Microservice-Oriented Service Placement for Mobile Edge Computing in Sustainable Internet of Vehicles

Leilei Wang, Xiaoheng Deng, *Senior Member, IEEE*, Jinsong Gui, *Member, IEEE*, Xuechen Chen, *Member, IEEE*, and Shaohua Wan, *Senior Member, IEEE*

***Abstract*— The integration of Mobile Edge Computing (MEC) and microservice architecture drives the implementation of the sustainable Internet of Vehicles (IoV). The microservice architecture enables the decomposition of a service into multiple independent, fine-grained microservices working independently. With MEC, microservices can be placed on Edge Service Providers (ESPs) dynamically, responding quickly and reducing service latency and resource consumption. However, the burgeoning of IoV leads to high computation and resource overheads, making service resource requirements an imminent issue. What's more, due to the limited computation power of ESPs, they can only host a few services. Therefore, ESPs should judiciously decide which services to host. In this paper, we propose a Microservice-oriented Service Placement (MOSP) mechanism for MEC-enabled IoV to shorten service latency, reduce high resource consumption levels and guarantee long-term sustainability. Specifically, we formulate the service placement as an integer linear programming program, where service placement decisions are collaboratively optimized among ESPs, aiming to address spatial demand coupling, service heterogeneity, and decentralized coordination in MEC systems. MOSP comprises an upper layer to map the service requests to ESPs and a lower layer to adjust the service placement of ESPs. Evaluation results show that the microservice-oriented service deployment mechanism offers dramatic improvements in terms of resource savings, latency reduction, and service speed.**

***Index Terms*— Mobile edge computing, microservice, Internet of Vehicles, service placement.**

Manuscript received 6 October 2022; revised 16 February 2023 and 17 April 2023; accepted 24 April 2023. Date of publication 17 May 2023; date of current version 30 August 2023. This work was supported in part by the National Natural Science Foundation of China Projects under Grant 62172441, Grant 62172449, and Grant 61772553; in part by the Local Science and Technology Developing Foundation Guided by the Central Government through the Free Exploration Project 2021Szvup166; in part by the Postgraduate Research and Innovation Project of Hunan Province under Grant CX20200211; in part by the Fundamental Research Funds for the Central Universities of Central South University under Grant 2020zzts138; in part by the Opening Project of State Key Laboratory of Nickel and Cobalt Resources Comprehensive Utilization under Grant GZSYS-KY-2022-018 and Grant GZSYS-KY-2022-024; in part by the Key Project of Shenzhen City Special Fund for Fundamental Research under Grant 202208183000751; and in part by the Hunan Provincial Department of Transportation Foundation under Grant 201929. The Associate Editor for this article was H. Wu. *(Corresponding author: Xiaoheng Deng.)*

Leilei Wang, Jinsong Gui, and Xuechen Chen are with the School of Computer Science and Engineering, Central South University, Changsha 410075, China (e-mail: llwang@csu.edu.cn; jsgui2010@csu.edu.cn; chenxuec@csu.edu.cn).

Xiaoheng Deng is with the School of Computer Science and Engineering, Central South University, Changsha 410075, China, and also with the Shenzhen Research Institute, Shenzhen 518057, China (e-mail: dxh@csu.edu.cn).

Shaohua Wan is with the Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China (e-mail: shaohua.wan@uestc.edu.cn).

Digital Object Identifier 10.1109/TITS.2023.3274307

I. INTRODUCTION

THE unprecedented proliferation of IoV has led to a sizeable set of modern intelligent transportation applications, such as traffic object detection, immersive communication, traffic environment awareness, Virtual/Augmented Reality (VR/AR), and path planning. Those applications are becoming increasingly latency-sensitive, computation-demanding, and data-hungry [1], [2]. Meanwhile, cloud computing [3] has gained much attention with its ability to provide elastic computing power and storage for resource-constrained devices in the last decade. However, aggregating all computation-intensive applications and distributed data into the cloud imposes a heavier burden on the already congested backbone network, thereby reducing the Quality of Service (QoS). To bridge this gap, Mobile Edge Computing (MEC) [4], a novel computing paradigm, sinks the resources such as computing and storage at the edge of the network, which eases the backbone network's transmission burden while improving the QoS.

Although computational offloading, resource allocation, and edge caching for MEC achieves tremendous research, the diversity and heterogeneity of services and how to place the service on the ESP [5] are often overlooked. The placement of each service affects the load on the ESPs and communication links, resulting in increased execution latency for existing services located on the same physical node or link. Unlike cloud computing [3], the limited resources and computing power of edge servers allow only a small group of services to be placed at a time [6]. While ESPs can adapt their service placement to different requirements, the heterogeneous, time-varying demands, especially when multi-users access the same ESP, lead to frequent service placement and tear operations [7]. That situation results in extra traffic load and service latency to the core network, reducing the Quality of Experience (QoE). Therefore, service placement is a long-term decision compared to computation offloading. To optimize the performance of ESPs, demand processing, and network effectively, we must intelligently decide which services to place on resource-constrained ESPs based on service requirements.

Parallel to MEC, the microservice is driven by container technology [8], which enables the decomposition of an application into multiple independent fine-grained components and running each component as a service. These services are built around business functions, and each service performs one function. As they operate independently, it is possible to update, deploy and extend each service to satisfy the demands for application-specific functionality. Unlike monolithic architectures, microservice has the advantages of flexibility, scalability, agility, and autonomy, and all components are decoupled and operate independently. As a result, microservices can be deployed, migrated, and deleted on demand while the application runs [9]. On the network side, MEC empowers the network edge with computing and storage capabilities, making it intelligent. On the service side, microservices decouple applications, allowing services to run in parallel. Integrating microservice and MEC into IoV brings more feasibility. First, business requirements can be sunk to the edge of the network for execution. In addition, services on ESPs can be dynamically placed [10].

Along with the benefits of microservice and MEC, microservices can further be dynamically and adaptively assigned to ESPs with low latency and low overhead, providing better service for vehicles in the vicinity. A simple example is that self-driving vehicles need to perceive traffic environment service in real-time to make effective decisions and ensure driving safety. The self-driving vehicle starts and sends a service request to the ESP, which receives the service request and provides the vehicle with comprehensive services, such as traffic sign detection service, lane detection service, traffic warning service, and target detection service. Each of these services requires complex computation and occupies a certain amount of memory space, then if all the services are placed on the ESP as a whole, it requires enormous computing power and huge memory space. However, vehicles can only send one request containing one task each time, and more importantly, the computational power and memory space of vehicles and ESPs are limited. Therefore, we seek to decompose the traffic information perception service into fine-grained microservices. That is, traffic information perception service can be decomposed into traffic sign detection microservice, lane detection microservice, traffic warning microservice, traffic target detection microservice, etc., running in parallel to achieve a quick response. Then, we leverage MEC to allocate vehicle requests and place the required microservices on ESPs.

Being motivated, we pursue to analyze the network resource consumption caused by the service placement from the perspective of multi-objective optimization and then adaptively implement the service placement for IoV. Consideration is systematically given to resource constraints and service shareable for efficient resource utilization. Moreover, considering the time-varying network load, we design a dynamic service placement strategy (place services or release redundant services) to maintain network performance, guarantee service latency, or save network resources. Then, to minimize resource consumption while ensuring latency requirements, we mathematically formulate the service placement problem as a multi-objective optimization problem where latency, ESP resources (CPU and memory), and bandwidth consumption conflict and need optimization to obtain a balance among them. Considering the differentiation among the objectives, we employ the normalization approach to further enhance the service placement problem. In this context, the NP-hard service placement problem is formulated as an integer linear programming problem and cannot be solved efficiently by traditional optimization methods. Inspired by the intelligent algorithm, we propose a microservices-oriented service placement mechanism with two layers: an upper layer and a lower layer, where the upper layer aims to map service requests to ESPs, and the lower layer is employed to adjust the service placement of ESPs. The main contributions of our research are summarized as follows:

- We investigate microservice-oriented service placement with the long-term sustainability of service resources in terms of the service requests, network resources, and service capability of ESPs. We formulate it as an integer linear programming model where resource utilization, ESP resource consumption, and network load are synthetically considered.
- To efficiently exploit the service latency, ESP resource consumption, and network resource consumption, we consider two factors that have been overlooked in most related work: the primary resource consumption of ESP and the shareable service mechanism and employ microservices to implement service placement.
- Optimizing the number of placed services by releasing idle services to accommodate time-varying network loads. When the network load is high, reducing the number of released redundant services ensures service latency while improving network performance. Conversely, when the network load is low, increasing the number of released redundant services contributes to reducing resource consumption.
- We employ the polynomial-time method to study the network resource utilization behavior during service placement while adopting a microservices-oriented mechanism to improve the service placement performance. A two-tier microservice-oriented service placement mechanism is designed, and evaluation results demonstrate that the MOSP mechanism can achieve near-optimal performance.

The outline of this paper is structured as follows. In Section II, the related work of service placement is introduced. Section III introduces the proposed system model. The proposed approach to MOSP is presented in Section IV. Measurement and evaluation are presented in Section V. The final section concludes this paper.

II. Related Work

This section provides an overview and investigation of the research on service placement. There have been considerable studies on service placement in cloud computing and fog computing environments, but they are not intuitively considered the edge service placement for IoV.

MEC as a new computing paradigm has received widespread attention from academia and industry. In MEC, computation offloading [11], [12], server deployment [13] and edge caching [14] are hot issues. These issues involve where/when/how to choose an edge server. However, the assumption implicit in these questions is that the edge server can handle any computational task sent from the terminal, regardless of the availability of the service in the edge server, which is critical in edge computing.

Natesha et al. [15] designed a two-stage fog resource provisioning framework by leveraging the doctor container to provide resources and host Internet of Things (IoT) application requests in a fog environment. Intuitively, the authors define the service placement problem as a multi-objective optimization problem that minimizes the service time, cost, and energy consumption between two-phase fog computing infrastructures and guarantee QoS regarding deadlines and service costs. Considering the networks' priority and energy consumption in cloud-fog computing environments, Hassan et al. [16] designed a heuristic scheme to place IoT services. However, the authors only consider IoT applications that contain at most two services, ignoring resource availability and service placement costs. Coordinating application time in a foggy environment is challenging for time-sensitive IoT applications. To alleviate this problem, Sriraghavendra et al. [17] proposed a new multi-tier fog computing architecture called Deadline-oriented Service Placement (DoSP), which jointly considers service response time and resource overhead. Kim [18] proposed a fog computing infrastructure called a fog portal and designed an optimal service placement scheme based on user participation in the infrastructure. To optimize the consumption of fog resources, the authors formulated the service placement problem as a mixed integer linear programming problem. However, the study ignores the service delay and available resource utilization.

Service placement in edge computing has attracted much attention until recently. For the MEC-enabled dense small-cell network environment, the literature [19] formulated collaborative service placement as a utility maximization problem. A distinctive feature of collaborative service placement is that it works in a decentralized manner and is well-suited for geographically distributed edge servers. Considering the edge server and wireless link resource constraints, the authors proposed decomposing the extensive services into multiple interconnected components, with multiple edge servers collaborating to provide services. For data flow management between services, the authors in [20] studied and constructed the joint problem of multi-copy service placement and data flow management to optimize services to prevent cost and service latency.

Considering the tight coupling of components in the monolithic service architecture and the mobility of users, the authors of [10] designed a service orchestration scheme that incorporates MEC and microservice to achieve seamless and real-time response for mobile users' requirements. To alleviate the coupling problem between the original "triple bindings" of the Internet, the literature [21] investigated the Virtual Network Functions (VNF) placement problem in SDN/NFV-enabled networks. The authors formulated the problem naturally as a binary integer planning problem and designed a Double Deep Q Network-based VNF Placement Algorithm (DDQN-VNFPA). The approach [22] studied the problem involving perceptual online multi-component service placement in edge cloud networks. Considering the inter-component dependencies, the authors map the component relationships into a tree service placement on which the variations of links, traffic, physical nodes, and service delays are defined and expressed. Salaht et al. [23] proposed an explicit scheme for service placement from four perspectives, which are control plane and coordination (centralized and distributed), implantation characteristics (online or offline), system nature (dynamic or static), and mobile support (yes or no).

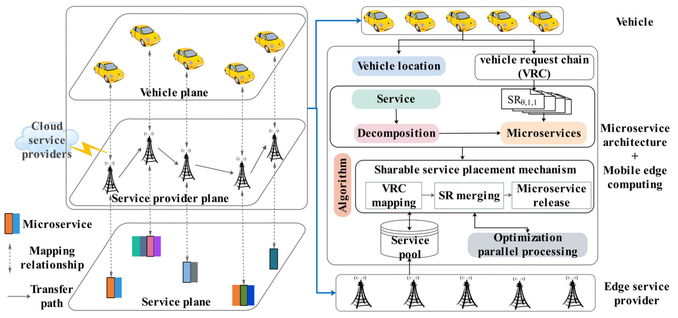


Fig. 1. Microservice-oriented service placement in MEC-enabled IoV.

As aforementioned, researchers have produced extensive work on service placement aiming to shorten service latency and diminish resource optimization. However, due to the complexity of the constraints, the above algorithms seem difficult to apply directly to our problem. In addition, most of the above studies ignore redundant services, service sharing, and network load time variability. Therefore, we propose a more exclusive service placement mechanism to ensure feasibility.

III. System Model And Problem Description

In this section, we formally describe the microservice-oriented service placement problem with a classic example and systematically formulate and define the problem.

A. System Model

To better implement the microservice-oriented service placement in the MEC environment, a system framework is designed, as depicted in Fig. 1, where it consists of three planes. The vehicle plane is a series of request initiators. The service provider plane consists of Cloud Service Providers (CSPs) and ESPs aiming to provide the required services to the vehicles. The service plane describes the microservices placed on each ESP used to serve the vehicles' requests, where the solid line indicates that the microservice is being executed.

By leveraging the microservice, the service can be decomposed into multiple independent fine-grained microservices, which can be executed in parallel. Then, MEC enables the adaptive placement of these microservices on the appropriate ESP for providing sustainable service in long run. In fact, each ESP can receive multiple requests from multiple vehicles. Therefore, we leverage the Vehicle Request Chain (VRC) to represent the Service Requests (SRs) received by ESPs from multiple vehicles. To enable the three planes to collaborate and achieve resource optimization, we design a microservice-oriented service placement mechanism that includes VRCs mapping, SR merging, and microservice release. The CSP is responsible for all ESPs in its coverage area and provides the image files of different types of microservices to ESPs. The ESP receives the image file and decides whether to place that type of microservice based on the VRC. The ESP provides services to the vehicle.

Let $G = (N \cup K, E)$ denote the MEC-enabled IoV, where N denotes the scale of ESPs, K denotes the collection of CSPs, and E denotes the set of physical links between ESPs. Let R indicate the collection of VRCs in the network at each moment and define the five-tuple $VRC_r \triangleq (ing_r, eg_r, H_r, l_r, D_r^{max})$ to represent each VRC_r, where ing_r and eg_r denote the ingress and egress ESPs of VRC_r, respectively. H_r represents the service requests (SRs, $SR_{\theta,i,r}$, where $\theta \in \{a, b, \ldots, \Theta\}$ stands for the type of microservice required by SR_i, r), L_r is a collection of links between SRs in VRC_r. $D_{i,r}^{max}$ is the maximum tolerated delay of $SR_{\theta,i,r}$ in VRC_r.

In the service placement mechanism that incorporates MEC and microservice, each VRC can access the network from a different ESP. To get a real-time response, VRCs are usually connected to nearby ESPs. In our problem, considering the real-time availability of VRCs, we define that the ingress and egress of each VRC must be restricted to the same ESP. For each SR, we are responsible for developing a microservice placement scheme and mapping that scheme to the services of the corresponding ESPs.

B. Microservice-Oriented Service Placement

The flexibility of the microservice-oriented service placement can drastically reduce the service latency, improve resource utilization and achieve the long-term sustainable run of service resources. For a service, the resource consumption for placing a service on the ESP is considered the Primary Resource Consumption (PRC) of ESP. Furthermore, the PRC of a service can be shared by other identical SRs, even if these SRs belong to different VRCs. Therefore, for the same type of microservice, we sometimes only need to place one in the network. Notably, for some specific types of microservices, there may be more than one throughout the network.

It is known that the vehicles' requests and microservice types are diverse. For microservice-oriented service provisioning, one service can be decomposed into multiple fine-grained microservices, which in turn enables flexible placement of services. For each newly arriving vehicle's request, we have to make an advisable decision whether to choose the already placed microservice or to re-place the new microservice on the ESPs. Different service placement mechanisms evoke drastic changes in service latency and resource consumption, and they conflict with each other. This study is dedicated to investigating the efficient service placement on ESP with respect to the minimization of service latency and resource consumption.

C. Problem Statement

A classic case of microservice-oriented service placement as depicted in Fig. 2, which is a network topology with 9 ESPs.

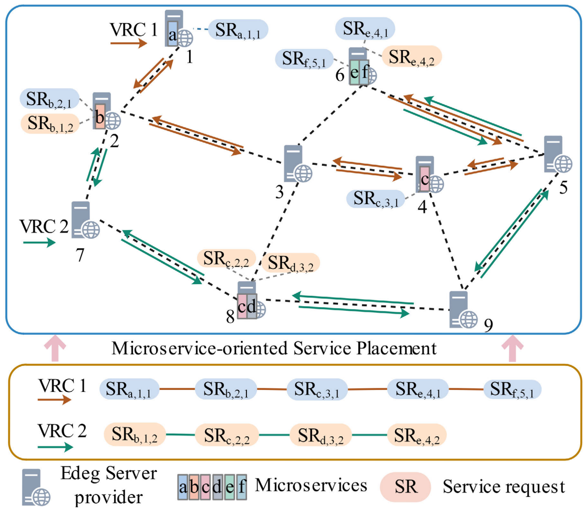


Fig. 2. A classic case of service placement.

By leveraging the microservice, VRC_1 and VRC_2 comprise five and four SRs, respectively. VRC_1 and VRC_2 access the network from ESP_1 and ESP_7, respectively. The orange and green arrows indicate the direction of information traffic for VRC_1 and VRC_2. The initial microservices and SRs mapping scheme is shown in Fig. 2. Then, the information traffic of VRC_1 is: 1-2-3-4-5-6, and the information traffic of VRC_2 is: 7-2-8-9-5-6.

The PRC of an ESP depends on the number of microservices placed on that ESP. If an ESP places more types of microservices, the higher the PRC of that ESP. Therefore, a feasible perspective is service sharing. For the microservice-oriented shareability feature, the key to service placement is to map $SR_{c,3,1}$ to ESP_8, i.e., $SR_{c,3,1}$ and $SR_{c,2,2}$ share the same microservice MS_c. Thus, the information traffic of VRC_1 becomes: 1-2-7-8-9-4-5-6.

Although the usage of shareable services enables the PRC of the ESP to be reduced, it requires that VRC_1 has to go through two more ESPs, leading to more ESP resource consumption and bandwidth resource consumption. In addition, the service latency of VRCs with respect to the resource utilization of the corresponding ESP. The higher resource utilization of an ESP indicates the higher processing latency of VRCs mapped on that ESP. Due to more types of microservices mapped on the ESP, the resource utilization of the ESP increases, resulting in higher processing latency for VRC_1 and VRC_2, which may affect the service level of the ESP.

Therefore, this paper aims to balance the PRC of ESP, the bandwidth consumption of links, and the latency of VRC. In this paper, for each moment, the ESPs receive a VRC with different vehicle requests (route planning, target detection, traffic monitoring, and entertainment services). Each request requires a real-time response from the corresponding type of microservice. We can assign multiple identical SRs to the same microservice. In this context, we can reduce the PRC of the ESP by reducing the types of microservices placed on the ESP. Regardless, there are two problems that must be solved, i. e., how to reasonably determine which microservices to place on ESPs and which SRs can share the same microservice.

D. Problem Formulation

In this section, we formulate the latency guarantee and resource optimization for the service placement problem. To guarantee service latency, we define and formulate the service latency for the service. Then, we explore and formulate resource consumption for service placement.

1) Service Latency: Service latency indicates the VRC processing latency, which specifically stands for the total time between the ESP receiving the VRC and getting feedback, including: link latency (transmission latency and propagation latency), queuing latency and ESP execution latency.

Transmission latency. The transmission latency contains the transmission latency of SR ($d^{Stran}_{(v,s,q)}$) and the transmission latency of processing result ($d^{Rtran}_{(q,s,v)}$). The SR transmission latency is the entire time consumption of the SR transmission from the vehicle to the ESP, which can be defined as:

$$d^{Stran}_{(v,s,q)} = pt_i/\Delta B^{link}_{(v,s)} + \mu \cdot h_{(s,q)} \cdot pt_i/\Delta B^{link}_{(s,q)}, \quad (1)$$

where, pt_i is the packet size of $SR_{\theta,i,r}$ in VRC_r. $\Delta B^{link}_{(v,s)} = TB^{link}_{(v,s)} - OB^{link}_{(v,s)}$, where $TB^{link}_{(v,s)}$ and $OB^{link}_{(v,s)}$ indicate the total and occupied bandwidth resources of link (v, s). $\Delta B^{link}_{(s,q)} = TB^{link}_{(s,q)} - OB^{link}_{(s,q)}$, where $TB^{link}_{(s,q)}$ and $OB^{link}_{(s,q)}$ indicate the total and occupied bandwidth resources of link (s, q). μ is a binary variable indicating that the $SR_{\theta,i,r}$ needs to be transferred from ESP s to another ESPs, if the ESP s cannot handle the $SR_{\theta,i,r}$ of VRC_r, $\mu = 1$. Otherwise, $\mu = 0$. $h_{(s,q)}$ is the number of communication hops between ESP s and ESP q.

Similarly, the transmission latency of the processing result can be expressed as:

$$d^{Rtran}_{(q,s,v)} = \mu \cdot h_{(q,s)} \cdot D^*_i/\Delta B^{link}_{(s,q)} + D^*_i/\Delta B^{link}_{(s,v)}, \quad (2)$$

where, $D^*_{i,r}$ is the data size of the processing result of $SR_{\theta,i,r}$.

Propagation latency. We define $d^{Sprop}_{(v,s,q)}$ and $d^{Rprop}_{(q,s,v)}$ to denote the time consumption for SRs and processing results to propagate through the communication medium, respectively. The propagation latency depends on the distance and the propagation speed. Therefore, $d^{Sprop}_{(v,s,q)}$ and $d^{Rprop}_{(q,s,v)}$ can be defined as:

$$\begin{aligned} d^{Sprop}_{(v,s,q)} &= L_{(v,s)}/S^{med}_{(v,s)} + L_{(s,q)}/S^{med}_{(s,q)} \\ d^{Rprop}_{(q,s,v)} &= L_{(q,s)}/S^{med}_{(q,s)} + L_{(s,v)}/S^{med}_{(s,v)}, \end{aligned} \quad (3)$$

where, $L_{(v,s)}$ and $L_{(q,s)}$ are the lengths of link (v, s) and link (s, q), which can be expressed by Euclidean distance. $S^{med}_{(s,q)}$ and $S^{med}_{(s,v)}$ are the propagation speeds of signals in different link media. Therefore, the latency of link (v, s, q) can be denoted as:

$$d_{(v,s,q)} = d^{tran}_{(v,s,q)} + d^{prop}_{(v,s,q)}. \quad (4)$$

Queuing latency. let $d^{queu}_{(v,s,q)}$ denotes the queuing latency of an SR. We employ the M/M/1 queuing model to capture the time consumption of an SR throughout the link. Specifically, the queuing latency can be calculated as:

$$\begin{aligned} d^{queue}_{(v,s,q)} &= d^{Squeue}_{(v,s,q)} + d^{Rqueue}_{(q,s,v)} \\ &= OB^{link}_{(v,s)}/\Delta B^{link}_{(v,s)} * d^{Stran}_{(v,s)} + OB^{link}_{(s,q)}/\Delta B^{link}_{(s,q)} * d^{Stran}_{(s,q)} \\ &\quad + OB^{link}_{(q,s)}/\Delta B^{link}_{(q,s)} * d^{Rtran}_{(q,s)} + OB^{link}_{(s,v)}/\Delta B^{link}_{(s,v)} \\ &\quad * d^{Rtran}_{(s,v)}. \end{aligned} \quad (5)$$

Note that $OB^{link}_{(v,s)}$ and $OB^{link}_{(s,q)}$ change dynamically with the number of SRs mapped on link (v, s) and link (s, q). This means that the queuing latency is determined by the number of SRs on the link. Therefore, the assignment of $SR_{\theta,i,r}$ in link (v, s, q) affects the mapping of other SRs on link (v, s, q).

Execution latency. When one $SR_{\theta,i,r}$ in VRC_r is successfully received by ESP s, then ESP s executes the $SR_{\theta,i,r}$. Therefore, the execution latency of $SR_{\theta,i,r}$ at the ESP s can be defined as:

$$d^{proc}_{s,i} = ca_i/(TC^{cpu}_s - OC^{cpu}_s), \quad (6)$$

where, $s \in N, i \in H_r$. OC^{cpu}_s stands for the occupied CPU resource of ESP s. TC^{cpu}_s is the total CPU resource of ESP s. ca_i reflects the computational amount of $SR_{\theta,i,r}$. Notably, OC^{cpu}_s changes dynamically with the number of SRs mapped on the ESP s. This indicates that the execution latency of $SR_{\theta,i,r}$ on ESP s is defined with consideration of all SRs on ESP s. Therefore, the allocation of $SR_{\theta,i,r}$ in the ESP affects the mapping of other SRs on the ESP.

2) Resource Consumption: One $SR_{\theta,i,r}$ in VRC_r can select one and only one microservice of an ESP to be hosted, i.e., $\sum_{s\in N} \alpha^r_{\theta,s,i} = 1, i \in H_r, r \in R$. Moreover, the ingress and egress ESP of each VRC must be the same ESP and its information traffic should be linked head-to-tail, i.e., $\sum_{s\in N} \alpha^r_{\theta,s,i} = 1,\ i \in \{ing_r, e\,g_r\}\ \&\alpha^r_{ing_r,s} = \alpha^r_{eg_r,s}, s \in N, r \in R$. Intuitively, the resource overhead for service placement encompasses two parts. One part involves placing the service on the ESP, i.e., the PRC of ESP including the required CPU resource pc^{cpu}_θ and memory resource pc^{mem}_θ. The other part is the resource consumption for serving $SR_{\theta,i,r}$ including CPU processing resource $cpu^r_{s,i}$ and memory resource $mem^r_{s,i}$. Consequently, we define the variables related to the PRC of the ESP as:

$$\chi^\theta_s = \begin{cases} 1, & \sum_{\forall r\in R} \sum_{\forall i\in \forall r} \alpha^r_{\theta,s,i} \cdot \beta^r_{\theta,s} \geq 1 \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

The above equation indicates whether a type-θ microservice is placed on ESP s, where $s \in N$ and $\beta^r_{\theta,s}$ stands for whether $SR_{\theta,i,r}$ in VRC_r demand a type-θ microservice. If the SR requires the type-θ microservice, $\beta^r_{\theta,s} = 1$. Otherwise, $\beta^r_{\theta,s} = 0$.

Moreover, we leverage binary variables to denote whether the ESP is activated:

$$\delta_s = \begin{cases} 1, & \sum_{\forall r\in R} \sum_{\forall i\in \forall H_r} \alpha^r_{\theta,s,i} \geq 1 \\ 0, & \text{otherwise,} \end{cases} \quad (8)$$

where $s \in N$. The Eq. 8 reflects that there is at least one SR mapped to a type-θ microservice on ESP s, and this type-θ microservice will likely be involved in completing the SR.

Each SR corresponds to a service link. Each SR-induced service link is capable of being mapped on a fixed physical link. We define the link variables as:

$$L_{(v,s)}^{i,r} = \begin{cases} 1, & i \text{ is mapped on } (v,s) \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where (v, s) stands for the service link of $SR_i, i \in r$. And (v, s) is a physical link between vehicle v and ESP s.

Each VRC corresponds to a service link. Each VRC-induced service link is capable of being mapped on a fixed physical link. We define the link variables as:

$$l_{(s,q)}^{r,(i,j)} = \begin{cases} 1, & (i,j) \text{ is mapped on } (s,q) \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where (i, j) stands for the service link between $SR_{\theta,i,r}$ and $SR_{\theta,j,r}$ in VRC_r, $(s, q) \in E$ is a physical link between two ESP s and ESP q.

For each VRC, the SRs in an VRC_r may be mapped to an ESP or different ESPs. Briefly, a portion of the SRs in the VRC may be restricted to one ESP without involving additional links. Therefore,

$$\sum_{r\in R}\sum_{(i,j)\in L_r}\sum_{\forall(s,q)\in E} l_{(s,q)}^{r,(i,j)} \geq 0, \tag{11}$$

where 0 implies that the traffic of $SR_{\theta,i,r}$ and $SR_{\theta,j,r}$ is restricted in one ESP.

We specify the following rules to determine whether $SR_{\theta,i,r}$ and $SR_{\theta,j,r}$ are mapped to different ESPs:

$$\alpha_{\theta,s,i}^{r} - \alpha_{\theta,s,j}^{r} = \begin{cases} 1, & i \text{ is placed on } s \\ -1, & j \text{ is placed on } s \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

where 0 reflects that $SR_{\theta,i,r}$ and $SR_{\theta,j,r}$ are mapped to the same ESP.

To keep the flow conservation for each VRC and ensure that each service link can be mapped to a physical link, we formulate the following conditions,

$$\sum_{r\in R}\sum_{(i,j)\in L_r}\sum_{(s,q)\in E} l_{(s,q)}^{r,(i,j)} - l_{(q,s)}^{r,(i,j)} = \alpha_{\theta,s,i}^{r} - \alpha_{\theta,s,j}^{r}. \tag{13}$$

Along with the aforementioned analysis, the total CPU resource consumption of ESPs can be given by:

$$\mathbb{P} = \sum_{\forall r\in R}\sum_{\forall i\in H_r}\sum_{s\in N} cpu_{s,i}^{r}\cdot\alpha_{\theta,s,i}^{r} + \sum_{\forall\theta\in\Theta}\sum_{s\in N} pc_{\theta}^{cpu}\cdot\chi_{s}^{cpu}. \tag{14}$$

Similarly, the entire memory resource consumption of ESPs can be defined as:

$$\mathbb{M} = \sum_{\forall r\in R}\sum_{\forall i\in H_r}\sum_{s\in N} mem_{s,i}^{r}\cdot\alpha_{\theta,s,i}^{r} + \sum_{\forall\theta\in\Theta}\sum_{s\in N} pc_{\theta}^{mem}\cdot\chi_{s}^{\theta}. \tag{15}$$

Likewise, the overall bandwidth overhead for links is denoted as:

$$\mathbb{B} = \sum_{\forall r\in R}\sum_{\forall(i,j)\in L,}\sum_{(s,q)\in E} b_{(i,j)}^{r}\cdot l_{(s,q)}^{r,(i,j)} + \sum_{\forall r\in R}\sum_{s\in N} b_{(v,s)}^{i,r}\cdot L_{(v,s)}^{i,r}, \tag{16}$$

where $b_{(i,j)}^{r}$ is the bandwidth consumption of the physical links of the ESPs mapped by $SR_{\theta,i,r}$ and $SR_{\theta,j,r}$ in a VRC_r. $b_{(v,s)}^{i,r}$ is the bandwidth consumption of the physical link between vechile and ESP.

Finally, the consumption of activating ESP is:

$$\mathbb{S} = \sum\nolimits_{s\in N} \varepsilon\cdot\delta_s, \tag{17}$$

where ε is a parameter.

E. Optimization Objective

The ultimate objective is to explore the optimal service placement mechanism that minimizes the total service latency and resource consumption. As aforementioned, we formulate the optimization objective as:

$$\begin{aligned} &Minimize: F(\mathrm{X}) = (f_1(\mathrm{X}), f_2(\mathrm{X}), f_3(\mathrm{X}), f_4(\mathrm{X}))^T \\ &\text{s.t.} \\ &\text{C1.} \sum_{\forall r\in R}\sum_{\forall i\in H_r}\sum_{s\in N} cpu_{s,i}^{r}\cdot\alpha_{\theta,s,i}^{r} + \sum_{\forall\theta\in\Theta} pc_{\theta}^{cpu}\cdot\chi_{s}^{cpu} \leq TC_{s}^{cpu} \\ &\text{C2.} \sum_{\forall r\in R}\sum_{\forall i\in H_r}\sum_{s\in N} m_{em}^{r}\cdot\alpha_{\theta,s,i}^{r} + \sum_{\forall\theta\in\Theta} pc_{\theta}^{mem}\cdot\chi_{s}^{\theta} \leq TM_{s}^{mem} \\ &\text{C3.} \sum_{\forall r\in R}\sum_{\forall(i,j)\in L_r} b_{(i,j)}^{r}\cdot l_{(s,q)}^{r,(i,j)} \leq TB_{(s,q)}^{link} \\ &\text{C4.} d_{(v,s,q)}\cdot l_{(s,q)}^{r,(i,j)} + d_{s,i}^{proc}\cdot\alpha_{\theta,s,i}^{r} \leq D_{i,r}^{\max} \\ &\text{C5.} \sum_{\forall r\in R}\sum_{s\in N} b_{(v,s)}^{i,r}\cdot L_{(v,s)}^{i,r} \leq TB_{(v,s)}^{link} \\ &\text{C6.} Eq.\ 7 \sim Eq.\ 13, \end{aligned} \tag{18}$$

where, $f_1(X)$, $f_2(X)$, $f_3(X)$ and $f_4(X)$ indicate the CPU consumption, memory consumption of ESP, the bandwidth consumption of links and the consumption of activating ESP, respectively. $X = (x_1, x_2, x_3, x_4)^T$ stands for the solution space of the problem and $x_i(k \in [1, |X|])$ is a feasible solution in the solution space. Specifically, $x_i = \left\{\alpha_{\theta,s,i}^{r} \mid \theta \in \Theta, s \in N, i \in H_r, r \in R\right\}$ records the complete mapping relationship between VRCs, ESP and services subject to resource and latency constraints. Moreover, we hope to explore the mapping relationship that minimizes the multi-objective optimization problem. Constraint C1 indicates that the CPU resource consumption for serving SRs on an ESP should not exceed the whole CPU resource of the ESP. C2 denotes that the memory resource consumption for serving SRs on an ESP should not exceed the whole memory resource of the ESP. C3 ensures that the transmission of each SR on link (s, q) is limited by the link bandwidth resources. C4 indicates that for each VRC_r, its service latency should satisfy its latency constraint. C5 stands for the link bandwidth resource consumption with respect to the SR should not exceed the bandwidth capacity of link (v, s).

IV. Proposed Approach

Considering the personalized differences between different objectives would degrade the algorithm performance. Therefore, we employ the normalization approach to improve the problem in (18).

A. Objective Normalization

Normalization is a classical methodology to convert a multi-objective problem into a single objective, which can alleviate the impact of computational consumption on the evolutionary direction of the algorithm. Thus, the problem in (18) can be reformulated as:

$$\begin{aligned} & Minimize : F(\mathrm{X}) = \sum_{i}^{m} \lambda_i f_i^{*}(X) \\ & \text{s.t.C1, C2, C3, C4, C5, C6.,} \end{aligned} \tag{19}$$

where $m = 4$ reflects the number of objectives of the multi-objective problem. λ_i is the tunable weight of each objective. $f_1^*(X) = (\mathbb{P} - \mathbb{P}_{min})/(\mathbb{P}_{max} - \mathbb{P}_{min})$, $f_2^*(X) = (\mathbb{M} - \mathbb{M}_{min})/(\mathbb{M}_{max} - \mathbb{M}_{min})$, $f_3^*(X) = (\mathbb{B} - \mathbb{B}_{min})/(\mathbb{B}_{max} - \mathbb{B}_{min})$ and $f_4^*(X) = (\mathbb{S} - \mathbb{S}_{min})/(\mathbb{S}_{max} - \mathbb{S}_{min})$. Furthermore, the minimum value for measuring the different objectives can be determined by searching the entire solution set, i.e., $f_i^{*,min} = \min_{j=1}^{|X|} f_i^*(x_j)$, $x_j \in X$ stands for the solution that minimizes the ith objective. Likewise, $f_i^{*,max} = \max_{1 \le i \le m} f_i^*(x_j)$, where $x_j \in X$ stands for the solution that maximizes the ith objective.

B. MOSP

To address the problem in (19), we designed a two-layers service placement scheme. Intuitively, a complete mapping relationship among the vehicles, the ESPs and microservices are required with the aim of addressing the issue of microservice-oriented service placement. Considering this situation and problem (19), we can conclude that problem (19) is an integer linear programming problem. For integer linear programming problems, when the scale of the problem is large, it is infeasible to achieve the optimal solution of the problem subject to the constraints. Inspired by intelligent heuristic algorithms, we construct a polynomial-time-based heuristic orchestrator with the aim of tackling the microservice-oriented service placement problem.

1) Upper Layer: Based on the VRCs and constraints, we first need to find the mapping relationship between VRCs and ESPs. Therefore, the outer layer of MOSP is mainly responsible for VRC mapping (VRCMA).

$VRC_r \triangleq (ing_r, eg_r, H_r, l_r, D_r^{max})$ indicates that each VRC has a maximum tolerated delay. The VRCs with minimum tolerated delay are selected by sorting all VRCs according to the maximum tolerated delay criterion. If more than one VRC satisfies the minimum tolerance delay, we map each VRC as a whole to an ESP and place a related microservice on ESP. Then, a calculation for each VRC's resource costs is performed, and the VRC with the lowest resource consumption is treated as the target. Finally, for a complete traversal of the remaining VRCs, the current collection of VRCs is copied, R, to the set Rc. Assuming that the target VRC has been mapped to the current ESP as a whole, calculate the total service latency of the target VRC according to the service delay section. If the total service latency of the target VRC satisfies its maximum tolerated latency constraint, check whether the current ESP can host the target VRC. If the current ESP can host the target VRC, the target VRC is mapped to that ESP in its entirety. Then, removed it from R and Rc. Then, select the next target VRC. Otherwise, for each VRC, map the SRs of VRC on ESP, and then the MEC assigns the SRs to the ESPs.

We treat the SRs in the VRC that have been mapped to the ESP as a whole, denoted by Q. Then, the remaining VRCs in Rc are compared with Q, and the VRC whose SRs with the smallest difference from Q are selected as the next target VRC. As a result, we can place fewer microservices in the ESP, which will effectively reduce the PRC of the ESP. If the ESP cannot completely map the SRs in the VRC, we need to provide an empty ESP for the target VRC. Then, the target VRC is removed from Rc, and the target VRC in R is retained for subsequent mapping. Furthermore, if the target VRC violated its maximum tolerated delay, remove the target VRC from Rc and keep the target VRC in R for subsequent mapping. Repeat the above steps until Rc is empty. When the loop ends, check whether the R is empty. If R is empty, it means we have mapped all VRCs on ESPs.

For the mapping results generated by the outer layer, most of the activated ESPs host all different types of microservices. This phenomenon requires that almost all microservices must be placed on each ESP, which will generate a large number of PRCs for the ESPs. However, since the bandwidth resources among ESPs are sufficient and the interaction traffic among them is small, we intend to reduce the PRC of ESPs by increasing the bandwidth resources. Moreover, we can release the redundant services according to the time-varying network load.

2) Lower Layer: The lower layer aims to adjust the service placement based on the VRC mapping results of the outer layer. The lower layer contains SR merging (SRME) and microservice release (MSRE). VRCMA consolidates identical SRs to reduce the duplicated microservices and the number of activated ESPs. Meanwhile, MSRE releases underutilized microservices by periodically checking the already placed microservices.

SRME. Since the activation of ESP not only increases the network load but also causes resource consumption, a minimum number of ESPs should be activated as much as possible. For the sake of brevity, let W indicate the collection of SRs pursuing the same microservice in the same ESP. We define the resource utilization of each ESP as ζ. If $\zeta \le \varsigma$, save that ESP in the list Ts, where ς is the threshold. Conversely, if $\zeta > \varsigma$, save this ESP in the list Th. For each ESP in Ts, the ω of W's in ESPs be calculated, where ω is the sum of traffic flow via all SRs in W. Next, record the state of the current network and start transferring SRs. First, for each ESP saved in Ts, we select the W with the smallest ω and save them in list Lw. In this case, by avoiding the transmission of W with large ω, we can reduce the traffic interaction between ESPs.

Then, a target ESP with adequate resources and capable of hosting the microservice demanded by W is selected for each W from the list Lw. For the Ts list, the ESPs with lower utilization have higher priority. Similarly, a higher priority of ESP indicates that it releases fewer microservices. If the SRs in the VRC satisfy the maximum tolerated latency, we migrate the SRs to the target ESP. If this is not viable, the state of the current network is rolled back to the point where W begins transmission, and the current transmission cycle terminates. After finishing one W transfer cycle, check whether the W has been transferred successfully. If the transfer has been successful, mark the W as processed and remove the corresponding microservice from the associated ESP. If not deleted, the W will simply be marked as processed and will re-enter the transfer process, resulting in duplicate work. For the service latency caused by the previous microservice, if the current service latency is lower than the previous latency, replace the current service placement result with the latest service placement result. If each W is marked as processed, the loop breaks.

The service placement of some ESPs can be removed after merging SRs with the same microservice, which reduces the PRC of the ESP. However, it may lead to resource conflicts on ESP. Therefore, we need to design a resource conflict elimination scheme. For resource conflict elimination, we need to move SRs from resource-violating ESPs to resource-sufficient ESPs one by one until there is no resource conflict problem. Notably, an empty ESP cannot be selected to serve SRs. If the resources do not conflict, record the current ESP-served SRs results. Otherwise, discard and reset the network state. Due to the time-varying network load, almost every ESP suffers from redundant microservices. Therefore, we need to free up unessential microservices.

MSRE. Let $\eta(t)$ reflect the network load, which means the number of SRs waiting to be mapped at the moment t. Thus, the average network load at interval $[t-T, t]$ denoted as $\varphi(t) = (\eta(t) - \eta(t-T))/T$. Thus, the network load variation at interval $[t-T, t]$ is $\gamma(t) = |(\eta(t) - \varphi(t)) - (\eta(t-T) - \varphi(t-T))|/T$. Let τ denote the threshold for network load variability. Under the stable network load state, the network throughput varies only around $\varphi(t)$ and $\gamma(t) \leq \tau$ at time T. At the time T, the network load may deviate from $\varphi(t)$ when the network load fluctuates drastically, resulting in $\gamma(t) > \tau$.

Based on the above description, let $\sigma(t)$ denote the microservice utilization threshold at time t. If $\varphi(t-T) > \varphi(t)$ and $\gamma(t) > \tau$, it signifies the reduction of network load. In this context, the value of $\sigma(t)$ will be adjusted to a larger value σ_l to ensure that more microservices are checked, thereby releasing more redundant microservices. Inversely, $\varphi(t-T) \leq \varphi(t)$ and $\gamma(t) \leq \tau$ demonstrate that the network load is increasing or stabilizing. In this case, the value of $\sigma(t)$ will be adjusted to a smaller value σ_s to assure that fewer microservices are checked, thereby maintaining a stable network state and providing sufficient microservices for SRs.

Assume that the running period of MSRE is T. We define ξ to represent the number of microservices waiting to be released in the ESP. Moreover, we set a higher priority for ESPs with smaller ξ so as to avoid mapping newly arrived VRCs to ESPs with larger ξ. $\sigma(t)$ is updated based on $\varphi(t)$ and $\gamma(t)$. Next, we calculate the utilization of all microservices and add the microservices with lower utilization to the list Ls. When the list Ls is not empty if the type-θ microservice does not serve VRC, release the type-θ microservice.

Algorithm 1 VRCMA

```
Input: $R$, $G = (N \cup K, E)$
Output: Mapping results between SRs and ESP: $MaR_0$
1  Sort all VRCs according to the maximum tolerated delay criterion ;
2  while $R$!=null do
3      Select an VRC with minimum $D_r^{max}$ as the target VRC ;
4      Copy the current sets of VRC $R$ as $Rc$ ;
5      while $Rc$!= null do
6          Calculate the total service latency of target VRC, denoted as TSE ;
7          if TSE< $D_r^{max}$ then
8              if ESP can host target VRC then
9                  Map target VRC on ESP, update $MaR_0$ and remove the target VRC form $R$ and $Rc$ ;
10             else
11                 Map the SRs of VRC on ESP ;
12                 if TSE < $D_r^{max}$ then
13                     Remove the target form $R$ and $Rc$ ;
14                 else
15                     Remove the target form $Rc$
16             if $Rc$!= null then
17                 Treat the SRs in the VRC that have been mapped to the ESP as a whole, denoted by $Q$ ;
18                 Compare the remaining VRCs with $Q$ and select the one with the smallest difference as the next target VRC ;
19             else
20                 Break
21         else
22             Step 11-20
23 return $MaR_0$
```

C. Algorithm Design

As aforementioned, MOSP comprises two layers: the upper layer (VRCMA) and the lower layer (SRME and MSRE).

VRCMA (Algorithm 1): Each newly arrived VRC is the input at each moment, which includes multiple SRs, and the MEC generates the mapping scheme. Considering the maximum tolerated latency of each VRC, we tended to map each VRC as a whole to the ESP, unless the ESP could not provide adequate resources for the VRC. Concomitantly, the SRs of each VRC will be mapped to different microservices on the ESP, respectively. VRCMA can significantly minimize the interaction traffic between ESPs and reduce bandwidth resource consumption.

SRME (Algorithm 2): According to the mapping results generated by VRCMA, each ESP must place the required

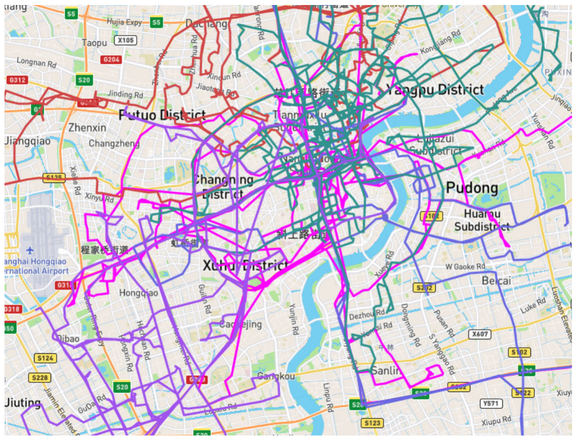


Fig. 3. Trajectories of five taxis randomly selected from Shanghai Taxi dataset.

microservice, which leads to an increase in the number of activated ESPs and the PRC of ESPs. Therefore, we introduce the idea of shared microservices. SRMA tries to merge SRs that require the same microservice together. As a result, the number of activated ESPs and the PRC of ESPs are reduced by placing fewer microservices.

MSRE (Algorithm 3): Considering the utilization of different microservices, MSRE releases placed microservices whose utilization is under the threshold by periodic checks. An adjustment of the threshold value is implemented according to network load changes. If there exists a considerable reduction in network load, the value of $\sigma(t)$ will be adjusted to a larger value σ_l to ensure that more services are checked, thereby releasing more redundant services. Otherwise, the value of $\sigma(t)$ will be adjusted to a smaller value σ_s to assure that fewer microservices are checked, thereby maintaining a stable network state and providing sufficient microservices for VRCs.

Algorithm 2 SRME

```
Input: $MaR_0$, the threshold of ESP utilization, the
       scale of SRs that need the same microservice
       on one ESP: $W$
Output: $MaR_1$
1  Calculate the utilization of each ESP ;
2  if $\zeta \le \varsigma$ then
3      Save those ESPs in the list $Ts$ ;
4  else
5      Save those ESPs in the list $Th$
6  Caculate the total amount of traffic passing throug of
     all $W$'s in those ESPs, expressed as $\omega$ ;
7  while exists unprocessed $W$ in $Ts$ do
8      Select the $W$ with the smallest $\omega$ and save them in
         list $Lw$, $Lw = \{Lw_0, Lw_1, \cdots, Lw_{a-1}\}$ ;
9      for each $W$ in $Lw$ do
10         for each SR in $W$ do
11             Find a target VRC to transfer the SRs ;
12             if satisfy the maximum tolerated latency
                 then
13                 Transfer the SRs to the target ESP ;
14             else
15                 Roll back the state of the network to
                     the $W$ begins transmission
16         if $W$ was successfully transferred then
17             Mark the $W$ as handled and remove the
                 associated microservice from the
                 concerned ESP ;
18             if $TSE < D_r^{max}$ then
19                 Replace the current service placement
                     result with the latest service placement
                     result ;
20             else
21                 Continue
22         else
23             Mark the $W$ as processed
24     Check whether there are resource conflicts ;
25     if exist resource conflicts then
26         Move SRs from resource-violating ESPs to
             resource-sufficient ESPs ;
27         if the SRs are processed successfully then
28             Record the current ESP served SRs results;
29         else
30             Discard and reset the network state
31     else
32         Record the current ESP served SRs results
33 return $MaR_1$
```

V. Performance Evaluation

In this section, we conduct an extensive number of empirical studies to evaluate the superiority of our proposed method. Unless otherwise stated, the following scenarios were used for most experiments. It is assumed for all tasks that the data size ds and the computational amount σ are generated by a probability distribution. To emulate an ultra-dense network scenario, we built a 500 m $\times$ 500 m square meters area with 260 ESPs and 15 CSPs, where CSPs are evenly deployed throughout the scene and the ESPs are evenly distributed around the CSPs. Vehicles can access ESPs and CSPs within 100 m and 300 m. Considering that vehicle movement is affected by many environmental factors, we set the vehicle speed to be generated randomly.

A. Dataset

The Shanghai Taxi trajectory dataset is a 24-hour trajectory of 4000 taxis [24]. The trajectory data is sampled at an interval of 1 min and consists of discrete data points: vehicle ID - time - latitude and longitude - speed - whether it carries passengers or not. Fig. 3 demonstrates the trajectories of five taxis randomly selected from the Shanghai Taxi dataset, with each vehicle's trajectory displayed in a different color.

To measure the performance of the shareable service mechanism, we combine the above dataset, CSP, and ESP deployments to simulate the processing of large service requests from

Algorithm 3 MSRE

```
1 Calculate $\varphi(t)$ and $\gamma(t)$ ;
2 Update $\sigma(t)$ ;
3 Find all the used microservices, expressed as $M$ ;
4 for each service in $m \in M$ do
5     if $\varkappa_m(t) \le \sigma(t)$ then
6         Store microservice $m$ to list $Ls$ ;
7 while $Ls$ != null do
8     if microservices m does not serve VRC then
9         Release microservices $m$ ;
```

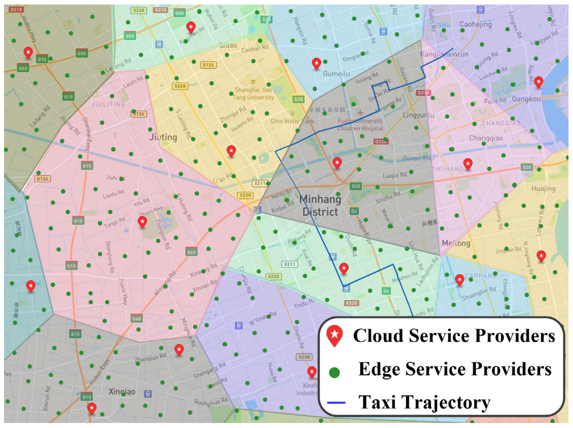


Fig. 4. Experimental scenario diagram.

vehicles. Fig. 4 elaborates on the combination of the dataset, CSP, and ESP. The red dots stand for CSPs, green dots denote ESPs, blue lines represent vehicle trajectories, and different color areas indicate the coverage of CSPs. The surrounding environment (e.g., ESP density and CSP distribution) and trajectory of each vehicle are different.

B. Experiment Setup

We define that each CSP is equipped with 20 microservices and the total CPU frequency is 20 GHz. The CPU frequency of each ESP is 4GHz, and the available memory is 2048MB. The available bandwidth of the link between ESPs is 1200Mbps, and the available bandwidth between the vehicle and the ESPs is 800Mbps. To improve the reliability of the latency, we adopted the Monte Carlo method and performed 500 simulations in 500 scenarios to optimize the service coordination of one vehicle. Since the vehicle trajectory data set is collected at an interval of 1 min, we use the Bessel curve approach to refine the trajectory data. All experimental results are the average of the results of multiple runs. We consider that the packet size of SR in each VRC generated by the vehicle follows a Poisson distribution with a mean value of 1 Mbits/s, and the computational amount is uniformly distributed in [2, 0.5] Gigacycles. Likewise, each service processing result is $pt_i/10$, and the maximum tolerated delay of each SR varies in the interval [50, 100] ms.

To simulate the dynamic changes of the network load and ensure the consistency of the comparison experimental setup,

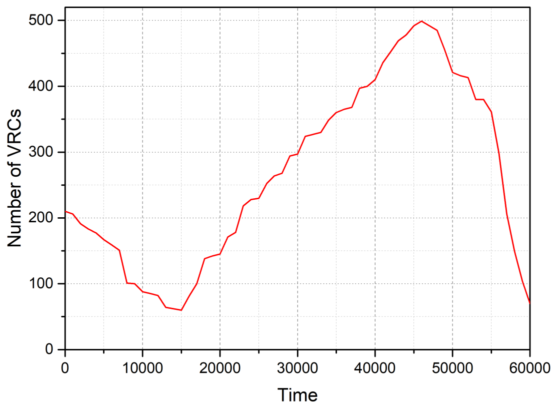


Fig. 5. The dynamic change of number of VRCs.

we set only one SR generated by the vehicle, and the total number of VRCs in the network at each moment is known. The TOTEM project [25] can dynamically track the network traffic changes of transit networks. According to that, we simulated the change in the number of VRCs in a certain time unit by compressing the observation time, as shown in Fig. 5. The number of SRs contained in each VRC ranges from [5, 12]. For each service request, the CPU consumption required to place service on the ESP is 60 MHz, and the memory consumption is 40 MB. The resource consumption for serving SR obeys a uniform distribution, with CPU resource consumption of [50, 200] MHz, memory resource consumption of [20, 100] MB, and bandwidth consumption of [10, 30] Mbps. The propagation speed of the signal in the medium between the vehicle and ESP is $S^{med}_{(v,s)} = 200m/\mu s$, and the propagation speed of the signal in the medium between ESPs is $S^{med}_{(s,q)} = 500m/\mu s$. The communication delay between any ESP and the CSP is 50ms. Considering that all network resources with equal importance, we set the tunable weight in problem (19) to $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.25$.

The lifetime LT of each VRC is an exponential distribution with a mean value of 5000. The occupied resources are released when the lifetime of the VRC expires. We set the network load threshold to 700Mbps. The utilization of each microservice determines the number of services checked and released. If there exists a considerable reduction in network load, the value of $\sigma(t)$ will be adjusted to a larger value, i.e. $\sigma_l = 50\%$. Conversely, the value of $\sigma(t)$ will be adjusted to a smaller value, i.e. $\sigma_s = 20\%$. In addition, we set the operation period of VRC to 300 ms. Table.1 lists the main parameter settings in this paper.

C. Contrastive Algorithms

To benchmark the proposed solution, we compared three contrastive algorithms, which are described as follows:

1) Random-Based: Any service request in the VRC is randomly assigned to an ESP.

2) Nearest ESP First: In this algorithm, the vehicle always prefers the closest ESP. If the nearest ESP lacks a certain microservice, the ESP requests that service image file from the CSP and places that service.

TABLE I
SCENARIO PARAMETERS

Parameters	Description	Value
N	ESP scale in the network	260
W	CSP scale in the network	15
R	Scale of VRC per time units	Fig. 5
H_r	Scale of SRs of per VRC_r	[5, 12]
LT	Lifetime of each VRC	$X \sim E(1/5000)$
$TB_{(v,s)}^{link}$	Link capacity on link (v,s)	800Mbps
$TB_{(s,q)}^{link}$	Link capacity on link (s,q)	1200Mbps
D_i^*	Size of processing result	0.1MB
$S_{(v,s)}^{med}$	Propagation speed on link (v,s)	$200m/\mu s$
$S_{(s,q)}^{med}$	Propagation speed on link (s,q)	$500m/\mu s$
$cpu_{s,i}^r$	CPU cost required by $SR_{\theta,i,r}$ in VRC_r	[50, 200]MHz
$mem_{s,i}^r$	Memory cost required by $SR_{\theta,i,r}$ in VRC_r	[20, 100]MB
pc_θ^{cpu}	CPU cost when placing a type-θ microservice on ESP s	60MHz
pc_θ^{mem}	Memory cost when placing a type-θ microservice on ESP s	40MB
$b_{(i,j)}^r$	Bandwidth cost between two SRs in VRC_r	[10, 30]Mbps
$b_{(v,s)}^{i,r}$	Bandwidth cost between vehicle and ESP	[10, 30]Mbps
TC_s^{cpu}	CPU capacity of ESP s	4000MHz
TM_s^{mem}	Memory capacity of ESP s	2048MB
T	Operation cycle of MSRE	300ms
τ	Threshold of network load	700Mbps
ς	Threshold of ESP utilization	20%
σ_l	Threshold of microservice utilization	50%
σ_s	Threshold of microservice utilization	20%
pt_i	Packet size of $SR_{\theta,i,r}$	Poisson distribution with a mean of 1Mbits/s
ca_i	Computational amount	[0.5, 2]Gigacycle
$D_{i,r}^{max}$	Maximum tolerable delay of $SR_{\theta,i,r}$	[50, 100]ms

3) Matching-Based Greedy: The VRC and the network are expressed as two weighted graphs: the physical network and service forward graphs. Then, the adjacent matrices of the two graphs are obtained and matched using bipartite matching.

D. Results

This section shows the experimental results.

1) Latency Guaranteed: To evaluate the MOSP performance, we verified the number of VRCs that satisfy the latency requirements at different moments. As Fig. 5 shows the variation of the number of VRCs in the network within the time unit. As time advances, the number of VRCs first decreases, then increases and finally decreases. This means that the network load is fluctuating. When the network load is small (a small number of VRCs), the network resources are sufficient to ensure that each SRs is processed with a limited delay constraint. When the network load is high (a large number of VRCs), there is resource competition among SRs so that some SRs are not necessarily processed on time. As illustrated in Fig. 6, when the network load is high (a large number of VRCs), the MOSP algorithm is capable of processing a high number of VRCs while satisfying the VRC latency requirement. That is because MOSP can periodically check and release redundant services, which saves ESP resources.

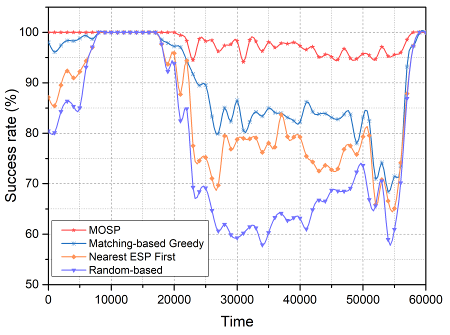


Fig. 6. Effectiveness of latency guarantee.

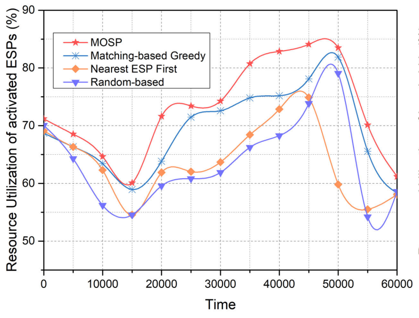
Furthermore, MOSP thoroughly focuses on the latency of service and the resource consumption of ESPs and bandwidth and dynamically adjusts resource utilization to avoid network resource bottlenecks.

2) Resource Utilization: We estimate the resource utilization of different algorithms for both ESP resources and network bandwidth resources. Resource utilization is defined as the ratio of occupied resources to total available resources.

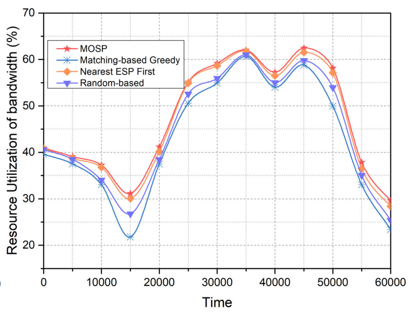
Fig. 7(a) demonstrates the resource utilization of ESP for different algorithms. The increase in the number of vehicle-generated VRCs (Fig. 5) leads to an increase in the network load which results in more ESP resources to complete the VRCs. MOSP seeks to optimize overall resources with the least number of ESPs throughout the optimization process. Moreover, MOSP adopts a microservice-oriented service mechanism to minimize the idle resources of ESPs and maximize the utilization of ESP resources. Unlike MOSP, the Matching-based Greedy tends to implement load balancing across the network. When the network load is higher, the number of ESPs required to complete vehicle requests increases, and the number of network links also increases. In this context, the ESP resource utilization of Matching-based Greedy is relatively low. Generally, MOSP maintains the ESP utilization of around 76% when the network load is high. When the network load is low, the ESP resource utilization also stays around 65%.

The variation of network bandwidth resource utilization with the network load of different algorithms is shown in Fig. 7(b). As the network load changes, the network bandwidth resource utilization of different algorithms fluctuates. Overall, the network bandwidth resources increase along with the network load. Since Matching-based Greedy occupies more links, it has the lowest bandwidth utilization. Due to the Nearest ESP First selecting the nearest ESP in each VRC mapping process, it consumes the least bandwidth resources and has lower link utilization. The bandwidth resource utilization of MOSP and Nearest ESP First are very close. The bandwidth resource utilization of MOSP stays around 48% and the bandwidth resource utilization of Nearest ESP First stays around 48.7%.

3) Impact of Time-Varying Network Load: The resource consumption of ES consists of placing service and serving VRC. Since the resource consumption of placing a service

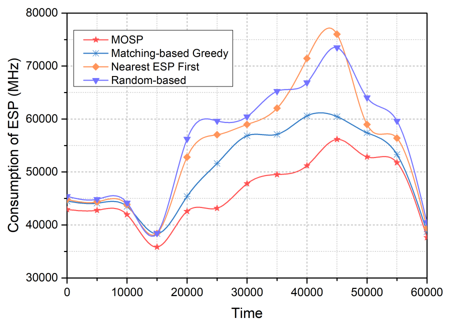


(a). Resource utilization of activated ESP

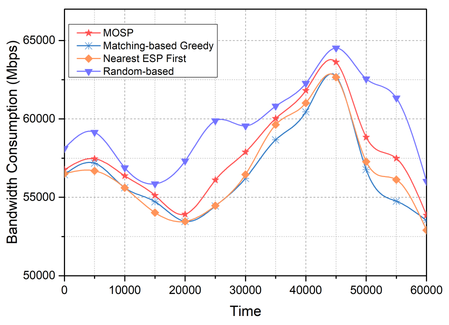


(b). Resource utilization of bandwidth

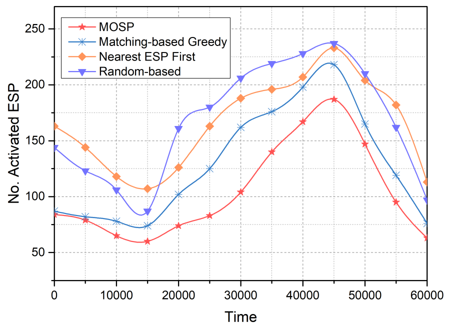
Fig. 7. Effectiveness of resource utilization.



(a). Resource consumption of ESP



(b). Resource consumption of bandwidth



(c). Number of activated ESP

Fig. 8. Impact of time-varying network load.

on each ESP is identical, we consider that the ESP resource consumption mainly relies on the resource consumption for serving SR.

Fig. 8(a) illustrates that MOSP outperforms the contrastive algorithms. This phenomenon is because MOSP can combine the same SRs during the adjustment phase and take advantage of the shareable properties of the services to complete the VRC. As a result, the number of services can be effectively reduced during the service placement process, thereby reducing ESP resource consumption. In contrast, Random-based and Nearest ESP First ignore service shareability and pursue only service completion and latency. Moreover, MOSP can periodically check and releases redundant services based on the time-varying network load to reduce resource consumption. Unlike MOSP, Matching-based Greedy ignores the release of redundant services, resulting in performance degradation.

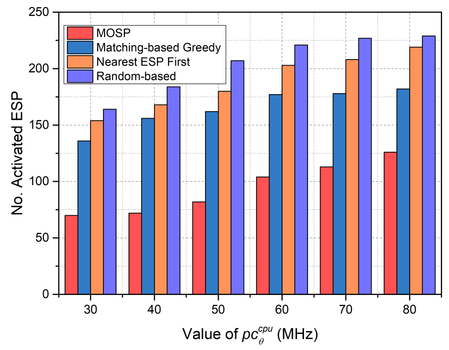
Fig. 8(b) illustrates that the Nearest ESP First and Matching-based Greedy algorithms achieve the best performance for network bandwidth consumption. Since the Nearest ESP First selects the nearest ESPs to process the VRC, the network bandwidth consumption is minimal. The Matching-based Greedy adopts the adjacency matrix of two weighted graphs to map the VRC in the physical network. In each mapping, Matching-based Greedy can map most of the service requests in the VRC to one ESP, so the information interactions in the VRC are restricted to one ESP, reducing network bandwidth resource consumption. The proposed MOSP considers the network bandwidth consumption, ESP resource consumption, and request latency.

Fig. 8(c) demonstrates the ESP usage for different algorithms. We conclude that MOSP performs best. As depicted in Fig. 8(a), MOSP adopts service shareability to minimize the number of services placed in the network, thus reducing ESP resource consumption. Hence, MOSP can accomplish different VRCs with less number of ESPs. In addition, MOSP can adaptively release redundant services by dynamically capturing the time-varying network load, thereby shutting down more idle ESPs. Unlike MOSP, all four contrastive algorithms ignore the time-varying network load. Therefore, they require more ESPs to piggyback on redundant services.

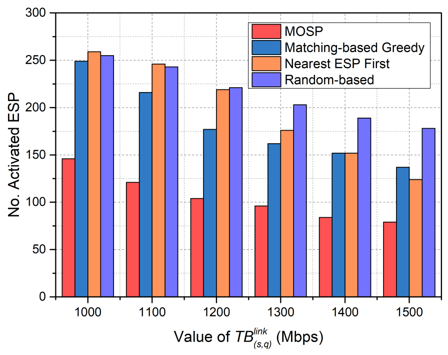
4) Impact of Placing a Service Resource Consumption and Link Capacity: To investigate the performance impact of link capacity and the primary resource overhead of ESP during service placement, we set the total number of VRCs in the network to 700.

The primary resource consumption of ESP is generated by placing a service on the ESP. Considering the non-differentiation of services, we set the resource consumption for placing services to be identical. Fig. 9(a) illustrates that the more resources consumed to place service, the fewer resources are available for the ESP to serve VRC, so more ESPs are needed to accomplish VRC. Additionally, MOSP can periodically check and release redundant services, which helps improve ESP resource utilization. Therefore, MOSP performs better as the consumption of service placement resources increases.

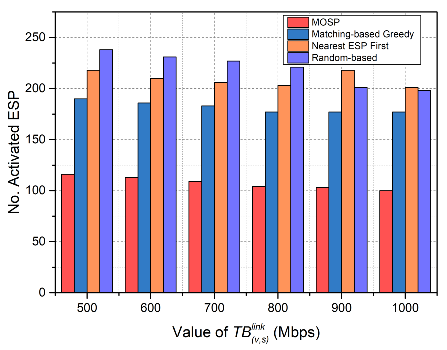
Link capacity is also an essential factor affecting the performance of the algorithm. Fig. 9(b) and Fig. 9(c) show that as link capacity increases, more bandwidth resources become



(a). Number of activated ESP with different PRC

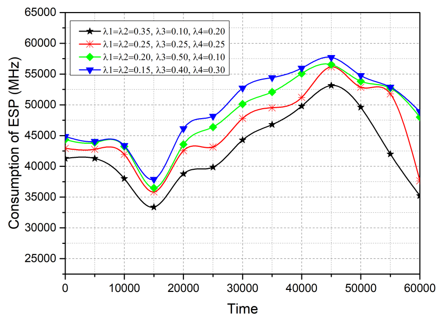


(b). Number of activated ESP with different $TB_{(s,q)}^{link}$

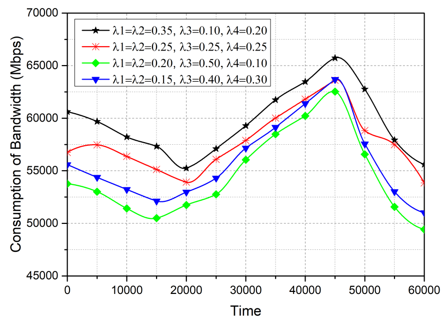


(c). Number of activated ESP with different $TB_{(v,s)}^{link}$

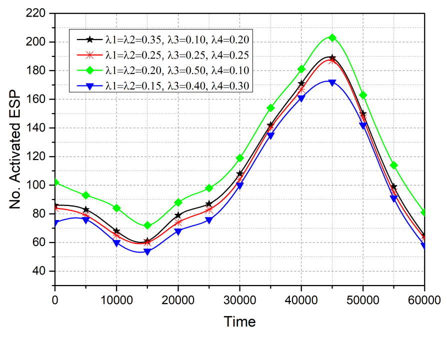
Fig. 9. Impact of placing a service resource consumption and link capacity.



(a). Resource consumption of ESP

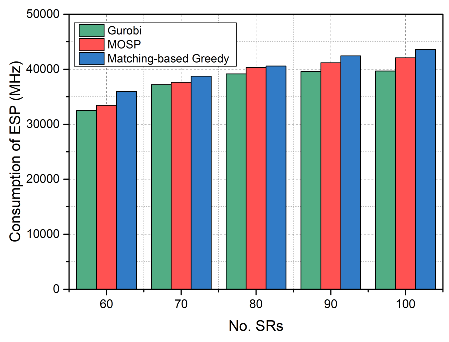


(b). Resource consumption of bandwidth

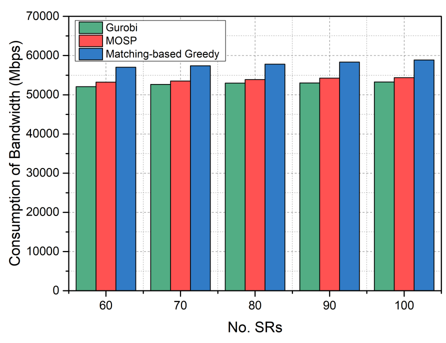


(c).Number of activated ESP

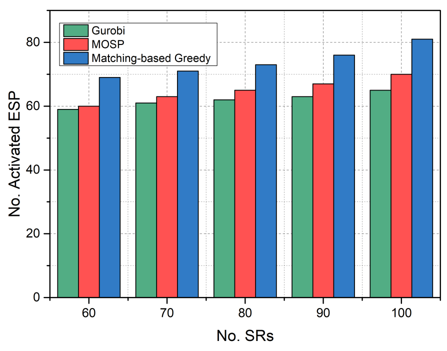
Fig. 10. Impact of tunable weights.



(a). Resource consumption of ESP



(b). Resource consumption of bandwidth



(c). Number of activated ESP

Fig. 11. Comparison of optimal solutions.

available to the network, which helps MOSP to adjust the service placement more rationally. As a result, MOSP can accomplish VRCs with fewer ESPs. However, with fewer bandwidth resources, MOSP still outperforms the contrastive algorithms. MOSP prefers to map the entire VRC to a single ESP during the mapping phase, allowing for less interactive traffic between ESPs. Therefore, MOSP is capable of avoiding bandwidth bottlenecks in the presence of time-varying network loads. Moreover, we can observe that the Matching-based Greedy algorithm is sensitive to link capacity. With the increase in link capacity, the quantity of ESPs employed by Matching-based Greedy diminishes significantly.

5) Impact of Tunable Weights: In this paper, we devote ourselves to investigating the latency of service, the resource cost of ESPs (CPU and memory), and the resource cost of bandwidth, while introducing tunable weights to optimize the MOSP performance. The performance of MOSP varies with the setting of different tunable weights. Considering that CPU and memory are directly related to the ESP, we set their weights to be equal.

From Fig. 10, we conclude that the higher the tunable weight of the optimization objective, the lower the resource consumption. In general, higher tunable weights indicate a preference for optimizing the objective, resulting in lower resource consumption for that objective. As illustrated in Fig. 10(a), the larger λ_1 and λ_2 are, the lower the ESP resource consumption is. As depicted in Fig. 10(b), the larger the λ_3, the lower the network bandwidth resource consumption.

As shown in Fig. 10(c), the number of activated ESPs increases as the value of λ_4 increases. This phenomenon indicates that we need to pay more attention to optimizing the activated ESP resources. The number of activated ESPs determines how many ESP resources are available. In order to optimize the number of activated ESPs, it is essential to increase the resource utilization of each ESP.

6) Comparison of Optimal Solutions: To further validate the MOSP performance, we verify the performance of the algorithm in a small-scale setting. We can derive the optimal solution with Gurobi. Fig. 11 shows the resource consumption of different algorithms in a small-scale environment. Compared to the resource consumption of network bandwidth, the MOSP algorithm is close to the optimal solution in terms of the resource overhead of ESPs and bandwidth, and the number of activated ESPs.

VI. Summary and Future Work

In this paper, we investigate the microservice-oriented service placement mechanism for resource optimization and latency guarantees in MEC-enabled IoV. We considered the time-varying network load and service shareability to reduce resource consumption and guarantee the service latency generated by the service placement. We formulate the service placement problem as an integer linear programming problem and propose a sustainable service mechanism with two phases. Experiments were conducted on real datasets, and the results indicated that the MOSP algorithm outperformed the contrastive algorithms with respect to resource consumption and service latency. Moreover, to further validate the MOSP performance, we consider a small-scale VRC environment and demonstrate that the MOSP can achieve near-optimal solutions. However, there are still many challenges to be addressed. For example, considering service replacement cost during service placement; how to efficiently guarantee service request processing in the event of service stoppage or corruption; generalizing the utility function to meet specific requirements; focusing on service scheduling to optimize the latency of service requests; and investigating microservice orchestration and migration issues.

References

[1] L. Wang, J. Gui, X. Deng, F. Zeng, and Z. Kuang, "Routing algorithm based on vehicle position analysis for Internet of Vehicle," *IEEE Internet Things J.*, vol. 7, no. 12, pp. 11701–11712, Dec. 2020.

[2] C. Chen, C. Wang, B. Liu, C. He, L. Cong, and S. Wan, "Edge intelligence empowered vehicle detection and image segmentation for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 5, 2023, doi: 10.1109/TITS.2022.3232153.

[3] S. S. Gill et al., "AI for next generation computing: Emerging trends and future directions," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100514.

[4] S. Liu, J. Yu, X. Deng, and S. Wan, "FedCPF: An efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1616–1629, Feb. 2022.

[5] Y. Liu et al., "Joint communication and computation resource scheduling of a UAV-assisted mobile edge computing system for platooning vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8435–8450, Jul. 2022.

[6] Y. Chen et al., "LOCUS: User-perceived delay-aware service placement and user allocation in MEC environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 7, pp. 1581–1592, Jul. 2022.

[7] Y. Mao, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[8] C. Roy, R. Saha, S. Misra, and K. Dev, "Micro-safe: Microservices- and deep learning-based safety-as-a-service architecture for 6G-enabled intelligent transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9765–9774, Jul. 2022.

[9] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual understanding of microservice architecture: Current and future directions," *ACM SIGAPP Appl. Comput. Rev.*, vol. 17, no. 4, pp. 29–45, Jan. 2018.

[10] S. Wang, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[11] X. Deng, Z. Sun, D. Li, J. Luo, and S. Wan, "User-centric computation offloading for edge computing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12559–12568, Aug. 2021.

[12] C. Chen, Y. Zeng, H. Li, Y. Liu, and S. Wan, "A multihop task offloading decision model in MEC-enabled Internet of Vehicles," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3215–3230, Feb. 2023.

[13] B. Cao et al., "Large-scale many-objective deployment optimization of edge servers," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3841–3849, Jun. 2021.

[14] J. Ni, K. Zhang, and A. V. Vasilakos, "Security and privacy for mobile edge caching: Challenges and solutions," *IEEE Wireless Commun.*, vol. 28, no. 3, pp. 77–83, Jun. 2021.

[15] B. V. Natesha and R. M. R. Guddeti, "Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," *J. Netw. Comput. Appl.*, vol. 178, Mar. 2021, Art. no. 102972.

[16] H. O. Hassan and S. M. A. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments," *IET Commun.*, vol. 2020, pp. 1–13, Jan. 2020.

[17] M. Sriraghavendra, P. Chawla, H. Wu, S. S. Gill, and R. Buyya, "DoSP: A deadline-aware dynamic service placement algorithm for workflow-oriented IoT applications in fog-cloud computing environments," in *Energy Conservation Solutions for Fog-Edge Computing Paradigms* (Lecture Notes on Data Engineering and Communications Technologies). Singapore: Springer, 2022, pp. 21–47.

[18] W. S. Kim and S. H. Chung, "User-participatory fog computing architecture and its management schemes for improving feasibility," *IEEE Access*, vol. 6, pp. 20262–20278, 2018.

[19] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.

[20] T. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5595–5608, Nov. 2021.

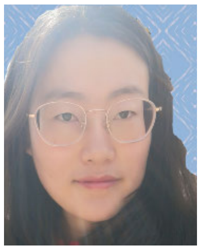
[21] J. Pei, P. Hong, M. Pan, J. Liu, and J. Zhou, "Optimal VNF placement via deep reinforcement learning in SDN/NFV-enabled networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 263–278, Feb. 2020.

[22] P. Han and Y. L. Liu Guo, "Interference-aware online multi-component service placement in edge cloud networks and its AI application," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10557–10572, Jul. 2021.

[23] F. A. Salaht, F. Desprez, and A. Lebre, "An overview of service placement problem in fog and edge computing," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–35, May 2021.

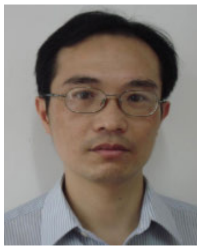
[24] *Smart City Research Group*. [Online]. Available: https://github.com/chilai1996/Shanghai-Taxi-Data

[25] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.



Leilei Wang received the M.S. degree from Central South University in 2019, where she is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. Her research interests include opportunistic networks, vehicular networks, the Internet of Vehicle, edge computing, and ITS.

Xiaoheng Deng (Senior Member, IEEE) received the Ph.D. degree in computer science from Central South University, Changsha, Hunan, China, in 2005. Since 2006, he has been an Associate Professor and then a Full Professor with the Department of Electrical and Communication Engineering, Central South University, where he is currently with the School of Computer Science and Engineering, Research Institute of Shenzhen. He is also a Joint Researcher with the Shenzhen Research Institute, Central South University. His research interests include edge computing, the Internet of Things, online social network analysis, data mining, and pattern recognition. He is also a Senior Member of CCF, a member of CCF Pervasive Computing Council, and a member of ACM. He has been the Chair of CCF YOCSEF CHANGSHA from 2009 to 2010.



Jinsong Gui (Member, IEEE) is currently a Professor with the School of Computer Science and Engineering, Central South University, China. His research interests include the general area of distributed systems and related fields, such as wireless network topology control, performance evaluation, and network security.

Xuechen Chen (Member, IEEE) received the B.S.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2007, and the Ph.D. degree in electrical engineering from the University of California at Riverside in 2012. She has been with the Bell Laboratories Research Center as a Research Scientist focusing on multimedia transmission over next-generation wireless networks. She is currently an Associate Professor with the Department of Computer Science, Central South University, Changsha, Hunan, China. Before she attended Central South University, she was an Assistant Professor with the School of Electronics and Information Technology, Sun Yat-sen University. Her research interests include distributed computing, joint source-channel coding, and distributed source coding, especially in delay-constrained applications, distributed compressed sensing, swarm intelligence in optimization and feature selection, and localization by wireless sensor networks.

Shaohua Wan (Senior Member, IEEE) received the Ph.D. degree from the School of Computer, Wuhan University, in 2010. Since 2015, he has been a Post-Doctoral Researcher with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology. From 2016 to 2017, he was a Visiting Professor with the Department of Electrical and Computer Engineering, Technical University of Munich, Germany. He is currently an Associate Professor with the School of Information and Safety Engineering, Zhongnan University of Economics and Law. He is the author of over 110 peer-reviewed SCI indexed papers. His main research interests include deep learning for the Internet of Things and edge computing. He had served as a lead Guest Editor for IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *ACM Transactions on Multimedia Computing Communication*, *Journal of Systems Architecture, Computer Communications*, *Pattern Recognition Letter*, *Multimedia Tools and Applications*, *Image and Vision Computing*, and *Computers and Electrical Engineering*.