Cost-Aware Digital Twin Migration in Mobile Edge Computing via Deep Reinforcement Learning

Yuncan Zhang, Weifa Liang
Department of Computer Science, City University of Hong Kong, Hong Kong, P. R. China

***Abstract*—The past decade experienced an explosive growth on the number of IoT devices connected to the Internet. Digital twins (DTs) emerge as key enablers to provide digital representations of objects for their monitoring, simulation, prediction and maintenance. At the same time, mobile edge computing (MEC) is envisioned as a promising paradigm to provide various delay-sensitive services for mobile users at the edge of core networks. In this paper, we study a novel cost-aware DT migration problem for effective service provisioning in an DT-empowered MEC network within a finite time horizon, with the aim to minimize the service cost. We first show the NP-hardness of the problem, and formulate an integer linear programming solution to the offline version of the problem, assuming that the mobility information of both objects and users for the given time horizon is given. Considering the system dynamics and heterogeneity of various resource usages, the mobility of objects and users, we then develop an efficient deep reinforcement learning algorithm for the online DT migration problem. We finally evaluate the performance of the proposed algorithms. Experimental results demonstrate that the proposed algorithms are promising.**

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

I. Introduction

In the last decades, the need for communications everywhere and anytime leads to billions of IoT devices connected to the Internet [7]. To enable users to access, control, and monitor these devices via the Internet, digital twins (DTs) are proposed to model the objects in a digital format. Through DT modeling, a virtual representation of an object is constructed to record its historical data, and apply artificial intelligence (AI) technology to simulate its behaviors and provide predictive decisions, which brings advanced capabilities in service provisioning and also improves the performance of the object [11].

The growth of mobile IoT applications has increased the volume of data generated at the network edge. Mobile edge computing (MEC) that brings the computing and storage capabilities to the edge of networks was conceived in a bid to fill the gap between centralized clouds and mobile devices [5]. Orthogonal to the development of the DT technology, how to make use of DTs for service provisioning in MEC networks has attracted much attention [2], [4]. Since each DT stores the data that are synchronized from its object to enable real-time simulation, and provides various service through data analysis, the DT-assisted provisioning in an MEC network incurs cost due to the consumption of computing, storage, and communication resources.

ISBN 978-3-903176-63-8 © 2024 IFIP

The placement locations of DTs in the MEC network are not trivial. In this work, we treat the objects that generate data and maintain their DTs in cloudlets as *suppliers* and treat users requesting information from the DTs as *consumers*. The DT placement affects not only the synchronization cost between DTs and their suppliers but also the service cost of consumer requesting for DT services [13]. Furthermore, the mobility of suppliers and consumers are inevitably involved in the MEC network, which makes it more challenging to decide the DT location of each supplier. When a supplier or a consumer moves out the area covered by an Access Point (AP), the DT update cost of the supplier from its new location or the service cost of consumer requesting for the DT will change due to that the routing paths between the DT location and its supplier or consumer locations change. The DT replicas placement has been considered in [14] to provide delay-sensitive service in a DT-empowered MEC network with the mobility of both users and suppliers. Migrating DTs to proper locations is another promising approach to mitigate the mobility impact of objects and users on DT services. The existing works [3], [9] considering DT migration focus on task offloading to reduce offloading latency or improve user utility. In this paper, we investigate the cost-aware DT migration in the MEC network.

DT migration among cloudlets poses several challenges. On one hand, the DT update cost and the service cost of consumers requesting the DT data change when a DT is migrated to a new location. On the other hand, the historical data stored at the DT must be transmitted from its original location to the new location, and such a data migration overhead cannot be neglected. As frequent DT migrations will lead to the excess migration cost, we need to decide whether and when to perform DT migration for each supplier. We also need to identify a new DT location for migration while respecting the limited storage and computing capacities of the new location, as the new location might be the migration destinations of DTs of multiple objects.

The main contributions of the paper are given as follows. We first formulate a novel DT migration problem in an MEC network with the aim to minimize the accumulative service cost that consists of the DT update cost from suppliers, the service cost of consumers requesting DT data of the suppliers, and the DT migration cost between cloudlets. We also show the NP-hardness of the problem. We then propose an integer linear programming (ILP) solution to the offline version of the problem, assuming that the mobility information of both suppliers and consumers is given. We also develop a novel deep reinforcement learning (DRL) algorithm for the online DT migration problem, by considering the dynamics of the system and heterogeneity of resource usages, the mobility of suppliers and consumers, and formulating the problem as a Markov Decision Process (MDP). We finally evaluate the performance of proposed algorithm. Simulation results show that the proposed algorithms are promising.

The remainder of the paper is organized as follows. Section II introduces the system model and defines the problem formally. Section III formulates an ILP solution to the offline version of the problem. Section IV devises an DRL algorithm for the DT migration problem. Section V evaluates the proposed algorithms, and Section VI concludes the paper.

II. Preliminary

In this section, we first introduce the system model. Then we detail the notations and cost modeling. We finally give the problem definition precisely.

A. System model

We consider an MEC network $G = (\mathcal{N}, E)$, where $\mathcal{N}$ is the set of APs; each AP is co-located with a cloudlet, and the communication delay between a AP and its co-located cloudlet is negligible. Without loss of generality, we use AP or its co-located cloudlet interchangeably. Each cloudlet $n \in \mathcal{N}$ has limited computing capacity C_n and storage capacity $\mathcal{K}_n$ for hosting DTs. Let ζ_n and $\varkappa_n$ denote the costs per unit computing and storage resource usage of cloudlet $n \in \mathcal{N}$, respectively. There is a set E of optical links between cloudlets interconnecting them together. Let ξ_e denote the communication cost per unit data on link $e \in E$.

We assume that there is a set O of physical objects, including suppliers and consumers under the coverage of APs in G; let V and U denote the sets of suppliers and consumers, respectively, and $O = V \cup U$. For the sake of convenience, we use index i, j, k to represent the indexes of suppliers, cloudlets, and consumers, respectively. Each supplier $v_i \in V$ has a DT, denoted by DT_i, placed in a cloudlet of the MEC network, which needs to allocate storage resource Φ_i of recording the DT data and the requested amount c_i of computing resource for data processing for supplier v_i. Each consumer $u_k \in U$ requests the DT information from a supplier $r_{u_k} \in V$. For a finite time horizon $\mathbb{T}$ that is divided into equal slots, i.e., $\mathbb{T} = \{1, 2, \cdots, T\}$, we assume that both suppliers and consumers are allowed to move around within the network. Considering the mobility of objects, let $g_o(t) \in \mathcal{N}$ denote the cloudlet at which object $o \in O$ is located at time slot $t \in T$. Denote by $U(v_i, t) \subseteq U$ the set of consumers requesting service on DT_i of supplier $v_i \in V$ at time slot $t \in \mathbb{T}$. We further assume that each user u_k only requests one DT at each time slot t, i.e., $U(v_i, t) \cap U(v_{i'}, t) = \emptyset$ if $i \neq i'$.

B. Cost modeling

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

DT synchronization cost of a supplier: At time slot $t \in T$, supplier $v_i \in V$ produces the amount $\Phi_i(t)$ of data for updating its DT_i. We assume that DT_i of supplier v_i needs to store the data generated within $[t - t_0^{(i)} + 1, t]$ time slots locally to ensure the functionality of the DT, where $t_0^{(i)}$ is its last synchronization time slot. Then, the amount of data stored for DT_i at time slot t is

$$\Phi_i(t) = \sum_{t'=t-t_0^{(i)}+1}^{t} \Phi_i(t'). \tag{1}$$

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

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

For the sake of convenience, denote by cloudlet $n_i(t)$ hosting 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

$$\begin{aligned}\mathcal{E}_i^s(t) = &\sum_{n \in N} \zeta_n \cdot c_i \cdot x_{i,n}(t) + \sum_{n \in N} \varkappa_n \cdot \Phi_i(t) \cdot x_{i,n}(t) \\ &+ \sum_{e \in P_{g_i(t), n_i(t)}} \xi_e \cdot \Phi_i(t),\end{aligned} \tag{3}$$

where $P_{g_i(t), n_i(t)}$ is the shortest path in G between cloudlets $g_i(t)$ and $n_i(t)$ in terms of the communication cost ξ_e of each link e on it. It can be seen that $\mathcal{E}_i^s(t)$ consists of the computing resource cost, storage resource cost, and communication resource cost.

Service request cost of a consumer: For each consumer $u \in U$ at time slot $t \in [1, T]$, its requested information needs to be transmitted from the DT of supplier $r_u \in V$ to cloudlet $g_u(t)$ at which u is located, such information transfer consumes network communication resource. Assuming that the amount of data transmitted from the DT of supplier $v_i \in V$ to its consumer at time slot t after being processed at DT_i is $\epsilon_i \cdot \Phi_i(t)$, where ϵ_i is a constant, which is determined by different applications with $0 < \epsilon_i < 1$.

Recall that $U(v_i, t)$ is the set of consumers requesting services from DT_i of supplier v_i. Assuming DT_i is hosted by cloudlet $n_i(t)$, the service cost of consumers in $U(v_i, t)$ requesting services from DT_i of v_i can be calculated through finding a Steiner tree $T_{v_i, n_i(t)}^{opt}$ in a graph $G_{v_i} = (\mathcal{N}, E; {w'}_i^t(\cdot))$, where $T_{v_i, n_i(t)}^{opt}$ is a tree rooted at cloudlet $n_i(t)$ that hosts DT_i of supplier v_i and spanning the cloudlets at which consumers in $U(v_i, t)$ are located, and G_{v_i} is a copy of the MEC network $G(\mathcal{N}, E)$ for supplier v_i. The cost ${w'}_i^t(e)$ of each edge $e \in E$ in graph G_{v_i} is defined as

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

where ξ_e is the communication cost per unit data on edge e.

The service request cost of all consumers in $U(v_i, t)$ requesting services from DT_i of v_i is the sum of edge costs in Steiner tree $T^{opt}_{v_i,n_i(t)}$, i.e.,

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

Due to the NP-hardness of the Steiner tree problem, we instead construct an approximate Steiner tree T_{v_i,n_j} in graph G_{v_i} for $T^{opt}_{v_i,n_j}$. It is well known that $\sum_{e \in T_{v_i,n_j}} {w'}^t_i(e) \le 2 \cdot OPT_{v_i}$ [10], where OPT_{v_i} is the optimal cost of Steiner tree $T^{opt}_{v_i,n_j}$.

DT migration cost: Due to the movement of both suppliers and consumers, the optimal DT placements in different time slots are different. For a given supplier v_i, if the location of DT_i changes at time slot $t \in [1, T]$, i.e., $n_i(t-1) \neq n_i(t)$, then the historical data $\Phi_i(t)$ of DT_i needs to be moved from its previous location to the current location too, and a migration cost thus incurs.

Let $P_{n,n'}$ be the shortest path in G between cloudlets n and n'. The migration cost $\mathcal{E}^{\text{m}}_i$ of DT_i of supplier v_i at time slot t is

$$\mathcal{E}^{\text{m}}_i(t) = \Phi_i(t) \cdot \sum_{n \in \mathcal{N}} \sum_{n' \in \mathcal{N} \setminus \{n\}} \sum_{e \in P_{n,n'}} \xi_e \cdot x_{i,n}(t-1) \cdot x_{i,n'}(t), \quad \forall t \in [2, T]. \tag{6}$$

Only when $x_{i,n}(t-1) \cdot x_{i,n'}(t) = 1$ in Eq. (6), DT_i of the supplier $v_i \in V$ can move from location n to location n'.

The sum $\mathcal{E}(t)$ of the DT migration cost of all suppliers and the service cost of all consumers requesting for DT data at each time slot $t \in [1, T]$ is

$$\begin{aligned}\mathcal{E}(t) &= \sum_{v_i \in V} \mathcal{E}^{\text{s}}_i(t) + \sum_{v_i \in V} \mathcal{E}^{\text{m}}_i(t) + \sum_{u_k \in U} \mathcal{E}_{u_k}(t) \\ &= \sum_{v_i \in V} [\mathcal{E}^{\text{s}}_i(t) + \sum_{u \in U(v_i,t)} \mathcal{E}_u(t) + \mathcal{E}^{\text{m}}_i(t)]. \end{aligned} \tag{7}$$

The overall service cost $\mathcal{E}_T$ of DT migrations within a given time horizon $\mathbb{T}$ thus is

$$\mathcal{E}_T = \sum_{t=1}^{T} \mathcal{E}(t). \tag{8}$$

C. Problem definition

Given an MEC network $G = (\mathcal{N}, E)$, a finite time horizon $\mathbb{T}$, and a set O of objects that is the union of set V of suppliers and set U of consumers with $O = V \cup U$, each supplier has a DT placed in a cloudlet in G for service provisioning, and each consumer requests DT data from a specific supplier at a time slot. Assume that both suppliers and consumers are allowed to move in G at different time slots. The *DT migration problem* is to determine whether the DT of each supplier to be migrated to a new location at each time slot so that the accumulative service cost $\mathcal{E}_T$ in Eq. (8) of all requests within $\mathbb{T}$ is minimized, subject to both computing and storage capacities on each cloudlet.

D. NP-hardness

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

Proof Due to limited space, the proof is omitted.

III. ILP FORMULATION

In this section, we consider the offline version of the DT migration problem, assuming that the mobility information of suppliers and consumers at each time slot t is given in advance, for which we first formulate an integer nonlinear program (INP) solution, and we then transform the INP solution to an equivalent ILP solution.

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

$$\begin{aligned}
&\text{Minimize} \quad \mathcal{E}_T && \text{(9a)}\\
&\text{s.t.} \quad (1) - (8) && \text{(9b)}\\
&\sum_{v_i \in V} c_i \cdot x_{i,n}(t) \le C_n, \quad \forall n \in \mathcal{N}, \forall t \in [1, T] && \text{(9c)}\\
&\sum_{v_i \in V} \Phi_i(t) \cdot x_{i,n}(t) \le \mathcal{K}_n, \quad \forall n \in \mathcal{N}, \forall t \in [1, T] && \text{(9d)}\\
&x_{i,n}(t) \in \{0, 1\}, \quad \forall v_i \in V, n \in \mathcal{N}, t \in [1, T] && \text{(9e)}
\end{aligned}$$

Eqs. (9c) and (9d) ensure that the total computing and storage resource consumptions on any cloudlet cannot exceed its computing and storage capacities at any time slot, respectively.

The problem in (9) is a nonlinear optimization problem due to that Eq. (6) is nonlinear, which contains the product of two binary variables $x_{i,n}(t-1)$ and $x_{i,n'}(t)$, respectively. We transform the INP solution into an equivalent ILP solution, through linearizing Eq. (6) by introducing a binary variable $y_{i,n,n'}(t)$, i.e.,

$$y_{i,n,n'}(t) \le x_{i,n}(t-1), \quad \forall t \in [2, T], v_i \in V, n, n' \in \mathcal{N} \text{ with } n \neq n' \tag{10}$$

$$y_{i,n,n'}(t) \le x_{i,n'}(t), \quad \forall t \in [2, T], v_i \in V, n, n' \in \mathcal{N} \text{ with } n \neq n' \tag{11}$$

$$y_{i,n,n'}(t) \ge x_{i,n}(t-1) + x_{i,n'}(t) - 1, \quad \forall t \in [2, T], v_i \in V, n, n' \in \mathcal{N} \text{ with } n \neq n' \tag{12}$$

Eq. (6) then is replaced by

$$\mathcal{E}^{\text{m}}_i(t) = \Phi_i(t) \sum_{n \in \mathcal{N}} \sum_{n' \in \mathcal{N} \setminus \{n\}} \sum_{e \in P_{n,n'}} \xi_e \cdot y_{i,n,n'}(t). \tag{13}$$

An ILP formulation for the problem then is given as follows.

$$\begin{aligned}
&\text{Minimize} \quad \mathcal{E}_T && \text{(14a)}\\
&\text{s.t.} \quad (1) - (5), (7), (8), (9c), (9d), (9e), (10) - (13) && \text{(14b)}\\
&\quad y_{v,n,n'}(t) \in \{0, 1\}, \forall t \in [2, T], v \in V, \; n, n' \in \mathcal{N} \text{ with } n \neq n' && \text{(14c)}
\end{aligned}$$

IV. ONLINE ALGORITHM

In this section, we present a DRL algorithm for the DT migration problem for a given monitoring period, in which the mobility information of suppliers and consumers is not known in advance. We formulate the DT migration problem as an MDP, where there is an agent for each supplier and the set of agents learn how to act (policy) in a sequential decision-makings through interacting with the environment. We then develop a DRL algorithm for the problem based on actor-critic networks.

A. MDP formalization

An MDP is expressed by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ is the state set of the environment, $\mathcal{A}$ is the set of actions, $\mathcal{P}$ is the transition function, and $\mathcal{R}$ is the reward function. We assume that the entire monitoring period is $\mathbb{T}$ that consists of equal time slots. Given state s_t at time slot t, an action $\mathbf{a}^t$ 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}, $\mathcal{P}(s_t, \mathbf{a}^t, s_{t+1})$ is the probability of the 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 reward obtained when transiting state from s_t to s_{t+1} by action $\mathbf{a}^t$. In the following we detail each component of the MDP.

State: In 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 stored data volumes of suppliers $\Phi(t) = (\Phi_1^t, \Phi_2^t, \ldots, \Phi_{|V|}^t)$, the DT placement $\mathbf{p}(t-1) = (p_1^{t-1}, p_2^{t-1}, \ldots, p_{|V|}^{t-1})$ of suppliers at time slot $t-1$, the load of each cloudlet $\mathbf{l}(t-1) = (l_1^{t-1}, l_2^{t-1}, \ldots, l_{|\mathcal{N}|}^{t-1})$ under the DT placement $\mathbf{p}(t-1)$, and the locations of consumers $\mathbf{b}(t) = (b_1^t, b_2^t, \ldots, b_{|U|}^t)$, where $g_i^t \in [1, |\mathcal{N}|], i \in [1, |V|]$ indicates the cloudlet in which supplier v_i is located, $b_k^t \in [1, |\mathcal{N}|], k \in [1, |U|]$ represents the cloudlet in which consumer u_k is located, $p_i^t \in [1, |\mathcal{N}|], i \in [1, |V|]$ is the cloudlet hosting the DT, DT_i, of supplier v_i, and l_j^t indicates the resource utilization of cloudlet $j \in [1, |\mathcal{N}|]$. Specifically, l_j^{t-1} of cloudlet n_j is a tuple consisting two elements, 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 on cloudlet n_j corresponding to DT placement $\mathbf{p}(t-1)$, respectively.

Let state $s_t \in \mathcal{S}$ be the system state at time slot t, which is characterized by the following tuple

$$s_t = (\mathbf{g}(t), \Phi(t), \mathbf{p}(t-1), \mathbf{l}(t-1), \mathbf{b}(t)). \tag{15}$$

Action: At each time slot t, each agent (supplier) decides whether to migrate its DT or not. Due to the capacity constraint on each cloudlet, the DT migration choices of agents are mutually influential. An action of an agent is the placement choice at time slot $t \in T$ and DT migration is required when the placement choice is different from its placement at previous time slot $t-1$. Specifically, the action of supplier v_i 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 the DT of supplier v_i is placed at cloudlet n_j, and $a_{i,j}^t = 0$ implies that cloudlet n_j is not selected to host the DT of supplier v_i, and $\sum_{j=1}^{|\mathcal{N}|} a_{i,j}^t = 1$ for any i with $1 \le i \le |V|$. Then, action $\mathbf{a}^t \in \mathcal{A}$ is defined as $\mathbf{a}^t = (a_1^t, a_2^t, \ldots, a_{|V|}^t)$.

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 $P(s_t, \mathbf{a}^t, s_{t+1})$. DT migrations are conducted for agents whose DT placement locations are different from the DT locations in the previous time slot $t-1$, and thus $\mathbf{p}(t)$ and $\mathbf{l}(t)$ are then updated accordingly. As suppliers and consumers can move around within the MEC network, $\mathbf{g}(t)$ 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 Eq. (1).

Reward: To maximize the accumulative reward of all agents, a reward function $\mathcal{R}(s_t, \mathbf{a}^t, s_{t+1})$ is defined, which is the amount of reward received by all agents after the state transition from s_t to s_{t+1}. The system utilizes the reward function to evaluate actions of agents, where the reward function is related to the total cost of selecting a specified cloudlet for DT migration of each supplier.

For 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 its DT from its current location to cloudlet n_j at time slot t. The cost $cost_{\mathrm{s}}^t(v_i, n_j)$ of supplier v_i synchronizing its DT on cloudlet n_j is

$$cost_{\mathrm{s}}^t(v_i, n_j) = \zeta_j \cdot c_i + \varkappa_{n_j} \cdot \Phi_i(t) + \sum_{e \in P_{g_i^t, n_j}} \xi_e \cdot \Phi_i(t), \tag{16}$$

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 $e \in E$; 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 of supplier v_i and ζ_j is cost per unit of computing resource of cloudlet n_j. The three items in the right side of Eq. (16) correspond the computing resource cost, storage resource cost, and data transmission cost of the DT data of supplier v_i, respectively.

Recall that $U(v_i, t)$ is the set of consumers requesting services from DT_i of supplier v_i. The service cost of consumers in $U(v_i, t)$ requesting services from DT_i of v_i can be calculated through finding an approximate Steiner tree T_{v_i, n_j} in a graph $G_{v_i} = (\mathcal{N}, E; {w'}_i^t(\cdot))$, where T_{v_i, n_j} is a tree rooted at cloudlet n_j and spanning the cloudlets at which consumers in $U(v_i, t)$ are located, and G_{v_i} is a copy of the MEC network $G(\mathcal{N}, E)$ for supplier v_i. The cost ${w'}_i^t(e)$ of each edge $e \in E$ in graph G_{v_i} is defined in Eq. (4).

The service cost $cost_{\mathrm{c}}^t(v_i, n_j)$ of all consumers in $U(v_i, t)$ requesting services from DT_i of v_i is the sum of edge costs in approximate Steiner tree T_{v_i, n_j}, i.e.,

$$cost_{\mathrm{c}}^t(v_i, n_j) = \sum_{e \in T_{v_i, n_j}} {w'}_i^t(e). \tag{17}$$

The communication cost $cost_{\mathrm{m}}^t(v_i, n_j)$ of migrating the DT of supplier v_i from its current cloudlet p_i^{t-1} to the next cloudlet

n_j is

$$cost^t_{\rm m}(v_i, n_j) = \Phi_i(t) \cdot \sum_{e \in P_{p_i^{t-1}, n_j}} \xi_e, \tag{18}$$

where $P_{p_i^{t-1}, n_j}$ is the shortest routing path in the MEC network G between cloudlets p_i^{t-1} and n_j.

The total cost $w^t_{i,j}$ of DT_i migration of supplier v_i from its current cloudlet p^{t-1} to cloudlet n_j at time slot t is

$$w^t_{i,j} = cost^t_{\rm s}(v_i, n_j) + cost^t_{\rm c}(v_i, n_j) + cost^t_{\rm m}(v_i, n_j). \tag{19}$$

The cost vector $\mathbf{w}_i$ of supplier v_i of migrating DT_i to all potential cloudlets in the MEC network is represented by

$$\mathbf{w}^t_i = (w^t_{i,1}, w^t_{i,2}, \ldots, w^t_{i,|\mathcal{N}|}). \tag{20}$$

The reward of choosing action a^t_i for agent i (supplier v_i) is

$$\mathcal{R}_i(s_t, a^t_i, s_{t+1}) = -a^t_i \cdot (\mathbf{w}^t_i)^{\rm tr}, \tag{21}$$

where $(\mathbf{w}^t_i)^{\rm tr}$ is the transpose of vector $\mathbf{w}^t_i$.

We define the reward obtained by action $\mathbf{a}^t$ for all agents at time slot t as follows.

$$\mathcal{R}(s_t, \mathbf{a}^t, s_{t+1}) = \sum_{i=1}^{|V|} \mathcal{R}_i(s_t, a^t_i, s_{t+1}) \tag{22}$$

For the convenience, let $\mathcal{R}_t = \mathcal{R}(s_t, \mathbf{a}^t, s_{t+1})$. The goal of the MDP is to maximize the expectation of the cumulative discounted reward $\mathcal{R}$ for a given monitoring period $\mathbb{T}$, which is defined as follows.

$$\mathcal{R} = \sum_{t=1}^{T} \gamma^{t-1} \mathcal{R}_t, \tag{23}$$

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

Note that the service cost $\mathcal{E}(t)$ of all suppliers at time slot t in Eq. (7) can be rewritten as

$$\begin{aligned}\mathcal{E}(t) &= \sum_{v_i \in V} \mathcal{E}^{\rm s}_i(t) + \sum_{u_k \in U} \mathcal{E}_{u_k}(t) + \sum_{v_i \in V} \mathcal{E}^{\rm m}_i(t) \\ &= \sum_{i=1}^{|V|} a^t_i \cdot (\mathbf{w}^t_i)^{\rm tr}, \end{aligned} \tag{24}$$

The goal of the DT migration problem is to minimize the service cost, which is to maximize the cumulative reward as follows.

$$\mathbb{E}(\mathcal{R}) = \mathbb{E}(\sum_{t=1}^{T} \gamma^{t-1} \mathcal{R}_t). \tag{25}$$

B. DRL-based algorithm

We develop an actor-critic based DRL algorithm to search the best policy for DT migrations. The proposed algorithm consists of two neural networks: the actor network and critic network.

The actor network tries to provide the optimal DT placement choice at each time slot by the policy π_i, i.e., the probability of selecting an action given a state, for each agent i. Specifically, the input of the actor network is a system state s_t, and the output is probability distribution of each action. Then the actor agent determines its action a_i following the policy $\pi_{\theta_i}(a_i \mid s_t)$, where θ_i is the parameter of the actor network. The actor network adopts the policy gradient descent to update parameter θ_i, by taking steps in the direction of

$$\nabla_{\theta_i} J(\theta_i) \approx \mathbb{E}_{\pi_{\theta_i}}[\nabla_{\theta_i} \log \pi_{\theta_i}(s_t, a^t_i) \delta_t(s_t, a^t_i)], \tag{26}$$

where $\delta_t(s_t, a^t_i)$ is the Temporal Difference (TD) error that will be detailed later. The actor network is trained as

$$\theta_i = \theta_i + \eta_{\theta_i} \nabla_{\theta_i} \log \pi_{\theta_i}(s_t, a^t_i) \delta_t(s_t, a^t_i), \tag{27}$$

where η_θ is the learning rate of the actor network.

The critic network evaluates the state-value function under the actor policy. For agent i, it takes state s_t as input, and its output is estimated by the state-value function $V^{\pi_i}(s_t \mid \omega_i)$, where ω_i is the parameter of the critic network. The TD error is calculated by

$$\delta_i(t) = \mathcal{R}_i(s_t, a_i, s_{t+1}) + \gamma V(s_{t+1} \mid \omega_i) - V(s_t \mid \omega_i), \tag{28}$$

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

$$\omega_i = \omega_i + \eta_w \delta^2_i(t) \nabla_{\omega_i} V(s_t \mid \omega_i), \tag{29}$$

where η_w is the learning rate of the critic network.

The detailed DRL algorithm for training the actor-critic networks is shown in `Algorithm.1`.

Algorithm 1 The actor-critic training algorithm.

```
Input: An MEC network $G = (\mathcal{N}, E)$, historical mobility and service
    request information of suppliers and users in the monitored area, a given
    discount factor $\gamma$.
Output: The trained actor and critic network parameters $\theta$ and $w$.
 1: Initialize the parameters $\theta$ of the actor network and $w$ of the critic network
    randomly;
 2: while The parameters of actor-critic networks have not converged do
 3:     Select a set of historical mobility and service request information of
        suppliers and users;
 4:     Initialize the DT placement for suppliers in the MEC network and
        obtain the initial state;
 5:     for $t \leftarrow 1$ to $T$ do
 6:         for $i \leftarrow 1$ to $|V|$ do
 7:             Obtain action $a^t_i$ at state $s_t$ using the output policy;
 8:             Calculate the reward $\mathcal{R}_i$ of agent $i$ at time slot $t$;
 9:         Obtain new state $s_{t+1}$ based on the actions of agents;
10:     for $i \leftarrow 1$ to $|V|$ do
11:         for $t \leftarrow 1$ to $T$ do
12:             Input states $s_t$ and $s_{t+1}$ into the critic network and obtain
                $V(s_t \mid \omega_i)$ and $V(s_{t+1} \mid \omega_i)$ of agent $i$;
13:             Calculate the TD error $\delta_i(t)$ using Eq. (28);
14:             Update the parameters of critic and actor networks for agent $i$
                by Eq. (29) and Eq. (26) respectively;
15: return the trained actor and critic network models.
```

The DRL algorithm for the DT migration problem is given in `Algorithm 2`, or `Alg.2` for short. The DT placement of `Alg.2` is initialized by setting state s_1 and the cumulative cost $\mathcal{E}_T$. Within each time slot $t \in [1, T]$, an action $\mathbf{a}^t$ is chosen by policy function $\pi_i(s|\theta_i)$ of each agent, and the value of a^t_{ij} of each cloudlet $n_j \in \mathcal{N}$ for each supplier $v_i \in V$ in $\mathbf{a}^t$ is

Algorithm 2 The DRL algorithm for the DT migration problem.

```
Input: An MEC network $G = (\mathcal{N}, E)$ with each cloudlet $n_j \in \mathcal{N}$ has
    storage capacity $\mathcal{K}_j$ and computing capacity $C_j$, a set of suppliers $V$, a
    set of consumers $U$, a given time horizon $\mathbb{T}$.
Output: The cumulative service cost for the time horizon $\mathbb{T}$, and the DT
    migration choices at each time slot $t \in \mathbb{T}$.
1: Train actor-critic networks by invoking Alg.1;
2: Initialize the DT placement of suppliers in cloudlets, and initialize state $s_1$;
3: $\mathcal{E}_T \leftarrow 0$; $P \leftarrow \emptyset$; /*migration cost; migration solution*/
4: for $t \leftarrow 1$ to $T$ do
5:     for each supplier $v_i \in V$ do
6:         Obtain action $a_i^t$ at state $s_t$ by the actor policy function $\pi_i(s_t|\theta_i)$;
7:         for each cloudlet $n_j \in \mathcal{N}$ do
8:             if $a_{ij}^t = 1$ and $n_j \neq p_i^{t-1}$ with sufficient resource then
9:                 Migrate $DT_i$ from cloudlet $p_i^{t-1}$ to cloudlet $n_j$;
10:                $p_i^t \leftarrow n_j$;
11:                Update resources in cloudlets $n_j$ and $p_i^{t-1}$;
12:                $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij}^t$;
13:            else if $a_{ij}^t = 1$ and $n_j \neq p_i^{t-1}$ and $n_j$ does not have sufficient
               remaining resource then
14:                $p_i^t \leftarrow n_{j'}$, cloudlet $n_{j'}$ has the lowest $w_{i,j'}^t$ in vector $\mathbf{w}_i^t$
                   defined in Eq. (20);
15:                Update the resource usages of cloudlets $p_i^{t-1}$ and $n_{j'}$;
16:                $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij'}^t$;
17:            else if $a_{ij}^t = 1$ and $n_j = p_i^{t-1}$ then
18:                Update the resource utilization of cloudlets $n_j$;
19:                $\mathcal{E}_T \leftarrow \mathcal{E}_T + w_{ij}^t$;
20:        $P \leftarrow P \cup \{p_i^t\}$;
21: return $P$ with cost $\mathcal{E}_T$
```

examined. When $a_{ij}^t = 1$, check whether $n_j = p_i^{t-1}$. If $n_j \neq p_i^{t-1}$ and cloudlet n_j has sufficient resource to accommodate DT_i, DT_i will be migrated from cloudlet p_i^{t-1} to cloudlet n_j; if cloudlet n_j does not have sufficient resource for DT migration of supplier v_i, cloudlet $n_{j'}$ with the lowest $w_{i,j'}^t$ in vector $\mathbf{w}_i^t$ defined in Eq. (20) is chosen. Then the resource utilization of all involving cloudlets for this migration, the location p_i^t of DT_i at time slot t, and the cumulative migration cost from beginning till time slot t, are updated accordingly.

V. Performance evaluation

In this section, we evaluated the performance of the proposed ILP solution and DRL algorithm for the DT migration problem. We also investigated the impact of parameters on the performance of the proposed algorithms.

A. Experimental settings

We consider an MEC network that consists of 10 APs, each of which is equipped with a cloudlet. The topology of the MEC network is generated via NetworkX [6]. The computing capacity $\mathcal{C}_j$ and storage capacity $\mathcal{K}_j$ of each cloudlet $n_j \in \mathcal{N}$ are randomly selected from the range of [20, 50] GHz and [150, 300] GB [8], respectively. For each cloudlet $n_j \in \mathcal{N}$, the cost ζ_j of per unit computing resource is randomly drawn in \$[0.4, 0.8] per GHz [1], and the cost $\varkappa_j$ of per unit storage resource varies from \$0.01 to \$0.014, respectively. The communication cost ξ_e of each link $e \in E$ per GB is randomly chosen from \$0.05 to \$0.12 [12]. Suppliers and consumers are randomly distributed under the coverage of APs in the network. The requested amount c_i of computing resource of each supplier $v_i \in V$ varies from 3GHz to 10GHz [1]. The amount $\Phi_i(t)$ of the update data generated by supplier $v_i \in V$ at each time slot is drawn in [1, 5] GB randomly. The DT synchronization intervals of different suppliers vary, which are drawn in an time slot interval [1, 5]. The compression rate ϵ_i of the processed update data at DT_i of each supplier v_i is randomly selected in [0.10, 0.25]. In addition, the time horizon is set at $T = 20$ [1], and both suppliers and consumers are not stationary, they can move within the network randomly between different time slots, but they are assumed to be stationary within a single time slot.

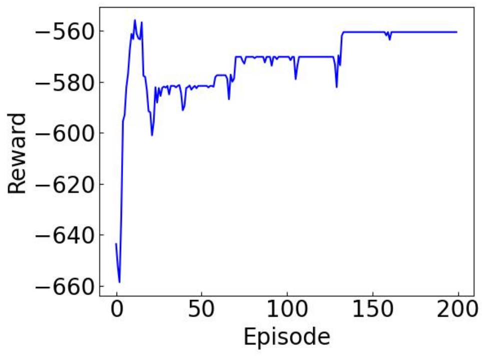


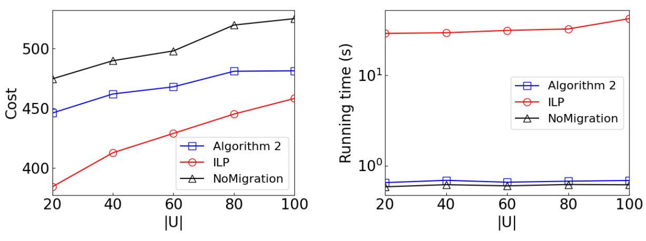
Fig. 1. Learning process of actor-critic networks by Alg.1.

We evaluated the proposed DRL algorithm, Alg.2, for the DT migration problem against the optimal solution obtained by the ILP in Eq. (14). We obtain the ILP solution with given the mobility information of all suppliers and consumers, and this optimal solution serves as a lower bound on the optimal solution of the online DT migration problem. We also consider another benchmark, NoMigration, in which the DT placement of each supplier stays the same as the initial placement through all the time slots. The value in each figure is the mean of the results out of 20 MEC instances with the same size. The actual running time of each algorithm is obtained on a desktop equipped with an Intel(R) Xeon(R) Platinum 8280 2.70GHz CPU, with 10GB RAM. These parameters are adopted as the default settings unless otherwise specified.

B. Performance evaluation of algorithms

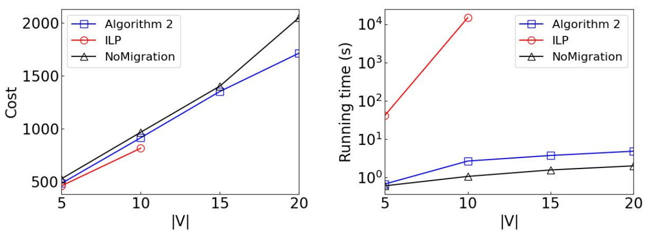
We first trained the actor-critic networks by Alg.1. We then compared the performance of Alg. 2, using the trained actor networks with ILP and NoMigration.

We studied the training of actor-critic networks in Alg.1 by considering five suppliers and 100 consumers in the MEC network. Fig. 1 depicts the learning process of actor-critic networks in terms of total reward of agents. It can be observed from Fig. 1 that the value of reward fluctuates greatly in the early stage of training. With the training continuing, the rewards increase, which means that the total cost of DT-assisted service provisioning decreases on the given time horizon. After about 200 episodes, the reward becomes stable,



(a) Total cost (b) Running time

Fig. 2. Performance comparison of ILP, Alg. 2, and `NoMigration` by varying the number of consumers.



(a) Total cost (b) Running time

Fig. 3. Performance comparison of ILP, Alg. 2, and `NoMigration` by varying the number of suppliers.

which implies that the training of actor-critic networks in `Alg`.1 converges in the end. In the following simulation, the training episode is set at 200 as default for different settings of the numbers of suppliers and consumers.

We investigated the performance of `Alg`. 2 for the DT migration problem against the ILP and `NoMigration`. We first vary the number of consumers while fixing the number of suppliers at five. Fig. 2 plots the performance curves of `Alg`. 2, ILP, and `NoMigration`, respectively. It can be seen from Fig. 2 that ILP obtains the lowest total cost among the three methods at the expense of the longest running time for the DT migration problem. Compared to the solution by ILP, the solution delivered by `Alg`. 2 has moderate total cost with a much faster running time. `NoMigration` achieves the largest cost for each given number of consumers. The results in Fig. 2(a) shows that performing DT migrations properly under the movements of suppliers and consumers can reduce the total service cost. We then vary the number of suppliers while fixing the number of consumers at 100, where the results are displayed in Fig. 3. As can be seen from Fig. 3(a), the cost of each method increases with the increase of the number of suppliers, as more suppliers consume more resources for DT synchronization and DT migration. It can be observed from Fig. 3(a) that ILP has a lower cost than `Alg`. 2 and `NoMigration` when there are 10 suppliers, i.e., $|V| = 10$. However, it can be seen from Fig. 3(b) that ILP takes a prohibitively long running time while `Alg`. 2 and `NoMigration` run much faster. When the number of suppliers reaches 15, ILP cannot obtain the optimal solution in a reasonable running time. From the above experiments, it can be seen that DT migration is an effective means of reducing the service cost when both suppliers and consumers are movable in the network. ILP can be applied to deliver the DT migration solution in the situation where the mobility information of suppliers and consumers are known and the number of suppliers and consumers are small, otherwise `Alg`. 2 is a better choice.

VI. Conclusion

In this paper, we studied a novel cost-aware DT migration problem of service provisioning in a DT-empowered MEC network. We first showed the NP-hardness of the problem, and formulated an ILP solution to the offline version of the problem, assuming that the mobility information of suppliers and consumers is given. Considering the mobility of suppliers and consumers and consumer service request changing overtime, we then developed a DRL algorithm for the DT migration problem through non-trivial cost modeling and an MDP formulation. We finally evaluated the performance of the proposed DRL algorithm against the benchmarks through experimental simulations. Simulation results demonstrate that the proposed algorithm is promising.

References

[1] J. Li, W. Liang, M. Chen, and Z. Xu. Mobility-aware dynamic service placement in D2D-Assisted MEC environments. *Proc. of WCNC'21*. pp. 1 – 6, IEEE, 2021.

[2] J. Li, S. Guo, W. Liang, Q. Chen, X. Xu, W. Xu, and A. Zomaya. Digital twin-assisted, SFC-enabled service provisioning in edge computing. *IEEE Transactions on Mobile Computing*, vol. 23, no. 1, pp. 393 – 408, 2024.

[3] Y. Lu, S. Maharjan, and Y. Zhang. Adaptive edge association for wireless digital twin networks in 6G. *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16219 – 16230, 2021.

[4] 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 of Things Journal* , vol. 9, no. 2, pp. 1427 – 1444, Jan. 2022.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: the communication perspective. *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[6] NetworkX. Accessed: Jul. 2022. [Online]. Available: https://networkx.org/.

[7] D. Raggett. The web of things: challenges and opportunities. *Computer*, vol. 48, no. 5, pp. 26 – 32, 2015.

[8] D. Ren, X. Gui, K. Zhang. Adaptive request scheduling and service caching for MEC-Assisted IoT networks: an online learning approach. *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17372 – 17386, 2022.

[9] W. Sun, H. Zhang, R. Wang, and Y. Zhang. Reducing offloading latency for digital twin edge networks in 6G. *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12240 – 12251, 2020.

[10] V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[11] H. Wang, Y. Wu, G. Min, and W. Miao. A graph neural network-based digital twin for network slicing management. *IEEE Trans. Indus. Info.*, vol. 18, no. 2, pp. 1367–1376, Feb., 2020.

[12] Z. Xu, H. Ren, W. Liang, Q. Xia, W. Zhou, P. Zhou, W. Xu, G. Wu, and M. Li. Near optimal learning-driven mechanisms for stable NFV markets in multitier cloud networks. *IEEE/ACM Transactions on Networking*, vol. 30, no. 6, pp.2601 – 2615, 2022.

[13] Y. Zhang, W. Liang, W. Xu, Z. Xu, and X. Jia. Cost minimization of digital twin placements in mobile edge computing. *ACM Transactions on Sensor Networks*, vol. 20, no. 3, pp.1 – 2615, 2024.

[14] Y. Zhang, W. Liang, Z. Xu, and X. Jia. Mobility-aware service provisioning in edge computing via digital twin replica placements. *IEEE Transactions on Mobile Computing*, to be published, 2023, doi: 10.1109/TMC.2024.3394839.