Distributed Digital Twin Migration in Multi-tier Computing Systems

Zhixiong Chen, *Graduate Student Member, IEEE*, Wenqiang Yi, *Member, IEEE*, Arumugam Nallanathan, *Fellow, IEEE*, and Jonathon Chambers, *Fellow, IEEE*

***Abstract*—At the network edges, the multi-tier computing framework provides mobile users with efficient cloud-like computing and signal processing capabilities. Deploying digital twins in the multi-tier computing system helps to realize ultra-reliable and low-latency interactions between users and their virtual objects. Considering users in the system may roam between edge servers with limited coverage and increase the data synchronization latency to their digital twins, it is crucial to address the digital twin migration problem to enable real-time synchronization between digital twins and users. To this end, we formulate a joint digital twin migration, communication and computation resource management problem to minimize the data synchronization latency, where the time-varying network states and user mobility are considered. By decoupling edge servers under a deterministic migration strategy, we first derive the optimal communication and computation resource management policies at each server using convex optimization methods. For the digital twin migration problem between different servers, we transform it as a decentralized partially observable Markov decision process (Dec-POMDP). To solve this problem, we propose a novel agent-contribution-enabled multi-agent reinforcement learning (AC-MARL) algorithm to enable distributed digital twin migration for users, in which the counterfactual baseline method is adopted to characterize the contribution of each agent and facilitate cooperation among agents. In addition, we utilize embedding matrices to code agents' actions and states to release the scalability issue under the high dimensional state in AC-MARL. Simulation results based on two real-world taxi mobility trace datasets show that the proposed digital twin migration scheme is able to reduce 23%-30% data synchronization latency for users compared to the benchmark schemes.**

***Index Terms*—Digital twin migration, multi-tier computing, multi-agent reinforcement learning**

I. Introduction

The emerging applications, e.g., holographic telepresence, brain-computer interaction, and extended reality, will enable a merger of digital and physical worlds. These applications have diverse performance requirements regarding the quality of experience, latency, and reliability, which are challenging to be fulfilled by traditional wireless systems [2]. To address these challenges, integrating digital twin technology with wireless networks becomes a promising solution to offer efficient interaction management for various entities under these applications [3]. Digital twin is a promising technology to establish the connection between physical objects and their corresponding digital virtual representations [4]. The digital twin of physical objects, e.g., mobile users and vehicles, are constructed according to their historical data and real-time running status. Digital twin objects utilize the real-time data generated by physical network entities to perform simulations and analysis in the digital space, which helps to enable close monitoring and provide intelligent decision-making for network entities [5].

To efficiently mirror and serve the physical network entities, the digital twins require continuously collecting real-time data from them and using machine learning for dynamic analysis, estimation, and prediction [6]. It involves low-latency interactions between network entities and their digital twins. The conventional cloud-based digital twin framework that deploys twin objects at the cloud centre may suffer significant latency and high communication costs for digital twin data synchronization, which reduces the service quality for mobile users [7]. To tackle these issues, the multi-tier computing paradigm integrating a large number of geographically distributed wireless edge servers is proposed to support services with different levels of intelligence and efficiency [8]. A primary research direction in multi-tier computing systems is to utilize computation offloading technology to alleviate users' resource bottlenecks when computing tasks [9]. Generally, a task can be divided into sub-tasks and offloaded to multiple edge servers for cooperative computing. Based on this, existing computation offloading schemes can be divided into sequential offloading and parallel offloading [10]. In sequential offloading [11], the sub-tasks are offloaded to servers in a time-sequential manner over a shared communication channel. For parallel offloading [12], [13], sub-tasks are offloaded simultaneously to servers over orthogonal communication channels. This work aims to deploy digital twins in multi-tier computing systems to enable signal processing and computing for twin objects at the network edge, which is able to handle the long latency issue caused by the conventional cloud-based structure. The digital twin objects are modelled and maintained by the edge servers instead of users, and they utilize the real-time status data of users to perform analysis in the digital space [4]. Note that there are no dependencies between data, and the user data are all processed by the digital twin models at the edge servers. Therefore, user data is concurrently offloaded to servers for processing, and the data synchronization process can be regarded as parallel data offloading.

Although multi-tier computing effectively improves relia-

Zhixiong Chen and Arumugam Nallanathan are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, U.K. (emails: {zhixiong.chen, a.nallanathan}@qmul.ac.uk).

W. Yi is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (email: wy23627@essex.ac.uk).

Jonathon Chambers is with the School of Engineering, University of Leicester, Leicester, U.K. (e-mail: jac89@leicester.ac.uk).

bility and latency performance for the interactions between users and digital twins, it also leads to operational complexity during service runtime due to the massively connected users and complex network dynamics [14]. Thus, it is essential to design an efficient digital twin access method for twin signal processing in multi-tier systems [2], [6]. To this end, there are two main aspects related to accessing the twin objects for mobile users: 1) *Digital Twin Placement*: In multi-tier computing systems, massive users may deploy and maintain their digital twins at edge servers. Due to the resource-intensive nature of maintaining digital twins, the digital twin placement strategy for users should be judiciously designed to reduce interaction latency between users and the digital twins [15]. 2) *Communication and Computation Resource Management*: To enable close monitoring of users at their corresponding digital twins, users are required to upload their real-time data to their corresponding digital twins for processing. Thus, wireless resources in multi-tier computing networks should be allocated for users to communicate with their digital twins. In addition, computing resources are required for digital twins to process users' data before applying for real-time applications. One server in the network may maintain multiple digital twins of different users. Thus, the computing resources management policies at the servers should be carefully designed [16].

To reduce the data synchronization latency from users to their corresponding digital twins, existing works focused on two aforementioned problems. Specifically, in [17], a potential game-based network selection approach was developed to reduce the data synchronization latency between the digital twin and vehicular users in heterogeneous vehicular networks. A reinforcement learning approach was proposed in [18] to optimize the network selection and power allocation strategies, which significantly reduced the delay and energy consumption of digital twin data synchronization in heterogeneous access networks. The multi-dimensional optimization of offloading policies, user association strategies, and transmit power in [19] efficiently reduced users' computation latency in the digital twin network. In [20], by integrating ultra-reliable and low latency communications into a digital twin-enabled mobile edge computing network, the co-design of the communication and computation scheme efficiently reduced the digital twin synchronization latency. The deep reinforcement learning-based digital twin association and bandwidth allocation scheme in [21] achieved a balance between twin model accuracy and latency. The joint digital twin placement and migration policy in [22] effectively reduced the system cost for maintaining digital twins in a wireless edge network.

Although the above digital twin placement and network resource management approaches in [17]–[21] improve the data transmission efficiency from physical objects and digital twin models, user mobility is less considered in designing placement strategies. In practical networks, users may move away from the coverage of the edge servers running their digital twin models to the area of other edge servers, which results in the temporally disconnecting of services. To tackle this issue, the digital twin models should be dynamically migrated to a more suitable server to improve the quality of service (QoS). In addition, the digital twin migration solution in [22] is a centralized algorithm and requires knowing the complete system-level information (including all users' communication conditions, data profiles, and workloads of all the edge servers) before making migration decisions. In practical multi-tier computing systems, gathering this information brings high signalling overheads. Moreover, the centralized solution has a scalability issue since the complexity rapidly scales with the number of users. To tackle thses issues, this work considers a decentralized digital twin migration framework where users make migration decisions for their digital twins with partially observed system information. To enable effective digital twin migration, we propose an agent contribution-aware multi-agent reinforcement learning (AC-MARL)-based digital twin migration algorithm to allow each user to learn a distributed migration policy. Note that the conventional MARL algorithms in [23], [24] utilize joint actions of agents to compute global rewards without measuring the contribution of each agent to the cooperative result. This may fail to encourage individual agents to sacrifice for greater global rewards. In this work, we compute a separate baseline for each agent to estimate their contributions to the global reward and facilitate cooperation among agents. The main contributions of this work are summarized as follows:

- We formulate an online digital twin migration and resource management problem in multi-tier computing networks to reduce data synchronization latency between the digital twin and the user. To solve this problem and enable distributed resources management, we first derive the optimal communication and computation resource allocation policies at each edge server by convex optimization methods. Then, we model the digital twin migration problem as a decentralized partially observable Markov decision process (Dec-POMDP) to capture the intrinsically complex system dynamics.
- To solve the Dec-POMDP, we develop an AC-MARL algorithm with centralized training and distributed execution to enable users to learn cooperative migration strategies for data synchronization latency minimization. Since using a global reward for each user may not encourage individual users to sacrifice for the greater global reward, we adopt a counterfactual baseline method to compute a separate baseline for each agent to estimate agents' contribution to the global reward and facilitate cooperation among agents. In addition, directly using the users' observations as the input of actor and critic networks may improve the learning complexity and hinder convergence. To this end, we devise an embedding matrix for coding agents' actions and states to accelerate the training of the proposed AC-MARL algorithm.
- We conduct simulations on two real-world mobility trace datasets, i.e., taxi mobility traces at Rome and San Francisco, to verify the effectiveness of the proposed algorithm. Specifically, the proposed digital twin migration approach is able to reduce 30% and 23.1% latency compared to the benchmark migration algorithms on the datasets of Rome and San Francisco, respectively.

The rest of this paper is organized as follows: Section

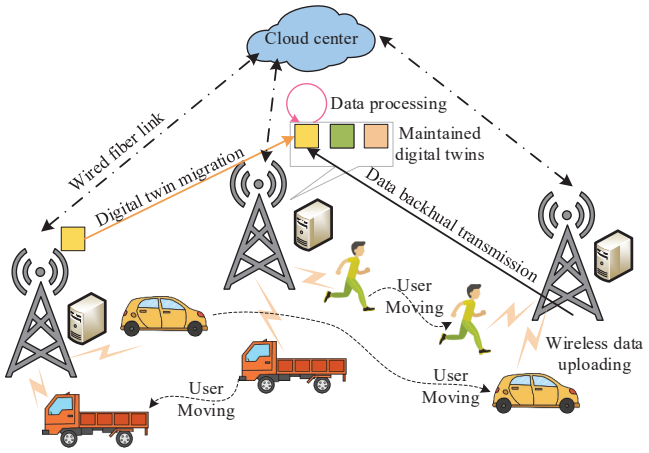


Fig. 1. Illustration of the considered digital twin network.

II introduces the system model and the long-term latency minimization problem. In section III, we derive the optimal communication and computation resource management strategy for each edge server and transform the original problem into a digital twin migration problem. Section IV illustrates the proposed AC-MARL-based digital twin migration algorithm. Section V evaluates the effectiveness of the proposed approaches by simulations. The conclusion is presented in Section VI.

TABLE I
NOTATION SUMMARY

Notation	Definition
$\mathcal{K}$; K;	Set of users; size of $\mathcal{K}$
$\mathcal{M}$; M	Set of MEC servers; size of $\mathcal{M}$
W_k	the size of user k's digital twin model
$x_{k,m}^{(t)}$	Digital twin placement indicator of user k at m-th server
$z_{k,m}^{(t)}$:	Connected indicator of user k to server m in slot t
ψ_t	Network bandwidth along the migration path
φ_t^M	Migration latency coefficient
B_m	Total bandwidth of MEC server m
$\beta_{k,m}^{(t)}$	Allocated bandwidth ratio for user k when it connected to MEC server m in slot t
$h_{k,m}^{(t)}$	Uplink channel gain from user k to MEC server m
p_k	Transmit power of user k
F_m	Available computation resource of MEC server m
$f_{k,m}^{(t)}$	Allocated computation resources to user k in time slot t
$c_{k,t}$	Required computing resources by user k to process its data in time slot t
ψ_t^{bh}	Bandwidth of the backhaul network
φ_t^{bh}	Backhaul latency coefficient
$D_{k,t}$	the generated data size of user k in slot t

II. SYSTEM MODEL

As shown in Fig. 1, we consider a digital twin-empowered multi-tier computing network, where K mobile users move in a geographical area covered by a set of M base stations connected to a cloud center. Each BS is equipped with an edge server to provide computing service for users. Users can connect to the edge servers through wireless channels and request computing services. The edge servers are interconnected via stable backhaul links, and each server can communicate with the cloud server via the wired fiber links. The cloud center is equipped with abundant computing and storage resources to efficiently handle computationally intensive tasks requiring global analysis. For simplicity, we denote the set of users by $\mathcal{K} = \{1, 2, \cdots, K\}$, the set of edge servers by $\mathcal{M} = \{1, 2, \cdots, M\}$. Each user k ($k \in \mathcal{K}$) has a digital twin, which is constructed by extracting the running state for serving specific applications. Similar to many existing works, e.g., [22], we consider that the digital twins of mobile users are modeled and maintained by the edge servers. In general, the number of users in the system is much larger than the number of edge servers. Thus, an edge server may maintain multiple users' digital twins, and all digital twins maintained in the same server share the communication and computing resources.

To better capture digital twin operation states and user mobility, we consider the system operates in a time-slotted structure, and the timeline is discretized into time frames, $\mathcal{T} = \{1, 2, \cdots, T\}$. The time-slotted model is widely utilized to characterize the service maintenance system, e.g., [21], [22], which can be regarded as a sampled version of a continuous-time model. In each time slot t ($t \in \mathcal{T}$), each user may generate some new data that required to be synchronized to its digital twin model for processing.

A. Digital Twin Placement Model

In the considered system, each mobile user possesses a digital twin maintained at one of the edge servers. To enable the digital twins to monitor their corresponding users closely, users are required to upload the real-time operational data to their digital twin for processing. In each time slot t, each user selects the edge server providing the strongest average signal strength as its locally connected server. Let $z_{k,m}^{(t)}$ denote the access association indicator of user k to server m, where $z_{k,m}^{(t)} = 1$ when user k is connected to server m, $z_{k,m}^{(t)} = 0$ otherwise. For each user k ($k \in \mathcal{K}$), the server that maintained its digital twin may be the locally connected server or other servers, called its digital twin server. We use $x_{k,m}^{(t)}$ to denote the digital twin placement indicator of user k, where $x_{k,m}^{(t)} = 1$ represents that user k' digital twin is placed at the server m in time slot t, and $x_{k,m}^{(t)} = 0$ otherwise. For ease of presentation, we use $\boldsymbol{x}_{k,t} = (x_{k,1}^{(t)}, x_{k,2}^{(t)}, \cdots, x_{k,M}^{(t)})$ to represent the digital twin placement decision of user k in time slot t and $\boldsymbol{x}_t = (\boldsymbol{x}_{1,t}, \boldsymbol{x}_{2,t}, \cdots, \boldsymbol{x}_{K,t})$ to denote all users' digital twin placement decisions in time slot t. Due to the resource-intensive nature of maintaining digital twins, maintaining digital twins in each edge server for each user incurs expensive transmission load, computation overhead, and energy consumption. In this work, we consider each digital twin can only be maintained by one edge server. Thus, we

have the following constraints for the digital twin placement decision:

$$\sum\nolimits_{m=1}^{M} x_{k,m}^{(t)} = 1, \forall k \in \mathcal{K}, \forall t. \tag{1}$$

In practice, the digital twin placement policy is closely related to its maintenance costs. To reach the desirable QoS, the digital twin should be dynamically migrated across servers as the user moves [2], [6]. In this work, we utilize latency as the measurement for the QoS, encompassing migration, computation, and communication latencies. Note that each user's digital twin can be deployed on multiple edge servers in practical systems. Our proposed solution in Section IV can be generalized to this case by repeating the process of finding the digital placement policy and removing the placement decision from the action space until reaching the number of digital twin maintenance servers.

B. Migration Latency Model

The migration latency is incurred when a user's digital twin is moved out from the previous server, which involves the digital twin model's backhaul transmission, processing, and queuing latencies among the servers on the migration path [25]. In general, the digital twin migration latency of each user k ($k \in \mathcal{K}$) increases with its digital twin model size and the hop distance from its original server to the current server. Specifically, the digital twin migration latency of user k from server m to server n can be expressed as [26]

$$T_{k,t}^{\mathrm{M},m\to n} = \begin{cases} 0, & \text{if } \ m = n, \\ \frac{W_k}{\psi_t} + \varphi_t^{\mathrm{M}} d(m,n), & \text{if } \ m \neq n. \end{cases} \tag{2}$$

where W_k is the size of user k's digital twin model, $d(m,n)$ denote the hop distance between server m and server n, ψ_t is the transmit rate for digital twin migration (i.e., the latency required to transmit one unit of data along the migration path), and φ_t^{M} is a positive coefficient. It is worth mentioning that if the original server m is same as the current server n, i.e., $m = n$, there is no migration of user k's digital twin model, thus the migration latency is $T_{k,t}^{\mathrm{M},m\to n} = 0$. Thus, the migration latency of user k in time slot t can be expressed as

$$T_{k,t}^{\mathrm{M}} = \sum_{m=1}^{M}\sum_{n=1}^{M} x_{k,m}^{(t-1)} x_{k,n}^{(t)} T_{k,t}^{\mathrm{M},m\to n}. \tag{3}$$

C. Communication Latency Model

In each time slot, the users generate new data and synchronize the data to their digital twin models for processing and maintenance. It is noted that user k ($k \in \mathcal{K}$) may not directly connect with its digital twin server. In this case, when user k requires to synchronize its data to its digital twin model, it first uploads its data to its locally connected server through wireless uplink transmission, and then the data is transmitted from its locally connected server to its digital server through backhaul links. Thus, the communication latency is composed of wireless uplink transmission latency and backhaul link transmission delay. In the following, we analyze these two kinds of latencies.

- **Wireless Transmission Latency**: In this work, we consider a frequency-division multiple access protocol for users transmitting their data to their locally connected servers, in which each edge server m ($m \in \mathcal{M}$) provides a total bandwidth B_mHz. Let $\beta_{k,m}^{(t)}$ denote the allocated bandwidth ratio for user k in slot t when it connected to edge server m and $\boldsymbol{\beta}_t = \{\beta_{k,m}^{(t)} : \forall k \in \mathcal{K}, \forall m \in \mathcal{M}\}$ to denote all servers' bandwidth allocation decisions for all users in time slot t. Note that the wireless bandwidth policy for each edge server is only determined by its connected users and is not related to users' digital twin placement policy. Denote the uplink channel gain from user k to edge server m in time slot t as $h_{k,m}^{(t)}$, and the transmit power of user k is p_k. The achievable transmission data rate of user k to server m is:

$$r_{k,m}^{(t)} = \beta_{k,m}^{(t)} B_m \log\Big(1 + \frac{p_k h_{k,m}^{(t)}}{\sigma^2}\Big), \tag{4}$$

  where σ^2 is the noise power. For each user k ($k \in \mathcal{K}$), let $D_{k,t}$ denote the generated data size that is required to upload to its digital twin server for processing in time slot t. Thus, the uplink transmission delay of user k to upload its generated data to its local connected server in slot t can be expressed as

$$T_{k,t}^{\mathrm{up}} = \sum_{m=1}^{M} z_{k,m}^{(t)} \frac{D_{k,t}}{r_{k,m}^{(t)}}. \tag{5}$$

- **Backhaul Transmission Latency**: The backhaul delay is incurred when the digital twin server and the locally connected server of user k are different. The backhaul delay primarily decided by the hop distance of the shortest path from the locally connected server to the digital twin server and the generated data size of user k. Specifically, the backhaul transmission latency of user k's data from edge server m to server n in time slot t is given by

$$T_{k,t}^{\mathrm{bh},m\to n} = \begin{cases} 0, & \text{if } \ m = n, \\ \frac{D_{k,t}}{\psi_t^{\mathrm{bh}}} + \varphi_t^{\mathrm{bh}} d(m,n), & \text{if } \ m \neq n. \end{cases} \tag{6}$$

  where ψ_t^{bh} is transmit rate of the backhaul network, φ_t^{bh} is a positive coefficient of the backhaul latency. Especially, when the locally connected server and the digital twin server of user k are them same, there is no backhaul transmission costs. Therefore, the backhaul transmission lateny of user k to transmit its generated data from its local connected server to its digital twin server is

$$T_{k,t}^{\mathrm{bh}} = \sum_{n=1}^{M}\sum_{m=1}^{M} z_{k,m}^{(t)} x_{k,n}^{(t)} T_{k,t}^{\mathrm{bh},m\to n}. \tag{7}$$

Through the above analysis, the overall communication latency of user k in time slot t is given by:

$$T_{k,t}^{\mathrm{Comm}} = T_{k,t}^{\mathrm{up}} + T_{k,t}^{\mathrm{bh}}. \tag{8}$$

D. Computation Latency Model

At each time slot t, the users upload their data to their digital twin servers for processing to maintain their digital twins.

Denote the available computation resource of edge server m as F_m. When multiple users maintain their digital twins at the same server, they share the computation resources of the server to maintain their digital twins [27]. The computing delay depends on three parts: 1) Processing capability of each server m; 2) The total number of devices that sharing server m and their computation load; and 3) The requested computing resources of the user k's data at time slot t. Let $c_{k,t}$ denotes the amount of computing resources (in CPU cycles) required by user k to process its data on its digital twin model in time slot t. Denote $f_{k,m}^{(t)}$ as the allocated computation resources to user k at edge server m in time slot t. For the sake of presentation, we use $\boldsymbol{f}_t = \{f_{k,m}^{(t)} : \forall k \in \mathcal{K}, \forall m \in \mathcal{M}\}$ to represent the computation resource allocation decisions of all the edge servers to all users in time slot t. Thus, the computation latency of user k in time slot t is

$$T_{k,t}^{\text{Comp}} = \sum_{m=1}^{M} x_{k,m}^{(t)} \frac{c_{k,t}}{f_{k,m}^{(t)}}. \tag{9}$$

Since the allocated computation resources of server m ($m \in \mathcal{M}$) cannot exceed its overall computation resources, the computation allocation policy of server m should satisfy

$$\sum_{k=1}^{K} x_{k,m}^{(t)} f_{k,m}^{(t)} \leq F_m. \tag{10}$$

E. Problem Formulation

According to the above analysis, the latency for user k in time slot t to synchronize its data to its digital twin for processing is given by:

$$T_{k,t} = T_{k,t}^{\text{M}} + T_{k,t}^{\text{Comm}} + T_{k,t}^{\text{Comp}}. \tag{11}$$

In this work, our objective is to jointly optimize the digital twin migration policies for each user, as well as the communication and computation resources management strategies at each edge server to minimize the long-term time-averaged digital twin synchronize latency for all users. To this end, we formulate the optimization problem as:

$$\mathcal{P}: \min_{\{\boldsymbol{x}_t, \boldsymbol{\beta}_t, \boldsymbol{f}_t,\}_{t=1}^{T}} \lim_{T\to\infty} \frac{1}{T} \sum\nolimits_{t=1}^{T} \sum\nolimits_{k=1}^{K} T_{k,t} \tag{12}$$

$$\text{s. t.} \quad \sum\nolimits_{k=1}^{K} \beta_{k,m}^{(t)} \leq 1, \forall m \in \mathcal{M}, \forall t, \tag{12a}$$

$$\sum\nolimits_{k=1}^{K} x_{k,m}^{(t)} f_{k,m}^{(t)} \leq F_m, \forall m \in \mathcal{M}, \forall t, \tag{12b}$$

$$\sum\nolimits_{m=1}^{M} x_{k,m}^{(t)} = 1, \forall k \in \mathcal{K}, \forall t, \tag{12c}$$

$$x_{k,m}^{(t)} \in \{0,1\}, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall t, \tag{12d}$$

$$0 \leq \beta_{k,m}^{(t)} \leq 1, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall t, \tag{12e}$$

$$0 \leq f_{k,m}^{(t)} \leq F_m, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}, \forall t. \tag{12f}$$

In problem $\mathcal{P}$, constraint (12a) signifies that the wireless bandwidth allocated to all users on each server cannot exceed its total available bandwidth resource. (12b) stipulates that the allocated computation resources to all users at any edge server cannot exceed its available computation resources. Constraints (12c) and (12d) impose restrictions on the digital twin placement decision, indicating that each digital twin can only be maintained by one edge server. Additionally, (12e) and (12f) restrict the wireless bandwidth and computation resources allocated to each user.

Finding the optimal solution for problem $\mathcal{P}$ is intractable since it requires exact user mobility patterns and complete system information over the entire horizon. However, it is impractical to collect all the relative information in advance in real-world scenarios. To tackle this challenge, we propose an efficient solution for problem $\mathcal{P}$ in the following sections to enable online resource management at the edge server and distributed digital twin migration for each user.

III. Online Resource Management and Problem Transformation

In this section, we propose an efficient communication and computation resources management algorithm at each edge server and transform $\mathcal{P}$ into an equivalent optimization problem that optimizes the digital twin migration policies for users. To this end, we decompose the computation and communication resource management problems from problem $\mathcal{P}$ at each edge server and derive their corresponding optimal communication and computation resource allocation policies by using convex optimization methods.

A. Communication and Computation Resource Management

According to the formulation of problem $\mathcal{P}$, the communication and computation resources allocation decisions in any time slot t, i.e., $\boldsymbol{\beta}_t$ and $\boldsymbol{f}_t$, only affects the digital twin synchronization latency in time slot t, and do not affect the digital twin synchronization latency in other slots. In addition, the communication and computation resource allocation decisions at each server do not interfere with that at other servers. In other words, the communication and computation resource allocation decisions can be solely optimized by each server at each time slot t. Inspired by this, we focus on the communication and computation resources allocation problem at a specific edge server m ($m \in \mathcal{M}$) in a specific time slot t, and derive the optimal communication and computation allocation policies. It is worth mentioning that the derived communication and computation allocation policies can be generalized to other edge servers at any time slot t. To this end, we first decompose the communication resources allocation problem at server m in time slot t as:

$$\mathcal{P}_1: \min_{\{\beta_{k,m}^{(t)}: \forall k \in \mathcal{K}\}} \sum_{k=1}^{K} z_{k,m}^{(t)} \frac{D_{k,t}}{r_{k,m}^{(t)}} \tag{13}$$
$$\text{s.t.} \quad (12a), (12e).$$

For problem $\mathcal{P}_1$, the second-order derivatives of the objective function (13) is given by

$$\frac{\partial^2 \left(\sum\limits_{k=1}^{K} \frac{z_{k,m}^{(t)} D_{k,t}}{r_{k,m}^{(t)}} \right)}{\partial \beta_{k,m}^{(t)} \partial \beta_{k,n}^{(t)}} = \begin{cases} \frac{2 z_{k,m}^{(t)} D_{k,t} (\beta_{k,m}^{(t)})^3}{B_m \log\left(1 + \frac{p_k h_{k,m}^{(t)}}{\sigma^2}\right)}, & m = n, \\ 0, & m \neq n. \end{cases}$$

It is straightforward to see that the Hessian matrix of the objective function (13) is semi-positive. Thus, the objective

function (13) is a convex function. In addition, the constraints of problem $\mathcal{P}_1$ are all linear functions. Consequently, problem $\mathcal{P}_1$ is a convex optimization problem. By solving the Karush-Kuhn-Tucker conditions of $\mathcal{P}_1$, we obtain its optimal solution as follows:

$$\beta_{k,m}^{(t,*)} = \frac{\sqrt{\frac{z_{k,m}^{(t)} D_{k,t}}{B_m \log(1+\frac{p_k h_{k,m}^{(t)}}{\sigma^2})}}}{\sum_{i=1}^{K} \sqrt{\frac{z_{i,m}^{(t)} D_{i,t}}{B_m \log(1+\frac{p_i h_{i,m}^{(t)}}{\sigma^2})}}}, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}. \quad (14)$$

From the optimal wireless bandwidth allocation policy in (14), if user k is not connected to server m, i.e., $z_{k,m}^{(t)} = 0$, server m will not allocate wireless bandwidth resource to it. In addition, when multiple users are connected to server m, they will share the available wireless bandwidth resources of server m. In this case, the allocated bandwidth for each user is decreased, and their digital twin synchronization latency would be increased.

Then, we decompose the computation resources allocation problem at server m in time slot t as:

$$\mathcal{P}_2: \quad \min_{\{f_{k,m}^{(t)}: \forall k \in \mathcal{K}\}} \quad \sum_{k=1}^{K} x_{k,m}^{(t)} \frac{c_{k,t}}{f_{k,m}^{(t)}} \quad (15)$$
$$\text{s.t.} \quad (12b), (12f).$$

Problem $\mathcal{P}_2$ is also a convex optimization problem, its proof is similar to the proof of problem $\mathcal{P}_1$, and thus omitted for brevity. Similar to the communication resource allocation problem, i.e., $\mathcal{P}_1$, we obtain the optimal solution of problem $\mathcal{P}_2$ by solving its Karush-Kuhn-Tucker conditions. The optimal computation resources allocation decision at server m in time slot t is given by:

$$f_{k,m}^{(t,*)} = \frac{\sqrt{x_{k,m}^{(t)} c_{k,t}}}{\sum_{i=1}^{K} \sqrt{x_{i,m}^{(t)} c_{i,t}}} F_m, \forall k \in \mathcal{K}, \forall m \in \mathcal{M}. \quad (16)$$

Similarly, from the optimal computation resources allocation decision at the server m in (16), the users that their digital twins do not maintained at server m (i.e., $x_{k,m}^{(t)} = 0$) will not obtain computation resources from server m. When multiple users deploy their digital twins on the same edge server, they collectively utilize the server's computational resources to update their digital twins based on the uploaded real-time data. This sharing of resources has the potential to result in an increase in synchronization latency for users' digital twins. Thus, it is essential to judiciously design the digital twin migration policies for users to reduce their synchronization latency as the users' mobility.

B. Problem Transformation

Up to now, each edge server can solve its optimal communication and computation resource management policies under any user access and digital twin placement state. In other words, we can compute users' optimal digital twin synchronization latency under any users' access state ($\{z_{k,m}^{(t)}: \forall k \in \mathcal{K}, \forall m \in \mathcal{M}\}$) and digital twin placement state $\boldsymbol{x}_t$. Based on the optimal communication and computation resource management policies at the servers, i.e., $\beta_{k,m}^{(t,*)}$ and $f_{k,m}^{(t,*)}$, the digital twin synchronization latency for user k ($k \in \mathcal{K}$) is given by

$$T_{k,t}^{*} = T_{k,t}^{\mathrm{M}} + T_{k,t}^{\mathrm{bh}} + \sum_{m=1}^{M} \frac{z_{k,m}^{(t)} D_{k,t}}{\beta_{k,m}^{(t,*)} B_m \log(1+\frac{p_k h_{k,m}^{(t)}}{\sigma^2})} + \sum_{m=1}^{M} x_{k,m}^{(t)} \frac{c_{k,t}}{f_{k,m}^{(t,*)}}. \quad (17)$$

Thus, problem $\mathcal{P}$ can be transformed into the following equivalent digital twin migration problem:

$$\mathcal{P}_3: \quad \min_{\{\boldsymbol{x}_t\}_{t=1}^{T}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{k=1}^{K} T_{k,t}^{*} \quad (18)$$
$$\text{s.t.} \quad (12c), (12d).$$

Note that, in the above analysis, we derive the optimal communication and computation resources management policies for each edge server and substitute them into problem $\mathcal{P}$. As a result, problem $\mathcal{P}$ is transformed into problem $\mathcal{P}_3$. This transformation does not change the optimality of problem $\mathcal{P}$.

Directly solving problem $\mathcal{P}_3$ is impractical because it requires precise knowledge of user movement and system-level information throughout the entire time horizon. In addition, the digital twin of each user is managed by itself. Each user can only obtain a part of the system information instead of the complete system information. Based on the objective function (18), users' digital twin migration decisions interfere with each other. When multiple users maintain their digital twins at the same edge server, their computation delay may increase since they share the computation resources on this server. Thus, it is important to carefully design the digital twin migration policy for each user with their incomplete system information.

IV. Online Digital Twin Migration

In this section, we develop an efficient AC-MARL algorithm that solves problem $\mathcal{P}_3$ to enable distributed digital twin migration. Specifically, we first reformulate problem $\mathcal{P}_3$ as a Dec-POMDP, in which each agent only has its local observation instead of the complete system state to make its digital twin migration decision. To tackle this issue, we proposed a centralized training and distributed execution framework to enable collaborative learning among agents. Finally, to further facilitate agents' cooperation, we propose estimating agents' contribution to the global reward by computing a separate baseline for each agent in the training process.

A. Dec-POMDP Formulation

According to the above analysis, the digital twin migration problem $\mathcal{P}_3$ is intrinsically a sequential decision-making problem where each user with a partially observable environment. Thus, it is natural to model problem $\mathcal{P}_3$ as a Dec-POMDP and utilize the multi-agent reinforcement learning (MARL) approaches to solve it. To this end, we formulate the Dec-POMDP as a tuple of $\boldsymbol{G} = \langle \mathcal{K}, \mathcal{S}, \{\mathcal{A}_k\}_{k\in\mathcal{K}}, \mathcal{P}, \mathcal{O}, \mathcal{R}, \gamma \rangle$, where $\mathcal{K} = \{1, 2, \cdots, K\}$ is the set of K agents. In this

work, each user k corresponds to the agent k. $\mathcal{S}$ is the system state space. $\mathcal{A}_k$ is the action space of user k, i.e., the set of available actions of user k. Let $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_K$ denote the joint action space. In each time step, each agent k ($k \in \mathcal{K}$) simultaneously chooses an action $a_{k,t} \in \mathcal{A}_k$, forming a joint action $\boldsymbol{a}_t = (a_{1,t}, a_{2,t}, \cdots, a_{K,t}) \in \mathcal{A}$. In this work, each agent in time slot t only has a partial observation $\boldsymbol{o}_{k,t}$ of the system state $\boldsymbol{s}_t$ ($\boldsymbol{s}_t \in \mathcal{S}$) to make local action decisions. All the agents' local observations form the global state, i.e., $\boldsymbol{s}_t = (\boldsymbol{o}_{1,t}, \boldsymbol{o}_{2,t}, \cdots, \boldsymbol{o}_{K,t})$. $\mathcal{O}$ is the observation probability function, which gives the probability of observing $\boldsymbol{o}_{k,t}$ if action $\boldsymbol{a}_{k,t-1}$ is executed and the resulting state is $\boldsymbol{s}_t$ ($\boldsymbol{s}_t \in \mathcal{S}$). $\mathcal{P}$ denotes the transition function from any state $\boldsymbol{s}_t \in \mathcal{S}$ to any state $\boldsymbol{s}_{t+1} \in \mathcal{S}$ for the joint action $\boldsymbol{a}_t \in \mathcal{A}$. Since the objective of problem $\mathcal{P}_3$ is to minimize the sum of all users' digital twin synchronization latency, the agents all share the same reward function, $\mathcal{R} : S \times A \to \mathbb{R}$. $\gamma \in [0, 1]$ is a discount factor.

The agents interact with the environment in each time slot to enhance their policies for reducing their digital twin synchronization latency. Specifically, in each slot t, each agent k obtains its observation $\boldsymbol{o}_{k,t}$ from the global environment state $\boldsymbol{s}_t = \{\boldsymbol{o}_{k,t} : \forall k \in \mathcal{K}\}$, then it takes action $a_{k,t} \in \mathcal{A}_k$ and obtains a reward $r_{k,t}$. Then, the environment transfer to a new state $\boldsymbol{s}_{t+1}$. The formulation of the Dec-POMDP elements of each user are given as:

- **Observation**: The observation of each user k ($k \in \mathcal{K}$) is its accessible information, which is defined as a tuple of its channel gain $h_{k,t}$, the generated data size $D_{k,t}$, the required CPU cycles for its digital twin to process the generated data $c_{k,t}$, its digital twin model size $S_{k,t}$, and locally connected server's ID, i.e, $m^{\mathrm{L}}_{k,t} = \sum_{m=1}^{M} z^{(t)}_{k,m} m$. Thus, the observation of user k in time slot t can be expressed as
$$\boldsymbol{o}_{k,t} = \left\langle m^{\mathrm{L}}_{k,t}, h_{k,t}, D_{k,t}, c_{k,t}, S_{k,t} \right\rangle. \tag{19}$$
- **Action**: In each time slot t, each user k can migrate its digital twin model to any of the servers. Thus, the action of each user k is defined as $a_{k,t} \in \mathcal{M}$ ($k \in \mathcal{K}$).
- **Reward**: Since the objective of problem $\mathcal{P}_3$ is to minimize the sum of all users' digital twin synchronization latency, all agents receive the same reward. In each time slot, the rewards of the agents are
$$r_{1,t} = r_{2,t} = \cdots = r_{K,t} = -\sum\nolimits_{k=1}^{K} T^*_{k,t}. \tag{20}$$

The primary challenges of solving the above Dec-POMDP arise from the intricate interplay between users and the complex dynamics of the multi-tier computing network. In this work, we adopt a MARL-based approach to solve the Dec-POMDP. In general, there are three MARL frameworks, i.e., 1) centralized training with centralized execution, 2) distributed training and distributed execution, and 3) centralized training with distributed execution [28]. For the centralized training with centralized execution framework, all agents are required to communicate their local observations in both the training and execution stages, which may induce expensive communication costs. In addition, the distributed training and distributed execution framework enables each agent to independently learn a policy based on its historical experience data. Since each agent can only obtain its local observation instead of the complete system state in the training and execution stages, independent agent learning is hard to learn cooperative migration strategies due to the lack of interactions between agents.

To efficiently solve the Dec-POMDP, we develop an AC-MARL-based digital twin migration algorithm to learn coordinated migration policies for users. Considering the fact that the agents only have their local observations for the decision-making, the proposed AC-MARL algorithm adopts the actor-critic framework to enable centralized training and decentralized execution. Specifically, each agent has an actor network, and a centralized critic network is deployed at the cloud server to evaluate the actions chosen by the agents' actor networks. The framework of the developed AC-MARL-based migration algorithm is shown in Fig. 2(a). In the training phase, the critic network is responsible for estimating the state value (or state-action value) based on the global state of the environment to facilitate agents' actor networks to learn cooperative migration strategies. After training, each agent independently makes decisions based on local observations only using its local actor network.

B. AC-MARL-based Digital Twin Migration Framework

In this subsection, we introduce the proposed AC-MARL-based digital twin migration algorithm. To better illustrate the proposed AC-MARL-based digital twin migration algorithm, we first briefly introduce the conventional multi-agent actor-critic algorithm. In the conventional multi-agent actor-critic algorithms, the critic network uses the global state $\boldsymbol{s}_t$ as its input and outputs the state value $v(\boldsymbol{s}_t; \boldsymbol{w})$, where $\boldsymbol{w}$ is the critic network parameter. Then, the critic network estimates the temporal difference (TD) error as $\delta_t = r_t + \gamma v(\boldsymbol{s}_{t+1}; \boldsymbol{w}) - v(\boldsymbol{s}_t; \boldsymbol{w})$ and return to each actor network. Each actor network computes its policy gradient based on the policy gradient theorem as $\boldsymbol{g}_k(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}_k) = (r_t + \gamma v(\boldsymbol{s}_{t+1}; \boldsymbol{w}) - v(\boldsymbol{s}_t; \boldsymbol{w})) \nabla_{\boldsymbol{\theta}_k} \ln \pi(a_k \,|\boldsymbol{o}_t; \boldsymbol{\theta}_k)$. Finally, the critic network and the actor network of each agent k ($k \in \mathcal{K}$) are updated as
$$\begin{cases} \boldsymbol{w} = \boldsymbol{w} - \eta_c \delta_t \nabla_{\boldsymbol{w}} v(\boldsymbol{s}_t; \boldsymbol{w}), \\ \boldsymbol{\theta}_k = \boldsymbol{\theta}_k + \eta_a \boldsymbol{g}_k(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}_k), \end{cases} \tag{21}$$
where η_c and η_a are the learning rate of the critic network and agent k's actor network, respectively. However, the TD error in the conventional multi-agent actor-critic algorithms only considers the global rewards. The computed policy gradient for each actor network, i.e., $\boldsymbol{g}_k(\boldsymbol{s}_t, \boldsymbol{a}_t; \boldsymbol{\theta}_k)$, does not measure the contribution of each agent to the global rewards. This may fail to encourage individual agents to sacrifice for the greater global reward. To tackle this issue, we adopt the counterfactual baseline method proposed in [29] to compute a separate baseline for each agent to estimate agents' contribution to the global reward and facilitate cooperation among agents.

To estimate the contribution of agents, the critic network is utilized to compute the state-action values (i.e, $Q(\boldsymbol{s}_t, \boldsymbol{a})$) for all agents' available actions instead of the state value $v(\boldsymbol{s}_t)$ in the conventional multi-agent actor-critic algorithms,

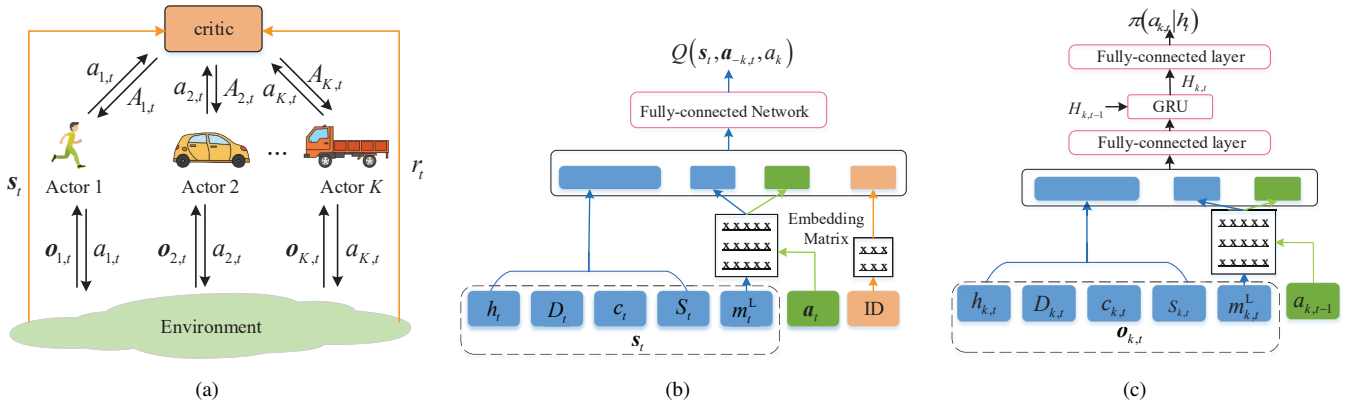


Fig. 2. The architecture of the AC-MARL-based digital twin migration algorithm : (a) the overall architecture of the AC-MARL algorithm, (b) the architecture of the critic network, (c) the architecture of the actor network.

as shown in Fig. 2(b). To this end, an intuitive method is to input the state $\boldsymbol{s}_t$ to the critic network and output the corresponding state-action value $Q(\boldsymbol{s}_t, \boldsymbol{a}) : \forall \boldsymbol{a} \in \mathcal{A}$. However, this intuitive method is impractical to train the critic network since it requires the number of neurons in the output layer of the critic network to equal the size of the joint action space, i.e., M^K, which scales with the number of agents and the size of their available actions. Thus, we use the critic network to estimate one agent's Q-values in a single forward process. The critic network moves other agents' actions as part of the input when it estimates the Q-values for a specific agent. For ease of presentation, we use a_k to denote agent k's action and $\boldsymbol{a}_{-k,t}$ to denote all agents' actions except agent k. For each agent k, the critic network estimates the Q-values of its available actions, i.e., $\{Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_k)) : \forall a_k \in \mathcal{M}\}$ based on the system state $\boldsymbol{s}_t = \{\boldsymbol{o}_{1,t}, \boldsymbol{o}_{2,t}, \cdots, \boldsymbol{o}_{K,t}\}$ and the fixed actions of other agents $\boldsymbol{a}_{-k,t}$. Specifically, when the critic network computes the Q-values for agent k, it uses the global state s_t, other agents' actions $\boldsymbol{a}_{-k,t}$, and the agent id of user k as its input, then output the Q-values for agent k, i.e., $(Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_k)) : \forall a_k \in \mathcal{M})$.

For the input of the critic network, the agents' locally connected server and agents' action is represented by the server order number, and the agents' ID is defined as the user order number. For example, $a_{k,t} = m \in \mathcal{M}$ indicates that agent k deploys its digital twin model at the m-th server in time slot t. The conventional design directly uses all agents' locally connected server $\{m^l_{k,t} \in \mathcal{M}, k \in \mathcal{K}\}$, the agents actions $(a_{k,t} = m \in \mathcal{M})$, and the agent ID $k (k \in \mathcal{K})$ as the input variables of the critic network. In this method, the server order number and agent order number would affect the output of the critic network. To eliminate the influence of servers' order numbers and agents' order numbers on the output of the critic network, many existing works, e.g., [29], use the one-hot coding of the server order and agent order as the input of the critic network. However, the one-hot coding scheme would induce a large input space that scales linearly in the number of agents and actions, i.e., KM input variables for the locally connected server state and agents' actions, as well as K input variables for the agent ID. This may negatively affect learning convergence. To tackle this issue, we provide a new design of the critic network as shown in Fig. 2(b), which adopts embedding matrices to code the locally connected server state, the agents' actions and the agent ID to reduce the dimension of the input space. Note that each agent's locally connected server and action are represented as the corresponding server order number. Thus, we use the same embedding matrix to code these two variables. For the agent ID, we also use an extra coding matrix to code it into a low-dimensional vector.

Through the above analysis, the centralized critic network is constructed to estimate the Q-values for agents. When estimating agent k's Q-values, the critic network uses the state $\boldsymbol{s}_t$, other agents' actions $\boldsymbol{a}_{-k,t}$, and agent k's ID as the input, in which all agents' locally connected server state and actions, as well as the agent k's ID are encoded by the corresponding embedding matrices. Then, these input variables are passed through the critic network and output agent k's Q-values, i.e., $(Q(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_k) : \forall a_k \in \mathcal{M})$. The critic network is trained using the TD(λ) scheme. In particular, the critic network is update by gradient descent to minimize the following loss function:

$$L^{\mathrm{c}}_t(\boldsymbol{w}) = \left(y^{(\lambda)}_t - Q(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t})\right)^2, \tag{22}$$

where the target $y^{(\lambda)}_t$ is $y^{(\lambda)}_t = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G^n_t$, where $G^n_t = \sum_{l=1}^{n} \gamma^{l-1} r_{t+l} + \gamma^n Q(\boldsymbol{s}_{t+n}, \boldsymbol{a}_{-k,t+n}, a_{k,t+n})$ is the n-step return.

Based on the above analysis, the critic network can compute the Q-values for each agent. In the following, we introduce the agents' actor network design and training. In the considered system, users can only obtain their local observations for decision-making because the global state is latent for users. The agents cannot choose their actions directly based on the state. However, it is possible for each agent k ($k \in \mathcal{K}$) to infer a belief over states based on its historical observations and actions, i.e., the probability of each state $p(\boldsymbol{s}_t | \boldsymbol{o}_{k,0:t-1}, a_{k,0:t}) = b_t$, and then use this information to decide upon an action [30]. Let $\{\boldsymbol{o}_{k,0}, a_{k,0}, \cdots, \boldsymbol{o}_{k,t-1}, a_{k,t-1}, \boldsymbol{o}_{k,t}, a_{k,t}\}$ to denote user k's historical observation-action up to time slot t. When user k executes action $a_{k,t}$ with the current belief $b_{k,t}$ and obtain the next observation $\boldsymbol{o}_{k,t+1}$, the next belief is estimated

by

$$b_{t+1}(\boldsymbol{s}_{t+1}) = \frac{\mathcal{O}(\boldsymbol{o}_{k,t+1}|\boldsymbol{s}_{t+1}, a_{k,t}) \sum_{\boldsymbol{s}_t \in \mathcal{S}} b_t(\boldsymbol{s}_t)\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_{k,t})}{\sum_{\boldsymbol{s}_{t+1} \in \mathcal{S}} \mathcal{O}(\boldsymbol{o}_{k,t+1}|\boldsymbol{s}_{t+1}, a_{k,t}) \sum_{\boldsymbol{s}_t \in \mathcal{S}} b_t(\boldsymbol{s}_t)\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_{k,t})}. \quad (23)$$

However, computing the above equation is impractical since it requires the exact knowledge of transition for the system state and agents' observations. To address this issue, similar to many existing works, e.g., [26], [31], we utilize an gate recurrent unit (GRU), a kind of recurrent neural network, to encode the full history (the agent's local observations and actions) and use the output hidden state of GRU to represent the latent system state and as the input of the remianing part of the actor network. Then, the actor network output the action probabilities. The architecture of the applied actor-network is shown in Fig. 2(c). Specifically, each agent uses its local observations and its action as the input of its actor-network. Then, the actor-network outputs the probabilities of the actions.

The objective of each actor network is to find an optimal action in each time step to maximize the accumulated reward. To train the actor-network for each agent k, we use the centralized critic network to estimate the Q-values of agent k's all available actions, i.e., $Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_k)) : \forall a_k \in \mathcal{A}_k$, and the actor network to compute the action probabilities of all the available actions based on its local observations. Then, the advantage function of agent k is given by:

$$A_k(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_{k,t})) = Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_{k,t})) - \sum_{a \in \mathcal{A}_k, a \neq a_{k,t}} \pi_k(a|\boldsymbol{o}_{k,t}) Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a)). \quad (24)$$

The advantage function estimates a baseline for agent k to reveal how its actions contribute to the global reward. Finally, agent k computes its gradient as

$$\boldsymbol{g}_{k,t} = \nabla_{\boldsymbol{\theta}_k} \log \pi(a_{k,t}|\boldsymbol{o}_{k,t}) A_k(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a_{k,t})) \quad (25)$$

and update its actor network as $\boldsymbol{\theta}_k = \boldsymbol{\theta}_k - \eta_a \boldsymbol{g}_{k,t}$.

For clarity, we summarize the detailed training steps of the proposed digital twin migration algorithm in Algorithm 1. For Algorithm 1, we have the following theorem:

Theorem 1. *The AC-MARL-based digital twin migration algorithm, i.e., Algorithm 1, is convergent. That is, with a compatible TD(1) critic network, the expected policy gradients satisfy*

$$\liminf_t \|\boldsymbol{g}_t\| = 0. \quad (26)$$

where $\boldsymbol{g}_t$ is the policy gradient in training step t.

Proof. Please see Appendix A. □

V. Simulation Results

In this section, we evaluate the performance of the proposed online digital twin migration approach using two real-world taxi mobility traces in Rome, Italy [32] and San Francisco, USA [33]. Similar to many works, e.g., [26], [34], we mainly focus on the taxi mobility traces in the central parts of these two cities. If not specified, the simulation settings are given as follows: We consider $K = 100$ users are moving in an 8 km $\times$ 8 km geographical area, where $M = 64$ edge servers are evenly deployed to provide computation services for users. Each server covers of 1 km $\times$ 1 km grid area. We calculate the hop distance between any two servers using the Manhattan distance. In each time slot, the data generation at each user k follows a Poisson process with parameter λ_k, where λ_k is randomly sampled from $[25, \lambda_{\max}]$. The digital twin model size of each user is uniformly sample from $[5, S_{\max}]$ MB. The required CPU cycles of each digital twin model to process one unit data, i.e., the data processing density, is uniformly selected from $[100, \rho_{\max}]$ cycles/bit. The coefficient of backhaul latency φ_t^{bh} is randomly selected from [1.0, 3.0]s/hop. The channel gain from user k to server m is modelled as $h_{k,t} = \rho_{k,t} d_k^{-\alpha}$ [35], [36], where $\rho_{k,t} \sim \exp(1)$

Algorithm 1 AC-MARL-based Digital Twin Migration Training

Initialize the critic network with random parameters $\boldsymbol{w}$ and copy to the target critic network $\bar{\boldsymbol{w}}$; initialize the actor network parameters for each agent; Initialize buffer memory size B
for each epoch i **do**
  Empty the buffer memory
  for episode $e = 1 : B$ **do**
    while $t < T$ **do**
      Each agent choose an action based on its current observation $\boldsymbol{o}_{k,t}$
      Receive reward r_t
      $t = t + 1$
      Append episode e to buffer memory
  for time slot $t = 1 : T$ **do**
    Unroll states, actions and rewards in buffer memory B;
    Compute TD(λ) target $y_t^{(\lambda)}$ using target critic network;
    $y_t^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^n$
    $G_t^n = \sum_{l=1}^{n} \gamma^{l-1} r_{t+l} + \gamma^n Q(\boldsymbol{s}_{t+n}, \boldsymbol{a}_{-k,t+n}, a_{k,t+n})$
  for time slot $t = T : 1$ **do**
    Unroll states, actions and rewards in buffer memory B;
    Compute critic gradient as $\boldsymbol{g}_{\boldsymbol{w}} = \nabla_{\boldsymbol{w}}(y_t^{(\lambda)} - Q(\boldsymbol{s}_t, \boldsymbol{a}_t))$
    Update the critic network as $\boldsymbol{w} = \boldsymbol{w} - \eta_c \boldsymbol{g}_{\boldsymbol{w}}$
    Every C steps copy $\boldsymbol{w}$ to $\bar{\boldsymbol{w}}$
  for time slot $t = T : 1$ **do**
    Compute the advantage for each agent based on (24)
    Compute policy gradient for each agent based on (25)
    Update each actor network as $\boldsymbol{\theta}_k = \boldsymbol{\theta}_k - \eta_a \boldsymbol{g}_{k,t}$

TABLE II
Simulation Environment Settings

Parameter	Value
CPU frequency of edge server m: F_m	50 GHz
White Gaussian noise variance σ^2	2×10^{-13}
Bandwidth resources of server m: B_m	10 MHz
Path loss exponent α	2
Maximum data generation rate $\lambda_{\max}$	50
Maximum digital twin model size $S_{\max}$	50
Maximum data processing density $\rho_{\max}$	500
Transmit power of user k: p_k	30 dBm
Bandwidth of backhaul network ψ_t^{bh}	500 Mbps
Coefficient of migration latency φ_t^{M}	0.02 s/hop

TABLE III
HYPERPARAMETER OF LEARNING ALGORITHM

Parameter	Value
GRU hidden units	256
Actor hidden units	128
Actor hidden units	256
Embedding dimension of actions	2
Embedding dimension of agent ID	6
Total training steps	1000
Length of an episode	20
Learning rate of actor	0.0001
Learning rate of critic	0.001
Discount factor γ	0.9
Discount factor λ	0.95
Optimizer	Adam

is the small-scale fading gain between device k and server m, d_k is the distance from device k to server m, and α is the path loss exponent. Unless otherwise stated, other simulation environment settings and hyperparameters of the proposed AC-MARL algorithm are summarized in Table II and Table III, according to parameter settings of a typical edge computing network [26], [27], [35], [36]. To evaluate the effectiveness of the proposed approach, we compare it with the following benchmarks:

- Always migration (AM): The digital twin model of each user always migration to its locally connected server in each time slot.
- Never migration (NM): All users' digital twin models are placed on the servers that they are first placed on, no matter where the users that they belong to are moving to.
- Independent advantage actor-critic-based migration (I-A2CM): Each user possesses an actor and a critic network and independently trains its actor and critic network based on the advantage actor-critic (A2C) algorithm [37] using its local observation-action history. The agents do not communicate with each other and do not share their observations, actions, and parameters. Note that the applied critic network in I-A2CM is the same as the adopted fully-connected neural network in the proposed AC-MARL, while its actor network is a fully-connected neural network formed by removing the GRU unit in the actor network of the proposed approach.
- Multi-agent A2C-based migration (MA-A2CM): Similar to the proposed digital twin migration algorithm, MA-A2CM adopts the centralized training and decentralized execution framework. All the actor networks at each agent and the centralized critic network use the fully-connected neural networks. The input of each actor network is the corresponding user's observations, while the critic network's input is the global state. The training algorithm is similar to the A2C algorithm.
- Multi-agent recurrent A2C-based migration (MAR-A2CM): MAR-A2CM is a variant of the MA-A2CM approach. It replaces the fully connected actor network architecture in MA-A2CM with the actor-network architecture of the proposed approach but uses the same training algorithm as MA-A2CM.

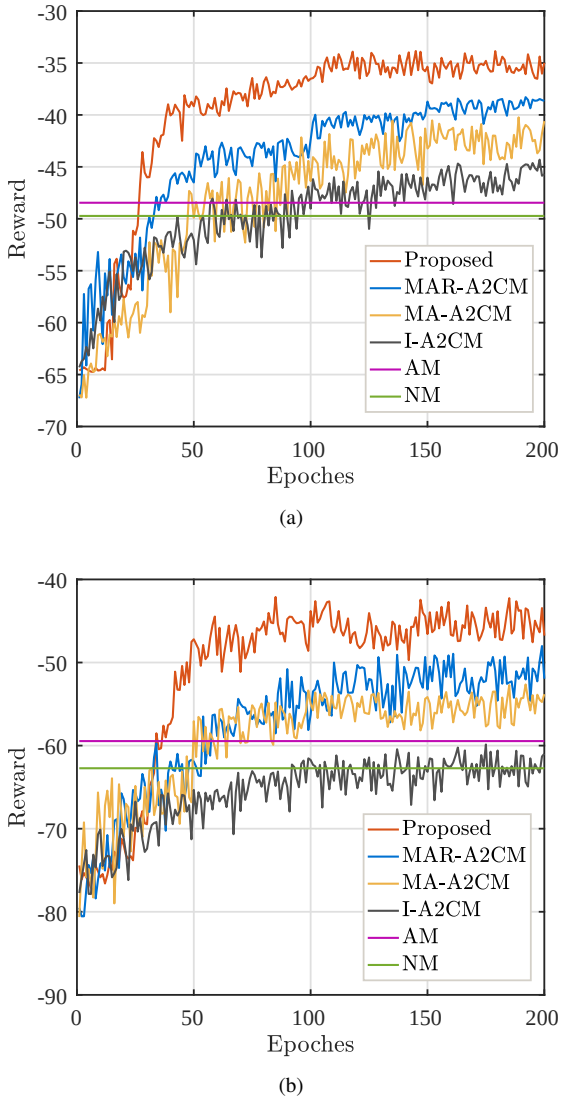


Fig. 3. Comparison of learning performance of different algorithms (a) on Rome dataset; (b) on San Francisco dataset.

In Fig. 3, we evaluate the learning performance of the proposed digital twin migration algorithm using the aforementioned two datasets. Each dataset includes K randomly selected mobility traces where each trace has 2000-time slots of the three-minute length of each. Note that we show the final performance of the always migration and never migration approaches since they do not involve the training of neural networks. For the learning-based migration schemes, it is observed that their rewards increase with the training process and then reach stable values. As expected, the proposed approach obtains a larger reward than the benchmarks on these two mobility trace datasets. Specifically, Fig. 3(a) evaluates all migration schemes' performance based on the Rome taxi trace. We can see that the proposed approach outperforms the benchmarks, and the rewards of the benchmarks satisfy MAR-A2CM $>$ MA-A2CM $>$ I-A2CM. The behind reasons are explained as follows: 1) Using the counterfactual baseline computes an agent-specific advantage function for each agent

to evaluate the contribution to the global reward is able to facilitate the users' cooperation to obtain a larger global reward. 2) Deploying GRU in the actor networks of agents is able to extract temporal features from agents' historical observations and estimate the hidden state of the Dec-POMDP. 3) The MARL algorithm is able to learn coordinated strategies among agents and achieve better performance than the independent agent learning, i.e., I-A2CM. I-A2CM may fail to exploit the interference among agents due to the lack of interactions between agents. Fig. 3(b) evaluates the proposed approach based on the taxi trace in San Francisco, showing a similar result to Fig. 3(a). These results verified the effectiveness of the proposed digital twin migration approach.

In Fig. 4, we show the impact of the server's computation capability on the average latency of users of the proposed scheme and the five baselines. It is observed that the average latency of all the evaluated algorithms decreases with the growth of servers' computation capabilities since the data processing latency can be reduced with the increase of servers' computation capabilities. The results show that the proposed algorithm adapts well to different computation capabilities of servers and outperforms the baseline algorithms. Specifically, from the evaluation results on Rome dataset in Fig. 4(a), the proposed algorithm is able to reduce around 30% latency compared to the MAR-A2CM algorithm. The performance gain mainly comes from using the counterfactual baseline to estimate each user's contribution to the global reward and thus facilitate users to sacrifice themselves to further reduce the average latency. From the results on the San Francisco dataset in Fig. 4(b), the proposed scheme saves 23% of time for digital twin synchronization compared to the benchmarks.

Fig. 5 shows how users' average latency varies with the available wireless bandwidth resources at the servers. Fig. 5(a) evaluates the proposed approach and benchmarks based on the Rome dataset. It is observed that the proposed approach achieves lower average latency than the benchmarks across all the available bandwidth configurations. Notably, the proposed approach is capable of reducing 27.4% of latency compared to the best benchmark scheme, i.e., MAR-A2CM. The evaluation on the San Francisco dataset in Fig. 5(b) shows a similar conclusion to Fig. 5(a). Specifically, the proposed approach saves up to 21.3% synchronization time for users' digital twins compared to the benchmarks. In addition, the average latency of all the schemes keeps decreasing with the increase of available bandwidth at servers because large bandwidth resources would reduce the wireless transmission latency for users.

Fig. 6 presents the performance of the proposed digital twin migration approach and MAR-A2CM under different maximum data generation rates (i.e., $\lambda_{\max}$) and user numbers. It is observed that the average latency of these two approaches keeps increasing with the increase of $\lambda_{\max}$ and user number on both Rome and San Francisco mobility trace datasets. The reason is that the growth of $\lambda_{\max}$ and user number will increase the generated data size of each user, as well as the total computation and communication load in the multi-tier computing network in each time slot, leading to the increase of the digital twin synchronization latency of all users.

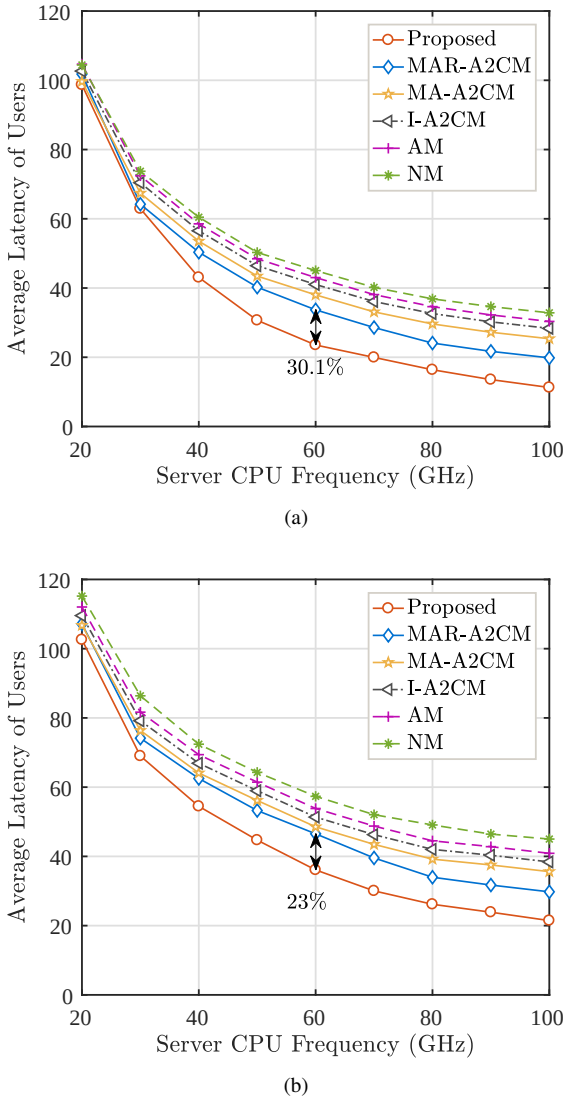
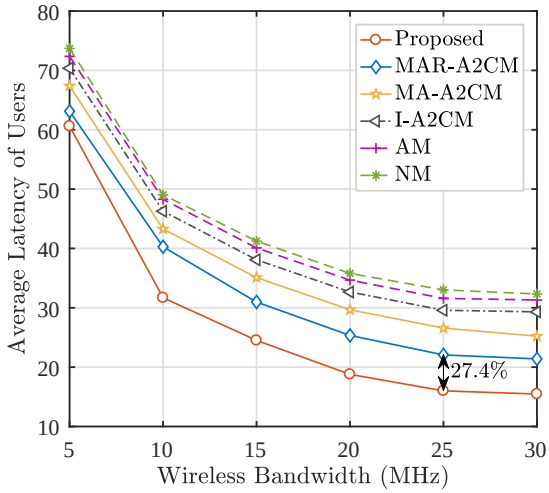


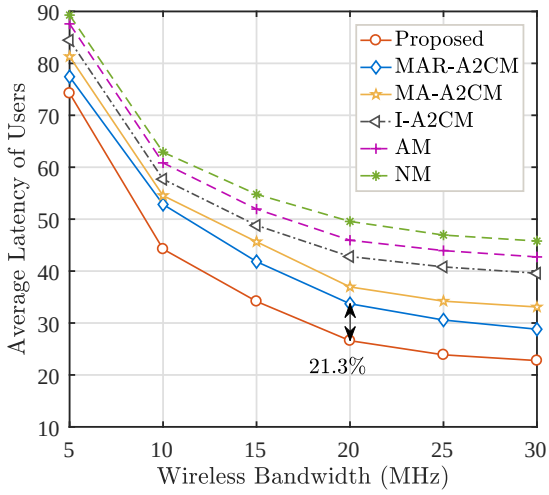
Fig. 4. Impact of the servers' computation capability on average latency of users: (a) on the Rome dataset; (b) on the San Francisco dataset.

In addition, the proposed approach outperforms MAR-A2C across different settings of $\lambda_{\max}$ and user number. Specifically, from Fig. 6(a), the proposed approach is able to reduce 28.1% latency compared to the MAR-A2CM scheme. This is because the proposed approach calculates a separate baseline for each user to characterize their contribution to the global reward in the training phase and facilitates users to sacrifice themselves for greater global reward. The experiment on the San Francisco dataset in Fig. 6(b) shows a similar conclusion to the results on the Rome dataset. Specifically, compared to MAR-A2CM, the proposed approach obtains lower digital twin synchronization latency and reduces up to 21.8% latency when $K = 80$ users are in the system.

Fig. 7 presents the impacts of maximum data processing density (i.e., $\rho_{\max}$) on the average digital twin synchronization latency of users of the proposed approach and MAR-A2CM. Fig. 7(a) shows the results on the Rome dataset. Compared to the MAR-A2CM approach, the proposed approach achieves

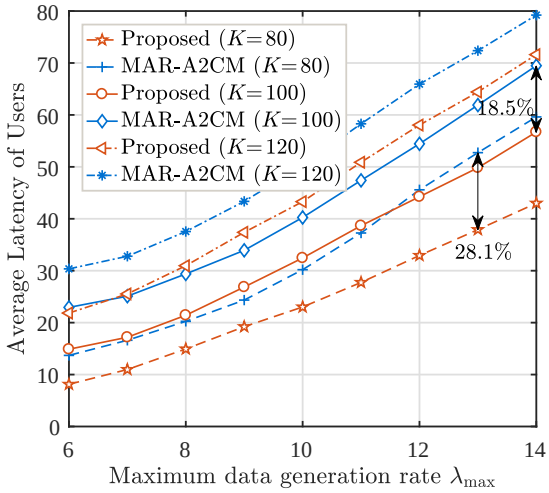


(a)

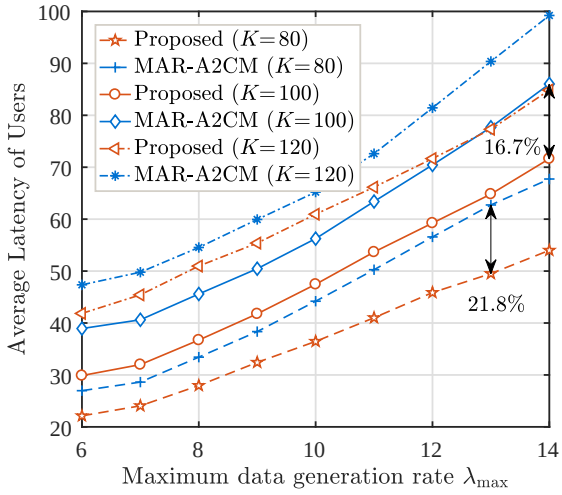


(b)

Fig. 5. Effect of servers' available bandwidth on average latency of users: (a) on the Rome dataset; (b) on the San Francisco dataset.



(a)



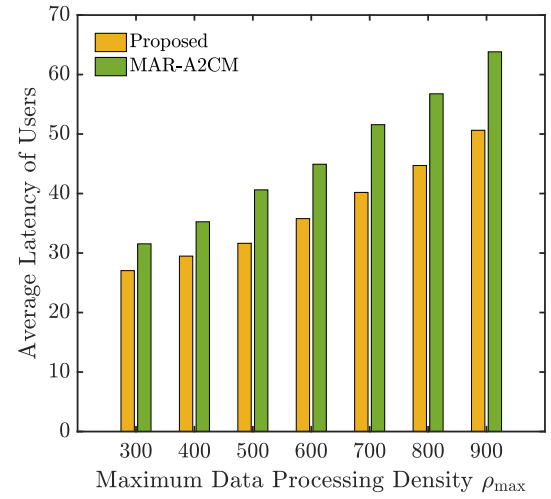
(b)

Fig. 6. Effect of users' maximum data generation rate on average latency of users: (a) on the Rome dataset; (b) on the San Francisco dataset.

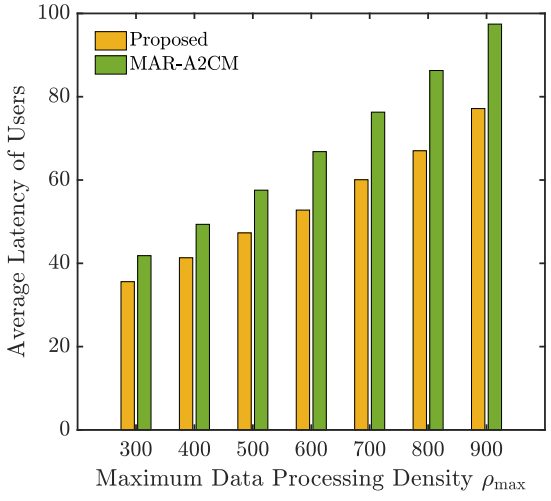
lower latency for users. In addition, along with the increases in $\rho_{\max}$, the proposed approach has a lower latency increasing speed than MAR-A2CM. The latent reason is that the proposed approach is more beneficial to enable users to learn a cooperative migration strategy than the MAR-A2CM approach. A similar simulation is conducted on the San Francisco dataset, as shown in Fig. 7(b). It is also observed that the proposed approach achieves a lower latency increasing speed along with the increases of $\rho_{\max}$ compared to MAR-A2C. These results indicate that the proposed scheme is able to reduce the digital twin synchronization latency for users efficiently.

VI. Conclusion

In this work, we have investigated the digital twin migration and resource management problem in the multi-tier computing system to minimize the data synchronization latency from users to their corresponding digital twins. To enable distributed resource management and digital twin migration, we first derived the optimal communication and computation resource management policies for each edge server using convex optimization methods. Then, we formulated the digital twin migration problem as a Dec-POMDP, in which users can only obtain partially observed system information to make migration decisions. To solve the Dec-POMDP, we proposed an AC-MARL-based digital twin migration approach. Unlike the conventional MARL algorithms, we adopted the counterfactual baseline to compute a separate baseline for each user to estimate its contribution to the global reward in the training process and facilitate cooperation among agents to reduce the overall latency. In addition, we devised embedding matrices to code users' observations and actions to accelerate the learning convergence for the proposed AC-MARL-based migration approach. Simulation results show that the proposed approach is able to capture users' mobility behaviour and facilitate users to learn a cooperative migration strategy for efficiently reducing their average data synchronization latency.



(a)



(b)

Fig. 7. Effect of the maximal data processing density on the average latency of users: (a) on Rome dataset; (b) on San Francisco dataset.

APPENDIX

A. *Proof of Theorem* 1

To prove the convergence of the proposed AC-MARL-based digital twin migration algorithm, we prove its expected policy gradient is equivalent to the policy gradient of the single-agent actor-critic algorithm in [38]. Firstly, the expected policy gradient of the Algorithm 1 in training step t is:

$$\boldsymbol{g}_t = \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_k \nabla_{\boldsymbol{\theta}_k} \log \pi_k\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) A_k\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)\Big], \tag{27}$$

where $A_k\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)$ is the advantage function of user k defined in (24). Let $b(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}) = \sum_{a\in\mathcal{A}_k} \pi_k(a|\boldsymbol{o}_t) Q(\boldsymbol{s}_t, (\boldsymbol{a}_{-k,t}, a))$ denote the counterfactual baseline of user k. Thus, the expected policy gradient $\boldsymbol{g}_t$ satisfy

$$\begin{aligned}\boldsymbol{g}_t = &\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_k \nabla_{\boldsymbol{\theta}} \log \pi\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) Q\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)\Big] \\ &- \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_k \nabla_{\boldsymbol{\theta}} \log \pi\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) b\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}\right)\Big],\end{aligned} \tag{28}$$

where $\boldsymbol{\theta}$ is the parameters of all the actor networks, i.e., $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_K\}$. Firstly, we consider the impact of the counterfactual baseline $b(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t})$ on the gradient $\boldsymbol{g}_t$, i.e., the second term on the right-hand side (RHS) of (28), we have

$$\boldsymbol{g}_b = -\mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_k \nabla_{\boldsymbol{\theta}} \log \pi\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) b\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}\right)\Big], \tag{29}$$

where the expectation is with respect to the state-action distribution, induced by the joint policy $\boldsymbol{\pi}$. Let $d_{\boldsymbol{\pi}}(\boldsymbol{s})$ be the discounted ergodic state distribution as defined in [39]. Then, the gradient $\boldsymbol{g}_b$ can be written as

$$\begin{aligned}\boldsymbol{g}_b = &-\sum_{\boldsymbol{s}_t} d_{\boldsymbol{\pi}}(\boldsymbol{s}) \sum_k \sum_{\boldsymbol{a}_{-k}} \boldsymbol{\pi}\left(\boldsymbol{a}_{-k,t}|\boldsymbol{s}_t - \boldsymbol{o}_{k,t}\right) \cdot \\ &\sum_{a_k} \pi_k(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k) \nabla_{\boldsymbol{\theta}} \log \pi_k\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) b\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}\right) \\ = &-\sum_{\boldsymbol{s}_t} d_{\boldsymbol{\pi}}(\boldsymbol{s}) \sum_k \sum_{\boldsymbol{a}_{-k}} \boldsymbol{\pi}\left(\boldsymbol{a}_{-k,t}|\boldsymbol{s}_t - \boldsymbol{o}_{k,t}\right) \cdot \\ &\sum_{a_k} \nabla_{\boldsymbol{\theta}} \pi_k\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) b\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}\right) \\ = &-\sum_{\boldsymbol{s}_t} d_{\boldsymbol{\pi}}(\boldsymbol{s}) \sum_k \sum_{\boldsymbol{a}_{-k}} \boldsymbol{\pi}\left(\boldsymbol{a}_{-k,t}|\boldsymbol{s}_t - \boldsymbol{o}_{k,t}\right) \cdot b\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}\right) \nabla_{\boldsymbol{\theta}} 1 \\ = &\, 0.\end{aligned} \tag{30}$$

Thus, $b(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t})$ do not affect the expected policy gradient $\boldsymbol{g}_t$. That is, the counterfactual baseline $b(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t})$ do not affect the convergence of the the proposed AC-MARL-based digital twin migration algorithm. Substituting (30) into (28), we have

$$\begin{aligned}\boldsymbol{g}_t &= \mathbb{E}_{\boldsymbol{\pi}}\Big[\sum_k \nabla_{\boldsymbol{\theta}} \log \pi_k\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) Q\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)\Big] \\ &= \mathbb{E}_{\boldsymbol{\pi}}\Big[\nabla_{\boldsymbol{\theta}} \log \prod_k \pi_k\left(a_k|\boldsymbol{o}_{k,t};\boldsymbol{\theta}_k\right) Q\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)\Big] \\ &= \mathbb{E}_{\boldsymbol{\pi}}\left[\nabla_{\boldsymbol{\theta}} \log \pi\left(\boldsymbol{a}_t|\boldsymbol{s}_t\right) Q\left(\boldsymbol{s}_t, \boldsymbol{a}_{-k,t}, a_{k,t}\right)\right],\end{aligned} \tag{31}$$

where $\boldsymbol{a}_t = (\boldsymbol{a}_{1,t}, \boldsymbol{a}_{2,t}, \cdots, \boldsymbol{a}_{3,t})$ is the joint actions of all users. According to [38], the expected gradient of the AC-MARL algorithm is equivalent to the policy gradient of the single-agent actor-critic algorithm that has already been proved to be convergent under the following conditions: 1) The policy $\boldsymbol{\pi}$ is differentiable. 2) The learning rates of actor and critic networks are sufficiently low and satisfy $\eta_a < \eta_c$. Thus, the proposed MARL-based digital migration algorithm is converged.

REFERENCES

[1] Z. Chen, W. Yi, and A. Nallanathan, "Multi-agent reinforcement learning-based digital twin migration over wireless networks." submitted to IEEE International Conference on Communications (ICC) 2024.

[2] L. U. Khan, Z. Han, W. Saad, E. Hossain, M. Guizani, and C. S. Hong, "Digital twin of wireless systems: Overview, taxonomy, challenges, and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 4, pp. 2230–2254, 2022.

[3] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, and B. Jabbari, "Task offloading with multi-tier computing resources in next generation wireless networks," *IEEE J. Sel. Areas in Commun.*, vol. 41, no. 2, pp. 306–319, 2023.

[4] R. Saracco, "Digital twins: Bridging physical space and cyberspace," *Computer*, vol. 52, no. 12, pp. 58–64, 2019.

[5] E. Ak, K. Duran, O. A. Dobre, T. Q. Duong, and B. Canberk, "T6conf: Digital twin networking framework for ipv6-enabled net-zero smart cities," *IEEE Commun. Mag.*, vol. 61, no. 3, pp. 36–42, 2023.

[6] Q. Guo, F. Tang, T. K. Rodrigues, and N. Kato, "Five disruptive technologies in 6g to support digital twin networks," *IEEE Wireless Commun.*, pp. 1–8, 2023.

[7] K. Wang, W. Chen, J. Li, Y. Yang, and L. Hanzo, "Joint task offloading and caching for massive mimo-aided multi-tier computing networks," *IEEE Trans. Commun.*, vol. 70, no. 3, pp. 1820–1833, 2022.

[8] K. Wang, D. Niyato, W. Chen, and A. Nallanathan, "Task-oriented delay-aware multi-tier computing in cell-free massive mimo systems," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2000–2012, 2023.

[9] Z. Cheng, M. Min, M. Liwang, L. Huang, and Z. Gao, "Multiagent ddpg-based joint task partitioning and power control in fog computing networks," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 104–116, 2022.

[10] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8805–8819, 2020.

[11] A. A. Al-Habob, A. Ibrahim, O. A. Dobre, and A. G. Armada, "Collision-free sequential task offloading for mobile edge computing," *IEEE Commun. Letters*, vol. 24, no. 1, pp. 71–75, 2020.

[12] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

[13] Z. Chen, Z. Zhou, and C. Chen, "Code caching-assisted computation offloading and resource allocation for multi-user mobile edge computing," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4517–4530, 2021.

[14] D. Van Huynh, V.-D. Nguyen, S. Chatzinotas, S. R. Khosravirad, H. V. Poor, and T. Q. Duong, "Joint communication and computation offloading for ultra-reliable and low-latency with multi-tier computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 521–537, 2023.

[15] P. Bellavista, C. Giannelli, M. Mamei, M. Mendula, and M. Picone, "Application-driven network-aware digital twin management in industrial edge environments," *IEEE Trans. Industrial Informatics*, vol. 17, no. 11, pp. 7791–7801, 2021.

[16] L. U. Khan, E. Mustafa, J. Shuja, F. Rehman, K. Bilal, Z. Han, and C. S. Hong, "Federated learning for digital twin-based vehicular networks: Architecture and challenges," *IEEE Wireless Commun.*, pp. 1–8, 2023.

[17] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.

[18] J. Zheng, T. H. Luan, Y. Hui, Z. Yin, N. Cheng, L. Gao, and L. X. Cai, "Digital twin empowered heterogeneous network selection in vehicular networks with knowledge transfer," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12 154–12 168, 2022.

[19] D. Van Huynh, V.-D. Nguyen, S. R. Khosravirad, V. Sharma, O. A. Dobre, H. Shin, and T. Q. Duong, "Urllc edge networks with joint optimal user association, task offloading and resource allocation: A digital twin approach," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7669–7682, 2022.

[20] D. Van Huynh, S. R. Khosravirad, A. Masaracchia, O. A. Dobre, and T. Q. Duong, "Edge intelligence-based ultra-reliable and low-latency communications for digital twin-enabled metaverse," *IEEE Wireless Commun. Letters*, vol. 11, no. 8, pp. 1733–1737, 2022.

[21] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 5098–5107, 2021.

[22] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6g," *IEEE Internet of Things J.*, vol. 8, no. 22, pp. 16 219–16 230, 2021.

[23] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[24] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[25] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, 2019.

[26] J. Wang, J. Hu, G. Min, Q. Ni, and T. El-Ghazawi, "Online service migration in mobile edge with incomplete system information: A deep recurrent actor-critic learning approach," *IEEE Trans. Mobile Comput.*, pp. 1–14, 2022.

[27] Z. Chen, W. Yi, A. S. Alam, and A. Nallanathan, "Dynamic task software caching-assisted computation offloading for multi-access edge computing," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6950–6965, 2022.

[28] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.

[30] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[31] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in mec by extending multi-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, 2021.

[32] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "Crawdad roma/taxi," 2022. [Online]. Available: https://dx.doi.org/10.15783/C7QC7M

[33] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "Crawdad epfl/mobility," 2022. [Online]. Available: https://dx.doi.org/10.15783/C7J010

[34] S. Redhu, M. Anupam, and R. M. Hegde, "Optimal relay node selection for robust data forwarding over time-varying iot networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9178–9190, 2019.

[35] Z. Chen, W. Yi, Y. Liu, and A. Nallanathan, "Knowledge-aided federated learning for energy-limited wireless networks," *IEEE Trans. Commun.*, vol. 71, no. 6, pp. 3368–3386, 2023.

[36] Z. Chen, W. Yi, and A. Nallanathan, "Exploring representativity in device scheduling for wireless federated learning," *IEEE Trans. Wireless Commun.*, vol. 23, no. 1, pp. 720–735, 2024.

[37] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. International Conference on Machine Learning*, 20–22 Jun 2016.

[38] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[39] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances in Neural Information Processing Systems*, vol. 12, 1999.