, n_{i+1,j'}) \mid 1 \leq i \leq K_s - 1,\ 1 \leq j < j' \leq |\mathbb{T}|\}$ of edges. Especially, we add virtual nodes $S$ and $D$ as the source and destination nodes, respectively. We also add nodes $n_{i,j}$ with $1 \leq i \leq K_s$ and $1 \leq j \leq |\mathbb{T}|$, i.e., the nodes are in a $K_s$-layer structure with $|\mathbb{T}|$ nodes in each layer. We then add edges $(S, n_{1,j})$ for each node $n_{1,j}$ with $1 \leq j \leq |\mathbb{T}|$ at the first layer, and edges $(n_{K_s,j}, D)$ for node $n_{K_s,j}$ with $1 \leq j \leq |\mathbb{T}|$ at the $K_s$th layer (the last layer). We add edges $(n_{i,j}, n_{i+1,j'})$ with $1 \leq i \leq K_s - 1$ and $1 \leq j < j' \leq |\mathbb{T}|$, i.e., for each $n_{i,j}$ at the $i$th layer, we add an edge $(n_{i,j}, n_{i+1,j'})$ for each $n_{i+1,j'}$ at the $(i+1)$th layer. The shortest path from source $S$ to destination $D$ will show the optimal update scheduling of sensor $s$.

The weight assignment of edges in $E'_s$ is given as follows. For each edge $(S, n_{1,j}) \in E'_s$ with $1 \leq j \leq |\mathbb{T}|$, its weight is $w(S, n_{1,j}) = \sum_{q \in Q_s(0,j)} W_1(u_q, t_q)$, with $W_1(u_q, t_q)$ defined in Eq. (7). If edge $(S, n_{1,j})$ is included in a shortest path in $G'_s$ 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 $(n_{i,j}, n_{i+1,j'}) \in E'_s$ with $1 \leq i \leq K_s - 1$ and $1 \leq j < j' \leq |\mathbb{T}|$, its weight is $w(n_{i,j}, n_{i+1,j'}) = \sum_{q \in Q_s(j,j')} W_1(u_q, t_q)$. Similarly, if edge $(n_{i,j}, n_{i+1,j'})$ is included in a shortest path in $G'_s$ from $S$ to $D$, the $i$th and $(i+1)$th updates are sent by sensor $s$ at the beginning of time slot $j$ and $j'$, respectively. For each edge $(n_{K_s,j}, D) \in E'_s$ with $1 \leq j \leq |\mathbb{T}|$, its weight is $w(n_{K_s,j}, D) = \sum_{q \in Q_s(j,|\mathbb{T}|+1)} W_1(u_q, t_q)$. If edge $(n_{K_s,j}, D)$ is included in a shortest path in $G'_s$ from $S$ to $D$, the $K_s$th update is sent by sensor $s$ at the beginning of time slot $j$.

We claim that the shortest path $P_s^*$ from $S$ to $D$ in $G'_s$ corresponds to the optimal solution to the update scheduling problem for sensor $s$, i.e., $\sum_{e \in P_s^*} w(e)$ is the minimum value of $\sum_{q \in Q_s} W_1(u_q, t_q)$, which will be shown later in Lemma 3. Hence, we first construct an auxiliary graph $G'_s$ for each sensor $s \in \mathbb{S}$, and find the shortest path $P_s^*$ in each $G'_s$, and $\sum_{s \in \mathbb{S}} \sum_{e \in P_s^*} w(e)$ is the minimum value of $\sum_{u \in U} \sum_{t \in \mathcal{T}_u} W_1(u, t)$. The found $P_s^*$, $\forall s \in \mathbb{S}$, thus is an optimal solution to the update scheduling problem. An illustrative construction of auxiliary graph $G'_s$ is given in Fig. 1, where sensor $s$ can have $K_s$ (=3) updates over the time horizon $\mathbb{T} = \{1, 2, 3, 4\}$. $P_s^* = \{(S, n_{1,1}), (n_{1,1}, n_{2,3}), (n_{2,3}, n_{3,4}), (n_{3,4}, D)\}$ is the shortest path in $G'_s$ from $S$ to $D$, which implies that the $K_s$ (=3) updates of sensor $s$ will be sent to its $DT(s)$ at 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

**Input:** 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 a finite time horizon $\mathbb{T}$.
**Output:** Minimize $\sum_{u \in U} \sum_{t \in \mathcal{T}_u} W_1(u, t)$ by scheduling the updates of sensors in $\mathbb{S}$ over the time horizon $|\mathbb{T}|$.
1: $\mathcal{A}_1 \leftarrow \emptyset$; /* the solution */
2: **for** each sensor $s \in \mathbb{S}$ **do**
3: Construct an auxiliary graph $G'_s = (N'_s, E'_s)$ for sensor $s$, and find the shortest path $P_s^*$ in $G'_s$ from source $S$ to destination $D$, which delivers the optimal solution $\mathcal{A}_{1,s}$ for updating scheduling of sensor$s$;
4: $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \{\mathcal{A}_{1,s}\}$;
5: **end for**
6: **return** Solution $\mathcal{A}_1$ to the problem.

### *B. An approximation algorithm for the IoT application placement problem*

We now deal with the IoT application placement problem. We first formulate an Integer Linear Programming (ILP) solution for the problem. We then propose an approximation algorithm with a provable approximation ratio for the problem, through reducing the problem to the minimum-cost Generalized Assignment Problem (GAP) [19]. An approximate solution to the minimum-cost GAP will in turn return an approximate solution to the IoT application placement problem.

Let $x_{u,v}$ be a binary variable with $x_{u,v} = 1$, indicating 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 $t$ if the IoT application instance of user $u$ is deployed in cloudlet $v \in V$. An ILP solution for the IoT application placement problem is provided as follows.

$$\text{Minimize} \quad \sum_{u \in U} \sum_{t \in \mathcal{T}_u} W_2(u, t) \tag{11}$$

subject to:

$$W_2(u, t) = \sum_{v \in V} (W'_2(u, t, v) \cdot x_{u,v}),\ \forall u \in U,\ \forall t \in \mathcal{T}_u \tag{12}$$

**Algorithm 2** An approximation algorithm for the IoT application placement problem

**Input:** 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 a finite time horizon $\mathbb{T}$.
**Output:** Minimize $\sum_{u \in U} \sum_{t \in \mathcal{T}_u} W_2(u,t)$ by deploying the IoT application instances of users in $U$ on cloudlets in $V$.
1: $\mathcal{A}_2 \leftarrow \emptyset$; /* the solution */
2: Find the shortest paths between each pair of cloudlets;
3: Solve the relaxed LP by ILP (11);
4: Obtain the optimal solution $\widetilde{OPT}$ to the LP, where $\tilde{x}_{u,v} \in [0,1]$ is the value of each $x_{u,v}$;
5: Build a bipartite graph $\mathcal{B} = (\mathcal{R}, \mathcal{V}; \mathcal{E})$ by [19].
6: Identify a minimum-cost matching $M$ in $\mathcal{B}$, where all nodes in $\mathcal{R}$ are matched, via Hungarian algorithm;
7: **for** each edge $(R_u, \gamma_{v,i}) \in M$ **do**
8: A solution $\mathcal{A}_{2,u}$ for user $u$ is obtained, by deploying the IoT application instance of user $u$ in cloudlet $v$;
9: $\mathcal{A}_2 \leftarrow \mathcal{A}_2 \cup \{\mathcal{A}_{2,u}\}$;
10: **end for**
11: **return** Solution $\mathcal{A}_2$ to the problem.

$$\sum_{u \in U} c_u \cdot x_{u,v} \leq C_v, \qquad \forall v \in V, \tag{13}$$

$$\sum_{v \in V} x_{u,v} = 1, \qquad \forall u \in U \tag{14}$$

$$x_{u,v} \in \{0,1\}, \qquad \forall u \in U, \ \forall v \in V. \tag{15}$$

Constraint (12) holds by the definition of $W_2'(u,t,v)$, and Constraint (13) means the computing capacity constraint on each cloudlet. Constraint (14) ensures that each user deploys an IoT application instance in a cloudlet.

We now reduce the IoT application placement problem to a minimum-cost GAP. Each IoT application instance of user $u \in U$ is regarded as an item $u$ with size $c_u$, while each cloudlet $v \in V$ is regarded as a bin $v$ possessing capacity $C_v$. Packing item $u$ to bin $v$ introduces a cost $\sum_{t \in \mathcal{T}_u} W_2'(u,t,v)$. By adopting the approximation technique [19] for the minimum cost GAP, we devise an approximation algorithm for the IoT application placement problem as follows.

We first obtain an optimal fractional solution $\widetilde{OPT}$ for the Linear Programming (LP) based on ILP (11). Let $\tilde{x}_{u,v} \in [0,1]$ be the fractional value of $x_{u,v}$ by $\widetilde{OPT}$. We build a bipartite graph $\mathcal{B} = (\mathcal{R}, \mathcal{V}; \mathcal{E})$ by [19]. Especially, $\mathcal{R}$ and $\mathcal{V}$ are two sets of nodes, while $\mathcal{E}$ is the set of edges connecting nodes in $\mathcal{R}$ and $\mathcal{V}$. Let $\mathcal{R} = \{R_u \mid u \in U\}$, where $R_u$ is a node corresponding to the IoT application instance of user $u \in U$, while $\mathcal{V} = \{\gamma_{v,i} \mid v \in V, 1 \leq i \leq \eta_v\}$, where $\eta_v = \lceil \sum_{R_u \in \mathcal{R}} \tilde{x}_{u,v} \rceil$ for each cloudlet $v \in V$.

Then we show how to construct the edge set $\mathcal{E}$. Considering each cloudlet $v \in V$ iteratively, each node in $\{\gamma_{v,i} \mid 1 \leq i \leq \eta_v\}$ is regarded as a bin possessing capacity 1. Each node $R_u \in \mathcal{R}$ with $\tilde{x}_{u,v} > 0$ is regarded as an item possessing size $\tilde{x}_{u,v}$. Sort nodes (items) in $\mathcal{R}$ in non-increasing order of $c_u$ with $u \in U$, and let $R_1 \geq R_2 \geq \ldots \geq R_{|U|}$ be the sorted items for notation simplicity. We now pack items into bins. The sorted items in $\mathcal{R}$ are packed into the first bin $\gamma_{v,1}$ one by one until it causes the capacity violation of $\gamma_{v,1}$ by packing $R_u$. Then we partition $R_u$ into two disjoint parts: $R_u^1$ and $R_u^2$, such that packing part $R_u^1$ into bin $\gamma_{v,1}$ makes $\gamma_{v,1}$ have no residual capacity. We pack part $R_u^2$ into the next bin $\gamma_{v,2}$. Such procedure continues until we have packed all items in $\mathcal{R}$. If we (partially) pack $R_u$ into bin $\gamma_{v,i}$, we add edge $(R_u, \gamma_{v,i})$ with cost $\sum_{t \in \mathcal{T}_u} W_2'(u,t,v)$ to set $\mathcal{E}$.

Finally we identify a minimum-cost maximum matching $M$ in the bipartite graph $\mathcal{B}$, which exactly matches all nodes in $\mathcal{R}$. 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 of joint freshness of query results and query service delays of all queries

**Input:** 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 a finite time horizon $\mathbb{T}$.
**Output:** Minimize $\sum_{u \in U} \sum_{t \in \mathcal{T}_u} W(u,t) / \sum_{u \in U} |\mathcal{T}_u|$, i.e., the average weighted sum of the AoI of query results and query service delays of all queries over $\mathbb{T}$.
1: Formulate an updating scheduling problem and solve it to obtain solution $\mathcal{A}_1$ by `Algorithm` 1.
2: Formulate an IoT application placement problem and solve it to obtain solution $\mathcal{A}_2$ by `Algorithm` 2;
3: A solution $\mathcal{A}$ to the problem of concern is obtained by adopting the update scheduling of sensors in $\mathcal{A}_1$ and the IoT application placement of users in $\mathcal{A}_2$;
4: **return** Solution $\mathcal{A}$ to the problem.

### *C. An approximation algorithm for the minimization problem of joint freshness of query results and query service delays of all queries*

We now devise an approximate algorithm for the minimization problem of joint freshness of query results and query service delays of all queries. The algorithm proceeds as follows. Given an instance of the problem, we decompose the problem into two sub-problems - the update scheduling problem and the IoT application placement problem. We first obtain an optimal solution $\mathcal{A}_1$ to the update scheduling problem by `Algorithm` 1. We then obtain an approximation solution $\mathcal{A}_2$ to the IoT application placement problem by `Algorithm` 2. We finally propose an approximation algorithm for the problem of concern, via adopting the update scheduling of sensors in $\mathcal{A}_1$ and the IoT application placement of users in $\mathcal{A}_2$. The algorithm is detailed in `Algorithm` 3.

### *D. Algorithm analysis*

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

Due to limited space, the proof of Lemma 1 is omitted.

*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, (1) 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 (2) the value of $W_2(u,t)$ by Eq. (8) is determined by the IoT application instance placement of user $u$.

**Proof** (1) Recall that $t_s^{update}$ 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.

(2) 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 $u$. As $f_u$ and $\lambda_s$ 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 $\mathbb{T}$, and the constructed auxiliary graph $G'_s = (N'_s, E'_s)$, the shortest path $P_s^*$ from source $S$ to destination $D$ in $G'_s$ corresponds to a feasible update scheduling of sensor $s$ over time horizon $\mathbb{T}$ with the minimum value of $\sum_{q \in Q_s} W_1(u_q, t_q)$ with $W_1(u_q, t_q)$ defined in Eq. (7).

Due to limited space, the proof of Lemma 3 is omitted.

*Theorem 2:* Given an MEC network $G = (V, E)$, a set $\mathbb{S}$ of sensors, a positive integer $K_s$ for each sensor $s \in \mathbb{S}$, a set $U$ of users with IoT application queries, and a finite time horizon $\mathbb{T}$, assuming that the digital twins of sensors have already been deployed in cloudlets $V$ of $G$, a user $u \in U$ may retrieve data from digital twins of sensors in $\mathbb{S}$ at the beginning of time slot $t \in \mathcal{T}_u \subseteq \mathbb{T}$. `Algorithm 1` delivers an optimal solution to the update scheduling problem, which takes $O(|U| \cdot |\mathbb{T}| + |\mathbb{S}| \cdot K_{max}^2 \cdot |\mathbb{T}|^2)$ time, with $K_{max} = \max\{K_s \mid s \in \mathbb{S}\}$.

**Proof** By Lemma 3, `Algorithm 1` delivers the optimal update scheduling of each $s \in \mathbb{S}$, and the update scheduling of sensor $s$ does not affect that of another sensor $s'$. Therefore, `Algorithm 1` delivers an optimal solution to the update scheduling problem, and its time complexity is analyzed as follows. For each $s \in \mathbb{S}$, we construct an auxiliary graph $G'_s = (N'_s, E'_s)$, where the cardinality of set $N'_s$ is $O(K_s \cdot |\mathbb{T}|)$. Finding a shortest path in $G'_s$ from $S$ to $D$ takes $O(K_s^2 \cdot |\mathbb{T}|^2)$ time. There are at most $|U| \cdot |\mathbb{T}|$ queries to be dealt with. Thus, `Algorithm 1` takes $O(|U| \cdot |\mathbb{T}| + |\mathbb{S}| \cdot K_{max}^2 \cdot |\mathbb{T}|^2)$ time. ■

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

**Proof** Because we reduce the IoT application placement problem to the minimum-cost GAP, and adopt the approximation technique in [19] to devise algorithm `Algorithm 2` for the problem, readers can refer to [19] for more details, which are omitted here due to space limitation. ■

*Lemma 4:* Let $OPT$ be the optimal solution to the problem of concern. Let $OPT_1$ and $OPT_2$ be the optimal solutions to the update scheduling problem and IoT application placement problem, respectively. We have $OPT = \frac{OPT_1 + OPT_2}{\sum_{u \in U} |\mathcal{T}_u|}$.

Due to limited space, the proof of Lemma 4 is omitted.

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

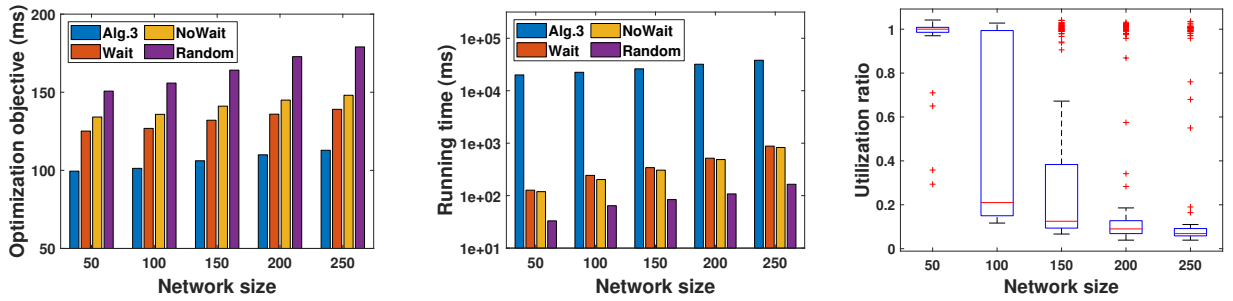
**Proof** Let $OPT$ be the optimal solution to the problem of concern. Denoted by $OPT_1$ and $OPT_2$ the optimal solutions to the update scheduling problem and IoT application placement problem, respectively. Let $\mathcal{A}$, $\mathcal{A}_1$, and $\mathcal{A}_2$ be the solution values by `Algorithm 3`, `Algorithm 1`, and `Algorithm 2`, respectively. We have $\mathcal{A}_1 = OPT_1$ by Theorem 2, $\mathcal{A}_2 \leq OPT_2$ by Theorem 3, and $OPT = \frac{OPT_1 + OPT_2}{\sum_{u \in U} |\mathcal{T}_u|}$ by Lemma 4. From Lemma 1, Eq. (7) and (8), we have $\mathcal{A} = \frac{\mathcal{A}_1 + \mathcal{A}_2}{\sum_{u \in U} |\mathcal{T}_u|} \leq \frac{OPT_1 + OPT_2}{\sum_{u \in U} |\mathcal{T}_u|} = OPT$ with the computing resource consumed of each cloudlet no more than twice its computing capacity, by Theorem 3.

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

## V. Performance Evaluation

### A. Experimental environment settings

We consider an MEC network $G = (V, E)$ consisting of from 50 to 250 APs, with each co-located with a cloudlet. We leverage GT-ITM [3] to construct the topologies of MEC networks. The bandwidth of each AP ranges from 5 MHz to 20 MHz [20], and the capacity on each cloudlet ranges from $10,000$ MHz to $20,000$ MHz [25]. The transmission delay of a unit of data (one MB) along a link ranges from 0.2 ms to 1 ms [25]. There are 500 sensors, and each sensor is in the




(a) The performance (b) The running times (c) Utilization ratios of cloudlets by Alg.3

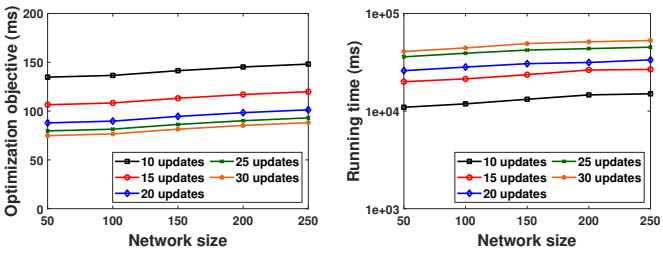
Fig. 2. Performance of different algorithms by varying the network size.

proximity of a random AP with a distance from 10 m to 50 m. The number of updates $K_s$ of a sensor $s$ ranges from 10 to 30. The transmission power $P_s$ of a sensor $s$ ranges from 0.1 Watt to 0.5 Watt [2], while the path loss factor $\alpha$ and the noise power $\eta^2$ are set as 4 [2] and $1 \times 10^{-10}$ Watt [9], 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 [15]. The number of CPU cycles needed for 1-bit task calculation is set within $[200, 400]$ cycles/bit [13]. There are 100 time slots, with each lasting 50 ms. At the beginning of each time slot, each user issues a query querying for a random sensor. The data size of the update of a sensor ranges from 1 MB to 5 MB [23], and the size of the data generated at a digital twin is set within $[2, 10]$ MB. The parameter $\beta$ in Eq. (4) is set as 0.5. The value in each figure represents the mean of the results based on 30 MEC instances of the same size. We obtain the actual running time of each algorithm by a desktop with a 3.60GHz Intel 8-Core i7 CPU and 16GB RAM. Unless otherwise specified, these parameters will be adopted by default.

We refer to `Algorithm` 3 as `Alg.3` proposed for the minimization problem of joint freshness of query results and query service delays of all queries, and evaluate its performance against three benchmarks: (1) `Wait`: each user deploys his 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 time horizon, and each query waits for the next update of the digital twin; (2) `NoWait`: similar to `Wait`, but queries retrieve current data at digital twins; and (3) `Random`: IoT application instances are deployed in cloudlets randomly. Sensors randomly send updates to 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. Algorithm performance evaluation*

We first evaluated the performance of algorithm `Alg.3` for the problem against algorithms `Wait`, `NoWait` and `Random`, by varying the network size from 50 to 250, where the utilization ratio of a cloudlet $v$ is the ratio of the amount of computing resource consumed to its computing capacity, in order to characterize the capacity violations by an algorithm. Fig. 2 plots the performance (the optimization objective (5)) and running times by different algorithms, as well as showing the utilization ratios of cloudlets by algorithm `Alg.3`. Shown in Fig. 2(a), when the network size is 250, `Alg.3` outperforms `Wait`, `NoWait` and `Random` by 18.9%, 23.8% and 36.9%, respectively. It can be seen from Fig. 2(b) that algorithm `Random` takes the least running time, because it randomly makes decisions. Meanwhile, `Alg.3` takes the longest running time, because of finding the shortest path in the auxiliary graph for each sensor and adopting the approximation algorithm from [19]. Fig. 2(c) shows that the computing capacity of each cloudlet is violated by no more than 4.1% by algorithm `Alg.3`. Therefore, Fig. 2 demonstrates `Alg.3` is promising, compared with `Wait`, `NoWait` and `Random`. The rationale behind this is that `Alg.3` efficiently optimizes the joint freshness of query results and query service delays of all queries.

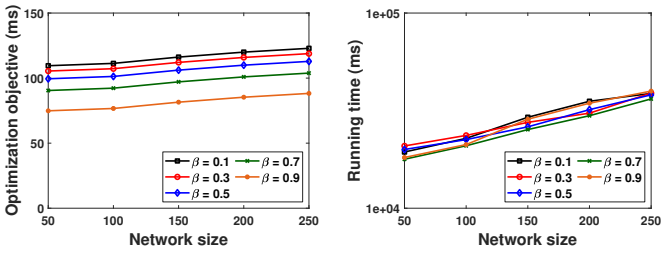



(a) The performance (b) The running times

Fig. 3. Impact of the number of updates of sensors on performance of `Alg.3`.

We then investigated the impact of the number of updates of each sensor on the performance of `Alg.3`. Fig. 3 shows the performance and running times of `Alg.3` with the number of updates of each sensor from 10 to 30. From Fig. 3(a), when the network size is 250, the performance of `Alg.3` with 30 updates is 40.5% higher than that of itself with 10 updates. The justification is that digital twins can obtain much fresher data through more updates from their sensors. In Fig. 3(b), algorithm `Alg.3` with 30 updates takes the most time because a larger number of updates from a sensor leads to a larger constructed auxiliary graph, as shown in Section IV-A.

We finally studied the impact of parameter $\beta$ on the performance of `Alg.3`, where $\beta$ is a coefficient associated with the AoI in Eq. (4). Fig. 4 plots the performance curves




(a) The performance (b) The running times

Fig. 4. Impact of the parameter $\beta$ on performance of Alg.3.

of Alg.3. When the network size is 250, the performance of Alg.3 with $\beta = 0.9$ is 28.2% higher than its performance with $\beta = 0.1$, as shown in Fig. 4(a). A larger value of $\beta$ represents that a larger weight is assigned to the average AoI of queries, and a less weight $(1 - \beta)$ is assigned to the average query service delay. It can also be seen from Fig. 4(b) that the impact of the value of $\beta$ on the running time of Alg.3 is negligible.

## VI. Conclusion

In this paper, we studied AoI-aware query services of IoT applications in MEC, empowered by the digital twin technology. We formulated a novel minimization problem of joint considerations of the freshness of query results and query service delays for IoT service queries, with the aim to minimize the average weighted sum of AoI of query results and query service delays of all queries for a given time horizon, and we showed the NP-hardness of the problem. We then devised a performance-guaranteed approximation algorithm for the problem, with moderate computing resource violations. We finally evaluated the algorithm performance via simulations. Simulation results demonstrate the proposed algorithm is promising, and outperforms the comparison baseline algorithms with performance improvement by no less than 18.9% in comparison with that of the baseline algorithms.

## References


[1] L. Corneo, C. Rohner, and P. Gunningberg. Age of information-aware scheduling for timely and scalable internet of things applications. *Proc. of INFOCOM'19*, IEEE, pp. 2476 – 2484, 2019.

[2] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795 – 2808, 2016.

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

[4] H. Gedawy, K. Habak, K. A. Harras, and M. Hamdi. RAMOS: a resource-aware multi-objective system for edge computing. *IEEE Transactions on Mobile Computing*, vol. 20, no. 8, pp. 2654 – 2670, 2021.

[5] A. Goldsmith. *Wireless communications*. Cambridge University Press, 2005.

[6] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya. An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1298 – 1311, 2021.

[7] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, and W. Xu. SFC-enabled reliable service provisioning in mobile edge computing via digital twins. *Proc of MASS'22*, IEEE, pp. 311 – 317, 2022.

[8] J. Li, S. Guo, W. Liang, Q. Chen, Z. Xu, W. Xu, and A. Y. Zomaya. Digital twin-assisted, SFC-enabled service provisioning in mobile edge computing. To appear in *IEEE Transactions on Mobile Computing*, 2022, doi: 10.1109/TMC.2022.3227248.

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

[10] J. Li, W. Liang, W. Xu, Z. Xu, Y. Li, and X. Jia. Service home identification of multiple-source IoT applications in edge computing. *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1417 – 1430, 2023.

[11] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, A. Zomaya, and S. Guo. Budget-aware user satisfaction maximization on service provisioning in mobile edge computing. To appear in *IEEE Transactions on Mobile Computing*, 2022, doi: 10.1109/TMC.2022.3205427

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

[13] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani. Stochastic digital-twin service demand with edge response: an incentive-based congestion control approach. *IEEE Transactions on Mobile Computing*, vol. 22, no. 4, pp. 2402 – 2416, 2023.

[14] 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 of Things Journal*, vol. 8, no. 4, pp. 2276 – 2288, 2021.

[15] Y. Ma, W. Liang, 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 Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2949 – 2963, 2022.

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

[17] 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 Transactions on Mobile Computing*, vol. 21, no. 11, pp. 4125 – 4138, 2022.

[18] R. Minerva, G. M. Lee, and N. Crespi. Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785 – 1824, 2020.

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

[20] W. Sun, H. Zhang, R. Wang, and Y. Zhang. Reducing offloading latency for digital twin edge networks in 6G. *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12240 – 12251, 2020.

[21] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya. Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks. *IEEE Transactions on Mobile Computing*, vol. 21, no. 3, pp. 940 – 954, 2022.

[22] C. Wang, Z. Cai, and Y. Li. Sustainable blockchain-based digital twin management architecture for IoT devices. *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6535 – 6548, 2023.

[23] X. Wang, Z. Ning, S. Guo, M. Wen, and V. Poor. Minimizing the age-of-critical-information: an imitation learning-based scheduling approach under partial observations. *IEEE Transactions on Mobile Computing*, vol. 21, no. 9, pp. 3225 – 3238, 2022.

[24] Z. Xu, W. Ren, W. Liang, W. Xu, Q. Xia, P. Zhou, and M. Li. Schedule or wait: age-minimization for IoT big data processing in MEC via online learning. *Proc. of INFOCOM'22*, IEEE, pp. 1809 – 1818, 2022.

[25] Z. Xu, L. Zhou, H. Dai, W. Liang, W. Zhou, P. Zhou, W. Xu, and G. Wu. Energy-aware collaborative service caching in a 5G-enabled MEC with uncertain payoffs. *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1058 – 1071, 2022.

[26] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus. Age of information: an introduction and survey. *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183 – 1210, 2021.

[27] S. Zhang, L. Wang, H. Luo, X. Ma, and S. Zhou. AoI-delay tradeoff in mobile edge caching with freshness-aware content refreshing. *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5329 – 5342, 2021.