Adaptive Digital Twin Migration in Vehicular Edge Computing and Networks

Fangyi Mou, Jiong Lou, *Member, IEEE*, Zhiqing Tang, *Member, IEEE*, Yuan Wu, *Senior Member, IEEE*, Weijia Jia, *Fellow, IEEE*, Yan Zhang, *Fellow, IEEE*, and Wei Zhao, *Fellow, IEEE*

***Abstract*—The surge in mobile vehicles and data traffic in Vehicular Edge Computing and Networks (VECONs) requires innovative approaches for low latency, stable connectivity, and efficient resource usage in fast-moving vehicles. Existing studies have identified that utilizing digital twins (DTs) can effectively improve service quality in VECONs. However, it still faces substantial challenges posed by large-scale complex DT communications in sustaining real-time collaborative endeavors. In particular, within the dynamic VECONs, the decision regarding DT migration plays a pivotal role in sustaining the quality of services. In this paper, we propose an adaptive DT migration (ADM) algorithm to minimize the overall migration costs when DTs deliver services. Specifically, 1) We formulate ADM as a combinatorial optimization problem in VECONs, comprehensively considering communication latency and migration latency under complex DT communications, vehicular mobilities, and dynamic states of edges; 2) An ADM algorithm based on off-policy actor-critic reinforcement learning is proposed to make migration decisions. Moreover, the ADM agent employs warm-up policies to address exploration challenges in sparse state spaces; 3) Simulations based on real-world, large-scale urban vehicular mobility datasets demonstrate that our method outperforms existing algorithms by approximately 39% on average, and it can achieve results close to the optimal.**

***Index Terms*—Digital twin, migration, vehicular edge computing, deep reinforcement learning.**

I. Introduction

THE paradigm of Vehicular Edge Computing and Networks (VECONs) [1] is a promising approach that can be embedded at the network edge to support massive data storage, computing, and sharing close to the vehicles [2]. Constrained by limited resources, current VECONs cannot efficiently satisfy the growing resource demands of vehicular applications [3]. To enable real-time services under large-scale communications, it is promising to conduct predictions and processing through specific virtual spaces, e.g., digital twins (DTs) [4], [5]. Real-world data and experiences, such as traffic patterns and environmental conditions, are captured and integrated into the virtual environment. Within the virtual realm, this data is used to model diverse scenarios and potential outcomes, evaluate the effectiveness of vehicle decisions, or analyze the impact of autonomous vehicle behavior [6], [7].

Using knowledge obtained through virtual analysis, DT facilitates bidirectional data exchange and enhances decision-making that can aid in formulating optimal operational approaches in diverse real-time VECONs [8]. Existing studies have investigated the performance enhancement of edge networks through the assistance of DTs, including network services improvement [9], [10], collaborative driving decisions-making [11], task scheduling and resource utilizations [12]. However, determining how to migrate DTs that handle complex communication and strict latency requirements to minimize service latency while preserving service quality remains a crucial, yet unresolved challenge, particularly for mobile vehicles in dynamic VECONs.

Achieving appropriate DT migration decisions requires addressing the following two challenges. *The first challenge is how to quantify the complex latency of DT communications during migration.* The complex communication relationships within DTs, which have been neglected by existing studies, primarily involve three components: interactions between physical objects and DTs, between DTs themselves, and the information flow between DTs and user applications. [8]. Different types of DTs can enable cooperative management of autonomous vehicle driving [11], [13]. Due to frequent status and data synchronization for DTs, it becomes essential to account for additional communication interactions and data transmissions. This inevitably leads to heightened system intricacy and increased communication costs. Moreover, the simultaneous exchange of information among multiple DTs significantly scales up the communication network. To address these issues, a feature extraction network is meticulously designed to capture interdependent complex communication features, which are then integrated into the system state.

The second challenge is how to make adaptive migration decisions for DTs in dynamic VECONs. It involves a joint optimization of DT migration latency and communications costs to achieve long-term benefits. Traditional heuristic algorithms often struggle to effectively address this challenge due to their static strategies and inability to capture long-

Fangyi Mou is with Hong Kong Baptist University and Beijing Normal University, Zhuhai 519087, China. (E-mail: moufangyi@uic.edu.cn)

Jiong Lou is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China. (E-mail: lj1994@sjtu.edu.cn)

Zhiqing Tang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. (E-mail: zhiqingtang@bnu.edu.cn)

Yuan Wu is with the State Key Lab of IoT for Smart City, University of Macau, SAR Macau 999078, China. (E-mail: yuanwu@um.edu.mo)

Weijia Jia is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China, and also with Guangdong Key Lab of AI and Multi-Modal Data Processing, BNU-HKBU United International College, Zhuhai 519087, China. (E-mail: jiawj@bnu.edu.cn)

Yan Zhang is with the Department of Informatics, University of Oslo, Norway. (E-mail: yanzhang@ieee.org)

Wei Zhao is with Shenzhen University of Advanced Technology, Shenzhen 518055, China. E-mail: weizhao86@outlook.com

(Corresponding author: Zhiqing Tang.)

term awards. Reinforcement learning (RL) is leveraged to optimize strategies and find optimal solutions. However, the interactive environment undergoes frequent changes due to the high-speed mobility of vehicles and variations in DT communications connections. Due to the insufficient state data, it is challenging for RL agents to make effective decisions from limited states to unseen ones. Therefore, warm-start policies [14], [15] from expert demonstrations are employed to improve policy exploration and accelerate the training process for RL agents. We first pre-train the agent based on expert demonstrations and progressively diminish the proportion of these demonstrations during the training process, guiding the RL agent to converge towards the expert region gradually. Then, adaptive DT migration decisions can be made by utilizing a policy gradient RL with warm-start policies.

In this paper, we first model the large-scale communication-aware adaptive DT migration problem in VECONs, aiming to minimize communication latency and migration cost. The features of large-scale interactive connections of DTs, DT mobilities, and the resources of heterogeneous roadside units (RSUs) are fully considered. Feature extraction is performed on these elements. Next, an adaptive DT migration (ADM) algorithm is proposed based on the off-policy actor-critic RL algorithm [16]. Expert demonstrations are utilized as prior knowledge for efficient initialization. The training process of the ADM algorithm is then guided by progressively diminishing demonstrations used as warm-up policies. Finally, simulations based on real-world vehicular mobility traces of Cologne, Germany [17], are conducted to evaluate the performance of the ADM algorithm compared to other migration algorithms. Results show that our ADM algorithm reduces total migration latency by approximately 39% on average compared to baseline algorithms. The main contributions are listed as follows:

- We address the significant system latency caused by large-scale complex communications during DT migrations in VECONs, fully modeling interactions among physical objects, DTs, and user applications. This model comprehensively accounts for communication latency, colocation, and migration costs to enhance the performance of DTs.
- An RL-based ADM algorithm is proposed for DT migration decisions. Features are extracted from large-scale DT communications. Expert demonstrations are then used to assist in feature learning, effectively guiding the RL agent in making adaptive migration decisions.
- Simulations utilizing real-world urban vehicular mobility datasets evaluate the ADM algorithm's performance. Results illustrate that our algorithm outperforms traditional baseline algorithms by roughly 39% on average.

II. Related Work

A. DT in VECONs

Recent advancements in VECONs with DT technologies have addressed challenges associated with leveraging virtualization to facilitate remote execution of vehicle-related tasks by RSUs or base stations. Wang *et al.* [4] define and discuss three kinds of DT-built microservices for mobility services, focusing on driver management, vehicle, and traffic DTs in vehicular edge networks. Zhao *et al.* [3] apply the DT technology to construct virtual replicas of vehicular networks with multiple twin versions. Zheng *et al.* [18] examine the problem of joint network selection and power allocation in DT-assisted networks and use DT to predict the waiting time for decision-making. Zhang *et al.* [19] present a DT-assisted platform for edge computing networks and introduce a novel service framework to address resource dynamics and mobile users. Feng *et al.* [5] simulate the DT network by mapping the traffic situation in the physical road network to the virtual space and transmitting vehicular data interactions based on blockchain technology. Yuan *et al.* [20] utilize DTs to gather real-time data and obtain representations of the physical operating environments. Lu *et al.* [21] propose a DT-assist prediction algorithm for vehicle-to-vehicle pairing to improve task offloading efficiency according to the real-time vehicular network state.

These studies leverage the DT technology to derive insights from physical network data and then employ deep learning methods to optimize overall networks. However, they overlook the importance of optimizing DT performance throughout the entire DT lifecycle, which includes initialization, migration, and updates. This optimization is crucial for enhancing the operational quality of DT edge networks.

B. Service and DT migration

In edge computing, traditional service migration has been widely explored to address various challenges, such as reducing startup latency [22], improving system energy efficiency [23], [24], optimizing overall costs [25]–[27], and satisfying network Quality of Service (QoS) [28]. However, the constant movement of vehicles, diverse resources available at edge nodes, and the requirement for multifaceted vehicular communications present unique challenges for DT migration.

Existing research has begun addressing these challenges, yet notable gaps persist. Sun *et al.* [29] utilize DT to estimate states of edge servers and provide training data for service offloading decisions. Lu *et al.* [30] formulate the edge association problem concerning the dynamic network states and varying network topologies and then decompose the problem into DT placement and DT migration. Most existing studies mainly concentrate on optimizing migration costs and making decisions based on mobility patterns and network dynamics. However, the complex nature of DT communications, especially in large-scale, real-time collaborative environments, presents additional challenges that have not been thoroughly explored. It is crucial to examine the ADM problem considering multi-dimensional and intricate DT communications. Our research differs from existing studies by specifically addressing these gaps.

III. System Model and Problem Formulation

A. System Model

Fig. 1 shows the ADM problem in VECONs. In the physical space, RSU servers $e_1, e_2, ... \in \mathbf{E}$ are placed along the roads,

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3492349

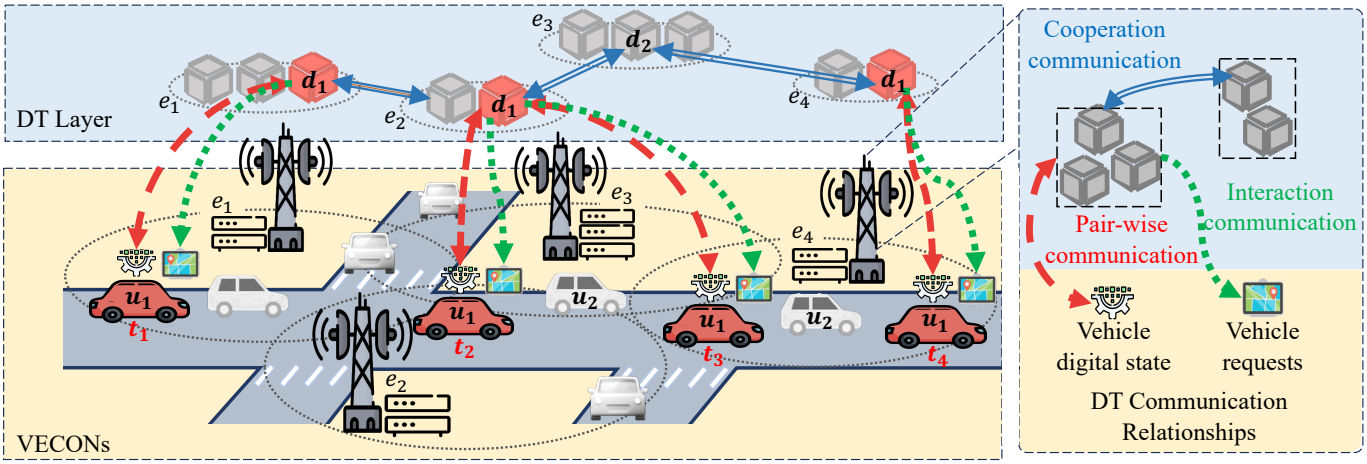


Fig. 1: An illustration of DT migration in VECONs. As the vehicle u_1 continuously moves, the DT d_1 must migrate intelligently on RSU servers $e_1, e_2, \ldots \in \mathbf{E}$ to enable concurrent running. The DT d_1 maintains constant wireless pair-wise communications to acquire updated vehicular digital states. It also returns the processing results of vehicular requests and cooperates with other DTs simultaneously. More detailed explanations are in Section III-A.

allowing moving vehicles to achieve continuous data sharing through multiple RSUs [8], [31]. Each vehicle can transmit requests to servers on RSUs, using data collected by their onboard sensors [32]. The requests are sent in discrete time slots $t_1, t_2, \cdots \in T$ until they are finished. DTs are deployed in parallel with vehicles and modeled within RSUs, using a portion of their resources [33].

As the vehicle u_1 moves from the coverage area of e_1 to e_4 over time, it starts communicating with DT d_1, which is currently hosted on e_1 at time t_1. The vehicle u_1 sends its current state and requests to DT d_1 for processing. This interaction helps u_1 to obtain real-time traffic updates and navigation assistance. As vehicle u_1 continues moving, it approaches RSU e_2. To maintain a high QoS, DT d_1 migrates from RSU e_1 to e_2 at time t_2. This migration involves transferring the DT's state and ongoing vehicular request data to RSU e_2. This handover is critical to reduce latency and ensure seamless service continuity. At time t_3, DT can also connect to other RSUs through multi-hop routing to serve the vehicle u_1 that approaches RSU e_4. During the continuous time interval from t_3 to t_4, vehicle u_1 is near RSU e_4. DT d_1 at e_2 keeps serving u_1 by moving through RSU e_3 to maintain low-latency services directly. Throughout this processing, DT d_1 ensures that u_1 receives continuous updates and processing results, effectively adapting to the vehicle's changing position.

Based on long-term optimizations of migration and communication latencies, DT migration facilitates communication between vehicles and DTs and ensures that vehicular requests are processed with minimal latency. The migration decisions are based on optimizing communication latencies and maintaining service quality, showcasing the dynamic interaction in a VECON environment. The vehicle, RSU, and DT are defined as follows. The main notations are summarized in TABLE I.

Vehicle: All moving vehicles in the system are denoted as a set $\mathbf{U} = \{u_1, u_2, \ldots, u_{|\mathbf{U}|}\}$, where $|\cdot|$ signifies the cardinality of the set. Thus, $|\mathbf{U}|$ represents the total number of vehicles in the moving vehicle set. Each vehicle $u \in \mathbf{U}$ is equipped with a set of attributes represented by $u(t) \triangleq \{\vartheta_u, z_u, o_u, m_u, l_u(t)\}$, which enable it to interact within the system environment. Specifically, ϑ_u is the running period, z_u represents the update state, o_u is the request CPU frequency, m_u is the request memory, and $l_u(t)$ is the location.

RSU: All heterogeneous RSUs in the system are denoted as a set $\mathbf{E} = \{e_1, e_2, \ldots, e_{|\mathbf{E}|}\}$, where $|\mathbf{E}|$ is the total number of RSUs in the VECON network. The properties of a RSU $e \in \mathbf{E}$ are defined as $e(t) \triangleq \{b_e(t), o_e(t), m_e(t), z_e(t), l_e(t)\}$, which support communications between vehicles and DTs. Specifically, $b_e(t)$ represents the remaining bandwidth, o_e denotes the total CPU frequency capacity, $o_e(t)$ is remaining CPU frequency, $m_e(t)$ indicates the remaining memory, $z_e(t)$ is the storage, and $l_e(t)$ denotes the location.

DT: A set of DT is defined as $\mathbf{D} = \{d_1, d_2, \ldots, d_{|\mathbf{D}|}\}$, representing the digital models of the physical vehicles in the environment. The properties of each $d(t)$ are defined as $d(t) \triangleq \{u(t), \ell_d(t), m_d, z_d, \Delta z_d, l_d(t)\}$. Here, $u(t)$ denotes the corresponding vehicle, $\ell_d(t)$ denotes the required computational CPU frequency, m_d represents the DT memory capacity, z_d indicates the size of DT, Δz_d is the size of synchronized data, and $l_d(t)$ represents the location.

B. Cost

To ensure a satisfactory QoS, DTs should be dynamically migrated among RSUs as vehicles move, comprehensively considering communication latency, colocation cost, and migration latency.

Communications latency: The communication model consists of three parts [8]: the DT cooperation communication for data synchronization among DTs, the DT interaction communication with user applications, and the DT pair-wise communication for data exchange with the respective vehicle.

DT cooperation communication: Based on the direct trust interactions among RSUs [34], the communication of DTs between any two RSUs e_i and e_j, denoted as $L(e_i, e_j)$,

TABLE I: Notations.

$\mathbf{U}$	Mobile vehicle set
u_i	$u^{i_{th}}$ mobile vehicle ($u \in \mathbf{U}, i \in \lvert\mathbf{U}\rvert$)
z_{u_i}	Update size of vehicle u_i
o_{u_i}	CPU request of vehicle u_i
m_{u_i}	Memory request of vehicle u_i
$l_{u_i}(t)$	Location of vehicle u_i at time t
$\mathbf{E}$	RSU set
e_k	$e^{k_{th}}$ RSU ($e \in \mathbf{E}, k \in \lvert\mathbf{E}\rvert$)
$b_{e_k}(t)$	Bandwidth of RSU e_k at time t
$o_{e_k}(t)$	CPU resource of RSU e_k at time t
$m_{e_k}(t)$	Memory resource of RSU e_k at time t
$z_{e_k}(t)$	Storage capacity of RSU e_k at time t
$l_{e_k}(t)$	Location of RSU e_k at time t
$\mathbf{D}$	DT set
d_j	$d^{j_{th}}$ DT ($d \in \mathbf{D}, j \in \lvert\mathbf{D}\rvert$)
$b_{d_j}(t)$	Allocated bandwidth for DT d_j at time t
$\ell_{d_j}(t)$	CPU request of DT d_j at time t
z_{d_j}	Size of DT d_j
Δz_{d_j}	Synchronized size of DT d_j
$f(d_i, d_j)$	Cooperation frequency between DT d_i and d_j
$f(d_i, e_k)$	Interaction frequency between DT d_i and closest RSU e_k to vehicle application
$f(d_i, u_i)$	Pair-wise frequency between DT d_i and its vehicle u_i
$\mathcal{L}^{com}(t)$	Communication latency for DT
$\mathcal{L}^{colo}(t)$	Colocation cost for DT
$\mathcal{L}^{mig}(t)$	Migration latency for DT
$\mathcal{L}(t)$	Total latency for DT

signifies the potential for collaboration. The interaction rate, represented as $f(d_i, d_j) \geq 0$, signifies the direct trust relationship between RSUs. This direct trust interaction fosters an environment that promotes seamless communication and coordinated task execution among DTs. $\delta_{d,e}(t)$ is a binary variable to indicate if the DT d is on the RSU e at time t, which is defined as follows:

$$\delta_{d,e}(t) \triangleq \begin{cases} 1, & \exists d(t) \in \mathbf{D}, e \in \mathbf{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the cooperation communication latency over all kinds of DTs at time t is denoted as:

$$\mathcal{L}^{coop}(t) = \sum_{e_i, e_j \in \mathbf{E}} \sum_{d_i, d_j \in \mathbf{D}} f(d_i, d_j) L(e_i, e_j) \delta_{d_i, e_i}(t) \delta_{d_j, e_j}(t). \tag{2}$$

DT interaction communication: The DT d_i and the user application e_k are processed on separate RSUs, denoted as e_i and e_k, respectively. The communicative relationship between these entities is denoted as $L(e_i, e_k)$, representing the connection between RSUs. The interaction rate between the DT and user application is defined as $f(d_i, e_k) \geq 0$ [35]. The interaction communication latency is calculated as follows:

$$\mathcal{L}^{inter}(t) = \sum_{e_i, e_k \in \mathbf{E}} \sum_{d_i \in \mathbf{D}} f(d_i, e_k) L(e_i, e_k) \delta_{d_i, e_i}(t). \tag{3}$$

DT pair-wise communication: A DT d_i communicates with its relative vehicle u_i based on the data exchange rate $f(d_i, u_i)$. The wireless uplink transmission rate is influenced by various factors, including path loss, modulation schemes, etc. [26]. The wireless transmission rate from vehicles to their DTs is formulated as:

$$\xi(t) = b_{d_i}(t) \log_2 \left(1 + \frac{p_{u_i} |h_{u_i, d_i}|^2}{b_{d_i}(t)\sigma}\right), \tag{4}$$

where $b_{d_i}(t)$ is the allocated bandwidth for DT d_i at time slot t, p_{u_i} is the transmission power of vehicle u_i, h_{u_i,d_i} is the channel gain between the vehicle u_i and its corresponding DT d_i, and σ is the power spectral density of the Gaussian white noise. The update for vehicle u_i to DT d_i is obtained as:

$$\mathcal{L}^{up}(t) = \frac{z_{u_i}}{\xi(t)}, \tag{5}$$

where z_{u_i} denotes the data size that vehicle u_i transfers.

The communication between a DT and the respective vehicle is bidirectional. The downlink communication primarily depends on the hop distance along the shortest path and the size of the synchronized data [36], [37]. The downlink communication latency is defined as:

$$\mathcal{L}^{down}(t) = \frac{\Delta z_{d_i}}{b_{e_i}(t)} + \alpha(t) l(u_i, e_i), \tag{6}$$

where $\alpha(t)$ is a positive coefficient and $l(u_i, e_i)$ is the hop distance between the vehicle u_i and the location e_i of the corresponding DT. The total pair-wise communication of the system is obtained as follows:

$$\mathcal{L}^{pair}(t) = \sum_{e_i \in \mathbf{E}} \sum_{d_i \in \mathbf{D}} f(d_i, u_i) \delta_{d_i, e_i}(t) \left(\mathcal{L}^{up}(t) + \mathcal{L}^{down}(t)\right). \tag{7}$$

Various DT instances engage in concurrent and independent communications, where all three communications coincide within the overall system. Therefore, communication latency in VECONs at time t can be defined as:

$$\mathcal{L}^{com}(t) = \max\left(\mathcal{L}^{coop}(t), \mathcal{L}^{inter}(t), \mathcal{L}^{pair}(t)\right). \tag{8}$$

Colocation cost: The colocation cost is associated with hosting DT data and computational resources for DT operation. While DT is migrating, resource contention may arise from CPU, memory, or network usage among DTs on the same RSU. The required CPU cycles of the DT d_i are defined as $\ell_{d_i}(t) = z_{d_i}\eta$, where η represents the processing density of the DT. The workload of the serving RSU is denoted as $w_{e_i}(t) = \sum_{d_i \in \mathbf{D}} \ell_{d_i}(t) \delta_{d_i, e_i}(t)$. The actual colocation cost for the DT d_i is obtained as [38]:

$$\mathcal{L}^{colo}_{d_i}(t) = \frac{\ell_{d_i}(t)}{w_{e_i}(t) + \ell_{d_i}(t)} o_{e_i}. \tag{9}$$

The total colocation cost for VECONs is defined as follows:

$$\mathcal{L}^{colo}(t) = \sum_{d_i \in \mathbf{D}} \sum_{e_i \in \mathbf{E}} \mathcal{L}^{colo}_{d_i}(t) \delta_{d_i, e_i}(t). \tag{10}$$

DT Migration latency: Low migration latency is critical for ensuring the seamless transition of DTs between different RSUs. When DT migration is required, real-time processing modules and computations must be transferred to target RSUs. This requires optimizing the network latencies, including the DT transmission, propagation, processing, and queueing latencies. Due to the ample bandwidth available in the network, processing and queuing latencies are relatively small and can be disregarded. The transmission latency of DT is defined as $\mathcal{L}^{tr}_{d_i}(t) = \frac{z_{d_i}}{\upsilon_t}$, where υ_t stands for the transmission rate. For DT migrating from node e_i to node e_j, the propagation latency is measured by the hop distance that is denoted as

$\mathcal{L}_{d_i}^{pr}(t) = \beta(t) l(e_i, e_j)$, where $\beta(t)$ is a positive coefficient, $l(e_i, e_j)$ refers to the shortest path from the source DT to the target DT. The DT migration latency is obtained as [36], [38]:

$$\mathcal{L}_{d_i}^{mig}(t) = \begin{cases} 0, & \text{if } l(e_i, e_j) = 0, \\ \mathcal{L}_{d_i}^{tr}(t) + \mathcal{L}_{d_i}^{pr}(t), & \text{if } l(e_i, e_j) \neq 0. \end{cases} \tag{11}$$

The migration latency of VECONs can then be calculated as:

$$\mathcal{L}^{mig}(t) = \sum_{d_i \in \mathbf{D}} \mathcal{L}_{d_i}^{mig}(t). \tag{12}$$

C. Problem Formulation

A comprehensive cost model including the above three types of costs is considered. The total cost at time t is defined as:

$$\mathcal{L}(t) = \mathcal{L}^{com}(t) + \mathcal{L}^{colo}(t) + \mathcal{L}^{mig}(t). \tag{13}$$

The main object is to minimize the total cost over time to guarantee the overall system QoS. The calculation of the total cost is processed across multiple RSUs, collecting DT colocation cost on each RSU, DT migration latencies between RSUs, and DT communication latencies between different RSUs, including DT state updates and synchronization.

Constraints: With the high mobility and handovers between vehicles and DTs, the dynamic environment leads to changing network connections. Therefore, the DT has to meet the latency requirements to realize real-time processing. The actual communication latency must be within the latency constraint for each time t, which can be denoted as:

$$\mathcal{L}^{pair}(t) < W_d, \forall d \in \mathbf{D}, t \in T, \tag{14}$$

where W_d denotes the latency constraint.

A binary variable $a_e(t) \in \{0, 1\}$ is defined to indicate whether RSU e is selected at time t. If the RSU e is selected at time t, then $a_e(t)$ is set to 1. Otherwise, $a_e(t) = 0$. It is imperative to guarantee the allocation of at least one RSU to a DT, as constrained by:

$$\sum_{e \in \mathbf{E}} a_e(t) = 1, \forall t \in T. \tag{15}$$

It should be noted that at any given time t, one DT must be actively running on an RSU, which is represented as:

$$\sum_{e \in \mathbf{E}} \delta_{d,e}(t) = 1, \forall d \in \mathbf{D}, t \in T. \tag{16}$$

To ensure the performance of RSUs, the maximum number of DTs running on each RSU is limited with a number N_e:

$$\sum_{d \in \mathbf{D}} \delta_{d,e}(t) \leq N_e, \forall e \in \mathbf{E}, t \in T. \tag{17}$$

The migration of DTs is affected by the available resources on RSUs. The limits of storage and memory resources of each RSU are defined as follows:

$$\sum_{d \in \mathbf{D}} z_d \delta_{d,e}(t) \leq z_e, \forall e \in \mathbf{E}, t \in T, \tag{18}$$

$$\sum_{d \in \mathbf{D}} m_d \delta_{d,e}(t) \leq m_e, \forall e \in \mathbf{E}, t \in T. \tag{19}$$

Problem: (DT Migration Problem) For each time t, our goal is to optimize DT migrations to minimize the overall system cost. This optimization takes into account the high-speed movement of vehicles, complex communication relationships, as well as constraints related to communication latencies and resource limitations of RSUs, which is formulated as:

Problem 1. $\min \mathcal{L} = \sum_{t \in T} \mathcal{L}(t)$,

$$\begin{aligned} \text{s.t.} \quad & \text{Eqs. (14), (15), (16), (17), (18), (19),} \\ & \delta_{d,e}(t) \in \{0, 1\}, \forall d \in \mathbf{D}, \forall e \in \mathbf{E}, \forall t \in T, \\ & a_e(t) \in \{0, 1\}, \forall e \in \mathbf{E}, \forall t \in T. \end{aligned}$$

The DT Migration problem is a binary non-linear programming problem. Eq. (8) includes three communication relationships and varies among DT interactions. The core challenge lies in the exponential rise in DT migration complexity when evaluating overall system performance.

Proposition 1. The DT migration problem is NP-Hard.

Proof. The DT migration problem can be polynomial-time reduced to the set cover problem (SCP) ideally [39]. Given a graph with vertices and edges, each vertex represents a potential facility, and each edge represents a customer. Facilities are assigned a cost of 1. The distances between customers and facilities are 1 if edges connect them and 0 otherwise. Each edge connected to different facilities has an assigned weight. The service cost is the sum of the distances multiplied by the corresponding edge weights. The SCP is NP-hard. Therefore, the solution to the DT migration problem is NP-hardness. □

The objective of the problem is to make continuous DT migration decisions under complex communication and dynamic VECONs to maximize the cumulative long-term reward. The effectiveness of heuristic solutions is closely linked to the quality of the predefined rules. Meta-heuristic algorithms efficiently explore vast problem spaces and provide high-quality solutions, though their computational time can be considerable. The first-order transition probability of the system state remains quasi-static over an extended period [40] when modeled as a Markov Decision Process (MDP). RL is ideal for MDPs, offering a structure for iteratively and interactively learning optimal policies. Additionally, RL algorithms utilize the value function based on immediate rewards to secure long-term benefits. Therefore, RL can address the complex and large-scale DT migration issue.

In RL, the centralized network consolidates all RSU data into one edge node, achieving precise reward computation and enabling scheduling decisions based on the global state. As the network scales up, this centralized approach may become burdensome, resulting in additional latency. Although the estimated cumulative reward in a distributed network may show greater variance, the advantages of reduced data collection overhead and faster communication significantly outweigh this concern. [41].

IV. Proposed Algorithms

A. Algorithm Settings

We formulate the ADM problem as an infinite-horizon MDP, which is represented by the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma\}$,

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3492349

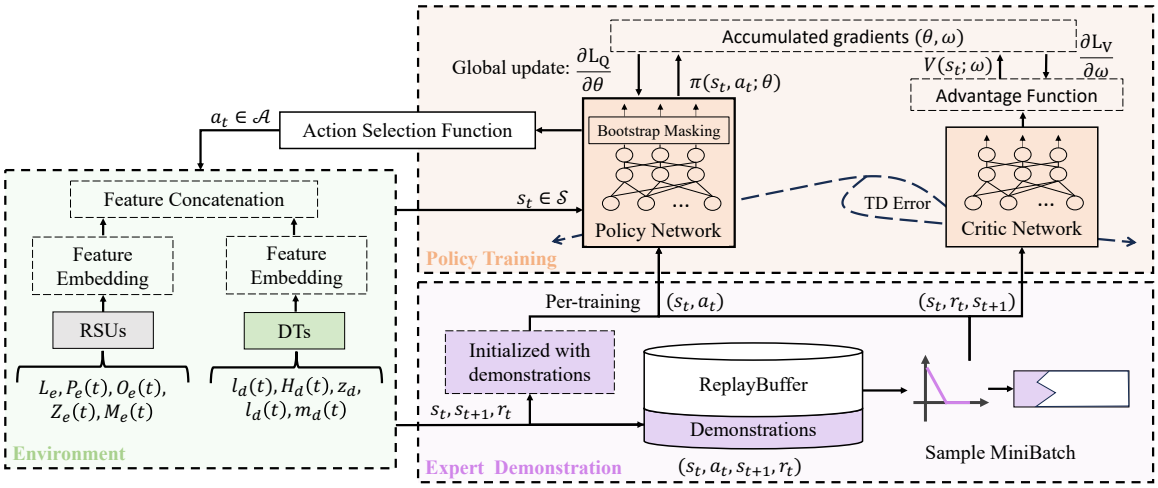


Fig. 2: Algorithm structure.

where $\mathcal{S}$ denotes a finite state set, $\mathcal{A}$ represents a finite action set, $\mathcal{P}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition density, ρ_0 stands for the initial state distribution, $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is a discount factor.

State: As shown in Fig. 2, the state is composed of properties of RSUs and DTs, which can be denoted as:

$$s_t = \{s_t^e, s_t^d\} \in \mathcal{S}, \tag{20}$$

where the state of RSUs is described as $s_t^e = \{L_e, P_e(t), O_e(t), Z_e(t), M_e(t)\}$. It includes the locations of RSUs $L_e = \{l_1, l_2, \cdots, l_{|\mathbf{E}|}\}$, the communications with cooperating DTs $P_e(t) = \{\mathcal{L}_{\mathbf{D},1}^{coop}, \mathcal{L}_{\mathbf{D},2}^{coop}, \cdots, \mathcal{L}_{\mathbf{D},|\mathbf{E}|}^{coop}\}$, the available CPU frequency $O_e(t) = \{o_1(t), o_2(t), \cdots, o_{|\mathbf{E}|}(t)\}$, the remaining storage $Z_e(t) = \{z_1(t), z_2(t), \cdots, z_{|\mathbf{E}|}(t)\}$, and the remaining memory $M_e(t) = \{m_1(t), m_2(t), \cdots, m_{|\mathbf{E}|}(t)\}$.

Moreover, the state of current processing DT is denoted as $s_t^d = \{l_d(t), H_d(t), z_d, \ell_d(t), m_d(t)\}$, where $l_d(t)$ is the location of DT, $H_d(t) = \{L(d, e_1), L(d, e_2), \cdots, L(d, e_{|\mathbf{E}|})\}$ is the distance between DT d and each RSU, z_d indicates the size of the DT, $\ell_d(t)$ is the CPU request, and $m_d(t)$ denotes the memory request of the DT d. Based on the dynamic state, a feature extraction and concatenation network is used to capture the intricate relationships in DT communication and migration for subsequent decision-making processes.

Action: An action $a_t \in \mathbf{E}$ indicates that the DT can migrate to any satisfied RSU e, where the RSU set $\mathbf{E}$ combines the action space.

Reward: The agent navigates the environment repetitively, learning how to optimize to reach its goal. Since the goal is to minimize the overall cost in ADM, the reward can be denoted as $r_t = -\mathcal{L}(t)$.

Policy network with feature extraction: In an MDP, an agent determines a sequence of parameterized policies $\pi(a|s; \theta_1), \ldots, \pi(a|s; \theta_T)$ to formalize the distribution of state-action trajectories $\tau = \{s_1, a_1, \ldots, s_T, a_T\}$.

To enhance decision-making, a feature network $\phi(s; \theta_f)$ is used to extract features from the state $s \in \mathcal{S}$. The policy is then defined on these features:

$$\pi(a|s; \theta) = \pi(a|\phi(s; \theta_f); \theta_T), \tag{21}$$

where θ_f is the parameters of the feature network.

The state distribution is defined as $d^\pi(s)$ when following policy π. The long-term discounted average return of the ADM agent is denoted as:

$$\eta(\pi) = \mathbb{E}_{s \sim d^\pi, a \sim \pi(a|\phi(s;\theta_f))} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]. \tag{22}$$

The Problem 1 can be transformed into a policy optimization problem, which is to obtain the optimal policy π^*:

$$\pi^* = \arg\max_{\pi} \eta(\pi). \tag{23}$$

B. Algorithm Overview

The ADM algorithm is introduced in Algorithm 1. All running DTs are in a priority queue $\mathbb{Q}^D$ saved with running duration time and DT index. In Lines 2 - 6, the current processing DTs are obtained, and their actions are determined by policy. Then, according to the problem constraints, in Lines 8-16, whether the decision satisfies DT communication constraints is checked. If not, a new finding that meets the above rules is resampled. The computation time is calculated, and the DT queue is updated in Line 18. Next, the processing queue is updated by recurring the tuple value in the DT queue in Lines 21 - 25. The environment is constantly updated.

C. Expert Demonstration

Due to the complexity and sparsity of the state space in VECONs, the agent encounters challenges in exploration and training instability. A substantial number of trajectories are needed for the agent to find optimal strategies. To help the agent explore the state and action space more purposefully and reduce unnecessary exploration, pre-training with expert demonstrations is employed to initialize the actor network.

The optimal policy resides within a particular region around the expert policy. Once the agent enters this region, its optimization is solely influenced by environmental interactions rather than expert demonstrations [42]. As shown in Fig. 2, we also merge expert trajectories with training trajectories,

enabling the agent to gradually converge towards regions of expertise under the guidance of expert features to improve sample efficiency.

The state-action value function $Q^{\pi}(s,a)$ is defined as:

$$Q^{\pi}(s,a) \equiv E\left[\sum_{t=1}^{\infty}(r(s_t,a_t)-\eta(\pi))|s_0=s,a_0=a,\pi\right]. \tag{24}$$

This calculation starts from an initial state-action pair $\{s_0,a_0\}$ and follows policy π for subsequent actions. The definitions of state-value function $V^{\pi}(s,\pi)$ and the advantage function $A^{\pi}(s,a)$ are denoted as:

$$V^{\pi}(s,\pi) \equiv E_{a\sim\pi(a|s)}\left[Q^{\pi}(s,a)\right], \tag{25}$$

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s,\pi). \tag{26}$$

The action-value function satisfies the well-known recurrence equation, which is defined as:

$$Q^{\pi}(s,a) = r(s,a) - \eta(\pi) + \gamma E_{s'\sim\mathcal{P}_{sa}}\left[V^{\pi}(s',\pi)\right], \tag{27}$$

where $s' \sim \mathcal{P}_{sa} = \rho(s'|s,a), s' \in \mathcal{S}$ stands for the possible transition probabilities of next states, $V^{\pi}(s',\pi)$ is the next state-value. Suppose there exists θ^E for expert policy $\pi(a|s;\theta^E)$ that yields high rewards, which the straight-forward constraint expressed as follows:

$$Q^{\pi^E}(s,a) = E\left[\sum_{t=1}^{T}\gamma^t r(t;\theta^E)\right] \geq Q^{\pi}(s,a). \tag{28}$$

The objective is to minimize the difference against the best fixed stationary policy, which can be defined as:

$$\eta^{\circ} \triangleq \min_{\pi}\left(\sup_{\theta} Q^{\pi} - Q^{\pi^E}\right). \tag{29}$$

Equivalently, we aim to seek an optimal solution π^* that satisfies $Q^{\pi^*} \leq Q^{\pi^E} + \eta^{\circ}$ with η° being always nonpositive, which means the optimal policy outperforms the expert policy.

Intuitively, for a specific state s, an action a sampled from the deterministic expert policy $\pi^E(a|s)$ is expected to yield a higher return compared to the policy π, which is defined as:

$$V^{\pi}(s,\pi^E) \equiv E_{a\sim\pi^E(a|s)}\left[Q^{\pi}(s,a)\right]. \tag{30}$$

The policy π^E selects actions differently for the states visited under policy π. The optimal policy is guided toward selecting actions with a potentially high expected return.

Lemma 1. *For any policies π and expert policy π^E,*

$$\eta(\pi^E) - \eta(\pi) = E_{s\sim d^{\pi^E}}\left[V^{\pi}(s,\pi^E) - V^{\pi}(s,\pi)\right]. \tag{31}$$

Proof. We consider stationary policies where state transitions maintain the same distribution. Formally, if $s \sim d^{\pi^E}$, $a \sim \pi^E(a|s)$, and $s' \sim P_{sa}$, then $s' \sim d^{\pi^E}$. With this and Eq. (27), Eq. (30), we have:

$$\begin{aligned}
&E_{s\sim d^{\pi^E}}\left[V^{\pi}(s,\pi^E)\right] = E_{s\sim d^{\pi^E},a\sim\pi^E}\left[Q^{\pi}(s,a)\right] \\
&= E_{s\sim d^{\pi^E},a\sim\pi^E}\left[r(s,a) - \eta(\pi) + E_{s'\sim P_{sa}}\left[V^{\pi}(s',\pi)\right]\right] \\
&= E_{s\sim d^{\pi^E},a\sim\pi^E}\left[r(s,a) - \eta(\pi)\right] + E_{s\sim d^{\pi^E}}\left[V^{\pi}(s,\pi)\right] \\
&= \eta(\pi^E) - \eta(\pi) + E_{s\sim d^{\pi^E}}\left[V^{\pi}(s,\pi)\right].
\end{aligned} \tag{32}$$

Algorithm 1: ADM Algorithm

Input : $\mathbb{Q}^D, \pi_\theta, j=0$
Output: A_τ

for $t \in \{1,2,3,\cdots\}$ **do**
  Get the processing DT set $\mathbb{Q}^D(t)$
  for $i = 1,2,\cdots,|\mathbb{Q}^D(t)|$ **do**
    Get DT processing property $d_i(t)$
    Select the node $e = a_t \sim \pi_\theta$
    Get the corresponding vehicle $u_i(t)$
    Compute communication latency $T^{com}(t)$
    // Action Validation
    while $z_e(t) + z_{d_i} > z_e$ *or* $m_e(t) + \ell_{d_i}(t) > m_e$ *or* $T^{pair}(t) > W_{d_i}$ **do**
      Resample the node e
      $j \longleftarrow j+1$
      if $j > |\mathcal{A}|$ **then**
        Select the node $e = a \in \mathcal{A}$
        break
      end if
    end while
    Compute Computation time $T^{colo}(t)$
    $\mathbb{Q}^D(t) \longleftarrow (T^{colo}(t) + \vartheta_{u_i}, d_i)$
    Update the environment
  end for
  while $\mathbb{Q}^D(t) \neq \emptyset$ **do**
    Get the sub-item (t_d, d) from $\mathbb{Q}^D(t)$
    if $t_d > t$ **then**
      $\mathbb{Q}^D(t) \longleftarrow (t_d, d)$
    end if
  end while
  Update the environment
end for
Return A_τ
end

Rearranging, the result follows. □

Lemma 1 shows that policy optimization relies on estimating state-action value functions until they converge to their optimal values. When the algorithm incorporates expert demonstrations, estimating the target $\eta(\pi)$ requires sampling π^E to generate expected trajectories.

The expected policy advantage objective, as derived from Eqs. (26) and (31), is denoted as follows:

$$\begin{aligned}
\mathbb{A}_{\theta^E}(\theta^*) &= E_{s\sim d^{\pi^*}}\left[V^{\pi^E}(s,\pi^*) - V^{\pi^E}(s,\pi^E)\right] \\
&= E_{s\sim d^{\pi^*}}\left[\sum_{t=0}^{\infty}\gamma^t\left[r(s_t,a_t) + \gamma V^{\pi^E}(s_{t+1}) - V^{\pi^E}(s_t)\right]\right] \\
&= E_{s\sim d^{\pi^*}}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi^E}(s_t,a_t)\right] \\
&= \sum_{t=0}^{\infty}\gamma^t E_{s_t\sim d^{\pi^*}} E_{a_t\sim\pi^*(\cdot|s_t)}\left[A^{\pi^E}(s_t,a_t)\right].
\end{aligned} \tag{33}$$

Thus, our goal is to determine a policy π^* that satisfies Lemma 1 while ensuring a monotonic increase in policy performance.

The proof of Lemma. 1 demonstrates that the difference in policy performance can be decomposed into the summation of per-timestep value function estimators. An optimal strategy can direct the state towards more advantageous values. Therefore, the objective of the training phase is to learn to emulate the expert demonstrations with a value function that adheres to the Bellman equation and can be updated through TD learning once the agent begins interacting with the environment.

Algorithm 2: Training of ADM Algorithm

Input : Initialize τ_E, mask distribution $\mathcal{M}$, parameters of policy network θ, value network ω; Define $\tau_{RL} = \emptyset$, pre-train step k.

Output: π^*

1 **for** *pre-train steps* $t \in \{1, 2, 3, \cdots, k\}$ **do**
2 Sample a set of trajectories τ_E from π^E
3 Compute the policy function by Eq. (35)
4 Update the policy parameter by Eq. (39)
5 Compute the value loss by Eq. (36)
6 Compute the total loss on Eq. (37)
7 $\theta_{t+1} \longleftarrow \theta_t$
8 $\omega_{t+1} \longleftarrow \omega_t$
9 **end for**
10 **for** *online training* $t \in \{1, 2, \ldots\}$ **do**
11 Call Algorithm 1 with policy $a \sim \pi_\theta$
12 Store trajectory $\tau_t = (s_t, a_t, s_{t+1}, r_t)$ into τ_{RL}, overwriting trajectory set if exceeded capacity
13 Sample mini-batch $\tau = \varepsilon\tau_{RL} \cup (1-\varepsilon)\tau_E$
14 Sample bootstrap mask $m_t \sim \mathcal{M}$
15 Compute action-value function $Q^{\pi_\theta}(\tau_t) + m_t$
16 Compute policy loss function by Eq. (35)
17 Update the policy parameter Eq. (39)
18 Compute value loss function by Eq. (36)
19 Update loss function parameter on Eq. (37)
20 $\theta_{t+1} \longleftarrow \theta_t$
21 $\omega_{t+1} \longleftarrow \omega_t$
22 $s_{t+1} \longleftarrow s_t$
23 **if** *done* **then**
24 Break
25 **end if**
26 **end for**
27 Return π^*
28 **end**

D. Policy Training

The objective is to minimize the norm of the Eq. (29) to find the optimal policy $\pi^*(\theta)$. We start by considering the self-generated trajectories and estimate the actor network policy gradient that is denoted as follows:

$$g = E_{s_t \sim d^\pi, a_t \sim \pi(\cdot|s_t)} \left[\nabla_\theta log\pi(a_t|s_t) \cdot A^\pi(s_t, a_t) \right]. \quad (34)$$

Eq. (34) is obtained by differentiating the actor loss function $L_Q(\theta)$ of policy gradient, which is denoted as:

$$L_Q(\theta) = -E_{\tau \sim \pi_\theta} \left[log\pi(a_t|s_t) A^\pi_\theta(s_t, a_t) \right]. \quad (35)$$

The loss function of the critic network, parameterized by ω, is represented as:

$$L_V(\omega) = E_{\tau \sim \pi_\theta} \left[r_t + \gamma^t V^\pi_\omega(s_{t+1}) - V^\pi_\omega(s_t) \right]. \quad (36)$$

Thus, the overall loss function is denoted as:

$$L_{ADM} = L_Q + \frac{1}{2} L_V. \quad (37)$$

To incorporate with expert demonstration, importance sampling ratio $w^e = \frac{\pi_\theta(a_t^E|s_t^E)}{\pi^E(a_t^E|s_t^E)}$ is used to improve the target policy π_θ based on expert trajectories. Thus, the corresponding policy gradient of off-policy actor-critic algorithm is estimated as:

$$\begin{aligned} \hat{g} &= \nabla_\theta(\eta(\pi) - \eta^E) \\ &= -\nabla_\theta \sum_{t=0}^{\infty} \gamma^t E_{s_t \sim d^\pi, a_t \sim \pi(\cdot|s_t)} \left[w^e A^{\pi^E}(s_t, a_t) \right]. \end{aligned} \quad (38)$$

At the beginning, the policy $\pi(\theta)$ is randomly initialized. To enhance the ADM agent in obtaining better estimate policy gradients, the expert policy π^E is added as follows:

$$g_\pi = g + \lambda \hat{g}, \quad (39)$$

where λ is a constant weighting parameter.

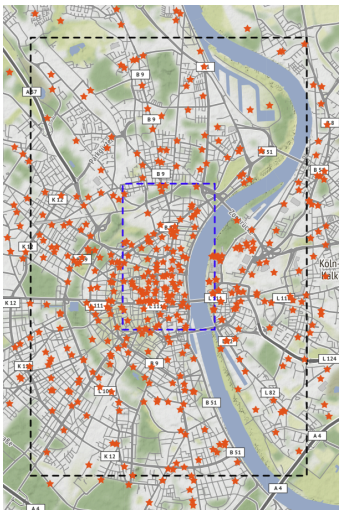
To promote deep exploration, a bootstrap mask is applied during the update of the actor network to mitigate overfitting to expert demonstrations. We introduce bootstrap mask $\mathcal{M}_t[n]$ for each estimated value function $Q_n(s, a)$. The mask is defined as $\mathcal{M}_t[n] \sim \psi(t) = \{0, \kappa\iota_{a_t}(n)\}, \forall t \in T, n \in |\mathcal{A}|$, where $\mathcal{M}_t[n]$ represents the mask value at dimension n in time step t. The distribution $\psi(t)$ takes a value κ on the selected action a_t. $\iota_{a_t}(n)$ is an indicator vector with a dimensionality matching that of the action space. It takes a value of 1 at the position corresponding to the selected action and 0 otherwise. The masking distribution $\mathcal{M}$ is responsible for generating each $m_t \in \mathcal{M}(t)$. This mechanism facilitates training the model to enhance exploratory behavior by introducing noise and uncertainty, thereby mitigating bias and overfitting that may arise from the influence of actual expert trajectories.

The training process of the ADM algorithm primarily involves updating the network weights as shown in Fig. 2, which is detailed in Algorithm 2. Lines 2 - 8 show that the policy $\pi(\theta)$ is pre-trained using the expert trajectories τ_E before interacting with the environment. The pre-training stage estimates the policy gradient g_π from expert demonstrations and updates the network loss. Next, the agent interacts with the environment and stores transitions τ_{RL}. In Lines 13 - 15, training batches τ are sampled from mixed replay buffers, gradually reducing the proportion of expert demonstrations. A bootstrap mask is applied to the policy network to mitigate overfitting. Then, parameters in both actor and critic networks are optimized and updated in Lines 16 - 22. The training process continues until convergence and the optimal policy π^* is sought.

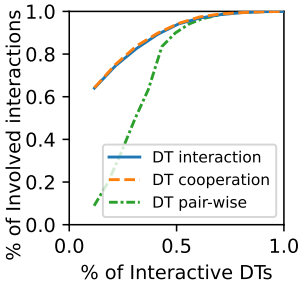
E. Computational Complexity Analysis

The ADM algorithm primarily involves two key operations: Algorithm 1 and Algorithm 2. In Algorithm 1, the computation of the system state Eq. (20) has a computational complexity

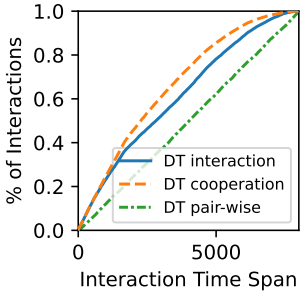
of $O(|\mathbf{E}|)$. The time complexity of the policy network is predominantly dependent on the network size, which can be treated as a constant O_t. Then, the computational complexity for action selection is $O(O_t)$. Algorithm 2 focuses on the update of network weights, as illustrated in Fig. II. This update can be assessed regarding floating-point operations (FLOPs) [25]. Denoting input and output dimensions of j-th linear layer as D_j^{in} and D_j^{out}, respectively, the FLOPs for the two layers involved in the feature extraction process are $2(D_1^{in}-1)D_1^{out}$ and $2(D_2^{in}-1)D_2^{out}$, respectively. The non-linear activation functions, e.g., ReLU and Softmax, can be excluded from FLOP counts as negligible in the overall computational time. It should be noted that the complexity of the training phase does not significantly impact the computational complexity of decision-making within the network, treated as polynomial time. These steps are executed sequentially, completing within polynomial time. Moreover, our experiments also demonstrate that the execution time of the ADM algorithm is acceptable, as shown in Section V.



(a) Mobility areas and edge nodes.



(b) CDF of DT interactions.



(c) CDF of remaining interactions.

Fig. 3: (a) The outer areas of Cologne, Germany (9.5 km $\times$ 9.5 km bounded by the coordinate pairs $[6.914, 50.902]$ and $[6.999, 50.987]$) as a large-scale dataset, the inner areas (3 km $\times$ 3 km bounded by the coordinate pairs $[6.924, 50.930]$ and $[6.971, 50.959]$) as a small-scale dataset. Orange stars refer to the location of base stations, with 448 base stations in the large-scale range and 194 base stations in the smaller one. (b,c) Distributions of DT communications.

V. Numerical Results

A. Simulation Scenario

Dataset and settings: The vehicular mobility trace of the city of Cologne, Germany [17] is used. Each edge node is equipped with an RSU server, which comprises multiple CPU cores with computational capacities o_e vary from 128 GHz, 256GHz, to 512GHz (i.e., different numbers of 16-core, servers with 2GHz for each core) [26]. The distribution coordinates of edge nodes in Cologne are obtained through web crawling techniques from the webpage [43], as depicted in Fig. 3a. The coverage radius of each RSU reaches up to 500 meters according to the C-V2X standard [44], [45].

To emulate DT direct trust interactions and frequencies within the available communication range of moving vehicles [34], scale-free interactive graphs are synthesized based on interaction graph distributions reported in [46]. Cumulative Distribution Functions (CDFs) of synthetically generated frequencies of DT interactions are presented in Fig. 3. As the number of arriving mobile vehicles fluctuates across different time slots, DT interactions exhibit significant temporal unevenness and task concentration, as shown in Fig. 3b, reflecting the varying frequencies of different types of DT interactions. This variability reflects the diverse traffic patterns seen in VECONs at various times. Fig. 3c illustrates the cumulative distribution of interactions among DTs over the observed period, showing a more balanced distribution when considered over longer time frames. This long-term perspective enhances data analysis and forecasting, providing a deeper understanding of systemic behaviors and trends in DT interactions.

The distances between two DTs are calculated by haversine distance based on their geographic coordinates. The Simulation of Urban Mobility (SUMO) package of Cologne [47] is used to obtain reference locations to measure the hop distance during DT migration. The positive coefficient β uniformly distributed in $[1.0, 3.0]$ s/hop [26] as the fluctuate empirical values to reflect the propagation conditions are not uniform across all hops. Transmission power p_u and noise power σ of wireless access are set to $0.2W$ and $1.07 \times 10^{-21}W$, respectively [48]. The channel gains h_{u_i,d_k} is calculated based on the distance between the vehicle and RSU with a path loss factor set to 6. The network bisection bandwidth of wired communication v is set to 1Gbps, and the bandwidth of backhaul communication network b_{back} is set to 500Mbps. The positive coefficient α for relatively stable backhaul communication latency is specified as 0.02 s/hop [37] by using the average value of the empirical α coefficient.

The size of DT is uniformly distributed within $[5, 100]$ MB, mirroring the variety of real-world data sizes. This range aligns with actual end-to-end vehicle data transmissions [49], where RSUs handle time-series information (such as position, velocity, and acceleration) that spans from basic telemetry data to complex diagnostics data. The required CPU cycles are uniformly distributed in $[200, 5000]$ cycles/bit [37]. These ranges encompass various tasks, from high-resource-demanding activities such as autonomous assistance navigation to micro instances like road sensor data transmission.

Simulator: The simulation of VECONs is implemented in Python, involving classes of RSUs, vehicles, DTs, interactive networks, and the scheduler. The details are as follows:

1) RSUs. Each RSU includes its ID, resource capacities, bandwidth, geographic coordinates, and running DT list.

2) Vehicles. Each vehicle contains its ID, arrival time, resource requests, running period, geographical coordinates, group ID, and request RSUs.
3) DT. The DT class includes the DT ID, size, resource requests, interaction ID, and the deployed RSU.
4) Interactions. The class of interactions mainly includes ID pairs of interactions between each DT and other objects.

The environment is constructed based on these classes and is updated online according to the interaction of the ADM agent. As vehicles move, the agent decides whether to migrate the DT and calculates total latencies. The hyperparameters of the ADM algorithm are listed in TABLE II.

TABLE II: Hyperparameter Settings.

Hyperparameter	Value	Hyperparameter	Value
Policy Layer Type	Dense	Layer Dimension	4
Layer Hidd. Units	128	Activation Function	ReLU
Loss Function	MSELoss	Weighted Param κ	0.8
Optimizer	Adam	Weighted Param λ	1
Discount Factor	0.99	Learning Rate	0.0003
Batch Size	256	Initial Sample Percent ε	0.95

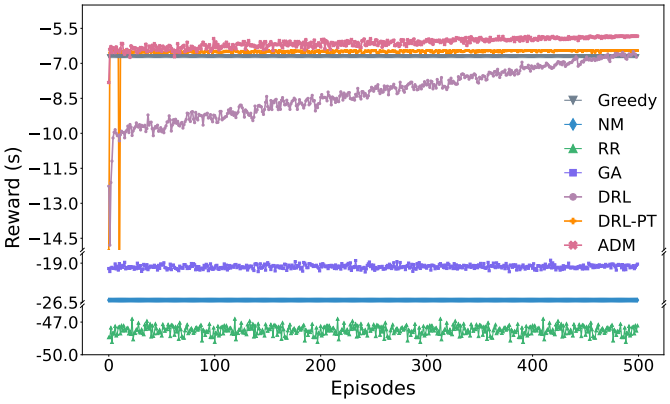


Fig. 4: Reward.

Baselines: We compare the performance of the ADM algorithm against the following five Baselines.

1) Greedy. A greedy algorithm selects the RSU with the minimized total costs of communications latency, resource colocation, and migration during DT migration.
2) Never Migration (NM). The DT does not undergo migration that remains initialized on an RSU while the corresponding physical objects continue to move.
3) Round-Robin (RR). The RR algorithm is a load-balancing algorithm that distributes requests to RSUs.
4) Genetic Algorithm (GA). The GA algorithm is a heuristic method drawing inspiration from natural selection, incorporating mutation, crossover, and selection operators, and utilizing a comprehensive cost fitness function during DT migration to identify the optimal RSUs.
5) DRL. A traditional actor-critic-based deep reinforcement learning (DRL) algorithm selects the best RSUs for DT migration directly from the same states of the ADM.
6) DRL-PT. A DRL algorithm integrates expert pre-training solely to initialize the ADM agent efficiently.

The neural network structure of DRL and DRL-PT algorithms is similar to the ADM algorithm with the same policy network. However, the key distinction lies in the initialization and training process. The ADM algorithm employs expert policies to guide its training, while the DRL algorithm starts with random parameters. The trajectories of the Greedy algorithm are saved as expert demonstrations. The ADM, DRL, and DRL-PT algorithms are trained with the same learning rate, mini-batch size, and number of gradient update steps.

B. Simulation Results

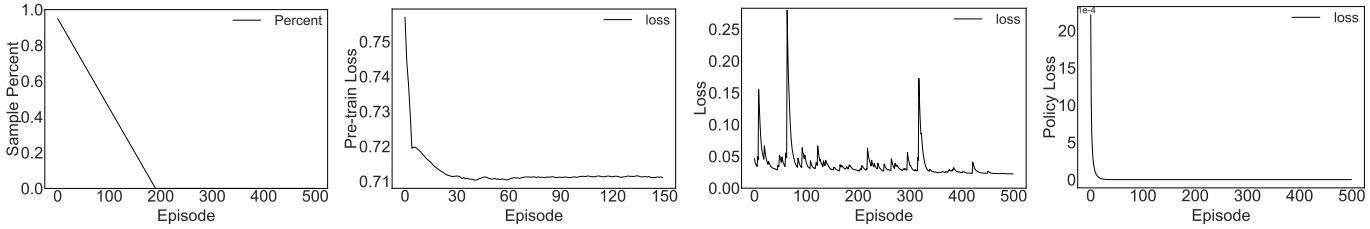
The performance of the proposed ADM algorithm and baselines is evaluated, and the results are analyzed through various experiments.

Performance of the ADM algorithm: We evaluate the training performance of the ADM algorithm on small-scale and large-scale mobility trace datasets with 500 and 2000 randomly selected mobility traces, respectively. Fig. 4 displays training results of different algorithms. The final rewards of the training are ADM $>$ DRL-PT $>$ DRL $>$ Greedy $>$ GA $>$ NM $>$ RR. The DRL algorithm initially selects random actions, leading to a period of exploration before convergence. It performs worse than the Greedy algorithm at the beginning, requiring approximately 470 training epochs to reach a more competitive performance. In contrast, the DRL-PT and ADM algorithms initialize their learning process with expert-provided policies, ensuring efficient learning of effective policies from the outset. Consequently, these two algorithms quickly surpass the Greedy algorithm after initialization and continue to improve their performance. Moreover, the ADM agent gradually reduces reliance on expert guidance during training, as illustrated in Fig. 5a. It rapidly converges to a higher value after approximately 100 epochs. This demonstrates that the ADM agent leverages expert knowledge to accelerate its learning in the initial stages and then refines its behavior through further exploration and experience.

To illustrate the convergence of the ADM algorithm, the total loss, policy loss, and pretrain loss are shown in Figs. 5b, 5c, and 5d, respectively. In the pre-training phase, the reduction and stabilization of pre-train loss indicate that the ADM agent has initialized a policy that aligns well with the desired sub-optimal policy, as shown in Fig. 5b. This significantly accelerates the subsequent RL training phase. As training progresses, the total loss curve in Fig. 5c initially decreases and stabilizes after approximately 100 epochs, signifying convergence of the ADM algorithm. As shown in Fig. 5d, the rapid and early convergence of the actor loss in the first 20 epochs illustrates that the actor network is initialized with appropriate parameters. This means expert policies help to generate meaningful actions from the beginning of training, contributing to the quick convergence of the actor loss.

We evaluate the ADM algorithm and baselines on large-scale vehicular mobility trace datasets to appraise their capacity for generalization. As illustrated in Fig. 6, the ADM algorithm outperforms the other baseline algorithms, which demonstrates the adaptability and robustness of the ADM algorithm. The ADM algorithm reduces the overall average

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3492349



(a) Sample percentage. (b) Pre-train loss. (c) Total loss. (d) Policy loss.

Fig. 5: Losses and sample percentage of ADM algorithm.

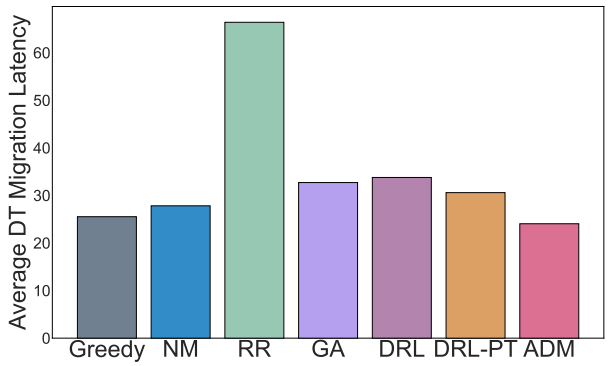


Fig. 6: Average latency.

DT migration latency of Greedy, NM, RR, GA, DRL, and DRL-PT algorithms by 6%, 14%, 63%, 26%, 28%, and 21%, respectively. This shows the effectiveness of the ADM algorithm in the context of the complex and dynamic VECONs.

Performance with different migration coefficients: To compare the impact of migration latency on the overall latency, we evaluate these algorithms with different coefficients of migration latency in Fig. 7. In Fig. 7a, as the migration cost coefficient increases, the performance of most algorithms degrades except for the NM algorithm, which does not involve DT migration. Due to the frequent migration decisions, the RR algorithm is more susceptible to variations in migration coefficient. It's observed that the DRL-PT and DRL algorithms exhibit poorer performance compared to the ADM and Greedy algorithms when migration coefficients exceed 3. This observation indicates that the former two algorithms encounter exploration challenges. Exploration becomes difficult in environments with high migration coefficients, as suboptimal actions result in significant penalties. Expert policies help mitigate the exploration challenge by providing more informed actions, enabling the ADM algorithm to benefit from integrating expert insights to bypass unnecessary exploration. In Fig. 7b, the ADM algorithm reduces the total migration latency of DTs than Greedy, NM, RR, GA, DRL, and DRL-PT algorithms by 25%, 61%, 93%, 83%, 54%, and 23% on average, respectively.

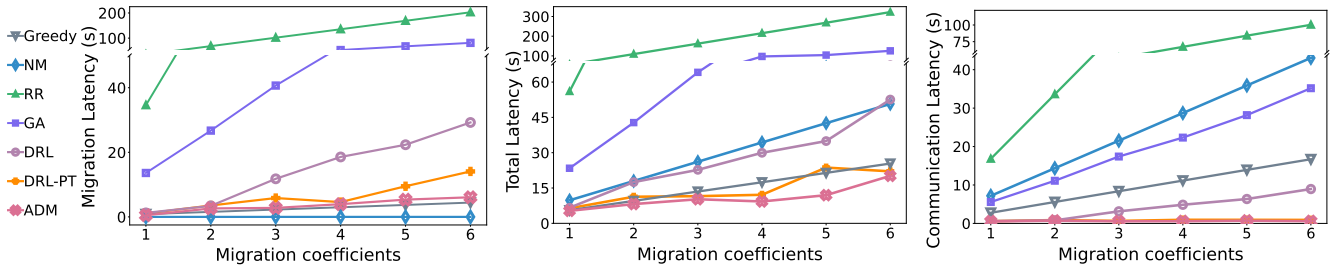
Fig. 7c reveals that ADM, DRL, and DRL-PT algorithms consistently demonstrate superior and stable communication latency compared to other heuristic algorithms as the migration coefficient progressively increases. Overall, the ADM algorithm learns to maintain the lowest communication latency among the scenarios with different migration costs, and the order of communication latencies of these algorithms is ADM $<$ DRL-PT $<$ DRL $<$ Greedy $<$ GA $<$ NM $<$ RR.

Performance with different numbers of moving vehicles: Fig. 8 shows the performance of each algorithm under the different numbers of moving vehicles. In Fig. 8a, the number of moving vehicles is set from 400 to 800 with an interval of 100. As the number of vehicles increases, DTs wait in queues for longer durations, amplifying total DT migration latency as shown in Fig. 8b. It can be seen from the figure that the ADM algorithm can reduce up to 40% of the total migration latency against the other algorithms. Overall, the total migration latency with different numbers of vehicles is reduced by 13%, 25%, 71%, 43%, 41%, and 25% on average compared with Greedy, NM, RR, GA, DRL, and DRL-PT algorithms, respectively. This means the ADM agent can adapt its policies based on real-time interactions, learning to make decisions that reduce DT migration latency in response to changing environments. As shown in Fig. 8c, the ADM algorithm achieves the lowest and most stable communication latency of DTs. As the number of vehicles grows, the ADM algorithm can dynamically select RSUs that minimize latency and ensure efficient data transfer in complex communication relationships. Overall, the order of average communication latency of these algorithms is ADM $<$ DRL-PT $<$ DRL $<$ GA $<$ Greedy $<$ NM $<$ RR.

Performance with different arriving rates of vehicles: We evaluate the performance with different vehicle arriving rates in Fig. 9. The migration latency is defined as the total migration time for all DTs divided by their count. As shown in Fig. 9a, the migration latency also rises with the increase in the arrival rate. The system seeks to expedite DT migration to meet DT response requirements. We find that the ADM effectively adapts to changes in the arrival rate, outperforming baselines that require DT migration. Fig. 9b shows the total latencies of all evaluated algorithms increase with the arrival rate increase since the number of DTs increases at each time slot. Overall, the total ADM latency outperforms Greedy, NM, RR, GA, DRL, and DRL-PT algorithms by 10%, 57%, 87%, 70%, 40%, and 36% on average, respectively. This is because the ADM algorithm considers the current resource availability and utilization across different RSUs and makes more informed migration decisions from a long-term perspective.

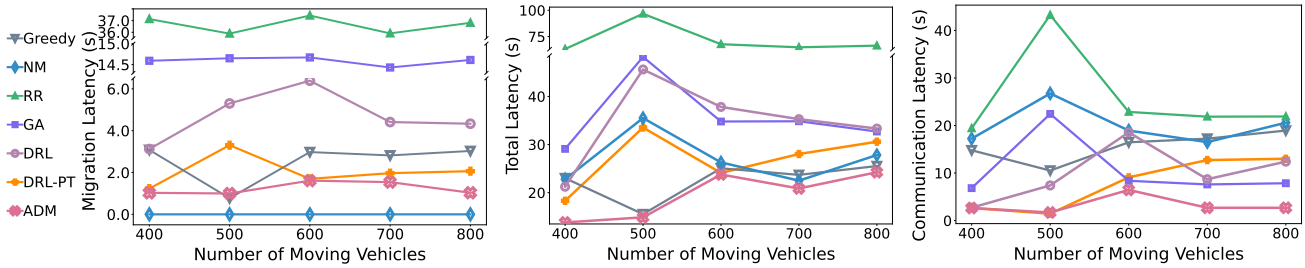
The total communication latency of different algorithms is shown in Fig. 9c. It shows that the ADM algorithm outperforms baselines with the order as ADM $<$ DRL-PT $<$

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3492349



(a) Migration Latency. (b) Total ADM Latency. (c) Communication Latency.

Fig. 7: Performance with different migration coefficients.



(a) Migration Latency. (b) Total ADM Latency. (c) Communication Latency.

Fig. 8: Performance with different numbers of moving vehicles.

DRL $<$ Greedy $<$ GA $<$ NM $<$ RR while also exhibiting more excellent stability. ADM, DRL, and DRL-PT algorithms perform better than other heuristic baselines. In complex DT communication relationships, where short-term gains might lead to suboptimal long-term performance, RL can make better trade-offs by considering the overall impact of decisions over time. The communication latency mainly includes pair-wise cost, cooperation cost, and interaction cost, as shown in Figs. 9d, 9e, and 9f, respectively. As the arrival rates of vehicles increase, the existing communications of DTs are more complex, so these three detailed costs increase.

TABLE III: Computation resources for each decision-making.

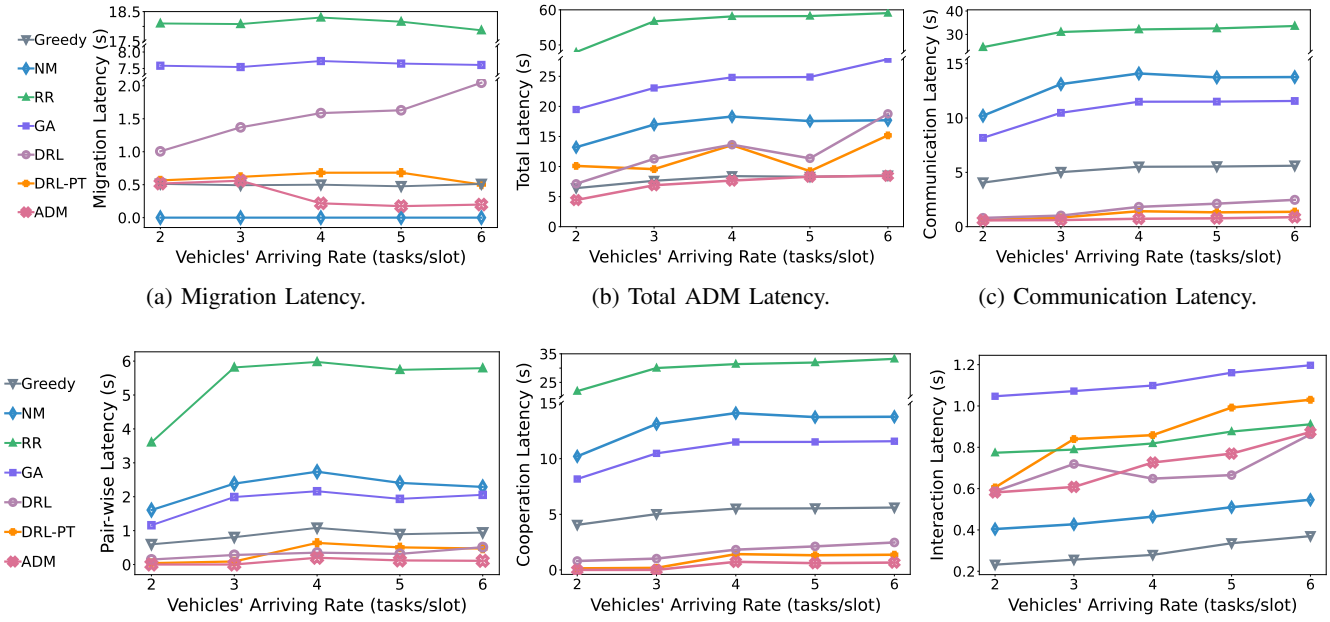
Algorithm	Vehicles	RAM	VRAM	Execution Time
Greedy	500	-	-	1.123×10^{-3}
NM	500	-	-	2.510×10^{-7}
RR	500	-	-	1.274×10^{-6}
GA	500	-	-	3.941×10^{-2}
DRL	500	112.4 Kb	112.8 Kb	1.730×10^{-3}
DRL-PT	500	115.6 Kb	120.4 Kb	1.218×10^{-3}
ADM	500	143.2 Kb	124.1 Kb	1.319×10^{-3}
ADM	1000	139.8 Kb	120.0 Kb	1.522×10^{-3}
ADM	2000	140.6 Kb	148.2 Kb	1.537×10^{-3}
ADM	3000	146.4 Kb	152.4 Kb	1.597×10^{-3}

Computational complexity and execution time: To further demonstrate the computational complexity of all algorithms, we use *torch.profiler* [50] to obtain the Random Access Memory (RAM), Video RAM (VRAM), and execution time for different algorithms. The results of the average decision-making resources and time are presented in TABLE III. The algorithms with the least execution time are NM and RR due to their lack of complex heuristic rules and neural network reasoning processes. The GA algorithm has the longest execution time because it needs to explore more possible solutions. The computational demands and execution time of the ADM algorithm closely resemble those of RL algorithms, demonstrating that our enhancements do not significantly increase operational overhead. Therefore, our algorithm maintains reasonable computational complexity and is practical.

Performance for DT migrations: To further illustrate the effect of the algorithm, the average migration frequency is shown in Fig. 10. It can be seen from the figure that the average migration frequency of the ADM algorithm is the lowest among all algorithms. This fully demonstrates that the ADM algorithm can effectively reduce the number of migrations while maintaining a high reward, further illustrating the effectiveness of the algorithm. On the other hand, the migration frequency of the Greedy algorithm is relatively higher. The Greedy algorithm cannot consider long-term rewards, resulting in frequent migrations. The latencies of different sizes of DTs are shown in Fig. 11. The end-to-end migration latencies range from hundreds of milliseconds to a few seconds. The algorithms with larger latency do not indicate flaws in the simulation model itself but rather reflect the practical limitations and trade-offs inherent to baselines. This highlights the necessity of algorithmic designs that our ADM algorithm could balance computational efficiency and decision effectiveness to minimize migration latencies.

Average migration latency for each DT: To ensure the stability of our proposed ADM, the detailed and average migration latencies over 200 diverse DTs are recorded in a single episode. As depicted in Fig. 12, the ADM algorithm demonstrates minimal volatility compared to other baselines during the DT migration process. By the average migration latency lines, it becomes evident that the ADM algorithm achieves the lowest average latency time throughout the entire

This article has been accepted for publication in IEEE Transactions on Vehicular Technology. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVT.2024.3492349



(a) Migration Latency. (b) Total ADM Latency. (c) Communication Latency.

(d) Pair-wise Cost. (e) Cooperation Cost. (f) Interaction Cost.

Fig. 9: Performance with different arriving rates of vehicles.

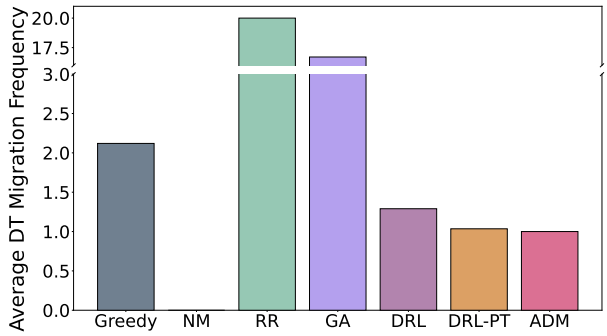


Fig. 10: Average DT migration frequency.

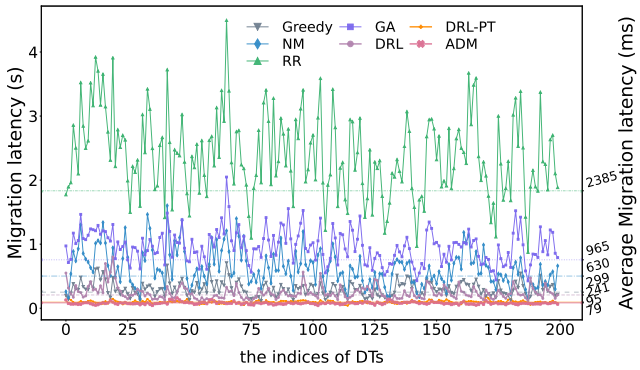


Fig. 12: Average migration latency for each DT.

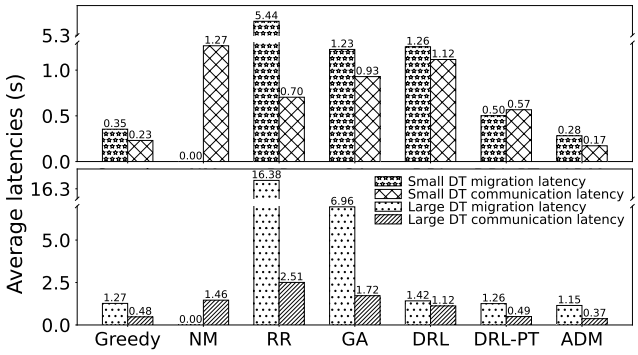


Fig. 11: Latency results of various DTs.

episode. Therefore, the ADM algorithm has created a stable, long-term optimized strategy that effectively addresses the challenges of DT migration in VECONs.

VI. Discussion

The ADM algorithm is built on a robust foundation of real-world data and can be deployed on RSUs using the Kubernetes scheduling framework [51]. The status of vehicles, e.g., speed and direction, can be seamlessly exchanged with RSUs through C-V2X technology [7]. RSUs possess sufficient computational and communication capabilities, allowing for continuous monitoring and processing of real-time resource load data collected by Prometheus via Kubernetes APIs [52]. Precisely, by utilizing custom iperf3 export files [53], the status of DTs at each time slot, connectivity status, and communication latency between each DT are monitored. The colocation costs on each RSU and the migration latencies for DTs across RSUs are recorded by the custom node export file.

There are two stages in the Kubernetes scheduling process: the scheduling cycle and the binding cycle [51]. During the scheduling cycle, the custom plugin of our ADM algorithm can select the most suitable edge node for each DT, ensuring

optimal migration for minimizing communication latency and migration latency. Then, the binding cycle binds the decision to the chosen RSU, effectively executing the migration decision. The Kubernetes scheduling framework with complete customization offers a robust platform to integrate our algorithm effectively.

Deploying such a sophisticated system demands significant engineering effort, and incorporating RL adds complexity due to its iterative training and decision-making processes. The main goal of this proposed algorithm is to evaluate DT migration in VECONs. To thoroughly assess its performance and efficiency, we utilized large-scale simulations as a robust tool. This method enabled us to create a controlled yet realistic environment that mirrors complex real-world scenarios. Additionally, we acknowledge the importance of practical implementation. As noted, our team is simultaneously working on integrating the algorithm into the Kubernetes system.

VII. Conclusion

In this work, we proposed the ADM algorithm to solve the adaptive DT migration problem in VECONs. We modeled the ADM problem comprehensively, considering the complex communication latency, colocation cost, and migration latency. Then, the ADM algorithm based on policy gradient RL was proposed for adaptive migration decisions. Expert demonstrations were utilized to improve the exploration and exploitation in sparse environments. We evaluated the proposed ADM algorithm using real-world data traces, and experimental results showed that our ADM algorithm consistently outperforms the baseline algorithms with an average 39% improvement in migration latency. Future work will include integrating DT-based complex task scheduling and edge caching, as well as exploring hybrid strategies that combine local centralized and global distributed decision-making. This dual approach seeks to balance responsiveness and overall network sum-rate performance.

References

[1] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[2] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, "Exploring mobile edge computing for 5g-enabled software defined vehicular networks," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 55–63, 2017.

[3] L. Zhao, Z. Bi, A. Hawbani, K. Yu, Y. Zhang, and M. Guizani, "Elite: An intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

[4] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari, "Mobility digital twin: Concept, architecture, case study, and future challenges," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 452–17 467, 2022.

[5] H. Feng, D. Chen, and Z. Lv, "Blockchain in digital twins-based vehicle management in vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 19 613–19 623, 2022.

[6] S. Kitajima, H. Chouchane, J. Antona-Makoshi, N. Uchida, and J. Tajima, "A nationwide impact assessment of automated driving systems on traffic safety using multiagent traffic simulations," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 302–312, 2022.

[7] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, "How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence," *IEEE Transactions on Intelligent Vehicles*, 2023.

[8] Y. Wu, K. Zhang, and Y. Zhang, "Digital twin networks: A survey," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 789–13 804, 2021.

[9] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, "Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach," *IEEE Transactions on Mobile Computing*, 2021.

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

[11] Y. Hui, X. Ma, Z. Su, N. Cheng, Z. Yin, T. H. Luan, and Y. Chen, "Collaboration as a service: Digital-twin-enabled collaborative and distributed autonomous driving," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 607–18 619, 2022.

[12] B. Li, W. Xie, Y. Ye, L. Liu, and Z. Fei, "Flexedge: Digital twin-enabled task offloading for uav-aided vehicular edge computing," *IEEE Transactions on Vehicular Technology*, pp. 1–6, 2023.

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

[14] H. Wang, S. Lin, and J. Zhang, "Warm-start actor-critic: From approximation error to sub-optimality gap," in *International Conference on Machine Learning*. PMLR, 2023, pp. 35 989–36 019.

[15] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai, "Policy finetuning: Bridging sample-efficient offline and online reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 27 395–27 407, 2021.

[16] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proceedings of the 29th International Conference on International Conference on Machine Learning*. ICML'12, 2012, pp. 179–186.

[17] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2013.

[18] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Transactions on Wireless Communications*, 2023, doi:10.1109/TWC.2023.3254158.

[19] R. Zhang, Z. Xie, D. Yu, W. Liang, and X. Cheng, "Digital twin-assisted federated learning service provisioning over mobile edge networks," *IEEE Transactions on Computers*, vol. 73, no. 2, pp. 586–598, 2024.

[20] X. Yuan, J. Chen, N. Zhang, J. Ni, F. R. Yu, and V. C. M. Leung, "Digital twin-driven vehicular task offloading and irs configuration in the internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 290–24 304, 2022.

[21] B. Lu, B. Fan, Y. Wu, L. Qian, H. Zhang, and R. Lu, "Predictive computation offloading and resource allocation in dt-empowered vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[22] J. Lou, Z. Tang, W. Jia, W. Zhao, and J. Li, "Startup-aware dependent task scheduling with bandwidth constraints in edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2023, doi:10.1109/TMC.2023.3238868.

[23] Y. Sun, S. Zhou, and J. Xu, "Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2637–2646, 2017.

[24] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 627–640, 2015.

[25] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 712–725, 2018.

[26] J. Wang, J. Hu, G. Min, Q. Ni, and T. El-Ghazawi, "Online service migration in mobile edge with incomplete system information: A deep recurrent actor-critic learning approach," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022, doi:10.1109/TMC.2022.3197706.

[27] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041–9052, 2020.

[28] Z. Tang, F. Mou, J. Lou, W. Jia, Y. Wu, and W. Zhao, "Multi-user layer-aware online container migration in edge-assisted vehicular networks," *IEEE/ACM Transactions on Networking*, 2023, doi:10.1109/TNET.2023.3330255.

[29] 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. 12 240–12 251, 2020.

[30] 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. 16 219–16 230, 2021.

[31] C. Chen, J. Hu, T. Qiu, M. Atiquzzaman, and Z. Ren, "Cvcg: Cooperative v2v-aided transmission scheme based on coalitional game for popular content distribution in vehicular ad-hoc networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 12, pp. 2811–2828, 2018.

[32] L. Liu, J. Feng, C. Wu, C. Chen, and Q. Pei, "Reputation management for consensus mechanism in vehicular edge metaverse," *IEEE Journal on Selected Areas in Communications*, 2023.

[33] L. Zhao, Z. Zhao, E. Zhang, A. Hawbani, A. Al-Dubai, Z. Tan, and A. Hussain, "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE Journal on Selected Areas in Communications*, 2023.

[34] M. Mao, P. Yi, J. Zhang, and J. Pei, "Detecting malicious roadside units in vehicular social networks for information service," *Wireless Personal Communications*, vol. 130, no. 4, pp. 2565–2588, 2023.

[35] Y. Zhang, L. Jiao, J. Yan, and X. Lin, "Dynamic service placement for virtual reality group gaming on mobile edge cloudlets," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1881–1897, 2019.

[36] 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 Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[37] T. Ouyang, R. Li, X. Chen, Z. Zhou, and X. Tang, "Adaptive user-managed service placement for mobile edge computing: An online learning approach," in *IEEE INFOCOM 2019-IEEE conference on computer communications (INFOCOM)*. IEEE, 2019, pp. 1468–1476.

[38] S.-H. Lim, J.-S. Huh, Y. Kim, G. M. Shipman, and C. R. Das, "D-factor: a quantitative model of application slow-down in multi-resource shared systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, pp. 271–282, 2012.

[39] V. Verter, "Uncapacitated and capacitated facility location problems," *Foundations of location analysis*, pp. 25–37, 2011.

[40] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. Lau, "Dynamic virtual machine management via approximate markov decision process," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[41] A. A. Khan and R. S. Adve, "Centralized and distributed deep reinforcement learning methods for downlink sum-rate optimization," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 8410–8426, 2020.

[42] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu, "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5109–5116.

[43] Radio equipment locations on bundesnetzagentur. [Online]. Available: https://www.bundesnetzagentur.de/DE/Vportal/TK/Funktechnik/EMF/start.html

[44] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[45] P. Zhou, X. Chen, Z. Liu, T. Braud, P. Hui, and J. Kangasharju, "Drle: Decentralized reinforcement learning at the edge for traffic light control in the iov," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2262–2273, 2020.

[46] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in *Proceedings of the 4th ACM European conference on Computer systems (EuroSys)*, 2009, pp. 205–218.

[47] D. Naboulsi and M. Fiore, "Characterizing the instantaneous connectivity of large-scale urban vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1272–1286, 2016.

[48] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[49] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari, "Mobility digital twin: Concept, architecture, case study, and future challenges," *IEEE Internet of Things Journal*, vol. 9, no. 18, pp. 17 452–17 467, 2022.

[50] Pytorch documentation. [Online]. Available: https://pytorch.org/docs/

[51] Scheduling framework. [Online]. Available: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/

[52] J. Turnbull, *Monitoring with Prometheus*. Turnbull Press, 2018.

[53] Y.-X. Huang and J. Chou, "Evaluations of network performance enhancement on cloud-native network function," in *Proceedings of the 2021 on Systems and Network Telemetry and Analytics*, 2020, pp. 3–8.

Fangyi Mou received the B.S. degree from Department of Electronic, Communication and Physics, Shandong University of Science and Technology, China, in 2017 and the M.Sc. degree from Department of Computer Science, University of Macau, China, in 2020. She is currently working toward the Mphil degree at BNU-HKBU United International College, and as a research assistant with Institute of AI and Network, Beijing Normal University, China. Her research interests include edge computing, resource allocation, and reinforcement learning.

Jiong Lou received my B.S. degree and Ph.D. degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2016 and 2023. Since 2023, he has held the position of research assistant professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He has published more than ten papers in leading journals and conferences (e.g., TMC, TSC and CN). His current research interests include edge computing, task scheduling and container management.

Zhiqing Tang received the B.S. degree from School of Communication and Information Engineering, University of Electronic Science and Technology of China, China, in 2015 and the Ph.D. degree from Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2022. He is currently an assistant professor with the Advanced Institute of Natural Sciences, Beijing Normal University, China. His current research interests include edge computing, resource scheduling, and reinforcement learning.

Yuan Wu (S'08-M'10-SM'16) received the PhD degree in Electronic and Computer Engineering from the Hong Kong University of Science and Technology in 2010. He is currently an Associate Professor with the State Key Laboratory of Internet of Things for Smart City, University of Macau and also with the Department of Computer and Information Science, University of Macau. During 2016-2017, he was a visiting scholar with Department of Electrical and Computer Engineering, University of Waterloo. His research interests include resource management for wireless networks, green communications and computing, mobile edge computing and edge intelligence. He was a recipient of the Best Paper Award from the IEEE International Conference on Communications in 2016, and the Best Paper Award from IEEE Technical Committee on Green Communications and Computing in 2017. Dr. Wu is currently on the Editorial Boards of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, and IEEE INTERNET OF THINGS JOURNAL.

Weijia Jia (Fellow, IEEE) is currently a Chair Professor, Director of BNU-UIC Institute of Artificial Intelligence and Future Networks, Beijing Normal University (Zhuhai) and VP for Research of BNU-HKBU United International College (UIC) and has been the Zhiyuan Chair Professor of Shanghai Jiao Tong University, China. He was the Chair Professor and the Deputy Director of State Kay Laboratory of Internet of Things for Smart City at the University of Macau. He received BSc/MSc from Center South University, China, in 82/84 and Master of Applied Sci./PhD from Polytechnic Faculty of Mons, Belgium in 92/93, respectively, all in computer science. From 93-95, he joined German National Research Center for Information Science (GMD) in Bonn (St. Augustine) as a research fellow. From 95-13, he worked at City University of Hong Kong as a professor. His contributions have been recognized as optimal network routing and deployment, anycast and QoS routing, sensors networking, AI (knowledge relation extractions, etc.), and edge computing. He has over 600 publications in the prestige international journals/conferences and research books. He has received the 1st Prize of Scientific Research Awards from the Ministry of Education of China in 2017 (list 2). He has served as area editor for various prestige international journals, chair and PC member/skeynote speaker for many top international conferences. He is the Fellow of IEEE and the Distinguished Member of CCF.

Yan Zhang (Fellow, IEEE) received the Ph.D. degree in electrical and electronic engineering from the School of Electrical and Electronics Engineering, Nanyang Technological University in 2005. He is currently a Full Professor with the Department of Informatics, University of Oslo. His research interests include next-generation wireless networks leading to 5G beyond/6G, green, and secure cyber-physical systems. Dr. Zhang was the recipient of the global Highly Cited Researcher Award in 2018 and 2019. He is a Fellow of IET, and an elected Member of Academia Europaea and Norwegian Academy of Technological Sciences. He is the Editor (or the Area Editor, an Associate Editor) for several IEEE publications, including IEEE COMMUNICATIONS MAGAZINE, IEEE NETWORK MAGAZINE, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE COMMUNICATIONS SURVEY AND TUTORIALS, IEEE INTERNET OF THINGS JOURNAL, IEEE SYSTEM JOURNAL, etc. He is the Symposium/Track Chair for a number of conferences, including IEEE ICC2021, IEEE Globecom 2017, IEEE PIMRC 2016, etc. He was an IEEE Vehicular Technology Society Distinguished Lecturer during 2016-2020. He is the Chair of IEEE Communications Society Technical Committee on Green Communications and Computing.

Wei Zhao (Fellow, IEEE) completed his undergraduate studies in physics at Shaanxi Normal University, China, in 1977, and received his MSc and PhD degrees in Computer and Information Sciences at the University of Massachusetts at Amherst in 1983 and 1986, respectively. Professor Zhao has served important leadership roles in academic including the Chief Research Officer at the American University of Sharjah, the Chair of Academic Council at CAS Shenzhen Institute of Advanced Technology, the eighth Rector of the University of Macau, the Dean of Science at Rensselaer Polytechnic Institute, the Director for the Division of Computer and Network Systems in the U.S. National Science Foundation, and the Senior Associate Vice President for Research at Texas A&M University. Professor Zhao has made significant contributions to cyber-physical systems, distributed computing, real-time systems, and computer networks. He led the effort to define the research agenda of and to create the very first funding program for cyber-physical systems in 2006. His research results have been adopted in the standard of Survivable Adaptable Fiber Optic Embedded Network. Professor Zhao was awarded the Lifelong Achievement Award by the Chinese Association of Science and Technology in 2005.