Digital Twin-Enabled Service Provisioning in Edge Computing via Continual Learning

Jing Li, Song Guo, *Fellow, IEEE*, Weifa Liang, *Senior Member, IEEE*, Jianping Wang, *Fellow, IEEE*, Quan Chen, *Member, IEEE*, Yue Zeng, Baoliu Ye, *Member, IEEE*, and Xiaohua Jia, *Fellow, IEEE*

Abstract—Propelled by recent advances in Mobile Edge Computing (MEC) and the Internet of Things (IoT), the digital twin technique has been envisioned as a de-facto driving force to bridge the virtual and physical worlds through creating digital portrayals of physical objects. In virtue of the flourishing of edge intelligence and abundant IoT data, data-driven modelling facilitates the implementation and maintenance of digital twins, where simulations of physical objects are usually performed based on Deep Neural Networks (DNNs). A significant advantage of adopting digital twins is to enable decisive prediction on the behaviours of objects in near future without waiting for that really happen. To provide accurate predictions, it is vital to keep each digital twin synchronized with its physical object in real-time. However, it is challenging to maintain the real-time synchronization between a digital twin and its physical object due to the dynamics of physical objects and sensing data drift over time, i.e., the live data from a physical object diverge from the model training data of its digital twin. To address this critical issue, continual learning is a promising solution to retrain models of digital twins incrementally. In this paper, we investigate digital twin synchronization issues via continual learning in an MEC environment, with the aim to maximize the total utility gain, i.e., the enhanced model accuracy. We study two novel optimization problems: the static digital twin synchronization problem per time slot and the dynamic digital twin synchronization problem for a finite time horizon. We first formulate an Integer Linear Program (ILP) solution for the static digital twin synchronization problem when the problem size is small; otherwise, we develop a randomized approximation algorithm at the expense of bounded resource violations for it. We also devise a deterministic approximation algorithm with guaranteed performance for a special case of the static digital twin synchronization problem. We thirdly consider the dynamic digital twin synchronization problem by proposing an efficient online algorithm for it. Finally, we evaluate the performance of the proposed algorithms for continuous digital twin synchronization through simulations. Simulation results show that the proposed algorithms are promising, outperforming counterpart benchmarks by no less than 13.2%, in terms of the total utility gain.

Index Terms—Digital twin, continual learning, mobile edge computing, resource allocation, inference model accuracy, model retraining, approximation algorithm, and online algorithm.

1 Introduction

With the far-reaching information technologies in full swing, digital twins have been conceived to catalyze the immersing digitization and intelligent upgrade, catering to the escalating momentum for the interaction and integration of digital and physical worlds [31]. Driven by the comprehensive interpretation of data and models, a digital twin is created as an accurate portrayal of a physical object in cyberspace [40]. The pairing of each digital twin with its physical object allows insightful data analysis or computer simulations to help decision-making or head off potential serious problems prior to their arising [42]. The digital twin technique has been widely applied for manufacturing [12], healthcare [8], aeronautics, astronautics [42], etc., and it is envisioned that the global market for digital twins will reach USD 73.5 billion by 2027 [33].

- *J. Li and S. Guo are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P. R. China. E-mails: jing5.li@polyu.edu.hk and song.guo@polyu.edu.hk*
- *W. Liang, J. Wang and X. Jia are with the Department of Computer Science, City University of Hong Kong, Hong Kong, P. R. China E-mails: weifa.liang@cityu.edu.hk, jianwang@cityu.edu.hk and csjia@cityu.edu.hk*
- *Q. Chen is with the School of Computers, Guangdong University of Technology, Guangzhou 51,0006, and the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, P. R. China. E-mail: quan.c@gdut.edu.cn*
- *Y. Zeng and B. Ye are with the Department of Computer Science and Technology, Nanjing University, Nanjing 210023, P. R. China. E-mails: zengyue@smail.nju.edu.cn and yebl@nju.edu.cn*

Linked with the proliferation of Internet of Things (IoT) devices and the continuous generation of abundant IoT data [22], data-driven modelling for the implementation and maintenance of digital twins has become popular, where their data-driven models are trained by Artificial Intelligence (AI) and Machine Learning (ML) methods [29], [38]. It is pivotal for digital twins to guarantee the high-fidelity reflection of physical objects during the operation process through real-time monitoring, in terms of dynamics, intricacy, and coupling effects in the real world [12], [44]. However, such requirements for digital twins cannot be met via the centralized cloud computing which causes long communication delays, and the raw data is usually private, which is sensitive to be sent to the centralized cloud [1].

Mobile Edge Computing (MEC) has been anticipated as a promising paradigm to distribute computing and storage resources at network edge to reduce the network service latency of users significantly, while preserving user privacy [4], [10], [20], [21], [32]. In the sequel, edge intelligence has gained mounting traction in boosting the performance of digital twins and endowing high consistency between digital twins and their physical objects via edge devices [9]. For example, empowered by edge intelligence, digital twins of vehicles can provide real-time services to users, such as collision prevention in autonomous driving.

In this paper, we focus on the state updating and maintenance of digital twins through data-driven modelling in an MEC environment, where the data-driven modelling on digital twins is usually performed by Deep Neural Networks (DNNs) for simulations of physical objects [7]. Preserving historical knowledge while learning live knowledge is important in realizing real-time digital twin synchronization to support digital twin-enabled services, such as in-depth data analysis, predicting the future states of physical objects and simulating the lifecycles of physical objects [6]. Considering the dynamics of physical objects in an MEC network, the quality of digital twin services gradually deteriorates due to data drift [1], i.e., the live streaming data uploaded by a physical object diverge from the model training data of its digital twin. To mitigate the impact of data drift, the continual learning technology has been envisaged as a promising solution to combat such data draft in the data-driven modelling of digital twins, through retraining the DNN models periodically to learn from live data without catastrophic forgetting [15]. Recognizing the advantages of cloudlets (edge servers), continual learning has been adopted in platforms in edge environments, e.g., AWS IoT Greengrass and Google Cloud IoT Edge [11].

Given a finite time horizon composed of equal time slots, each time slot is treated as a *retraining window* in continual learning [1], i.e., the DNN model of a digital twin is retrained at each time slot. With a set of requests for digital twin synchronization in the beginning of a time slot, each synchronization request is associated with new data samples uploaded for the model retraining of a digital twin. To enable efficient digital twin synchronization with their objects via continual learning in an MEC environment, it poses the following questions.

The first one is how to determine the amount of retraining data for each digital twin at each time slot to maximize the overall model performance of digital twins? With a large volume of IoT data of physical objects collected periodically for digital twin synchronization, model retraining for data-driven modelling of digital twins is considered to be compute-intensive and data-intensive [11]. Namely, the model performance depends on the amount of input data for model training, while a larger amount of training data leads to consuming more computing resource [10], [43]. As the data of objects are used for various DNN models of digital twins, to maintain the accuracy of the DNN models, they need to be trained by the updated data from objects. It thus is critical to perform synchronizations between the digital twins and their objects through efficient resource allocations.

The second one is how to determine the cloudlet of each request for model retraining, subject to computing capacities of cloudlets? Because a cloudlet may not have sufficient computing resource for model retraining of all digital twins stored in the cloudlet, a synchronization request can be offloaded to another cloudlet from the cloudlet holding the digital twin within its maximum transmission distance, through transmitting current model parameters and the determined amount of retraining data. The parameters of the retrained model are then sent back to the cloudlet holding the digital twin for aggregating with historical model parameters, e.g., adopt temporal model-averaging to update model parameters, incorporating the knowledge from both live and historical data [11].

Without the future knowledge of arrivals of synchronization requests, the final question is how to maximize the overall model performance of digital twins for each model retraining during a finite time horizon? DNN models of digital twins are retrained in an incremental manner by continual learning, i.e., the performance of a retrained model depends on how the model was retrained previously.

The novelties of this work lie in dealing with continuous digital twin synchronization in MEC networks through exploring system dynamics, with the aim to maximize the total utility gain, i.e., the enhanced model accuracy via continual learning. Two novel problems of digital twin synchronization are formulated. Approximation and online algorithms for the two problems are developed and analyzed.

The main contributions of this paper are given as follows.

- We formulate two novel problems of continuous digital twin synchronization: the static digital twin synchronization problem per time slot and the dynamic digital twin synchronization problem for a finite time horizon, and demonstrate their NP-hardness.
- We provide an Integer Linear Program (ILP) solution for the static digital twin synchronization problem when the problem size is small. Otherwise, we propose a randomized algorithm with a provable approximation ratio for the problem, at the expense of bounded resource violations. Also, we consider a special case of the problem where each digital twin synchronization request can be offloaded to any cloudlet with sufficient computing resource, for which we devise a determined approximation algorithm with guaranteed performance.
- We develop an efficient online algorithm for the dynamic digital twin synchronization problem.
- We evaluate the performance of the proposed algorithms for continuous digital twin synchronization in MEC through simulations. The results demonstrate the proposed algorithms are promising, outperforming their corresponding benchmarks by at least 13.2%, in terms of the total utility gain.

The rest of this article is arranged as follows. Section 2 surveyed the related work on continuous digital twin synchronization in MEC environments via continual learning. Section 3 offers problem definitions and the NP-hardness proofs of the defined problems. Section 4 devises a randomized algorithm for the static digital twin synchronization problem, and Section 5 develops an approximation algorithm for a special static digital twin synchronization problem. Section 6 designs an online algorithm for the dynamic digital twin synchronization problem. The algorithm performance is evaluated in Section 7, and the paper is concluded in Section 8.

2 Related work

The MEC paradigm offers a variety of mobile applications at network edge close to users and physical objects (i.e., data sources), meeting the demands of digital twins for real-time processing and modelling. Existing work in the literature has explored the dynamic characteristics in MEC networks to enable efficient long-term digital twin services [8], [17], [18], [23], [27], [31], [40]. For example, considering the dynamic states of the MEC network, He *et al.* [8] utilized digital twins to estimate the collected data sizes of user devices and assist local computation of edge servers in each iteration of federated learning, mitigating the energy consumption and transmission delay. Li *et al.* [17], [18] explored the dynamic reliability prediction of Virtual Network Function (VNF) instances via digital twins, and proposed performance-guaranteed algorithms to mitigate the service cost and maximize request admissions with reliability requirements for service function chains. Li *et al.* [23] also incorporated the mobility of objects in edge-cloud environments to deploy digital twins in edge servers dynamically, and proposed approximate and online solutions to provide fresh digital twin data to users. Lin *et al.* [27] studied the stochastic service demands and mobility of users in a digital twin edge network, and developed a long-term congestion control scheme to enhance the accumulated profit of the digital twin service provider, by the Lyapunov optimization technology. Lu *et al.* [31] investigated the deployment and migration issue of digital twins to minimize service delay and energy cost in MEC environments, based on Deep Reinforcement Learning (DRL). Sun *et al.* [40] leveraged the digital twins for edge servers to anticipate their states under erratic user mobility, and adopted a DRL-based framework to devise an efficient mobile offloading scheme, with the aim to mitigate the average offloading latency among users. However, the aforementioned studies did not consider both the continuous evolution of digital twins and the dynamic retraining of their DNN models, through incorporating live data streaming from their physical objects.

Efficient synchronization between digital twins and their physical objects in a dynamic environment has attracted lots of attentions recently [5], [6], [8], [9], [31], [46]. For example, Gehrmann *et al.* [5] designed a security architecture for a digital twin-assisted automation and control scheme with synchronization demands. Han *et al.* [6] addressed the digital twin synchronization problem to maximize the payoff of virtual service providers, through employing Unmanned Aerial Vehicles (UAVs) to collect up-to-date sensory data from objects. They formulated the problem as an evolutionary game and delivered open-loop Nash solutions. Zheng *et al.* [46] studied the dynamic data synchronization between a vehicle and its digital twin, adopting a transfer learning technique. They developed an actor-critic learning approach to reduce the long-term cost in a dynamic vehicular network. However, the above-mentioned studies did not incorporate the continual learning technique for the dynamic and continuous evolution of digital twins for synchronization in MEC.

Enabled by edge intelligence, data-driven modelling has become a popular approach for building digital twins, where Deep Neural Networks (DNNs) are usually leveraged for the simulations of digital twins [29], [38]. Thanks to the continual learning technique, a DNN model can be retrained incrementally while preserving its earlier knowledge [15], [36], facilitating the efficient digital twin service provisioning based on DNNs. For example, Hashash *et al.* [7] concentrated on DNN-based digital twins and developed an edge continual learning framework to keep a digital twin synchronized with its physical object, improving the accuracy of the retrained DNN model of the digital twin while minimizing the desynchronization time. However, they considered only a single digital twin with a single edge server, and they did not pay attention to the efficient resource allocation on dynamic synchronizations for a set of digital twins in MEC. Moreover, the continual learning technique has been adopted for dynamic DNN model retraining in MEC networks [1], [3], [11], [13]. For example, Bhardwaj *et al.* [1] designed a solution, *Ekya*, to schedule resources to enable retraining and inference jobs simultaneously for continuous learning in MEC networks. Demosthenous *et al.* [3] leveraged the TensorFlow Lite library to build a continual learning model on edge devices, via adding a replay buffer to a transfer learning model. Jia *et al.* [11] explored how to combine edge and cloud resources to carry out continual learning on DNN models. Based on Lyapunov optimization, they proposed an online optimization framework for model retraining with dynamic arrivals of new data samples at a low cost while augmenting model performance. Khani *et al.* [13] developed a video-analytics mechanism based on a single server machine, combining model reusing and continuous retraining. However, the mentioned studies did not consider adopting continual learning to preserve the synchronization between digital twins and their physical objects in MEC environments with dynamics.

In contrast to the aforementioned works, in this paper we study the dynamic synchronization of digital twins with their physical objects in an MEC network via continual learning. We develop efficient approximation and online algorithms for the problems.

3 PRELIMINARIES

In this section, we introduce the system model, notions, notations, and problem definitions.

3.1 System model

Considering an MEC network $G = (V, E)$, there is a set V of cloudlets scattered in a geographical area. Each cloudlet is co-located with an Access Point (AP), while a cloudlet and its co-located AP are connected through an optical cable [21]. Denote by v a cloudlet or its co-located AP with $v \in V$ for simplicity, and let cap_v be the computing capacity of cloudlet $v \in V$. For each AP $v \in V$, there is a neighbour set $Neighbour(v)$ ($\subseteq V$), where $Neighbour(v) = \{u \mid u \in V \ \& \ dist(v, u) \leq d_{max}(v)\}$, $dist(v, u)$ is the Euclidean distance from node v to node u, and $d_{max}(v)$ is the maximum transmission distance of AP v [41]. A set E of links between APs with $E = \{(v, u) \mid$ AP u is within the maximum transmission distance of AP $v\}$. The model retraining for digital twin synchronization can be offloaded from a cloudlet v holding the digital twin to another cloudlet in $Neighbour(v)$, which will be detailed later.

Existing studies have been devoted to investigating placing digital twins in edge networks, considering the

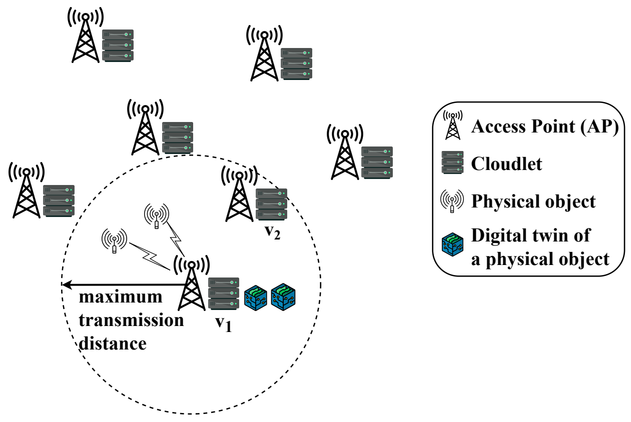

Access Point (AP)
Cloudlet
Physical object
Digital twin of a physical object
maximum transmission distance
v_1
v_2

Fig. 1. An illustrative example of an MEC network with 7 APs, where each AP is co-located with a cloudlet. There are two physical objects with their digital twins stored in a cloudlet v_1 within the maximum transmission distance of its co-located AP. The model retraining of the two digital twins could be offloaded from cloudlet v_1 to another cloudlet v_2 within the maximum transmission distance of the co-located AP of cloudlet v_1.

dynamic movements of physical objects [23], [31], [40]. In this paper, we assume that digital twins of physical objects have been deployed in cloudlets. In other words, given a set N of physical objects in the MEC network, we assume that a digital twin $DT(n)$ for each object $n \in N$ has been deployed in a cloudlet $v_n \in V$ to offer services to users, with object n under the coverage of AP v_n, while the digital twins are based on data-driven modelling and managed by deep learning frameworks, i.e., each digital twin is simulated by a Deep Neural Network (DNN) model [7]. Figure 1 illustrates an MEC network with digital twins stored in its cloudlets.

Given a finite time horizon $T = \{1, 2, \ldots, |T|\}$, the MEC network is considered to run in a discrete-time manner. Due to the system dynamics of the MEC network, the live incoming data for digital twins may be distinguished from the training data of their DNN models. It is necessary to collect up-to-date data from objects periodically, and to use the data to retrain the DNN models of their digital twins at each time slot via continual learning.

3.2 Synchronization requests, submodular functions and the utility gain

Each object $n \in N$ may issue a synchronization request $r_{n,t}$ through uploading $a_{n,t}$ units of collected data samples for synchronization with its digital twin $DT(n)$ in the beginning of time slot t, where $a_{n,t}$ is a positive integer with $a_{n,t} \geq 1$, e.g., a unit data can be 1MB. Denote by R_t the set of synchronization requests issued at time slot $t \in T$. Ideally, incorporating all incoming data for model retraining of each request in R_t for synchronization will endow the inference results with more accuracy and robustness [11], providing digital twin services with better performances.

Retraining DNN models with a large amount of data however consumes much computing resource, while the computing resource in cloudlets is limited. Suppose that the computing resource consumption for model retraining is proportional to the size of input data samples. Let $\delta_{n,t}$ be a constant as the amount of computing resource consumed for model retraining on one unit of data samples for digital twin $DT(n)$ at time slot t. To handle resource competitions among digital twins, we need to determine the number $b_{n,t}$ of units of retraining data samples of each request $r_{n,t} \in R_t$ at each time slot t, and $b_{n,t}$ is an integer with $0 \leq b_{n,t} \leq a_{n,t}$. Then, it consumes the amount $b_{n,t} \cdot \delta_{n,t}$ of computing resource for each request $r_{n,t}$. Notice that if $b_{n,t} = 0$, request $r_{n,t}$ is rejected.

In the following, we define the utility gain $h_{n,t}(b_{n,t})$ of request $r_{n,t}$, through retraining the model of digital twin $DT(n)$ at time slot t with $b_{n,t}$ units of data samples.

$$h_{n,t}(b_{n,t}) = Accuracy(n, t, b_{n,t}) - Accuracy(n, t, 0), \quad (1)$$

where $Accuracy(n, t, b_{n,t})$ is the model accuracy of $DT(n)$ after its retraining with $b_{n,t}$ units of data samples at time slot t. $Accuracy(n, t, 0)$ is the model accuracy of $DT(n)$ without any retraining at time slot t. We observe $h_{n,t}(0) = 0$, i.e., request $r_{n,t}$ is rejected, and no utility gain is obtained.

There are several works to estimate the accuracy of a retrained model [1], [13]. For example, the authors in [1] proposed a *micro-profiler*, which can be used to estimate the accuracy of the retrained model with different retraining data sizes, by retraining the model with a small subset of training data and several epochs. Through conducting experiments on various well-known DNNs, they claimed the proposed micro-profiler is $100\times$ more efficient than exhaustive profiling, and the error of accuracy estimation is around 5.8%, which is low enough in practice [1]. Moreover, the authors in [13] developed a profiler to estimate the retraining speed of each model, by measuring the actual accuracy improvement of each model within its learning progress.

Through adopting existing techniques, e.g., micro-profiler [1], we can estimate the utility gain $h_{n,t}(\cdot)$ in a short time. However, it is difficult to estimate $h_{n,t}(\cdot)$ before time slot t. This is because the DNN models of digital twins are retrained in an incremental manner via continual learning, i.e., the model quality of $DT(n)$ at time slot $(t-1)$ impacts its digital twin service performance at time slot t. This means that for a given digital twin $DT(n)$, the utility gain function $h_{n,t}(\cdot)$ at time slot t depends on not only the new data samples, but also the quality of its DNN model retrained at all previous time slots. Therefore, the utility gain function $h_{n,t}(\cdot)$ is unknown until time slot t.

Definition 1. Given a finite set Ω and a set $\mathbb{R}^{\geq 0}$ of non-negative real numbers, *a submodular function* on Ω is a set function $g : 2^{\Omega} \longmapsto \mathbb{R}^{\geq 0}$. Function $g(\cdot)$ is a submodular function if (i) $g(\emptyset) = 0$, and (ii) for every two sets $\mathbb{S}_1, \mathbb{S}_2 \subseteq \Omega$ with $\mathbb{S}_1 \subseteq \mathbb{S}_2$ and every element $s \in \Omega \setminus \mathbb{S}_2$, $g(\mathbb{S}_1 \cup \{s\}) - g(\mathbb{S}_1) \geq g(\mathbb{S}_2 \cup \{s\}) - g(\mathbb{S}_2)$.

It is worth mentioning that with the increase on the retraining data for digital twin $DT(n)$, its model performance increases, while the growth rate of its model performance becomes slower by theoretical and empirical practices [10], [43]. Hence, we assume that the utility gain function $h_{n,t}(\cdot)$ is a submodular function.

Recall that digital twin $DT(n)$ of object $n \in N$ is established in cloudlet v_n. Because the computing resource possessed by cloudlet v_n may be insufficient for the synchronization of all digital twins in cloudlet v_n, we can offload the synchronization request of object n to another cloudlet within the maximum transmission distance of the

AP co-located with cloudlet v_n, through transmitting current model parameters and the determined amount of retraining data. The parameters of the retrained DNN model are then sent back to cloudlet v_n for aggregating with the historical model parameters of the digital twin, e.g., adopt temporal model-averaging to update the model parameters, incorporating the knowledge from both the live and historical data of the physical object [11]. Let $\mathcal{V}(n)$ be a set of cloudlets, consisting of cloudlet v_n and all cloudlets within the maximum transmission distance of the AP co-located with the cloudlet v_n, i.e., $\mathcal{V}(n) = \{v_n\} \cup Neighbour(v_n)$. It can be seen that the synchronization request of object n for its digital twin can be processed in any cloudlet $v' \in \mathcal{V}(n)$.

3.3 Problem definitions

We first consider the static digital twin synchronization problem for a single time slot in the defined MEC network as follows. In the beginning of time slot t, for each synchronization request $r_{n,t} \in R_t$, given the utility gain function $h_{n,t}(\cdot)$ of each digital twin $DT(n)$, we determine the number $b_{n,t}$ of units of retraining data samples for each request, which consumes the amount $b_{n,t} \cdot \delta_{n,t}$ of computing resource. Because the request of $DT(n)$ in cloudlet v_n can be processed in any cloudlet $v' \in \mathcal{V}(n)$ with $\mathcal{V}(n) = \{v_n\} \cup Neighbour(v_n)$, we also determine the cloudlet for each request in which it is processed, considering computing capacities on cloudlets.

Definition 2. Given an MEC network $G = (V, E)$, a single time slot t, a set N of physical objects, and a set R_t of synchronization requests, each request $r_{n,t} \in R_t$ of object n has $a_{n,t}$ units of data samples uploaded for synchronization with its digital twin $DT(n)$ in the beginning of time slot t, and the utility gain function $h_{n,t}(\cdot)$. The *static digital twin synchronization problem* in G is to maximize the total utility gain of synchronization requests in R_t at time slot t, i.e., maximize $\sum_{r_{n,t}\in R_t} h_{n,t}(b_{n,t})$, through determining the number $b_{n,t}$ of units of retraining data samples of each request $r_{n,t}$, subject to computing capacities on cloudlets.

Let $\mathcal{N}(R_t)$ be the set of objects with each issuing a synchronization request in R_t in the beginning of time slot t. Let $x_{n,v,k}(t)$ be a binary decision variable, and $x_{n,v,k}(t) = 1$ means the synchronization request $r_{n,t}$ of object $n \in \mathcal{N}(R_t)$ is processed in cloudlet $v \in V$ with k units of retraining data samples, where k is a positive integer with $1 \le k \le a_{n,t}$, and $a_{n,t}$ is the number of units of data samples uploaded of request $r_{n,t}$; otherwise $x_{n,v,k}(t) = 0$.

The Integer Linear Program (ILP) solution to the static digital twin synchronization problem at time slot t is formulated as follows.

$$\text{Maximize} \sum_{n\in\mathcal{N}(R_t)} \sum_{v\in V} \sum_{k=1}^{a_{n,t}} h_{n,t}(k) \cdot x_1 n, v, k(t), \tag{2}$$

subject to:

$$\sum_{n\in\mathcal{N}(R_t)} \sum_{k=1}^{a_{n,t}} k \cdot \delta_{n,t} \cdot x_{n,v,k}(t) \le cap_v, \quad \forall v \in V \tag{3}$$

$$\sum_{v\in\mathcal{V}(n)} \sum_{k=1}^{a_{n,t}} x_{n,v,k}(t) \le 1, \quad \forall n \in \mathcal{N}(R_t) \tag{4}$$

$$x_{n,v,k}(t) = 0, \ \forall n \in \mathcal{N}(R_t), \ \forall v \in V \setminus \mathcal{V}(n), \ \forall k \in [1, a_{n,t}] \tag{5}$$

$$x_{n,v,k}(t) \in \{0,1\}, \ \forall n \in \mathcal{N}(R_t), \ \forall v \in V, \ \forall k \in [1, a_{n,t}], \tag{6}$$

where objective (2) indicates the total utility gain of synchronization requests in R_t at time slot t. Constraint (3) means no computing resource capacity of cloudlet is violated. Constraint (4) means the request of each object $n \in \mathcal{N}$ is processed on at most one cloudlet in $\mathcal{V}(n) = \{v_n\} \cup Neighbour(v_n)$, with the determined amount of retraining data. Constraint (5) shows each synchronization request cannot be processed on any cloudlet out of the maximum transmission distance of the AP co-located with the cloudlet holding the digital twin, i.e., each request of object $n \in \mathcal{N}(R_t)$ cannot be processed in any cloudlet $v \in V \setminus \mathcal{V}(n)$. Constraint (6) shows $x_{n,v,k}(t)$ is a binary decision variable showing whether the synchronization request of object $n \in \mathcal{N}(R_t)$ is processed in cloudlet $v \in V$, and k units of data samples are utilized for model retraining of digital twin $DT(n)$, with $k \in [1, a_{n,t}]$.

We then consider the dynamic digital twin synchronization problem as follows. Given a finite time horizon T consisting of sequential time slots, a set R_t of synchronization requests are issued in the beginning of each time slot $t \in T$, without any knowledge of future requests. Note that the utility gain function $h_{n,t}(\cdot)$ of digital twin $DT(n)$ at time slot t depends on its model retrained at previous time slots, because the models are retrained in an incremental manner by continual learning. The dynamic digital twin synchronization problem is to dynamically determine the amount of retraining data for each digital twin synchronization and at which cloudlet for its implementation at that time slot, without any future knowledge.

Definition 3. Given an MEC network $G = (V, E)$, a finite time horizon T, a set N of physical objects, and a set R_t of synchronization requests at each time slot $t \in T$ without the knowledge of future request arrivals, each request $r_{n,t} \in R_t$ of object n at time slot t has $a_{n,t}$ units of data samples uploaded for synchronization with its $DT(n)$ and a utility gain function $h_{n,t}(\cdot)$. The *dynamic digital twin synchronization problem* in G is to maximize the accumulated utility gain of all admitted synchronization requests over time horizon T, i.e., maximize $\sum_{t\in T} \sum_{r_{n,t}\in R_t} h_{n,t}(b_{n,t})$, through dynamically determining the number $b_{n,t}$ of units of data samples for the model retraining of each synchronization request $r_{n,t}$ in the beginning of time slot $t \in T$, subject to computing capacities on cloudlets.

All symbols adopted are listed in Table 1.

3.4 NP-hardness of the defined problems

Theorem 1. the static digital twin synchronization problem in an MEC network $G = (V, E)$ is NP-hard.

Proof We show the NP-hardness of the static digital twin synchronization problem via a polynomial reduction from the Generalized Assignment Problem (GAP) [35], which is NP-hard.

Given a set $\mathcal{B}$ of bins with each bin $b \in \mathcal{B}$ possessing the capacity $C(b)$, there are a set $\mathcal{I}$ of items, and a profit $profit(i, b)$ is collected by packing item $item(i) \in \mathcal{I}$ into

TABLE 1
Table of Symbols

Notations	Descriptions
$G = (V, E)$	In an MEC network G, V is the set of APs (cloudlets) and E is the set of links between APs.
cap_v	The computing capacity of cloudlet v.
$Neighbour(v)$	The neighbour node set of AP v, with $Neighbour(v) = \{u \mid u \in V \ \& \ dist(v, u) \leq d_{max}(v)\}$, $dist(v, u)$ is the Euclidean distance from node v to node u, and $d_{max}(v)$ is the maximum transmission distance of AP v.
$T = \{1, 2, \ldots, \|T\|\}$	The monitoring time horizon.
N	The set of physical objects.
$DT(n)$ and v_n	The digital twin of object n, and the cloudlet holding digital twin $DT(n)$.
$r_{n,t}$	The synchronization request issued by object n at time slot t.
R_t	The set of synchronization requests issued at time slot t.
$a_{n,t}$	The number of units of collected data samples for synchronization with digital twin $DT(n)$ at time slot t.
$b_{n,t}$	The determined number $b_{n,t}$ of units of retraining data samples of request $r_{n,t} \in R_t$ at time slot t.
$\delta_{n,t}$	The amount of computing resource consumed for model retraining on one unit of data samples for digital twin $DT(n)$ at time slot t.
$Accuracy(n, t, b_{n,t})$	The model accuracy of $DT(n)$ after its retraining with $b_{n,t}$ units of data samples at time slot t.
$h_{n,t}(b_{n,t})$	The utility gain of request $r_{n,t}$, through retraining the model of digital twin $DT(n)$ at time slot t with $b_{n,t}$ units of data samples, defined by Eq. (1).
$\mathcal{V}(n)$	The set of cloudlets, consisting of cloudlet v_n and all cloudlets within the maximum transmission distance of the AP co-located with the cloudlet v_n, i.e., $\mathcal{V}(n) = \{v_n\} \cup Neighbour(v_n)$.
$\mathcal{N}(R_t)$	The set of objects with each issuing a synchronization request in R_t at time slot t.
$x_{n,v,k}(t)$	The binary decision variable showing whether the synchronization request $r_{n,t}$ of object n is processed in cloudlet $v \in V$ with k units of retraining data samples.
$p_{n,k,t}$	The utility gain margin for model training of $DT(n)$ by utilizing k units of data samples, compared with that utilizing the $(k-1)$ units of data samples at time slot t.
$y_{n,k}(t)$	Then binary variable showing whether the model training of $DT(n)$ utilizes the kth unit of data samples with $1 \leq k \leq a_{n,t}$ at time slot t.
$\mathcal{R}_{v,t}^{initial}$	The set of synchronization requests assigned to cloudlet v at time slot t by the initial solution by invoking `Algorithm` 1.
$\Delta\psi_{n,t}(b_{n,t})$	The marginal utility loss of reducing one unit of retraining data samples for request $r_{n,t}$ with $b_{n,t} \geq 1$.

bin b, where each item $item(i)$ has a size $size(i)$. The GAP is to maximize the total collected profit.

In the following, we consider a special case of the static digital twin synchronization problem, where each request $r_{n,t} \in R_t$ is associated with one unit of data samples uploaded at time slot t, i.e., $a_{n,t} = 1$. We treat each cloudlet $v \in V$ as a bin with capacity cap_v, while we treat each request as an item with size of $\delta_{n,t}$, i.e., the computing resource demand of model retraining for digital twin $DT(n)$ with one unit of data samples at time slot t. If synchronization request $r_{n,t}$ is processed in cloudlet $v' \in \mathcal{V}(n)$, a profit $h_{n,t}(1)$ is collected. Otherwise (request $r_{n,t}$ is rejected or processed in a cloudlet $v' \in V \setminus \mathcal{V}(n)$), the collected profit is 0. We can observe this special problem is to maximize the total collected profit by allocating requests R_t to cloudlets V for digital twin synchronization at time slot t. Such a special case of the static digital twin synchronization problem is equivalent to the NP-hard GAP [35]. Thus, the static digital twin synchronization problem is NP-hard. ∎

Corollary 1. The dynamic digital twin synchronization problem in an MEC network $G = (V, E)$ is NP-hard.

Proof We can observe that the static digital twin synchronization problem is a special case of the dynamic digital twin synchronization problem when the monitoring period consists of a single time slot only. Therefore, the dynamic digital twin synchronization problem is NP-hard, because of the NP-hardness of the static digital twin synchronization problem by Theorem 1. Hence, the corollary follows. ∎

4 Randomized Algorithm for the Static Digital Twin Synchronization Problem

In this section, we devise a randomized algorithm for the static digital twin synchronization problem, built upon the Integer Linear Program (ILP) solution (2).

4.1 Randomized algorithm

Although the ILP formulation (2) can find an exact solution for the static digital twin synchronization problem, its running time is excessively large when the problem is large. To provide a feasible solution in a reasonable amount of time when the problem size is large, the randomized technique is a powerful tool [37], which has been used for reliable service function chain services in MEC [22], [25], [26]. Following the same spirit, we develop a randomized algorithm for the static digital twin synchronization problem as follows.

We first relax the values of binary variables $x_{n,v,k}(t)$ into real numbers between 0 and 1, i.e., we have

$$\widetilde{x}_{n,v,k}(t) \in [0, 1], \ \forall n \in \mathcal{N}(R_t), \ \forall v \in V, \ \forall k \in [1, a_{n,t}]. \quad (7)$$

We then obtain an optimal fractional solution to the static digital twin synchronization problem based on its relaxed Linear Program (LP) solution by (2). We finally obtain an integral solution, through performing randomized rounding [37] on the optimal fractional solution. That is,

Algorithm 1 A randomized algorithm for the static digital twin synchronization problem

```
Input: Given an MEC network $G = (V, E)$, a single time slot $t$, a set $N$ of physical objects, a set $R_t$ of synchronization requests, and the utility gain function $h_{n,t}(\cdot)$, each request $r_{n,t} \in R_t$ of object $n$ has $a_{n,t}$ units of data samples uploaded in the beginning of time slot $t$.
Output: A solution to maximize the total utility gain of synchronization requests at time slot $t$.
1: Solve the relaxed LP formulation (2);
2: Obtain the optimal fractional solution to the LP formulation, with $\tilde{x}_{n,v,k}(t) \in [0,1]$ the fractional value of each $x_{n,v,k}(t)$;
3: Due to the randomized rounding by [37], we set the integral value $\hat{x}_{n,v,k}(t)$ of $x_{n,v,k}(t)$ as 1 with the probability of $\tilde{x}_{n,v,k}(t)$; we set $\hat{x}_{n,v,k}(t)$ as 0 otherwise. We perform such randomized rounding in an exclusive manner by Constraint (4), and the randomized rounding is also performed independently for each object $n \in \mathcal{N}(R_t)$;
4: Return An integral solution, i.e., $\{\hat{x}_{n,v,k}(t) \mid \forall n \in \mathcal{N}(R_t), \forall v \in V, \forall k \in [1, a_{n,t}]\}$.
```

each binary decision variable $x_{n,v,k}(t)$ is set as 1 with the probability $\widetilde{x}_{n,v,k}(t)$. Note that we perform the randomized rounding exclusively based on Constraint (4), i.e. at most one $x_{n,v,k}(t)$, $\forall n \in \mathcal{N}(R_t)$ can be set as 1, while the rest is set as 0. Such randomized rounding is performed for each object $n \in \mathcal{N}(R_t)$ independently.

The proposed randomized algorithm is detailed in `Algorithm` 1.

4.2 Algorithm analysis

The rest is to prove the approximation ratio of `Algorithm` 1 and bound its capacity violation on any cloudlet. Let γ be defined as follows.

$$\gamma = \max\{h_{n,t}(a_{n,t}),\ a_{n,t} \cdot \delta_{n,t} \mid \forall n \in \mathcal{N}(R_t)\}, \quad (8)$$

where $\max\{h_{n,t}(a_{n,t}) \mid \forall n \in \mathcal{N}(R_t)\}$ is the maximum utility gain by a synchronization request at time slot t with $a_{n,t}$ units of retraining data samples, and $\max\{a_{n,t} \cdot \delta_{n,t} \mid \forall n \in \mathcal{N}(R_t)\}$ is the maximum demand of computing resource for a synchronization request with its all retraining data samples at the time slot t.

Theorem 2. Given an MEC network $G = (V, E)$, a single time slot t, a set N of physical objects, and a set R_t of synchronization requests, each synchronization request $r_{n,t} \in R_t$ of object n has $a_{n,t}$ units of data samples uploaded at time slot t, and the utility gain function $h_{n,t}(\cdot)$. There is a randomized algorithm, `Algorithm` 1, for the static digital twin synchronization problem. The approximation ratio of the proposed randomized algorithm is $\frac{1}{2}$, and the amount of computing resource consumed by any cloudlet is no more than twice its capacity, with a high probability of $\min\{1 - \frac{1}{|R_t|}, 1 - \frac{1}{|V|}\}$, provided that $\widetilde{OPT} \geq 8\gamma \ln |R_t|$ and $\min\{cap_v \mid v \in V\} \geq 6\gamma \ln |V|$, where $\widetilde{OPT}$ is the optimal fractional solution by LP (2) and γ is a constant given by Eq. (8).

Proof We start by analyzing the approximation ratio of the randomized algorithm.

Denote by OPT the optimal solution by the ILP formulation (2), and $\widetilde{OPT}$ the optimal fractional solution by its LP. Because the problem of concern is a maximization problem, we have $\widetilde{OPT} \geq OPT$. Recall that $\tilde{x}_{n,v,k}(t) \in [0,1]$ is a real value obtained by $\widetilde{OPT}$, while we set $\hat{x}_{n,v,k}(t)$ as 1 with the probability of $\tilde{x}_{n,v,k}(t)$, and we set $\hat{x}_{n,v,k}(t)$ as 0 otherwise.

Let $z_{n,v,k}(t) = \frac{h_{n,t}(k)}{\gamma} \cdot \hat{x}_{n,v,k}(t)$ be a random variable deriving from random variable $\hat{x}_{n,v,k}(t)$. We observe that $z_{n,v,k}(t)$ is $\frac{h_{n,t}(k)}{\gamma}$ with probability $\tilde{x}_{n,v,k}(t)$; otherwise, $z_{n,v,k}(t)$ is 0. Thus, we have $0 \leq z_{n,v,k}(t) \leq 1$, because $\frac{h_{n,t}(k)}{\gamma} \cdot \hat{x}_{n,v,k}(t) \leq \frac{h_{n,t}(k)}{\max\{h_{n,t}(a_{n,t}) \mid \forall n \in \mathcal{N}(R_t)\}} \leq 1$. The expected value of $z_{n,v,k}(t)$ is $\mathbb{E}[z_{n,v,k}(t)] = \frac{h_{n,t}(k)}{\gamma} \cdot \tilde{x}_{n,v,k}(t)$. We then have

$$\mathbb{E}[\sum_{n \in \mathcal{N}(R_t)} \sum_{v \in V} \sum_{k=1}^{a_{n,t}} z_{n,v,k}(t)]$$
$$= \sum_{n \in \mathcal{N}(R_t)} \sum_{v \in V} \sum_{k=1}^{a_{n,t}} \frac{h_{n,t}(k) \cdot \tilde{x}_{n,v,k}(t)}{\gamma} = \frac{\widetilde{OPT}}{\gamma}. \quad (9)$$

Let α be a constant with $0 < \alpha < 1$. By Chernoff Bounds [34], we have

$$\mathbf{Pr}[\sum_{n \in \mathcal{N}(R_t)} \sum_{v \in V} \sum_{k=1}^{a_{n,t}} h_{n,t}(k) \cdot \hat{x}_{n,v,k}(t) \leq (1-\alpha) \cdot OPT]$$
$$\leq \mathbf{Pr}[\sum_{n \in \mathcal{N}(R_t)} \sum_{v \in V} \sum_{k=1}^{a_{n,t}} h_{n,t}(k) \cdot \hat{x}_{n,v,k}(t) \leq (1-\alpha) \cdot \widetilde{OPT}],$$
$$\text{because } \widetilde{OPT} \geq OPT$$
$$= \mathbf{Pr}[\sum_{n \in \mathcal{N}(R_t)} \sum_{v \in V} \sum_{k=1}^{a_{n,t}} z_{n,v,k}(t) \leq (1-\alpha) \cdot \frac{\widetilde{OPT}}{\gamma}],$$
$$\text{by } z_{n,v,k}(t) = h_{n,t}(k) \cdot \hat{x}_{n,v,k}(t)$$
$$= \mathbf{Pr}[\sum_{n \in \mathcal{N}(R_t)} Z_n \leq (1-\alpha) \cdot \frac{\widetilde{OPT}}{\gamma}], \text{ let } Z_n = \sum_{v \in V} \sum_{k=1}^{a_{n,t}} z_{n,v,k}(t)$$
$$\leq exp(\frac{-\alpha^2 \cdot \widetilde{OPT}}{2\gamma}), \text{ by the Chernoff Bounds [34].} \quad (10)$$

Assuming that $exp(\frac{-\alpha^2 \cdot \widetilde{OPT}}{2\gamma}) \leq \frac{1}{|R_t|}$ with $0 < \alpha < 1$, we have $\alpha \geq \sqrt{\frac{2\gamma \ln |R_t|}{\widetilde{OPT}}}$. When $\frac{1}{2} \leq \alpha < 1$, we have $\sqrt{\frac{2\gamma \ln |R_t|}{\widetilde{OPT}}} \leq \frac{1}{2}$, i.e., we must have $\widetilde{OPT} \geq 8\gamma \ln |R_t|$. Because $1 - \alpha \leq \frac{1}{2}$, the approximation ratio of `Algorithm` 1, is $\frac{1}{2}$ with high probability $1 - \frac{1}{|R_t|}$, by $\widetilde{OPT} \geq 8\gamma \ln |R_t|$.

We then show the violation of computing capacity on each cloudlet $v \in V$ as follows.

Let $q_{n,v,k}(t)$ be a random variable derived from random variable $\hat{x}_{n,v,k}(t)$, i.e., $q_{n,v,k}(t) = \frac{k \cdot \delta_{n,t}}{\gamma} \cdot \hat{x}_{n,v,k}(t)$, and its expectation is $\mathbb{E}[q_{n,v,k}(t)] = \frac{k \cdot \delta_{n,t}}{\gamma} \cdot \tilde{x}_{n,v,k}(t)$ with $0 \leq q_{n,v,k}(t) \leq 1$, because $\frac{k \cdot \delta_{n,t}}{\gamma} \cdot \hat{x}_{n,v,k}(t) \leq \frac{k \cdot \delta_{n,t}}{\max\{a_{n,t} \cdot \delta_{n,t} \mid \forall n \in \mathcal{N}(R_t)\}} \leq 1$.

For each cloudlet $v \in V$, we have

$$\mathbb{E}[\sum_{n \in \mathcal{N}(R_t)} \sum_{k=1}^{a_{n,t}} q_{n,v,k}(t)]$$

$$= \sum_{n\in\mathcal{N}(R_t)} \sum_{k=1}^{a_{n,t}} \frac{k\cdot\delta_{n,t}\cdot\tilde{x}_{n,v,k}(t)}{\gamma} = \frac{\widetilde{cap}_v}{\gamma}, \tag{11}$$

where $\widetilde{cap}_v$ is the amount of computing resource consumed on cloudlet v by LP (2).

From Constraint (3), the computing resource consumption on cloudlet v is $\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} k\cdot\delta_{n,t}\cdot\hat{x}_{n,v,k}(t)$ by Algorithm 1. Let λ be a constant with $\lambda > 0$. The probability of violating the computing capacity on any cloudlet $v\in V$ is

$$\begin{aligned}
&\mathbf{Pr}[\bigvee_{v\in V}\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} k\cdot\delta_{n,t}\cdot\hat{x}_{n,v,k}(t) \geq (1+\lambda)\cdot cap_v]\\
\leq&\mathbf{Pr}[\bigvee_{v\in V}\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} k\cdot\delta_{n,t}\cdot\hat{x}_{n,v,k}(t) \geq (1+\lambda)\cdot \widetilde{cap}_v],\\
&\qquad\qquad\text{since } \widetilde{cap}_v \leq cap_v\\
\leq&\sum_{v\in V}\mathbf{Pr}[\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} q_{n,v,k}(t) \geq (1+\lambda)\cdot\frac{\widetilde{cap}_v}{\gamma}],\\
&\text{by } q_{n,v,k}(t)=\frac{k\cdot\delta_{n,t}}{\gamma}\cdot\hat{x}_{n,v,k}(t) \text{ and Union Bound Inequality}\\
\leq&\sum_{v\in V}\mathbf{Pr}[\sum_{n\in\mathcal{N}(R_t)} Q_{n,v} \geq (1+\lambda)\cdot\frac{\widetilde{cap}_v}{\gamma}],\\
&\qquad\qquad\text{let } Q_{n,v} = \sum_{k=1}^{a_{n,t}} q_{n,v,k}(t)\\
\leq&|V|\cdot exp(\frac{-\lambda^2\cdot\widetilde{cap}_v}{(2+\lambda)\cdot\gamma}),\\
&\qquad\text{by Eq. (11) and the Chernoff Bounds [34].}
\end{aligned} \tag{12}$$

Now, we set $\lambda \leq 1$, and assume

$$exp(\frac{-\lambda^2\cdot\widetilde{cap}_v}{(2+\lambda)\cdot\gamma}) \leq \frac{1}{|V|^2}. \tag{13}$$

Because $0 < \lambda \leq 1$, by Eq. (13), we have $exp(\frac{-\lambda^2\cdot\widetilde{cap}_v}{3\gamma}) \leq \frac{1}{|V|^2}$. Therefore, we have $\lambda \geq \sqrt{\frac{6\gamma\ln|V|}{\widetilde{cap}_v}} \geq \sqrt{\frac{6\gamma\ln|V|}{cap_v}}$ due to $\widetilde{cap}_v \leq cap_v$, and we also have $cap_v \geq 6\gamma\ln|V|$ for any cloudlet $v\in V$, i.e., $\min\{cap_v \mid \forall v\in V\} \geq 6\gamma\ln|V|$. This means Eq. (13) holds, provided by $\min\{cap_v \mid \forall v\in V\} \geq 6\gamma\ln|V|$.

By Eq. (12) and Eq. (13), thus,

$$\begin{aligned}
&\mathbf{Pr}[\bigvee_{v\in V}\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} k\cdot\delta_{n,t}\cdot\hat{x}_{n,v,k}(t) \geq (1+\lambda)\cdot cap_v]\\
\leq&|V|\cdot\frac{1}{|V|^2} = \frac{1}{|V|}.
\end{aligned} \tag{14}$$

Because $1+\lambda \leq 2$, the computing resource consumption on each cloudlet $v\in V$ is no more than twice its capacity, with high probability $1-\frac{1}{|V|}$, provided by $\min\{cap_v \mid \forall v\in V\} \geq 6\gamma\ln|V|$.

Hence, the approximation ratio of Algorithm 1 is $\frac{1}{2}$, and the computing resource consumed on each cloudlet $v\in V$ is no more than twice its computing capacity, with high probability $\min\{1-\frac{1}{|R_t|}, 1-\frac{1}{|V|}\}$, provided that $\widetilde{OPT} \geq 8\gamma\ln|R_t|$ and $\min\{cap_v \mid v\in V\} \geq 6\gamma\ln|V|$. ■

5 Approximation Algorithm for a Special Static Digital Twin Synchronization Problem

In this section, we consider a special static digital twin synchronization problem where each synchronization request $r_{n,t}$ can be offloaded to any cloudlet $v\in V$ in the MEC network, which can be achieved by multi-hop communications at the expense of longer communication delays [28], [39]. Notice that it is common sense that the communication delay caused by the multi-hop communication within the MEC network is far less than the delay between any AP in the MEC network and the remote cloud [20]. Also, the proposed randomized algorithm is not a deterministic algorithm, which may cause computing capacity violations on cloudlets. In the following, we propose a deterministic approximation algorithm for this special problem without any capacity violation.

The basic idea for the deterministic approximation algorithm is given as follows.

We first consider a special MEC network that consists of a single cloudlet v_0 only with computing capacity being the sum of the computing capacities of all cloudlets in V, i.e., $cap_{v_0} = \sum_{v\in V} cap_v$, and a solution for the problem with this special MEC network then can be transformed into a feasible solution to the MEC network with multiple cloudlets. We formulate an Integer Linear Program (ILP) solution in this special MEC network, and then propose an approximate solution based on the ILP solution. We then derive an approximation algorithm for the problem in an MEC network with multiple cloudlets through a transformation the approximate solution in the special MEC network.

5.1 ILP formulation for the problem in an MEC network consisting of a single cloudlet v_0 only

Recall that $\mathcal{N}(R_t)$ is the set of objects, with each issuing a synchronization request in R_t in the beginning of time slot t. Because each physical object $n\in\mathcal{N}(R_t)$ uploads $a_{n,t}$ units of data samples in the beginning of the time slot t, let $p_{n,k,t}$ be the utility gain for model training of $DT(n)$ by utilizing the kth unit of data samples with $1\leq k\leq a_{n,t}$ at time slot t, i.e., the utility gain margin for model training of $DT(n)$ by utilizing k units of data samples, compared with that by utilizing $(k-1)$ units of data samples. Then,

$$p_{n,k,t} = h_{n,t}(k) - h_{n,t}(k-1). \tag{15}$$

Let $y_{n,k}(t)$ be a binary variable, and $y_{n,k}(t) = 1$ means that the model training of $DT(n)$ utilizes the kth unit of data samples with $1\leq k\leq a_{n,t}$ at time slot t; otherwise $y_{n,k}(t) = 0$. We formulate an ILP for the problem, assuming there is a single cloudlet v_0 with capacity cap_{v_0} $(=\sum_{v\in V} cap_v)$ in the MEC network as follows.

$$\text{Maximize} \sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} p_{n,k,t}\cdot y_{n,k}(t), \tag{16}$$

subject to:

$$\sum_{n\in\mathcal{N}(R_t)}\sum_{k=1}^{a_{n,t}} \delta_{n,t}\cdot y_{n,k}(t) \leq cap_{v_0}, \ \ \forall v\in V \tag{17}$$

$$y_{n,k}(t)\in\{0,1\}, \ \ \forall n\in\mathcal{N}(R_t),\ \forall k\in[1,a_{n,t}], \tag{18}$$

where the objective (16) is the total utility gain of synchronization requests processed in cloudlet v_0. Constraint (17) indicates the computing capacity constraint on cloudlet v_0.

5.2 Approximation algorithm

We propose an approximate solution to the problem where it exists a single cloudlet v_0 in the MEC network. Built upon the solution for this special case, we then devise an approximation algorithm for the problem with multiple cloudlets in the MEC network.

We first devise an approximation algorithm for the problem of concern where it exists a single cloudlet v_0, via reducing it to the knapsack problem [14] as follows. The cloudlet v_0 is treated as a bin possessing capacity $cap_{v_0} = \sum_{v \in V} cap_v$. The kth unit of data samples in each synchronization request of each object $n \in N$ is treated as an item $i_{n,k}$ associated with a profit $p_{n,k,t}$ by Eq. (15) and size of $\delta_{n,t}$, i.e., the computing resource demand for utilizing the kth unit of retraining data samples in each request. The knapsack problem is to maximize the total collected profit through packing items into the bin, considering its capacity. We observe this problem is equivalent to the knapsack problem. By adopting the approximation algorithm [14], we obtain an approximate solution to the problem when there is a single cloudlet v_0 only, i.e., $\{y_{n,k}(t) \mid \forall n \in \mathcal{N}(R_t), \forall k \in [1, a_{n,t}]\}$.

We then construct a solution $\mathbb{A}$ for the problem with multiple cloudlets, by setting the number $b_{n,t}$ of units of retraining data sample for each object $n \in \mathcal{N}(R_t)$ with $b_{n,t} = \sum_{k=1}^{a_{n,t}} y_{n,k}(t)$, and $\mathbb{A}$ is a set of requests with $b_{n,t} \geq 1$. However, directly assigning requests in $\mathbb{A}$ to cloudlets in V may cause resource violations, because $\mathbb{A}$ is derived from the problem when there is only a single cloudlet v_0 with $cap_{v_0} = \sum_{v \in V} cap_v$.

We finally show how to implement requests in $\mathbb{A}$ on cloudlets to avoid resource violations through dividing set $\mathbb{A}$ into two disjoint subsets $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. We sort cloudlets in V in non-increasing order of their computing capacity. For simplicity, let $v_1, v_2, \ldots, v_{|V|}$ be the sorted cloudlet sequence with $cap_{v_1} \geq cap_{v_2} \geq \cdots \geq cap_{v_{|V|}}$. We also sort synchronization requests in $\mathbb{A}$ in non-increasing order of $b_{n,t}$ - the number of units of retraining data samples. Let $r_{n_1,t}, r_{n_2,t}, \ldots, r_{n_{|\mathbb{A}|},t}$ be the sorted request sequence with $b_{n_1,t} \geq b_{n_2,t} \geq \cdots \geq b_{n_{|\mathbb{A}|},t}$. We then assign the sorted requests in $\mathbb{A}$ to cloudlets one by one as follows.

Initially, set $\mathcal{A}_1$ and $\mathcal{A}_2$ are empty. We first consider assigning requests in $\mathbb{A}$ to cloudlet v_1 one by one, until after assigning request $r_{n_L,t}$ with $1 \leq L \leq |\mathbb{A}|$, cloudlet v_1 has no residual computing resource (i.e., its computing resource consumption is exactly its computing capacity), or its computing capacity is violated (i.e., its computing resource consumption is more than its computing capacity). If assigning request $r_{n_l,t}$ to cloudlet v_1 results in no resource violation with $1 \leq l \leq L$, we add request $r_{n_l,t}$ to set $\mathcal{A}_1$; otherwise, we add request $r_{n_l,t}$ to set $\mathcal{A}_2$. We then consider assigning the rest requests one by one to cloudlet v_2, similarly. The procedure continues until all requests have been assigned. Note that all requests in $\mathbb{A}$ can be assigned to cloudlets V, because the solution $\mathbb{A}$ is for the problem of concern with the single cloudlet v_0 under the computing capacity of cap_{v_0} $(= \sum_{v \in V} cap_v)$, and computing capacity violations are allowed. The set $\mathbb{A}$ has been divided into two disjoint subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ with $\mathbb{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ and $\mathcal{A}_1 \cap \mathcal{A}_2 = \emptyset$. The approximation algorithm chooses one of them with a larger total utility as the approximate solution to the problem.

Algorithm 2 Approximation algorithm for a special static digital twin synchronization problem

Input: Given an MEC network $G = (V, E)$, a single time slot t, a set N of physical objects, a set R_t of synchronization requests, and the utility gain function $h_{n,t}(\cdot)$, each request $r_{n,t} \in R_t$ of object n has $a_{n,t}$ units of data samples uploaded in the beginning of time slot t.

Output: A solution to maximize the total utility gain of synchronization requests at time slot t.

```
1:  Create a cloudlet $v_0$ with computing capacity $cap_{v_0} = \sum_{v \in V} cap_v$;
2:  for each synchronization request $r_{n,t} \in R_t$ do
3:      for each $k \in [1, a_{n,t}]$ do
4:          Create an item $i_{n,k}$, with a profit $p_{n,k,t}$ defined in Eq. (15) and a size $\delta_{n,t}$;
5:      end for
6:  end for
7:  Obtain an approximate solution $\{y_{n,k}(t) \mid \forall n \in \mathcal{N}(R_t), \forall k \in [1, a_{n,t}]\}$ to the problem when there is a single cloudlet $v_0$ only, by invoking the approximation algorithm in [14];
8:  Find a solution $\mathbb{A}$ by setting the number $b_{n,t}$ of units of retraining data samples for each object $n \in \mathcal{N}(R_t)$ with $b_{n,t} = \sum_{k=1}^{a_{n,t}} y_{n,k}(t)$;
9:  Sort cloudlets in $V$ in non-increasing order of their computing capacity;
10: Sort requests in $\mathbb{A}$ in non-increasing order of their $b_{n,t}$;
11: $\mathcal{A}_1 \leftarrow \emptyset$; $\mathcal{A}_2 \leftarrow \emptyset$; $j \leftarrow 1$;
12: for each request $r_{n,t} \in \mathbb{A}$ do
13:     Assign request $r_{n,t}$ to cloudlet $v_j$ with $b_{n,t}$ units of retraining data samples;
14:     if cloudlet $v_j$ has its computing capacity violated then
15:         $\mathcal{A}_2 \leftarrow \mathcal{A}_2 \cup \{r_{n,t}\}$; $j \leftarrow j+1$;
16:     else if cloudlet $v_j$ has no residual computing resource then
17:         $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \{r_{n,t}\}$; $j \leftarrow j+1$;
18:     else
19:         $\mathcal{A}_1 \leftarrow \mathcal{A}_1 \cup \{r_{n,t}\}$;
20:     end if;
21: end for
22: if $\sum_{r_{n,t} \in \mathcal{A}_1} h_{n,t}(b_{n,t}) > \sum_{r_{n,t} \in \mathcal{A}_2} h_{n,t}(b_{n,t})$ then
23:     return $\mathcal{A}_1$;
24: else
25:     return $\mathcal{A}_2$;
26: end if.
```

The proposed approximation algorithm for the special static digital twin synchronization problem is detailed in Algorithm 2.

5.3 Algorithm analysis

The rest is to analyze the performance of Algorithm 2, in terms of its approximation ratio and time complexity.

Lemma 1. Given the utility gain $p_{n,k,t}$, which is defined in Eq. (15), for utilizing the kth unit of retraining data samples for request $r_{n,t}$, we have $p_{n,k,t} \geq p_{n,k',t}$ with $1 \leq k < k' \leq a_{n,t}$.

Proof By Eq. (15), we have

$$p_{n,k,t} - p_{n,k',t} = h_{n,t}(k) - h_{n,t}(k-1) - (h_{n,t}(k') - h_{n,t}(k'-1)). \quad (19)$$

Recall that the utility gain function $h_{n,t}(\cdot)$ is assumed to be submodular, therefore, we have $p_{n,k,t} \geq p_{n,k',t}$ with $1 \leq k < k' \leq a_{n,t}$. ■

Lemma 2. Given an ILP solution (16) to the problem when there is only a single cloudlet v_0, i.e., $\{y_{n,k}(t) \mid \forall n \in \mathcal{N}(R_t), \forall k \in [1, a_{n,t}]\}$, let $b_{n,t}$ be the number of units of retraining data samples for synchronization request $r_{n,t}$ by the ILP solution (16), i.e., $b_{n,t} = \sum_{k=1}^{a_{n,t}} y_{n,k}(t)$. Then, for each synchronization request $r_{n,t}$ with $n \in \mathcal{N}(R_t)$, its utility gain by the ILP solution (16) is $h_{n,t}(b_{n,t})$, i.e., $\sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) = h_{n,t}(b_{n,t})$.

Proof Assuming that $b_{n,t} = 0$, i.e., $y_{n,k}(t) = 0, \forall k \in [1, a_{n,t}]$, the utility gain of request $r_{n,t}$ is $\sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) = h_{n,t}(0) = 0$. This means that synchronization request $r_{n,t}$ is rejected.

We now assume that $b_{n,t} = 1$, i.e., it exists a value of k with $1 \leq k \leq a_{n,t}$, such that $y_{n,k}(t) = 1$, while $y_{n,k'} = 0, \forall k' \in [1, a_{n,t}]$ with $k' \neq k$. We claim that $p_{n,k,t} = p_{n,1,t} = h_{n,t}(1)$ by distinguishing it into two cases.

Case 1. If $p_{n,k,t} > p_{n,1,t}$, by Lemma 1, we can see $p_{n,k,t} \leq p_{n,1,t}$ because $1 \leq k \leq a_{n,t}$, which results in a contradiction.

Case 2. Otherwise ($p_{n,k,t} < p_{n,1,t}$), we have that $k > 1$ by Lemma 1, i.e., the kth unit of data samples is input for its model retraining with $k > 1$, instead of the first one. The computing resource demands of each unit of retraining data samples of $r_{n,t}$ are identical at time slot t, i.e., $\delta_{n,t}$. Then with $p_{n,k,t} < p_{n,1,t}$, the ILP solution (16) can always replace the kth unit of data samples with the first one to obtain a higher utility gain without violating any resource capacity, i.e., set $y_{n,1}(t) = 1$ and $y_{n,k}(t) = 0$. This results in a contradiction. We thus have $p_{n,k,t} = p_{n,1,t} = h_{n,t}(1)$, i.e., $\sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) = p_{n,1,t} = h_{n,t}(1)$.

We can show that for any $b_{n,t}$ with $0 \leq b_{n,t} \leq a_{n,t}$, $\sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) = \sum_{k=1}^{b_{n,t}} p_{n,k,t}$ similarly. Recall that $h_{n,t}(0) = 0$. By the definition (15) of $p_{n,k,t}$,

$$\begin{aligned}\sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) &= \sum_{k=1}^{b_{n,t}} p_{n,k,t} \\ &= h_{n,t}(1) - h_{n,t}(0) + h_{n,t}(2) - h_{n,t}(1) \\ &+ \cdots + h_{n,t}(b_{n,t}) - h_{n,t}(b_{n,t}-1) = h_{n,t}(b_{n,t}). \quad (20)\end{aligned}$$

The lemma then follows. ■

Lemma 3. The ILP formulation (16) delivers an optimal solution to the special static digital twin synchronization problem where it exists a single cloudlet v_0 in an MEC network.

Proof Given an ILP solution (16) to the problem of concern where it exists a single cloudlet v_0, i.e., $\{y_{n,k}(t) \mid \forall n \in \mathcal{N}(R_t), \forall k \in [1, a_{n,t}]\}$, and $b_{n,t}$ the number of units of retraining data samples for each request $r_{n,t}$ by the ILP solution, we have that the solution value of the ILP is $\sum_{n \in \mathcal{N}(R_t)} \sum_{k=1}^{a_{n,t}} p_{n,k,t} \cdot y_{n,k}(t) = \sum_{n \in \mathcal{N}(R_t)} h_{n,t}(b_{n,t})$ by Lemma 2. Thus, the lemma follows. ■

Theorem 3. Given an MEC network $G = (V, E)$, a single time slot t, a set N of physical objects, and a set R_t of synchronization requests, each synchronization request $r_{n,t} \in R_t$ of object n has $a_{n,t}$ units of data samples uploaded at the time slot, and the utility gain function $h_{n,t}(\cdot)$. There is an approximation algorithm with the approximation ratio of $(\frac{1-\epsilon}{2})$, `Algorithm` 2, for the special static digital twin synchronization problem. The algorithm takes $O(\sum_{n \in \mathcal{N}(R_t)} a_{n,t} \cdot \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4} + |R_t| \cdot \log |R_t| + |V| \cdot \log |V|)$ time, where ϵ is a constant with $0 < \epsilon < 1$, and $\sum_{n \in \mathcal{N}(R_t)} a_{n,t}$ is the total number of units of data samples of all synchronization requests.

Proof Denote by OPT_1 the optimal solution to the special static digital twin synchronization problem. Denote by OPT_2 the optimal solution to the special problem when there is only a single cloudlet v_0 with capacity $cap_{v_0} = \sum_{v \in V} cap_v$. We have $OPT_2 \geq OPT_1$.

Recall that $\mathbb{A}$ is the set of synchronization requests with the determined number $b_{n,t}$ of units of retraining data samples for each request by `Algorithm` 2 for the special problem where it exists a single cloudlet v_0 in the MEC network. Denote by $U(\mathbb{A})$ the total utility gain from synchronization requests in $\mathbb{A}$. We have $U(\mathbb{A}) \geq (1-\epsilon) \cdot OPT_2$, where $\mathbb{A}$ with an approximation ratio of $(1-\epsilon)$ is obtained, by invoking the approximation algorithm [14], where ϵ is a given constant with $0 < \epsilon < 1$. We then have $U(\mathbb{A}) \geq (1-\epsilon) \cdot OPT_1$ due to that $OPT_2 \geq OPT_1$. We further divide set $\mathbb{A}$ into two disjoint subsets $\mathcal{A}_1$ and $\mathcal{A}_2$, and denote by $U(\mathcal{A}_1)$ and $U(\mathcal{A}_2)$ the utility gain derived from $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. We choose the set between $\mathcal{A}_1$ and $\mathcal{A}_2$ with a larger utility gain, we then have

$$\max\{U(\mathcal{A}_1), U(\mathcal{A}_2)\} \geq \frac{1}{2} \cdot U(\mathbb{A}) \geq \frac{1-\epsilon}{2} \cdot OPT_1. \quad (21)$$

Recall that a synchronization request $r_{n,t}$ is added to set $\mathcal{A}_1$ if its assignment to a cloudlet causes no computing capacity violation. Meanwhile, up to one request is added to set $\mathcal{A}_2$ for each cloudlet $v \in V$, because cloudlet v is excluded from implementing any more request, if its computing capacity is violated with the assignment of request $r_{n,t}$. Also, each cloudlet is considered to possess sufficient computing resource to process model retraining of any request $r_{n,t}$ with $a_{n,t}$ units of data samples. Therefore, assigning the requests in either $\mathcal{A}_1$ or $\mathcal{A}_2$ to cloudlets V will not cause any computing resource violation.

The rest is to analyze the time complexity of `Algorithm` 2. It invokes algorithm [14] that takes $O(\sum_{n \in \mathcal{N}(R_t)} a_{n,t} \cdot \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4})$ time. Sorting all requests in $\mathbb{A}$ takes $O(|R_t| \cdot \log |R_t|)$ time, and sorting cloudlets takes $O(|V| \cdot \log |V|)$ time. The proposed algorithm thus takes $O(\sum_{n \in \mathcal{N}(R_t)} a_{n,t} \cdot \log \frac{1}{\epsilon} + \frac{1}{\epsilon^4} + |R_t| \cdot \log |R_t| + |V| \cdot \log |V|)$ time. ■

6 Online algorithm for the dynamic digital twin synchronization problem

In this section, we study the dynamic digital twin synchronization problem by incorporating system dynamics. For a given time horizon divided into equal time slots, a set of synchronization requests is issued in the beginning of each time slot without any knowledge of future synchronization requests. Each synchronization request $r_{n,t}$ has $a_{n,t}$ units of newly uploaded data samples for retraining, along with a utility gain function $h_{n,t}(\cdot)$ based on the current DNN model retrained in previous time slots. We devise an efficient online algorithm for the problem as follows.

The general strategy is that an *initial solution* to maximize the total utility gain at time slot t is obtained, by invoking `Algorithm` 1. However, this solution is likely to cause resource violation by Theorem 2. If it does not cause any resource violation, it is the final solution to schedule synchronization requests at time slot t. Otherwise, this initial solution needs to be refined in order to become a feasible solution for the problem, which is given as follows.

Denote by $\mathcal{R}_{v,t}^{initial}$ the set of synchronization requests assigned to cloudlet $v \in V$ at time slot t by the initial solution, and for each request $r_{n,t} \in \mathcal{R}_{v,t}^{initial}$, let $b_{n,t}$ be its determined number of units of retraining data samples with $1 \leq b_{n,t} \leq a_{n,t}$. We define the *marginal utility loss* $\Delta\psi_{n,t}(b_{n,t})$ of reducing one unit of retraining data samples for request $r_{n,t}$ with $b_{n,t} \geq 1$ as follows.

$$\Delta\psi_{n,t}(b_{n,t}) = h_{n,t}(b_{n,t}) - h_{n,t}(b_{n,t}-1). \tag{22}$$

We remove the resource violation in cloudlets one by one. If the capacity of cloudlet $v \in V$ is not violated, we do nothing. Otherwise, we reduce the number of units of retraining data samples for requests in $\mathcal{R}_{v,t}^{initial}$ one by one until there is no computing capacity violation on cloudlet v anymore. Within each round, we first calculate the marginal utility loss $\Delta\psi_{n,t}(b_{n,t})$ of each request $r_{n,t} \in \mathcal{R}_{v,t}^{initial}$ with $b_{n,t} \geq 1$. We then reduce one unit of retraining data samples for request $r_{n',t}$ with the least marginal utility loss, i.e., $\Delta\psi_{n',t}(b_{n',t}) = \min\{\Delta\psi_{n,t}(b_{n,t}) \mid r_{n,t} \in \mathcal{R}_{v,t}^{initial}\}$, while replacing $b_{n',t}$ by $b_{n',t}-1$. We will remove request $r_{n',t}$ from $\mathcal{R}_{v,t}^{initial}$ if $b_{n',t}=0$, i.e., request $r_{n',t}$ is rejected. This procedure continues until no capacity violation on cloudlet v happens. The solution obtained is the final solution to the problem at time slot t.

The detailed online algorithm for the dynamic digital twin synchronization problem is in `Algorithm` 3.

We now claim that the proposed online algorithm, `Algorithm` 3, delivers a feasible solution to the dynamic digital twin synchronization problem. It can be seen that an initial solution of `Algorithm` 3 at each time slot $t \in T$ is obtained through invoking `Algorithm` 1. We observe that this initial solution may cause resource capacity violations on cloudlets by Theorem 2. To avoid any violation by refining the solution, we can reduce the number of retraining data samples of synchronization requests assigned to a cloudlet iteratively until no capacity violation in any cloudlet exists. Therefore, `Algorithm` 3 delivers a feasible online solution to the dynamic digital twin synchronization problem.

Algorithm 3 Online algorithm for the dynamic digital twin synchronization problem

Input: Given an MEC network $G=(V,E)$, a finite time horizon T, a set N of physical objects, a set R_t of synchronization requests at each time slot $t \in T$ without any future knowledge, each request $r_{n,t} \in R_t$ has $a_{n,t}$ units of data samples uploaded at each time slot t with the utility gain function $h_{n,t}(\cdot)$.

Output: Maximize the accumulative utility gain of synchronization requests during T.

```
1:  for each time slot $t \in T$ do
2:      Obtain the initial solution via invoking Algorithm 1 at time slot $t$;
3:      for each cloudlet $v \in V$ do
4:          while cloudlet $v$ has its computing capacity violated do
5:              for each request $r_{n,t} \in \mathcal{R}_{v,t}^{initial}$ with $b_{n,t} \geq 1$ do
6:                  Calculate the marginal utility loss $\Delta\psi_{n,t}(b_{n,t})$ of the request;
7:              end for
8:              Identify request $r_{n',t} \in \mathcal{R}_{v,t}^{initial}$ with the least marginal utility loss $\Delta\psi_{n',t}(b_{n',t})$;
9:              Update $b_{n',t} \leftarrow b_{n',t} - 1$;
10:             if $b_{n',t} = 0$ then
11:                 $\mathcal{R}_{v,t}^{initial} \leftarrow \mathcal{R}_{v,t}^{initial} \setminus \{r_{n',t}\}$
12:             end if;
13:         end while
14:     end for
15: end for
16: return An online solution to the dynamic digital twin synchronization problem;
```

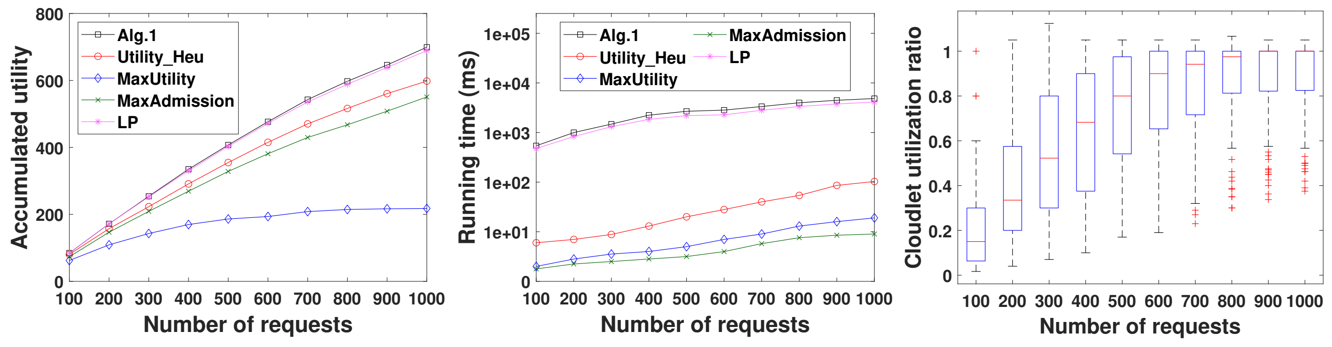
7 Performance Evaluation

In this section, we evaluated the performance of the proposed algorithms for the problems of digital twin synchronization in an MEC network. We also investigated the impacts of important parameters on the performance of the proposed algorithms.

7.1 Experimental Settings

We consider a 1×1 km^2 geographical area [41], in which there are 100 cloudlets deployed to form a 10×10 grid MEC network [19]. Each cloudlet is co-located with an AP, and the maximum transmission distance of each AP is 150m [41]. Each cloudlet has a computing capacity ranging from $1,000$MHz to $5,000$MHz [45]. There are $1,000$ physical objects randomly scattered over the defined geographical area, and each physical object can upload its data to a cloudlet for establishing its digital twin within the maximum transmission distance of the AP co-located with the cloudlet. The digital twin of each physical object is placed in its nearest cloudlet.

There are 10 time slots. With regard to the data drift, the tasks of digital twins are to continually learn to recognize the handwritten digits in the dataset MNIST [16]. Each physical object $n \in N$ uploads $a_{n,t}$ units of data samples in the beginning of each time slot t, where $a_{n,t}$ is randomly set within $[4, 10]$, and we set a unit of data samples as a minibatch containing 128 samples from the MNIST dataset. The computing resource consumption for model retraining



(a) Accumulative utility (b) Running times (c) Cloudlet utilization ratios by Alg.1

Fig. 2. Performance of different algorithms for the static digital twin synchronization problem.

with one unit of data samples ranges from 50MHz to 150MHz. The model accuracy $Accuracy(n, t, b_{n,t})$ in Eq. (1) is obtained by invoking the continual learning algorithm LwF [24] on an open-source continual learning framework (Avalanche [30]). The framework is based on a Python emulator and implemented using PyTorch. The neural network for each digital twin is set up as a Multilayer Perceptron (MLP) with three hidden layers, where the hidden layer size is 256, and the drop rate is 0.5. For each model retraining, the number of training epochs is 10, and the learning rate is 0.001. The parameter ϵ for `Algorithm` 2 is set as 0.5 by Theorem 3.

To evaluate the performance of `Algorithm` 1, referred to as `Alg.1`, for the static digital twin synchronization problem, we consider four benchmark algorithms as follows.

`Utility_Heu`: It is a utility-based heuristic algorithm in [2] to maximize the total utility, through jointly considering selecting services, allocating resources and offloading tasks. Following the spirit of [2], `Utility_Heu` first assigns synchronization requests to cloudlets by retraining models of digital twins with all their data samples, allowing capacity violations. It then reduces the number of units of retraining data samples for synchronization requests with the least *resource efficiency* one by one until the capacities of cloudlets are not violated, where the resource efficiency of reducing a unit of retraining data samples from synchronization request $r_{n,t}$ with $b_{n,t}$ (≥ 1) units of data samples is $\frac{\Delta\psi_{n,t}(b_{n,t})}{\delta_{n,t}}$, i.e., the ratio of the marginal utility loss defined in Eq. (22) to its consumed computing resource.

`MaxUtility`: It assigns synchronization requests to cloudlets one by one iteratively, in which a synchronization request is admitted if its model is retrained with all its uploaded data samples to obtain the maximum utility gain of the request. Then it assigns the synchronization request to a random cloudlet with sufficient residual computing resource. This procedure continues until no more synchronization request can be assigned to any cloudlet.

`MaxAdmission`: It aims to maximize the number of admitted synchronization requests, through assigning as many synchronization requests as possible to cloudlets. Especially, it first assumes each request only utilizes one unit of retraining data samples, and assigns requests one by one to a random cloudlet with sufficient residual computing resource iteratively. If it cannot admit any more request, the procedure terminates; otherwise, it augments the utility gain of requests, by considering cloudlets one by one. That is, for each cloudlet with sufficient residual computing resource, it iteratively augments the utility gain of a random request assigned to the cloudlet with as much retraining data of the request as possible, until the cloudlet has no more sufficient residual computing resource for any such utility augmentation.

`LP`: It is the Linear Program solution (2), which is a relaxation of the ILP solution to the static digital twin synchronization problem, as an upper bound on its optimal solution.

Recall that we assume each synchronization request $r_{n,t}$ can be offloaded to any cloudlet, i.e., $\mathcal{V}(n) = V$, in the special static digital twin synchronization problem. To evaluate the performance of `Algorithm` 2, referred to as `Alg.2`, for the special static digital twin synchronization problem, we consider four benchmark algorithms: `Utility_Heu_s`, `MaxUtility_s`, `MaxAdmission_s` and `LP_s`, which are derived from `Utility_Heu`, `MaxUtility`, `MaxAdmission` and `LP`, respectively, by letting $\mathcal{V}(n) = V$.

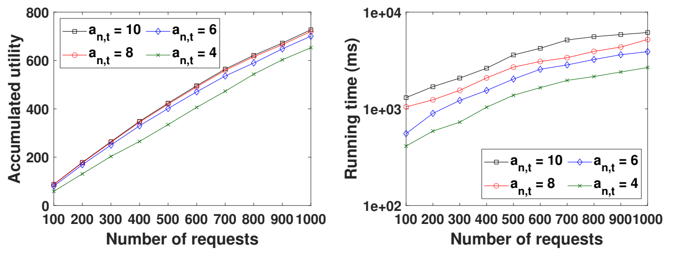
To evaluate the performance of `Algorithm` 3, referred to as `Alg.3`, for the dynamic digital twin synchronization problem, we consider three benchmark algorithms: `Utility_Heu_on`, `MaxUtility_on` and `MaxAdmission_on`, by invoking `Utility_Heu`, `MaxUtility` and `MaxAdmission` and at each time slot, respectively.

The result in each figure is the mean of the results of 15 different MEC instances with the same network size. The running time of each algorithm is based on a desktop with an Octa-Core Intel(R) Xeon(R) CPU @ 2.30GHz, 8G RAM. It is noted that the running time of algorithms shown in the figures does not include the model retraining time. Unless otherwise specified, we adopt the above-mentioned parameters by default.

7.2 Algorithm performance for the static digital twin synchronization problem

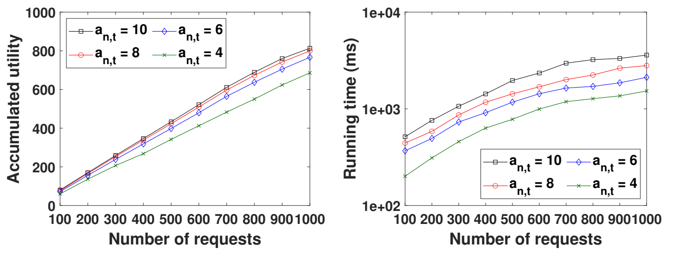
We first evaluated the performance of algorithm `Alg.1` for the static digital twin synchronization problem against algorithms `Utility_Heu`, `MaxUtility`, `MaxAdmission` and `LP`, via varying the number of synchronization requests from 100 to 1,000.

Fig. 2 plots the performance of different algorithms, where the cloudlet utilization ratio is the amount of computing resource consumed of a cloudlet to its capac-



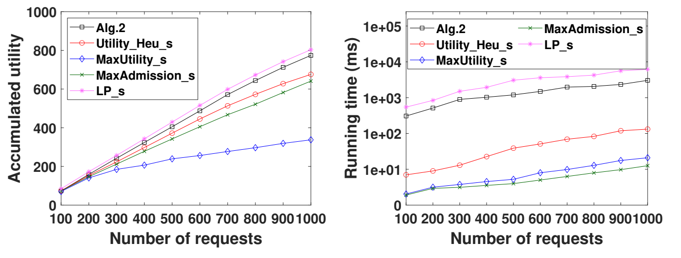
(a) Accumulative utility (b) Running times

Fig. 3. Performance of Alg.1 for the static digital twin synchronization problem by varying the number $a_{n,t}$ of units of uploaded data samples of each request.



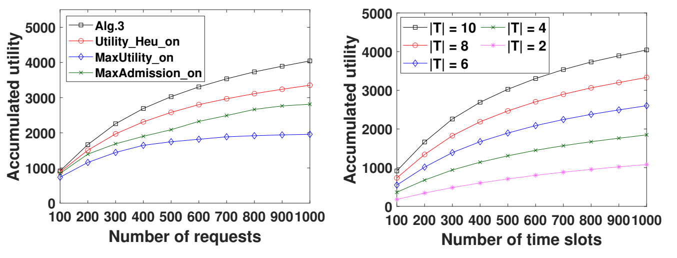
(a) Accumulative utility (b) Running times

Fig. 5. Performance of Alg.2 for the special static digital twin synchronization problem by varying the number $a_{n,t}$ of units of uploaded data samples of each request.



(a) Accumulative utility (b) Running times

Fig. 4. Performance of different algorithms for the special static digital twin synchronization problem.



(a) Performance of different algorithms for the problem with 10 time slots (b) Performance of Alg.3 for the problem by varying the number of time slots

Fig. 6. Performance of different algorithms for the dynamic digital twin synchronization problem.

ity. Fig. 2(a) shows that algorithm Alg.1 achieves near-optimal performance, referring to LP. When the number of synchronization requests reaches $1,000$, the accumulative utility by MaxUtility is 31.1% of that by Alg.1, while Alg.1 outperforms Utility_Heu by 16.9% and outperforms MaxAdmission by 27.1%, respectively. From Fig. 2(b), it can be observed that Alg.1 takes the longest running time, because of solving LP (2) and conducting randomized rounding. We can observe from Fig. 2(c) that the actual capacity violation on a cloudlet is no greater than 12.4%, which is moderate. Fig. 2 indicates that algorithm Alg.1 is promising, compared with the benchmarks, because Alg.1 determines the number of units of retraining data samples for each request efficiently, preserving the synchronization of digital twins and their objects.

The rest is to investigate the impact of the number $a_{n,t}$ of units of uploaded data samples of each request on the performance of algorithm Alg.1, by setting $a_{n,t} = 4$, 6, 8, and 10, respectively. Fig. 3(a) demonstrates the accumulative utility of Alg.1 when $a_{n,t} = 10$ is 11.4% higher than that by itself when $a_{n,t} = 4$ with 1,000 synchronization requests. The justification is that more data samples uploaded by objects can leverage the synchronizations between digital twins and their objects, improving digital twin service quality. Fig. 3(b) shows that algorithm Alg.1 when $a_{n,t} = 10$ takes the longest running time, because more data samples of each synchronization request are to be considered for its model retraining.

7.3 Algorithm performance for the special static digital twin synchronization problem

We then studied the performance of algorithm Alg.2 for the special static digital twin synchronization problem against algorithms Utility_Heu_s, MaxUtility_s, MaxAdmission_s and LP_s, via varying the number of synchronization requests from 100 to $1,000$.

It can be seen from Fig. 4(a) that when the number of synchronization requests is $1,000$, the performance by Alg.2 is 96.4% of that by LP_s, while the performance by MaxUtility_s is 43.6% of that by Alg.2. Also, Alg.2 outperforms Utility_Heu_s by 13.2% and outperforms MaxAdmission_s by 20.7%, with $1,000$ requests, respectively. Fig. 4(b) illustrates that Alg.2 takes a longer running time than Utility_Heu_s, MaxUtility_s and MaxAdmission_s, due to invoking the approximation algorithm in [14]. Fig. 4 depicts that algorithm Alg.2 is promising, and the rationale behind is that Alg.2 efficiently utilizes the computing resource in all cloudlets to facilitate the model retraining of different synchronization requests.

We also studied the impact of the number $a_{n,t}$ of units of uploaded data samples of each request on the performance of Alg.2, by setting $a_{n,t} = 4, 6, 8$ and 10, respectively. Fig. 5 demonstrates the accumulative utility and running times of algorithm Alg.2, through varying the values of $a_{n,t}$. Considering there are $1,000$ synchronization requests, the results in Fig. 5(a) show that the accumulative utility by Alg.2 when $a_{n,t} = 4$ is 84.3% of that of itself when $a_{n,t} = 10$. This is because Alg.2 can make use of more data samples of objects for model retraining.

7.4 Algorithm performance for the dynamic digital twin synchronization problem

We finally investigated the performance of algorithm Alg.3 for the dynamic digital twin synchronization problem against algorithms Utility_Heu_on, MaxUtility_on and MaxAdmission_on, via varying the number of synchronization requests.

When there are $1,000$ synchronization requests, Fig. 6(a) indicates that the performance of `MaxUtility_on` is 48.4% of that of `Alg.3`, the performance of `MaxAdmission_on` is 69.6% of that of `Alg.3`, and the performance of `Utility_Heu_on` is 88.3% of that of `Alg.3`. This can be justified by that algorithm `Alg.3` delivers an efficient online method for digital twin synchronization via continual learning during the monitoring time horizon without any future knowledge. Fig. 6(b) plots the performance curves of `Alg.3` when the number $|T|$ of time slots are 2, 4, 6, 8 and 10, respectively. It can be observed from the figure that the accumulative utility by `Alg.3` when $|T| = 2$ is 26.8% of that by itself when $|T| = 10$, if there are 1,000 synchronization requests. The reason is that `Alg.3` dynamically allocates resources for efficient digital twin synchronization during the given time horizon, and the accumulative utility increases with the growth of the number of time slots.

8 Conclusion

In this paper, we investigated digital twin synchronization in MEC environments via continual learning. We formulated two novel digital twin synchronization problems and showed their NP-hardness. For the static digital twin synchronization problem, we first provided an ILP solution and then proposed a randomized algorithm at the expense of a moderate resource violation. We also proposed an approximation algorithm with a provable approximation ratio for the special static digital twin synchronization problem. We thirdly developed an online algorithm for the dynamic digital twin synchronization problem, considering system dynamics. We finally evaluated the performance of the proposed algorithms via simulations. Simulation results demonstrate the proposed algorithms are promising, improving the algorithm performance by no less than 13.2%, compared with those corresponding benchmarks.

Acknowledgement

The authors appreciate the three anonymous referees and the Associate Editor for their constructive comments and invaluable suggestions, which help us to improve the quality and presentation of the paper greatly. The work by Jing Li and Song Guo was funded by Hong Kong RGC Research Impact Fund (RIF) with the Project No. R5060-19, General Research Fund (GRF) with the Project No. 152221/19E and 15220320/20E, the National Natural Science Foundation of China (61872310), and Shenzhen Science and Technology Innovation Commission (R2020A045). The work by Weifa Liang was supported by the City University of Hong Kong (Grant No. 9380137 and 7005845). The work by Jianping Wang was supported by a grant from Hong Kong Research Grant Council under NSFC/RGC N_CityU 140/20. The work by Yue Zeng and Baoliu Ye was supported by the National Natural Science Foundation of China under Grant 61832005, and the Future Network Scientific Research Foundation Project (FNSRFP-2021-ZD-7).

References

[1] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica. Ekya: continuous learning of video analytics models on edge compute servers. *USENIX NSDI'22*, pp. 119 – 135, 2022.

[2] W. Chu, P. Yu, Z. Yu, J. C. S. Lui, and Y. Lin. Online optimal service selection, resource allocation and task offloading for multi-access edge computing: a utility-based approach. *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4150 – 4167, 2023.

[3] G. Demosthenous and V. Vassiliades. Continual learning on the edge with tensorflow lite. *arXiv preprint arXiv:2105.01946*, 2021.

[4] S. Duan, F. Lyu, H. Wu, W. Chen, H. Lu, Z. Dong, and X. Shen. MOTO: mobility-aware online task offloading with adaptive load balancing in small-cell MEC. To appear in *IEEE Transactions on Mobile Computing*, 2022, doi: 10.1109/TMC.2022.3220720.

[5] C. Gehrmann and M. Gunnarsson. A digital twin based industrial automation and control system security architecture. *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 669 – 680, 2020.

[6] Y. Han, D. Niyato, C. Leung, D. I. Kim, K. Zhu, S. Feng, X. Shen, and C. Miao. A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse. *IEEE Internet of Things Journal*, vol. 10, no. 1, pp. 268 – 284, 2023.

[7] O. Hashash, C. Chaccour, and W. Saad. Edge continual learning for dynamic digital twins over wireless networks. *2022 IEEE 23rd International Workshop on Signal Processing Advances in Wireless Communication (SPAWC)*, 2022.

[8] Y. He, M. Yang, Z. He, and M. Guizani. Resource allocation based on digital twin-enabled federated learning framework in heterogeneous cellular network. *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1149 – 1158, 2023.

[9] H. Huang, L. Yang, Y. Wang, X. Xu, and Y. Lu. Digital twin-driven online anomaly detection for an automation system based on edge intelligence. *Journal of Manufacturing Systems*, vol 59, pp. 138 – 150, 2021. ,

[10] Y. Huang, Y. Zhu, X. Fan, X. Ma, F. Wang, J. Liu, Z. Wang, and Y. Cui. Task scheduling with optimized transmission time in collaborative cloud-edge learning. *Proc of ICCCN'18*, IEEE, 2018.

[11] L. Jia, Z. Zhou, F. Xu, and H. Jin. Cost-efficient continuous edge learning for artificial intelligence of things. *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7325 – 7337, 2022.

[12] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. Characterising the digital twin: a systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36 – 52, 2020.

[13] M. Khani, G. Ananthanarayanan, K. Hsieh, J. Jiang, R. Netravali, Y. Shu, M. Alizadeh, and V. Bahl. RECL: responsive resource-efficient continuous learning for video analytics. *USENIX NSDI'23*, pp. 917 – 932, 2023.

[14] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, vol. 4, no. 4, pp. 339 – 356, 1979.

[15] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366 – 3385, 2022.

[16] Y. LeCun. The MNIST database of handwritten digits. 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[17] 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, 2022.

[18] 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.

[19] 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.

[20] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, and J. Zhao. Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 5, pp. 1199 – 1212, May, 2022.

[21] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, A. Y. 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

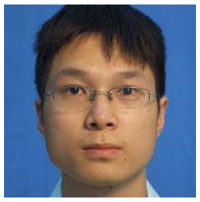
[22] J. Li, W. Liang, W. Xu, Z. Xu, Y. Li, and X. Jia. Service home identification of multiple-source IoT applications in edge computing. *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1417 – 1430, 2023.
[23] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya. Digital twin-enabled service satisfaction enhancement in edge computing. *Proc of INFOCOM'23*, IEEE, 2023.
[24] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935 – 2947, 2018.
[25] W. Liang, Y. Ma, W. Xu, Z. Xu, X. Jia, and W. Zhou. Request reliability augmentation with service function chain requirements in mobile edge computing. *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4541 – 4554, 2022.
[26] S. Lin, W. Liang, and J. Li. Reliability-aware service function chain provisioning in mobile edge-cloud networks. *Proc of 29th ICCCN'20*, IEEE, 2020.
[27] 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.
[28] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2169 – 2182, 2023.
[29] M. Liu, S. Fang, H. Dong, and C. Xu. Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, vol. 58, pp. 346 – 361, 2020.
[30] V. Lomonaco *et al*. Avalanche: an end-to-end library for continual learning. *Proc of IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 3595 – 3605, 2021.
[31] Y. Lu, S. Maharjan, and Y. Zhang. Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16219 – 16230, 2021.
[32] F. Lyu, J. Ren, N. Cheng, P. Yang, M. Li, Y. Zhang, and X. Shen. LeaD: large-scale edge cache deployment based on spatio-temporal wifi traffic statistics. *IEEE Transactions on Mobile Computing*, vol. 20, no. 8, pp. 2607 – 2623, 2021.
[33] MarketsandMarkets. Digital Twin Market by Enterprise, Application (Predictive Maintenance,Business optimization), Industry (Aerospace, Automotive & Transportation, Healthcare, Infrastructure, Energy & Utilities) and Geography - Global Forecast to 2027. 2022. [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/digital-twin-market-225269522.html
[34] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge Univ. Press, 2005.
[35] T. Öncan. A survey of the generalized assignment problem and its applications. *Information Systems and Operational Research*, vol. 45, no. 3, pp. 123 – 142, 2007.
[36] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: a review. *Neural Networks*, vol. 113, pp. 54 – 71, Elsevier, 2019.
[37] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, vol. 7, no. 4, pp. 365 – 374, 1987.
[38] A. Rasheed, O. San, and T. Kvamsdal. Digital twin: values, challenges and enablers from a modeling perspective. *IEEE Access*, vol. 8, pp. 21980 – 22012, 2020.
[39] Y. Sahni, J. Cao, L. Yang, and Y. Ji. Multi-hop multi-task partial computation offloading in collaborative edge computing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133 – 1145, 2021.
[40] 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.
[41] Y. Sun, S. Zhou, and J. Xu. EMM: energy-aware mobility management for mobile edge computing in ultra dense networks. *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637 – 2646, 2017.
[42] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, vol. 94, no. 9, pp. 3563 – 3576, 2018.
[43] W. Wu, P. Yang, W. Zhang, C. Zhou, and X. Shen. Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4988 – 4998, 2021.
[44] Y. Wu, K. Zhang, and Y. Zhang. Digital twin networks: a survey. *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13789 – 13804, 2021.
[45] Z. Xu *et al*. HierFedML: aggregator placement and UE assignment for hierarchical federated learning in mobile edge computing. *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 328 – 345, 2023.
[46] J. Zheng, T. H. Luan, Y. Hui, Z. Yin, N. Cheng, L. Gao, and L. X. Cai. Digital twin empowered heterogeneous network selection in vehicular networks with knowledge transfer. *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 12154 – 12168, 2022.

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

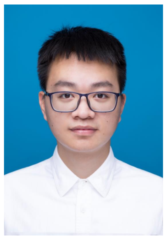
Song Guo (Fellow, IEEE) is a Full Professor in the Department of Computing at The Hong Kong Polytechnic University. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. His research interests are mainly in the areas of big data, edge AI, mobile computing, and distributed systems. Prof. Guo is the Editor-in-Chief of IEEE Open Journal of the Computer Society. He has served on IEEE Communications Society Board of Governors, IEEE Computer Society Fellow Evaluation Committee, and editorial board of a number of prestigious international journals like IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, IEEE Internet of Things Journal, etc. He has also served as chair of organizing and technical committees of many international conferences. Prof. Guo is an IEEE Fellow and an ACM Distinguished Member.

Weifa Liang (Senior Member, IEEE) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in Computer Science. He is a Full Professor in the Department of Computer Science at City University of Hong Kong. Prior to that, he was a Full Professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Mobile Edge Computing (MEC), Network Function Virtualization (NFV), Internet of Things and digital twins, design and analysis of parallel and distributed algorithms, approximation algorithms, and graph theory. He currently serves as an Editor of IEEE Transactions on Communications.

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



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



Yue Zeng received the M.S. degree in the department of electronic information engineering from Southwest University, Chongqing, China, in 2019. He is currently working toward the Ph.D. degree in the department of computer science and technology in Nanjing University, China. His research interests include network functions virtualization, software defined networking, machine learning for networking, distributed computing, and edge computing.

Baoliu Ye (Member, IEEE) is a full professor at Department of Computer Science and Technology, Nanjing University. He received his Ph.D. in computer science from Nanjing University, China in 2004. His current research interests mainly include distributed systems, cloud computing, wireless networks with over 70 papers published in major conferences and journals.

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