Deep Reinforcement Learning for Mobility-Aware Digital Twin Migrations in Edge Computing

Yuncan Zhang, *Member, IEEE*, Luying Wang, and Weifa Liang, *Senior Member, IEEE*

***Abstract*—The past decade witnessed an explosive growth on the number of IoT devices (objects/suppliers), including portable mobile devices, autonomous vehicles, sensors and intelligence appliances. To realize the digital representations of objects, Digital Twins (DTs) are key enablers to provide real-time monitoring, behavior simulations and predictive decisions for objects. On the other hand, Mobile Edge Computing (MEC) has been envisioned as a promising paradigm to provide delay-sensitive services for mobile users (consumers) at the network edge, e.g., real-time healthcare, AR/VR, online gaming, smart cities, and so on. In this paper, we study a novel DT migration problem for high quality service provisioning in an MEC network with the mobility of both suppliers and consumers for a finite time horizon, with the aim to minimize the sum of the accumulative DT synchronization cost of all suppliers and the total service cost of all consumers requesting for different DT services. To this end, we first show that the problem is NP-hard, and formulate an integer linear programming solution to the offline version of the problem. We then develop a Deep Reinforcement Learning (DRL) algorithm for the DT migration problem, by considering the system dynamics and heterogeneity of different resource consumptions, mobility traces of both suppliers and consumers, and workloads of cloudlets. We finally evaluate the performance of the proposed algorithms through experimental simulations. Simulation results demonstrate that the proposed algorithms are promising.**

***Index Terms*—Digital twin synchronization, mobility-aware DT migration, cost modeling of DT migration, deep reinforcement learning algorithm, mobile edge computing.**

I. Introduction

THE past decade experienced an explosive growth on the number of Internet of Things (IoT) devices (objects), and these devices are connected with the Internet to enable users access, control, and monitor their surroundings anytime and anywhere [21]. However, most IoT devices are constrained by limited energy and computing resource, which means that they cannot perform computing-intensive tasks on themselves [3]. Instead, a virtual representation of an IoT device (an object) can be implemented in an edge server by leveraging the Digital Twin (DT) technology. The DT can provide real-time monitoring of the object, store all its historical data traces, and apply Artificial Intelligence (AI) and machine learning to simulate behaviors of the object and provide predictive decisions, and so on.

The growth of mobile IoT applications has increased the volume of data generated at the network edge. Mobile Edge Computing (MEC) that brings computing and storage capabilities to the edge of networks was conceived in a bid to fill the gap between centralized clouds and mobile devices [18]. Orthogonal to the development of DT technology, there is growing interest in DT-assisted service provisioning in MEC networks [7], [16]. For example, Li et al. [7] studied DT placements to improve user service satisfaction in an MEC network, by considering the Age of Information (AoI) of DT data. Liu et al. [16] addressed the problem of mobile users intelligently offloading their tasks to cooperative mobile edge servers with the DT assistance. Since each DT stores the data that were synchronized with its object to enable real-time simulation and provide various services, DT-assisted service provisioning in an MEC network incurs cost, due to the consumption of computing, storage, and communication resources.

The DT placement locations in the MEC network critically impact the cost of DT-assisted service provisioning. In this paper, objects (devices) that generate data and maintain their DTs in the network are regarded as *suppliers* and users requesting DT services are regarded as *consumers*. The DT placements of suppliers in an MEC network is challenging. On one hand, DT placements determine not only the synchronization cost between DTs and their suppliers but also the service cost of consumers requesting DT services. On the other hand, the mobility of both suppliers and consumers makes choosing DT placement locations become difficult. When a supplier or consumer moves out its AP coverage area, the DT synchronization cost with its supplier from the new location or the service cost of its consumers requesting the DT service may dramatically change, as the routing path between the DT location and its supplier/consumer location changes. To provide delay-sensitive services in a DT-empowered MEC network with mobile suppliers and consumers, placing multiple DT replicas is an efficient approach, which was studied in [33]. Another is the DT migration approach, which migrates DTs from their current locations to the locations near to their suppliers and/or consumers. Existing studies on DT migrations focused mainly on reducing the service latency including the task offloading latency, data synchronization latency, etc [2], [17], [25].

Received 30 May 2024; revised 28 November 2024; accepted 5 January 2025. Date of publication 10 January 2025; date of current version 10 April 2025. The work of Yuncan Zhang and Weifa Liang was supported by University Grants Committee in Hong Kong (HK UGC) under CityUHK Under Grant 7005845, Grant 8730103, Grant 9043510, Grant 9043668, and Grant 9380137. *(Corresponding author: Weifa Liang.)*

Yuncan Zhang and Weifa Liang are with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: yuncan.zhang@cityu.edu.hk; weifa.liang@cityu.edu.hk).

Luying Wang is with the School of Computer Science and Engineering, Central South University, Changsha 410017, China (e-mail: luyingwang@csu.edu.cn).

Digital Object Identifier 10.1109/TSC.2025.3528331

In this paper, we investigate cost-aware DT migrations in the MEC network under the mobility assumption of both suppliers and consumers. However, performing DT migrations is not easy, it poses important challenges. First, both the DT synchronization cost of suppliers and the service cost of consumers requesting DT services fluctuate significantly when DTs are migrated to new locations. Since each supplier usually has multiple consumers requesting its DT service, the supplier and its consumers may move to the opposite directions. Then, it is difficult to migrate the DT of the supplier to such a location that can reduce the total service cost. Second, as all historical update data of the supplier is stored at its DT, when the DT is migrated from one location to another, all historical update data of the DT must be transmitted from its current location to the new location too. Such a data migration overhead cannot be neglected. As frequent DT migrations lead to the excess migration cost, it is desirable to intelligently decide when to perform the DT migration for each supplier. Finally, prior to a DT migration, a new location needs to be identified to facilitate the DT migration, and the cloudlet at the location must have sufficient computing and storage resources to host the DT. In the worst scenario, a potential new location might be the migration destinations of multiple DTs, and it will not have sufficient resources to accommodate all such DT migrations. Thus, workload balancing among cloudlets should be taken into account when performing DT migrations. In this paper we will address the aforementioned challenges.

The novelty of this paper lies in the study of a novel cost-aware DT migration problem, which jointly optimizes the accumulative cost of DT synchronization and migrations, as well as the total service cost of consumers requesting DT services over a finite time horizon. A novel Deep Reinforcement Learning (DRL) algorithm is devised for the DT migration problem without the knowledge of future mobility profiles of suppliers and consumers.

The main contributions of this paper are given as follows.

- We consider DT-assisted service provisioning in edge computing with the mobility assumption of both suppliers (objects) and consumers (users), where the supplier needs to continually synchronize with its DT. We formulate a novel DT migration problem with the aim to minimize the total service cost that consists of the DT synchronization cost of suppliers, DT migration cost, and service cost of consumers requesting for DT services.
- We show that the DT migration problem is NP-hard, and propose an Integer Linear Programming (ILP) solution for it when the problem is small, providing that the mobility profiles of suppliers and consumers are given.
- We develop a novel DRL algorithm for the DT migration problem, by jointly considering the system dynamics and heterogeneity of network resource consumption, the mobility of suppliers and consumers, and workloads of computing and storage resources in cloudlets.
- We evaluate the performance of proposed algorithms for the DT migration problem through simulations, using real-world datasets. Simulation results show that the proposed algorithms are promising.

The remainder of the paper is organized as follows. Section II reviews related studies on the topic. Section III introduces the system model, cost modeling, and defines the problem formally. Section IV formulates an Integer Nonlinear Programming (INP) to the offline version of the problem and transforms the INP to an equivalent ILP. Section V devises a DRL algorithm for the DT migration problem. Section VI evaluates the proposed algorithms through simulations, and Section VII concludes the paper.

II. Related Work

As an emerging enabling technology, digital twin has attracted a lot of attentions in both academia and industry. Existing studies explored various services driven by DTs including inference services, federated learning (FL) services [30], and SFC-enabled services provisioning [6]. Particularly, intelligent services in DT-assisted MECs has been explored recently. For example, Li et al. [7], [13] investigated DT placements to improve user service satisfaction, and proposed algorithms for user satisfaction maximization problems under static and dynamic digital twin placement schemes. Li et al. [8], [9] considered query services for IoT applications built upon DT data in an MEC network, with the aim to optimize the accumulative freshness of query results while reducing query delays simultaneously. They proposed an approximation algorithm for the problem by exploring nontrivial trade-offs between the accumulative AoI of query results and the total query delay. Li et al. [11] also made use of continual learning to retrain service models in a DT-empowered MEC, using the update data from DTs to ensure that the service models maintain high fidelity on their services. Liang et al. [15] designed efficient algorithms to maximize the state freshness of DTs and a set of inference service models built upon DTs, while the state freshness of a DT or a service model is achieved through frequent synchronizations between the DT and its physical object.

Service migrations were investigated in cloud networks [4], [29]. For instance, Zhang et al. [29] considered a cost-minimizing data migration problem in a cloud platform by proposing two online algorithms with provable competitive ratios, where the competitive ratio of the overall migration cost to the total cost is determined by a control parameter. Huang et al. [4] studied VNF service instance migrations to improve the network throughput while minimizing the operational cost of the network, by adopting both horizontal and vertical scaling techniques. These mentioned migration approaches however perform service instance migrations for all mobile devices, regardless of their heterogeneous movement patterns. There were studies on traditional service migrations in MEC networks [12], [19], [27]. For example, Li et al. [12] studied mobility-aware service placements to minimize the total cost of task offloading, which consists of the computing cost, communication cost and migration cost. Wang et al. [27] investigated dynamic service migration based on Markov Decision Process (MDP). They approximated the underlying state space by the distances between users and their service locations, and showed several properties of the distance-based MDP for computing the optimal migration

policy. Miao et al. [19] formulated a service migration problem to minimize the long-term system delay that consists of the queuing, communication, and migration delays in small-cell MEC networks. They devised a reinforcement learning approach for real-time decisions on service migration for mobile users. Unfortunately, the aforementioned service migration approaches are not applicable to DT service migrations, as the latter needs to consider the mobility of both DT data suppliers and DT consumers simultaneously.

Several efforts on DT-assisted service provisioning in MEC networks were taken recently [2], [10], [14], [17], [25], [32], [34], particularly in DT migration issues [2], [17], [25], [31], [32]. For instance, Lu et al. [17] formulated an adaptive edge association problem for placing and migrating DTs of mobile IoT devices in edge servers to reduce the average system latency and improve user utility. They developed a DRL algorithm for the optimal DT placement and proposed a transfer learning method for DT migrations to deal with user mobility. They however did not consider the DT migration cost at all. Sun et al. [25] considered the task offloading problem in DT-empowered edge networks under user mobility with the aim to minimize the average offloading latency under the budget constraint on a long-term migration cost. Chen et al. [2] formulated a joint optimization problem of DT migration, communication and computation resource managements, with the aim to minimize the data synchronization latency between each user and its requested DT. They developed an agent-contribution-enabled multi-agent reinforcement learning algorithm for the problem. These mentioned works however focused on reducing the service latency without taking into account the cost of resource consumption of DT migration. Furthermore, these mentioned studies only considered the mobility of consumers, and ignored the mobility of suppliers. Zhang et al. [32] explored non-trivial trade-offs between the DT update cost and the total service cost of users requesting for DT service through intelligent DT placements and migrations, where the user service cost is the weighted sum of the end-to-end delay cost, the DT data transmission cost within the network, and the request processing cost.

Unlike the aforementioned studies that only considered the mobility of suppliers or consumers, in this paper we consider a cost-aware DT migration problem with the mobility assumption of both suppliers and consumers. We explore non-trivial trade-offs between the cost of DT placements and migrations and the service cost of users requesting for DT services. We aim to develop a novel DRL algorithm for the DT migration problem. It must be mentioned that this paper is an extension of a conference paper [31].

III. Preliminary

In this section, we first introduce the system model. We then provide notions, notations and cost modeling. We finally present the problem definition precisely.

A. System Model

We consider an MEC network $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of APs. Each AP is co-located with a cloudlet and the communication delay between the AP and its co-located cloudlet is negligible. Without loss of generality, an AP and its co-located cloudlet are used interchangeably if no confusion arises. Each cloudlet $n_j \in \mathcal{N}$ has limited computing capacity C_j and storage capacity $\mathcal{M}_j$ for hosting DTs. Let ζ_j and $\varkappa_j$ denote the usage costs per unit computing and storage resources in cloudlet $j \in \mathcal{N}$, respectively. $\mathcal{L}$ is a set of optical links between APs interconnecting them together. Let ξ_e denote the communication cost per unit data on link $e \in \mathcal{L}$.

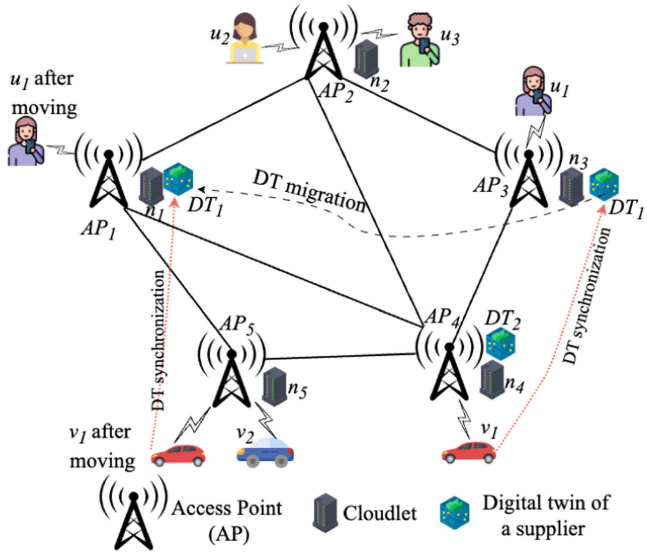


Fig. 1. An illustrative example of an MEC network with five APs and co-located cloudlets. There are two suppliers v_1 and v_2, and their DTs are deployed in cloudlets n_3 and n_4 initially. There are three consumers u_1, u_2 and u_3 requesting for DT services of the suppliers.

Let V and U denote the sets of suppliers and consumers, respectively. For the sake of convenience, we use i, j, and k to represent the indices of a supplier, a cloudlet, and a consumer respectively. Each supplier $v_i \in V$ has a DT, denoted by DT_i, placed in a cloudlet, which needs to allocate the amount of Φ_i of storage resource for its DT data and the amount c_i of computing resource for its DT data processing. Each consumer $u_k \in U$ requests the DT service from a supplier $r_{u_k} \in V$.

Given a finite time horizon $\mathbb{T}$ that is divided into equal time slots, i.e., $\mathbb{T} = \{1, 2, \ldots, T\}$, we assume that both suppliers and consumers are allowed to move around within the network at different time slots. Considering the mobility of suppliers and consumers, let $g_{v_i}(t) \in \mathcal{N}$ denote the cloudlet at which supplier $v_i \in V$ is located at time slot $t \in T$, and $g_{u_k}(t)$ can be defined similarly. Denote by $U(v_i, t) \subseteq U$ the set of consumers requesting service of DT_i of supplier $v_i \in V$ at time slot $t \in \mathbb{T}$. We further assume that each user $u_k \in U$ only requests one DT service at each time slot t, i.e., $U(v_i, t) \cap U(v_{i'}, t) = \emptyset$ if $i \neq i'$. For the sake of convenience, the notations in the system model are summarized in Table I.

Fig. 1 is an illustrative example of DT-assisted service provisioning in a MEC network, where consumer u_1 requests the update data of supplier v_1 by querying its DT, DT_1. Consumers u_2 and u_3 request the update data of supplier v_2 by querying

TABLE I
TABLE OF NOTATIONS IN SYSTEM MODEL

Notation	Descriptions
$G = (\mathcal{N}, \mathcal{L})$	An MEC network with a set $\mathcal{N}$ of APs and a set $\mathcal{L}$ of links
C_j and $\mathcal{M}_j$	The computing capacity C_j and storage capacity $\mathcal{M}_j$ of cloudlet $n_j \in \mathcal{N}$
ζ_j and $\varkappa_j$	The costs per unit computing and storage resource usages of cloudlet $n_j \in \mathcal{N}$
ξ_e	The communication cost per unit data on link $e \in \mathcal{L}$
V, U	Set V of suppliers and set U of consumers
DT_i	DT of supplier $v_i \in V$
Φ_i and c_i	Required amounts of storage and computing resources for hosting DT_i of supplier v_i
$gd_i(t)$	the amount of data generated by $v_i \in V$ at time slot t
r_{u_k}	The requested supplier of consumer $u_k \in U$, where $r_{u_k} \in V$
$\mathbb{T}$	The monitored period that is divided into equal slots, where $\mathbb{T} = \{1, 2, \cdots, T\}$
$U(v_i, t)$	The set of consumers requesting service on DT_i of supplier v_i at time slot t
$g_{v_i}(t)$ or $g_{u_k}(t)$	The cloudlet at which supplier v_i or consumer u_k is located at time slot $t \in T$
$x_{i,j}(t)$	Binary variable to indicate whether DT_i of supplier $v_i \in V$ is placed in cloudlet $n \in \mathcal{N}$ at time slot $t \in T$ or not
ϵ_i and $t_0^{(i)}$	Constants that determine the amount of data after being processed by DT_i, where $0 < \epsilon_i < 1$ and $t_0^{(i)} \geq 1$
$P_{n_j, n_{j'}}$	The shortest path in G between cloudlets n_j and $n_{j'}$
G_{v_i}	A copy of the MEC network $G(\mathcal{N}, \mathcal{L})$ for supplier v_i with edge weight $w'^{(t)}_i(\cdot)$ defined in Eq. (4)
$T^{opt}_{v_i, n_i(t)}$	A Steiner tree rooted at cloudlet $n_i(t)$ in G_{v_i} that hosts DT_i of supplier v_i and spanning the cloudlets at which consumers in $U(v_i, t)$ are located
$\mathcal{E}^{syn}_i(t)$	DT synchronization cost of supplier v_i at time slot t
$\mathcal{E}_{u_k}(t)$	Service request cost of consumer $u_k \in U$ at time slot t
$\mathcal{E}^{mig}_i(t)$	Migration cost of DT_i of supplier v_i at time slot t

its DT, DT_2. Initially supplier v_1 and consumer u_1 are located at AP_4 and AP_3, respectively, and DT_1 is placed at cloudlet n_3. Then, supplier v_1 moves to AP_5 and consumer u_1 moves to AP_1. To reduce the service cost, DT_1 of supplier v_1 is migrated from cloudlet n_3 to cloudlet n_1.

B. Cost Modeling

We detail the total cost of consumers requesting information from DTs for a given time horizon, where DTs of suppliers are migrated between cloudlets, considering the mobility of both consumers and suppliers. Let binary variable $x_{i,j}(t)$ indicate whether DT_i of supplier $v_i \in V$ is placed in cloudlet $n_j \in \mathcal{N}$ at time slot $t \in \mathbb{T}$, i.e., $x_{i,j}(t) = 1$ or not $x_{i,j}(t) = 0$.

DT synchronization cost of each supplier: At time slot $t \in \mathbb{T}$, supplier $v_i \in V$ generates the amount $gd_i(t)$ of update data for its DT_i updating, and the generated data then is transmitted from cloudlet $g_{v_i}(t)$ at which v_i is located to the cloudlet that hosts DT_i at time slot t. We assume that DT_i of supplier v_i needs to store all update data generated by v_i locally to ensure its functionality. Then, the amount of cumulative update data of DT_i stored at time slot t is

$$\Phi_i(t) = \sum_{t'=1}^{t} gd_i(t') \tag{1}$$

Notice that DT_i of supplier v_i at each time slot is placed at one cloudlet only, i.e.,

$$\sum_{n_j \in \mathcal{N}} x_{i,j}(t) = 1, \quad \forall v_i \in V, \forall t \in [1, T] \tag{2}$$

For the sake of convenience, denote by cloudlet $n_i(t)$ that hosts DT_i of supplier v_i at time slot $t \in [1, T]$, i.e., $x_{i,n_i(t)}(t) = 1$. Then, the DT synchronization cost of supplier v_i at time slot t is defined as

$$\mathcal{E}^{syn}_i(t) = \sum_{n_j \in \mathcal{N}} \zeta_j \cdot c_i \cdot x_{i,j}(t) + \sum_{n_j \in \mathcal{N}} \varkappa_j \cdot \Phi_i(t) \cdot x_{i,j}(t) + \sum_{e \in P_{g_{v_i}(t), n_i(t)}} \xi_e \cdot gd_i(t), \tag{3}$$

where $P_{g_{v_i}(t), n_i(t)}$ is the shortest path in G between cloudlets $g_{v_i}(t)$ and $n_i(t)$ in terms of the communication cost ξ_e of unit data transfer along each link $e \in P_{g_{v_i}(t), n_i(t)}$. It can be seen that $\mathcal{E}^{syn}_i(t)$ consists of the computing resource cost, storage resource cost, and communication resource cost.

Service request cost of each consumer: For each consumer $u_k \in U$ at time slot $t \in \mathbb{T}$, its requested information needs to be transmitted from the DT of supplier $r_{u_k} \in V$ to cloudlet $g_{u_k}(t)$ at which u_k is located, and such information transfer consumes communication resource. Assuming that the amount of data transmitted from the DT, DT_i, of supplier $v_i \in V$ to its consumer at time slot t is $\epsilon_i \cdot \Delta\Phi_i(t)$, where $\Delta\Phi_i(t)$ is the data generated by v_i in the past $t_0^{(i)}$ slots and ϵ_i is data compression rate. ϵ_i and $t_0^{(i)}$ are both constants that are determined by different applications with $0 < \epsilon_i < 1$ and $t_0^{(i)} \geq 1$.

Recall that $U(v_i, t)$ is the set of consumers requesting for DT_i service at time slot t. Assuming that DT_i is hosted by cloudlet $n_i(t)$, the total service cost of all consumers in $U(v_i, t)$ is calculated as follows.

Let $G_{v_i} = (\mathcal{N}, \mathcal{L}; w'^{(t)}_i(\cdot))$ be a graph derived from the MEC network $G(\mathcal{N}, \mathcal{L})$ for supplier v_i, and let $T^{opt}_{v_i,n_i(t)}$ be a Steiner tree in G_{v_i} rooted at cloudlet $n_i(t)$ hosting DT_i and spanning cloudlets at which consumers in $U(v_i, t)$ are located. The cost $w'^{(t)}_i(e)$ of edge $e \in \mathcal{L}$ in G_{v_i} is defined as

$$w'^{(t)}_i(e) = \xi_e \cdot \epsilon_i \cdot \Delta\Phi_i(t), \tag{4}$$

where ξ_e is the communication cost per unit data on edge $e \in \mathcal{L}$, and $\epsilon_i \cdot \Delta\Phi_i(t)$ is the amount of resulting data of v_i that needs to be transmitted from DT_i to the home cloudlets of consumers in $U(v_i, t)$.

The total service cost of all consumers in $U(v_i, t)$ requesting for DT_i service at time slot t is the cost sum of edges in Steiner tree $T^{opt}_{v_i,n_i(t)}$, i.e.,

$$\sum_{u_k \in U(v_i,t)} \mathcal{E}_{u_k}(t) = \sum_{e \in T^{opt}_{v_i,n_i(t)}} w'^{(t)}_i(e). \tag{5}$$

Due to NP-hardness of the Steiner tree problem, we instead find an approximate Steiner tree T_{v_i,n_j} for $T^{opt}_{v_i,n_j}$ in G_{v_i}. It is well known that $\sum_{e \in T_{v_i,n_j}} w'^{(t)}_i(e) \leq 2 \cdot OPT_{v_i}$ [26], where OPT_{v_i} is the optimal cost of the Steiner tree $T^{opt}_{v_i,n_j}$.

DT migration cost: Due to the movement of suppliers and consumers, minimizing the service cost of all consumers requires DTs to be placed nearby their consumers and suppliers. Therefore, the optimal DT placements at different time slots are different. For a given supplier v_i, if the location of its DT_i changes at time slot $t \in [1, T]$, i.e., $n_i(t-1) \neq n_i(t)$, then the accumulative historical update data $\Phi_i(t)$ in DT_i must be migrated from its previous location to the current location too, and the incurred migration cost is defined as follows.

Let $P_{n_j,n_{j'}}$ be a shortest path in G between cloudlets n_j and $n_{j'}$. The migration cost $\mathcal{E}^{mig}_i(t)$ of DT_i of supplier v_i at time slot t is

$$\mathcal{E}^{mig}_i(t) = \Phi_i(t) \sum_{n_j \in \mathcal{N}} \sum_{n_{j'} \in \mathcal{N} \setminus \{n_j\}} \sum_{e \in P_{n_j,n_{j'}}} \xi_e x_{i,j}(t-1) x_{i,j'}(t), \quad \forall t \in [2, T] \tag{6}$$

Only when $x_{i,j}(t-1) \cdot x_{i,j'}(t) = 1$ in (6), DT_i of supplier $v_i \in V$ can move from its location n at time slot $t-1$ to its current location j' at time slot t. The sum $\mathcal{E}(t)$ of the accumulative DT migration cost of all suppliers and the accumulative service cost of all consumers requesting for DT services at time slot t is

$$\begin{aligned}\mathcal{E}(t) &= \sum_{v_i \in V} \mathcal{E}^{syn}_i(t) + \sum_{v_i \in V} \mathcal{E}^{mig}_i(t) + \sum_{u_k \in U} \mathcal{E}_{u_k}(t) \\ &= \sum_{v_i \in V} \left(\mathcal{E}^{syn}_i(t) + \mathcal{E}^{mig}_i(t) + \sum_{u_k \in U(v_i,t)} \mathcal{E}_{u_k}(t) \right)\end{aligned} \tag{7}$$

The overall service cost $\mathcal{E}_T$ of DT migrations within a given time horizon $\mathbb{T}$ is $\sum_{t=1}^{T} \mathcal{E}(t)$.

C. Problem Definition

Given an MEC network $G = (\mathcal{N}, \mathcal{L})$ and a finite time horizon $\mathbb{T}$ that is divided into T equal time slots, a set V of suppliers and a set U of consumers, each supplier has a DT placed in a cloudlet for service provisioning, and each consumer requests for DT service from a specific supplier at each time slot. Assume that both suppliers and consumers are movable at different time slots but are stationary within each time slot. The *DT migration problem* is to determine whether the DT of each supplier needs to migrate to a new location at each time slot so that the total service cost $\sum_{t=1}^{T} \mathcal{E}(t)$ of all consumer requests for different DT services within the given time horizon $\mathbb{T}$ is minimized, subject to computing and storage capacities on each cloudlet.

D. Np-Hardness

In the following, we show that the defined problem is NP-hard.

Theorem 1: The DT migration problem in an MEC network $G = (\mathcal{N}, \mathcal{L})$ is NP-hard.

Proof: We prove the NP-hardness of the DT migration problem by a reduction from a well-known NP-hard problem - the minimum-cost generalized assignment problem (GAP) as fo.

Given $|\mathcal{N}|$ bins with capacity C_j on each bin j with $1 \leq j \leq |\mathcal{N}|$, there are $|V|$ items to be assigned to the bins, where if item v_i is assigned to bin j, it incurs a cost $cost_{i,j}$ with the amount c_j of computing resource consumed of bin j. *The minimum-cost GAP* is to assign as many items as possible to bins such that the total cost of assigned items is minimized, subject to the computing resource capacity on each bin [23].

We consider a special case of the DT migration problem, where both suppliers and consumers are stationary, the storage capacity on each cloudlet is unbounded, and the monitoring period consists of one time slot only. Under this assumption, we consider the DT placements of suppliers without DT migrations. We assume that each cloudlet (bin) n_j has computing capacity C_j. Let U denote the set of consumers requesting services from supplier $v_i \in V$ at the time slot. If DT_i (item) of object v_i is placed in cloudlet $n_j \in \mathcal{N}$, it consumes the amount c_i of computing resource of cloudlet j, and incurs a cost $cost_{i,j}$ ($= \mathcal{E}^{syn}_i + \sum_{u_k \in U(v_i)} \mathcal{E}_{u_k}$) by (3) and (5) that consists of the DT synchronization cost and the total service cost of all consumers in $U(v_i)$ requesting DT_i service, respectively. We aim to find an optimal DT placement schedule for all suppliers to minimize the total service cost $\sum_{t=1}^{T} \mathcal{E}(t)$, subject to the computing capacity on each cloudlet. It can be seen that this special DT migration problem is equivalent to the minimum-cost GAP. As the minimum-cost GAP is NP-hard, the DT migration problem is NP-hard, too. □

IV. ILP Formulation

In this section, we consider the offline version of the DT migration problem, assuming that the mobility profiles of suppliers and consumers at each time slot $t \in \mathbb{T}$ are given, for which we formulate an INP solution for the problem. We then transform the INP solution to an equivalent ILP solution.

We start the INP solution for the offline version of the DT migration problem as follows.

$$\text{Minimize} \quad \sum_{t=1}^{T} \mathcal{E}(t) \tag{8a}$$

$$\text{subject to} \quad (1)-(7) \tag{8b}$$

$$\sum_{v_i \in V} c_i \cdot x_{i,j}(t) \leq C_j, \quad \forall n_j \in \mathcal{N}, \forall t \in [1,T] \tag{8c}$$

$$\sum_{v_i \in V} \Phi_i(t) \cdot x_{i,j}(t) \leq \mathcal{M}_j, \quad \forall n_j \in \mathcal{N}, \forall t \in [1,T] \tag{8d}$$

$$x_{i,j}(t) \in \{0,1\}, \quad \forall v_i \in V, n_j \in \mathcal{N}, t \in [1,T] \tag{8e}$$

(8c) and (8d) ensure that the total consumption of computing and storage resources of any cloudlet cannot exceed its computing and storage capacities at each time slot.

The optimization problem in (8) is nonlinear due to that (6) is nonlinear. We now transform this INP solution into an equivalent ILP solution through linearizing (6), by introducing a new binary variable $y_{i,j,j'}(t)$, i.e.,

$$y_{i,j,j'}(t) \leq x_{i,j}(t-1), \quad \forall t \in [2,T], v_i \in V, n_j, n_{j'} \in \mathcal{N} \text{ with } j \neq j' \tag{9}$$

$$y_{i,j,j'}(t) \leq x_{i,j'}(t), \quad \forall t \in [2,T], v_i \in V, n_j, n_{j'} \in \mathcal{N} \text{ with } j \neq j' \tag{10}$$

$$y_{i,j,j'}(t) \geq x_{i,j}(t-1) + x_{i,j'}(t) - 1, \quad \forall t \in [2,T], v_i \in V, n_j, n_{j'} \in \mathcal{N} \text{ with } j \neq j' \tag{11}$$

(6) then is replaced by

$$\mathcal{E}_i^{mig}(t) = \Phi_i(t) \sum_{n_j \in \mathcal{N}} \sum_{n_{j'} \in \mathcal{N} \setminus \{n_j\}} \sum_{e \in P_{n_j,n_{j'}}} \xi_e \cdot y_{i,j,j'}(t) \tag{12}$$

An equivalent ILP formulation for the offline version of the DT migration problem is given as follows.

$$\text{Minimize} \quad \sum_{t=1}^{T} \mathcal{E}(t) \tag{13a}$$

$$\text{s.t.} \quad (1)\text{–}(5), (7), (8c), (8d), (8e), (9)\text{–}(12) \tag{13b}$$

$$y_{i,j,j'}(t) \in \{0,1\}, \quad \forall t \in [2,T], v \in V, n_j, n_{j'} \in \mathcal{N} \text{ with } j \neq j' \tag{13c}$$

It must be mentioned that the ILP solution (13) is an optimal solution to the offline version of the DT migration problem, and its value is a lower bound on the optimal solution of the DT migration problem.

V. Online Algorithm for the DT Migration Problem

In this section, we deal with the DT migration problem in G for a finite time horizon $\mathbb{T}$, assuming that the mobility profiles of both suppliers at each time slot are not known. To tackle the problem, we first formulate the problem as a Markov Decision Process (MDP), where there is an agent i for each supplier $v_i \in V$, and the set of agents learns how to act (policy) in successive decision-making through interacting with the environment. We then develop a Deep Reinforcement Learning (DRL) algorithm for the problem based on actor-critic networks, with the aim to maximize the accumulative reward for the given monitoring period. For the sake of convenience, the notations adopted in the DRL algorithm are summarized in Table II.

A. MDP Formalization

An MDP usually is expressed by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state set, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the transition function, and $\mathcal{R}$ is the reward function. Recall that the finite time horizon $\mathbb{T}$ consists of $|\mathbb{T}|$ equal time slots. Given state $s_t \in \mathcal{S}$ at time slot $t \in \mathbb{T}$, an action $\mathbf{a}^{(t)} \in \mathcal{A}$ with $\mathbf{a}^{(t)} = \langle a_1^{(t)}, a_2^{(t)}, \ldots, a_{|V|}^{(t)} \rangle$ that consists of DT migration decisions of $|V|$ suppliers in the system, and the next state $s_{t+1} \in \mathcal{S}$, $\mathcal{P}(s_t, \mathbf{a}^{(t)}, s_{t+1})$ is the probability of state transition from s_t to s_{t+1} through action $\mathbf{a}^{(t)}$, and $\mathcal{R}(s_t, \mathbf{a}^{(t)}, s_{t+1})$ is the amount of reward obtained when transiting states from s_t to s_{t+1} by action $\mathbf{a}^{(t)}$.

Specifically, the key components of an MDP are detailed as follows.

State: At each time slot $t \in [1,T]$, the running status of the MEC network consists of the locations of suppliers $\mathbf{g}(t) = (g_1^{(t)}, g_2^{(t)}, \ldots, g_{|V|}^{(t)})$, the volumes of update data of suppliers $\Phi(t) = (\Phi_1^{(t)}, \Phi_2^{(t)}, \ldots, \Phi_{|V|}^{(t)})$, the DT placements of suppliers at time slot $t-1$, $\mathbf{p}(t-1) = (p_1^{(t-1)}, p_2^{(t-1)}, \ldots, p_{|V|}^{(t-1)})$, the workloads of cloudlets $\mathbf{l}(t-1) = (l_1^{(t-1)}, l_2^{(t-1)}, \ldots, l_{|\mathcal{N}|}^{(t-1)})$ under the DT placement $\mathbf{p}(t-1)$ (which will be defined later), and the locations of consumers $\mathbf{b}(t) = (b_1^{(t)}, b_2^{(t)}, \ldots, b_{|U|}^{(t)})$, where $g_i^{(t)} \in [1, |\mathcal{N}|], i \in [1, |V|]$ is the cloudlet at which supplier v_i is located, $b_k^{(t)} \in [1, |\mathcal{N}|], k \in [1, |U|]$ represents the cloudlet at which consumer u_k is located, $p_i^{(t)} \in [1, |\mathcal{N}|], i \in [1, |V|]$ is the cloudlet hosting DT_i of supplier v_i, and $l_j^{(t)}$ indicates the resource utilization metric of cloudlet $j \in [1, |\mathcal{N}|]$. Specifically, $l_j^{(t-1)}$ of cloudlet n_j consists of two components, i.e., $l_j^{(t-1)} = (lc_j^{(t-1)}, ls_j^{(t-1)})$, where $lc_j^{(t-1)}$ and $ls_j^{(t-1)}$ represent the utilization ratios of computing and storage resources of cloudlet n_j corresponding to the DT placements $\mathbf{p}(t-1)$, respectively. Let state $s_t \in \mathcal{S}$ be the system state at time slot t, which is defined as follows.

$$s_t = (\mathbf{g}(t), \Phi(t), \mathbf{p}(t-1), \mathbf{l}(t-1), \mathbf{b}(t)) \tag{14}$$

Action: At each time slot t, each agent (supplier) decides whether to migrate its DT. An action of an agent is a placement choice at time slot $t \in T$, and a DT migration is activated when its placement at time slot t is different from its placement at time slot $t-1$. Specifically, the action of supplier v_i at time slot t is defined by a vector $a_i^{(t)} = (a_{i,1}^{(t)}, a_{i,2}^{(t)}, \ldots, a_{i,|\mathcal{N}|}^{(t)})$, where $a_{i,j}^{(t)} = 1$ indicates that DT_i of supplier v_i is placed in cloudlet n_j, otherwise $(a_{i,j}^{(t)} = 0)$, cloudlet n_j will not be selected to host DT_i, $\sum_{j=1}^{|\mathcal{N}|} a_{i,j}^{(t)} = 1$ for any $i \in V$. Then, action $\mathbf{a}^{(t)} \in \mathcal{A}$ is

TABLE II
TABLE OF NOTATIONS IN DRL ALGORITHM

Notation	Descriptions
$\mathbf{g}(t)$ and $\mathbf{b}(t)$	The location vectors of suppliers and consumers at time slot t
$\Phi(t)$	The stored data volumes of suppliers at time slot t
$\mathbf{p}(t)$	The DT placement of suppliers at time slot t
$\mathbf{l}(t)$	The computing and storage loads of cloudlets at time slot t
$\mathcal{S}$ and s_t	Set of system states and the state at time slot t, where $s_t \in \mathcal{S}$
$\mathcal{A}$ and $\mathbf{a}^{(t)}$	Set of actions and an action taken at time slot t, where $\mathbf{a}^{(t)} \in \mathcal{A}$
$\mathbf{w}_i$	Cost vector of supplier v_i of migrating DT_i to all potential cloudlets
$\mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1})$	The reward of choosing action $a_i^{(t)}$ for agent i if the state transits from s_t to s_{t+1}
$\mathcal{R}(s_t, \mathbf{a}^{(t)}, s_{t+1})$	The reward of choosing action $\mathbf{a}^{(t)}$ for all agents if the state transits from s_t to s_{t+1}
γ	The reward discount factor
$\mathcal{R}$	The expected cumulative discounted reward for the given monitoring period
$\pi_i(a_i \mid s_t ; \theta_i)$	The policy function of agent i, where θ_i is the DNN parameters of actor network of agent i
$V^{\pi_i}(s_t, \omega_i)$	The state-value function of agent i, where ω_i is the DNN parameter of the critic network of agent i

defined as

$$\mathbf{a}^{(t)} = \left(a_1^{(t)}, a_2^{(t)}, \ldots, a_{|V|}^{(t)}\right) \tag{15}$$

State Transition: Having taken action $\mathbf{a}^{(t)}$, the system state transits from the current state s_t to the next state s_{t+1} by the state transition function $P(s_t, \mathbf{a}^{(t)}, s_{t+1})$, where $P(s_t, \mathbf{a}^{(t)}, s_{t+1})$ denotes the probability of reaching state s_{t+1} at time slot $t+1$, provided that the agents take action $\mathbf{a}^{(t)}$ in state s_t at time slot t. DT migrations are conducted for agents whose DT locations at time slot t are different from their DT locations at time slot $t-1$. Vectors $\mathbf{p}(t)$ and $\mathbf{l}(t)$ then are updated accordingly. As suppliers and consumers can move around within the MEC network, both $\mathbf{g}(t)$ and $\mathbf{b}(t)$ are updated based on their movements at time slot t. $\Phi(t)$ is updated by calculating the amount of data of each supplier by (1).

Workload among cloudlets: It is noted that the workload on each cloudlet determines whether a DT migration toward the cloudlet is successful or not, as an overloaded cloudlet is unable to accommodate those DTs that will migrate to it. On the other hand, a cloudlet below the average workload can host more migrated DTs. To reduce the service cost while ensuring most DT migrations to be successful, workloads among different cloudlets need to be balanced. Recall lc_j^t and ls_j^t are the utilization ratios of computing and storage resources in cloudlet $n_j \in \mathcal{N}$ at time slot t, respectively, which are defined as follows.

$$lc_j^t = \frac{\sum_{i=1}^{|V|} a_{i,j}^t \cdot c_i}{C_j}, \tag{16}$$

$$ls_j^t = \frac{\sum_{i=1}^{|V|} a_{i,j}^t \cdot \Phi_i(t)}{\mathcal{M}_j}. \tag{17}$$

Reward function: To maximize the accumulative reward of all agents, a reward function $\mathcal{R}(s_t, \mathbf{a}^{(t)}, s_{t+1})$ is defined as the total amount of rewards received by all agents after the state transition from s_t to s_{t+1}. The system then evaluates the actions of all agents by utilizing the reward function, where the value of the reward function is inversely proportional to the total cost of selecting a cloudlet for its DT migration of each supplier and the degree of workload unbalancing among cloudlets.

For each supplier $v_i \in V$, different migration destinations will incur different costs. Let $w_{i,j}^{(t)}$ denote the cost of supplier v_i migrating DT_i from its current location to cloudlet n_j at time slot t. The DT synchronization cost $cost_{syn}^{(t)}(v_i, n_j)$ of supplier v_i synchronizing its DT in cloudlet n_j is

$$cost_{syn}^{(t)}(v_i, n_j) = \zeta_j \cdot c_i + \varkappa_{n_j} \cdot \Phi_i(t) + \sum_{e \in P_{g_i^{(t)}, n_j}} \xi_e \cdot gd_i(t), \tag{18}$$

where $P_{g_i^{(t)}, n_j}$ is the shortest routing path in the MEC network G between cloudlets $g_i^{(t)}$ and n_j in terms of the communication cost ξ_e of each link $e \in \mathcal{L}$, and cloudlet $g_i^{(t)}$ is the cloudlet at which supplier v_i is located at time slot t. Recall that c_i is the amount of computing resource for hosting DT_i and ζ_j is the cost per unit of computing resource of cloudlet n_j. The three items in the right side of (18) are the costs of computing resource and storage resource consumptions, and the transmission cost of the DT data transfer of supplier v_i, respectively. Recall that $U(v_i, t)$ is the set of consumers requesting for DT_i services of supplier v_i. The service cost of consumers in $U(v_i, t)$ requesting services from DT_i can be calculated through finding an approximate Steiner tree T_{v_i, n_j} in a graph $G_{v_i} = (\mathcal{N}, \mathcal{L}; w'^{(t)}_i(\cdot))$ rooted at cloudlet n_j and spanning the cloudlets at which consumers in $U(v_i, t)$ are located. G_{v_i} in fact is a copy of the MEC network $G(\mathcal{N}, \mathcal{L})$ for supplier v_i, and the cost $w'^{(t)}_i(e)$ of each edge $e \in \mathcal{L}$ is defined in (4). The service cost $cost_{\mathrm{c}}^{(t)}(v_i, n_j)$ of all consumers in $U(v_i, t)$ requesting DT_i services is the cost sum of the edges in approximate Steiner tree T_{v_i, n_j}, i.e., $cost_{\mathrm{c}}^{(t)}(v_i, n_j) = \sum_{e \in T_{v_i, n_j}} w'^{(t)}_i(e)$. The communication cost $cost_{mig}^{(t)}(v_i, n_j)$ of migrating the DT of supplier v_i from its current cloudlet $p_i^{(t-1)}$ to the next cloudlet n_j is $cost_{mig}^{(t)}(v_i, n_j) = \Phi_i(t) \sum_{e \in P_{p_i^{(t-1)}, n_j}} \xi_e$, where $P_{p_i^{(t-1)}, n_j}$ is a shortest routing path in G between cloudlets $p_i^{(t-1)}$ and n_j. The total cost $w_{i,j}^{(t)}$ of DT_i migration from its current cloudlet $p^{(t-1)}$ to cloudlet n_j at time slot t is $w_{i,j}^{(t)} = cost_{syn}^{(t)}(v_i, n_j) + cost_{\mathrm{c}}^{(t)}(v_i, n_j) +$

$cost_{mig}^{(t)}(v_i, n_j)$. The cost vector $\mathbf{w}_i^{(t)}$ of migrating DT_i to all potential cloudlets at time slot t is represented by

$$\mathbf{w}_i^{(t)} = \left(w_{i,1}^{(t)}, w_{i,2}^{(t)}, \ldots, w_{i,|\mathcal{N}|}^{(t)}\right) \tag{19}$$

By incorporating the workload balancing among cloudlets into the reward function, the reward received by agent i (supplier v_i) taking action $a_i^{(t)}$ is

$$\mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1}) = -a_i^{(t)} \cdot (\mathbf{w}_i^{(t)})^{\text{tr}} - \beta_1 \frac{\sum_{j=1}^{|\mathcal{N}|}(lc_j^{(t)} - \kappa_1)}{|V|} - \beta_2 \frac{\sum_{j=1}^{|\mathcal{N}|}(ls_j^{(t)} - \kappa_2)}{|V|}, \tag{20}$$

where $(\mathbf{w}_i^{(t)})^{\text{tr}}$ is the transpose of vector $\mathbf{w}_i^{(t)}$, β_1 and β_2 are the scaling coefficients between the service cost and resource utilization ratios, respectively. The first item in the right hand side of (20) is the service cost of various resources consumed of supplier v_i, the second and third items are the computing and storage workload utilization ratios of cloudlets, and κ_1 and κ_2 are the workload utilization ratio thresholds, implying how much overloads of computing resource and storage resource from their average workload utilization ratios. The rationale behind the settings of κ_1 and κ_2 is to strive for workload balancing among cloudlets.

Notice that the cost sum of DT updatings and migrations of suppliers and the total service cost $\mathcal{E}(t)$ of consumers requesting DT services in (7) can be rewritten as

$$\mathcal{E}(t) = \sum_{v_i \in V} \mathcal{E}_i^{syn}(t) + \sum_{v_i \in V} \mathcal{E}_i^{mig}(t) + \sum_{v_i \in V} \sum_{u_k \in U(v_i,t)} \mathcal{E}_{u_k}(t) = \sum_{i=1}^{|V|} a_i^{(t)} \cdot (\mathbf{w}_i^{(t)})^{\text{tr}} \tag{21}$$

The accumulative reward obtained by action $\mathbf{a}^{(t)}$ at time slot t is the sum of the expected rewards of all agents, which is defined as follows.

$$\mathcal{R}(s_t, \mathbf{a}^{(t)}, s_{t+1}) = \sum_{v_i \in V} \mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1}). \tag{22}$$

The MDP for the DT migration problem is to maximize the expected cumulative discounted reward $\mathcal{R}$ for the given time horizon $\mathbb{T}$, which is defined as follows.

$$\mathcal{R} = \sum_{t=1}^{T} \gamma^{t-1} \mathcal{R}(s_t, \mathbf{a}^{(t)}, s_{t+1}), \tag{23}$$

where $\gamma \in (0, 1]$ is the reward discount factor.

B. Actor-Critic Networks

Due to large state and action spaces and the lack of future mobility information of suppliers and consumers, it is impossible to calculate the state transition function $P(s_t, \mathbf{a}^{(t)}, s_{t+1})$ in a reasonable amount of time. We instead design an actor-critic network based DRL algorithm to search an optimal policy for DT migrations. Specifically, for each agent i in the proposed algorithm, there are two neural networks: an actor network and a critic network, where the actor network explores an optimal policy $\pi_{\theta_i}(a_i \,|s_t)$ that provides the possibility of taking action a_i at state s_t for the MDP through Deep Neural Networks (DNNs), θ_i is the policy parameter of agent i. For policy $\pi_{\theta_i}(a_i|s_t)$, a state-value function $V^{\pi_i}(s_t)$ is used to predict the expected total discounted reward when agent i is at state s_t. On the other hand, the critic network is used to evaluate the policy $\pi_{\theta_i}(a_i \,|s_t)$ of the actor network, with an approximate state value function $V^{\pi_i}(s_t)$. Both the actor and critic networks for the MDP are detailed as follows.

The critic network evaluates the state-value function under the actor policy. For each agent i at time slot t, its critic network takes state s_t as input, and the output is an estimate on the state-value function $V^{\pi_i}(s_t, \omega_i^{(t)})$, where $\omega_i^{(t)}$ is the parameter of the critic network of agent i at time slot t. The critic network of agent i is trained based on the Temporal Difference (TD) learning [24], and the TD error $\delta_i^{(t)}$ is calculated as follows.

$$\delta_i^{(t)} = \mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1}) + \gamma V(s_{t+1}, \omega_i^{(t)}) - V(s_t, \omega_i^{(t)}), \tag{24}$$

where $\mathcal{R}_i(s_t, a_i, s_{t+1}) + \gamma V(s_{t+1}, \omega_i^{(t)})$ is the true cumulative reward and $V(s_t|\omega_i^{(t)})$ is the predicted cumulative reward of agent i at state s_t. The parameter of the critic network is trained by

$$\omega_i^{(t+1)} = \omega_i^{(t)} + \eta_1 {\delta_i^{(t)}}^2 \nabla_{\omega_i} V(s_t, \omega_i^{(t)}), \tag{25}$$

where η_1 is the learning rate of the critic network of agent i with $1 \le i \le |V|$.

The actor network of each agent aims to provide an optimal DT placement choice at each time slot for the agent. For agent i at time slot t, the input of its actor network is the system state s_t, and its output is an action that is determined by policy $\pi_{\theta_i^{(t)}}(a_i|s_t)$, where $\pi_{\theta_i^{(t)}}(a_i|s_t)$ is the probability of taking action $a_i^{(t)}$ by agent i at time slot t. The actor network of agent i adopts the gradient descent policy to update parameter θ_i, by taking steps in the direction of

$$\nabla_{\theta_i^{(t)}} J(\theta_i^{(t)}) \approx \mathbb{E}_{\pi_{\theta_i^{(t)}}}[\nabla_{\theta_i^{(t)}} \log \pi_{\theta_i^{(t)}}(s_t, a_i^{(t)}) \delta_i^{(t)}] \tag{26}$$

Once the actor network of agent i is trained, its parameter is updated accordingly, i.e.,

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \eta_2 \nabla_{\theta_i^{(t)}} \log \pi_{\theta_i^{(t)}}(s_t, a_i^{(t)}) \delta_i^{(t)}, \tag{27}$$

where η_2 is the learning rate of the actor network of agent i with $1 \le i \le |V|$.

The training algorithm for the actor-critic networks proceeds as follows.

We assume that a collection of sets of historical mobility and service request information for both suppliers and consumers is given in advance. We train the parameters of these networks to approximate the optimal policy of the actor network and the state-value function of the critic network with the help of historical information. The training process of the actor-critic networks is illustrated in Fig. 2. Assume that s_t is the current state of the agent network actor i at time slot t, then the output

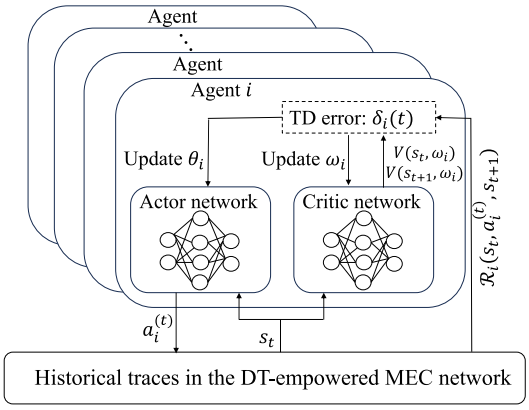


Fig. 2. Training process of the actor-critic networks.

policy $\pi_{\theta_i^{(t)}}$ chooses an action $a_i^{(t)}$ of agent i. By action, DT migration is performed and the state of the system transits from state s_t to state s_{t+1}. The amount of reward obtained by the agent i is then $\mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1})$. Notice that the state of the system s_t, the action $a_i^{(t)}$ of agent i, the reward $\mathcal{R}_i(s_t, a_i^{(t)}, s_{t+1})$ of agent i, and the next system state s_{t+1} are stored in a buffer for later replay.

We then update both parameters θ_i and ω_i for the actor and critic networks of agent i. Specifically, for each agent i (supplier $v_i \in V$), we first calculate the TD error $\delta_i^{(t)}$ by (24), using the recorded state s_t and the next state s_{t+1}. We then update parameter $\omega_i^{(t)}$ in (25), followed by updating parameter $\theta_i^{(t)}$ in (27). If both actor and critic networks of agent i do not converge after training using the selected historical data, then another set of historical data from the collection of datasets is chosen for their training. This procedure continues until the convergence of both θ_i and ω_i of agent i for all i with $1 \leq i \leq |V|$.

The algorithm for the training of both actor and critic networks of each agent is given in Algorithm 1.

Algorithm 1: The Training Algorithm for Actor-Critic Networks.

Input: An MEC network $G = (\mathcal{N}, \mathcal{L})$, historical mobility and service request information of suppliers and users in the monitored area, a given discount factor γ, learning rate η_1 of a critic network, and learning rate η_2 of an actor network.

Output: The trained actor and critic network parameters θ_i and ω_i for each supplier $v_i \in V$.

Initialize parameters θ_i of the actor network and ω_i of the critic network randomly of agent i;
while θ_i and ω_i have not converged **do**
  Select a set of historical mobility and service request information of suppliers and consumers;
  Initialize the DT placement for suppliers in the MEC network and obtain the initial state;
  for $t \leftarrow 1$ to T **do**
    for $i \leftarrow 1$ to $|V|$ **do**
      Obtain the cost vector $\mathbf{w}_i$ of supplier v_i;
      Obtain action $a_i^{(t)}$ at state s_t, using the output policy of the actor network of agent i;
      Calculate the reward $\mathcal{R}_i$ of agent i at time slot t;
    end for
    Obtain the next state s_{t+1} based on the actions of all agents;
    for $i \leftarrow 1$ to $|V|$ **do**
      Input states s_t and s_{t+1} into the critic network, and obtain $V(s_t, \omega_i)$ and $V(s_{t+1}, \omega_i)$ of agent i;
      Calculate the TD error $\delta_i^{(t)}$ using (24);
      Update parameter ω_i of the critic network of agent i by (25);
      Update parameter θ_i of the actor network of agent i by (27);
    end for
  end for
end while
return the trained actor and critic networks of agent i with $1 \leq i \leq |V|$.

C. DRL Algorithm

Having trained both actor and critic networks of all agents by Algorithm 1, we now deal with the DT migration problem, by proposing a DRL algorithm. Specifically, the algorithm starts by initializing state s_1 and setting the value of the total cost $\mathcal{E}_T$.

For each time slot $t \in \mathbb{T}$, an action $\mathbf{a}^{(t)}$ is chosen by the policy function π_{θ_i} of each agent i at state s_t, and the value of $a_{i,j}^{(t)}$ of each cloudlet $n_j \in \mathcal{N}$ for each supplier $v_i \in V$ in $\mathbf{a}^{(t)}$ is examined. That is, if $a_{ij}^{(t)} = 1$, then check whether $n_j = p_i^{(t-1)}$ and cloudlet n_j has sufficient resource for the placement of DT_i. If yes, DT_i is migrated from cloudlet $p_i^{(t-1)}$ to cloudlet n_j at time slot t; otherwise (cloudlet n_j does not have sufficient resource for DT_i migration), a cloudlet $n_{j'}$ with the smallest $w_{i,j'}^{(t)}$ in the vector $\mathbf{w}_i^{(t)}$ of (19) is chosen. The resource utilizations of all involving cloudlets for DT_i migration, the location $p_i^{(t)}$ of DT_i at time slot t, and the total migration cost of DT_i is updated accordingly.

The detailed DRL algorithm is given in Algorithm 2.

D. Algorithm Analysis

Theorem 2: Given an MEC network $G = (\mathcal{N}, \mathcal{L})$, a finite time horizon $\mathbb{T}$, a set V of mobile suppliers and a set U of mobile consumers, each supplier has a DT placed in a cloudlet for service provisioning, and each consumer requests DT data from a specific supplier at each time slot. There is an efficient algorithm Algorithm 2 for the DT migration problem that takes $O(|V| \cdot |\mathcal{N}|^3)$ time per time slot $t \in \mathbb{T}$.

Proof: We first show that the solution delivered by Algorithm 2 is feasible. At each time slot $t \in \mathbb{T}$, the algorithm chooses a DT migration destination with sufficient resource for the DT of a supplier, the storage and computing capacities on the DT destination cloudlet will not be violated in the solution delivered by the algorithm.

Algorithm 2: The DRL Algorithm for the DT Migration Problem.

Input: An MEC network $G = (\mathcal{N}, \mathcal{L})$ with each cloudlet $n_j \in \mathcal{N}$ having storage capacity $\mathcal{M}_j$ and computing capacity C_j, a set of suppliers V, a set of consumers U, the trained actor-critic networks, and a finite time horizon $\mathbb{T}$.

Output: The cumulative service cost for the finite time horizon $\mathbb{T}$, and DT migration decisions at each time slot $t \in \mathbb{T}$.

Initialize DTs of suppliers in cloudlets and set state at s_1;
$\mathcal{E}_T \leftarrow 0$; $P \leftarrow \emptyset$; /* the migration cost and the migration solution */
for $t \leftarrow 1$ to T **do**
  for each supplier $v_i \in V$ **do**
    Obtain the cost vector $\mathbf{w}_i$ of supplier v_i;
    Obtain an action $a_i^{(t)}$ at the current state s_t by the actor policy function $\pi_i(s_t|\theta_i)$;
    for each cloudlet $n_j \in \mathcal{N}$ **do**
      if $a_{i,j}^{(t)} = 1$ and cloudlet $n_j \neq p_i^{(t-1)}$ with sufficient resource **then**
        $p_i^{(t)} \leftarrow n_j$; /* migrate DT_i from $p_i^{(t-1)}$ to n_j */;
        Update resources in cloudlets n_j and $p_i^{(t-1)}$;
        $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij}^{(t)}$;
      else if $a_{i,j}^{(t)} = 1$ and $n_j \neq p_i^{(t-1)}$ and cloudlet n_j does not have adequate resource **then**
        $p_i^{(t)} \leftarrow n_{j'}$; cloudlet $n_{j'}$ with the smallest $w_{i,j'}^{(t)}$ in vector $\mathbf{w}_i^{(t)}$;
        Update the resource utilization of cloudlets $p_i^{(t-1)}$ and $n_{j'}$;
        $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij'}^{(t)}$;
      else if $a_{i,j}^{(t)} = 1$ and $n_j = p_i^{(t-1)}$ **then**
        Update the resource utilization of cloudlet n_j;
        $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij}^{(t)}$;
      end if
    end for
    $P \leftarrow P \cup \{p_i^{(t)} \mid p_i^{(t)} \neq p_i^{(t-1)}\}$;
  end for
end for
return P with cost $\mathcal{E}_T$.

We then analyze the time complexity of Algorithm 2 as follows. The training of actor-critic networks makes use of historical data of suppliers, Algorithm 1 runs in an offline manner, the model training time thus is not critical.

The rest is to analyze the time complexity of Algorithm 2. Algorithm 2 utilizes the trained actor network of each agent i to perform the DT migration decision. At each time slot $t \in \mathbb{T}$, the algorithm takes $O(|\mathcal{N}|^3)$ time for each supplier i to find an approximate Steiner tree rooted at each potential location $n_j \in \mathcal{N}$ of its DT migration destination. As the actor network has been trained successfully, the output by the actor network involves only matrix manipulations in the neural network, which is tractable. Then, each supplier $v_i \in V$ takes $O(|\mathcal{N}|)$ time to update its DT placement and the system resource utilization, based on the output action. The time complexity of Algorithm 2 thus is $O(|V| \cdot |\mathcal{N}|^3)$ per time slot.

In practice, for a large-scale MEC network, to mitigate the high time complexity $O(|V| \cdot |\mathcal{N}|^3)$, there is a balancing between the solution accuracy and the time spent for finding the solution, a shortest path tree instead of a Steiner tree can be used to deliver a sub-optimal solution, and the shortest path tree can be found by the Dijkstra algorithm with time complexity of $O(|V| \cdot |\mathcal{L}| \cdot \log |\mathcal{N}|)$. However, almost every MEC network G is a sparse, planar graph, implying $|\mathcal{L}| = O(|\mathcal{N}|)$. Then, the Dijkstra algorithm only takes $O(|V| \cdot |\mathcal{N}| \cdot \log |\mathcal{N}|)$ time per time slot. □

VI. Performance Evaluation

In this section, we evaluated the performance of the proposed algorithms for the DT migration problem. We also investigated the impact of important parameters on the performance of the proposed algorithms.

A. Experimental Settings

We considered an MEC network $G = (\mathcal{N}, \mathcal{L})$ that consists of 30 APs evenly distributed in a 23 km × 69 km geographical area. The topology of G is generated via NetworkX [20]. Each AP covers a grid with 4.6 km × 11.5 km. Associated with each AP $j \in \mathcal{N}$, there is a co-located cloudlet $n_j \in \mathcal{N}$ with computing capacity C_j and storage capacity $\mathcal{M}_j$, where the values of C_j and $\mathcal{M}_j$ are randomly drawn in the range of [30, 50] GHz and [200, 300] GB, respectively [22]. Given each cloudlet $n_j \in \mathcal{N}$, the cost ζ_j of computing resource consumption per GHz is randomly drawn in $[0.4, 0.8] [12], and the cost $\varkappa_j$ of per unit storage resource varies from $0.01 to $0.014, respectively. The communication cost ξ_e per GB in each link $e \in \mathcal{L}$ is randomly chosen from $0.05 to $0.12 [28]. To balance the computing and storage workloads among cloudlets, both workload thresholds κ_1 and κ_2 are set at 0.6, and coefficients β_1 and β_2 are set at 10in the default setting.

We considered an application scenario, where there are 20 suppliers and 100 consumers. We evaluated the performance of the proposed DRL algorithm, using the real-world taxi mobility trace in Rome, Italy [1], where taxis are service suppliers. The Rome taxi dataset contains mobility trajectories of 320 taxis over 30 days. The mobility trajectories of the chosen taxis during the monitoring period were used as historical mobility traces of the suppliers. The consumers are randomly deployed under the coverage of different APs at different time slots, and their initial locations and movement directions are randomly generated. The amount c_i of computing resource demanded by supplier $v_i \in V$ varies from 3 GHz to 10 GHz respectively [12]. The amount $gd_i(t)$ of update data generated by supplier $v_i \in V$ at time slot t is randomly drawn in [1, 5] GB. The compression rate ϵ_i of the processed update data at DT_i of supplier v_i is randomly selected in the range of [0.10, 0.25]. The value $t_0^{(i)}$ of calculating $\Delta\Phi_i(t)$ for supplier v_i is randomly drawn in the range of [1, 5]. The vehicle location in the dataset is sampled every three minutes for in total 100 sampling, with a time horizon consisting

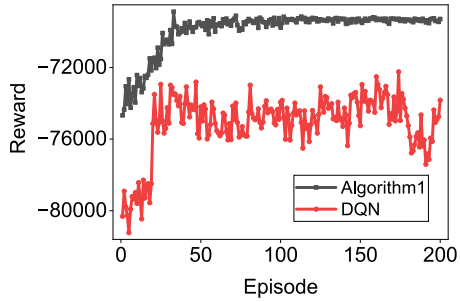


Fig. 3. Learning performance of Algorithm 1 and the DQN-based method.

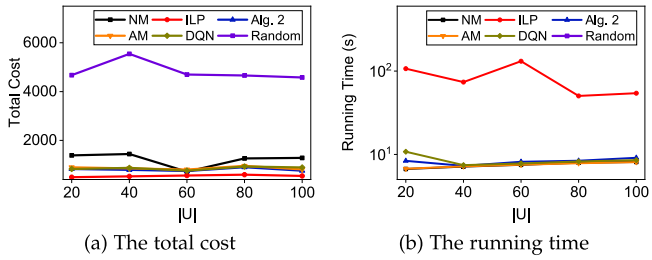


Fig. 4. Performance of Algorithm 2, the ILP, AM, NM, the DQN-based method, and Random, by varying the number $|U|$ of consumers.

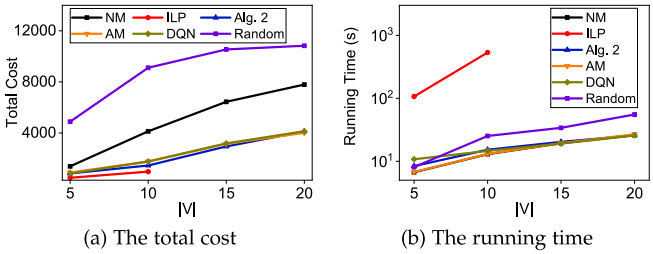


Fig. 5. Performance of Algorithm 2, the ILP, AM, NM, the DQN-based method, and Random, by varying the number $|V|$ of suppliers.

of $T = 100$ time slots [12]. Both suppliers and consumers are movable within the network at different time slots, but keep stationary at each time slot.

We evaluated the proposed DRL algorithm, Algorithm 2, against four baselines for the DT migration problem, which are described as follows.

- ILP: The ILP in (13) is an offline version of the problem, assuming that the mobility profiles of each supplier and each consumer at each time slot are given. The value of the ILP is a lower bound on the optimal solution to the DT migration problem.
- NoMigration: The DT location of each supplier does not change since its initial deployment for the entire time horizon.
- AlwaysMigration: The DT of each supplier is allowed to migrate from its current location to another location at different time slots.
- DQN-based method: The Deep Q Network (DQN) is a value-based DRL method, which combines the value function approximation and experience replay techniques for model training.
- Random: At each time slot, the algorithm chooses random numbers of suppliers and migrates the DT locations of chosen suppliers to new destinations that are also randomly chosen.

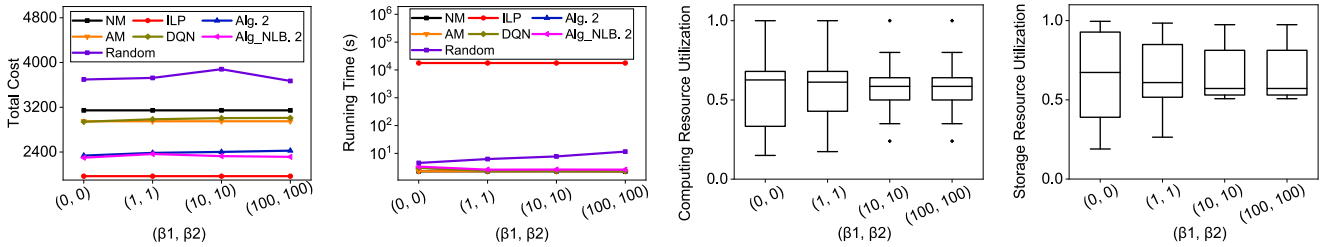
The ILP in (13) is solved by IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.7.1.0 [5]. The value in each figure is the mean of the results out of 20 network instances with the same network size. The actual running time of each algorithm is obtained on a desktop equipped with an Intel(R) Xeon(R) Platinum 8280 2.70 GHz CPU, with 10 GB RAM. These parameters are adopted in the default setting unless otherwise specified.

B. *Performance of Different Algorithms for the DT Migration Problem*

We first evaluated Algorithm 1 for the training of actor-critic networks. Fig. 3 depicts the learning performance curves of Algorithm 1 and the DQN-based method in terms of the cumulative reward. It can be seen from Fig. 3 that the reward received by Algorithm 1 fluctuates greatly in the early training stage. With the advance of the training process, the cumulative reward increases, implying that the total cost of DT-assisted service provisioning decreases. Having trained around 200 episodes, the reward value stabilizes and the actor-critic networks ultimately converge. In contrast, the change of the reward by the DQN-based method is relatively small and the network slowly converges. This is because the DQN-based method makes its decision based on a Q-function only, resulting in a large bias, while the value update by Algorithm 1 is based on both the policy and the value function. Additionally, the DQN-based method relies on experience replays for training, which is effective only when there is sufficient accumulated data, thus slowing down the convergence speed. Unless otherwise specified, in the rest of this paper, we assume that the number of training episodes is 200.

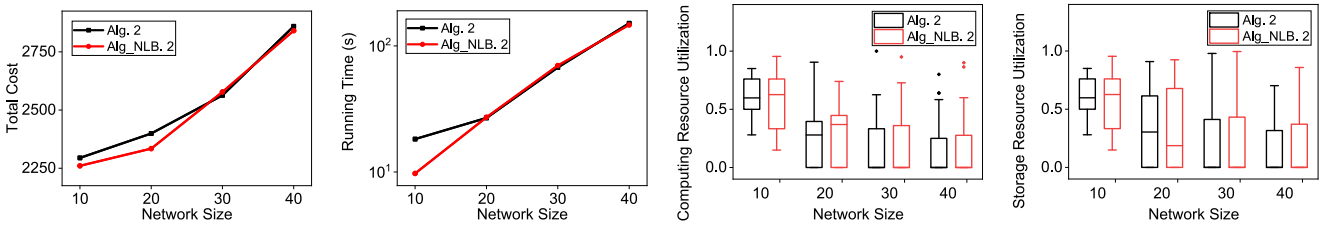
Having the actor-and-critic networks of each agent been trained, we studied the performance of different algorithms for the DT migration problem. We refer to Algorithm 2, the ILP, NoMigration, AlwaysMigration, and DQN-based method as Algorithm 2, ILP, NM, AM and DQN, respectively.

We investigated Algorithm 2 against the comparison algorithms, by varying the number of consumers while fixing the number of suppliers at 5. It can be seen from Fig. 4 that the ILP has the least total cost while the solution by algorithm Random has the largest cost. On the other hand, Fig. 5 plots the performance curves of different algorithms, by varying the number of suppliers while fixing the number of consumers at 100. With the increase on the number of suppliers, it is observed from Fig. 5(a) that the cost of the solution by each comparison algorithm increases due to the fact that synchronizations between suppliers and their DTs consume more resources. It is noted from Fig. 5 that although the ILP delivers an optimal solution when the problem size is small; otherwise its running time is prohibitively large.



(a) The total cost (b) The running time (c) Computing resource utilization by Alg 2 (d) Storage resource utilization by Alg 2

Fig. 6. Impact of workload balancing coefficients β_1 and β_2 on the performance of different algorithms.



(a) The total cost (b) The running time (c) Computing resource utilization (d) Storage resource utilization

Fig. 7. Impact of network size $|\mathcal{N}|$ on the performance of Algorithm 2 and Alg_NLB 2.

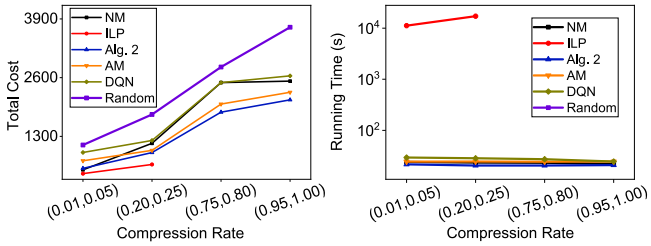
C. Impact of Parameters on the Performance of the Online Algorithm

We then studied impacts of important parameters on the performance of Algorithm 2 including workload balancing coefficients β_1 and β_2, network size $|\mathcal{N}|$, and data compression rate ϵ_i at each device $v_i \in V$, by varying the value of each parameter while fixing the number of suppliers at 20 and the number of consumers at 100, respectively.

Workload balance coefficients: We evaluated the impact of workload balancing coefficients β_1 and β_2 on the performance of Algorithm 2, in terms of service cost and resource utilization while fixing the network size at 10. We refer to Algorithm 2 without considering workload balancing as Alg_NLB 2. Fig. 6 illustrates the performance curves of the algorithms. It can be seen from Fig. 6(c) and (d) that when the values of β_1 and β_2 are zeros (which means that Algorithm 2 does not take into account the workload balance), the computing and storage resource utilization ratios on cloudlets heavily fluctuate. When the values of β_1 and β_2 become larger, the workload balance impacts the agent reward in (20) significantly, and the workloads among cloudlets are well balanced as shown in Fig. 6 and (d), respectively. Meanwhile, with the increase on the values of β_1 and β_2, it is observed from Fig. 6(a) that the costs of the solutions by Algorithm 2 and DQN-based method increase while the costs by the other mentioned methods do not. The rationale behind this is that suppliers may not migrate their DTs to the new locations with the lowest costs while Algorithm 2 does include workload balancing among cloudlets into consideration.

Network size: We investigated the impact of network size $|\mathcal{N}|$ on the performance of different algorithms. Fig. 7 plots the performance curves of Algorithm 2 and Alg_NLB 2. It can be observed from Fig. 7(a) that Alg_NLB 2 outperforms Algorithm 2 when the network size is no greater than 20. This is because when workload balancing is considered in the reward function, a supplier is very likely not to choose the cloudlet with the lowest cost as its DT migration destination. With the increase on the network size, the total service costs and running times of both comparison algorithms increase, due to longer DT data transfer paths from supplier locations to their DT locations in a large network. Migrating DTs between different cloudlets, and DT data transmission between DT locations and the cloudlets processing consumer requests consume much more network resources. Also, a large network usually contains more cloudlets, and thus it takes more time to find a potential DT migration destination for each supplier, as shown in Fig. 7(b). It can be seen from Fig. 7(c) and (d) that the median and the 75th percentile resource utilization of both comparison algorithms decrease with the increase on the network size, since a large network has more computing and storage resources for DT placements. For a given network size, the resource utilizations of cloudlets are within a narrower range in the solution delivered by Algorithm 2 compared to the one delivered by Alg_NLB 2. This indicates that the solution by Algorithm 2 has a better workload balancing among cloudlets.

Data compression rate: We considered the impact of the data compression rate on the performance of Algorithm 2. Fig. 8 plots the performance curves of different algorithms by varying the data compression rate in four different ranges of $[0.01, 0.05]$, $[0.20, 0.25]$, $[0.75, 0.80]$, and $[0.95, 1.00]$, respectively. It can be seen from Fig. 8 that the solution by Algorithm 2 has the least cost among the comparison algorithms. With the increase on the



(a) The total cost (b) The running time

Fig. 8. Impact of the data compression rate ϵ_i of each supplier $v_i \in V$ on the performance of different algorithms.

range of the data compression rate, the cost by each comparison algorithm increases too, because a smaller data compression rate implies that a large amount of DT data is transmitted between DTs of suppliers and consumers, and more communication resources are consumed.

VII. Conclusions and Future Work

In this article, we studied the cost-aware DT migration problem for service provisioning in a DT-empowered MEC network, with the aim to minimize the total service cost. We first showed that the problem is NP-hard, and formulated an INP solution and transformed the solution to an equivalent ILP solution to the offline version of the problem, assuming that the mobility profiles of both suppliers and consumers are given in advance. We then developed a novel DRL algorithm for the DT migration problem, by considering the system dynamics and heterogeneity of resource usages, the past mobility traces of suppliers and consumers, and changing consumer service request patterns over time. We finally evaluated the performance of the proposed DRL algorithm against other baselines through simulations. Simulation results demonstrate that the proposed algorithm is promising.

Built upon this work, we will focus on the following potential topics in future. Firstly, we will explore parallelism for the implementation of the proposed DRL algorithm in order to reduce its running time further, considering that modern computers consist of not only CPUs but also GPUs and other AI model training accelerators. Secondly, in an insecure edge computing environment, providing secure inference services is of paramount importance, we will take data security into consideration when dealing with DT migrations, study different encryption approaches for to-be-migrated DT data encryption, and analyze the encryption overhead. Finally, since most (IoT) portable mobile devices have limited energy capacity, we will count the energy consumption on cloudlets, APs, and mobile devices as part of the service cost, and investigate the impact of energy consumption on the performance of the proposed DRL algorithm too.

Acknowledgment

The authors appreciate for the four anonymous referees and the Associate Editor for their constructive comments and invaluable suggestions, which help us to improve the quality and presentation of the paper greatly.

References

[1] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "Crawdad roma/taxi," 2024. [Online]. Available: https://ieee-dataport.org/open-access/crawdad-romataxi
[2] Z. Chen, W. Yi, A. Nallanathan, and J. Chambers, "Distributed digital twin migration in multi-tier computing systems," *IEEE J. Sel. Topics Signal Process.*, vol. 18, no. 1, pp. 109–123, Jan. 2024.
[3] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1298–1311, Apr. 2021.
[4] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing throughput of delay-sensitive NFV-enabled request admissions via virtualized network function placement," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1535–1548, Oct.-Dec., 2021.
[5] "IBM ILOG CPLEX optimization studio," Accessed: Jul. 2023. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio
[6] J. Li et al., "Digital twin-assisted, SFC-enabled service provisioning in edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 393–408, Jan. 2024.
[7] J. Li et al., "AoI-aware user service satisfaction enhancement in digital twin-empowered edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1677–1690, Apr. 2024.
[8] J. Li et al., "Wait for fresh data? digital twin empowered IoT services in edge computing," in *Proc. 20th Int. Conf. Mobile Ad Hoc Smart Syst.*, 2023.
[9] J. Li et al., "AoI-aware, digital twin-empowered IoT query services in mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 4, pp. 3636–3650, Aug. 2024.
[10] J. Li et al., "AoI-aware service provisioning in edge computing for digital twin network slicing requests," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14607–14621, Dec. 2024.
[11] J. Li et al., "Digital twin-enabled service provisioning in edge computing via continual learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 6, pp. 7335–7350, Jun. 2024.
[12] J. Li, W. Liang, M. Chen, and Z. Xu, "Mobility-aware dynamic service placement in D2D-Assisted MEC environments," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2021, pp. 1–6.
[13] J. Li, J. Wang, Q. Chen, Y. Li, and A. Y. Zomaya, "Digital twin-enabled service satisfaction enhancement in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.
[14] Y. Li, W. Liang, Z. Xu, W. Xu, and X. Jia, "Budget-constrained digital twin synchronization and its application on fidelity-aware queries in edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 165–182, Jan. 2025.
[15] X. Liang, W. Liang, Z. Xu, Y. Zhang, and X. Jia, "Multiple service model refreshments in digital twin-empowered edge computing," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2672–2686, Sep./Oct. 2024.
[16] T. Liu, L. Tang, W. Wang, Q. Chen, and X. Zeng, "Digital-twin-assisted task offloading based on edge collaboration in the digital twin edge network," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1427–1444, Jan. 2022.
[17] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive edge association for wireless digital twin networks in 6G," *IEEE Internet Things J.*, vol. 8, no. 22, pp. 16219–16230, Nov. 2021.
[18] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter, 2017.
[19] Y. Miao et al., "Mobility-aware service migration for seamless provision: A reinforcement learning approach," in *Proc. IEEE Int. Conf. Commun.*, 2022, pp. 1–6.
[20] NetworkX, 2023. [Online]. Available: https://networkx.org/
[21] D. Raggett, "The web of things: Challenges and opportunities," *IEEE Comput.*, vol. 48, no. 5, pp. 26–32, May 2015.
[22] D. Ren, X. Gui, and K. Zhang, "Adaptive request scheduling and service caching for MEC-assisted IoT networks: An online learning approach," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17372–17386, Sep. 2022.
[23] D. Shomys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, pp. 461–474, 1993.

[24] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–14, 1988.
[25] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12240–12251, Oct. 2020.
[26] V. Vazirani, *Approximation Algorithms*. Berlin, Germany: Springer, 2001.
[27] 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, Jun. 2019.
[28] Z. Xu et al., "Near optimal learning-driven mechanisms for stable NFV markets in multitier cloud networks," *IEEE/ACM Trans. Netw.*, vol. 30, no. 6, pp. 2601–2615, Dec. 2022.
[29] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. M. Lau, "Moving Big Data to the cloud: An online cost-minimizing approach," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2710–2721, Dec. 2013.
[30] R. Zhang, Z. Xie, D. Yu, W. Liang, and X. Cheng, "Digital twin-assisted federated learning service provisioning over mobile edge networks," *IEEE Trans. Comput.*, vol. 73, no. 2, pp. 586–598, Feb. 2024.
[31] Y. Zhang and W. Liang, "Cost-aware digital twin migration in mobile edge computing via deep reinforcement learning," in *Proc. 23rd IFIP Netw. Conf.*, 2024, pp. 441–447.
[32] Y. Zhang, W. Liang, W. Xu, Z. Xu, and X. Jia, "Cost minimization of digital twin placements in mobile edge computing," *ACM Trans. Sensor Netw.*, vol. 20, no. 3, pp. 1–2615, 2024.
[33] Y. Zhang, W. Liang, Z. Xu, and X. Jia, "Mobility-aware service provisioning in edge computing via digital twin replica placements," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11295–11311, Dec. 2024.
[34] Y. Zhang, W. Liang, Z. Xu, W. Xu, and M. Chen, "AoI-aware inference services in edge computing via digital twin network slicing," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3154–3170, Nov./Dec. 2024.

Yuncan Zhang (Member, IEEE) received the BE degree from the Dalian University of Technology, Dalian, China, in 2013, the ME degree from the University of Science and Technology of China, Hefei, China, in 2016, and the PhD degree from Kyoto University, Kyoto, Japan, in 2021. She now is a postdoc in the Department of Computer Science, City University of Hong Kong. She worked with Baidu, Beijing, China, from 2016 to 2017. Her research interests include edge computing, digital twin, network function virtualization, and optimization problems.

Luying Wang received the BE and ME degrees from Central South University, China, in 2021 and 2024. Her research interests include edge computing, digital twin and resource allocation.

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