Improving the Freshness of Digital Twins in Edge Computing

Jing Li[1], Jianping Wang[1(✉)], Weifa Liang[1], Sajal K. Das[2], and Quan Chen[3]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong, China
{jing.li,jianwang,weifa.liang}@cityu.edu.hk

[2] Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA
sdas@mst.edu

[3] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China
quan.c@gdut.edu.cn

Abstract. With the advent of Mobile Edge Computing (MEC) that shifts powerful computing resource from remote data centers to the network edge, Digital Twins (DTs) have emerged as a promising technology to provide comprehensive descriptions of physical objects in cyberspace with real-time interactions. Recent advancements of the Internet of Things (IoT) have contributed abundant and continuous data to the explosion of DTs, spurring the need to address the freshness of DTs through timely synchronizations. In this paper, we investigate innovative methodologies to improve the freshness of DTs while minimizing the cost of diverse resources consumed for improving freshness. Specifically, we first formulate a novel optimization problem: the DT freshness optimization problem. Next, we provide an Integer Linear Program (ILP) solution for the DT freshness optimization problem when the problem size is small; and devise a randomized algorithm at the expense of bounded resource violations, otherwise. We finally evaluate the performance of our algorithm through simulations. The simulation results show that our algorithm is promising.

Keywords: Digital Twin (DT) · DT synchronization · Mobile edge computing · Cost modelling · Online and approximation algorithms · Resource allocation

1 Introduction

Ushering into the era of the Internet of Things (IoT), big data in the physical world is proliferating, mainly due to the exponential growth of IoT devices and services [1]. It is anticipated that worldwide over 30 billion IoT devices will be in use by 2027 [2]. As a concept beyond reality, Digital Twins (DTs) are poised to offer a compelling technology capable of integrating big IoT data and achieving

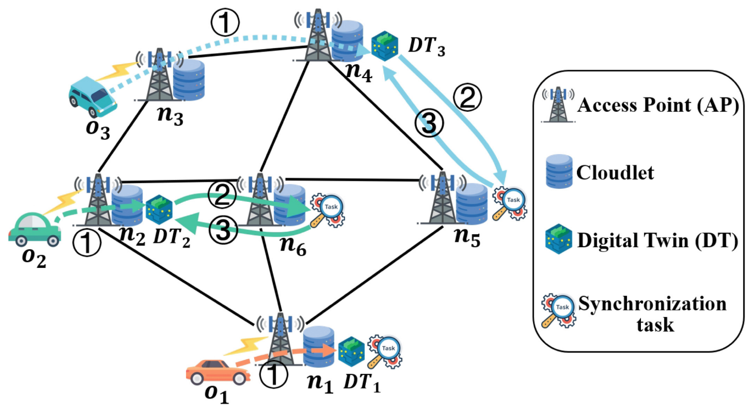


Fig. 1. A DT-assisted MEC network, where each Access Point (AP) has a co-located cloudlet. IoT devices (physical objects) o_1, o_2 and o_3 have their DTs (DT_1, DT_2 and DT_3) deployed in cloudlets n_1, n_2 and n_4, respectively. ① An object may generate and upload new data to the cloudlet holding its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

digital visualization [3,4]. Indeed, DTs bridge the physical and cyber dimensions by establishing high-fidelity virtual representations of physical objects in virtual worlds, which rely on processing real-time data from physical objects [5].

Conventional (remote) cloud platforms are not suitable for applications having real-time requirements in the continuous evolution of DTs, because clouds being usually located far away from physical objects, incur significant communication delays in addition to the possibility of being stale [6]. To this end, Mobile Edge Computing (MEC) shifts the cloud-like computing resource to the proximity of physical objects in the network edges [7,8]. Introducing the MEC paradigm to DTs will help improve the freshness of DTs through timely synchronizations, while optimizing physical operations by real-time feedback [9].

This paper explores the freshness improvement of DTs in MEC networks. As illustrated in Fig. 1, each physical object has a DT deployed in a cloudlet, and each object may upload new data to its DT by issuing a synchronization task. However, optimizing the freshness of DTs in an MEC network through DT synchronization scheduling poses the following significant challenges.

1) *How to determine where to perform a DT synchronization?* It is likely that a cloudlet has insufficient computing resource to synchronize all DTs it hosts, due to its limited capacity. To resolve this issue, a DT in the cloudlet can offload its synchronization to another cloudlet with sufficient computing resource by transmitting its new update data to the chosen cloudlet for processing, and then sending the result back to the original cloudlet [10]. How-

ever, such a synchronization offloading will inevitably lead to non-negligible overheads, thus necessitating to determine whether to conduct a synchronization of a DT in its hosting cloudlet or another cloudlet in the MEC network with sufficient computing resource.

2) *How to determine when to conduct a DT synchronization?* Considering the MEC system running in discrete time slots, we need to determine whether to conduct a synchronization of a DT in the current or a future time slot. Choosing to conduct the DT synchronizations in future time slots can be due to several reasons, e.g., there may be a large amount of new data generated from objects in the current time slot, whereas there is no sufficient computing resource within the MEC network to process all such new data. Therefore, the MEC network service provider may intend to postpone the synchronizations of some DTs. However, this may be at the expense of freshness loss of such DT synchronizations.
3) *How to jointly optimize the freshness and cost of DT synchronizations?* It is important to optimize the freshness of DT synchronizations to facilitate the quality enhancement of DT-enabled services, while optimizing the incurred cost of the network service provider.

The novelty of the work lies in jointly optimizing the freshness and cost of performing DT synchronizations for efficient DT service provisioning, and determining where and when to conduct each DT synchronization task in the MEC network. To this end, we formulate a novel DT freshness optimization problem, and solve it by developing a randomized approximation algorithm.

The main contributions of this paper are as follows.

- We formulate a DT freshness optimization problem, and prove the NP-hardness of the defined problem.
- We develop an Integer Linear Program (ILP) solution and a randomized approximation algorithm with bounded resource violations for the DT freshness optimization problem.
- We evaluate the performance of the proposed algorithm by simulations. Experimental results demonstrate that our algorithm is promising.

2 Related Work

DT-Assisted MEC: Empowered by the powerful computation, communication and storage in cyberspace, the DT technology offers to portray the physical world in a digital way, and has attracted much attention recently [2,3,7,11,12]. The authors in [3] designed a satellite-terrestrial scheme for cooperative edge computing. They leveraged a DT-based cloud to make decisions for resource sharing, while offering a Deep Reinforcement Learning (DRL) method to reduce the experienced system delay and the amount of satellite energy consumed. The authors in [11] explored the incentive mechanism to facilitate the trading between the DT service providers and network resource providers. They proposed a DRL-based methodology to maximize the revenue of DT service providers, while designing a semantic communication method to mitigate the data collection cost. Long-term performance optimization in DT-empowered MEC networks is suggested in [7], which adopted Lyapunov optimization, and proposed a long-term queue-aware scheme to minimize the energy consumption, by managing computing and communication resources. However, these studies do not consider the freshness improvement of DTs in MEC environments.

Freshness-Aware DT Service Provisioning: There exist several studies on improving the freshness of DTs [5,6,9,10,13–15]. Movable IoT devices are leveraged in [5] to collect the state data from physical objects via DT synchronizations. It models the freshness of a DT using a value metric, and provides an evolutionary game approach to optimize the cumulative value of all DTs. In [10], the data drift issue due to the system dynamics in edge environments is examined, by devising a continual learning-based scheme for continuous training of DT models to ensure that all DTs are as fresh as possible. The problem of placing DTs is investigated in [9], aiming to mitigate the response time of service requests, while optimizing the synchronization delay between DTs and other physical objects to meet the data age targets. However, these studies do not consider the joint optimization of freshness and cost of DT synchronizations, nor determine where and when to conduct each DT synchronization task in the MEC environments.

Different from the aforementioned works, we instead study how to improve the freshness of DTs in MEC, while mitigating the incurred cost. We tackle the DT freshness optimization problem via developing an efficient approximation algorithm.

3 Preliminaries

3.1 System Model

Consider an MEC network $G = (N, E)$, where there are a set N of Access Points (APs) and a set E of links interconnecting pairs of APs, with Fig. 1 illustrating the system model. Each AP has a co-located cloudlet with negligible communication delay. We assume an AP and its co-located cloudlet are interchangeable if no confusion arises. Let C_n denote the amount of available computing resource in each cloudlet $n \in N$.

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

3.2 Cost Model

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

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

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

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

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

3.3 Utility of Each DT Synchronization

Consider a synchronization task $q_{i,t}$ issued at time slot t while processed at time slot $\tau \geq t$, then the utility of the synchronization task $q_{i,t}$ is defined as

$$u(i,t,n,\tau) = w_1 \cdot \lambda_{i,t}^{\tau-t} - w_2 \cdot (c_{proc}(i,t,n) + c_{comm}(i,t,n)), \tag{3}$$

where $\lambda_{i,t}^{\tau-t}$ measures the freshness of DT synchronization by $q_{i,t}$, and the term $\lambda_{i,t}$ is a coefficient with $0 < \lambda_{i,t} < 1$ with $(\tau - t)$ the duration from the issuing to the processing time of $q_{i,t}$, similar to the definition of Age of Information (AoI) [16]. The freshness $\lambda_{i,t}^{\tau-t}$ decreases with the increasing value of τ. Meanwhile, $c_{proc}(i,t,n)$ and $c_{comm}(i,t,n)$ are the processing and communication costs according to Eq. (1) and Eq. (2), respectively; w_1 and w_2 are constant weights associated with the freshness and the cost, respectively.

3.4 Computing Resource Capacity of Cloudlets

We assume the consumed computing resource of a DT synchronization is proportional to the size of the uploaded data of its object [1,10], and the computing resource consumption of DT_i for synchronization with the uploaded update data of size $a_{i,t}$ at time slot t is $\sigma_i \cdot a_{i,t}$, where σ_i is a constant. Recalling each cloudlet $n \in N$ is associated with a computing resource capacity C_n, the resource consumed by each cloudlet must not exceed its capacity in any time slot.

3.5 Problem Definitions

In practice, physical objects may periodically send their update data to DTs in fixed intervals [6,9], and we assume that each DT synchronization task is given in advance with its issuing time and data size, which can be analyzed and predicted based on historical traces. The *DT freshness optimization problem* is defined as follows.

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

Let $x_{i,t,n,\tau}$ denote a binary decision variable, where $x_{i,t,n,\tau} = 1$ indicates that the synchronization task $q_{i,t}$ of DT_i is offloaded to the cloudlet n for processing at time slot $\tau \in \mathbb{T}$; otherwise $x_{i,t,n,\tau} = 0$. The Integer Linear Program (ILP) of the DT freshness optimization problem is formulated as:

$$\text{Maximize} \sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} u(i,t,n,\tau) \cdot x_{i,t,n,\tau}, \tag{4}$$

subject to:

$$\sum_{\tau=t}^{T} \sum_{n \in N} x_{i,t,n,\tau} \le 1, \qquad \forall t \in [1,T],\ \forall q_{i,t} \in Q_t \tag{5}$$

$$x_{i,t,n,\tau} = 0, \qquad \forall t \in [1,T],\ \forall q_{i,t} \in Q_t,\ \forall \tau \in [1, t-1],\ \forall n \in N \tag{6}$$

$$\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sigma_i \cdot a_{i,t} \cdot x_{i,t,n,\tau} \le C_n, \qquad \forall n \in N,\ \forall \tau \in [1,T] \tag{7}$$

$$x_{i,t,n,\tau} \in \{0,1\}, \qquad \forall t \in [1,T],\ \forall q_{i,t} \in Q_t,\ \forall \tau \in [1,T],\ \forall n \in N, \tag{8}$$

where the objective function (4) provides the accumulative collected utility from synchronization tasks defined by Eq. (3). Constraint (5) ensures that each synchronization task can be processed in at most one cloudlet; according to Constraint (6), each synchronization task of DTs is processed no earlier than its issuing time; and Constraint (7) ensures that the computing resource consumed at a cloudlet does not exceed its capacity.

Theorem 1. *The DT freshness optimization problem is NP-hard.*

Proof. The NP-hardness is proven by a reduction from a known NP-hard problem – the Generalized Assignment Problem (GAP) [17], where there is a set of bins and a set of items. Packing an item $item_i$ into a bin bin_j leads to a profit $profit_{i,j}$, while each item has a weight and each bin has a capacity. The GAP aims to collect as much profit as possible, respecting the capacities of bins.

We now look into a special case of the DT freshness optimization problem, where the time horizon has a single time slot. Namely, the synchronization tasks are issued by DTs at time slot 1 and can only be processed at time slot 1. Each cloudlet $n \in N$ is treated as a bin with a capacity C_n, and each synchronization task $q_{i,1}$ of DTs is treated as an item with the weight $\sigma_i \cdot a_{i,1}$ (its consumed computing resource). If $q_{i,1}$ is executed in a cloudlet $n \in N$, then a profit $u(i, 1, n, 1)$ is collected according to Eq.(3). This special case of the DT freshness optimization problem aims to collect as much profit as possible via an efficient assignment of synchronization tasks to cloudlets to improve DT freshness.

Observe that this special case of the DT freshness optimization problem is equivalent to GAP [17], implying that the DT freshness optimization problem is NP-hard. ■

4 Randomized Algorithm for the DT Freshness Optimization Problem

4.1 Randomized Algorithm

There is an optimal solution for the DT freshness optimization problem by formulating the problem as an ILP (4). The solution however is not scalable and takes unacceptable running time when the problem size is large, and this work here proposes a randomized approximation algorithm for the DT freshness optimization problem within a reasonable amount of time as follows.

We first perform a relaxation operation on binary variables $x_{i,t,n,\tau}$, and denote by $\widetilde{x}_{i,t,n,\tau}$ the relaxed variables, which are supposed to be real numbers within $[0, 1]$. In other words, we have

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

The Linear Program (LP) in (4) delivers an optimal fractional solution for the DT freshness optimization problem. We then perform randomized rounding [18] on the LP to find an integral solution. This means that we set the value of $x_{i,t,n,\tau}$ as 1 with the probability $\widetilde{x}_{i,t,n,\tau}$; otherwise the value of $x_{i,t,n,\tau}$ is 0. Notice that we conduct the randomized rounding operation in an exclusive way, with regard to Constraint (5), i.e., we conduct the randomized rounding for each synchronization task independently, and at most one $x_{i,t,n,\tau}$, $\forall t \in [1, T]$, $\forall q_{i,t} \in Q_t$, is set as 1, and the rest are set as 0s. `Algorithm` 1 describes the proposed randomized approximation algorithm.

Algorithm 1. Randomized approximation algorithm for the DT freshness optimization problem

```
Input: An MEC network $G = (N, E)$, a set $O$ of physical objects with their DTs placed in cloudlets, and a set $Q_t$ of synchronization tasks issued at each time slot $t$ for a given time horizon $\mathbb{T}$.
Output: Improve the freshness of DT.
1: Relax the ILP (4) to obtain the LP formulation;
2: Solve the LP to get an optimal fractional solution, where $\tilde{x}_{i,t,n,\tau} \in [0, 1]$ is a real number;
3: Perform randomized rounding [18] on the LP solution exclusively by Constraint (5). Namely, set the integral value $\hat{x}_{i,t,n,\tau}$ of $x_{i,t,n,\tau}$ as 1 with probability $\tilde{x}_{i,t,n,\tau}$; otherwise set $\hat{x}_{i,t,n,\tau}$ as 0, and each synchronization task is assigned to at most one cloudlet;
4: Return An integral solution.
```

4.2 Algorithm Analysis

We analyze the performance of `Algorithm` 1, and define ρ as follows.

$$\rho = \max\{u(i, t, n, \tau),\ \sigma_i \cdot a_{i,t} \mid \forall t \in [1, T], \forall q_{i,t} \in Q_t, \forall \tau \in [t, T], \forall n \in N\}, \tag{10}$$

where $\max\{u(i, t, n, \tau) \mid \forall t \in [1, T], \forall q_{i,t} \in Q_t, \forall \tau \in [t, T], \forall n \in N\}$ is the maximum utility obtained by any synchronization task, and $\max\{\sigma_i \cdot a_{i,t} \mid \forall t \in [1, T],\ \forall q_{i,t} \in Q_t\}$ is the maximum consumed resource of any task.

Lemma 1. *The amount of utilized computing resource at any cloudlet by the proposed* `Algorithm` *1 is no greater than twice of the computing capacity of the cloudlet, with high probability* $(1 - \frac{1}{|N|})$*, provided that* $\min\{C_n \mid n \in N\} \geq 6\rho \ln |N|$*, where* ρ *is defined in Eq.* (10).

The detailed proof is omitted, due to limited space.

Theorem 2. *Given an MEC network* $G = (N, E)$*, a set* O *of physical objects with DTs placed in cloudlets, and a set* Q_t *of synchronization tasks issued in each time slot* t *given in advance,* `Algorithm` *1 is a randomized approximation algorithm for the DT freshness optimization problem with an approximation ratio of* $\frac{1}{2}$*, and the utilized computing resource in a cloudlet is no greater than twice of the computing capacity, with high probability* $\min\{1 - \frac{1}{\sum_{t=1}^{T} |Q_t|}, 1 - \frac{1}{|N|}\}$*, provided that* $\widetilde{S} \geq 8\rho \ln(\sum_{t=1}^{T} |Q_t|)$ *and* $\min\{C_n \mid n \in N\} \geq 6\rho \ln |N|$*, where* $\widetilde{S}$ *is the optimal fractional solution by* (4) *and* ρ *is defined in Eq.* (10).

Proof. Suppose that S is an optimal solution for the DT freshness optimization problem, delivered by the ILP (4). Suppose that $\widetilde{S}$ is its optimal fractional solution based on the LP. We observe that $\widetilde{S} \geq S$, because we aim to collect as much utility as possible. Recall that $\tilde{x}_{i,t,n,\tau} \in [0, 1]$ is the relaxed variables, and we treat $\tilde{x}_{i,t,n,\tau}$ as the probability to set the value of $\hat{x}_{i,t,n,\tau}$ as 1.

Suppose that $y_{i,t,n,\tau} = \frac{u(i,t,n,\tau)}{\rho} \cdot \hat{x}_{i,t,n,\tau}$ is a random variable. We can see $y_{i,t,n,\tau}$ is $\frac{u(i,t,n,\tau)}{\rho}$ with probability $\tilde{x}_{i,t,n,\tau}$; otherwise $y_{i,t,n,\tau}$ is 0, and we have $0 \leq y_{i,t,n,\tau} \leq 1$, by $\frac{u(i,t,n,\tau)}{\rho} \cdot \hat{x}_{i,t,n,\tau} \leq \frac{u(i,t,n,\tau)}{\max\{u(i,t,n,\tau) \mid \forall t \in [1,T], \forall q_{i,t} \in Q_t, \forall \tau \in [t,T], \forall n \in N\}} \leq 1$.

Because the expected value of $y_{i,t,n,\tau}$ is $\mathbb{E}[y_{i,t,n,\tau}] = \frac{u(i,t,n,\tau)}{\rho} \cdot \tilde{x}_{i,t,n,\tau}$, we have

$$\mathbb{E}[\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} y_{i,t,n,\tau}] = \sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} \frac{u(i,t,n,\tau) \cdot \tilde{x}_{i,t,n,\tau}}{\rho} = \frac{\widetilde{S}}{\rho}. \quad (11)$$

Let μ_1 be a constant with $0 < \mu_1 < 1$, and let $Y_{i,t} = \sum_{\tau=t}^{T} \sum_{n \in N} y_{i,t,n,\tau}$. With regard to the Chernoff Bound [19],

$$\begin{aligned}
&\mathbf{Pr}[\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} u(i,t,n,\tau) \cdot \hat{x}_{i,t,n,\tau} \leq (1-\mu_1) \cdot S] \\
\leq &\mathbf{Pr}[\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} u(i,t,n,\tau) \cdot \hat{x}_{i,t,n,\tau} \leq (1-\mu_1) \cdot \widetilde{S}], \text{ because } \widetilde{S} \geq S \\
= &\mathbf{Pr}[\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} \sum_{\tau=t}^{T} \sum_{n \in N} y_{i,t,n,\tau} \leq (1-\mu_1) \cdot \frac{\widetilde{S}}{\rho}], \text{ by} y_{i,t,n,\tau} = \frac{u(i,t,n,\tau)}{\rho} \cdot \hat{x}_{i,t,n,\tau} \\
= &\mathbf{Pr}[\sum_{t=1}^{T} \sum_{q_{i,t} \in Q_t} Y_{i,t} \leq (1-\mu_1) \cdot \frac{\widetilde{S}}{\rho}] \leq exp(\frac{-\mu_1^2 \cdot \widetilde{S}}{2\rho}), \quad (12)
\end{aligned}$$

where Inequality (12) holds by the Chernoff bound due to the fact that $\{Y_{i,t} \mid \forall t \in [1,T],\ \forall q_{i,t} \in Q_t\}$ are independent random variables.

Supposing $exp(\frac{-\mu_1^2 \cdot \widetilde{S}}{2\rho}) \leq \frac{1}{\sum_{t=1}^{T} |Q_t|}$ with $0 < \mu_1 < 1$, we then have $\mu_1 \geq \sqrt{\frac{2\rho \ln(\sum_{t=1}^{T} |Q_t|)}{\widetilde{S}}}$. When $\frac{1}{2} \leq \mu_1 < 1$, we have $\sqrt{\frac{2\rho \ln(\sum_{t=1}^{T} |Q_t|)}{\widetilde{S}}} \leq \frac{1}{2}$. This means $\widetilde{S} \geq 8\rho \ln(\sum_{t=1}^{T} |Q_t|)$.

Therefore, `Algorithm` 1 has an approximation ratio of $\frac{1}{2}$, due to $1 - \mu_1 \leq \frac{1}{2}$, with high probability $1 - \frac{1}{\sum_{t=1}^{T} |Q_t|}$, provided by $\widetilde{S} \geq 8\rho \ln(\sum_{t=1}^{T} |Q_t|)$.

Referring to Lemma 1, `Algorithm` 1 solves the DT freshness optimization problem with an approximation ratio of $\frac{1}{2}$, and the amount of utilized computing resource of a cloudlet is no greater than twice of the computing capacity, with high probability $\min\{1 - \frac{1}{\sum_{t=1}^{T} |Q_t|}, 1 - \frac{1}{|N|}\}$, provided that $\widetilde{S} \geq 8\rho \ln(\sum_{t=1}^{T} |Q_t|)$ and $\min\{C_n \mid n \in N\} \geq 6\rho \ln |N|$. ■

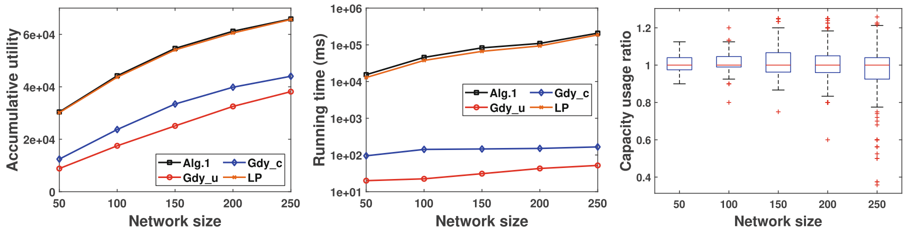
5 Performance Evaluation

5.1 Experimental Settings

Consider an MEC network with the number of APs (and their co-located cloudlets) ranging from 50 to 250. The topology of each network is generated by the GT-ITM tool [20]. There exist $1,000$ physical objects (IoT devices) with their DTs placed in cloudlets. The computing capacity on a cloudlet is drawn between $3,000$ MHz to $6,000$ MHz [21]. The monitoring period consists of 10 time slots, and the number of synchronization tasks at each time slot is randomly drawn from 600 and $1,000$. Each synchronization task is issued by a randomly chosen object, and the size of the update data is drawn from $[10, 50]$ MB [9]. The amount of computing resource consumed for processing 1MB data is drawn from 30MHz to 60MHz, and the size of the processed data is within $[2, 10]$ MB. The cost of data processing in a cloudlet is set within $[0.01, 0.1]\$$ per MB, and the cost of data transferring along a link is set within $[0.01, 0.1]\$$ per MB [21]. With regard to the utility definition (3), the value of parameter $\lambda_{i,t}$ is within $[0.5, 0.9]$, and we set weights w_1 and w_2 as 10 and 1, respectively.

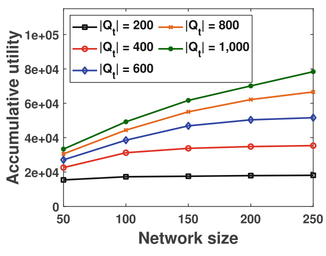
We evaluated the performance of `Algorithm 1` (refered to as `Algorithm 1`) against the following benchmarks.

- `Gdy_u`: At each time slot, it considers the synchronization tasks from DTs that have not been processed one by one, and it keeps assigning the synchronization task to the cloudlet with the largest utility greedily.
- `Gdy_c`: At each time slot, it considers cloudlets one by one. For each cloudlet, it iteratively selects a synchronization task with the largest utility greedily.
- `LP`: It is a solution by Linear Program (4) with the relaxed variables in real numbers. It provides an upper bound on the optimal solutions to the DT freshness optimization problem.

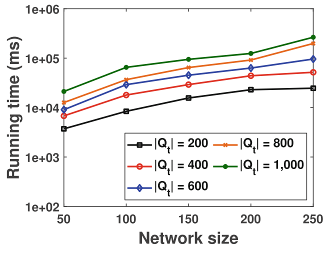


(a) Accumulative utility (b) Running time (c) Capacity utilization ratio of cloudlets by `Alg.1`

Fig. 2. Performance of different algorithms.



(a) Accumulative utility



(b) Running time

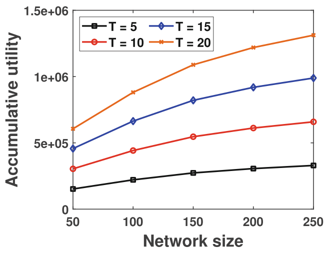
Fig. 3. Impact of the number $|Q_t|$ of synchronization tasks in each time slot on `Algorithm` 1.

5.2 Algorithmic Performance

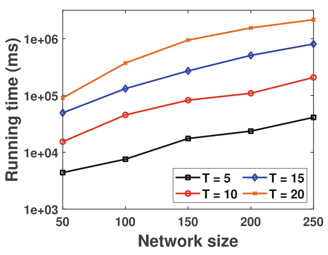
We first evaluated the performance of `Algorithm` 1 and against benchmarks `Gdy_u`, `Gdy_c` and `LP`, with the varying number of cloudlets from 100 to 1, 000. Figure 2 depicts their algorithm performance, and we adopt the *capacity utilization ratio of a cloudlet* to characterize its computing capacity violation, i.e., the ratio of the utilized computing resource in a cloudlet to its computing capacity.

Figure 2(a) demonstrates that the performance of `Algorithm` 1 is near-optimal for the DT freshness optimization problem, with regard to the performance of `LP`. Especially, the accumulative utility by `Gdy_u` and `Gdy_c` are 57.9% and 66.8% of that by `Algorithm` 1, respectively, for the network size of 250. This is because `Algorithm` 1 leverages the randomized rounding technique to deliver an efficient solution. The running time of different algorithms in Fig. 2(b) shows that `Algorithm` 1 takes the longest running time, due to obtaining the LP solution from ILP formulation (4). Figure 2(c) shows the capacity utilization ratio of a cloudlet is no more than 124.8%, i.e., the cloudlet's computing capacity is violated by no greater than 24.8%, therefore, the capacity violations on cloudlets are moderate.

We then considered the impact of the number $|Q_t|$ of synchronization tasks issued in each time slot on the performance of `Algorithm` 1 with $|Q_t| = 200$, 400, 600, 800 and 1, 000, respectively. From Fig. 3a, we can observe that when there are 250 cloudlets, the accumulative utility of `Algorithm` 1 with $|Q_t| = 200$ is 23.1% of that by itself with $|Q_t| = 1,000$. The justification is that `Algorithm` 1 can collect more utility gain with a larger number of issued synchronization tasks. Figure 3b illustrates that the running time of `Algorithm` 1 with $|Q_t| = 1,000$ is the longest, because solving the LP with a larger number of synchronization tasks takes more time.



(a) Accumulative utility



(b) Running time

Fig. 4. Impact of the length T of the time horizon on `Algorithm` 1.

We finally paid attention to the impact of the length T of the time horizon on the performance of `Algorithm` 1, supposing there are 5, 10, 15 and 20 time slots, respectively. Considering the network size of 250, Fig. 4(a) shows the utility of `Algorithm` 1 with $T = 5$ is 25.1% of that by itself with $T = 20$. The reason is that `Algorithm` 1 determines where and when to process each synchronization task during the time horizon in an efficient way, and more utility gains are earned with a longer time horizon. Observed from Fig. 4(b), a larger number of time slots leads to a longer running time of `Algorithm` 1, due to examining more time slots.

6 Conclusion

In this paper, we investigated the DT service provisioning in MEC networks by addressing the DT synchronization issue to improve DT freshness, with the aim to jointly optimize DT freshness through synchronizing with their objects while minimizing the cost of various resource consumptions on the synchronizations. To this end, we formulated a novel DT freshness optimization problem, and developed a randomized approximation algorithm with bounded resource violations for it. Finally, we evaluated the performance of our algorithm by simulations, and results show that our algorithm is promising.

Acknowledgement. The work was supported by Hong Kong Research Grants Council (RGC) under the Collaborative Research Fund (CRF) grant C1042-23GF. The work by Jing Li was supported by the Post-Doc Fellowship Award from the Hong Kong RGC under Project No. CityU PDFS2425-1S02.

References

1. Xu, Z., et al.: Schedule or wait: age-minimization for IoT big data processing in MEC via online learning. In: Proceedings of INFOCOM, pp. 1809–1818 (2022)
2. Zhang, Y., Hu, J., Min, G.: Digital twin-driven intelligent task offloading for collaborative mobile edge computing. IEEE J. Sel. Areas Commun. **41**(10), 3034–3045 (2023)

3. Ji, Z., Wu, S., Jiang, C.: Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks. IEEE J. Sel. Areas Commun. **41**(11), 3414–3429 (2023)
4. Wang, Z., Jiang, D., Mumtaz, S.: Network-wide data collection based on in-band network telemetry for digital twin networks. IEEE Trans. Mob. Comput. **24**(1), 86–101 (2025)
5. Han, Y., et al.: A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse. IEEE Internet Things J. **10**(1), 268–284 (2022)
6. Li, J., et al.: AoI-aware user service satisfaction enhancement in digital twin-empowered edge computing. IEEE/ACM Trans. Netw. **32**(2), 1677–1690 (2024)
7. Peng, Y., Duan, J., Zhang, J., Li, W., Liu, Y., Jiang, F.: Stochastic long-term energy optimization in digital twin-assisted heterogeneous edge networks. IEEE J. Sel. Areas Commun. **42**(11), 3157–3171 (2024)
8. Li, J., Liang, W., Li, Y., Xu, Z., Jia, X., Guo, S.: Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism. IEEE Trans. Mob. Comput. **22**(5), 3017–3030 (2021)
9. Vaezi, M., Noroozi, K., Todd, T.D., Zhao, D., Karakostas, G.: Digital twin placement for minimum application request delay with data age targets. IEEE Internet Things J. **10**(13), 11547–11557 (2023)
10. Li, J., et al.: Digital twin-enabled service provisioning in edge computing via continual learning. IEEE Trans. Mob. Comput. **23**(6), 7335–7350 (2024)
11. Luong, N.C., Le Van, T., Feng, S., Du, H., Niyato, D., Kim, D.I.: Edge computing for metaverse: incentive mechanism versus semantic communication. IEEE Trans. Mob. Comput. **23**(5), 6196–6211 (2024)
12. Li, J., et al.: Mobility-aware utility maximization in digital twin-enabled serverless edge computing. IEEE Trans. Comput. **73**(7), 1837–1851 (2024)
13. Li, J., et al.: AoI-aware service provisioning in edge computing for digital twin network slicing requests. IEEE Trans. Mob. Comput. **23**(12), 14607–14621 (2024)
14. Li, J., Wang, J., Chen, Q., Li, Y., Zomaya, A.Y.: Digital twin-enabled service satisfaction enhancement in edge computing. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications, pp. 1–10. IEEE (2023)
15. Li, J., et al.: AoI-aware, digital twin-empowered IoT query services in mobile edge computing. IEEE/ACM Trans. Netw. (2024)
16. Yates, R.D., Sun, Y., Brown, D.R., Kaul, S.K., Modiano, E., Ulukus, S.: Age of information: an introduction and survey. IEEE J. Sel. Areas Commun. **39**(5), 1183–1210 (2021)
17. Öncan, T.: A survey of the generalized assignment problem and its applications. INFOR: Inf. Syst. Oper. Res. **45**(3), 123–141 (2007)
18. Raghavan, P., Tompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica **7**(4), 365–374 (1987)
19. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis. Cambridge University Press (2017)
20. GT-ITM (2019). http://www.cc.gatech.edu/projects/gtitm/
21. Ma, Y., Liang, W., Wu, J., Xu, Z.: Throughput maximization of NFV-enabled multicasting in mobile edge cloud networks. IEEE Trans. Parallel Distrib. Syst. **31**(2), 393–407 (2020)