Enabling Streaming Analytics for Digital Twin Applications in Mobile Edge Computing Networks

Qiufen Xia, *Member, IEEE,* Peichen Liu, *Student Member, IEEE,* Zichuan Xu, *Member, IEEE,* Jiankang Ren, *Member, IEEE,* Weifa Liang, *Senior Member, IEEE,* Guangyuan Xu, Wenzheng Xu, *Member, IEEE,* Pan Zhou, *Senior Member, IEEE,* Hao Li, and Zhen Feng

***Abstract*—Digital twin is emerging as a key technology to monitor the status of complex industry systems. Valuable insights, such as running statuses and anomalies, can be analyzed from the collected system status timely. Considering that the data updating from each system component (known as a physical object) to its digital twin is performed continuously, timely and accurate streaming analytics based on machine learning models is a key technology to analyze such data efficiently. In this paper, we focus on enabling low-delay yet highly-accurate streaming analytics for digital twin applications in mobile edge computing (MEC) networks. Specifically, we formulate a fundamental optimization problem of digital twin placements and model selections for streaming analytics, with the aim of minimizing both the analytic loss and the processing delay. To this end, we first consider the problem with a single query, for which, we propose an approximation algorithm with provable approximation ratio for a special case, and then devise an efficient algorithm for the original problem with a single query. We then study the online digital twin placement and model selection problem for streaming analytics with multiple queries under real scenarios, where resource demands of arrival queries and resource availability of MEC network are uncertain. We propose an online learning algorithm with a bounded regret to make admission policies. We finally evaluate the performance of the proposed algorithms by extensive simulations. Results show that the weighted sums of the total processing delay and the cumulative loss in the solution delivered by the proposed algorithms outperform their counterparts by 12.5% with a single query and 13.3% with multiple queries, respectively.**

***Index Terms*—Digital twin, streaming analytics, mobile edge computing, approximation, and online learning algorithms.**

The work by Qiufen Xia and Zichuan Xu was supported by the National Natural Science Foundation of China (NSFC) with grant numbers 62572094, 62172071, 62172068, Shandong Provincial Natural Science Foundation (No. ZR2023LZH008, ZR2023LZH013), and Natural Science Foundation of Liaoning (2023-MS-111). The work by Weifa Liang was supported by grants from the RGC of the Hong Kong SAR, China with grant No: CityU 11202723 and CityU 11202824, respectively.

Q. Xia and P. Liu are with the International School of Information Science and Engineering, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, Liaoning, 116024, P. R. China. E-mails: qiufenxia@dlut.edu.cn, liupeichen@mail.dlut.edu.cn.

Z. Xu and G. Xu are with the School of Software, Dalian University of Technology, and the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, Liaoning, 116024, P. R. China. E-mails: z.xu@dlut.edu.cn, 32017112@mail.dlut.edu.cn.

J. Ren is with the School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning, 116024, P. R. China. E-mail: rjk@dlut.edu.cn.

W. Liang is with the Department of Computer Science, City University of Hong Kong, P. R. China. E-mail: weifa.liang@cityu.edu.hk.

W. Xu is with the Department of Computer Science, Sichuan University, Chengdu, Sichuan, P. R. China, 610000. E-mail: wenzheng.xu@scu.edu.cn.

P. Zhou is with the School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, Hubei, P. R. China, 430074. E-mail: panzhou@hust.edu.cn.

H. Li is with Institute of Software, China Coal Research Institute, Beijing, P. R. China. E-mail: hateif@163.com.

Z. Feng is with Jinan Inspur Data Technology Co., Ltd., Jinan, Shandong, P. R. China, 250101. E-mail: fengzh@inspur.com.

I. Introduction

Digital twin technique is emerging as a key enabler for various intelligent industrial applications to reduce operation cost [1]. A digital twin is an intelligent and constantly evolving system that is able to monitor, control, and predict the status of each physical object through its life cycle by dynamically mirroring a physical asset, a process, a system, or an environment to its virtual-world counterpart [2], [3]. For instance, different component digital twins of an autonomous driving vehicle are used to analyze the video streaming data captured by the on-board camera, to enhance the vehicle driving reliability [4]. Digital twins need frequent communications with physical objects to periodically update data and accurately monitor the states of their physical objects. To ensure monitoring timeliness, digital twins need to be deployed to an adjacent node with abundant computing power.

Considering various physical objects, such as vehicles and industrial equipments, are usually located at the network edge. Mobile edge computing (MEC) is envisioned as a viable solution to guarantee timeliness of digital twin applications, by placing digital twins of physical objects into *cloudlets* in the proximity of physical objects [5]. However, physical objects in industrial scenarios are complex with many intertwining subsystems and complex interactions [6]. The subsystems of a physical object usually operate continuously, and digital twins need to reflect and analyze the states of these subsystems in real time. These states are then fed into their digital twins by data streams [7]. To predict future events related to different subsystems, digital twin applications perform analytics on these data streams by machine learning models [8].

In this paper, we study how to enable low-delay yet accurate streaming analytics for digital twin applications in an MEC network. However, achieving this goal poses significant challenges. First, mirroring the subsystems of each physical object and analyzing their data in digital twins requires a huge amount of computing resources in an MEC network [9]. Thus, if a digital twin is placed into a cloudlet with inadequate computing resources, a low resource overhead but high loss machine learning model of the digital twin may be selected to perform streaming analytics [10]. Further, placing the digital

twins of subsystems into a cloudlet with sufficient computing resources to perform streaming analytics may easily push up the updating delays of states if the cloudlet is located at a location far from their physical objects. To select proper models for digital twins, a fine trade-off between available computing resources and the location of physical objects must be found. Thus, it is challenging to jointly select a machine learning model for each digital twin and place the digital twins into resource-restricted cloudlets, such that loss and delays of streaming analytics of digital twins are jointly minimized [11], [12]. Second, digital twin applications must meet the different updating delay requirements of each physical object, ensuring that the streaming analytics can reflect the current states of physical objects and accurately predicts future events. However, the limited computing resources may not host all digital twins of a physical object in a single cloudlet which adjacent to the physical object. Besides, users may simultaneously query multiple digital twins of different physical objects. Given the fact that digital twins of subsystems of a physical object may be placed into multiple cloudlets, the analytical results need to be transmitted to the cloudlet where the user is located for aggregation. If the digital twin is deployed in a cloudlet far from the location of the query, it may cause higher delays due to analytical results transmission. Thus, how to jointly select cloudlets for each digital twin by considering not only the processing delays but also the updating delays is challenging. Third, the MEC network may contain multiple digital twin applications, each of which may concurrently receive multiple queries from users without the knowledge of future arrival times. If queries are highly concurrent in the MEC network, computing resources may become insufficient to implement such highly-concurrently queries. Thus a dynamic admission policy needs to be designed to admit as many queries as possible. Admitting such queries requires to consider multiple factors such as the resource availability, the resource demand of the machine learning model of each digital twin, and so on [13]. How to learn the impact of these factors on admission policies in MEC networks while maximizing the number of query admissions is challenging.

To the best of our knowledge, we are the first to consider the problem of joint machine learning model selections and digital twin placements in an MEC network, such that the weighted sum of loss and processing delay are minimized. Existing studies considered them separately, by focusing on either model selections [11], [14] or digital twin placements [15]. Further, we are the first to consider the fact that a physical object may consist of multiple intertwining subsystems. The main contributions are as follows.

- We formulate the digital twin placement and model selection problems for streaming analytics in an MEC network, with the aim of minimizing the processing delay and model accuracy loss of the digital twin applications.
- For a single query, we propose an approximation algorithm with a provable approximation ratio for a special case of the problem with selected models. Based on the proposed approximation algorithm, we propose an efficient heuristic algorithm to make the model selection and digital twin placement decision jointly.
- For multiple queries, we propose an online learning algorithm for query admissions with a provably bounded regret, by adopting a contextual bandit learning technique.
- We evaluate the performance of the proposed algorithms by simulations. Simulation results show that the proposed algorithms outperform benchmarks, by reducing the weighted sum of loss and delay by 25.3% with a single query and by 13.3% with multiple queries, respectively.

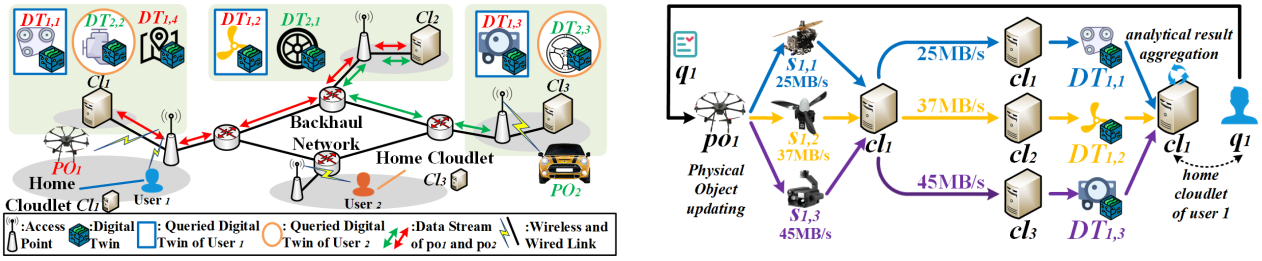
The rest of the paper is organized as follows. Section II surveys related works. Section III describes the system model and formulates the problems. Section IV proposes two algorithms for the problem with a single query. Section V proposes an online learning algorithm for the problem with multiple queries. Section VI evaluates the proposed algorithms, and Section VII concludes this work.

II. Related Work

Digital twin exhibits great potentials in the monitoring of physical systems. Digital twin applications in MEC networks enable real-time situational awareness for physical objects [16], [17], [18], [19]. For example, Gu *et al.* [16] adopted an online learning algorithm for dynamic digital twins placement to minimize delay and energy consumption under cost constraints. Zhang *et al.* [17] investigated an adaptive placement method for digital twins and proposed a game theory-based method to enhance reliability and resource utilization. Li *et al.* [18] proposed a dynamic digital twin placement method for improving user satisfaction on services in MEC. However, these studies did not incorporate the updating delay requirements and processing delay of streaming analytics.

To capture states of physical objects timely, users can issue queries for digital twins. The data analytics of digital twins has been studied in [20], [21]. These studies cannot be directly applied to streaming analytics in the resource-constrained MEC networks. There are several studies on streaming analytics in MEC networks [22], [23], [24]. For example, Fu *et al.* [22] developed a stream processing framework and proposed a method to improve throughput and reduce delay. Liu *et al.* [23] proposed an adaptive edge stream processing engine that enables fast response to users' queries. Ding *et al.* [24] proposed a method for the stateful data stream partitioning to reduce processing delay. However, none of these streaming analytic techniques can track the dynamic state of multiple physical objects simultaneously, and they only consider the accuracy of queries under delay constraints for the digital twin applications.

Most existing studies considered the accuracy provided by digital twin models as an indicator of performance [25]. However, none of the mentioned studies considered digital twin model selection under limited edge computing resources. There are several studies on model selection problems [11], [14], [26], [27], [28]. For example, Zhao *et al.* [11] proposed an online algorithm for the object detection model selection problem by exploring trade-offs among delay, accuracy, and cost. Chen *et al.* [14] focused on the feature accuracy enhancement of digital twins by using model selection under resource constraints. However, these studies may not be directly applied to digital



(a) An example of a DT-MEC network with two users.

(b) The streaming analytics of user1 in the DT-MEC network.

Fig. 1: Examples of a DT-MEC network and streaming analytics. User1 queries three different digital twins of physical object po_1, where po_1 has three subsystems, i.e., $s_{1,1}$, $s_{1,2}$, and $s_{1,3}$. po_1 transmits data streams to its digital twins deployed in cl_2, cl_3, and cl_4. When digital twins $DT_{1,1}$, $DT_{1,2}$, and $DT_{1,3}$ are queried by a user, each digital twin performs streaming analytics and then transmits its analytical result to the home cloudlet of the user for data aggregation.

twin applications in MEC networks that require real-time streaming analytics with stringent delay constraints.

III. Preliminary

We introduce the system model of a digital-twin-enabled MEC (DT-MEC) network with physical objects and digital twins to perform streaming analytics. We also define two optimization problems precisely.

A. System Model

We consider a DT-MEC network $G = \{\mathcal{AP} \cup \mathcal{CL}, E\}$, where $\mathcal{AP}$ is a set of Access Points (APs), $\mathcal{CL}$ is a set of cloudlets that are located in the edge of the backhaul of G, and E is a set of communication links/paths that interconnect the APs. Due to limited space in edge locations, each cloudlet $cl_i \in \mathcal{CL}$ usually contains several servers or accelerators (e.g., FPGA or neural network accelerators) to run digital twin services. Each cloudlet cl_i has a limited amount of computing resource. The amount of available computing resource in cloudlet cl_i at different time slots varies, assuming that time is divided into equal time slots, and let t be one time slot. For clarity, we use $C_{i,t}$ to represent the amount of available computing resource of cloudlet cl_i in the beginning of time slot t. Let ap_q be an AP in $\mathcal{AP}$. Fig. 1(a) is an illustrative example of a DT-MEC network.

B. Physical Objects and Digital Twins

There are a number of physical objects, such as vehicles, AR/VR handsets, industrial equipments, and video cameras, that upload states to their digital twin applications located in the DT-MEC network via its nearest AP. Without loss of generality, we assume that physical objects continuously generate data. It is important to timely process the generated data to unveil valuable hidden patterns. Let po_k be a physical object with $1 \leq k \leq K$. Each physical object has a number of subsystems due to complicated functionalities of many complex industrial systems. For example, a space station may include multiple subsystems such as a thermoregulation subsystem, a barometric pressure balance subsystem, an engine control subsystem, etc. Denote by $\mathcal{S}_k$ and M_k a set of subsystems of physical object po_k and the number of subsystems of the kth physical object, respectively, where $|\mathcal{S}_k| = M_k$. Let $s_{k,m}$ be the mth subsystem of physical object po_k, i.e., $s_{k,m} \in \mathcal{S}_k$.

Due to the complexity of each physical object po_k, we assume that each subsystem of physical object po_k is mirrored as a *digital twin* that is implemented as a piece of software. Let $DT_{k,m}$ be the digital twin of the mth subsystem of physical object po_k. Each digital twin $DT_{k,m}$ accurately monitors the running state of po_k's subsystem $s_{k,m}$, by updating the state of the subsystem to its digital twins. Let $\rho_{k,m}$ be the *data rate* of the state updates. To ensure that $DT_{k,m}$ mirror $s_{k,m}$ of po_k in real time, $DT_{k,m}$ usually is placed into cloudlets close to its physical object po_k. Note that the digital twins of the subsystems of a physical object may be placed into different cloudlets due to limited computing capacity on each cloudlet. Let $x_{k,m,i}$ be a binary variable that indicates whether $DT_{k,m}$ is placed into cloudlet cl_i, assuming that $x_{k,m,i} = 1$ meaning that $DT_{k,m}$ is placed into cloudlet cl_i. It consumes an amount of computing resource of cl_i to mirror and predict the state of $s_{k,m}$. Once placed in a cloudlet, digital twin $DT_{k,m}$ of subsystem $s_{k,m}$ dynamically processes data at rate $\rho_{k,m}$.

C. Streaming Analytics of Digital Twins

Besides updating the state of physical objects in real time, digital twins need to analyze the state of physical objects, using the updated data streams. To this end, users can query different digital twins of physical objects by issuing *queries*. For instance, to pinpoint which physical objects need attention, users issue queries to jointly analyze the data streams collected by different digital twins, based on machine learning models. Fig. 1(b) illustrates an example of streaming analytics in a DT-MEC network. Without loss of generality, we consider that different models can be selected by each digital twin $DT_{k,m}$ to infer its data streams.

Let $\mathcal{R}_t$ be a set of arrived queries in the DT-MEC network at time slot t, and each query needs to be scheduled in the beginning of time slot t. Let q_n be the nth query in $\mathcal{R}_t$. Note that query q_n may need to jointly analyze the digital twins of different physical objects. For example, a user queries the digital twins of a vehicle to access information regarding its positioning, navigation, etc., aiding in autonomous driving decision-making. Denote by $\mathcal{DT}_n$ the digital twins queried by q_n. We assume that each query q_n has a *home cloudlet* to store

its analytical results, which is denoted by $cl(q_n)$. The evaluation of each query q_n involves the processing of the data stream of each digital twin in $\mathcal{DT}$ and the aggregation of intermediate results of data processing. For clarity, we consider that the intermediate result of analyzing a digital twin is transferred to the home cloudlet $cl(q_n)$ for aggregation with the results of the other digital twins in $\mathcal{DT}_n$.

D. Model Selection for Streaming Analytics

To accurately analyze the state of physical objects, each query q_n uses machine learning models to make predictions based on digital twins of the physical objects. However, due to limited capacities on cloudlets, not all machine learning models can be executed in a cloudlet at the same time. We thus assume that there are multiple machine learning models with different accuracy and computing resource demands that can be leveraged to perform streaming analytics. Let $\mathcal{J}_{k,m}$ be the set of machine learning models for digital twin $DT_{k,m}$, where $j_{k,m} \in \mathcal{J}_{k,m}$ represents the jth model of digital twin $DT_{k,m}$. That is, considering the available computing resource of cloudlet at time slot t, a proper machine learning model need to be selected to process the query. Let $y_{k,m,j}$ be a binary variable that represents whether machine learning model $j_{k,m} \in \mathcal{J}_{k,m}$ is selected for digital twins $DT_{k,m}$. Denote by $Loss_{k,m,j}$ the loss of the machine learning model $j_{k,m}$. The loss of the streaming analytics obtained by query q_n is

$$Loss(q_n) = \sum_{DT_{k,m} \in \mathcal{DT}_n} \sum_{j_{k,m} \in \mathcal{J}_{k,m}} y_{k,m,j} \cdot Loss_{k,m,j}. \quad (1)$$

Machine learning models with different losses consume different amounts of computing resources. Let $\eta_{k,m,j}$ be the amount of computing resource allocated to machine learning model $j_{k,m}$ to infer a unit amount of state data of digital twin $DT_{k,m}$. The total amount of computing resource demanded to perform streaming analytics of $DT_{k,m}$ at rate $\rho_{k,m}$ is

$$\sum\nolimits_{j_{k,m} \in \mathcal{J}_{k,m}} y_{k,m,j} \cdot \eta_{k,m,j} \cdot \rho_{k,m}. \quad (2)$$

E. Delay Model

Given the digital twin placements of physical objects, the digital twins' state needs to be updated timely. We here consider that the implementation of an query incurs two types of delays: the state updating delay and the data processing delay, which are defined in the following.

State updating delay: The state updating delay incurs due to the state updating at a rate $\rho_{k,m}$ from the subsystem $s_{k,m}$ of physical object po_k to its digital twin $DT_{k,m}$. Recall that $x_{k,m,i}$ is a binary variable that shows whether digital twin $DT_{k,m}$ is placed into cloudlet cl_i. Assume that physical object po_k accesses the MEC network via AP ap_q, the delay of the state updating of each digital twin is

$$d^{upd}_{k,m} = \sum\nolimits_{cl_i \in \mathcal{CL}} x_{k,m,i} \sum\nolimits_{e \in p_{q,i}} d_e \cdot \rho_{k,m}, \quad (3)$$

where $p_{q,i}$ is the shortest path in the MEC network between AP ap_q and cloudlet cl_i, e is a link in path $p_{q,i}$, and d_e represents the transmission delay of a unit data in link e.

To ensure that the state of each physical object is timely synchronized with its digital twins, the state update must meet the updating delay requirement of each physical object. Otherwise, outdated data streams are processed by streaming analytics, and the analytical results may not reflect the current state of physical objects. Let D^{rqe}_k be the delay requirement of physical object po_k, we have

$$d^{upd}_{k,m} \leq D^{rqe}_k, \forall s_{k,m} \in \mathcal{S}_k. \quad (4)$$

Data processing delay: The processing of data streams in digital twins incurs data processing delays. Following existing studies [29], [30], the processing delay is inversely proportional to the amount of computing resource allocated to each query. The processing delay of a query against a set of digital twins consists of: the inference delay, the transmission delay of the intermediate results, and the aggregation delay, which are defined as follows.

Inference delay: Let $d_{k,m,j,i}$ be the processing delay of a unit data by machine learning model $j_{k,m}$ of digital twin $DT_{k,m}$ located in cloudlet cl_i. We have

$$d^{inf}_{k,m} = \sum_{j_{k,m} \in \mathcal{J}_{k,m}} y_{k,m,j} \sum_{cl_i \in \mathcal{CL}} x_{k,m,i} \cdot d_{k,m,j,i} \cdot \rho_{k,m}. \quad (5)$$

Transmission delay of the intermediate results: The inference results of each digital twin are called intermediate results, which are then transmitted from cloudlet cl_i to the home cloudlet $cl(q_n)$ that incurs the transmission delay. Denote by $p_{i,cl(q_n)}$ the shortest path in the network between cloudlets cl_i and $cl(q_n)$. Following the study in [31], let ra be the ratio of intermediate result to its data stream rate $\rho_{k,m}$. We have

$$d^{tra}_{k,m} = \sum\nolimits_{cl_i \in \mathcal{CL}} x_{k,m,i} \sum\nolimits_{e \in p_{i,cl(q_n)}} d_e \cdot ra \cdot \rho_{k,m}. \quad (6)$$

The aggregation delay: The aggregation of intermediate results in the home cloudlet $cl(q_n)$ of query q_n incurs the aggregation delay. Let d^{agg}_n be the aggregation delay of a unit of intermediate data. Then, the aggregation delay is

$$d^{agg}(q_n) = d^{agg}_n \cdot \sum\nolimits_{DT_{k,m} \in \mathcal{DT}_n} ra \cdot \rho_{k,m}. \quad (7)$$

As digital twins may be placed into different cloudlets, each digital twin can perform inference and transmit intermediate results simultaneously. The overall processing delay of a query is determined by the participating cloudlet with the slowest processing speed. Let $d^{pro}(q_n)$ be the processing delay of q_n, we have

$$d^{pro}(q_n) = \max_{DT_{k,m} \in \mathcal{DT}_n} (d^{inf}_{k,m} + d^{tra}_{k,m}) + d^{agg}(q_n). \quad (8)$$

F. Problem Formulations and NP-Hardness

We consider a DT-MEC network G with digital twin applications and their users issuing queries to predict the status of physical objects. There are K physical objects with each having M subsystems indexed by $S_k = \{s_{k,1}, ..., s_{k,M}\}$. Assume that each subsystem $s_{k,m}$ uploads its data stream at rate $\rho_{k,m}$ to its digital twin $DT_{k,m}$. There is a set $\mathcal{J}_{k,m}$ of machine learning models on digital twins, which are used for analyzing the data streams collected in each digital twin, where

TABLE I: Symbols

Notations	Descriptions
K and M_k	the number of physical objects and the number of subsystems of the kth physical object
$\mathcal{CL}$, cl_i, and $C_{i,t}$	the set of cloudlets, ith cloudlet, and the amount of available computing resource of cl_i at time slot t
$\mathcal{PO}$ and po_k	the set of physical objects, and kth physical object where $po_k \in \mathcal{PO}$
$\mathcal{S}_k$ and $s_{k,m}$	the set of subsystems of physical object po_k, and mth subsystem of po_k where $s_{k,m} \in \mathcal{S}_k$ and $\lvert\mathcal{S}_k\rvert = M_k$
$DT_{k,m}$ and $\rho_{k,m}$	the digital twin of the mth subsystem of physical object po_k and the updating data rate of po_k
$\mathcal{J}_{k,m}$ and $j_{k,m}$	the set of available models for digital twin $DT_{k,m}$, and the jth model of $DT_{k,m}$
$\mathcal{R}_t$ and q_n	the set of queries arrived into the system in time slot t, and the nth query where $q_n \in \mathcal{R}_t$
$\mathcal{DT}_n$ and $cl(q_n)$	the digital twins queried by query q_n and the home cloudlet of nth query q_n
$Loss_{k,m,j}$, $\eta_{k,m,j}$, and η'	the loss of model $j_{k,m}$, the amount of allocated computing resource of model $j_{k,m}$ and aggregation operator
$d^{upd}_{k,m}$, $d^{pro}_{k,m}$, and $d(e)$	the state updating delay, the data processing delay, and the transmission delay of link e for a unit data rate
$d^{inf}_{k,m}$, $d^{tra}_{k,m}$, and $d^{agg}_{k,m}$	the inference delay, the intermediate result transmission delay, and the data aggregation delay
$d(\eta_{k,m,j})$ and $d^{pro}_{max,v}$	the processing delay for a unit data rate and the minimum processing delays of the digital twins in DT^{vio}_k
α and $\mathcal{A}_t$	the ratio between the rate of the data stream to its intermediate result and the set of admitted queries at t
D^{rqe}_k	the state updating delay requirement of physical object po_k
ra	the ratio between the original data stream rate $\rho_{k,m}$ to its intermediate result
$p_{q,i}$ and $p_{i,cl(q_n)}$	the shortest path from AP ap_q to cloudlet cl_i and the shortest path from cloudlet cl_i to the home cloudlet of query q_n
Q and $q_{k,m}$	the number of equal intervals and the current interval of digital twin $DT_{k,m}$
$CL_{q,k,m}$	the set of cloudlets whose updating delay meets the updating requirement and processing delays lie in the interval $q_{k,m}$
$\mathcal{DT}^{vio}_k$ and $DT^{vio}_{k'}$	the set of digital twins of po_k that violate the updating delay constraints, and a digital twin where $DT^{vio}_k \in \mathcal{DT}^{vio}_k$
$LD_{k,m}$ and $LD_{k,m}$	the obtained weighted sum of loss and processing delay of the current iteration and its previous iteration
Θ and θ	the set of contexts and a context where $\theta \in \Theta$
$pr_{\theta,n,t}$ and $pr_{n,t}$	the admission probability of query q_n under the context θ and the admission probability of query q_n
$\omega_{\theta,n,t}$ and $\phi_{\theta,n,t}$	the weight of context θ on request q_n and the penalty coefficient of the online learning

a machine learning model $j_{k,m} \in \mathcal{J}_{k,m}$ needs to be selected to perform streaming analytics. There are multiple queries $\mathcal{R}_t$ arriving concurrently at time slot t. On one hand, the operator of the DT-MEC network needs to meet the stringent requirement on processing delay and loss of every query, such that the best quality of service for each user is guaranteed. On the other hand, the operator also needs to maximize its revenue by admitting as many queries as possible. We thus formulate the following two optimization problems, by firstly considering the delay and cost optimization for a single query and secondly dealing with the admissions of multiple queries.

Problem 1: Given a query q_n at time slot t, the problem of *digital twin placement and model selection for streaming analytics with a single query* in G is to evaluate query q_n, by jointly placing queried digital twins into cloudlets of the MEC network, and selecting a proper machine learning model to analyze the state data of digital twins required by query q_n, such that both the processing delay and the accuracy loss of the selected model are minimized, subject to the computing capacity constraints of MEC network, and the updating delay constraint. The optimization objective is to

$$\textbf{P1:} \min \sum_{cl_i \in \mathcal{CL}} \sum_{DT_{k,m} \in \mathcal{DT}_n} Loss(q_n) + \alpha \cdot d^{pro}(q_n), \quad (9)$$

subject to the following constraints,

$$x_{k,m,i}, y_{k,m,j} \in \{0,1\}, \quad (10)$$

$$\sum\nolimits_{cl_i \in \mathcal{CL}} x_{k,m,i} = 1, \forall DT_{k,m} \in \mathcal{DT}_n, \quad (11)$$

$$\sum\nolimits_{j_{k,m} \in \mathcal{J}_{k,m}} y_{k,m,j} = 1, \forall DT_{k,m} \in \mathcal{DT}_n, \quad (12)$$

$$\sum_{DT_{k,m} \in \mathcal{DT}_n} x_{k,m,i} \sum_{j_{k,m} \in \mathcal{J}_{k,m}} y_{k,m,j} \cdot \eta_{k,m,j} \cdot \rho_{k,m} \leq C_i, \forall cl_i \in \mathcal{CL}, \quad (13)$$

$$d^{upd}_{k,m} \leq D^{rqe}_k, \forall s_{k,m} \in \mathcal{S}_k, \quad (14)$$

where α is a non-negative constant that strives for a fine tradeoff between the processing delay and the model loss. Constraint (10) ensures that $x_{k,m,i}$ and $y_{k,m,j}$ are binary decision variables. Constraints (11) and (12) say that each digital twin can only be deployed to at most one cloudlet and one model is selected. Constraint (13) indicates that the computing capacity on each cloudlet cl_i is not violated. Constraint (14) ensures that the updating delay constraint of each digital twin of q_n is satisfied.

Problem 2: Queries arrive into the system dynamically without the knowledge of future query arrivals. Given a finite time horizon with T time slots, *the online digital twin placement and model selection for streaming analytics with multiple queries* in G is to minimize the average weighted sum of model losses and processing delays of admitted queries by admitting as many queries in $\sum_{t=1}^{T} \mathcal{R}_t$ as possible over the time horizon, through jointly placing digital twins that are required by each admitted query $q_n \in \mathcal{R}_t$ into cloudlets. Then each query is evaluated sequentially in the order of admission. The problem optimization objective is to

$$\textbf{P2:} \min \sum_{t=1}^{T} \sum_{cl_i \in \mathcal{CL}} \sum_{DT_{k,m} \in \mathcal{DT}_n} Loss(q_n) + \alpha \cdot d^{pro}(q_n) - \gamma \cdot \lvert\mathcal{A}_t\rvert, \quad (15)$$

subject to the following constraints (10), (11), (12), (13), and (14), where γ is a scaling factor and $\mathcal{A}_t$ is the set of admitted queries at time slot t. All symbols are listed in Tab. I.

Theorem 1: The defined problems of digital twin placements and model selections for streaming analytics with a single or multiple queries are NP-hard.

Please see **Appendix A**, available in the supplemental file.

IV. Algorithms for the Digital Twin Placement and Model Selection for Streaming Analytics with a Single Query

In the section, we devise approximate and efficient solutions to the problem of the digital twin placement and model selection for streaming analytics with a single query.

A. Overview

An optimal solution to the problem is to jointly place digital twins of the subsystems of each physical object and select proper machine learning models for the placed digital twins. This however poses a great challenge due to the complexity of the problem of concern. We thus adopt a progressive way to jointly make decisions on digital twin placements and model selections. Specifically, we first assume the models for all digital twins are selected, and propose an approximation algorithm for digital twin placements. We then devise another approximation algorithm for the original problem that enables the joint digital twin placements and model selections, by removing the initial assumption.

B. An Approximation Algorithm for the Digital Twin Placement with Selected Models

We devise an approximation algorithm for the problem, assuming that the models of digital twins are selected. Given the selected models, variables $y_{k,m,j}$ are determined. We now determine decision variables $x_{k,m,i}$ by finding proper locations for digital twins of the subsystems of physical object po_k.

A special case of the problem with selected models (i.e., $y_{k,m,j}$ for any $DT_{k,m} \in \mathcal{DT}_n$ is given) and without considering the updating delay constraint (14) is the General Assignment Problem (GAP). We then can remove constraint (14) and find an approximate solution to the problem with selected models and without the updating delay requirements, by applying the approximation algorithm due to [32]. The solution obtained however may not be feasible to the one with selected models, because the updating delay requirements of the placed digital twins may be violated. We then convert the solution into a feasible solution. To this end, we find those placed digital twins that violate the updating delay requirements. Let DT_k^{vio} be the set of digital twins of physical object po_k that violate the updating delay requirements. Recall that the objective of the problem with selected models is to minimize the processing delays of placed digital twins, as the model selection decisions have been made. Consequently, the reason that digital twins in DT_k^{vio} have their updating delay requirement violated may be because that they are placed into cloudlets with low processing delays and high updating delays.

Let $d_{max,v}^{pro}$ be the minimum processing delay of digital twins in $\mathcal{DT}_k^{vio}$. To meet the updating delay requirements of the digital twins in $\mathcal{DT}_k^{vio}$, we carefully tune the updating delay of each digital twin such that its processing delay does not increase too much to meet its updating delay requirement. To this end, we divide the range of $[0, d_{max,v}^{pro}]$ into the number Q of equal intervals. We then find the locations for each digital twin in $\mathcal{DT}_k^{vio}$ according to the current interval of its processing delay. Specifically, let $q_{k,m}$ be the current interval of $DT_{k,m}$ in the range of $[0, d_{max,v}^{pro}]$. Let $\mathcal{CL}_{q,k,m}$ be the set of cloudlets whose updating delay meets the updating delay requirement of $DT_{k,m}$ and their processing delays lie in the interval $q_{k,m}$. We place $DT_{k,m}$ into a cloudlet in $\mathcal{CL}_{q,k,m}$ with the minimum processing delay. Considering that there may not exist any cloudlet that lies in the interval $q_{k,m}$ with adequate computing resource, we proceed to the next interval $(q+1)_{k,m}$. If there are no such cloudlets in the rest of the intervals, we then examine the next interval with less processing delays, i.e., the one that is smaller than interval $q_{k,m}$. This procedure continues until $DT_{k,m}$ is placed into a cloudlet that meets both its updating delay requirement. If there does not exist any cloudlet in all interval Q with adequate computing resources, the digital twin will be placed in the cloudlet with the most available resources that meets the updating delay requirements. The detailed steps of the proposed algorithm `ApproS` are shown in **Algorithm** 1.

Algorithm 1 An Approximation Algorithm for the Digital Twin Placement with Selected Models (`ApproS`).

Require: A DT-MEC Network G, a set $\mathcal{PO}$ of physical objects with each object $po_k \in \mathcal{PO}$ has a set $\mathcal{S}_k$ of subsystem, a set $\mathcal{J}$ of models and model selection decision $y_{k,m,j}$, a query q_n, and a set of home cloudlet $\mathcal{H}(cl_i)$.

Ensure: A digital twin placement strategy $x_{k,m,i}$ for selected models of a single query q_n.

Compute loss function of the selected model by the model selection strategy $y_{k,m,j}$ by Eq. (2).
Get updating and processing delay by Eqs. (3) and (8);
Find an approximate solution by applying an approximation algorithm proposed by Shmoys and Tardos [32];
Evaluate the updating delay violation of digital twins and obtain the minimum processing delays $d_{max,v}^{pro}$;
Divide the processing delay of digital twin $[0, d_{max,v}^{pro}]$ into Q intervals;
for each $DT_{k,m} \in DT_k^{vio}$
  for each $q_{k,m} \in Q_{k,m}$
    for each $CL_{q,k,m} \in \mathcal{CL}_{q,k,m}$
      if $CL_{q,k,m}$ has enough resource and the delay requirement D_k^{rqe} can be met;
        Adjust the deployment location of $DT_{k,m}$ into $CL_{q,k,m}$;
      end if
    end for
  end for
end for
The digital twin is assigned to the cloudlets for streaming analytics and analytical result is transmitted to its home cloudlet.

C. An Efficient Algorithm for the Digital Twin Placement and Model Selection for Streaming Analytics with a Single Query

Having `ApproS`, we now propose an approximation algorithm for the digital twin placement and model selection for streaming analytics with a single query. Recall that we need to select a model $j_{k,m} \in \mathcal{J}_{k,m}$ for each digital twin $DT_{k,m}$ in $\mathcal{DT}_n$. We observe that a model with a lower loss usually requires more computing resources to maintain a rate of data stream processing. This unfortunately may reduce the chances of choosing cloudlets with lower processing delays. We thus adopt binary search to select a proper machine learning model for each digital twin.

The algorithm proceeds iteratively. It first ranks machine learning models $\mathcal{J}_{k,m}$ of each digital twin $DT_{k,m}$ in decreasing order of loss. It then randomly selects a model from $\mathcal{J}_{k,m}$ for digital twin $DT_{k,m}$, by invoking algorithm `ApproS`, assuming

Algorithm 2 An Efficient Algorithm for the Digital Twin Placement and Model Selection for Streaming Analytics with a Single Query `HeuS`.

Require: A DT-MEC Network G, a set $\mathcal{PO}$ of physical objects with each object $po_k \in \mathcal{PO}$ has a set $\mathcal{S}_k$ of subsystem, a set $\mathcal{J}$ of models and model selection decision $y_{k,m,j}$, a query q_n, and a set of home cloudlet $\mathcal{H}(cl_i)$.

Ensure: A digital twin placement strategy $x_{k,m,i}$ and model selection strategy $y_{k,m,j}$ of a single query q_n.

Initialize the number of models $|\mathcal{J}_{k,m}|$;
Set $j_{k,m} \rightarrow \lfloor \frac{|\mathcal{J}_{k,m}|}{2} \rfloor$ and $j'_{k,m} \rightarrow 0$;
Invoke `ApproS` to obtain the weighted sum of loss and processing delay $LD'_{k,m}$ with $j'_{k,m}$;
for Each $DT_{k,m}$ in $\mathcal{DT}_n$
  while $j'_{k,m} < j_{k,m}$
    Invoke `ApproS` to obtain the weighted sum of loss and processing delay $LD_{k,m}$ with $j_{k,m}$;
    if $LD_{k,m} \leq LD'_{k,m}$
      $j'_{k,m} = j_{k,m}$, $LD'_{k,m} = LD_{k,m}$, and $j_{k,m} = j_{k,m} + 1$
    else if $LD_{k,m} > LD'_{k,m}$
      $j'_{k,m} = j_{k,m}$, $LD'_{k,m} = LD_{k,m}$, and $j_{k,m} = j_{k,m} - 1$
    end if
  end while
end for

each digital twin selects machine learning model $j_{k,m}$. It then selects the middle machine learning model, i.e., $j = \lfloor \frac{|\mathcal{J}_{k,m}|}{2} \rfloor$ for each digital twin $DT_{k,m}$ by invoking algorithm `ApproS`, provided that there is a digital twin with the weighted sum of processing delay and loss is lower than that of the randomly selected model. This is implemented by a number of iterations. Specifically, let $LD_{k,m}$ and $LD'_{k,m}$ be the obtained weighted sum of processing delay and loss of the current iteration and its previous iteration with selected models $j_{k,m}$ and $j'_{k,m}$, respectively. In the first iteration, $LD'_{k,m}$ is set to that when $j = \lfloor \frac{|\mathcal{J}_{k,m}|}{2} \rfloor$ for each digital twin $DT_{k,m}$; otherwise, it is set to that of the previous iteration. If $LD_{k,m} \leq LD'_{k,m}$ and $j_{k,m} \leq j'_{k,m}$, we let $j_{k,m} = j_{k,m} + 1$. If $LD_{k,m} \leq LD'_{k,m}$ and $j_{k,m} > j'_{k,m}$, we let $j_{k,m} = j_{k,m} - 1$. The above procedure continues until there is no models with $LD_{k,m} \leq LD'_{k,m}$. The proposed heuristic algorithm is referred to as `HeuS` for simplicity, which is shown in **Algorithm** 2.

D. Algorithm Analysis

Lemma 1: Given the obtained solution to a special case of the digital twin placement and model selection for streaming analytics with a single query, a given model and without the state updating delay requirement, there exists at least a value for equal intervals Q to make sure that the updating delay requirement of a digital twin $DT_{k,m} \in \mathcal{DT}_k^{vio}$ with its processing delays lie in $q_{k,m}$ can be met by cloudlets at a higher interval with $q'_{k,m} \geq q_{k,m}$.

Please see **Appendix B**, available in the supplemental file.

Theorem 2: The approximation ratio of the proposed algorithm `ApproS` for the digital twin placement and model selection for streaming analytics with a single query and selected model is $4Q|\mathcal{CL}|$, where Q is the number of equal intervals.

Please see **Appendix C**, available in the supplemental file.

We then analyze the time complexity of `ApproS` and `HeuS`.

Theorem 3: The time complexity of the proposed algorithm `ApproS` is $\mathcal{O}(|\mathcal{CL}||\mathcal{DT}_n| + n^\omega L + \sqrt{|\mathcal{CL}||\mathcal{DT}_n|}E + |DT_k^{vio}|(log(|DT_k^{vio}|) + Q|\mathcal{CL}|))$, and `HeuS` is $\mathcal{O}(|\mathcal{DT}_n|log|\mathcal{J}_{k,m}|(|\mathcal{CL}||\mathcal{DT}_n| + n^\omega L + \sqrt{|\mathcal{CL}||\mathcal{DT}_n|}E + |DT_k^{vio}|(log(|DT_k^{vio}|) + Q|\mathcal{CL}|)))$.

Please see **Appendix D**, available in the supplemental file.

V. Online Learning Algorithm for the Online Digital Twin Placement and Model Selection for Streaming Analytics with Multiple Queries

We now devise an online learning algorithm for the online digital twin placement and model selection for streaming analytics with multiple queries.

A. Overview

Given a finite time horizon with T equal time slots, queries arrive into the DT-MEC network dynamically without the knowledge of future query arrivals. If the DT-MEC network has sufficient computing resources, we can invoke the model selection and digital twin placement for a single query algorithm, i.e., `HeuS`, to respond to each query sequentially. However, when queries are highly concurrent in the DT-MEC network, computing resources may become insufficient. We thus design an algorithm to dynamically admit as many queries as possible such that the weighted sum of loss and processing delay of admitted queries is minimized.

The losses and processing delays of different queries with different admission decisions are determined by several factors, including the current resource availability of the network, the resource requirement of the model size of digital twins, the preferences of currently-arrived queries, the locations of home cloudlets, etc. First, the availability of network resources is most relevant to admission decisions. For instance, a cloudlet can place digital twins with a large model size with lower loss if there is abundant available resource; otherwise, the cloudlet tends to deploy digital twins with small-size models to improve overall resource utilization and processing delays. Second, the admission of a query may depend on the size of its selected machine learning model. For instance, when the model size of a query is larger than others, this query may need to be admitted later than others. The reason is that it may occupy a large amount of resource of the network, leading to the starvation of other queries with small-size models. Third, the weights on the loss and processing delays of currently arrived queries may also impact the admissions of queries. For instance, if a currently arrived query has low accuracy and high processing delay demand, the processing delay needs to be given a larger weight in the objective function (i.e., Eq. (15)). It causes the algorithm to admit queries with smaller models, such that the digital twins of admitted queries can be placed into cloudlets close to their home cloudlets. Fourth, the home cloudlet location of a query may also affect its admission. When computing resource of the cloudlets close to the home cloudlet of a query run out, the query needs to be admitted earlier than others. This is because the digital twins with a higher probability are placed into locations closer to their home cloudlet. Otherwise, the processing delay may be increased,

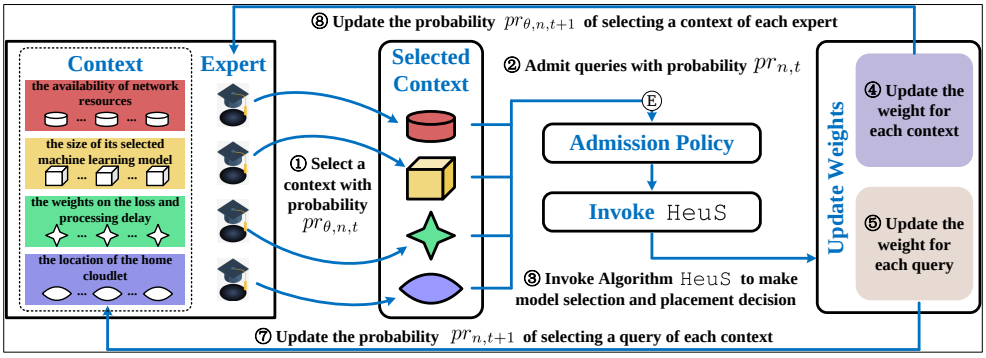


Fig. 2: An illustration of the steps of the proposed online learning algorithm OLM for admission policy.

or the state updating delay may be violated. Therefore, the proposed admission policy needs to consider the above factors and dynamically select which queries should be admitted.

To this end, we use an online learning method to learn these factors that impact the number of admitted queries and the weighted sum of their admissions, and dynamically make admission decisions for queries. We consider different factors as *contexts*, and devise a novel *contextual multi-armed bandit* algorithm to dynamically explore and exploit the impacts of different contexts on the admission decision of each arrived query. Specifically, we consider the compositions of the current available resource, the resource preferences of currently-arrived queries, and the locations of home cloudlets of arrived queries as contexts. Let Θ be the set of contexts, where each $\theta \in \Theta$ corresponds to a combination of the aforementioned factors.

B. Online Learning Algorithm

In the following, we develop the proposed online learning algorithm for the online digital twin placement and model selection for streaming analytics with multiple queries, by formulating it as a contextual multi-armed bandit problem.

We assume that the admission decision of queries as an 'arm'. Further, we consider that there is an expert of each context θ in Θ, which observes the values of each context, such as the available resource of the network, the data rates of physical objects, the processing delay of models, and the preferences of arrived queries. The expert also learns the correlations among contexts dynamically, and makes recommendations of the admission or rejection of a query, based on its observed values of different contexts. Note that the probability of recommending admitting a query is updated dynamically as the contexts change. As long as a query is recommended for admission by the contexts, algorithm HeuS is then invoked to select models and place their digital twins to cloudlets in the MEC network. If a query $q_n \in \mathcal{R}_t$ is not admitted at the current time slot t, it will be added to the set of queries $\mathcal{R}_{t+1}$ in the next time slot. However, this may cause some queries to be postponed for processing frequently. Although the response delay of queries is not considered, the proposed algorithm can be easily extended to avoid some queries being postponed. Specifically, the proposed online learning algorithm, referred to OLM, can consider the average number of rejected times as an additional context. If the average number of rejected queries in $\mathcal{R}_t$ increases, their probabilities of being considered for admissions will increase too.

Specific steps of algorithm OLM are in the following, as shown in Fig. 2. Initially, the algorithm plays each arm with equal probability. Let θ be an available context in the DT-MEC network, and $pr_{\theta,n}$ is the admission probability of the query q_n under the context θ. Notice that the probability $pr_{\theta,n}$ dynamically adjusts as the context updates. For each context θ, we maintain a weight on query q_n at each time slot t, which is denoted by $\omega_{\theta,n,t}$. The probability $pr_{\theta,n}$ is calculated by

$$pr_{\theta,n,t} = \omega_{\theta,n,t} / (\sum_{n'=1}^{|\mathcal{R}_t|} \omega_{\theta,n,t}). \tag{16}$$

Denote by pr_n the probability that the algorithm admits query q_n, which can be calculated by

$$pr_{n,t} = (\sum_{\theta \in \Theta} pr_{\theta,n}) / |\Theta|, \tag{17}$$

where $|\Theta|$ is the number of contexts in the current system. We then decide whether query q_n should be admitted according to probability pr_n. If q_n is admitted, algorithm HeuS is invoked to select models and place digital twins; otherwise, query q_n is rejected.

After queries are admitted at different time slots, different admission decisions may cause different rewards. Based on the reward value, the relationship between admission decisions of queries and network contexts is revealed. To minimize the weighted sum of loss and processing delay and admit as many queries as possible, we define the penalty of admitting query q_n with network context θ by Eq.(15).

Specifically, the weight $\omega_{\theta,n,t+1}$ of each query q_n with context θ is updated in the end of each time slot, where the penalty is assigned to each context with lower rewards, and the weight of context θ on request q_n at the next time slot $t+1$ is updated by

$$\omega_{\theta,n,t+1} = \omega_{\theta,n,t} \cdot (1-\epsilon)^{\phi_{\theta,n,t}}, \tag{18}$$

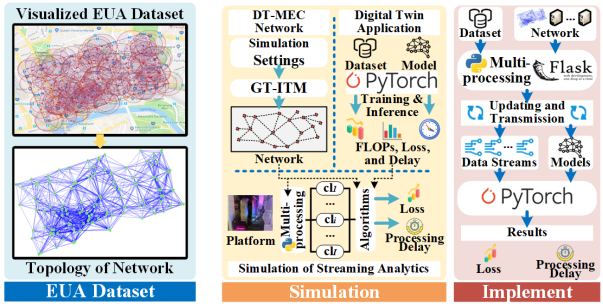
until every query $q_n \in \mathcal{R}_t$ is considered. The detailed steps of the proposed algorithm OLM are shown in **Algorithm** 3.

C. Algorithm Analysis

The rest is to analyze the expected cumulative regret and time complexity of the proposed online learning algorithm OLM.

Algorithm 3 A Contextual Online Learning Algorithm for the Digital Twin Placement and Model Selection for Streaming Analytics with Multiple Queries (OLM).

Require: A DT-MEC Network G, a time slot t, a set $\mathcal{PO}$ of physical objects with each object po_k has a set $\mathcal{S}_k$ of subsystem, a set $\mathcal{J}$ of models, a set of queries $\mathcal{R}_t$ with requesting the digital twin data analytic from a set $DT_{k,m} \in \mathcal{DT}_n$, and a set of home cloudlet $\mathcal{H}(cl_i)$.
Ensure: An admit policy of multiple queries $\mathcal{R}_t$ at time slot t.
for each t in T
  Define a set of admitted queries $\mathcal{Q} \leftarrow \emptyset$.
  for each expert in exp
    /* Update the probabilities for context and expert selection.*/
    Each context updates its probability of selecting a query by Eq. (16).
    Each expert selects a context by probability $pr_{\theta,n}$.
    Each query is admitted with probability pr_n.
    if q_n is admitted
      Set $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup q_n$ and $\mathcal{R}_t \leftarrow \mathcal{R}_t \setminus q_n$.
    end if
  end for
  /* Invoke proposed algorithm HeuS with the admitted queries.*/
  Obtain the digital twin placement and model selection decisions by invoking algorithm HeuS.
  Update the weights of every query by Eq. (18).
  Set $\mathcal{R}_{t+1} \leftarrow \mathcal{R}_{t+1} \bigcup \mathcal{R}_t$.
end for



(a) Dataset. (b) The implement of streaming analytics.

Fig. 3: The simulation settings.

TABLE II: Running results of ResNet on NEU-CLS-64 dataset

Model Name	FLOPs	Inference Loss(%)	Inference Delay(ms)
ResNet 18	1.81e+09	0.354	1.0-1.5
ResNet 36	3.66e+09	0.212	1.7-2.2
ResNet 50	3.86e+09	0.152	2.6-2.9
ResNet 101	7.57e+09	0.109	3.2-3.7
ResNet 152	1.13e+10	0.066	3.7-4.0

Definition 1: Since the purposed online learning algorithm OLM is to admit as many queries as possible into the MEC network with the aim to minimize the weighted sum of loss and processing delay of admitted queries over a given time horizon, the expected cumulative regret $Reg(T)$ of OLM is defined as

$$Reg(T) = \mathbb{E}(\Phi(OLM)) - \mathbb{E}(\Phi^*), \tag{19}$$

where $\Phi(OLM)$ is the reward obtained by algorithm OLM, and Φ^* is the reward obtained by the penalty admission policy.

Theorem 4: Given a DT-MEC network G and a set $\mathcal{R}_t$ of queries at time slot t, there is an online learning algorithm OLM for the online digital twin placement and model selection for streaming analytics with multiple queries. The expected cumulative regret in the solution delivered by algorithm OLM is upper bounded by $Reg(T) < \frac{ln|\mathcal{R}|}{\epsilon} + 2\epsilon(\delta^{max} + \gamma|\mathcal{R}|)$, where ϵ is a constant in the range of (0,1/2).

Please see **Appendix E**, available in the supplemental file.

Theorem 5: The time complexity of the proposed online learning algorithm OLM for the digital twin placement for streaming analytics with multiple queries is $\mathcal{O}(T(|\Theta| + |\mathcal{R}|))$.

Please see **Appendix F**, available in the supplemental file.

VI. Simulations

In this section, we evaluated the performance of the proposed algorithms by extensive simulations based on real datasets.

A. Simulation Settings

Consider a DT-MEC network that consists of from 10 to 50 cloudlets, where each network topology is generated via a network topological tool, GT-ITM [33]. Besides, we also considered a real-world network topology based on EUA dataset [34] in Fig. 3(a), which contains the geographical locations and the coverage radius of 124 APs and cloudlets in Melbourne, Australia. The coverage radius of an cloudlet is set between 450 and 750 meters, where a network link is established between two cloudlets if the coverage radius of the two APs can cover each other. There is a co-located cloudlet with each AP, and the computing capacity of each cloudlet is set from 6,000 MHz to 14,000 MHz randomly [35]. There are 100 physical objects in each network topology, where the number of subsystems $\mathcal{S}_k$ of each physical object is drawn from 1 to 20 [36]. The transmission delay for transmitting a unit of data (one MB) along a link between a pair of APs is drawn from 0.2 to 1 ms [37]. The data size of each update of a physical object is drawn from 2 to 5 MB [38] randomly. The query result size per query is set from 0.1 to 0.5 MB [36] randomly. The update delay requirement of each digital twin is set from 1 to 5 ms randomly.

We implemented a digital twin application for steel production using NEU-CLS-64 dataset [39]. We considered the steel production line as a physical object. The digital twins of production lines have been constructed using 3D modeling tools and machine learning models, i.e., ResNet [40]. Digital twins utilize steel data collected from production lines to perform defect detection. At each time slot, the subsystems of production lines obtain images of the steel using cameras, then synchronously update data streams to digital twins. The data consists of 1,800 grayscale images with 200×200 pixels, and there are six typical surface defects in this dataset: rolled-in scale, patches, crazing, pitted surface, inclusions, and scratches. The queries analyze data streams by machine learning models of digital twins to identify defects in steel produced on current production lines. The floating point operations per second (FLOPs), loss, and inference delay of these models are given in Tab. II. We treat the inference loss and the inference delay as part of the loss and processing delay, i.e., the loss and inference delay of unit data, respectively, and FLOPs as computing resource requirements.

We performed our simulation on a server with a 2.60 GHz Intel i9-13900H CPU, 32 GB RAM, and Nvidia RTX 4070 GPU. We implemented our algorithms and their benchmarks based on Python with some software packages, e.g., Pulp [41],

NetworkX [42], and Pytorch [43]. We simulated the streaming analytics of digital twin applications and implemented a real streaming analytics on the server, which can be shown in Fig. 3(b). We first constructed a DT-MEC network and a digital twin application. We then implement real streaming analytics in our simulation. We select a set of data randomly in the dataset as a data stream according to a given data rate to simulate its transmission among cloudlets in the DT-MEC network via Flask. Once the data stream is transmitted to the cloudlet, we invoke a container to perform streaming analytics. We use the machine learning model with PyTorch in the container to process each data stream and send the results to the home cloudlet of the query. The obtained results are based on the average of 10 runs of the proposed algorithms and their benchmarks.

The following five state-of-the-art algorithms are considered as benchmarks. Note that there does not exist an algorithm that jointly considers model selection and digital twin placement, we carefully extend the benchmarks for the sake of fairness.

- `Clipper+` [26]: the algorithm only selects the appropriate model based on Exp3 [44]. We extend `Clipper` by introducing an auxiliary decision variable $z_{k,m,i,j} = x_{k,m,i} \cdot y_{k,m,j}$ that determines both model selection and placement location simultaneously for the digital twin.
- `EdgeAdaptor` [11]: The authors use an auxiliary variable $z_{k,m,i,j}$ for model selection and placement based on a randomized rounding algorithm.
- `Gr-NoUpd` [27], [30]: The model selection for digital twins employs a greedy algorithm in [27], and the placement of digital twins by adopting an approximation algorithm in [30] that ignores the updating delay requirement.
- `HeuS-ILP`: This algorithm uses an ILP solver for the problem of digital twin placement while the model selection problem is solved using steps 1-4 of `HeuS`.
- `Optimal`: This algorithm jointly determines the digital twin placement strategy $x_{k,m,i}$ and model selection $y_{k,m,j}$ for the problem by solver Gurobi [45].
- `HeuS-LP` [41]: This algorithm relaxes the decision variables of digital placement in `HeuS-ILP` and uses an LP solver to get the placement locations of digital twins, which is the lower bound of the problem of digital twin placement. Note that the reason we adopted this algorithm is to find an estimation of the optimal digital twin placement decisions. The reason we use an LP is that an ILP takes a prohibitively long time to obtain the optimal solution to digital twin placement. Note that this is a conservative estimation, since the value of the solution due to LP is smaller than that of ILP.
- `HeuS-Upd` [46]: In [46], the digital twins are placed with minimum updating delay by a greedy algorithm, and models are selected by invoking steps 1-4 of `HeuS`.

We evaluated the proposed online learning algorithm `OLM` for the digital twin placement and model selection for streaming analytics with multiple queries problems, against the following three benchmarks.

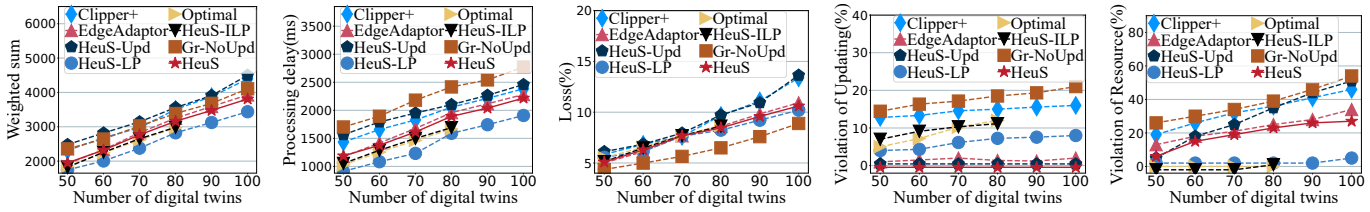
- `Random-HeuS`: The queries are admitted randomly, and the digital twin placement and model selection are implemented by invoking algorithm `HeuS`.
- `FCFS-HeuS`: The queries will be executed by their arrival order. The digital twin placement and model selection are implemented by invoking algorithm `HeuS`.
- `Admission` [13]: The admission policies are performed using the deep reinforcement learning algorithm.

B. Evaluation of the proposed approximation algorithm

We first studied the performance of the proposed approximation algorithm `HeuS` for the digital twin placement and model selection problem with a single query against other benchmarks in Figs. 4-9 in terms of the weighted sum of delay and loss, the violation of updating requirement, and the violation of computing resource.

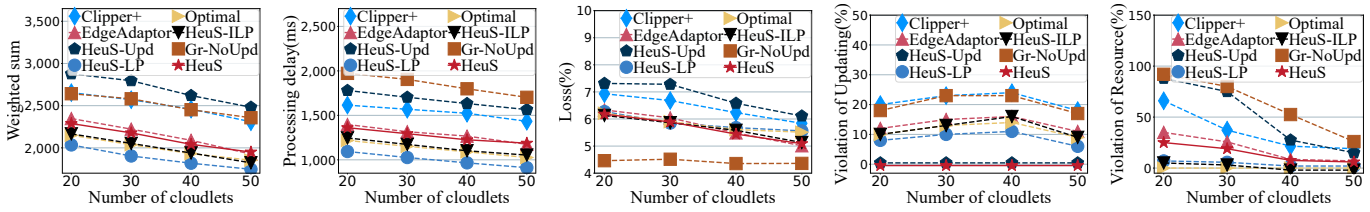
Fig. 4 illustrates the performance of different algorithms as the number of queried digital twins increases from 50 to 100. From Fig. 4(a), we can see that `HeuS` achieves a lower weighted sum of processing delay and loss compared to those of `Clipper+`, `HeuS-Upd`, `Gr-NoUpd`, `EdgeAdaptor` when the number of digital twins is 100. `HeuS` also achieves 81.7% weighted sum of processing delay and loss of `HeuS-LP`. The reason is that `HeuS` can select more accurate models and find better locations for digital twins. Note that the reason we do not provide the performance of the algorithms `HeuS-ILP` and `Optimal` when querying 90 and 100 digital twins is because of the unacceptable running times. In Fig. 4(b), we can observe that the processing delay increases significantly as the number of digital twins grows. The reason is that more digital twins may be placed into more cloudlets, causing higher processing delays due to aggregations among digital twins. Further, we observed that `Gr-NoUpd` suffers the highest processing delay, as the greedy algorithm selects models with the lowest loss and the highest inference delay. Fig. 4(c) demonstrates that `HeuS` achieves a loss loss close to that of `Gr-NoUpd`. The reason is that `HeuS` leverages the DT-MEC network resources through binary search and selects models with high accuracy for streaming analytics. Also, Figs. 4(d) and 4(e) show the violations on updating requirement and computing resource. We observed that as the number of digital twins increases, the violation rates of both the updating delay requirement and computing resource increase significantly. Furthermore, it can be seen that `HeuS` does not incur any updating requirement violation, because it has an updating delay-aware replacement mechanism. Due to this mechanism, `HeuS` causes a slightly higher violation rate of resource, compared to `HeuS-LP`, `EdgeAdaptor`, `HeuS-ILP`, and `Optimal` as the number of digital twins increases.

We then investigated the impact of the number of cloudlets on the performance of algorithms in Fig. 5, by varying the number of cloudlets from 20 to 50. From Fig. 5(a), we can see that the weighted sum of processing delay and loss by different algorithms shows downward trends with the increase on the number of cloudlets. `HeuS` achieves 15.8%, 25.6%, 14.2% less weighted sum of processing delay and loss than those of `Clipper+`, `HeuS-Upd`, `Gr-NoUpd`. `HeuS` achieves 88.9%, 93.7%, and 94.4% of the weighted sum of delay and loss of



(a) Weighted sum of delay and loss (b) Processing delay (c) Loss of selected model (d) Violation rate of updating delay requirement (e) Violation rate of computing resource

Fig. 4: The impact of the number of queried digital twins on the performance of algorithm HeuS and its benchmarks



(a) Weighted sum of delay and loss (b) Processing delay (c) Loss of selected model (d) Violation rate of updating delay requirement (e) Violation rate of computing resource

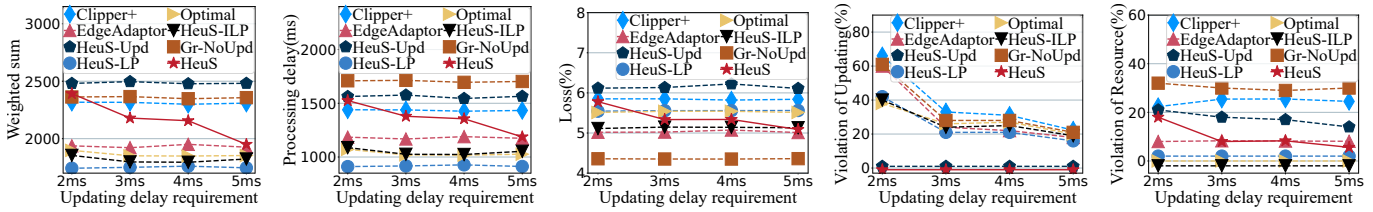
Fig. 5: The impact of the number of cloudlets on the performance of algorithm HeuS and its benchmarks

HeuS-LP, HeuS-ILP, and Optimal respectively, when the number of cloudlets is set at 50. In Figs. 5(b) and 5(c), we can observe that the processing delay and loss decrease as the number of cloudlets increases. The reason is that the increase in the number of cloudlets enables more available computing resources, and the models with lower loss can be selected. In Fig. 5(d), we can see that the violation rate of updating delay requirement for Clipper+, Gr-NoUpd, EdgeAdaptor, HeuS-LP, HeuS-ILP, and Optimal increases when the number of cloudlets is between 20 and 30, and decreases when it is between 40 and 50. The reason is that when the number of servers is 20, the network connections between cloudlets are more dense, resulting in lower updating delay. Meanwhile, as the number of cloudlets increases, the digital twins can be placed in locations that are close to physical objects with lower updating delays. We also observer that HeuS and HeuS-Upd meet the updating delay requirement. Accordingly, as the number of cloudlets increases, the resource violation rate demonstrates downward trends in Fig. 5(e), due to the increase in the amount of available computing resources.

We also investigated the impact of the updating delay requirement on the performance of different algorithms by varying the value of the updating delay requirement from 2ms to 5ms in Fig. 6. Fig. 6(a) presents the effect of updating the delay requirement on the weighted sum of processing delay and loss. We can see that algorithms Clipper+, HeuS-Upd, Gr-NoUpd, HeuS-LP, HeuS-ILP, and EdgeAdaptor show steady trends in terms of the weighted sum, which means that the updating delay requirement only has a little effect on it. However, HeuS shows a significant downward trend. This is due to the HeuS uses a re-placement mechanism, which is designed to stratify updating delay constraints by placing digital twins adjacent to physical objects, resulting in higher processing delays. The impact of updating delay requirements on processing delay and loss is shown in Figs. 6(b) and 6(c), respectively, from which we can see that HeuS shows a trend in the figures similar to that in Fig. 6(a). Further, the decrease in HeuS has a slow stall between 3 ms and 4 ms. The rationale behind this is that the updating delay requirement for 3ms and 4ms results in an identical resource violation rate. Thus, HeuS did not need to replace more digital twin models. Fig. 6(d) presents the violation rates of updating the delay requirement. As the updating delay requirement increases, the violation rate decreases with all algorithms, and a stall is observed between 3 ms and 4 ms. Also, Fig. 6(e) describes the resource violation rates of seven algorithms. Due to the re-placement mechanism, the resource violation rate of HeuS decreases, with the highest resource violation rate observed when the updating delay constraint is 2 ms and comparable to HeuS-LP when the updating delay constraint is 5 ms.

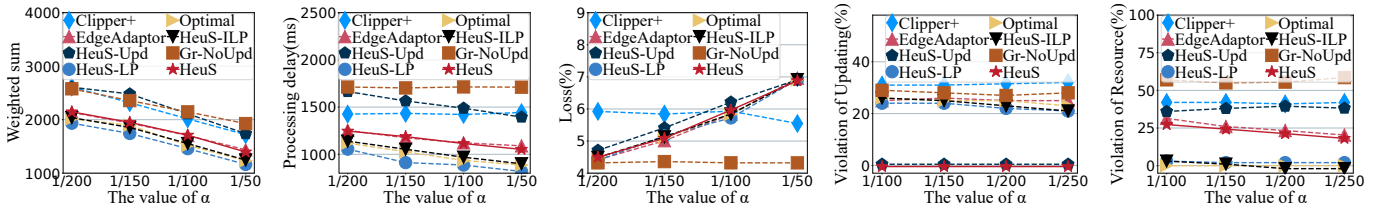
We further evaluated the performance of algorithms under the impact of the value of weight, i.e., α, by decreasing α from 1/200 to 1/50, in Fig. 7. Fig. 7(a) represents that the weighted sum of processing delay and loss for all other algorithms shows a decreasing trend. Besides, HeuS has 21.7% and 20.9% superior to Clipper+ and HeuS-Upd, respectively, when α is 1/200. Further, we can see that the processing delay has a downward trend and the loss has an upward trend of algorithms with the decrease in value of α except for Gr-NoUpd in Figs. 7(b) and 7(c). The reason behind this is that with the growth of α in Eq. (9), the algorithms tend to select a small model for streaming analytics with lower processing delay and higher loss. Besides, the reason why Gr-NoUpd does not decrease is that the greedy algorithm does not proactively adjust its model selection strategy based on the size of α. In Figs. 7(d) and 7(e), all methods have a stable trend in terms of the violation of updating delay and show a downward trend in terms of the violation of computing resource.

In the following, we investigated the impact of the number of queried digital twins on the performance when there are 20, 30, and 40 digital twins with EUA dataset in Fig. 8. In Fig. 8(a), with the increase of the number of queried digital



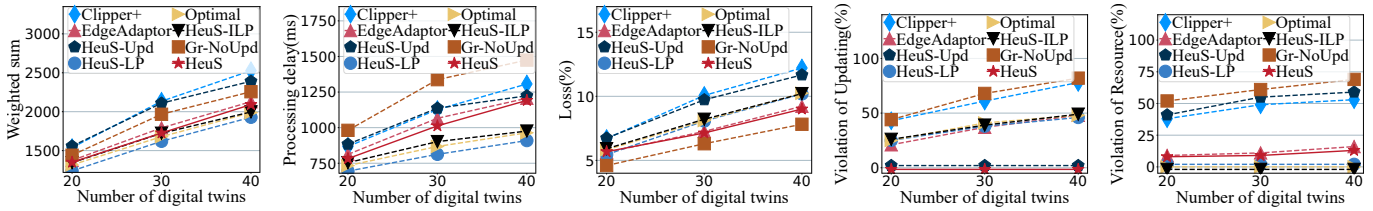
(a) Weighted sum of delay and loss (b) Processing delay (c) Loss of selected model (d) Violation rate of updating delay requirement (e) Violation rate of computing resource

Fig. 6: The impact of the updating delay constraint on the performance of algorithm HeuS and its benchmarks



(a) Weighted sum of delay and loss (b) Processing delay (c) Loss of selected model (d) Violation rate of updating delay requirement (e) Violation rate of computing resource

Fig. 7: The impact of the value of α on the performance of algorithm HeuS and its benchmarks

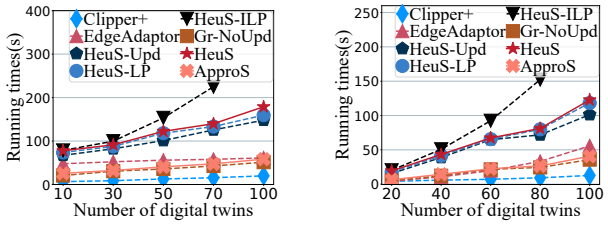


(a) Weighted sum of delay and loss (b) Processing delay (c) Loss of selected model (d) Violation rate of updating delay requirement (e) Violation rate of computing resource

Fig. 8: The impact of the number of queried digital twins on the performance of algorithms HeuS and its benchmarks with EUA dataset

twins, the weighted sum of processing delay and loss of the seven algorithms increases. HeuS achieves 92.3% of the weighted sum of processing delay and loss of HeuS-LP, and outperforms those of Clipper+, HeuS-Upd, Gr-NoUpd, and EdgeAdaptor by 13.4%, 12.5%, 7.3%, and 1.7%, respectively, when the number of digital twin is 30. The tendency of experimental results of HeuS on EUA dataset is similar to those in Fig. 4, where the network topology is generated by GT-ITM. The reasons are similar to the arguments for Fig. 4, we do not repeat here for the sake of space. This shows that HeuS also has superiority in real-world datasets.

We finally evaluated the running times of different algorithms with different numbers of digital twins and cloudlets in Fig. 9. We can observe that as the number of cloudlets or queried digital twins increases, the running times of the comparison algorithms grow significantly. We can see that HeuS-ILP has the highest runtime. Besides, when the number of cloudlets or queried digital twins is 100, the result of HeuS-ILP cannot be obtained within an acceptable running time. Moreover, due to binary search, the runtime of HeuS, HeuS-Upd, and HeuS-LP is significantly greater than that of the other comparison algorithms. HeuS-LP and HeuS-Upd have lower runtimes compared to HeuS due to that the latter has an updating delay requirement-aware re-placement mechanism.



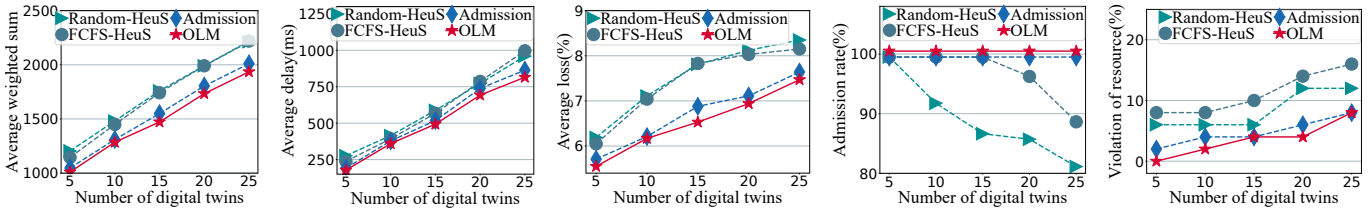
(a) Running time of different numbers of cloudlets (b) Running time of different numbers of digital twins

Fig. 9: The running times of various algorithms

Similarly, the running time of ApproS is higher than that of Gr-NoUpd for the same reason. It can be observed that EdgeAdaptor and Clipper+ are stable, as their computing complexity is less correlated with the number of cloudlets.

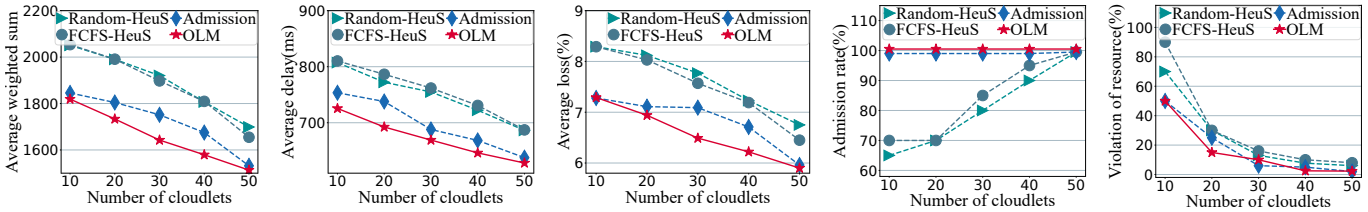
C. Evaluation of the proposed online algorithm

In this subsection, we investigated the performance of the proposal online algorithm OLM for the digital twin placement and model selection problem with multiple queries against its benchmarks in Figs. 10-15 in terms of the weighted sum of delay and loss, the admission rate of queries, and the violation of computing resource.



(a) Weighted sum of delay and loss (b) Average processing delay (c) Average loss of selected models (d) Admission rate of queries (e) Violation rate of computing resource

Fig. 10: The impact of the number of queried digital twins on the performance of algorithm OLM and its benchmarks



(a) Weighted sum of delay and loss (b) Average processing delay (c) Average loss of selected model (d) Admission rate of queries (e) Violation rate of computing resource

Fig. 11: The impact of the number of cloudlets on the performance of algorithm OLM and its benchmarks

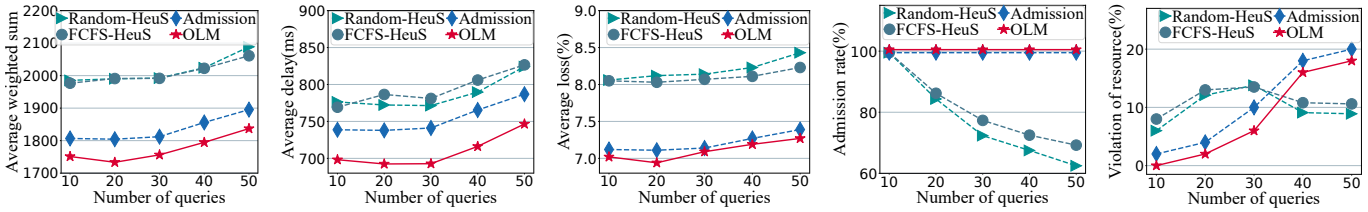
We first evaluated the performance of different algorithms, by varying the number of queried digital twins from 5 to 25 while fixing the number of cloudlets at 50 and queries at 10 in Fig. 10. In Fig. 10(a), the weighted sum of processing delay and loss tends to increase as the number of queries increases. OLM achieves the best performance, for example, it is about 13.4% lower than other benchmarks when the number of queried digital twins is 20. The reason is that OLM adopts the content-aware multi-armed bandit method to dynamically aware the network information, and generates a more reasonable admission policy with lower average processing delay and loss, which are shown in Figs. 10(b) and 10(c), respectively. In Fig. 10(d), OLM and Admission admit all queries and guarantee the admission rate. The rationale behind this is that OLM considers the weighted sum and the admission rate of queries as an optimization objective, which ensures that all queries can be responded to and admitted within a reasonable time. In Fig. 10(e), we observe that OLM achieves the lowest computing resource violation rate.

We also investigated the impact of the number of cloudlets on the performance of the algorithms while fixing the number of queried digital twins and queries at 10, respectively, in Fig. 11. From Figs. 11(a), 11(b), and 11(c), it can be seen that as the number of cloudlets increases, processing delay and loss decrease significantly, which also causes the weighted sum of processing delay and loss to show a downward trend. As shown in Fig. 11(a), the performance of OLM is 10.8% and 8.5% better than those of Random-HeuS and FCFS-HeuS, respectively, when the number of cloudlets is 50. Fig. 11(d) represents the admission rate of queries of the four algorithms, where all queries are admitted by OLM. Fig. 11(e) shows that the resource violation rate decreases as the number of cloudlets increases. The rationale behind this is that as available computing resources increase, the number of admitted queries increases, and the violation of computing resources decreases.

We then explored the impact of the number of queries on the performance of the algorithms while fixing the number of cloudlets at 50 and queried digital twins at 10 in Fig. 12. Fig. 12(a) illustrates that OLM achieves the lowest weighted sum of processing delay and loss. Fig. 12(b) presents a growing trend in terms of processing delay in the long-range view. Fig. 12(c) shows that the loss firstly decreases and then increases with the growth of the number of queries. The reason has two folds: 1) As the number of queries grows, the number of queried digital twins increases, leading to a growth in the average processing delay. 2) Since the weights of queries are different, the algorithm OLM admits the queries with lower loss requirements first, which decreases the average loss but increases the average processing delay. This shows that OLM can better respond to different requirements of queries. From Fig. 12(d), we can see that the admission rates of algorithms Random-HeuS and FCFS-HeuS have decreased. This leads to fewer admitted queries and a lower resource violation rate, which can be verified in Fig. 12(e).

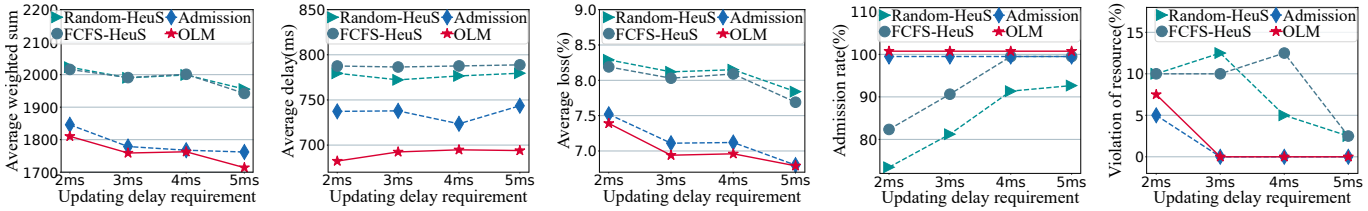
We thirdly evaluated the impact of the updating delay constraint of physical objects by varying the delay from 2ms to 5ms on the performance of different algorithms by setting the number of cloudlets at 20 and queries at 10, as shown in Fig. 13. From Fig. 13(a), we can see that the weighted sum demonstrates a downward trend with the relaxation of updating delay, and OLM achieves the lowest value of the weighted sum, for example, OLM is about 10.9% superior to Random-HeuS and FCFS-HeuS, as well as 2.7% better than Admission when the updating delay requirement is 4 ms. In Figs. 13(b) and 13(c), we can see that the loss of algorithms shows the same trend as HeuS in Fig. 6(c). The reason is that the algorithms are all implemented based on HeuS, thus presenting the same trend when the updating delay requirement changes. In Figs. 13(d) and 13(e), we see that the updating delay requirement grows as the admission rate of queries increases and the violation of computing resources decreases.

We fourthly evaluated the impact of the number of queried



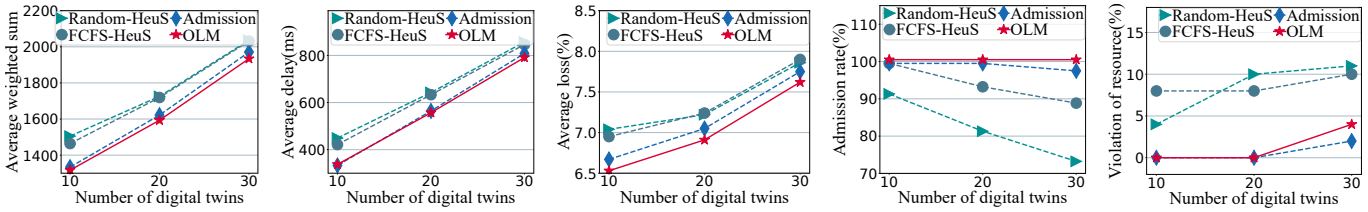
(a) Weighted sum of delay and loss (b) Average processing delay (c) Average loss of selected model (d) Admission rate of queries (e) Violation rate of computing resource

Fig. 12: The impact of the number of queries on the performance of algorithm OLM and its benchmarks



(a) Weighted sum of delay and loss (b) Average processing delay (c) Average loss of selected model (d) Admission rate of queries (e) Violation rate of computing resource

Fig. 13: The impact of the updating delay constraint on the performance of algorithm OLM and its benchmarks

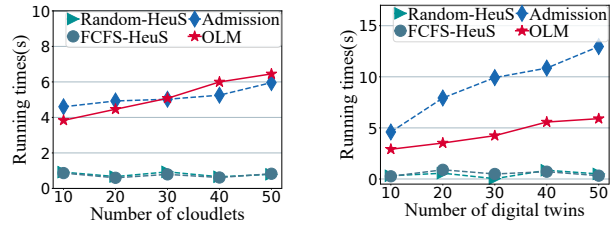


(a) Weighted sum of delay and loss (b) Average processing delay (c) Average loss of selected model (d) Admission rate of queries (e) Violation rate of computing resource

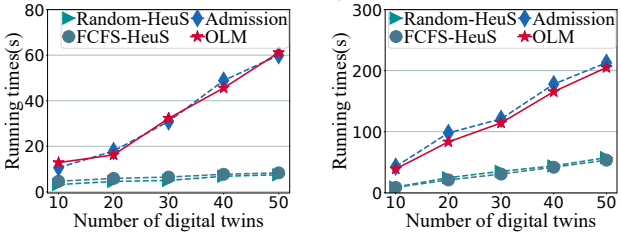
Fig. 14: The impact of the number of queried digital twins on the performance of algorithm OLM and its benchmarks with EUA dataset

digital twins on the performance of different algorithms with the real-world EUA dataset, assuming that there are 20 cloudlets and 10 queries in Fig. 14. Similar to experimental results of Fig. 10(a), the weighted sum of processing delay and loss increases with the growth of the number of queried digital twins. The same trend can be seen in Figs. 14(b) and 14(c). We can see that OLM is superior to Random-HeuS, FCFS-HeuS, and Admission by 7.2%, 5.6%, and 1.9% in terms of processing delay and loss, respectively. The rationales behind this are similar to the arguments for Fig. 10(a) with network topology generated by GT-ITM, we do not repeat here for the sake of space. From Figs. 14(d) and 14(e), we can see that the OLM and Admission algorithms have achieved the optimal admission rate of queries and resulted in a lower violation of computing resources. The reasons are similar to those in Figs. 10(d) and 10(e).

We finally evaluated the running time of algorithms with different numbers of digital twins and cloudlets in Fig. 15. From Fig. 15, we can see that the running time of OLM increases with the growth of the number of cloudlets, the queried digital twins, the number of queries, and the number of time slots. The reason is that the increase in these numbers has led to an increase in the complexity of the algorithms and decision-making time. Although the running time of OLM is higher than some other benchmarks, it however achieves much better performance in terms of the weighted sum, the admission rate, and the violation rate of computing resources.



(a) Running time with multiple cloudlets (b) Running time with multiple digital twins



(c) Running time with multiple queries (d) Running time with multiple time slots

Fig. 15: The running time of algorithms OLM and its benchmarks

D. Discussion on Implementation Direction for Streaming Analytics of Digital Twin Applications in a DT-MEC Network

The implementation of the proposed algorithms for streaming analytics relies on a platform that manages the network and responds to queries in a DT-MEC network. Considering the implementation of a real DT-MEC network requires the real deployment of physical objects in a real industrial scenario and the lack of an open experimental platform, the real implementation of a DT-MEC network is not the major focus of this paper. However, to enable efficient transmission and synchronization between physical objects and their digital twins of the DT-MEC network, the KubeEdge platform can be adopted, which utilizes an asynchronous communication model to support the transmission and synchronization of data streams. When physical objects can be attached to a DT-MEC network based on KubeEdge [47], KubeEdge manages the data streams and performs streaming analytics to respond to admitted queries.

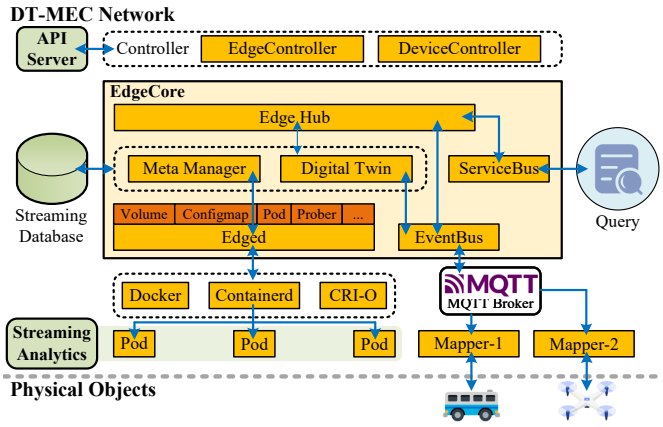


Fig. 16: An example of implementation for streaming analytics of digital twin applications in a DT-MEC network using the KubeEdge platform.

Fig. 16 gives an example of implementation for streaming analytics of digital twin applications in a DT-MEC network using the KubeEdge platform. Specifically, physical objects are connected to the DT-MEC network via the message queuing telemetry transport (MQTT) protocol, where the MQTT Broker is used to manage the data stream uploading. The data streams of physical objects are transmited to the EventBus module and the EdgeHub module, synchronizing data streams with the digital twin deployed in the cloudlet. Meanwhile, these data streams are stored in a streaming database via the MetaManager module. When a query is issued to the ServiceBus module, the information queried digital twin is sent to the EdgeHub module. Based on the deployment location of the digital twin in the EdgeController module, a pod is invoked, and a container is created to host an instance of the selected model by the Edged module. Finally, streaming analytics is executed within the container to obtain the analytical results.

VII. Conclusion and Future Work

In this paper, we studied a novel digital twin placement and model selection problem in DT-MEC networks. We first designed an approximation algorithm for the problem with a single query, which delivers an approximate solution to a special case of the problem with selected models. We then devised a heuristic algorithm for the problem to select appropriate models. We also proposed an online learning algorithm for dynamic query admissions by a contextual multi-armed bandit problem and showed the regret bound of the solution obtained by the online algorithm. We finally evaluated the performance of the proposed algorithms by simulations on real datasets. Experimental results demonstrate that our proposed method for the problem with a single query achieves a 23.6% and 36.4% reduction in processing delay and loss, respectively, compared to existing methods, without any violation of updating delays. For the problem with multiple query admissions, our proposed method can reduce the weighted sum of processing delay and loss by 13.3% compared to the other algorithms while maintaining the admission rate of queries.

The future work from this research is to investigate the interaction among multiple physical objects in an MEC network. The specific works include: (1) Implement streaming analytics of digital twin in a real-world test-bed. (2) Developing a distributed platform for digital twins to enable real-time state synchronization and data transmission between digital twins across different cloudlets. (3) Based on the proposed digital twin model selection and placement algorithm, propose an algorithm to implement low-delay yet accurate stream analytics for physical object interactions.

References

[1] B. Wang, H. Zhou, X. Li, G. Yang, P. Zheng, C. Song, Y. Yuan, T. Wuest, H. Yang, and L. Wang, "Human digital twin in the context of industry 5.0," *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102626, 2024.

[2] H. Xu, J. Wu, Q. Pan, X. Guan, and M. Guizani, "A survey on digital twin for industrial internet of things: Applications, technologies and tools," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2569–2598, 2023.

[3] Y. Qiu, M. Chen, W. Liang, L. Ai, and D. Niyato, "Information sharing in multi-tenant metaverse via intent-driven multicasting," *IEEE Transactions on Computers*, pp. 1–14, 2025.

[4] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, "How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 593–612, 2024.

[5] Y. Zhang, J. Hu, and G. Min, "Digital twin-driven intelligent task offloading for collaborative mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3034–3045, 2023.

[6] Y. Jiang, M. Li, W. Wu, X. Wu, X. Zhang, X. Huang, R. Y. Zhong, and G. G. Huang, "Multi-domain ubiquitous digital twin model for information management of complex infrastructure systems," *Advanced Engineering Informatics*, vol. 56, p. 101951, 2023.

[7] X. Huang, W. Wu, S. Hu, M. Li, C. Zhou, and X. Shen, "Digital twin based user-centric resource management for multicast short video streaming," *IEEE Journal of Selected Topics in Signal Processing*, vol. 18, no. 1, pp. 50–65, 2024.

[8] K. Willcox and B. Segundo, "The role of computational science in digital twins," *Nature Computational Science*, vol. 4, no. 3, pp. 147–149, 2024.

[9] Y. Li, S. Wang, Y. Li, A. Zhou, M. Xu, X. Ma, and Y. Liu, "Seamless cross-edge service migration for real-time rendering applications," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7084–7098, 2024.

[10] H. Huang, W. Zhan, G. Min, Z. Duan, and K. Peng, "Mobility-aware computation offloading with load balancing in smart city networks using mec federation," *IEEE Transactions on Mobile Computing*, vol. 23, no. 11, pp. 10 411–10 428, 2024.

[11] K. Zhao, Z. Zhou, X. Chen, R. Zhou, X. Zhang, S. Yu, and D. Wu, "Edgeadaptor: Online configuration adaption, model selection and resource provisioning for edge dnn inference serving at scale," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5870–5886, 2023.

[12] L. Zhou, Z. Xu, Q. Xia, Z. Xu, W. Ren, W. Qi, J. Ma, S. Yan, and Y. Yang, "Chasing common knowledge: Joint large model selection and pulling in mec with parameter sharing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 36, no. 3, pp. 437–454, 2025.

[13] T. Wang, L. Shen, Q. Fan, T. Xu, T. Liu, and H. Xiong, "Joint admission control and resource allocation of virtual network embedding via hierarchical deep reinforcement learning," *IEEE Transactions on Services Computing*, vol. 17, no. 3, pp. 1001–1015, 2024.

[14] H. Chen, T. D. Todd, D. Zhao, and G. Karakostas, "Digital twin model selection for feature accuracy," *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 11 415–11 426, 2024.

[15] Y. Zhou, Y. Fu, Z. Shi, K. Hung, T. Q. S. Quek, and Y. Zhang, "Sustainable placement with cost minimization in wireless digital twin networks," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2024.

[16] J. Gu, Y. Fu, and K. Hung, "On intelligent placement decision-making algorithms for wireless digital twin networks via bandit learning," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2024.

[17] Y. Zhang, H. Zhang, Y. Lu, W. Sun, L. Wei, Y. Zhang, and B. Wang, "Adaptive digital twin placement and transfer in wireless computing power network," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10 924–10 936, 2024.

[18] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya, "Digital twin-enabled service satisfaction enhancement in edge computing," in *IEEE INFOCOM'23*, 2023, pp. 1–10.

[19] K. Peng, H. Huang, M. Bilal, and X. Xu, "Distributed incentives for intelligent offloading and resource allocation in digital twin driven smart industry," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 3133–3143, 2023.

[20] X. Huang, H. Yang, S. Hu, and X. Shen, "Digital twin-driven network architecture for video streaming," *IEEE Network*, vol. 38, no. 6, pp. 334–341, 2024.

[21] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *IEEE VTC'20-Spring*. IEEE, 2020, pp. 1–6.

[22] X. Fu, T. Ghaffar, J. C. Davis, and D. Lee, "{EdgeWise}: A better stream processing engine for the edge," in *USENIX ATC'19*, 2019, pp. 929–946.

[23] P. Liu, D. Da Silva, and L. Hu, "{DART}: A scalable and adaptive edge stream processing engine," in *USENIX ATC'21)*, 2021, pp. 239–252.

[24] S. Ding, L. Yang, J. Cao, W. Cai, M. Tan, and Z. Wang, "Partitioning stateful data stream applications in dynamic edge cloud environments," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2368–2381, 2022.

[25] Y. Gao, X. Li, and G. Liang, "A deep lifelong learning method for digital twin-driven defect recognition with novel classes," *Journal of Computing and Information Science in Engineering*, vol. 21, no. 3, p. 031004, 2021.

[26] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A Low-Latency online prediction serving system," in *USENIX NSDI'17*, 2017, pp. 613–627.

[27] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "Infaas: Automated model-less inference serving," in *USENIX ATC'21*, 2021, pp. 397–411.

[28] V. Nigade, P. Bauszat, H. Bal, and L. Wang, "Jellyfish: Timely inference serving for dynamic edge networks," in *IEEE RTSS'22*, 2022, pp. 277–290.

[29] L. Zhao, C. Wang, K. Zhao, D. Tarchi, S. Wan, and N. Kumar, "Interlink: A digital twin-assisted storage strategy for satellite-terrestrial networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 5, pp. 3746–3759, 2022.

[30] X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia, "Multiple service model refreshments in digital twin-empowered edge computing," *IEEE Transactions on Services Computing*, pp. 1–15, 2023.

[31] Q. Xia, W. Ren, M. Li, and J. Ren, "Age-aware query evaluation for big data analytics in mobile edge clouds," in *IEEE HPCC'20*, 2020, pp. 214–222.

[32] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical programming*, vol. 62, no. 1, pp. 461–474, 1993.

[33] K. Calvert, J. Eagan, S. Merugu, A. Namjoshi, J. Stasko, and E. Zegura, "Extending and enhancing gt-itm," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, 2003, pp. 23–27.

[34] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[35] Z. Xu, D. Li, W. Liang, W. Xu, Q. Xia, P. Zhou, O. F. Rana, and H. Li, "Energy or accuracy? near-optimal user selection and aggregator placement for federated learning in mec," *IEEE Transactions on Mobile Computing*, vol. 23, no. 3, pp. 2470–2485, 2023.

[36] J. Li, S. Guo, W. Liang, J. Wu, Q. Chen, Z. Xu, W. Xu, and J. Wang, "Aoi-aware, digital twin-empowered iot query services in mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 32, no. 4, pp. 3636–3650, 2024.

[37] Z. Xu, L. Zhou, W. Liang, Q. Xia, W. Xu, W. Ren, H. Ren, and P. Zhou, "Stateful serverless application placement in mec with function and state dependencies," *IEEE Transactions on Computers*, vol. 72, no. 9, pp. 2701–716, 2023.

[38] Y. Zhang, W. Liang, Z. Xu, and X. Jia, "Mobility-aware service provisioning in edge computing via digital twin replica placements," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 295–11 311, 2024.

[39] Y. Bao, K. Song, J. Liu, Y. Wang, Y. Yan, H. Yu, and X. Li, "Triplet-graph reasoning network for few-shot metal generic surface defect segmentation," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[40] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, 2016. [Online]. Available: https://arxiv.org/abs/1603.08029

[41] S. Mitchell, M. OSullivan, and I. Dunning, "Pulp: a linear programming toolkit for python," *The University of Auckland, Auckland, New Zealand*, vol. 65, p. 25, 2011.

[42] "Networkx: Network analysis with python," 2024. [Online]. Available: https://networkx.github.io

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

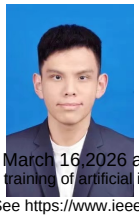
[44] T. Lattimore and C. Szepesvari, *Bandit algorithms*. Cambridge University Press, 2020.

[45] J. P. Pedroso, "Optimization with gurobi and python," *INESC Porto and Universidade do Porto, Portugal*, vol. 1, 2011.

[46] L. Chen, S. Zheng, Y. Wu, H.-N. Dai, and J. Wu, "Resource and fairness-aware digital twin service caching and request routing with edge collaboration," *IEEE Wireless Communications Letters*, vol. 12, no. 11, pp. 1881–1885, 2023.

[47] Y. Xiong, Y. Sun, L. Xing, and Y. Huang, "Extend cloud to edge with kubeedge," in *2018 IEEE/ACM Symposium On Edge Computing (SEC)*. IEEE, 2018, pp. 373–377.

Qiufen Xia received her PhD degree from the Australian National University in 2017, the ME degree and BSc degree from Dalian University of Technology in China in 2012 and 2009, all in Computer Science. She is currently an Associate Professor at the Dalian University of Technology. Her research interests include mobile cloud computing, query evaluation, big data analytics, big data management in distributed clouds, and cloud computing.



Peichen Liu is currently working towards the Ph.D. degree with the School of Software. His research interests include mobile edge computing, reinforcement learning, approximation algorithm, and AI infrastructure/systems.

Zichuan Xu (M'17) received his PhD degree from the Australian National University in 2016, ME and BSc degrees from Dalian University of Technology in China in 2011 and 2008, all in Computer Science. From 2016 to 2017, he was a Research Associate at Department of Electronic and Electrical Engineering, University College London, UK. He is currently a Professor in School of Software at Dalian University of Technology. He is also a "Xinghai Scholar' in Dalian University of Technology. His research interests include mobile edge computing, serverless computing, network function virtualization, Internet of Things, and algorithm design. He serves as an Associate Editor of the IEEE Transactions on Parallel and Distributed Systems.

Jiankang Ren (Member, IEEE) received the B.Sc., M.E., and the Ph.D. degrees in Computer Science from Dalian University of Technology in China, in 2008, 2011, and 2015, respectively. He was a Visiting Scholar with the Computer and Information Science Department, University of Pennsylvania, USA, from September 2013 to September 2014. He is currently an Associate Professor with the School of Computer Science and Technology at Dalian University of Technology. His research interests include cyber-physical systems (CPS), cloud computing, and computational intelligence.

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 Professor in the Department of Computer Science at City University of Hong Kong. Prior to that, he was a Professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for Internet of Things, Mobile Edge Computing (MEC), Network Function Virtualization (NFV), Software-Defined Networking (SDN), approximation algorithms, combinatorial optimization, and graph theory. He currently serves as an Associate Editor in the Editorial Board of IEEE TRANSACTIONS ON COMMUNICATIONS, and he is a senior member of IEEE.

Guangyuan Xu is currently working towards the Ph.D degree with the School of Software, Dalian University of Technology, Dalian, China. His current research interests include mobile edge computing, satellite edge computing, and resource allocation.

Wenzheng Xu received the BSc, ME and PhD degrees in computer science from Sun Yat-Sen University, Guangzhou, PR China, in 2008, 2010, and 2015, respectively. He currently is an Associate Professor at the Sichuan University and was a visitor at the Australian National University. His research interests include wireless ad hoc and sensor networks, mobile computing, approximation algorithms, combinatorial optimization, online social networks, and graph theory. He is a member of the IEEE.

Pan Zhou (S07M14-SM20) is currently a full professor and PhD advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He is currently an associate editor of IEEE Transactions on Network Science and Engineering. His current research interest includes: security and privacy, big data analytics, machine learning, and information networks.

Hao Li is currently an associate researcher in the Institute of Ningxia at China Coal Research Institute. Dr. Li's research interests include Coal mine intellectualization, automation of fully mechanized mining face, edge computing, AI and Internet of Things.

Zhen Feng received his PhD degree from Dalian University of Technology in China in 2014. He is currently a senior research engineer in Jinan Inspur Data Technology. His research interests include cloud computing, server virtualization and software-defined storage.