Mobility-aware Seamless Service Migration and Resource Allocation in Multi-edge IoV Systems

Zheyi Chen, *Member, IEEE*, Sijin Huang, Geyong Min, *Member, IEEE*, Zhaolong Ning, *Senior Member, IEEE*, Jie Li, *Fellow, IEEE*, and Yan Zhang, *Fellow, IEEE*

Abstract—Mobile Edge Computing (MEC) offers low-latency and high-bandwidth support for Internet-of-Vehicles (IoV) applications. However, due to high vehicle mobility and finite communication coverage of base stations, it is hard to maintain uninterrupted and high-quality services without proper service migration among MEC servers. Existing solutions commonly rely on prior knowledge and rarely consider efficient resource allocation during the service migration process, making it hard to reach optimal performance in dynamic IoV environments. To address these important challenges, we propose *SR-CL*, a novel mobility-aware seamless Service migration and Resource allocation framework via Convex-optimization-enabled deep reinforcement Learning in multi-edge IoV systems. First, we decouple the Mixed Integer Nonlinear Programming (MINLP) problem of service migration and resource allocation into two sub-problems. Next, we design a new actor-critic-based asynchronous-update deep reinforcement learning method to handle service migration, where the delayed-update actor makes migration decisions and the one-step-update critic evaluates the decisions to guide the policy update. Notably, we theoretically derive the optimal resource allocation with convex optimization for each MEC server, thereby further improving system performance. Using the real-world datasets of vehicle trajectories and testbed, extensive experiments are conducted to verify the effectiveness of the proposed *SR-CL*. Compared to benchmark methods, the *SR-CL* achieves superior convergence and delay performance under various scenarios.

Index Terms—Mobile Edge Computing, Internet-of-Vehicles, service migration, deep reinforcement learning, convex optimization.

1 Introduction

With the fast development of 5G techniques, Internet-of-Vehicles (IoV) has become a new module in smart cities. Equipped with intelligent sensors and components, vehicles are capable of hosting various applications such as automatic driving, image recognition, and path planning [1]. However, the real-time demands of IoV applications pose significant challenges for onboard processors with limited computational capabilities [2]. Although Cloud Computing (CC) allows offloading tasks to the remote cloud for processing, the excessive delay caused by long distance is inevitable [3]. To relieve this issue, the emerging Mobile Edge Computing (MEC) offers low-latency and high-bandwidth services by deploying resources at the network edge [4]. Through combining MEC and IoV, a new computing architecture has emerged to enable efficient interconnection and real-time data exchange between vehicles and road infrastructures [5]. A typical MEC-enabled IoV system usually considers an area covered by multiple Base Stations (BSs), and vehicles can offload their applications' tasks to nearby MEC servers that are collocated with BSs, where virtualization is deemed as a key technique to conduct resource management [6]. When vehicles offload tasks, MEC servers create dedicated service instances via virtualization techniques for the vehicles and allocate proper resources to them [7]. The service instances encapsulate the runtime data and user context information, which offer fine computing services for vehicles while ensuring resource isolation.

Considering the high mobility of vehicles and finite communication coverage of BSs, it is challenging for a single MEC server to provide seamless services to a vehicle without multi-edge cooperation, which may seriously degrade the Quality-of-Service (QoS) [8]. To guarantee high QoS, the service instances created by MEC servers are expected to be properly migrated along with the mobility of vehicles [9]. The performance of service migration depends on multiple factors including vehicle mobility, attribute of offloaded tasks, and available MEC resources. An improper migration strategy might cause excessive task response delay and degraded QoS. Commonly, the process of service migration in multi-edge IoV systems can be regarded as a long-term decision-making problem. The current migration decisions might affect future system performance, and thus it is challenging to optimize the long-term performance without foreknowing the potential mobility of vehicles. Moreover, there might exist mutual influence among the migration decisions of different vehicles, making it extremely hard to simultaneously optimize service migration for all vehicles with the concern of minimizing system delay.

- *Zheyi Chen and Sijin Huang are with the College of Computer and Data Science, Fuzhou University, Fuzhou 350116, China, the Engineering Research Center of Big Data Intelligence, Ministry of Education, Fuzhou 350002, China, and also with the Fujian Key Laboratory of Network Computing and Intelligent Information Processing (Fuzhou University), Fuzhou 350116, China. E-mail: z.chen@fzu.edu.cn, sjinhuang@163.com.*
- *Geyong Min is with the Department of Computer Science, Faculty of Environment, Science and Economy, University of Exeter, Exeter EX4 4QF, United Kingdom. E-mail: g.min@exeter.ac.uk.*
- *Zhaolong Ning is with the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. E-mail: z.ning@ieee.org.*
- *Jie Li is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: li-jiecs@sjtu.edu.cn.*
- *Yan Zhang is with the Department of Informatics, University of Oslo, 0316 Oslo, Norway. E-mail: yanzhang@ieee.org.*

Corresponding authors: Geyong Min and Zhaolong Ning.

Following migration decisions, service instances of vehicles will be migrated to target MEC servers for processing tasks. Compared to the remote cloud, MEC servers own limited computational resources, and the continuous influx of tasks commonly imposes various resource demands. Therefore, it is necessary to allocate suitable resources to service instances on resource-constrained MEC servers. Most of the existing studies [10], [11] did not well consider optimizing resource allocation when they dealt with service migration, which seriously hindered the improvement of system performance. Few studies [12], [13] investigated the joint optimization of service migration and resource allocation, regarded as a Mixed Integer Nonlinear Programming (MINLP) problem. These studies typically adopted classic optimization theories, which usually required numerous iterations and caused excessive overheads. Meanwhile, when facing complex and dynamic multi-edge IoV systems, they did not well consider future vehicle mobility and thus they easily fell into the local optimum. As an emerging technique, Deep Reinforcement Learning (DRL) [14] is deemed a promising method for handling this problem. Through interactive learning with the environment, the DRL agent can gradually adjust the policy to maximize long-term cumulative rewards. Most of the existing studies [15], [16], [17] adopted the value-based DRL, which learned deterministic policies by selecting actions with the maximum Q-value. However, the huge decision-making space in multi-edge IoV systems may result in low learning efficiency and even training failure. In contrast, the policy-based DRL [18], [19] can handle the large action space by directly outputting the probability distribution of actions, but high variance may happen when estimating the policy gradient. Moreover, they reveal limited capability to explore continuous action space, and thus the policy may easily fall into the local optimal.

To address the above important challenges, we propose *SR-CL*, a novel mobility-aware seamless Service migration and Resource allocation framework via Convex-optimization-enabled deep reinforcement Learning in multi-edge IoV systems. The *SR-CL* introduces an effective problem decoupling to relieve the limitations of classic DRL on exploring the high-dimensional space of continuous actions. By leveraging the convex optimization theory, the optimal resource allocation can be obtained and then embedded into the DRL model, thereby significantly reducing the dimension of the action space in the original problem. Moreover, a new asynchronous update mechanism is developed to further alleviate the training fluctuations when exploring the optimal policy. The main contributions of this work are summarized as follows.

- We design a unified model for service migration and resource allocation in complex and dynamic multi-edge IoV systems. First, the long-term QoS is set as the optimization objective that consists of migration, communication, and computation delays. Next, two sub-problems including service migration and resource allocation of the original MINLP problem are decoupled and formulated, respectively.
- For the sub-problem of service migration, we propose a new actor-critic-based DRL method with the asynchronous update to explore the optimal policy. Specifically, the actor with a delayed update makes migration decisions based on system states, while the critic with a one-step update evaluates the decisions and offers accurate guidance for updating the actor.
- For the sub-problem of resource allocation, we develop a customized convex-optimization-based method. First, we prove that the sub-problem is a convex programming problem by using the Hessian matrix with constraints. Next, based on the convex optimization theory, we define the generalized Lagrange function and Karush-Kuhn-Tucker (KKT) conditions. Finally, we theoretically derive the optimal resource allocation for each MEC server under given migration decisions.
- Using the real-world datasets of vehicle trajectories in Rome City and testbed, we conduct extensive experiments to validate the effectiveness of the proposed *SR-CL*. We focus on the area of the city center with multiple complex mobility patterns of vehicles, making the experiments more practical. The results show that the *SR-CL* can always obtain better convergence and delay performance than other benchmark methods under different scenarios.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the model and formulates the problem. Section 4 details the proposed *SR-CL*. Section 5 conducts the performance evaluation. Section 6 concludes this paper.

2 Related Work

The problems of service migration and resource allocation in MEC have garnered considerable research attention while many scholars have made contributions. In this section, we review and analyze the related studies.

2.1 Service Migration

Network interruption, signal attenuation, and service instability may happen when users or vehicles move, leading to degraded QoS. To relieve this issue, the emerging technique of service migration is expected to maintain service continuity and data integrity. Velrajan $et\ al.$ [10] designed a closed-loop particle swarm optimization approach for service migration, considering QoS, resource utilization, and application characteristics. Ouyang $et\ al.$ [11] divided the service placement into a series of sub-problems and addressed them via Lyapunov optimization theory under constraints of the long-term budget. Liang $et\ al.$ [12] relied on an optimal iterative scheme to deal with the integer-relaxed issue of service migration in multi-cell MEC. Liu $et\ al.$ [13] investigated the joint problem of request assignment and service migration, aiming to minimize the total response delay of requests in MEC. Scotece $et\ al.$ [20] proposed a data-handoff framework to migrate inference and training tasks between different edge nodes for reducing accuracy degradation. Maia $et\ al.$ [21] proposed an improved Genetic Algorithm (GA) to cope with the load distribution and service placement in edge computing. Wang $et\ al.$ [15] designed a Q-Learning-based approach for solving the online micro-service migration among edge servers. Zhang $et\ al.$ [16] implemented a single-user service migration strategy

based on Deep Q-Network (DQN) to reduce migration costs and ensure QoS. Peng $et\ al.$ [17] developed a DQN-based service migration method to reconcile QoS and migration overheads. Zhang $et\ al.$ [22] proposed a DQN-based service migration approach with the consideration of network status and user mobility. Labriji $et\ al.$ [23] designed a proactive real-time migration method to migrate computing services in vehicular networks, which integrated a mobility prediction algorithm based on neural networks and Markov chains with a Lyapunov-based online optimizer. Perin $et\ al.$ [24] developed a resource scheduler that incorporated a dual-layer optimization mechanism to reduce the carbon footprint of edge networks while ensuring QoS.

In general, most of the existing studies handled service migration by using classic optimization theories or heuristics. The optimization theories usually require numerous iterations, resulting in high system costs. The heuristics commonly rely on expert experiences, which limits their applicability and causes excessive rule-setting overheads. Moreover, due to the complexity and dynamics of multi-edge IoV environments, the above methods might tend to fall into the local optimum. Some studies adopted the value-based DRL for service migration, exhibiting good performance with small action space. Nevertheless, as the action space grows explosively, they cannot collect sufficient training samples to learn an accurate value function, causing unstable learning processes and unsatisfying migration results. Although few studies [18], [19] can better handle this problem by using the policy-based DRL to directly output the probability distribution of actions, a high variance may occur when estimating the policy gradient. This problem is caused by policy parameterization and environment dynamics, significantly degrading the stability and efficiency of the learning process. Moreover, most of these studies did not conduct further optimization for resource allocation, which seriously restrained the performance enhancement of service migration.

2.2 Resource Allocation

In MEC systems, end devices can offload application tasks to nearby MEC servers for processing, alleviating their capacity constraints. However, compared to the remote cloud, MEC servers own fewer resources while facing a continuous influx of tasks with various demands, and thus it is necessary to design proper strategies for resource allocation. Su $et\ al.$ [25] proposed two computation offloading and resource allocation methods by combining the weighted minimum Mean Squared Error (MSE) algorithm, quadratic transformation, and difference of convex functions algorithm. Du $et\ al.$ [26] designed a sub-optimal strategy for computation offloading and resource allocation in hybrid fog/cloud systems. Deng $et\ al.$ [27] proposed an approximate optimization method for resource allocation to balance delay and power consumption. Li $et\ al.$ [28] investigated the joint problem of computation offloading and resource allocation in multi-edge scenarios and designed a sub-optimal iterative scheme based on alternating and convex optimization, aiming to reduce energy consumption under various constraints. Jošilo $et\ al.$ [29] proposed a decentralized equilibrium calculation approach for resource allocation to reduce task completion delay in edge systems. Huang $et\ al.$ [30] designed an online DRL-based resource allocation framework to maximize the computation rate with constraints of energy consumption. Zhou $et\ al.$ [31] developed an improved DRL-based computation offloading and resource allocation method to reduce the energy consumption of MEC systems with delay constraints. Seid $et\ al.$ [32] designed a model-free DRL-based method for resource allocation to minimize task execution delay and energy consumption in multi-UAV networks.

Generally, the above studies targeted the problem of resource allocation or computation offloading in MEC, but they did not well consider the mobility of users or vehicles during the decision-making process. In real-world scenarios, the geographic locations of users or vehicles may change frequently, posing significant challenges in maintaining high-quality and seamless services. In this regard, service instances are expected to be dynamically migrated according to the mobility of users or vehicles, leading to time-varying numbers of service instances and fluctuating workloads on different MEC servers. Under such dynamic and complex scenarios, it is extremely challenging to allocate proper resources to service instances on MEC servers, which will severely affect the performance of service migration.

To the best of our knowledge, this is the first of its kind to propose a mobility-aware framework that integrates DRL with convex optimization theory for addressing the joint problem of seamless service migration and resource allocation in multi-edge IoV systems.

3 System Model and Problem Formulation

As shown in Fig. 1, the proposed multi-edge IoV system consists of a MEC controller, M BSs, and U Intelligent Vehicles (IVs). Each BS is equipped with a MEC server, named edge node. The set of edge nodes is denoted as $\mathcal{F} = \{f_1, f_2, ..., f_m, ..., f_M\}$, and the set of IVs is denoted as $\mathcal{IV} = \{IV_1, IV_2, ..., IV_u, ...IV_U\}$. In the proposed system, IV_u first sends a task to its service instance via the 5G network, and the BS forwards the task to the MEC server that locates the service instance. Next, the MEC server processes the task and returns results. During this process, the switch gathers and transmits information to the MEC controller (where the DRL agent is located), which generates migration decisions according to the mobility of IVs, ensuring seamless and high-quality services.

Specifically, the proposed system is running in discrete time slots, and locations of IV_u may change at the beginning of each time slot t, where $t \in \{0, 1, 2, ..., T\}$. Meanwhile, the intelligent application on IV_u generates a computation-intensive task at each time slot, denoted by $Task_{u,t}$. Due to the limited computational capability of IV_u, these tasks are continuously offloaded to the MEC server for processing. At the first time slot, IV_u accesses the system via its nearest BS and creates a service instance on the associated MEC server. In subsequent time slots, this service instance offers computing services to IV_u. The moving IV_u connects to the nearest edge node and sends tasks. Due to the high mobility of vehicles, IV_u may drive out of the communication coverage of the current edge node. Thus, the connection will be interrupted and IV_u will reconnect to the nearest edge

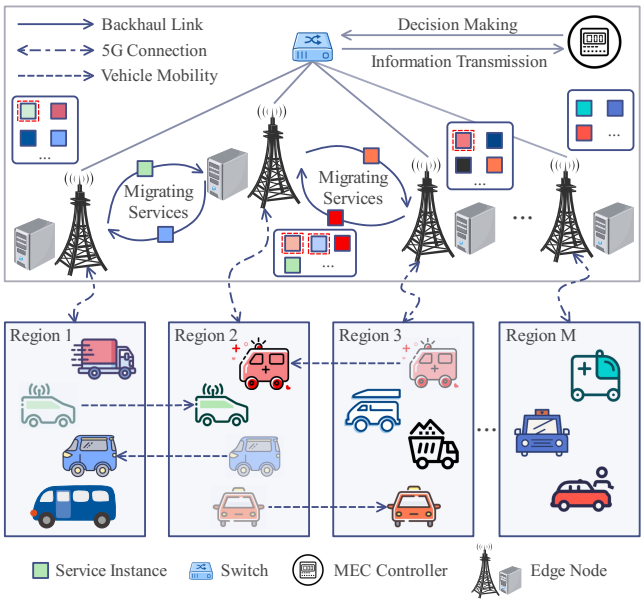

Fig. 1. The proposed multi-edge IoV system.

node. The tasks from IVs are processed by their own service instances, which run in parallel and consume part of the computational resources on MEC servers.

At the time slot t, the edge node connected to IV_u is denoted as $f_{i(1\leq i\leq M)} \in \mathcal{F}$, and the edge node that locates the service instance of IV_u is denoted as $f_{j(1\leq j\leq M)} \in \mathcal{F}$. Meanwhile, edge nodes are inter-connected via stable backhaul links. In the case that IV_u disconnects to f_j, its task can still be transmitted to f_j via the backhaul link, but this may cause extra communication delay. This extra delay can be avoided if the service instance moves to f_i, but the migration of service data also leads to migration delay. Therefore, it is tough to reduce these delays effectively while ensuring QoS by determining the suitable time and destination of service migration. Moreover, when a service instance is migrated to an edge node, proper resources should also be allocated to reduce task computation delay. With the above concerns, we formulate the joint optimization problem of service migration and resource allocation with the objective of reducing system delays including migration, communication, and computation delays.

3.1 Migration Model

We define $f_{c(1\leq c\leq M)} \in \mathcal{F}$ as the edge node that locates the service instance of IV_u at the time slot t-1 and $x_{u,t} \in \{1, 2, ..., M\}$ as the migration decision of IV_u at the time slot t, and thus f_j is determined by $x_{u,t}$ (i.e., $j = x_{u,t}$). The set of migration decisions of all IVs at the time slot t is denoted as $\mathcal{X}_t = \{x_{u,t}|IV_u \in \mathcal{IV}\}$. Moreover, we use $\delta_{u,t}^{c,j}$ to measure the hop distance between f_c and f_j. If $c = j$,then $\delta_{u,t}^{c,j} = 0$, indicating that no service migration happens at the current time slot. Otherwise, the service instance of IV_u will be migrated from f_c to f_j, and the migration delay occurs. Such delay is commonly caused by service interruption and grows with the increasing service data amount and hop distance. Therefore, the migration delay is a monotonic non-decreasing function with respect to $\delta_{u,t}^{c,j}$, and it is defined as

$$MT_{u,t} = \begin{cases} 0, & \delta_{u,t}^{c,j} = 0 \\ \frac{S_{u,t}}{\chi} + \mu^y \delta_{u,t}^{c,j}, & \delta_{u,t}^{c,j} \neq 0 \end{cases}, \tag{1}$$

where $S_{u,t}$ is the amount of service data, χ is the bandwidth of the backhaul link, and μ^y is the unit migration delay coefficient that indicates the migration delay per hop.

3.2 Communication Model

After service migration, IV_u will offload $Task_{u,t}$ to its service instance on f_j for processing, and the communication delay occurs, which consists of the data transmission delay between IV_u and f_i and the data transmission delay between f_i and f_j on the backhaul link. The Signal-to-Noise Ratio (SNR) between IV_u and f_i is defined as

$$SNR_{u,i} = \frac{P_u \alpha}{\sigma^2 |Len_{u,i}|^2}, \tag{2}$$

where P_u is the transmission power of IV_u, α is the channel gain per unit distance, $Len_{u,i}$ is the distance between IV_u and f_i, and σ^2 is the Gaussian noise.

Next, the total bandwidth of a BS is denoted as B, and the Orthogonal Frequency Division Multiplexing (OFDM) is used to equally distribute B to vehicles at each time slot. Thus, the data transmission delay of IV_u is defined as

$$PT_{u,t} = \frac{D_{u,t}}{B_{u,t} \log_2(1 + SNR_{u,i})}, \tag{3}$$

where $B_{u,t}$ indicates the available bandwidth of IV_u, and $D_{u,t}$ is the data amount of $Task_{u,t}$.

If $i \neq j$, $Task_{u,t}$ will be transmitted on the backhaul link. The data transmission delay on the backhaul link depends on $D_{u,t}$ and the hop distance between f_i and f_j. Similar to Eq. (1), we use $\phi_{u,t}^{i,j}$ to measure the hop distance. Since the data amount of task results is small, the downloading delay is negligible. Thus, the transmission delay on the backhaul link is defined as

$$ST_{u,t} = \begin{cases} 0, & \phi_{u,t}^{i,j} = 0 \\ \frac{D_{u,t}}{\chi} + \mu^h \phi_{u,t}^{i,j}, & \phi_{u,t}^{i,j} \neq 0 \end{cases}, \tag{4}$$

where μ^h is the unit transmission delay coefficient that indicates the transmission delay per hop.

Therefore, the communication delay is defined as

$$HT_{u,t} = PT_{u,t} + ST_{u,t}. \tag{5}$$

3.3 Computation Model

When $Task_{u,t}$ is offloaded to f_j, the MEC server will allocate computational resources to the service instance for processing it. The CPU cycles required to process $Task_{u,t}$ are defined as

$$K_{u,t} = D_{u,t} C_{u,t}, \tag{6}$$

where $C_{u,t}$ is the computational density of $Task_{u,t}$ that indicates the CPU cycles for processing one-bit task data.

The maximum computational capability (i.e., CPU frequency) of a MEC server is denoted as F, the proportion of computational resources allocated to the service instance of IV_u is denoted as $e_{u,t}$, and the set of allocation decisions of

all IVs at time slot t is denoted as $\mathcal{E}_t = \{e_{u,t} | IV_u \in \mathcal{IV}\}$. Therefore, the computation delay is defined as

$$CT_{u,t} = \frac{K_{u,t}}{e_{u,t}F}. \tag{7}$$

3.4 Problem Formulation

Under the decision sets of service migration $\mathcal{X}_t$ and resource allocation $\mathcal{E}_t$, we define the total delay of the proposed multi-edge IoV system as follows including migration, communication, and computation delays.

$$\mathcal{G}(\mathcal{X}_t, \mathcal{E}_t) = \sum_{IV_u \in \mathcal{IV}} \left(MT_{u,t} + HT_{u,t} + CT_{u,t}\right). \tag{8}$$

Within T time slots, we aim to minimize the long-term system delays and guarantee seamless service provisioning. Thus, the MINLP optimization problem is formulated as

$$\begin{aligned} P1: & \min_{\mathcal{X}_t, \mathcal{E}_t} \sum_{t=0}^{T} \mathcal{G}(\mathcal{X}_t, \mathcal{E}_t) \\ \text{s.t.} \quad & C1: x_{u,t} \in \{1, 2, ..., M\}, t \in T, IV_u \in \mathcal{IV}, \\ & C2: e_{u,t} \in [0, 1], t \in T, IV_u \in \mathcal{IV}, \\ & C3: \sum_{IV_u \in \mathcal{IV}_m} e_{u,t} \leq 1, t \in T, f_m \in \mathcal{F}, \end{aligned} \tag{9}$$

where the IVs that migrate their service instances to the edge node f_m are denoted as the set $\mathcal{IV}_m$. *C*1 indicates the constraint of a migration decision. *C*2 indicates the constraint of the resource proportion allocated to a service instance. A service instance can only run on a single edge node in each time slot. *C*3 indicates the constraint of the proportion sum of the computational resources allocated to service instances hosted on the edge node f_m.

Lemma 1. *P*1 *is an NP-hard problem.*

Proof. For clarity, we introduce the Knapsack Problem (KP) that is proved to be NP-hard. In KP, there is a backpack with the capacity of W and G items, denoted by the set $\mathcal{B} = \{b_1, b_2, ..., b_g, ..., b_G\}$. The aim of solving KP is to find an item sub-set $\mathcal{B}' \subseteq \mathcal{B}$ that can maximize the total value of the items in the backpack. Thus, the KP is defined as

$$\max \sum_{b_g \in \mathcal{B}'} v_g \qquad \text{s.t.} \sum_{b_g \in \mathcal{B}'} w_g \leq W. \tag{10}$$

where w_g and v_g are the weight and value of item b_g.

Next, we consider a specific example of *P*1. At the time slot t, there are U IVs and M edge nodes. IVs are moving and the computational resources of edge nodes are limited, thus it is necessary to migrate service instances among edge nodes. After service migration, IV_u can offload its task to the service instance for processing. The total computational resources of MEC servers is denoted as F_{ALL}, and the total delay is redefined as $AT_{u,t}$, where $AT_{u,t} = -(MT_{u,t} + HT_{u,t} + CT_{u,t})$. Therefore, the optimization problem in this example can be described as

$$\begin{aligned} & \max_{\mathcal{X}_t, \mathcal{E}_t} \sum_{IV_u \in \mathcal{IV}} AT_{u,t} = \max_{\mathcal{X}_t, \mathcal{E}_t} -\mathcal{G}(\mathcal{X}_t, \mathcal{E}_t) \\ & \text{s.t.} \sum_{IV_u \in \mathcal{IV}} e_{u,t}F \leq F_{ALL}. \end{aligned} \tag{11}$$

Through the above analysis, we prove that this example of *P*1 is a KP and NP-hard. Further, *P*1 considers the long-term system optimization and requires problem stacking of this example over multiple time slots. Therefore, *P*1 is an NP-hard problem.

3.5 Problem Decoupling

Notably, service migration and resource allocation belong to two dimensions of *P*1. Service migration focuses on service instances within the regional scope, while resource allocation is specific to service instances on a single edge node, posing challenges in finding a unified strategy. Considering the distinction of decision-making types between service migration and resource allocation, inspired by [33], we have discovered that by fixing the migration decisions $\mathcal{X}_t$, *P*1 can be decoupled into multiple sub-problems with separate optimization objectives and constraints. Specifically, we employ the Tammer decoupling method [34], which can promise the optimality of the solution, to decouple the highly complex *P*1 into two sub-problems with lower complexity. First, we rewrite *P*1 as

$$\min_{\mathcal{X}_t} \sum_{t=0}^{T} \left(\min_{\mathcal{E}_t} \mathcal{G}\left(\mathcal{X}_t, \mathcal{E}_t\right) \right) \qquad \text{s.t. } C1 - C3. \tag{12}$$

It can be observed that the three constraints in Eq. (12) regarding $\mathcal{X}_t$ and $\mathcal{E}_t$ are decoupled. Therefore, we can decouple *P*1 into two sub-problems as follows.

- *P*2: Minimize long-term system delays by optimizing service migration. This sub-problem is defined as

$$P2: \min_{\mathcal{X}_t} \sum_{t=0}^{T} \mathcal{G}^*\left(\mathcal{X}_t\right) \qquad \text{s.t. } C1. \tag{13}$$

- *P*3: Minimize the computation delay under the given migration decisions by optimizing resource allocation. This sub-problem is defined as

$$P3: \mathcal{G}^*\left(\mathcal{X}_t\right) = \min_{\mathcal{E}_t} \mathcal{G}\left(\mathcal{X}_t, \mathcal{E}_t\right) \qquad \text{s.t. } C2 - C3. \tag{14}$$

Once $\mathcal{X}_t$ is determined, $\sum_{IV_u \in \mathcal{IV}} \left(MT_{u,t} + HT_{u,t}\right)$ can be obtained subsequently, and we denote it as $\mathcal{I}_t$. At this point, we need to optimize $\mathcal{E}_t$ for minimizing $\sum_{IV_u \in \mathcal{IV}} CT_{u,t}$, which is the objective of *P*3. It should be noted that the resource allocation processes on different MEC servers are independent and parallel. Thus, we can further transform *P*3 into the following form.

$$\begin{aligned} \min_{\mathcal{E}_t} \mathcal{G}\left(\mathcal{X}_t, \mathcal{E}_t\right) =& \mathcal{I}_t + \min_{\mathcal{E}_t} \sum_{IV_u \in \mathcal{IV}} CT_{u,t} \\ =& \mathcal{I}_t + \min_{\mathcal{E}_t} \sum_{f_m \in \mathcal{F}} \sum_{IV_u \in \mathcal{IV}_m} CT_{u,t} \\ \text{s.t.} \quad & C2 - C3. \end{aligned} \tag{15}$$

4 The Proposed *SR-CL*

To address *P*2 and *P*3, we propose a novel mobility-aware seamless Service migration and Resource allocation framework via Convex-optimization-enabled deep reinforcement Learning (*SR-CL*) in multi-edge IoV systems.

4.1 Service Migration with Improved DRL

Service migration in multi-edge IoV systems is a sequential decision-making problem that can be modeled as a Markov Decision Process (MDP). In DRL, the 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ is commonly employed to handle the MDP, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, and γ indicate the state space, action space, state transition, reward function, and discount factor, respectively. The policy $\pi(\cdot|s_t)$ indicates the action distribution at the state s_t. Given a policy π, the DRL agent first chooses and executes an action a_t at the state s_t. Next, the environment feedbacks the instant reward r_t and steps to the next state s_{t+1}. Through this interaction process, the DRL agent will obtain a trajectory $\tau = \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, ..., s_T, a_T, r_T\}$ under the policy π. For this trajectory τ, the discounted cumulative rewards can be calculated by

$$G_t(\tau) = r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_T = \sum_{i=t}^{T} \gamma^{i-t} r_i. \quad (16)$$

Guided by a policy, multiple trajectories might be generated, and $G_t(\tau)$ follows a random distribution. The expected reward is used to evaluate the value of taking a_t at s_t, and the state-action value function is defined as

$$\begin{aligned} Q_\pi(s_t, a_t) &= \mathbb{E}_\pi[G_t \mid S = s_t, A = a_t] \\ &= r_t + \gamma \mathbb{E}_\pi[Q_\pi(s_{t+1}, a_{t+1}) \mid S = s_t, A = a_t]. \end{aligned} \quad (17)$$

The goal of DRL is to learn the optimal policy π^* for maximizing the discounted cumulative reward at any initial state. The optimal action at each state can be located by the optimal state-action value function, which is defined as

$$Q^*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t). \quad (18)$$

The value-based DRL (e.g., DQN) uses Deep Neural Networks (DNNs) to approximate $Q^*(s_t, a_t; \theta_Q)$, where θ_Q indicates DNN parameters, learning a deterministic policy by selecting the action with maximum Q-value. However, the huge action space in multi-edge IoV systems seriously affects learning efficiency. Meanwhile, the value-based DRL updates the target network by bootstrapping the Q-network, which is a biased estimation of true action values and may fall into the local optimum. In contrast, the policy-based DRL can better handle the huge action space, which selects actions based on probability distributions. Unless the policy tends to be deterministic, the probability of selecting good actions might be small, causing a high variance in estimating the policy gradient and an unstable training process.

To address these issues, we propose an improved actor-critic-based DRL with the asynchronous update to explore the optimal service migration in dynamic multi-edge IoV systems. As illustrated in Fig. 2, during the optimization process of service migration, the DRL agent selects the action a_t at the state s_t according to the policy π, and the environment feedbacks the instant reward r_t and transits to the next state s_{t+1}, which can be expressed as an MDP. The DRL agent then extracts mini-batch samples from the replay memory for network training. Specifically, the state space, action space, and reward function are defined as follows.

State space: At the time slot t, the state consists of the information about IVs (including the locations of IVs, the data amount of tasks, the computational density of tasks, and the data amount of service data) and the service migration decisions at time slot t-1. Therefore, the state is defined as

$$s_t = \{Loc_{u,t}, D_{u,t}, C_{u,t}, S_{u,t}, x_{u,t-1} \mid IV_u \in \mathcal{IV}\}. \quad (19)$$

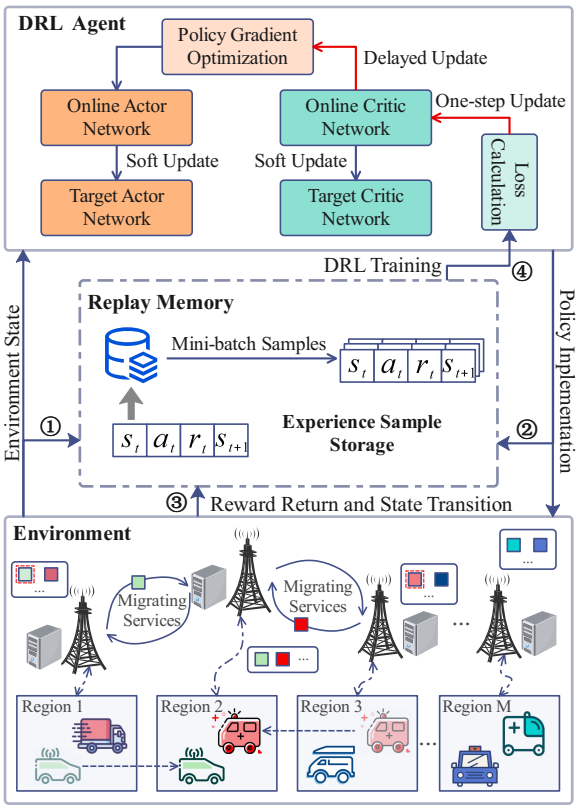

Fig. 2. The proposed improved DRL for service migration.

Action space: At the time slot t, the DRL agent takes an action of service migration a_t at s_t. For IV_u, its service instance can be migrated to another edge node. Therefore, the action is defined as

$$a_t = \{x_{u,t} \mid IV_u \in \mathcal{IV}\}. \quad (20)$$

Reward function: The reward is negatively correlated with long-term system delays including migration, communication, and computation delays, and it is defined as

$$r_t = -\sum_{IV_u \in \mathcal{IV}} (MT_{u,t} + HT_{u,t} + CT_{u,t}). \quad (21)$$

The proposed method adopts an actor-critic architecture and deep deterministic policy gradient to train and optimize service migration policies in complex and dynamic multi-edge IoV systems. Specifically, the actor generates actions of service migration, while the critic evaluates the Q-values of actions. The critic with one-step update enables fast convergence and accurately evaluates the actions, which effectively guides the actor's update and significantly reduces the errors when estimating the policy gradient.

The key steps of the proposed method are given in Algorithm 1. First, we initialize the actor network μ, critic network ψ, target actor network $\hat{\mu}$, target critic network $\hat{\psi}$, replay memory X, number of training episodes E, and the maximum time slots per episode T (Lines 1∼3). For each

Algorithm 1: Improved DRL for service migration

Input: Actor network μ , critic network ψ, target actor network $\hat{\mu}$, target critic network $\hat{\psi}$, and the multi-edge IoV system
Output: Well-trained actor network μ
1 **Initialize:** μ and ψ with θ_μ and θ_ψ
2 **Initialize:** $\hat{\mu}$ and $\hat{\psi}$ with $\theta_{\hat{\mu}} \leftarrow \theta_\mu$ and $\theta_{\hat{\psi}} \leftarrow \theta_\psi$
3 **Initialize:** Replay memory X, training episodes E, and maximum time slot per episode T
4 **for** $episode = 1, 2, ..., E$ **do**
5 Get initial state: $s_0 = env.reset()$;
6 Create service instances on edge nodes;
7 **for** $t = 1, 2, ..., T$ **do**
8 Update IVs' locations and generate service migration decision a_t: $a_t = \mu(s_t \mid \theta_\mu) + \xi$;
9 Execute service migration decision a_t;
10 **for** $each\ edge\ node\ f_m \in \mathcal{F}$ **in parallel do**
11 Derive the optimal resource allocation via convex optimization theory (Section 4.2);
12 **end**
13 Calculate reward and perform state transition: $r_t, s_{t+1} = env.step(a_t)$;
14 Store training sample: $X.push(s_t, a_t, r_t, s_{t+1})$;
15 Draw N samples from X: $N^*(s_t, a_t, r_t, s_{t+1}) = X.sample(N)$;
16 Calculate cumulative expected rewards: $y_t = r_t + \gamma\hat{\psi}(s_{t+1}, \hat{\mu}(s_{t+1} \mid \theta_{\hat{\mu}}) \mid \theta_{\hat{\psi}})$;
17 Calculate loss function and update critic: $Loss_\psi = \frac{1}{N}\sum_1^N (y_t - \psi(s_t, a_t \mid \theta_\psi))^2$;
18 **if** $t\%\lambda = 0$ **then**
19 Update actor using gradient ascent: $\nabla_{\theta_\mu} J = \frac{1}{N}\sum_1^N \nabla_{a_t}\psi(s_t, a_t \mid \theta_\psi) \nabla_{\theta_\mu}\mu(s_t \mid \theta_\mu)$;
20 Soft update target network parameters: $\theta_{\hat{\mu}} \leftarrow \omega\theta_\mu + (1-\omega)\theta_{\hat{\mu}}$, $\theta_{\hat{\psi}} \leftarrow \omega\theta_\psi + (1-\omega)\theta_{\hat{\psi}}$;
21 **end**
22 **end**
23 **end**

training episode, IVs create service instances on their nearest edge nodes (Lines 5∼6). For the time slot t, the system state s_t is input into the actor μ, which generates and executes an action of service migration a_t. Specifically, the Gaussian noise ξ is introduced to enhance the exploration capability of the policy in the initial training phase. Meanwhile, to ensure the stable convergence of the policy in the later training phase, the variance of ξ gradually declines with the increase of time slots, thereby achieving a balance between exploration and convergence. (Lines 8∼9). Next, we derive the optimal resource allocation for the service instances on each edge node via convex optimization theory (Lines 10∼12), where the details are given in Section 4.2. Then, we calculate the reward r_t and the system state transits to s_{t+1} (Line 13). Next, the training sample (s_t, a_t, r_t, s_{t+1}) is stored into the replay memory X, and we randomly draw N samples from X to train network parameters (Lines 14∼15). Notably, the correlation of training samples is broken by using the replay memory, which alleviates the instability that occurs in the training process. Then, we calculate the cumulative expected discount rewards by combining the reward r_t with the target critic $\hat{\psi}$, and we adopt the Adam optimizer to minimize the loss of the critic ψ (Lines 16∼17), enabling the updated critic can better fit in $Q^*(s_t, a_t)$. However, the critic is prone to instability, which also leads to fluctuations in the update of the actor. To solve this issue, we design a delayed update mechanism, where the actor is updated only after the critic has been updated λ times (Lines 18∼21). Specifically, the actor is used to fit a high-dimensional mapping from s_t to a_t, whose objective is defined as

$$J(\theta_\mu) = \mathbb{E}_{\theta_\mu}[\psi(s_t, \mu(s_t \mid \theta_\mu) \mid \theta_\psi)]. \tag{22}$$

For a given state s_t, the actor is updated by adjusting θ_μ, and thus its output $\mu(s_t \mid \theta_\mu)$ can be updated in an upward direction according to $\psi(s_t, \mu(s_t \mid \theta_\mu) \mid \theta_\psi)$ calculated by the critic. Specifically, the gradient ascent is used to update the actor, and the network parameters of the target actor and the target critic are updated via the soft update (Lines 19∼20). In addition, we incorporate gradient clipping during the network update to enhance model stability.

4.2 Resource Allocation with Convex Optimization

As aforementioned, *P*3 is to address the problem of resource allocation for all MEC servers, where the resource allocation processes of different MEC servers are independent and parallel. In this regard, we further separate the problem of resource allocation for each MEC server from *P*3, which can be formulated as

$$\begin{aligned} P4: \min_{e_{u,t}} \sum_{IV_u \in \mathcal{IV}_m} CT_{u,t} = \min_{e_{u,t}} \sum_{IV_u \in \mathcal{IV}_m} \frac{K_{u,t}}{e_{u,t}F} \\ \text{s.t.} \quad C2 - C3. \end{aligned} \tag{23}$$

Therefore, the solution of *P*3 relies on addressing *P*4. To optimize *P*4, we design a new method based on convex optimization theory. First, we prove that *P*4 is a convex programming problem with the constraints of *P*4 and Hessian matrix. Next, guided by convex optimization theory, we define the generalized Lagrange function and solve the KKT conditions for *P*4. Finally, we theoretically derive the optimal resource allocation for each MEC server under given decisions of service migration.

Lemma 2. *P*4 *is a convex programming problem.*

Proof. This lemma can be proved if *P*4 and its constraints are both convex functions. It can be observed that *C*2 and *C*3 are linearly constrained, reflecting the convexity. Thus, we only need to prove that *P*4 is also a convex function.

To clarify the derivation process, we reassign the index of $\mathcal{IV}_m$ into $\mathcal{Z} = \{IV_1, IV_2, ..., IV_z, ..., IV_Z\}$ and rewrite *C*2 accordingly. Therefore, Eq. (23) can be redefined as

$$\begin{aligned} \mathcal{Y}[e_{1,t}, e_{2,t}, ..., e_{Z,t}] = \min_{e_{z,t}} \sum_{IV_z \in \mathcal{Z}} \frac{K_{z,t}}{e_{z,t}F} \\ \text{s.t.} \quad C3: \sum_{IV_z \in \mathcal{Z}} e_{z,t} - 1 \le 0, t \in T, IV_z \in \mathcal{Z}, \\ C4: -e_{z,t} \le 0, t \in T, IV_z \in \mathcal{Z}, \\ C5: e_{z,t} - 1 \le 0, t \in T, IV_z \in \mathcal{Z}, \end{aligned} \tag{24}$$

where *C*2 is converted into *C*4 and *C*5, which indicate the upper and lower bounds of the proportion of allocated resources, respectively.

The Hessian matrix is a square matrix that is established by the second-order partial derivatives of a multivariate function, which reflects the local curvature of a function. The Hessian matrix of Eq. (24) is defined as

$$H = \begin{bmatrix} \frac{\partial^2 \mathcal{Y}}{\partial (e_{1,t})^2} & \frac{\partial^2 \mathcal{Y}}{\partial e_{1,t} \partial e_{2,t}} & \cdots & \frac{\partial^2 \mathcal{Y}}{\partial e_{1,t} \partial e_{Z,t}} \\ \frac{\partial^2 \mathcal{Y}}{\partial e_{2,t} \partial e_{1,t}} & \frac{\partial^2 \mathcal{Y}}{\partial (e_{2,t})^2} & \cdots & \frac{\partial^2 \mathcal{Y}}{\partial e_{2,t} \partial e_{Z,t}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{Y}}{\partial e_{Z,t} \partial e_{1,t}} & \frac{\partial^2 \mathcal{Y}}{\partial e_{Z,t} \partial e_{2,t}} & \cdots & \frac{\partial^2 \mathcal{Y}}{\partial (e_{Z,t})^2} \end{bmatrix}, \quad (25)$$

where

$$\frac{\partial^2 \mathcal{Y}}{\partial e_{z_1,t} \partial e_{z_2,t}} = \begin{cases} 0, & z_1 \neq z_2 \\ \frac{2K_{z_1,t}}{(e_{z_1,t})^3 F}, & z_1 = z_2 \end{cases}, \quad (26)$$

and $\forall IV_{z_1}, IV_{z_2} \in \mathcal{Z}$, $K_{(\cdot),t} \geq 0$ and $e_{(\cdot),t} \geq 0$.

F is a non-zero real number. Moreover, the values on the diagonal of H are all positive, and thus it is a symmetric positive definite matrix. Based on the above analysis and convex optimization theory [35], we prove that *P*4 is a convex programming problem.

Following the optimality theorem of convex programming, any feasible KKT points can be the global optimum. Further, the generalized Lagrange function is defined as

$$\begin{aligned} L = & \sum_{IV_z \in \mathcal{Z}} \frac{K_{z,t}}{e_{z,t} F} + \beta \Big(\sum_{IV_z \in \mathcal{Z}} e_{z,t} - 1 \Big) \\ & + \sum_{IV_z \in \mathcal{Z}} \eta_z (-e_{z,t}) + \sum_{IV_z \in \mathcal{Z}} \zeta_z (e_{z,t} - 1), \end{aligned} \quad (27)$$

where $\beta, \eta_z, \zeta_z, IV_z \in \mathcal{Z}$ are the Lagrange multipliers associated with *C*3, *C*4, and *C*5, respectively.

Therefore, the KKT conditions of the redefined *P*4 in Eq. (24) can be described as

$$\begin{cases} -\frac{K_{z,t}}{(e_{z,t})^2 F} + \beta - \eta_z + \zeta_z = 0, \\ \beta \big(\sum_{IV_z \in \mathcal{Z}} e_{z,t} - 1 \big) = 0, \sum_{IV_z \in \mathcal{Z}} e_{z,t} - 1 \leq 0, \\ \eta_z (-e_{z,t}) = 0, -e_{z,t} \leq 0, \\ \zeta_z (e_{z,t} - 1) = 0, e_{z,t} - 1 \leq 0, \\ \beta \geq 0, \eta_z \geq 0, \zeta_z \geq 0. \end{cases} \quad (28)$$

According to the above KKT conditions, the optimal resource allocation can be derived by

$$e_{z,t} = \frac{\sqrt{K_{z,t}}}{\sum_{IV_z \in \mathcal{Z}} \sqrt{K_{z,t}}}, IV_z \in \mathcal{Z}. \quad (29)$$

By using the proposed method, $\forall t \in T$, each MEC server can obtain optimal resource allocation for service instances (Lines 10∼12 in Algorithm 1), supporting the DRL agent learning better migration policies.

4.3 Complexity Analysis of *SR-CL*

For each training step of service migration in Algorithm 1, N samples are extracted from X to train network parameters, involving forward computation and backward propagation. Referring to [36], the time complexity of a DRL-based algorithm mainly depends on the network structures. Specifically, the actor takes the state s_t as input and outputs the action a_t, while the critic takes the concatenated vector of s_t and a_t as input and outputs the Q-values. We define the dimensions of s_t, a_t, and the two hidden layers of neural networks as $\mathcal{D}$, $\mathcal{W}$, $\mathcal{H}_1$, and $\mathcal{H}_2$, respectively. Therefore, the time complexity of training the actor using a sample is

$$O_{actor} = O\left(\mathcal{D}\mathcal{H}_1 + \mathcal{H}_1\mathcal{H}_2 + \mathcal{H}_2\mathcal{W}\right). \quad (30)$$

Similarly, the time complexity of training the critic is

$$O_{critic} = O\left((\mathcal{D} + \mathcal{W})\mathcal{H}_1 + \mathcal{H}_1\mathcal{H}_2 + \mathcal{H}_2\right). \quad (31)$$

In Algorithm 1, the critic is updated every time step, while the actor is updated only after the critic has been updated λ times, where their update frequencies are denoted as TE and TE/λ, respectively. Therefore, the time complexity of Algorithm 1 can be calculated by

$$\begin{aligned} O_{agent} = & O\Bigg(NTE\Bigg((\mathcal{D} + \mathcal{W} + \frac{\mathcal{D}}{\lambda})\mathcal{H}_1 \\ & + (1 + \frac{1}{\lambda})\mathcal{H}_1\mathcal{H}_2 + (1 + \frac{\mathcal{W}}{\lambda})\mathcal{H}_2 \Bigg) \Bigg). \end{aligned} \quad (32)$$

For the resource allocation that is performed according to Eq. (29), its time complexity is $O(1)$. If both service migration and resource allocation are incorporated into the action space of DRL, the time complexity will become

$$\begin{aligned} O'_{agent} = & O\Bigg(NTE\Bigg((\mathcal{D} + \mathcal{W} + \mathcal{W}_{RA} + \frac{\mathcal{D}}{\lambda})\mathcal{H}_1 \\ & + (1 + \frac{1}{\lambda})\mathcal{H}_1\mathcal{H}_2 + (1 + \frac{\mathcal{W} + \mathcal{W}_{RA}}{\lambda})\mathcal{H}_2 \Bigg) \Bigg), \end{aligned} \quad (33)$$

where $\mathcal{W}_{RA}$ is the dimension of resource allocation.

It is noted that the time complexity mainly depends on the first term in Eq. (33), and this term will grow rapidly if the action and state spaces become huge. Moreover, the decision space will become hybrid, and it is impractical to conduct thorough exploration in a continuous space. The strict constraints of resource allocation make it challenging to find satisfactory results, leading to massive invalid exploration and thus diminishing learning efficiency.

Therefore, by combining DRL with convex optimization theory for addressing the joint problem of service migration and resource allocation in multi-edge IoV systems, the proposed *SR-CL* not only can efficiently reach the optimal policy but also effectively reduce the exploration complexity.

4.4 Implementation of *SR-CL* in Real-world Scenarios

As illustrated in Fig. 3, the proposed *SR-CL* contains five main components including experience collector, replay memory, migration decision maker, target policy trainer, and allocation decision maker. The allocation decision maker is deployed on each edge node, while the others are deployed in the MEC controller. Specifically, the experience collector receives the state and reward information from the environment (Step 1) and sends the state to the migration decision maker (Step 2). The migration decision maker consists of

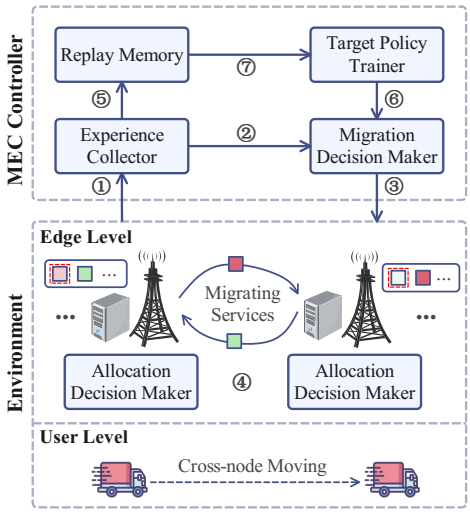

Fig. 3. Implementation of *SR-CL* in real-world scenarios.

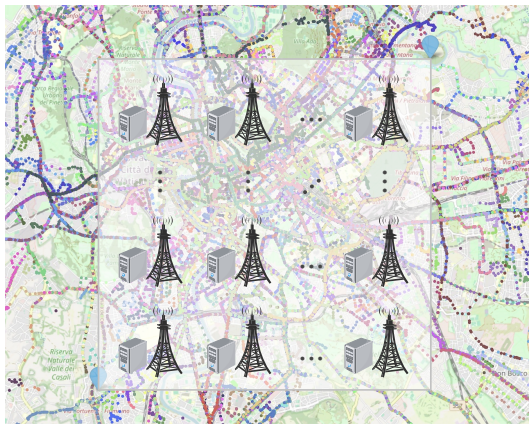

Fig. 4. Real-world vehicle trajectories and BS distribution.

the online critic and actor, which synchronizes parameters from the target policy trainer (Step 6) and makes migration decisions based on the state (Step 3). After executing the migration decisions, both tasks and service instances are transmitted to target edge nodes. Meanwhile, the allocation decision makers on each edge node perform optimization in parallel to obtain the optimal resource allocation (Step 4). Next, the experience collector sends the collected experience samples to the replay memory (Step 5), and then the target policy trainer periodically extracts samples from the replay memory for updating policies (Step 7). Therefore, when the network topology or traffic pattern changes, the proposed *SR-CL* can respond to the changes in time and maintain high generality.

5 Performance Evaluation

In this section, we evaluate and analyze the proposed *SR-CL* through extensive simulation and testbed experiments.

5.1 Experiment Setup

Datasets. We evaluate the performance of the proposed *SR-CL* using the real-world datasets of vehicle trajectories in Rome City [37]. The datasets contain the driving data from 320 taxis over 30 days, spanning from February 1st, 2014, to March 2nd, 2014. Each record includes the unique taxi ID, timestamp, and trajectory coordinates. As illustrated in Fig. 4, the scatter points with different colors indicate the trajectories of different vehicles in cities. By tracking the vehicle trajectories, we can obtain the sources and locations of different vehicles' tasks and then simulate dynamic and varying vehicle mobility patterns.

Simulation Settings. The simulation experiments are conducted on a workstation with one 8-core Intel(R) Xeon(R) Silver 4208 CPU @3.2GHz and 2 NVIDIA GeForce RTX 3090 GPUs with 32GB RAM. As shown in Fig. 4, we consider the region enclosed by the coordinate pairs [41.856, 12.442] and [41.928, 12.5387] of the city center as our experiment scenario, which encompasses the complex mobility patterns of numerous vehicles. To facilitate our analysis, we scale down this region to 16 km^2 area served by 16 edge nodes. The communication coverage of each edge node is 1 km^2. The bandwidth of each BS is 20 MHz, and the computational capability of each MEC server is 60 GHz. For the default experiment settings, we consider a high-connectivity network topology, where each node can communicate with its adjacent edge nodes. The traffic load distribution among edge nodes dynamically changes with migration decisions. At the initial, there are 100 IVs connected to their nearest edge nodes, and the MEC servers create service instances for IVs. An episode consists of 240 time slots, and IVs continuously send tasks to their service instances at fixed intervals. Based on Python 3.8 and Pytorch, we build and train the neural networks in *SR-CL*, which owns two hidden layers with 512 and 256 neurons, respectively. Moreover, the batch size N is 512, the delayed update parameter λ is 5, the soft update parameter ω is 1e-2, the learning rate of the actor and critic are 1e-5 and 1e-4, the gradient clipping value is 2.0, the Gaussian noise is $\xi \sim \mathcal{N}(0, 0.15^2)$, the size of replay memory is 1e4, and the discount factor γ is 0.95. After the *SR-CL* completes training, it can be applied to various scenarios. Other main parameters are listed in Table 1.

TABLE 1
Settings of simulation parameters

Parameter	Default value
Transmission power, P_u	$\mathcal{U}(0.4, 0.6)$ W [38]
Data amount of each task, $D_{u,t}$	$\mathcal{U}(0.5, 1.5)$ MB [38]
Computational density, $C_{u,t}$	$\mathcal{U}(200, 1000)$ cycles/bit [39]
Data amount of service data, $S_{u,t}$	$\mathcal{U}(0.5, 50)$ MB [22], [40]
Gaussian noise, σ^2	10^{-13} W [39]
Channel gain per unit distance, α	10^{-5} [39]
Transmission delay coefficient, μ^h	0.3 s
Migration delay coefficient, μ^y	1.5 s
Network bandwidth, χ	500 Mpbs [40]

Comparison Methods. We compare the *SR-CL* with the following benchmark methods to verify its superiority.

- *Always Migrate (AM)* [11]: Service instances are always migrated to the nearest edge nodes along with IVs.

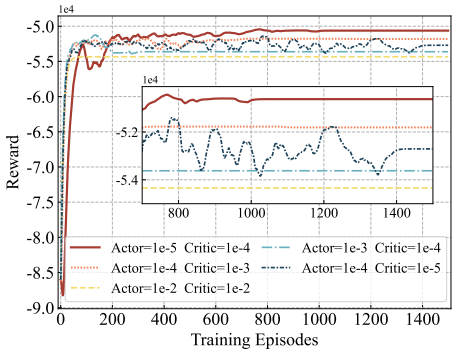

Fig. 5. Convergence vs. learning rate.

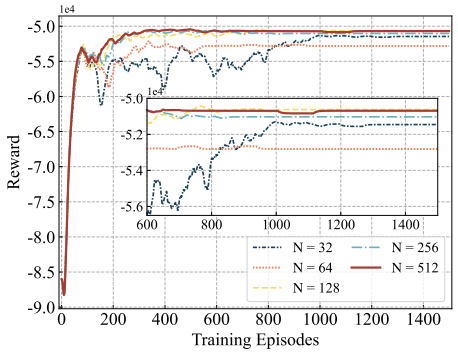

Fig. 6. Convergence vs. batch size.

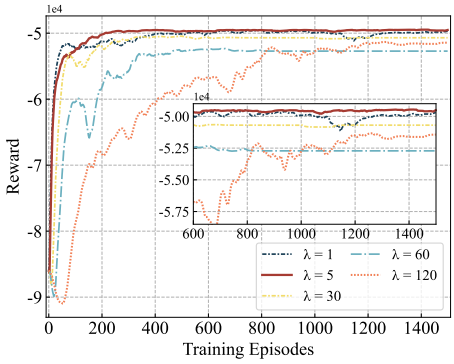

Fig. 7. Convergence vs. update frequency.

- *Never Migrate (NM)* [40]: Service instances are always located at the initial edge nodes.
- *Genetic Algorithm (GA)* [21]: The reward is defined as the fitness value, and a heuristic evolutionary algorithm is used to search for the optimal migration individual.
- *Independent Deep Q-Network (IDQN)*: As an extension of Independent Q-Learning (IQL) [41], the *IDQN* utilizes DNNs to approximate the Q-function. In the experiments, DRL agents are deployed on vehicles that make migration decisions based on local observations, which learn deterministic policies and select the actions with the maximum Q-value. There is no communication process among different DRL agents.
- *Joint optimization of Service migration and Resource allocation (JSR)*: To verify the validity of the proposed problem decoupling, DRL is individually used to optimize the joint problem of service migration and resource allocation. In the experiments, it adopts the same network structure and parameter settings as the *SR-CL*.
- *Deep Deterministic Policy Gradient (DDPG)* [18]: As an advanced DRL, it adopts deterministic policy gradient and one-step update to optimize migration policies.

In the experiments, all the comparison methods (except the *JSR*) allocate resources to service instances in proportion to task demands. To ensure the fairness of experiments, scenario settings are consistent for all methods.

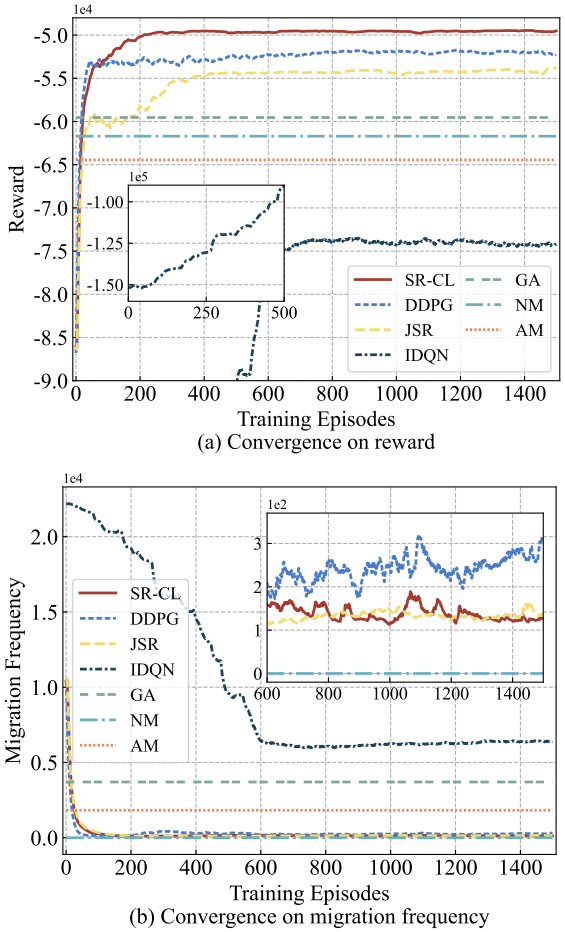

Fig. 8. Convergence comparison of different methods in terms of reward and migration frequency.

5.2 Experiment Results and Analysis

5.2.1 Hyperparameter Tunning of SR-CL

Learning Rate. As depicted in Fig. 5, we evaluate the convergence of the proposed *SR-CL* with different learning rates. It is common practice for the critic to have a higher learning rate than the actor, enabling the critic to approximate the Q-value function more accurately. When the critic's learning rate is lower than the actor's, the inaccurate estimation of Q-values leads to the persistent oscillations of the training process. Moreover, an excessively large learning rate may cause premature convergence to the local optimum. By setting the actor's learning rate lower than the critic's, the *SR-CL* achieves stable training and better convergence. Through extensive experiment tests and analysis, we set the critic's and the actor's learning rates in the *SR-CL* to 1e-4 and 1e-5, respectively.

Batch Size. As illustrated in Fig. 6, we test the effect of different batch sizes on the convergence of the proposed *SR-CL*. The results show that larger batch sizes can offer more sample information and thus improve the model's generalization ability. Moreover, the *SR-CL* with larger batch sizes yields more stable gradient estimation because it can better represent the statistical characteristics of the entire training datasets and reduce gradient variance, thereby stabilizing the training process. Specifically, when the batch size N is set to 32, the training process of the *SR-CL* exhibits

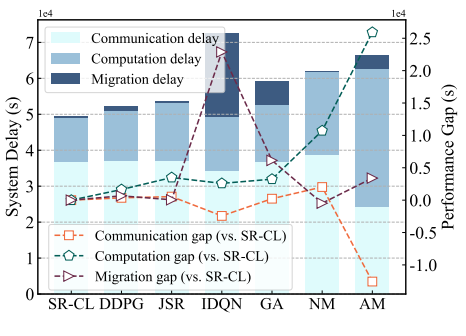

Fig. 9. Various delays of different methods with default experiment settings.

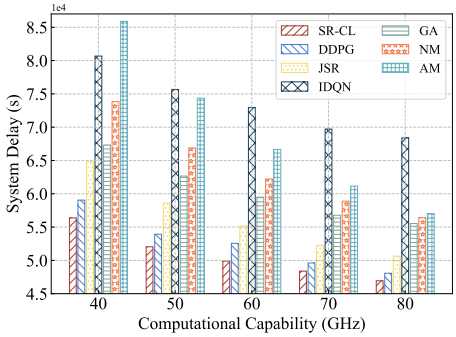

Fig. 10. Comparison of different methods with various computational capabilities.

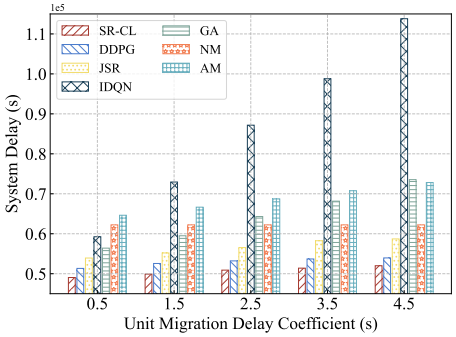

Fig. 11. Comparison of different methods with various delay coefficients.

significant fluctuations. As the batch size increases, the *SR-CL* achieves more stable convergence performance. Based on the above analysis, we set the batch size N of the *SR-CL* to 512 in the following experiments.

Update Frequency. As shown in Fig. 7, we assess the convergence of the proposed *SR-CL* with different update frequencies. Specifically, when the update frequency λ is set too small (e.g., 1), the *SR-CL* implies frequent updates of the actor by using the critic that is not well trained. This leads to inaccurate Q-value estimation, significant update fluctuations, and a potential trap in the local optimum. Conversely, if λ is set too large, the training process of the *SR-CL* will become overly sluggish. To strike a good balance between accuracy and efficiency, we set the update frequency λ to 5.

5.2.2 Convergence Comparison of Different Methods

Convergence on Reward. We compare the convergence of different methods in terms of reward. As shown in Fig. 8(a), the *AM*, *NM*, and *GA* never involve the training process of neural networks, and thus the reward remains unchanged with the increase of training episodes. The *IDQN* makes migration decisions only based on local observation but ignores other agents in the environment, deviating from the global optimum. The *AM* and *NM* make migration decisions by preset rules, working better than the *IDQN*. The *GA* only considers the instant reward without long-term perspective. In the early stage, the *SR-CL*, *DDPG*, and *JSR* do not show superior performance. As the training progresses, these three methods gradually outperform other methods by continuously exploring and learning. The *JSR* incorporates resource allocation into the action space. However, it is extremely challenging for the *JSR* to fully explore the continuous action space, causing premature convergence to the local optimum. The *DDPG* focuses on optimizing migration decisions and outperforms the *JSR*. Compared to the *DDPG*, the *SR-CL* integrates an improved DRL with convex optimization theory, achieving a more stable training process and superior convergence.

Convergence on Migration Frequency. We compare the convergence of different methods in terms of migration frequency. As illustrated in Fig. 8(b), the *IDQN* and *NM* exhibit the highest and lowest migration frequency, respectively. Commonly, IVs tend to stay in the communication coverage of a BS within consecutive time slots, and thus the migration frequency of the *AM* is low. The *GA* always makes migration decisions by searching the randomly initialized population, resulting in higher migration frequency than the *AM*. As the training progresses, the *SR-CL*, *DDPG*, and *JSR* are able to learn better policies. Due to the one-step update mechanism in *DDPG*, the critic generates inaccurate Q-value estimation for actions, which consequently affects the update of the actor in the right direction. As a result, the migration frequency of the *DDPG* remains fluctuating within a certain range. Compared to the *DDPG*, the *SR-CL* achieves better convergence performance and lower migration frequency, validating the superior design of the *SR-CL*.

5.2.3 Delay Comparison of Different Methods

Various Delays of Different Methods. As depicted in Fig. 9, we compare various delays (including communication, computation, migration, and total delays) of the *SR-CL* with other methods. Since the *NM* does not perform service migration, there is no migration delay. However, the *NM* assembles service instances on the initial edge nodes, leading to a significant delay of the backhaul link transmission as IVs move. The *AM* always migrates service instances along with the movement of IVs, which reduces communication delay but causes the overload issue and increases computation delay. The *IDQN* adopts the strategy of individual independent optimization and fails to capture the inherent complexity of the whole system, resulting in excessive migration delay. Therefore, the performance of the *IDQN* is inferior to other methods. Frequent migrations may happen in *GA*, but it is superior to the *NM*, *AM*, and *IDQN* with the guidance of prior knowledge. Compared to the above methods, the *JSR* achieves better performance. However, due to insufficient exploration in continuous space, it cannot obtain the global optimum. In contrast, the *DDPG* and *SR-CL* outperform other methods, where the *SR-CL* enables the optimal resource allocation by using convex optimization and thus exhibits the lowest total delay.

Total Delay with Various Computational Capabilities. As shown in Fig. 10, there is a downward trend in total system delay as the computational capability of each MEC server increases. This is because MEC servers can provide IVs with more available computational resources, thereby reducing the computation delay. For the *AM*, the total system delay primarily depends on computation delay caused by the overload issue (as analyzed in Fig. 9), and thus the

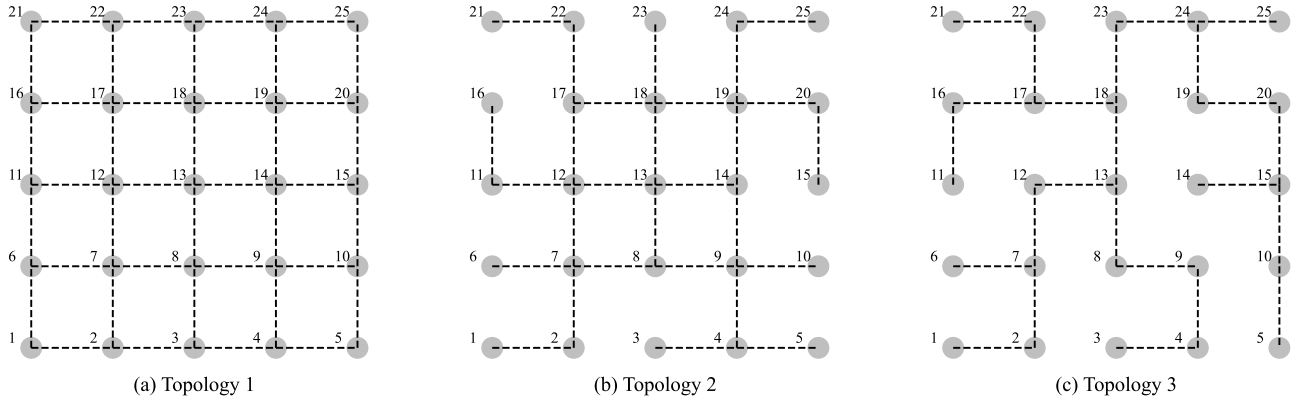

Fig. 12. Different topologies of a large-scale network with 25 edge nodes.

increase of computational capabilities can better alleviate it. However, the high communication delay in *NM* can not be mitigated, limiting the performance improvement. The *GA* and *IDQN* are able to find suitable migration strategies when the computational capability is low. With the increase of this factor, the computation delay is gradually reduced, but frequent migration leads to excessive migration delay. In contrast, the *JSR* is superior to the above methods, but it may fall into the local optimum due to improper resource allocation. The *DDPG* and *SR-CL* are capable of performing appropriate service migration. In particular, the proposed *SR-CL* introduces the convex optimization theory to achieve optimal resource allocation, thereby obtaining the best performance. Specifically, compared to the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM*, the *SR-CL* achieves the performance improvement by around 3.63%, 9.90%, 30.96%, 15.95%, 20.29%, and 26.49%, respectively.

Total Delay with Dynamic Traffic Changes. We evaluate the impact of traffic changes on system delay. Specifically, we simulate traffic changes by varying the number of tasks in each time slot. As shown in Fig. 13, system delay increases with rising traffic. This is because the growing number of tasks intensifies the resource competition. The *NM* and *AM* perform well in low-traffic scenarios, but their migration strategies cannot adapt to dynamic traffic changes. The *JSR*, *DDPG*, *GA*, and *IDQN* can adjust their migration strategies through real-time feedback, outperforming the *NM* and *AM*. The *JSR* jointly optimizes service migration and resource allocation but struggles to find the optimal joint strategy due to the large action space. In contrast, the proposed *SR-CL* decouples the sub-problem of resource allocation by introducing the convex-optimization theory and dynamically adjusts the migration strategy via the real-time feedback mechanism of DRL. Therefore, the *SR-CL* can effectively fit into dynamic traffic changes. Specifically, compared to the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM*, the *SR-CL* achieves performance improvement of approximately 2.77%, 14.09%, 20.11%, 13.86%, 21.98%, and 36.15%, respectively.

Total Delay with Various Unit Migration Delay Coefficients. The unit migration delay coefficient is an essential factor that affects the performance of service migration. As illustrated in Fig. 11, when there is no service migration by using the *NM*, the change of the coefficient does not influence the system's total delay. With the increase of the coefficient, the performance of all methods (except the *NM*) exhibits a descending trend. When the value of the coefficient grows to 4.5, both the *GA* and *IDQN* become inferior to the *NM* and *AM*. Moreover, the *IDQN* performs worst because it lacks communication between agents and thus cannot capture the complexity and dynamics of the system. In contrast, the proposed *SR-CL* designs the critic with the one-step update to guide the actor with the delayed update for accurately evaluating the Q-values of migration actions. Therefore, the *SR-CL* can always make proper migration decisions to optimize the long-term system delay and outperform the *DDPG* and *JSR*. Especially, when the value of the coefficient is 4.5, the *SR-CL* can reach the performance improvement by approximately 3.62%, 11.42%, 54.31%, 29.29%, 16.42%, and 28.59% compared to the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM*, respectively.

Total Delay with Various Network Topologies. As shown in in Fig. 12, to further evaluate the generality of the proposed *SR-CL* for different network topologies, we construct 3 different topologies of a large-scale network with 25 edge nodes as follows.

- Topology 1 (high-connectivity): Each edge node can communicate with its adjacent edge nodes, and thus it is a high-connectivity topology.
- Topology 2 (middle-connectivity): The edge nodes in the central region are with a high-connectivity topology, but the connectivity between edge nodes in the other region is sparse, simulating the typical network topology in cities.
- Topology 3 (low-connectivity): The connectivity between edge nodes is sparse and there is no closed loop, simulating the network topology in remote areas.

Fig. 14 illustrates the total delay of different methods with various network topologies. As the connectivity of a network topology decreases, the performance of all methods exhibits a downward trend. This is because service migration and task transmission need to be routed via more hops, leading to more time consumption. The *IDQN* lacks efficient collaboration between agents, and thus the decrease in connectivity causes great performance degradation. Since the *AM* and *NM* rely on preset rules, and the decrease in connectivity increases the routing hops of service migration and task transmission. The *GA* employs heuristics

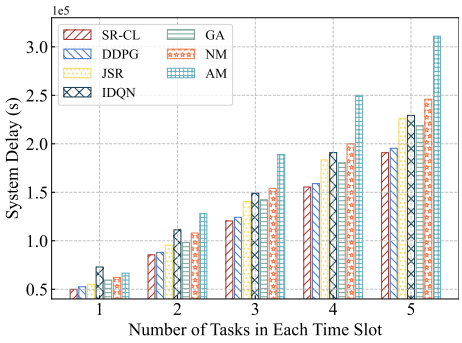

Fig. 13. Comparison of different methods with dynamic traffic changes.

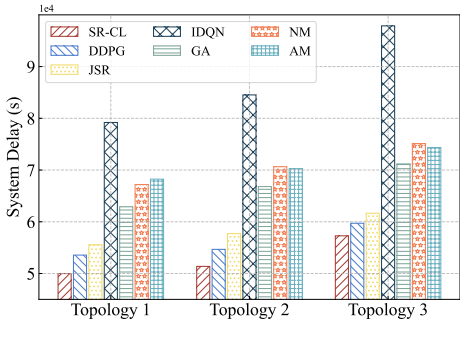

Fig. 14. Comparison of different methods with various network topologies.

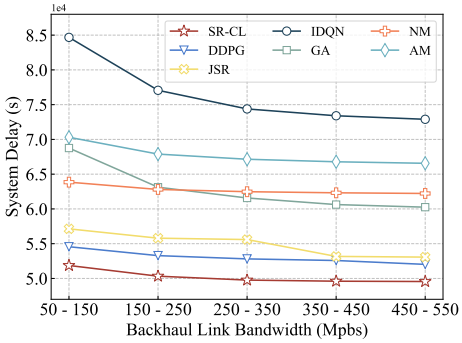

Fig. 15. Comparison of different methods with various backhaul link bandwidths.

with multiple iterations, outperforming the above methods. The *SR-CL*, *DDPG*, and *JSR* adapt to changes in network topologies by interacting with environments. The proposed *SR-CL* utilizes the convex optimization theory to obtain the optimal resource allocation, achieving better performance than the *DDPG* and *JSR* under similar migration decisions. Specifically, compared to the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM*, the *SR-CL* achieves the performance improvement by approximately 5.53%, 9.29%, 39.35%, 21.03%, 25.50%, and 25.46%, respectively.

Total Delay with Various Backhaul Link Bandwidths. We evaluate the impact of varying backhaul link bandwidths on the performance of different methods. As shown in Fig. 15, the performance of all methods exhibits an upward trend as the bandwidth increases. This is because high bandwidth reduces the time of transmitting data on the backhaul link, where services instances own more data volumes than tasks and thus consume more transmission time. The *IDQN* fails to capture system dynamics, resulting in frequent service migrations. The *GA* makes migration decisions by randomly searching initialized populations, which also leads to high migration frequency. In the scenarios with low bandwidth, the *GA* performs worse than the *NM*. Compared to the above methods, the *SR-CL*, *DDPG*, and *JSR* can better adapt to the changes of backhaul link bandwidths and avoid unnecessary service migrations. The *DDPG* allocates computational resources in proportion, while the *SR-CL* adopts the convex optimization theory for resource allocation that can further improve system performance. Specifically, compared to the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM*, the *SR-CL* achieves the performance improvement by about 5.33%, 8.61%, 34.34%, 20.11%, 19.94%, and 25.86%, respectively.

Average Task Response Delay with Various Numbers of IVs. As shown in Table 2, when the number of IVs is small, MEC servers possess sufficient resources that can be allocated to service instances, and thus there is no obvious performance gap among different methods. As the number of IVs grows, it becomes imperative to properly migrate service instances for avoiding high computation delay caused by the overloads of MEC servers. Additionally, migration policy also need to consider the high mobility of vehicles to mitigate high communication delay. The *IDQN* lets each agent make migration decisions without communicating with others, causing the overload issue and local optimum.

TABLE 2
Average task response delay (s) with various numbers of IVs

Methods	Numbers of IVs				
	60	100	140	180	220
SR-CL	**1.5302**	**2.0876**	**2.6381**	**3.1579**	**3.7279**
DDPG	1.5871	2.1991	2.7106	3.2505	3.8374
JSR	1.5778	2.3093	2.8611	3.4857	4.2280
IDQN	2.4014	3.0519	3.4952	4.2332	4.8808
GA	2.1002	2.4895	2.9957	3.4720	4.0633
NM	1.9002	2.6032	3.1612	3.7369	4.3681
AM	1.8455	2.7887	3.7321	4.5315	5.6026

Due to incomplete exploration of continuous space, the *JSR* is inferior to the *GA* as the number of IVs increases. In contrast, the *DDPG* and *SR-CL* consider the interaction among IVs by observing the global state, enabling better migration decisions. Compared to the *DDPG*, the *SR-CL* incorporates the convex optimization theory to obtain the optimal resource allocation on each MEC server, contributing to the improved performance of service migration. Especially, when the number of IVs is 220, the *SR-CL* outperforms the *DDPG*, *JSR*, *IDQN*, *GA*, *NM*, and *AM* by around 2.85%, 11.83%, 23.62%, 8.25%, 14.66%, and 33.46%, respectively.

TABLE 3
Decision-making time (ms) of different methods with various numbers of IVs

Methods	Numbers of IVs				
	60	100	140	180	220
SR-CL $(\times 10^{-1})$	7.8726	7.8999	8.0534	8.5952	8.6426
DDPG $(\times 10^{-1})$	7.4958	7.7554	8.0017	8.0030	8.1746
JSR $(\times 10^{-1})$	8.1798	8.6549	8.8191	9.5902	9.9233
IDQN $(\times 10^{-1})$	1.5480	1.5108	1.5081	1.5080	1.5173
GA $(\times 10^{3})$	1.3669	2.4292	3.4660	4.3594	5.1198
NM $(\times 10^{-4})$	4.9978	5.2472	4.7883	5.1375	5.0477
AM $(\times 10^{-2})$	2.8634	2.8672	2.8357	2.7956	2.7983

Decision-making Time with Various Numbers of IVs. As shown in Table 3, we evaluate the decision-making time of different methods with various numbers of IVs in a large-scale network topology (25 edge nodes). The *GA* searches for the optimal solution by performing mutation and crossover

operations on the individuals in the population. When the number of IVs increases, the time of computing the population fitness significantly increases, degrading the decision-making efficiency. The *NM* always maintains the original migration decisions, and the *AM* makes the migration decisions simply based on the distance between IVs and edge nodes. Therefore, there is no obvious change in the decision-making time of these two methods as the number of IVs increases. The DRL-based methods perform training and inference via neural networks, leading to longer decision-making time than the rule-based methods. The *IDQN* makes decisions only based on local observations, and thus the input and output dimensions of each agent are not affected by the number of IVs. Although the *IDQN* exhibits lower decision-making time than the other DRL-based methods, its decision-making performance is unsatisfying, causing excessive delay. The *SR-CL*, *DDPG*, and *JSR* consider the global environment state as input of neural networks, and thus their state dimensions are related to the number of IVs. The *JSR* learns the policy of resource allocation by neural networks, thereby causing longer decision-making time. Compared to other methods, the proposed *SR-CL* is able to achieve significant performance improvement with satisfying decision-making time.

5.2.4 Ablation Experiments for SR-CL

We conduct ablation experiments to verify the effectiveness of the components designed in *SR-CL*. As shown in Fig. 16, we revert the delayed update to the one-step update, and thus the actor and critic are updated simultaneously, which leads to the high similarity between the weights of the online and target networks. Under this setting, the biased estimation of the target Q-values occurs, which causes inaccurate guidance for the action selection in the actor and seriously impacts the convergence stability of the training process. This instability affects the robustness of the *SR-CL* and makes it unsuitable for dynamic multi-edge IoV systems. Furthermore, we modify the convex-optimization-based resource allocation in *SR-CL* to the proportional resource allocation, which allocates resources to service instances in proportion to task demands. However, this manner might not fully optimize the performance of service migration. In contrast, when adopting the convex-optimization-based resource allocation, the *SR-CL* can guarantee the global optimal solution with the support of the mathematical theory. Therefore, the *SR-CL* is able to promise high adaptability in addressing the complex problem of service migration and resource allocation in changeable multi-edge IoV systems.

5.3 Simu5G-based Testbed Validation

Simu5G [42] is an open-source framework built on OMNeT++, which is designed to simulate the key features and complex scenarios of 5G networks. Simu5G supports the simulation of modules such as the 5G Radio Access Network (RAN), Core Network (CN), and Multi-access Edge Computing (MEC), which can be used to simulate the communication interactions among User Equipments (UEs), base stations (i.e., *gNodeBs* (*gNBs*)), edge nodes, and the remote cloud. Based on a PC with an Intel(R) Core(TM) i5-12400F CPU and 16GB RAM, we install OMNeT++ 6.1.0 and

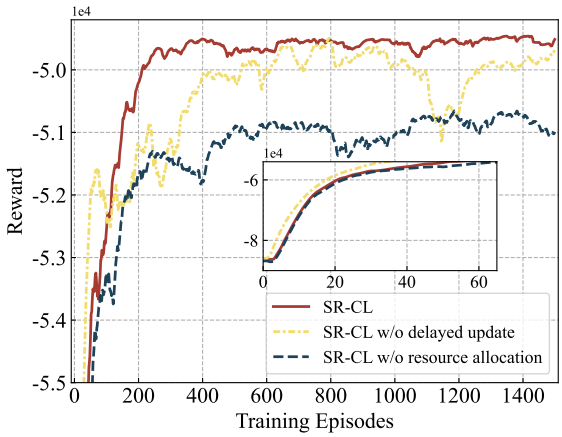

Fig. 16. Ablation experiments for the proposed *SR-CL*.

INET 4.5.2 and build a Simu5G-based testbed[1]. As shown in Fig. 17, the topology of the Simu5G-based testbed includes mobile UEs, *gNBs* that provide wireless access, *mecHost* that offers computing services, User Plane Function (UPF) and Intermediate UPF (IUPF) for transmitting routing data between the access network and edge nodes, User Application Layer Control and Management Plane (UALCMP) for application-level control and management, *mecOrchestrator* for application deployment, resource allocation, and service scheduling, and *channelControl* for channel management.

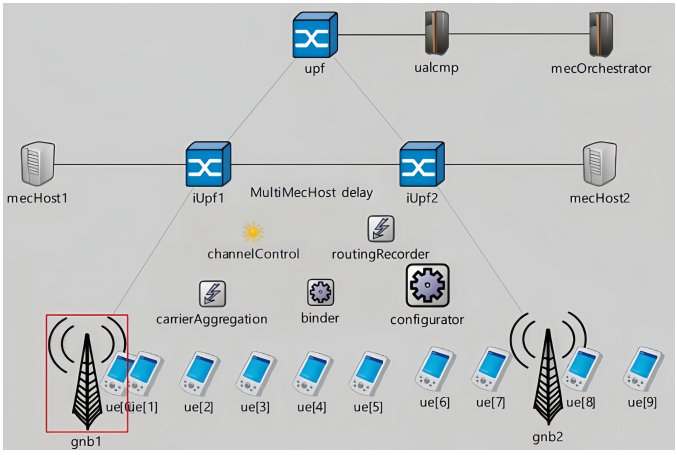

Fig. 17. Topology of Simu5G-based testbed.

On the Simu5G-based testbed, we set the transmission power to 26 dBm for UEs and 46 dBm for *gNBs* and turn on the interference in uplink and downlink. We randomly place 10 UEs within the communication coverage areas of 2 *gNBs* with different distances to *gNBs*. All UEs follow the movement pattern of *LinearMobility* with a speed of 5 m/s, *updateInterval* is 0.05 s, and *dynamicCellAssociation* and *enableHandover* are enabled. We regard *requestResponseApp* provided by Simu5G as the service requested by UEs, which is deployed in *mecHost* with *requestServiceTime* of 4 ms and subscriptionServiceTime of 11 µs. With the ETSI GS MEC 003 standard, Simu5G uses Virtualization Infrastructure

1. https://github.com/JoyZhang20/SRCL_Simu5G

Manager (VIM), a part of *mecOrchestrator*, to allocate, manage, and release virtualized infrastructure resources. VIM has two built-in scheduling strategies (i.e., *SEGREGATION* and *FAIR*). The *SEGREGATION* allocates resources strictly according to predefined demands, and the *FAIR* considers both demands and current loads to allocate resources proportionally. We implement the resource allocation policy based on the convex-optimization theory in *calculateProcessingTime* function of VIM. For service migration, we construct the DRL agent in Python programs, treating Simu5G as the environment and making interactions. Simu5G offers many delay metrics including *downLinkTime*, *upLinkTime*, *serviceResponseTime*, *processingTime*, and *responseTime*, where *responseTime* is the total delay. During the offline training phase, we randomly make service migration decisions and collect delay data after each simulation to train the DRL agent. Due to the limited compatibility of Simu5G with mainstream communication frameworks (e.g., curl and Boost.Asio), we use local files to transfer the decisions of the DRL agent during the real-time evaluation phase and synchronize Python programs and Simu5G. Other parameters are set by using the default configurations of Simu5G.

First, we compare the response time (i.e., *responseTime*) of different methods with various vehicle speeds. As illustrated in Fig. 18, the *SEGREGATION* exhibits the highest *responseTime*. This is because the *SEGREGATION* employs static resource allocation but ignores the load of *mecHost*, and thus some tasks cannot be allocated with proper resources. Different from the *SEGREGATION*, the *FAIR* considers the load variations of *mecHost* and allocates resources proportionally, thereby enhancing resource efficiency and reducing *responseTime* to a certain extent. Compared to other methods, the proposed *SR-CL* can allocate resources according to task features, which further improves resource efficiency and *responseTime*. Meanwhile, the *SR-CL* incorporates proactive service migration, which can make a better balance between migration delay and load status, and then it can choose a more reasonable *mecHost* with lower delay for offloading tasks. As the vehicle speed increases, *responseTime* of all methods rises. On one hand, faster movement causes signal instability, increasing both *downLinkTime* and *upLinkTime*. On the other hand, vehicle mobility between different *gNBs* introduces extra *responseTime*. The results demonstrate the superior performance of the *SR-CL* under scenarios with varying vehicle speeds.

Next, we compare the uplink time (i.e., *upLinkTime*) on different locations of vehicles. As depicted in Fig. 19, each box indicates the distribution of *upLinkTime* during 15 offloading processes between vehicles and *mecHost*. *UpLinkTime* is closely related to the locations of vehicles. For example, the vehicles that are closer to *gNBs* experience less signal attenuation and faster upload rates, leading to lower *upLinkTime* and more concentrated distributions. In contrast, the vehicles that are farther from *gNBs* face more severe signal attenuation and more interference during movement, resulting in greater fluctuations in upload rates, higher *upLinkTime*, and more dispersed distributions. When vehicles are located at the boundary of the communication coverage of two *gNBs* (i.e., index = 5), the highest *upLinkTime* happens. This is because vehicles trigger the switching of *gNBs* during movement, which introduces extra communication overheads and further increases *upLinkTime*.

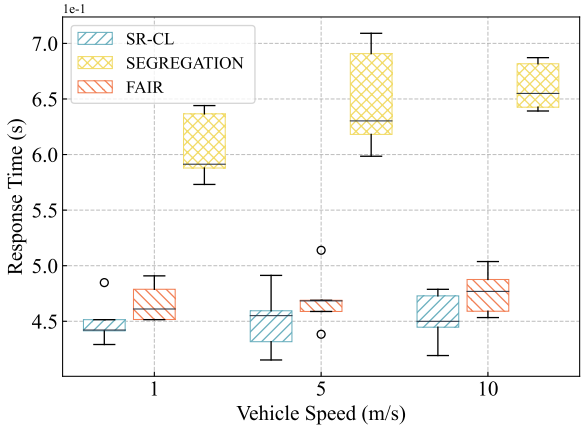

Fig. 18. Comparison of different methods with various vehicle speeds.

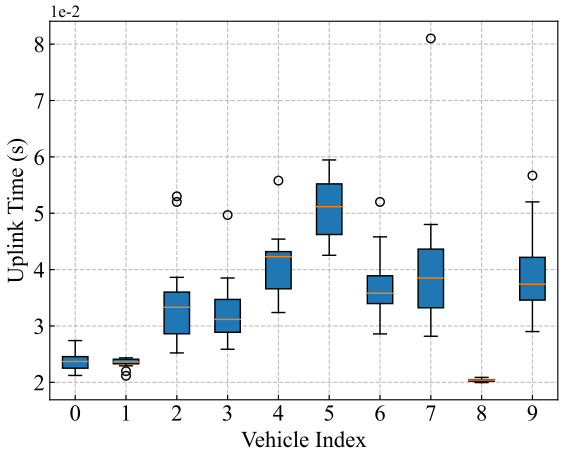

Fig. 19. Comparison of uplink time on different locations of vehicles.

6 Conclusion

In this paper, we propose *SR-CL*, a novel mobility-aware seamless service migration and resource allocation framework for multi-edge IoV systems. First, we formulate the service migration and resource allocation as a long-term QoS optimization problem and decouple it into two subproblems. For the sub-problem of service migration, we design a new improved DRL-based method with delayed and one-step update mechanisms. For the sub-problem of resource allocation, we theoretically derive the optimal resource allocation based on convex optimization. Using the real-world datasets of vehicle trajectories and testbed, extensive experiments are conducted to verify the effectiveness of the proposed *SR-CL*. Compared to benchmark methods, the *SR-CL* achieves better performance under different scenarios with various computational capabilities, unit migration delay coefficients, network topologies, backhaul link bandwidths, and numbers of IVs. Meanwhile, the *SR-CL* exhibits faster and more stable convergence than the advanced *IDQN* and *DDPG*. Further, testbed experiments validate the practicality and superiority of the *SR-CL* in offering seamless services. In our future work, we plan

to integrate energy costs into our optimization objective and explore a distributed solution to better balance delay and energy costs, aiming to further enhance the scalability of the proposed system in real-world scenarios. Moreover, we will explore more recent datasets of vehicle trajectories from different cities to evaluate the generality of the *SR-CL* in various scenarios and extend the testbed for further approaching real-world environments.

REFERENCES

[1] D. Pliatsios, P. Sarigiannidis, T. D. Lagkas, V. Argyriou, A. A. Boulogeorgos, and P. Baziana, "Joint wireless resource and computation offloading optimization for energy efficient internet of vehicles," *IEEE Transactions on Green Communications and Networking (TGCN)*, vol. 6, no. 3, pp. 1468–1480, 2022.

[2] Z. Ning, P. Dong, X. Wang, X. Hu, J. Liu, L. Guo, B. Hu, R. Y. K. Kwok, and V. C. M. Leung, "Partial computation offloading and adaptive task scheduling for 5g-enabled vehicular networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 21, no. 4, pp. 1319–1333, 2022.

[3] G. Luo, H. Zhou, N. Cheng, Q. Yuan, J. Li, F. Yang, and X. Shen, "Software-defined cooperative data sharing in edge computing assisted 5g-vanet," *IEEE Transactions on Mobile Computing (TMC)*, vol. 20, no. 3, pp. 1212–1229, 2021.

[4] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Transactions on Mobile Computing (TMC)*, vol. 21, no. 11, pp. 3836–3851, 2022.

[5] G. Luo, C. Shao, N. Cheng, H. Zhou, H. Zhang, Q. Yuan, and J. Li, "Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 42, no. 1, pp. 207–222, 2024.

[6] Y. Dai, L. Lyu, N. Cheng, M. Sheng, J. Liu, X. Wang, S. Cui, L. Cai, and X. Shen, "A survey of graph-based resource management in wireless networks -part ii: Learning approaches," *IEEE Transactions on Cognitive Communications and Networking (TCCN)*, pp. 1–1, 2024.

[7] X. Zhou, S. Ge, T. Qiu, K. Li, and M. Atiquzzaman, "Energy-efficient service migration for multi-user heterogeneous dense cellular networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 22, no. 2, pp. 890–905, 2023.

[8] W. Zhang, J. Luo, L. Chen, and J. Liu, "A trajectory prediction-based and dependency-aware container migration for mobile edge computing," *IEEE Transactions on Services Computing (TSC)*, vol. 16, no. 5, pp. 3168–3181, 2023.

[9] T. Kim, S. D. Sathyanarayana, S. Chen, Y. Im, X. Zhang, S. Ha, and C. Joe-Wong, "Modems: Optimizing edge computing migrations for user mobility," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1159–1168, 2022.

[10] S. Velrajan and V. Ceronmani Sharmila, "Qos-aware service migration in multi-access edge compute using closed-loop adaptive particle swarm optimization algorithm," *Journal of Network and Systems Management (JNSM)*, vol. 31, no. 17, 2022.

[11] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 36, no. 10, pp. 2333–2345, 2018.

[12] Z. Liang, Y. Liu, T. Lok, and K. Huang, "Multi-cell mobile edge computing: Joint service migration and resource allocation," *IEEE Transactions on Wireless Communications (TWC)*, vol. 20, no. 9, pp. 5898–5912, 2021.

[13] H. Liu, X. Long, Z. Li, S. Long, R. Ran, and H. Wang, "Joint optimization of request assignment and computing resource allocation in multi-access edge computing," *IEEE Transactions on Services Computing (TSC)*, vol. 16, no. 2, pp. 1254–1267, 2023.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[15] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing (TMC)*, vol. 20, no. 3, pp. 939–951, 2021.

[16] C. Zhang and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Generation Computer Systems (FGCS)*, vol. 96, pp. 111–118, 2019.

[17] Y. Peng, L. Liu, Y. Zhou, J. Shi, and J. Li, "Deep reinforcement learning-based dynamic service migration in vehicular networks," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2019.

[18] D. Lan, A. Taherkordi, F. Eliassen, Z. Chen, and L. Liu, "Deep reinforcement learning for intelligent migration of fog services in smart cities," in *Algorithms and Architectures for Parallel Processing: 20th International Conference (ICA3PP)*, pp. 230–244, 2020.

[19] Q. Wei, C. Haiming, W. Lei, X. Yinshui, N. Alfredo, and P. Antonio, "Mcotm: Mobility-aware computation offloading and task migration for edge computing in industrial iot," *Future Generation Computer Systems (FGCS)*, vol. 151, pp. 232–241, 2024.

[20] D. Scotece, C. Fiandrino, and L. Foschini, "Handling data handoff of ai-based applications in edge computing systems," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 20, no. 4, pp. 4435–4447, 2023.

[21] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing," *Computer Networks (CN)*, vol. 194, p. 108146, 2021.

[22] Y. Zhang, R. Li, Y. Zhao, and R. Li, "Deep reinforcement learning based mobility-aware service migration for multi-access edge computing environment," in *IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2022.

[23] I. Labriji, F. Meneghello, D. Cecchinato, S. Sesia, E. Perraud, E. C. Strinati, and M. Rossi, "Mobility aware and dynamic migration of mec services for the internet of vehicles," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 18, no. 1, pp. 570–584, 2021.

[24] G. Perin, F. Meneghello, R. Carli, L. Schenato, and M. Rossi, "Ease: Energy-aware job scheduling for vehicular edge networks with renewable energy resources," *IEEE Transactions on Green Communications and Networking (TGCN)*, vol. 7, no. 1, pp. 339–353, 2023.

[25] N. Su, J.-B. Wang, Y. Chen, H. Yu, C. Ding, Y. Pan, and J. Wang, "Joint mu-mimo precoding and computation optimization for energy efficient industrial iot with mobile edge computing," *IEEE Transactions on Green Communications and Networking (TGCN)*, vol. 7, no. 3, pp. 1472–1485, 2023.

[26] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications (TCOM)*, vol. 66, no. 4, pp. 1594–1608, 2018.

[27] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things (IoT) Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[28] Q. Li and H. Shao, "Cooperative resource allocation for computation offloading in mobile-edge computing networks," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, 2021.

[29] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 2467–2475, 2019.

[30] L. Huang, S. Bi, and Y. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing (TMC)*, vol. 19, no. 11, pp. 2581–2593, 2020.

[31] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things (IoT) Journal*, vol. 9, no. 2, pp. 1517–1530, 2022.

[32] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-uav-assisted iot networks: A deep reinforcement learning approach," *IEEE Internet of Things (IoT) Journal*, vol. 8, no. 15, pp. 12203–12218, 2021.

[33] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 68, no. 1, pp. 856–868, 2019.

[34] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized

primal decomposition approach," *Mathematical research*, vol. 35, pp. 376–386, 1987.
[35] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[36] J. Du, Z. Kong, A. Sun, J. Kang, D. Niyato, X. Chu, and F. R. Yu, "Maddpg-based joint service placement and task offloading in mec empowered air–ground integrated networks," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 10600–10615, 2024.
[37] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "Crawdad roma/taxi," 2022. Accessed: Aug. 10, 2023. [Online]. Available: https://dx.doi.org/10.15783/C7QC7M.
[38] R. Chai, M. Li, T. Yang, and Q. Chen, "Dynamic priority-based computation scheduling and offloading for interdependent tasks: Leveraging parallel transmission and execution," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 70, no. 10, pp. 10970–10985, 2021.
[39] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for uav-assisted mobile edge computing: a deep deterministic policy gradient approach," *Wireless Networks*, vol. 27, no. 4, pp. 2991–3006, 2021.
[40] 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 (TMC)*, vol. 22, no. 11, pp. 6663–6675, 2023.
[41] Z. Yao, S. Xia, Y. Li, and G. Wu, "Cooperative task offloading and service caching for digital twin edge networks: A graph attention multi-agent reinforcement learning approach," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 41, no. 11, pp. 3401–3413, 2023.
[42] G. Nardini, G. Stea, and A. Virdis, "Simu5G: Simulator for 5g new radio networks." Accessed: Sept. 10, 2024. [Online]. Available: https://simu5g.org.

Zheyi Chen is a Professor and Qishan Scholar with the College of Computer and Data Science at the Fuzhou University, China. He received his Ph.D. degree in Computer Science from the University of Exeter, UK, in 2021, and his M.Sc. degree in Computer Science and Technology from the Tsinghua University, China, in 2017, respectively. His research interests include cloud-edge computing, resource optimization, deep learning, and reinforcement learning. Dr. Chen has published over 40 research papers in reputable international journals and conferences such as IEEE TPDS, IEEE JSAC, IEEE TMC, IEEE/ACM TON, IEEE INFOCOM, ACM SIGKDD, IEEE TII, IEEE ComMag, IEEE TCC, IEEE IOTJ, and IEEE ICC.

Sijin Huang is currently working toward his M.S. degree in Computer Applied Technology with the College of Computer and Data Science at the Fuzhou University. He received his B.S. degree in Computer Science and Technology from Huaqiao University, China, in 2022. His current research interests include cloud/edge computing, deep reinforcement learning and service migration.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the Faculty of Environment, Science and Economy at the University of Exeter, United Kingdom. He received the Ph.D. degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include future Internet, computer networks, wireless communications, multimedia systems, information security, high-performance computing, ubiquitous computing, modelling, and performance engineering.

Zhonglong Ning (Senior Member, IEEE) received the Ph.D. degree from Northeastern University, China in 2014. He was a Research Fellow at Kyushu University from 2013 to 2014, Japan. Currently, he is a full professor with the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include mobile edge computing, 6G networks, machine learning, and resource management. He has published over 150 scientific papers in international journals and conferences. Dr. Ning serves as an associate editor or guest editor of several journals, such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Social Computational Systems, IEEE Internet of Things Journal and so on. He is a Highly Cited Researcher (Web of Science) since 2020.

Jie Li (Fellow, IEEE) received the B.E. degree in computer science from Zhejiang University, Hangzhou, China, the M.E. degree in electronic engineering and communication systems from China Academy of Posts and Telecommunications, Beijing, China. He received the Dr. Eng. degree from the University of Electro-Communications, Tokyo, Japan. He is with Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China where he is a chair professor. His current research interests are in big data and AI, blockchain, edge computing, networking and security, OS, information system architecture. He serves as the director of Shanghai Jiao Tong University Blockchain Research Centre. He was a professor in Department of Computer Science, University of Tsukuba, Japan. He was a visiting Professor in Yale University, USA, Inria Sophia Antipolis and Inria Grenoble-Rhone-Aples, France. He is the co-chair of IEEE Technical Community on Big Data and the founding Chair of IEEE ComSoc Technical Committee on Big Data. He serves as an associated editor for many IEEE journals and transactions. He has also served on the program committees for several international conferences.

Yan Zhang (Fellow, IEEE) received the Ph.D. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. He is currently a Full Professor with the Department of Informatics, University of Oslo, Oslo, Norway. His research interests include next-generation wireless networks leading to 6G and green and secure cyber-physical systems (e.g., smart grid and transport). He was a recipient of the Global Highly Cited Researcher Award (Web of Science top 1% most cited worldwide), since 2018. He is a Symposium/Track Chair in a number of conferences, including IEEE ICC 2021, IEEE Globecom 2017, IEEE PIMRC 2016, and IEEE SmartGridComm 2015. He is the Chair of IEEE Communications Society Technical Committee on Green Communications and Computing. He is an Editor (or an Area Editor, a Senior Editor, and an Associate Editor) for several IEEE Transactions/magazines, 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 Systems Journal, IEEE Vehicular Technology Magazine, and IEEE Blockchain Technical Briefs. He is a CCF Senior Member, an Elected Member of CCF Technical Committee of Blockchain, and a CCF Distinguished Speaker in 2019. He is a Fellow of IET and an Elected Member of Academia Europaea and the Norwegian Academy of Technological Sciences.