Computing and Communication Cost-Aware Service Migration Enabled by Transfer Reinforcement Learning for Dynamic Vehicular Edge Computing Networks

Yan Peng, Xiaogang Tang, Yiqing Zhou, *Senior Member, IEEE*, Jintao Li, *Member, IEEE*, Yanli Qi, Ling Liu, and Hai Lin

Abstract—Due to the high mobility of vehicles, service migration is inevitable in vehicular edge computing (VEC) networks. Frequent service migrations incur prohibitive migration cost including the computing cost (e.g., increased computing delay) and communication cost (e.g., occupied backhaul bandwidth). Yet existing service migration schemes are usually designed without considering the impact of the computing cost. This paper considers the impact of computing and communication cost jointly, and proposes a computing and communication cost-aware service migration scheme for VEC networks (i.e., CA-migration). Taking the service delay as a QoS metric for VEC networks, this paper formulates a migration optimization problem aiming to maximize the services' satisfaction degree of delay (i.e., the probability that the service delay is smaller than the service delay requirement), where both the communication cost and computing cost affect the services' satisfaction degree. Since the optimization problem is a constrained non-linear integer programming problem, it is difficult to solve. Moreover, the VEC networks are highly dynamic. Thus, a fast transfer reinforcement learning (fast-TRL) method combining transfer learning and reinforcement learning is proposed to provide an adaptive service migration scheme in dynamic VEC networks. Simulation results show that compared with existing schemes, the proposed CA-migration scheme can increase the satisfaction degree by up to 30%, and needs 25% less training time to obtain the optimal service migration policy.

Index Terms—Vehicular edge computing, service migration, computing cost, services' satisfaction degree, fast transfer reinforcement learning

1 Introduction

Recently, with the rapid development of unmanned vehicles and mobile communications [1], [2], [3], [4], [5], vehicular networking has become a hot research topic to support various vehicular services. Many vehicular services are compute-intensive with massive data to be processed in real-time [6], [7]. For example, one traffic safety service, pre-sense crash warning, requires intensive computing of 10^6 million instructions per second, and an end-to-end delay of less than 20 ms [6]. However, the computing capacity of the on-board processor is about $5*10^4$ million instructions per second [6], [7], which cannot meet the intensive computing requirement of the pre-sense crash warning service. Although the service can be offloaded to the cloud, the service delay is about 50-100 ms, which is too high to meet the delay requirements [8].

To satisfy such rigorous quality of service (QoS) requirements, vehicular edge computing (VEC) has been proposed to enhance computing capabilities and reduce the service latency [6], with computing servers deployed at the edge of connected vehicle networks such as base stations (BSs). However, due to the limited coverage of a VEC server, vehicles with high mobility may leave the coverage of the VEC server to whom they have offloaded tasks [9]. As the vehicle moves, the distance between the serving VEC server and the vehicle increases. Hence, the service delay could be high and the service QoS will be seriously degraded [9], [10], [11]. In order to improve the service QoS, service migration caused by high mobility is inevitable, and an efficient service migration policy should be designed [12], [13]. Although service migration has been conducted extensively in cloud datacenters, they cannot be employed in VEC networks. This is because the research motivations of service migration in datacenters and in VEC networks are totally different. The former targets at power consumption reduction, resources utilization optimization, and load balancing [14], [15]. However, the latter is triggered

- *Yan Peng, Yiqing Zhou, Jintao Li, Yanli Qi, and Ling Liu are with the State Key Lab of Processors, Institute of Computing Technology, Beijing 100190, China, and with the University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Beijing 100190, China. E-mail: {pengyan, zhouyiqing, jtli, qiyanli, liuling}@ict.ac.cn.*
- *Xiaogang Tang and Hai Lin are with the School of Aerospace Information, Space Engineering University, Beijing 100015, China. E-mail: {titantxg, 13838886000}@163.com.*

Manuscript received 14 March 2022; revised 3 November 2022; accepted 15 November 2022. Date of publication 28 November 2022; date of current version 5 December 2023.
This work was supported by the National Key Research and Development Program of China under Grant 2020YFB1807803.
(Corresponding authors: Xiaogang Tang and Yiqing Zhou.)
This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2022.3225239, provided by the authors.
Digital Object Identifier no. 10.1109/TMC.2022.3225239

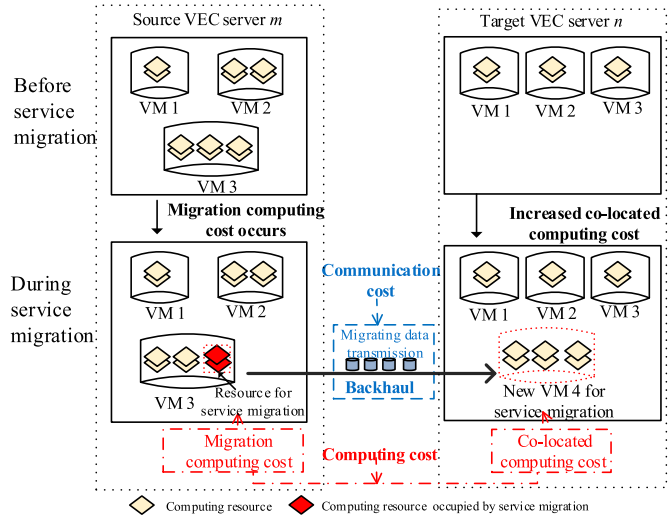


Fig. 1. The communication and computing cost during service migration.

by the increased service delay due to the increased distance between the serving VEC server and the moving vehicles, which is specifical for VEC networks.

Considering VEC networks, some researches [9], [10], [11], [16], [17], [18], [19] have been carried out on the service migration caused by the mobility of vehicles. To reduce the transmission delay between the vehicle and its serving VEC server, the "always migration" scheme is proposed in [9], where the service is always migrated to the VEC server nearest to the vehicle. However, this frequent service migration may incur prohibitive migration cost.

Various migration costs have been defined for VEC networks [11], [16], [17], [18], [19]. As shown in Fig. 1, service migration needs to transmit migrating data (e.g., service memory pages in virtual machines (VMs)) cross different VEC servers. This would incur enormous usage of backhaul bandwidth. Thus, some studies model the migration cost as the communication cost, which is the product of the amount of migrating data and the migration distance between the source VEC server (i.e., the serving VEC server before service migration) and the target VEC server (i.e., the serving VEC server after service migration) [11]. This communication cost has been employed in various ways to design migration schemes. In [11], the migration problem is formulated as an optimization problem to minimize the sum of communication cost and serving delay, while [16] minimizes the communication cost while satisfying the service delay requirements. Differently, [17] minimizes the service latency under the constraint of communication cost budget to limit excess backhaul use for service migration.

In addition to the communication cost in [16], [17], service migration also brings computing cost. When a service is migrated from the source VEC server to the target one, two types of computing cost[1], i.e., migration computing cost and co-located computing cost will occur, which bring considerable computing cost. As shown in Fig. 1, when the service migration is initiated from the source VEC server, the computing resource (i.e., CPU cycle) at the server will be occupied by the service migration process, i.e., the VM needs to allocate a part of (e.g., 30% [14], [15]) computing resources to process and monitor the service migration. Thus, available computing resource for migrated service is reduced and service performance will be degraded. This is called *migration computing cost*. When the service is migrated to the target VEC server, the server needs to generate a new VM for the service. Since current virtualization technology cannot provide complete performance isolation among co-located VMs [20], even if there are sufficient resources, as the number of co-located VMs increases, the resource contention caused by resource sharing of the physical resources on the target VEC server increases, which results in the degraded computing performance of all services on the target VEC server [21]. This is called *co-located computing cost*, which increases with the number of co-located VMs.

In order to mitigate the negative influence of migration computing cost, a delay aware lazy migration scheme is proposed in [19]. Only when the service delay requirements cannot be met, the service is migrated to the farthest VEC server satisfying the service delay requirement. Differently, [18] targets to minimize the completion time of services, which includes the computing delay affected by the migration computing cost. Although migration computing cost has been considered, few works have captured the co-located computing cost. In fact, the co-located computing cost leads to severe degradation of computing performance. It is shown in [20], [22] that the computing delay may increase by 30% because of co-located computing cost, as more VMs are located on the same VEC server. Considering the stringent delay requirement of vehicular services, delay degradation caused by VM co-located computing cost can be intolerable. Thus, it is necessary to take the VM co-located computing cost into consideration when making service migration decisions.

Besides the traditional communication cost, this paper pays much attention to the computing cost (i.e., migration computing cost and co-located computing cost), and investigates the service migration in dynamic VEC networks considering the impact of computing and communication cost jointly. Due to the dynamic nature of the vehicles, vehicles frequently move in and out of the coverage area of the VEC server, and the density of vehicles changes from time to time. Therefore, the density change of vehicles is used to represent the dynamics of VEC networks. The main contributions are summarized as follows:

- We propose a computing and communication cost-aware migration scheme (CA-migration). First, considering that the service delay is a critical QoS metric for VEC networks, a new performance criterion, i.e., the services' satisfaction degree of delay (SSDD), is defined to quantify the probability that the service delay is smaller than the service delay requirement, which is related to the computing and communication cost. Then, the service migration problem is

1. The computing cost in this paper refers to VM interference in cloud datacenters, which is a standard concept in virtualization technology [14]. The VM interference includes the migration interference and co-located interference. To avoid confusion with communication interference, this paper uses the computing cost to replace interference. Thus, the migration interference and co-located interference are called as the migration computing cost and co-located computing cost, respectively.

formulated as an optimization problem to maximize the SSDD, where the computing cost increases the service delay and affects the target function, and the communication cost affects the resource constraint.
- Since the optimization problem is NP-hard and the VEC networks are highly dynamic, a fast transfer reinforcement learning (fast-TRL) method is proposed to acquire the optimal migration policy rapidly (i.e., the fast-TRL has smaller time complexity compared to traditional RL). First, reinforcement learning (RL) algorithm is carefully designed considering the specific property of VEC networks. Next, a time-varying policy transferring method is designed to obtain the optimal migration policy of dynamic VEC networks. When the VEC network scenario changes, instead of learning a new policy from scratches, we transfer the RL policy of previous task to new task to pre-train the new policy quickly.
- We conduct extensive simulations to verify the effectiveness of the proposed schemes. It will be shown that the proposed CA-migration can increase the SSDD by up to 30% compared with existing schemes [18], [19]. In addition, confronting with highly dynamic VEC networks (e.g., the vehicle number changes from 15 to 25), the fast-TRL method needs 25% less iterations for acquiring the optimal migration policy in a new task, compared to traditional RL which trains from scratch.

The rest of the paper is organized as follows. The system model is presented in Section 2. In Section 3, the problem formulation of CA-migration is described. The proposed fast-TRL method are presented in Section 4. Then, the CA-migration is compared with benchmark migrations, and simulation results are demonstrated in Section 5. Conclusions are presented in Section 6.

2 System Model

2.1 A VEC Network

As shown in Fig. 2, a VEC network is considered with one centralized unit (CU) and N BSs along the road, each with a co-located VEC server. Adjacent BSs can communicate with each other via a backhaul link [17], [18], [19], where the backhaul link between the BS i and BS $i+1$ is called the i^{th} backhaul link, and the BS can communicate with the CU via a fronthaul link. Assume that the distance between adjacent BSs is D_{BS}. A vehicle selects the BS providing the strongest average signal strength as its serving BS, while the serving VEC server of the vehicle can be the one co-located at its serving BS or other BS via traffic routing [23]. The VEC server provides computing, communication, and memory resources in the manner of VMs [23]. Assume that one service occupies a VM [14], and there are total $K \geq 0$ services to be served. A VEC server can support multiple services simultaneously, and all services in the same VEC server share the communication, computing, and memory resources.

When a vehicle generates the service (e.g., updating the high-definition map for autonomous driving) with highly intensive computations, it sends an offloading request to the CU, which decides the serving VEC server based on the resource of VEC servers and distance between the vehicle and VEC servers. If the VEC server co-located with the serving BS has enough resource, the service is offloaded to this VEC server. Otherwise, the service can be offloaded to other VEC servers. The vehicle can initiate a service request to its VM at any time slot $t \in \{0, 1, 2, \ldots, T\}$. Note that the mobility of vehicles is a fundamental feature of VEC networks. As the vehicle moves, its serving BS changes and the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server increases, as shown in Fig. 2. The CU will make decisions on service migration to maximize SSDD of all services.

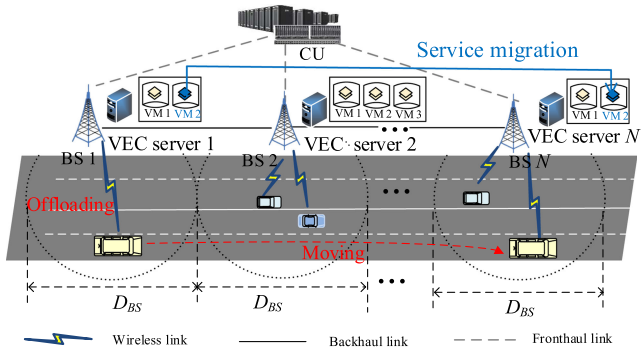


Fig. 2. A VEC network.

2.2 Service Delay

Service delay is critical to the driving safety of vehicles and QoS of passengers, and thus is chosen as a performance metric. For a specific service $k \in \{1, 2, \ldots, K\}$, based on the given serving VEC server n by the CU, the vehicle initiates a request to its VM on the serving VEC server n, and receives the response from its serving VEC server after being processed. Thus, service delay $T^R_{k,n}(t)$ is the sum of the uplink transmission delay $T^U_{k,n}(t)$, computing delay $T^C_{k,n}(t)$ at the serving VEC server, and downlink transmission delay $T^D_{k,n}(t)$, i.e.,

$$T^R_{k,n}(t) = T^U_{k,n}(t) + T^C_{k,n}(t) + T^D_{k,n}(t). \tag{1}$$

Noting that the serving VEC server n may not be co-located with the serving BS b, the transmission delay $T^U_{k,n}(t)$ is composed of the wireless uplink transmission delay from the vehicle to the serving BS b and the backhaul link transmission delay from the serving BS b to the serving VEC server n, given by

$$T^U_{k,n}(t) = \frac{S^{req}_k}{R^{ul}_{k,n}} + \sum_{i=1}^{d^{s,q}_{k,n}(t)} \frac{S^{req}_k}{B^{ul,i}_{k,n}}, \tag{2}$$

where S^{req}_k is the data size of request k, $R^{ul}_{k,n}$ is the wireless uplink data rate, $B^{ul,i}_{k,n}$ is the backhaul bandwidth of the i^{th} backhaul link for transmitting the service request k to the VEC server $n(t)$, and $d^{s,q}_{k,n}(t)$ is number of backhaul transmissions needed between the serving BS $b(t)$ and the BS co-located with the serving VEC server $n(t)$, which is given by $d^{s,q}_{k,n}(t) = |b(t) - n(t)|$. The mobility of vehicle affects the serving BS, given by

$$b(t)=\left\lceil \left(l(t-1) + \int_{t-1}^{t} v(t)dt \right) D_{BS} \right\rceil, \quad (3)$$

where $l(t-1)$ is the location of the vehicles at time slot $t-1$, $v(t)$ is the velocity of vehicles at time slot t, and $\lceil x \rceil$ represents the rounding up of x. It can be seen that as the velocity increases, in a given time, the number of BSs that the vehicle passes through increases. Therefore, given the serving VEC server $n(t)$, the vehicle moves more quickly away from it, which increases the number of backhaul transmissions $d_{k,n}^{s,q}(t)$ and the service delay in (1) faster. Thus, in a given time, the probability that the service delay is larger than the required service delay is increased, and service migration is triggered more frequently. Therefore, as the velocity increases, the service migration frequency(i.e., the number of service migration per unit time) gets larger.

It should be noted that the communication cost caused by service migration to transmit migrating data (e.g., service memory pages in VMs), reduces the available communication resource (i.e., $B_{k,n}^{ul,i}$) to transmit service data (e.g., data of high-definition map for autonomous driving). If no service migration occurs, all the backhaul resource can be used to transmit service data (e.g., service request of AR/VR video). When service migration happens, backhaul resource is needed to transmit migrating data, which reduces backhaul resource for service data and thus incurs the communication cost. The definition and impact of communication cost will be discussed in detail in Section 3.

Denoting the computation intensity of request k as γ_k, the computing delay of a request can be given by

$$T_{k,n}^{C}(t) = \frac{S_k^{req} \cdot \gamma_k}{\delta_{k,n}^{co}(t)}, \quad (4)$$

where $\delta_{k,n}^{co}(t)$ is the effective CPU cycle (i.e., available CPU cycle resource to process service request) of service k' VM running on the VEC server n at time t, considering the computing cost. If the computing cost caused by service migration is not considered [16], [17], [18], the effective CPU cycle is equal to the maximum CPU cycle $\delta_{k,n}^{\max}$. However, the non-negligible computing cost caused by service migration will increase the computing delay considerably. The impact of computing cost will be described in detail in Section 3.

Next, let S_k^{res} be the data size of the result. The downlink transmission delay is given by

$$T_{k,n}^{D}(t) = \frac{S_k^{res}}{R_{k,n}^{dl}} + \sum_{i=1}^{d_{k,n}^{s,p}(t)} \frac{S_k^{res}}{B_{k,n}^{dl,i}}, \quad (5)$$

where $d_{k,n}^{s,p}(t)$ is the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server after computing, $B_{k,n}^{dl,i}$ is the backhaul bandwidth of the i^{th} backhaul link for transmitting the service result, and $R_{k,n}^{dl}$ is the wireless downlink data rate.

2.3 Service Satisfaction Degree of Delay

The SSDD is defined as the probability that the service delay is not larger than the service delay requirement. Given the service delay $T_{k,n}^{R}(t)$, the SSDD of service k running on the VEC server n at time t $SD_{k,n}(t)$ is determined by whether the service delay requirement T_k^{re} is satisfied, given by

$$SD_{k,n}(t) = \begin{cases} 1, T_{k,n}^{R}(t) \le T_k^{re} \\ \alpha, T_{k,n}^{R}(t) > T_k^{re}, \end{cases} \quad (6)$$

where the value of α is determined by the service property, i.e., whether the delayed computing result is useful. According to this property, services can be classified into two categories [24], [25], i.e., services with hard and soft delay requirement. For services with hard delay requirement, such as the automatic/intelligent auxiliary driving services, $\alpha = 0$ since the computing result is outdated. For services with soft delay requirement, such as the augmented reality (AR)/virtual reality (VR) services for passengers, $0 < \alpha < 1$ since the quality of experience gets worse with a higher service delay (e.g., passengers get dizziness with a higher service delay).

The average SSDD of all services at time t is given by

$$SSDD(t) = \frac{1}{K} \sum_{n=1}^{N} \sum_{k=1}^{K} x_{k,n}(t) \cdot SSDD_{k,n}(t), \quad (7)$$

where $x_{k,n}(t) = 1$ indicates the service k is running on the VEC server n at time t. Otherwise, $x_{k,n}(t) = 0$. It should be noted that if $x_{k,n}(t) = x_{k,n}(t-1)$ for all $n \in \{1, 2, \ldots, N\}$, then the service k does not migrate at time t. Otherwise, $x_{k,n}(t) \neq x_{k,n}(t-1)$. In this case, if $x_{k,n}(t) = 1$ and $x_{k,n}(t-1) = 0$, the service k is migrated to VEC server n at time t. On the other hand, if $x_{k,n}(t) = 0$ and $x_{k,n}(t-1) = 1$, the service k is migrated from VEC server n to another VEC server at time t. The target of service migration in VEC networks is to maximize the SSDD.

3 Computing and Communication Cost-Aware Service Migration

Since the computing delay is seriously impacted by the computing cost, a computing cost and communication-aware service migration scheme (CA-migration) is proposed. First, the computing cost is quantified in terms of degraded computing delay based on its relationship to the available computing resource. Moreover, considering that transmitting migrating data during service migration needs to occupy backhaul resource, the communication cost is defined as occupied backhaul bandwidth during service migration. Thus, the service migration problem can be formulated as an optimization problem to maximize the SSDD, taking both computing cost and communication cost into consideration, where the computing cost affects the service delay and the communication cost affects the resource requirement.

3.1 Computing Cost Caused by Service Migration

When a service is migrated from the source VEC server to the target one, two types of computing cost, i.e., migration computing cost and co-located computing cost will occur, as shown in Fig. 1, which bring considerable computing performance degradation.

From (4), it can be seen that the computing delay is defined as the computation demand of service k divided by the effective CPU cycle $\delta_{k,n}^{co}(t)$, which is closely related to service migration. Above two types of computing cost cause that the effective CPU cycle $\delta_{k,n}^{co}(t)$ is smaller than the

maximum CPU cycle, which increases the computing delay. In this section, the computing cost (i.e., CPU performance degradation caused by service migration) and effective CPU cycle are analyzed.

3.1.1 Migration Computing Cost

When the service migration is initiated from the source VEC server, the VM needs to allocate computing resource for the migration process. Thus computing cost occurs and the service would suffer performance degradation. The duration and performance of service migration are closely related to the migration approach that mainly includes post-copy and pre-copy [14], [26]. The post-copy migration first stops the service on the source VEC server, and starts a new VM in the target VEC server for the service. The memory pages[2] are transmitted by on-demand fetching, i.e., if the new VM tries to access memory pages that have not been transmitted, this new VM is stopped and the memory pages are transmitted from the source VEC server. Thus, when there are a large number of memory pages to be transmitted, the post-copy may result in a long migration time and considerably degraded performance [26].

The pre-copy service migration can overcome the above problems, and has been widely used in most virtualization environments [14], [15]. A pre-copy migration may last for a few seconds consisted of five stages (see Appendix A for details, available online). Stage 1-3 (i.e., the Pre-migration, Iterative pre-copy, and Stop-and-copy phase) is to transmit the migrating data (e.g., memory pages), so that the target VEC server will have identical copies of the migrated service with the source VEC server. The duration T_{trans} of Stage 1-3 is given by [24], [25] $T_{trans} = \sum_{i=1}^{d_m} \frac{D_m}{B_i^m}$, where d_m is the migration distance between the source VEC server and the target one, $D_m = M(1 - (\frac{R_m}{B_i^m})^{\beta_i+1})(1 - \frac{R_m}{B_i^m})$ is the amount of migrating data, and B_i^m is the backhaul bandwidth of the i^{th} backhaul link occupied by service migration. In D_m, M is the size of memory, R_m is the memory dirty rate, and $\beta_i = \log_{\frac{R_m}{B_i^m}} \frac{T_d B_i^m}{M}$ is the number of iterative copy from the source VEC server to the target one during the iterative pre-copy stage, where T_d is the downtime in the stop-and-copy stage. Then, at Stage 4-5 (i.e., the Commitment and Activation phase), the migrated service is being resumed on the target VEC server, whose duration T_{resu} mainly depends on the type of service. Thus, the migration duration is given by [27], [28]

$$T_{mig} = T_{trans} + T_{resu}. \tag{8}$$

During migrations, the migrated service at the source VEC server can undergo severe computing performance degradation because of migration computing cost, as shown in Fig. 1. In detail, at Stage 1-2, the VM at the source VEC server needs to allocate a part of (e.g., 30% [14], [15]) computing resources to process and monitor the service migration. Then, at Stage 3-5, service is terminated in the source VEC server and being resumed in the target VEC server, which results in all the resources of VM unavailable for the migrated service [14].

Thus, the migration computing cost, i.e., the CPU performance degradation of service k' VM running on the VEC server n at time t, is given by [14]

$$\alpha_{k,n}^{mig\_deg}(t) = \begin{cases} 0, & migration\ is\ not\ performing, \\ \kappa_{k,n}(t), & migration\ is\ performing, \end{cases} \tag{9}$$

where $\kappa_{k,n}(t) \geq 0$ is the ratio of computing resources occupied by the service migration of VM. As a result, in order to avoid potential violations of the service delay requirement, it is essential to consider the migration computing cost, which affects the effective CPU cycle $\delta_{k,n}^{co}(t)$ of service k' VM hosting on the VEC server n at time t, given by [14]

$$\delta_{k,n}^{co}(t) = \delta_{k,n}^{\max}\left(1 - \alpha_{k,n}^{mig\_deg}(t)\right). \tag{10}$$

3.1.2 Co-Located Computing Cost

Before the service is migrated from the source VEC server to the target VEC server, the target server needs to generate a new VM (i.e., target VM) for the service, as shown in Fig. 1. So the number of co-located VMs in the target VEC server is increased, and the resource contention is aggravated. Because current virtual technology cannot provide complete performance isolation between co-located VMs [22], and it is technically difficult to allocate some resources, such as CPU cache and memory bandwidth, to different VMs. As the number of co-located VMs increases, the resource contention caused by resource sharing of the physical resources on the target VEC server increases, which results in higher cache miss ratios and more memory bandwidth conflict. Thus, the read and write access slow down, and CPU performance degrades [21], [22], which is known as the co-located computing cost.

Considering the mutual effect of CPU performance of co-located services, a demand-supply model has been proposed in [21], [22] to capture the co-located computing cost running on the VEC server n at time t, given by

$$\alpha_n^{co\_deg}(t) = \left(\varepsilon_0 + \varepsilon_1 \cdot \delta_n^{dem}(t)\delta_n^{sup}(t)\right), \tag{11}$$

where ε_0 and ε_1 are the factors of performance degradation, $\delta_n^{dem}(t)$ and $\delta_n^{sup}(t)$ are the CPU resource demand of VMs locating at VEC server n, and CPU capacity of VEC server n, respectively. It can be seen that the CPU performance degradation $\alpha_n^{co\_deg}$ caused by co-located computing cost exists, i.e., $\alpha_n^{co\_deg}$ is larger than zero, as soon as the CPU resource demand is not zero, even if the CPU resource demand $\delta_n^{dem}(t)$ is smaller than CPU capacity $\delta_n^{sup}(t)$. Moreover, $\alpha_n^{co\_deg}$ increases with $\delta_n^{dem}(t)$ and thus increases with the number of VMs hosting on the VEC server n. As a result, in order to avoid potential violations of the service requirement, it is essential to consider the co-located computing cost.

2. A memory page is a fixed-length contiguous data block of VM. It is the smallest unit of data blocks for memory management, described by a single entry in the page table. During service migration, multiple memory pages of VM should be transmitted from the source VEC server to the target one.

Thus, after considering the computing cost caused by service migration (i.e., migration computing cost and co-located computing cost), the effective CPU cycle $\delta^{co}_{k,n}(t)$ of service k' VM hosting on the VEC server n at time t is given by [14], [21], [22]

$$\delta^{co}_{k,n}(t)=\delta^{\max}_{k,n}\Big(1-\alpha^{mig\_deg}_{k,n}(t)\Big)\big(1-\alpha^{co\_deg}_{n}(t)\big), \tag{12}$$

where $\delta^{\max}_{k,n}$ is the maximum computing capacity of service k' VM running on the VEC server n.

Obviously, the computing cost caused by service migration (i.e., migration computing cost and co-located computing cost) will increase the computing delay in (4), and thus has a negative effect to SSDD in (7).

3.2 Communication Cost Caused by Service Migration

During service migrations, the backhaul link is needed to transmit migrating data from the source VEC server to the target one, as shown in Fig. 1. Thus, service migrations would incur communication cost, which is defined as the backhaul bandwidth occupied by service migration.

Let $\overrightarrow{H}=\begin{bmatrix} H_{11}, & \ldots, & H_{1N} \\ \ldots, & \ldots, & \ldots \\ H_{K1}, & \ldots, & H_{KN} \end{bmatrix}^T$ be the backhaul occupation matrix for service migration, where $[\cdot]^T$ stands for transpose operation, $H_{k,l}=1$ represents that the service k occupies the l^{th} backhaul link, otherwise, $H_{k,l}=0$. $\vec{B}^m=(B^m_1,B^m_2,\ldots,B^m_K)^T$ is the bandwidth requirement vector for migrating data, where B^m_k represents the backhaul bandwidth required by migrated service k. Considering all the K services, the total service migration cost at time slot t is given by

$$Comm\_cost=\sum_{k=1}^{K}\sum_{n=1}^{N}\overrightarrow{H}(k,n)\cdot\vec{B}^m(k). \tag{13}$$

If no service migration occurs, all of the backhaul resource can be used to transmit service data (e.g., data of high-definition map for autonomous driving). However, during the service migration, communication cost is inevitable, which increases the contention of backhaul resource and may result in a fail of service data transmission. Thus SSDD may be decreased.

3.3 Computing and Communication Cost-Aware Service Migration Problem

The computing and communication cost-aware service migration is formulated as an optimization problem to maximize the SSDD in long-term, taking the impact of computing cost and communication cost into account jointly, where the computing cost affects the service delay and the communication cost affects the resource requirement.

The average SSDD of all services at time t is $SSDD\ (t)=\frac{1}{K}\sum_{n=1}^{N}\sum_{k=1}^{K}x_{k,n}(t)\cdot SSDD_{k,n}(t)$ defined in (7), where $x_{k,n}(t)=1$ indicates the service k is running on the VEC server n at time t. Otherwise, $x_{k,n}(t)=0$. The service migration decision decides the value of $x_{k,n}(t)$(i.e., 0 or 1) for all $n\in\{1,2,\ldots,N\}$ and $k\in\{1,2,\ldots,K\}$ at time t aiming to maximize the average SSDD of all services, under the constraints of limited resources such as backhaul and CPU, given by

$$\begin{aligned}
&\max_{x_{k,n}\in\{0,1\}}\lim_{T\to\infty}\sum_{t=1}^{T}SSDD(t)\\
&=\max_{x_{k,n}\in\{0,1\}}\frac{1}{KT}\lim_{T\to\infty}\sum_{t=1}^{T}\sum_{n=1}^{N}\sum_{k=1}^{K}x_{k,n}(t)\cdot SSDD_{k,n}(t)\\
&s.t.\overrightarrow{H}\cdot\vec{B}^m+\overrightarrow{G}\cdot\vec{B}^s\le\vec{B}_{total}\\
&\overrightarrow{x}\cdot(\vec{c}^r,\vec{\eta}^r,\vec{R}^r)\le\Big(\vec{c}_{total},\vec{\eta}_{total},\vec{R}_{total}\Big),
\end{aligned} \tag{14}$$

where T is the considered cumulative time slots in the long-term. The first constraint means that total backhaul resource demand of migrating data (i.e., $\overrightarrow{H}\cdot\vec{B}^m$, the communication cost of service migration) and service data ($\overrightarrow{G}\cdot\vec{B}^s$) cannot exceed the backhaul resource capacity. $\overrightarrow{G}=[G_{11},\ \ldots,\ G_{1N}\ldots,\ldots,\ldots G_{K1},\ldots,G_{KN}]^T$ is the backhaul occupation matrix for service data, where $G_{k,l}=1$ represents that the service k occupies the l^{th} backhaul link, otherwise, $G_{k,l}=0$. $\vec{B}^s=(B^s_1,B^s_2,\ldots,B^s_K)^T$ is the bandwidth requirement vector for service data, where B^s_k is the backhaul bandwidth required by service k. $\vec{B}_{total}=(B_1,B_2,\ldots,B_{N-1})^T$ is the backhaul capacity vector, where B_l stands for the total backhaul bandwidth of l^{th} backhaul link. The second constraint means that the total resource demand of computing resource, memory resource, and wireless communication of co-located services cannot exceed its resource capacity. $\vec{c}^r=(c^r_1,c^r_2,\ldots,c^r_K)^T$, $\vec{\eta}^r=(\eta^r_1,\eta^r_2,\ldots,\eta^r_K)^T$ and $\vec{R}^r=(R^r_1,R^r_2,\ldots,R^r_K)^T$ are the computing resource requirement vector, memory resource requirement vector, and wireless resource requirement vector, respectively, where R^r_k is the wireless bandwidth allocated to the vehicle by its accessed BS for service k. $\vec{c}_{total}=(c_1,c_2,\ldots,c_N)^T$, $\vec{\eta}_{total}=(\eta_1,\eta_2,\ldots,\eta_N)^T$ and $\vec{R}_{total}=(R_1,R_2,\ldots,R_N)^T$ are the resource capacity vector for three kinds of resources of each VEC server, respectively, where R_n is the wireless bandwidth capacity of the BS co-located with the VEC server n. It should be noted that the computing cost (i.e., migration computing cost and co-located computing cost) caused by service migration decreases the effective CPU cycle of service' VM in (12), and thus increases the service delay in (1) significantly, which decreases the SSDD, and thus affects the target function of (14). Moreover, the communication cost is quantified in terms of occupied backhaul bandwidth needed to transmit migrating data, which increases the contention of backhaul resource and affects the first constraint of (14).

In summary, jointly considering the communication and computing cost, the computing and communication cost-aware service migration for VEC networks is formulated as a constrained non-linear integer programming problem, which is NP-hard (see Appendix B, available in the online supplemental material). This problem cannot be readily solved by conventional optimization algorithms [29]. Reinforcement learning (RL) is promising to solve NP-hard problems [16], [29]. However, VEC networks are highly dynamic, i.e., the vehicle density changes from time to time, which changes the target function of optimization problem (14) and the optimal service migration policy [30].

Correspondingly, a new RL should be trained each time the vehicle density changes to improve the SSDD [31]. However, the time cost to acquire experience data and train the RL can be prohibitive (e.g., some hours) [32]. Moreover, the performance of RL during training can be extremely poor [33]. Hence, the well-trained RL may not work effectively in dynamic VEC networks [30]. Transfer learning (TL) [34], [35] can intelligently apply knowledge learned in a previous task to solve the problem in a new task. Thus, a fast transfer reinforcement learning method (fast-TRL) combining TL and RL can be designed to solve the optimization problem (i.e., the computing and communication cost-aware migration problem) in a dynamic vehicular network.

4 Fast Transfer Reinforcement Learning Method

In this section, based on Deep Q-learning (DQL, a commonly used RL algorithm) and TL, a fast-TRL method is proposed which adopts time-varying policy transfering method to acquire the optimal migration policy fast. The fast-TRL method includes two procedures. First, considering the specific property of VEC networks, the actions, states and reward function of DQL are carefully designed. Second, considering the dynamics of VEC networks and the slow convergence speeds of RL, a time-varying policy transfering method is designed to speed up the convergence. It can exploit the knowledge learned in the previous task, where the DQL of previous task has a random decreasing impact on the current task, while the effect of DQL of current task randomly increases gradually. So the acquisition of optimal migration policy in a new task can be sped up by fast-TRL.

4.1 Deep Q-Learning of Fast-TRL Method

In DQL, the environment is represented as (S, A, R, γ),where S, A and R are states, actions, and immediate rewards, respectively [36]. $\gamma \in [0,1]$ is the discount factor used to balance immediate and accumulative reward. The target of DQL is to maximize the cumulative reward, which is estimated by a deep neural network, and given by $Q(s,a;\theta) = E[\sum_{i=0}^{\infty} \gamma^i r(i+t)|s,a;\theta]$, where $s \in S$, $a \in A$, $r \in R$, and the weight parameter θ stands for the weights of the deep neural network of DQL. In the training process of DQL, the ε-greedy policy is utilized to balance the exploration with the probability of ε (i.e., randomly select an action) and exploitation with the probability of $1-\varepsilon$ (i.e., $arg\max_a Q(s,a;\theta)$). At each state s, once executing the action a, the instant reward r, and the next state s' are obtained. The experience $\{s,a,s',r\}$ is stored into the experience replay buffer. With samples from the replay memory periodically, the DQL can be trained (i.e., updating weight parameter θ) using the random gradient descent algorithm by minimizing the loss function. Thus, it can accurately approximate the Q-values for each action based on states, and those Q values are used to derive the optimal policy, i.e., choose the action with the highest Q-value. The details of DQL are given in [36].

Considering the specific property (e.g., computing and communication cost-aware and stringent delay requirement) of VEC networks in the concerned optimization problem (14), to obtain the optimal policy of service migration, the actions, states and reward function should be carefully designed.

States: The state observed for service k at time t is defined as $s_k(t) = \{b_k(t), m_k(t), n_{mec}(t), n_{bh}(t)\}$, where $b_k(t)$ is the index of serving BS of service k, and $m_k(t)$ is the index of serving VEC server. Moreover, $n_{mec}(t)$ represents the number of services processed by each VEC server and $n_{bh}(t)$ denotes the number of services transmitting service data in each backhaul link, both of which stand for the resource demand (i.e., the amount of computing, memory, and communication resource needed by the services). It should be noted that the first 2 variables are indices, and the last 2 variables are N-dimensional vectors. So the state is arranged to a state matrix with 2 rows and $N+1$ columns.

Action: The DQL agent has to decide where to migrate the service k. Therefore, the action at time t is represented as $a_k(t) \in \{1,2,\ldots,N\}$. If the service k is migrated to VEC server n at time t (i.e., $a_k(t) = n$), after the service migration being completed at time $t + T_{mig}$, the service k begins running on the VEC service n, and $x_{k,n}(t + T_{mig})$ changes to 1 in the optimization problem (14), where T_{mig} is the migration duration.

Reward Function: The design of reward function should consider the migration cost (including the computing cost and communication cost), the service QoS (whether the service requirement can be met), and resource load. Thus, the reward function at time t (i.e., immediate reward) is defined as

$$r_{k,n}(t) = C_{k,n}^{mig}(t) + C_{k,n}^{ser}(t) + C_{k,n}^{res}(t), \tag{15}$$

where $C_{k,n}^{mig}(t) = \begin{cases} C_{mig}^{-}, & migrate\ at\ time\ t \\ 0, & otherwise \end{cases}$ with $C_{mig}^{-} < 0$ as a penalty accounting for the cost when migration occurs at time t. $C_{k,n}^{ser}(t) = \begin{cases} C_{ser}^{+}, T_{k,n}^{R}(t) \le T_k^{re} \\ 0, T_{k,n}^{R}(t) > T_k^{re} \end{cases}$ is the service reward, where $C_{ser}^{+} > 0$ is a reward obtained when the service delay at time t satisfies the required delay. To properly trigger the service migration when the service delay $T_{k,n}^{R}(t)$ is larger than the required delay T_k^{re}), the service reward $C_{ser}^{+} > 0$ obtained from the service migration should be set to a value large enough to compensate for the migration penalty $C_{mig}^{-} < 0$, i.e., $|C_{ser}^{+}| \ge |C_{mig}^{-}|$. Moreover, $C_{k,n}^{res}(t) = \begin{cases} C_{res}^{-}, & overloadedattimet \\ 0, & otherwise \end{cases}$ is the resource penalty, where $C_{res}^{-} < 0$ is a penalty if the VEC server n is overloaded at time t, i.e., any resource (computing, memory or communication resource) needed by services on VEC server n is larger than its resource capacity of VEC server n. Generally speaking, to make the training of DQL faster and more stable, the values of above parameters (i.e., C_{ser}^{+}, C_{mig}^{-} and C_{res}^{-}) can be set to a constant within $[-1,1]$[37], [38]. It should be noted that once the service migration occurs, the overall reward (15) decreases because the migration penalty decreases from zero to $C_{mig}^{-} < 0$ due to the computing and communication cost. Moreover, the computing cost (i.e., migration computing cost and co-located computing cost) increases the service delay $T_{k,n}^{R}(t)$, and decreases the SSDD.

Thus, the service reward $C_{k,n}^{ser}(t)$ has a smaller probability to obtain $C_{ser}^{+} > 0$. Overall, the reward (15) decreases as migration occurs.

Moreover, note that DQL has low sampling efficiency. This is because experiences, i.e.,$\{s, a, s', r\}$, were uniformly sampled from the experience replay buffer at the same frequency that they were originally experienced, regardless of their significance [39]. However, in the considered service migration problem, most of the time, the service does not need to migrate. Migration occurs only in a few occasions, but it has a significant impact on the SSDD, and should be learned emphatically. To deal with this situation, prioritized experience replay [39] is used to sample important experiences (i.e., service migration) more frequently, thus DQL can learn more efficiently.

4.2 Time-Varying Policy Transfering Method of Fast-TRL Method

DQL enables the agent to learn the optimal policy at each state. However, the VEC networks are inherently dynamic. The well-trained agents may suffer from severe performance deterioration as the VEC network scenario changes. Thus, as the scenario changes, a new DQL should be trained. However, the time cost to acquire experience data and train the DQL can be prohibitive. Moreover, the performance of DQL during training can be extremely poor. Fortunately, in the VEC networks, the DQL environment represented by (S, A, R, γ) in different scenarios exhibits high relevancies. For example, when the number of vehicles changes, the environment states S are different, while A, R and γ keep the same. The high relevancies of DQL environment make the migration policy at different scenarios relevant [30], [40]. Therefore, the dynamic issue of VEC networks can be solved by transfer learning (TL) [34]. TL can intelligently apply knowledge learned from a previous task in a scenario (the source task) to solve tasks in new but related scenarios (the target task) effectively. Thus, combining TL with RL, a fast-TRL method based migration policy is proposed, which adopts a time-varying policy transfering method to acquire the knowledge of learned strategies in source tasks to pre-train and speed up the ongoing learning process of new DQL agent.

In the source task, this paper assumes that the optimal policy π_s^* has been learned by using DQL. The proposed fast-TRL method is used to update the migration policy of target task. Considering the relevancy of migration policy, the policy π_s^* of source tasks can be transferred to the target task, so that the target task can exploit the transferred policy π_s^* to make a better migration decision, compared to training from the scratch. However, in spite of the similarities between the source task and the target task, differences do exist. For example, the distribution of states is different. In this case, if the transferred policy π_s^* of source tasks is used directly, suboptimal solutions may be obtained. Moreover, as training goes on, the target task obtains more and more its own experience data. The DQL of target task can be trained and the policy π_t of target task is improved gradually. Thus, as training goes on, the transferred policy π_s^* should have a decreasing impact on the final policy of the target task, while the effect of DQL of target task π_t increases [30].

Taking the above considerations into account, a time-varying policy transfering is proposed for fast-TRL, which selects an action depends on two sub-policies. One is $\pi_s^*(s, a)$, which is transferred from the source task. The other is $\pi_t(s, a)$, which is obtained by the DQL of target task. With the proposed fast-TRL method, the overall updating policy $\pi(s, a)$ of the target task is given by [30]

$$\begin{aligned}\pi(s,a) = &(1 - \mathrm{func}(\xi - U[0,1]))\pi_t(s,a) \\ &+ \mathrm{func}(\xi - U[0,1])\pi_s^*(s,a),\end{aligned} \tag{16}$$

where $\mathrm{func}(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$, $\xi \in [0,1]$ is the transfer rate which decreases as training goes on, and $U[0,1]$ is a random real number which is taken from [0,1] with a uniform distribution probability, and thus $U[0,1]$ is greater than or equal to 0 and less than or equal to 1. Initially, the transfer rate ξ is high (e.g., 1), and thus the transferred policy of source task $\pi_s^*(s, a)$ dominates in the overall policy. Hence, when the environment enters a new state s, the system makes migration decisions based on $\pi_s^*(s, a)$, which might be better compared with making random decisions. Consequently, the proposed policy update leads to a possible performance jumpstart at the beginning. At the same time, the DQL in target task acquires experience data to train and update itself using the random gradient descent algorithm. Thus, the policy $\pi_t(s, a)$ of target task is improved gradually. Considering the possible negative effect of $\pi_s^*(s, a)$ and improved $\pi_t(s, a)$ as training goes on, ξ is decreased to reduce the effect of the transferred policy $\pi_s^*(s, a)$ and increase the effect of the improved policy $\pi_t(s, a)$ of target task. Therefore, the agent of target task can not only take advantage of the learned expertise in the source task, but also swiftly get rid of the negative guidelines as it gets more experience data.

5 Performance Evalution

Simulations are carried out to verify the performance of the CA-migration scheme and fast-TRL method. Assume that the considered BS number is 16, and the distance between adjacent BSs is 300 m. Initially, all the vehicles are randomly distributed in the coverage area of all BSs. Each vehicle initiates a service with a delay requirement of 0.02s. These services are randomly offloaded to any VEC server that can satisfy the resources constraint and service delay requirement. In order to simulate the dynamic of VEC networks, the number of vehicles in the coverage of considered BSs is changed from 15 (i.e., the number of services is 15) to 25 in the simulations, and the learning capability of the proposed fast-TRL method in a dynamic VEC network is evaluated. The migration policy in DQL is represented by a 7-layer convolutional neural network (CNN) consisting of two convolution layers with ReLu activation, each followed by a max pooling, two fully-connected layer, and a final softmax output layer [41]. Other typical system parameters [19], [20], [24], [42] and DQL parameters are listed in Table 1.

The proposed CA-migration scheme is compared with three schemes. The first one is the delay aware lazy migration (i.e., lazy migration, LM) [19], which outperforms the

TABLE 1
System Configurations

Parameter	Value
Number of BSs N	16
Performance degradation ε_0 and ε_1	0.01 and 0.02
Distance between adjacent BSs D_{BS}	300 m
Computing intensity γ	1000 CPU cycles/bit
Wireless data rate of each VM	150 Mbps
Backhaul data rate of each VM	400 Mbps
Maximum CPU cycle of each VM	$0.5*10^6$ CPU cycles
Service delay requirement T_{re}	0.02s
Service reward	0.1
Migration reward	0.2
Resource reward	-1
Replay size	1000
Minibatch size	128
Exploration probability	0.9-0.99

never migration and always migration schemes. The second baseline is the migration scheme minimizing service delay (i.e., minimizing service delay), which targets to minimizing the service delay including the computing delay affected by the migration computing cost [18]. The third baseline is the VM migration computing and communication cost-aware service migration (i.e., MCA-migration) ignoring the impact of co-located computing cost, whose state, action, and reward are the same as those of the proposed CA-Migration. The main difference between this work and above works ([18], [19] and MCA-migration) is that this work investigates the impact of the co-located computing cost which has been ignored in existing works.

First of all, to investigate the learning capability of the proposed fast-TRL method in a dynamic VEC network, the average SSDD is shown in Fig. 3 as a function of iteration number, when fast-TRL method, well-trained DQL of source task (i.e., source DQL), and retrained DQL of target task (i.e., target DQL) are employed. In the source task, the number of vehicles in the coverage of considered BSs is 15, i.e., the number of services is 15. Then, in the target task, this number changes to 25, which changes the resource demand of vehicles, and thus the distribution of states. It can be seen that using traditional training strategy, the SSDD of target DQL is low at the beginning of training due to the lack of experience data, and the target DQL requires a long training time to reach reasonable performance. Differently, fast-TRL adopts the time-varying policy transferring method to exploit the transferred policy from the source task, so it can learn a relatively good migration policy at the very beginning. So the SSDD of fast-TRL is much higher than that of the target DQL when the iteration number is small. Then, as training goes on with more iterations, the migration policy continuously improves and quickly converges. Thus, the fast-TRL can provide SSDD performance gain at the beginning, and speed up the convergence of policy. As a comparison, the performance of source DQL is also shown, where the target task directly employs the policy of source task. Not surprisingly, the source DQL can only provide a sub-optimal performance for the target task since the state distributions are different. The simulation results verify that the proposed fast-TRL method presents good learning capability in dynamic VEC networks.

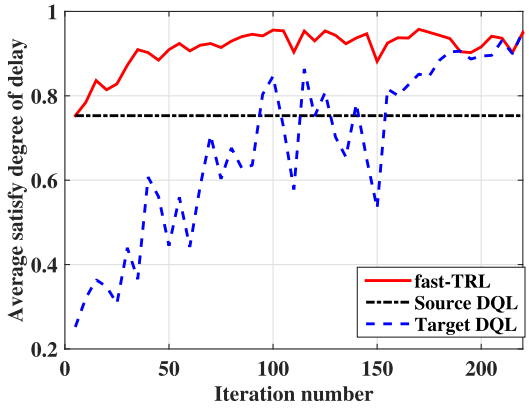


Fig. 3. The average SSDD as a function of iteration number.

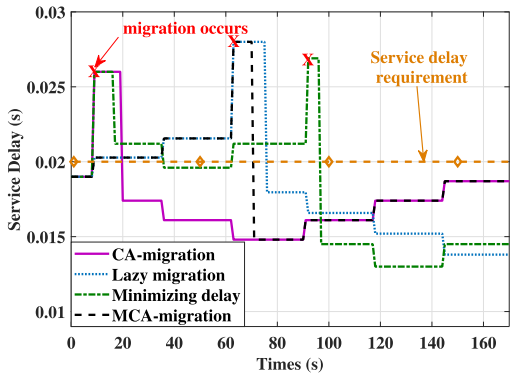


Fig. 4. The service delay as a function of times.

Using fast-TRL method, the service delay performance of the proposed CA-migration scheme is shown in Fig. 4, compared with various migration schemes. The total number of services is 25. Without loss of generality, service 1 is taken as an example and its service delay is shown in Fig. 4 as a function of time. At 0s, the vehicle generating service 1 accesses BS 3, and service 1 is offloaded to VEC server 1, which provides computing resource for 3 services including service 1. As the vehicle generating service 1 moves, it handovers into BS 4 at 9s, and the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server increases. Increased number of backhaul transmissions causes the service delay (i.e., 0.021s) to be larger than the delay requirement (i.e., 0.02s). So service migration should be considered, during which the VM computing cost caused by other co-located services affects the service delay and should be taken into account.

However, using the LM [19], the system ignores the VM co-located computing cost and obtains a service delay of 0.017s, which is smaller than 0.02s, although the actual service delay is 0.021s. So the system still runs service 1 in VEC server 1, at 9s. A similar thing happens at 35s. Until at 62s when the vehicle handovers into BS 6, the service delay without considering VM co-located computing cost is greater than 0.02s. Then the service 1 is migrated from the source VEC server 1 to the target VEC server 10, which occurs from 63s to 76s, according to (8), i.e., the formula of migration duration. During the service migration, the service delay increases to 0.027s because of the computing cost caused by migration computing cost, i.e., a part of (e.g., 30% [14]) computing resource at the source VEC server 1 is

occupied by the service migration process. It should be noted that in order to minimize the migration number, service 1 is migrated to VEC server 10, which is the farthest VEC server on the premise that can meet the delay requirement. Hence, after the service migration at 76s, the migrated service begins to run on the target VEC server 10, the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server changes from 5 to 4, and the service delay decreases. As the vehicle moves on, handovers occur at 90s, 117s, and 144s, which reduces the number of backhaul transmissions needed to 3, 2, and 1, respectively. Thus, the service delay decreases accordingly.

Compared with the lazy migration, the proposed CA-migration scheme can improve the service delay. Since VM computing cost is considered, as the vehicle handovers to BS 4 at 9 s, the system can obtain the actual service delay of 0.021s, which is larger than 0.02s. So the system migrates the service from VEC server 1 to 6, which occurs from 10s to 18s to let the service delay satisfy the service requirement. Compared with the LM that does not migrate service at 9s, the service delay increases to 0.026s during migration. So during the migration, the service delay of the proposed CA-migration is larger than that of the lazy migration. This is because the migration computing cost reduces the available computing resource of service. So the computing delay and the service delay are increased. After the service migration is completed at 19s, the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server is reduced from 3 to 2, thus the communication delay is reduced and the service delay decreases to 0.018s, which is smaller than the delay requirement (i.e., 0.02s). While the lazy migration scheme still keeps the service delay larger than the delay requirement (i.e.,0.02s). Thus, the SSDD of CA-migration improves. Moreover, this migration duration is 9s, which is smaller than the duration (i.e., 13s) of LM. That is because the migration distances of CA-migration scheme and LM scheme are 5 and 9, respectively, and larger migration distance causes larger migration duration, as shown in (8).

Similar to the LM, the minimizing delay migration [18] and MCA-migration also ignore the VM co-located computing cost. Thus, both schemes may not migrate service when the actual delay is larger than 0.02s. For example, at 9s, the MCA-migration still runs service 1 in VEC server 1. Moreover, both schemes may migrate the service to VEC servers with high CPU load since co-located computing cost is not considered. For example, at 9s, the minimizing delay migration chooses to migrate service 1 to the VEC server 5 running 5 services, where co-located computing cost causes large service delay (i.e., 0.021s). Thus, the larger co-located computing cost causes the service delay of minimizing delay migration after the service migration to be larger than that of CA-migration. Note that the service delay of both schemes before the service migration and during the service migration is the same, because they run the service on the same VEC server 1.

In summary, the SSDD of the proposed CA-migration, lazy migration, minimizing delay migration, and MCA-migration are 91%, 62%, 64%, and 67%, respectively. By taking both VM migration computing cost and co-located computing cost into consideration, the proposed CA-migration can accurately obtain the actual service delay, and migrate service when the actual delay is larger than the delay requirement. Moreover, since co-located computing cost is concerned, the CA-migration will not migrate the service to VEC servers with high CPU load.

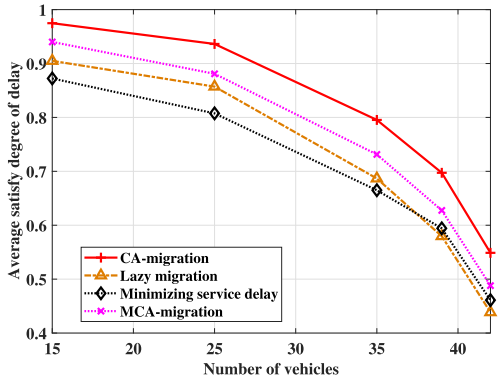


Fig. 5. The average SSDD as a function of the number of vehicles.

Considering a dynamic VEC network, the average SSDD is shown in Fig. 5 as a function of the number of vehicles. It can be seen that the average SSDD decreases rapidly as the number of vehicles increases. This is because as the number increases, resource competition of computing resource and communication resource gets more and more serious. Compared with the other schemes, the proposed CA-migration can increase the average SSDD by up to 20%. This is because the proposed CA-migration considers the increased computing delay caused by computing cost, which can trigger migration in time when the service delay requirement can not be satisfied. It also avoids migrating services to VEC servers having high CPU load to mitigate VM co-loacted computing cost.

Fig. 6 shows the average SSDD as a function of the backhaul bandwidth(the communication resource). It can be seen that as the backhaul bandwidth increases, the average SSDD of all considered methods increases. This is because the resource contention over backhaul links decreases as the backhaul bandwidth increases. Service can be migrated to a farther VEC server that satisfies the resource constraints and service's delay requirement, which can effectively reduce the migration times and migration cost, and thus improve the SSDD. Among all the schemes, the proposed CA-migration obtains the highest average SSDD. This is because it shortens the number of backhaul transmissions needed between the serving BS and the BS co-located with the serving VEC server when the backhaul resource is limited, which relieves the contention of backhaul resource. And it increases the number of backhaul transmissions needed when the backhaul resource is sufficient, which decreases the migration frequency and migration cost, and thus improves the SSDD.

The average SSDD is shown in Fig. 7 as a function of CPU capacity (the computing resource). It can be seen that, as the CPU capacity increases, the average SSDD of all considered methods increases. This is because the resource contention over CPU and co-located computing cost decreases as the CPU capacity increases, and thus the SSDD increases. Among all the schemes, the proposed CA-migration

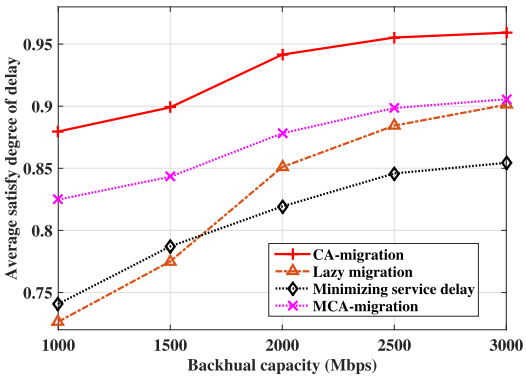


Fig. 6. The average SSDD as a function of backhaul capacity.

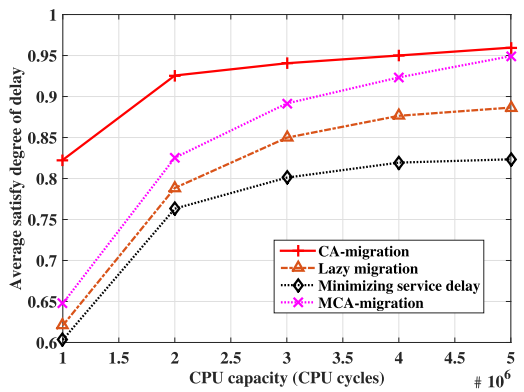


Fig. 7. The average SSDD as a function of CPU capacity.

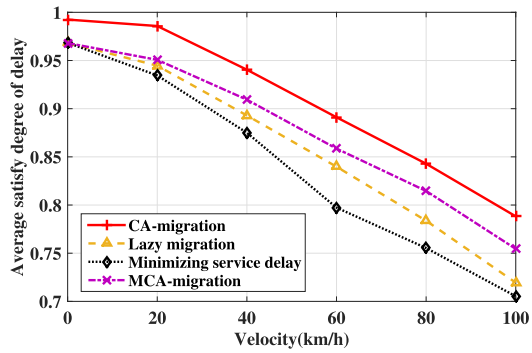


Fig. 8. The average SSDD as a function of velocity.

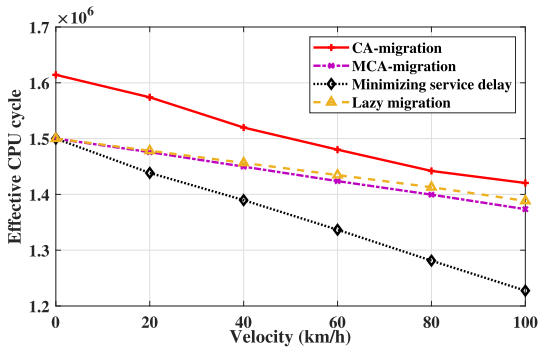


Fig. 9. The average effective CPU cycle as a function of velocity.

achieves the best performance, because of considering the co-located computing cost. Moreover, the performance gap between the proposed CA-migration and the MCA-migration decreases. This is because as the CPU capacity increases, the co-located computing cost decreases, which can be obtained from (11), and its impact on the service delay and SSDD gradually disappears. This observation suggests that when CPU capacity is sufficiently high for the services (i.e., $\delta_n^{dem}(t) \ll \delta_n^{sup}(t)$ in (11)), co-located computing cost may be small and can be ignored. However, if the CPU capacity is constrained, co-located computing cost must be considered.

In order to evaluate the impact of mobility, the average SSDD is shown in Fig. 8 as a function of velocity of vehicles. It can be seen that as the velocity increases, the average SSDD of all considered methods decreases. The reasons are as follows. As discussed below (3), as the velocity increases, the migration frequency f (i.e., the number of service migration per unit time) increases. Considering a time period of T, the number of service migration can be $T \cdot f$, and the total migration duration becomes $T \cdot f \cdot T_{mig}$, where T_{mig} is the service migration duration given by (8). Note that the service migration is triggered when the service delay is larger than the required delay. As shown in Fig. 5, during migration, the service delay keeps being larger than the required delay, and it will be improved only when the migration is completed. Thus, the SSDD can be given by $\frac{T-T\cdot f\cdot T_{mig}}{T}$. Hence, as the velocity increases, f gets larger and the SSDD decreases. Among all the schemes, the proposed CA-migration achieves the best performance, since it has taken the co-located computing cost into account.

Finally, consider that the proposed CA-migration focuses on the impact of the computing cost (i.e., migration computing cost and co-located computing cost), which determines the effective CPU cycle of each VM, given by in (12). Fig. 9 evaluates the effective CPU cycle of each VM as a function of vehicle velocity, where the maximum computing capacity of each VM is $2 \cdot 10^6$ CPU cycle. It can be seen that, as the velocity increases, the effective CPU cycle decreases. This is because as the velocity increases, the migration frequency, the number of service migration per unit time, and the total migration computing cost increases. Moreover, among all the schemes, the proposed CA-migration achieves the largest effective CPU cycle of VM, since it will not migrate the service to VEC servers with high CPU load, which can decrease the co-located computing cost, and thus improve the effective CPU cycle.

6 Conclusion

Service migration is inevitable in VEC networks. Yet frequent service migrations incur prohibitive migration cost including the communication cost (i.e., occupied backhaul bandwidth) and computing cost (i.e., increased computing delay). Focusing on the impact of the VM computing cost, this paper considered the computing and communication cost jointly, and proposed a computing and communication cost-aware service migration scheme for VEC networks (i.e., CA-migration). First, a migration optimization problem was formulated aiming to maximize the SSDD. Then, considering that the formulated problem is a constrained non-linear integer programming problem, and the VEC networks are highly dynamic, a fast-TRL method was proposed to provide an adaptive migrate service scheme in dynamic VEC networks. Simulations showed that the proposed CA-migration

scheme can increase the SSDD by up to 30% and needs 25% less training time for acquiring the optimal service migration policy compared with existing schemes. The RL is a promising technique to improve the SSDD in VEC networks, and also to enable other intelligent networks, such as Industrial Internet. Future work will consider more advanced RL methods such as actor-critic deep deterministic policy gradient to further improve the network performance.

REFERENCES

[1] Y. Zhou et al., "Service aware 6G: An intelligent and open network based on convergence of communication, computing and caching," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 253–260, Aug. 2020.

[2] L. Liu, Y. Zhou, J. Yuan, W. Zhuang, and Y. Wang, "Economically optimal MS association for multimedia content delivery in cache-enabled heterogeneous cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1584–1593, Jul. 2019.

[3] L. Liu, Y. Zhou, V. Garcia, L. Tian, and J. Shi, "Load aware joint CoMP clustering and inter-cell resource scheduling in heterogeneous ultra dense cellular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2741–2755, Mar. 2018.

[4] L. Liu, Y. Zhou, W. Zhuang, J. Yuan, and L. Tian, "Tractable coverage analysis for hexagonal macrocell-based heterogeneous UDNs with adaptive interference aware CoMP," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 503–517, Jan. 2019.

[5] V. Garcia, Y. Zhou, and J. Shi, "Coordinated multipoint transmission in dense cellular networks with user-centric adaptive clustering," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4297–4308, Aug. 2014.

[6] Y. Qi, L. Tian, Y. Zhou, and J. Yuan, "Mobile edge computing-assisted admission control in vehicular networks," *IEEE Veh. Mag.*, vol. 14, no. 1, pp. 37–44, Mar. 2019.

[7] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2X access technologies: Regulation, research, and remaining challenges," *IEEE Commun. Surv. Tuts.*, vol. 20, no. 3, pp. 1858–1877, Third Quarter 2018.

[8] Y. Qi, Y. Zhou, L. Liu, L. Tian, and J. Shi, "MEC coordinated future 5G mobile wireless networks," *J. Comput. Res. Develop.*, vol. 55, pp. 478–486, 2018.

[9] W. Bao et al., "Follow me fog: Toward seamless handover timing schemes in a fog computing environment," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 72–78, Nov. 2017.

[10] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Merwe, and K. Webb, "MobiScud: A fast moving personal cloud in the mobile network," in *Proc. 5th Workshop All Things Cellular: Operations Appl. Challenges*, 2015, pp. 19–24.

[11] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 369–382, Second Quarter 2019.

[12] I. Labriji et al., "Mobility aware and dynamic migration of MEC services for the internet of vehicles," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 570–584, Mar. 2021.

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

[14] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surv. Tuts.*, vol. 20, no. 2, pp. 1206–1243, Second Quarter 2018.

[15] H. Abdah, J. P. Barraca, and R. L. Aguiar, "QoS-aware service continuity in the virtualized edge," *IEEE Access*, vol. 7, pp. 51570–51588, 2019.

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

[17] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

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

[19] J. Li et al., "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, vol. 7, pp. 13704–13714, 2019.

[20] F. Xu, F. Liu, L. Liu, H. Jin, B. Li, and B. Li, "iAware: Making live migration of virtual machines interference-aware in the cloud," *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3012–3025, Dec. 2014.

[21] R. Chiang and H. H. Huang, "TRACON: Interference-aware schedulingfor data-intensive applicationsin virtualized environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1349–1358, May 2014.

[22] Q. Zhang, F. Liu, and C. Zeng, "Adaptive interference-aware VNF placement for service-customized 5G network slices," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2449–2457.

[23] ETSI GS MEC 003 V2.1.1, "Multi-access edge computing (MEC); Framework and reference architecture," Nov. 2019.

[24] Y. Peng, L. Liu, Y. Zhou, J. Shi, and J. Li, "Deep reinforcement learning-based dynamic service migration in vehicular networks," in *Proc. IEEE Glob. Commun. Conf.*, 2019, pp. 1–6.

[25] I. Anbagi, M. Kantarci, and H. Mouftah, "A survey on cross-layer quality-of-service approaches in WSNs for delay and reliability-aware applications," *IEEE Commun. Surv. Tuts.*, vol. 18, no. 1, pp. 525–552, First Quarter 2016.

[26] V. Medina and J. M. Garcia, "A survey of migration mechanisms of virtual machines," *ACM Comput. Surv.*, vol. 46, 2014, Art. no. 30.

[27] H. Maziku and S. Shetty, "Towards a network aware VM migration: Evaluating the cost of VM migration in cloud data centers," in *Proc. IEEE Int. Conf. Cloud Netw.*, 2014, pp. 114–119.

[28] G. Xiao, "Better live migration," 2017. [Online]. Available: https://events19.lfasiallc.com/wp-content/uploads/2017/11/Better-Live-Migration-on-KVM_QEMU_Guangrong-Xiao.pdf

[29] M. Chen, W. Saad, C. Yin, and M. Debbah, "Data correlation-aware resource management in wireless virtual reality (VR): An echo state transfer learning approach," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4267–4280, Jun. 2019.

[30] R. Li, Z. Zhao, X. Chen, J. Palicot, and H. Zhang, "TACT: A transfer actor-critic learning framework for energy saving in cellular radio access networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 4, pp. 2000–2011, Apr. 2014.

[31] D. Ghosh, J. Rahme, A. Kumar, A. Zhang, R. Adams, and S. Levine, "Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 25502–25515.

[32] M. Laskin et al., "URLB: Unsupervised reinforcement learning benchmark," 2021, *arXiv:2110.15191*.

[33] L. Wang, Q. Weng, W. Wang, C. Chen, and B. Li, "Metis: Learning to schedule long-running applications in shared container clusters at scale," in *Proc. IEEE Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2020, pp. 1–17.

[34] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[35] T. Nguyen, N. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[36] V. Mnih et al., "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[38] K. Louie, E. Grattan, and W. Glimcher, "Reward value-based gain control: Divisive normalization in parietal cortex," *J. Neurosci.*, vol. 31, no. 29, pp. 10627–10639, 2011.

[39] T. Schaul, J. Quan, I. Antonoglou, and D. Sliver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Representations*, Nov. 2016, pp. 1–6.

[40] M. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jun. 2009.

[41] X. Cao, "A practical theory for designing very deep convolutional neural networks classifier level," *Sci. Res.*, Mar. 2015.

[42] X. Yu, M. Guan, M. Liao, and X. Fan, "Pre-migration of vehicle to network services based on priority in mobile edge computing," *IEEE Access*, vol. 7, pp. 3722–3730, 2019.

Yan Peng received the BS degree in communication engineering from Shandong Normal University, in 2016, and the PhD degree in computer science and technology from the University of Chinese Academy of Sciences, in 2022. Her research focuses on the convergence of communication, computing, and artificial intelligence.

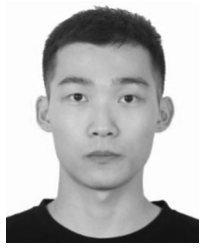
Xiaogang Tang received the MS and PhD degrees from Xi'an Jiaotong University, Xi'an, Shann'xi, China, in 2005 and 2014. Now, he is an assistant professor of Space Engineering University, and his research is mainly about pattern recognition and information intelligent processing. He has published several research papers in scholarly journals in the above research areas and has participated in several conferences.

Yiqing Zhou (Senior Member, IEEE) received the BS degree in communication and information engineering and the MS degree in signal and information processing from Southeast University, China, in 1997 and 2000, respectively, and the PhD degree in electrical and electronic engineering from the University of Hong Kong, Hong Kong, in 2004. She is currently a professor with the State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences. She has published more than 150 articles and four books/book chapters in the areas of wireless mobile communications. She received best paper awards from WCSP2019, IEEE ICC2018, ISCIT2016, PIMRC2015, ICCS2014, and WCNC2013. She also received the 2014 Top 15 Editor Award from IEEE TVT and the 2016-2017 Top Editors of ETT. She is also the TPC co-chair of ChinaCom2012, an executive co-chair of IEEE ICC2019, a symposia co-chair of ICC2015, a symposium co-chair of GLOBECOM2016 and ICC2014, a tutorial co-chair of ICCC2014 and WCNC2013, and the workshop co-chair of SmartGridComm2012 and GlobeCom2011. She is also the associate/guest editor of the *IEEE Internet of Things Journal*, *IEEE Transactions on Vehicular Technology* (TVT), *IEEE Journal on Selected Areas in Communications* (JSAC) (Special issue on "Broadband Wireless Communication for High Speed Vehicles" and "Virtual MIMO"), *Transactions on Emerging Telecommunications Technologies* (ETT), and *Journal of Computer Science and Technology* (JCST).

Jintao Li (Member, IEEE) received the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 1989. He is currently a professor with the Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include multimedia technology, virtual reality technology, and pervasive computing.

Yanli Qi received the PhD degree in computer science and technology from the University of Chinese Academy of Sciences, Beijing, China, in 2021. She is currently an assistant professor with the State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing. Her research focuses on mobile edge computing, the convergence of sensing, communication, and computation, and resource management. She has also served as a reviewer for a number of referred journals and international conferences.

Ling Liu received the BS degree in communication engineering from Nanchang University, in 2012, and the PhD degree in computer science and technology from the University of Chinese Academy of Sciences, in 2018. She is currently an associate professor with the Wireless Communication Research Center, Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include interference and resource management in ultra-dense networks, and the convergence of communication, computing, and caching. She has also served as a reviewer for a number of referred journals and international conferences. She has received the Best Paper Award from the IEEE ICC 2018.



Hai Lin is currently working toward the master's degree with Space Engineering University, and his research is mainly about Intelligent optimization algorithm.