Cost-Effective Two-Stage Network Slicing for Edge-Cloud Orchestrated Vehicular Networks

Wen Wu[‡], Kaige Qu[⋆], Peng Yang[*], Ning Zhang[†], Xuemin (Sherman) Shen[⋆], and Weihua Zhuang[⋆]
Frontier Research Center, Peng Cheng Laboratory, Shenzhen, China[‡]
Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada[⋆]
School of Electronic Information and Communications, Huazhong University of Science and Technology, China[*]
Department of Electrical and Computer Engineering, University of Windsor, Windsor, Canada[†]
Email: wuw02@pcl.ac.cn[‡], {k2qu, sshen, wzhuang}@uwaterloo.ca[⋆],
yangpeng@hust.edu.cn[*], and ning.zhang@uwindsor.ca[†]

***Abstract*—In this paper, we study a network slicing problem for edge-cloud orchestrated vehicular networks, in which the edge and cloud servers are orchestrated to process computation tasks for reducing network slicing cost while satisfying the quality of service requirements. We propose a two-stage network slicing framework, which consists of 1) *network planning* stage in the large timescale to perform slice deployment, edge resource provisioning, and cloud resource provisioning, and 2) *network operation* stage in the small timescale to perform resource allocation and task dispatching. Particularly, we formulate the network slicing problem as a two-timescale stochastic optimization problem to minimize the network slicing cost. Since the problem is NP-hard, we develop a Two timescAle netWork Slicing (TAWS) algorithm by collaboratively integrating reinforcement learning (RL) and optimization methods, which can jointly make network planning and operation decisions. Specifically, by leveraging the timescale separation property of decisions, we decouple the problem into a large-timescale network planning subproblem and a small-timescale network operation subproblem. The former is solved by an RL method, and the latter is solved by an optimization method. Simulation results based on real-world vehicle traffic traces show that the TAWS can effectively reduce the network slicing cost as compared to benchmark schemes.**

I. Introduction

To make autonomous driving from a mere vision to reality, future vehicular networks are required to support various Internet of vehicles (IoV) services, such as object detection, in-vehicle infotainment, and safety message dissemination [1]. Those IoV services have diversified quality of service (QoS) requirements in terms of delay, throughput, reliability, etc. Emerging network slicing is deemed as a *de-facto* solution to support diversified IoV services in vehicular networks. Its basic idea is to construct multiple isolated logical sub-networks (i.e., slices) for different services on top of the physical network, thereby facilitating flexible, agile, and cost-effective service provision. Starting from the fifth-generation (5G) era, standardization efforts from the 3rd generation partnership project (3GPP) body, e.g., Releases 15-17 [2]–[4], and proof-of-concept systems, e.g., Orion [5], have fuelled the widespread of network slicing. In the coming 6G era, advanced network slicing techniques are expected to play an increasingly important role [6], [7].

In the literature, significant research efforts have been devoted to network slicing. Ye *et al.* investigated a radio spectrum resource slicing problem, in which radio spectrum is sliced between macro base stations (MBSs) and small BSs (SBSs) [8]. To achieve efficient resource allocation, a deep learning-based algorithm was proposed to jointly allocate radio spectrum and transmit power in a slicing-based network [9]. The previous work in [10] considered the resource provisioning problem and proposed a constrained learning algorithm to solve it. However, this work differs from the existing works in several important aspects. Firstly, the existing works focus on utilizing resources on the network edge, low-cost cloud resources are yet to be considered. As a remedy, a certain amount of computation tasks processed at congested BSs can be dispatched to the remote cloud, i.e., *task dispatching*, such that network slicing cost can be reduced. Secondly, network slicing includes two stages: 1) *network planning* stage to provision network resources for slices in the large timescale, and 2) *network operation* stage to allocate the reserved resources to end users in the small timescale [3], [11]. The existing works mainly decouple network slicing into two independent stages, while the interaction between these two stages is seldom considered. Hence, designing a cost-effective network slicing scheme should take cloud resources and such interaction into consideration.

Optimizing network slicing performance in dynamic vehicular networks faces the following *challenges*. Firstly, network planning and operation decisions are *nested*. Large-timescale network planning decisions (e.g., resource reservation), will condition small-timescale network operation decisions (e.g., resource allocation). Meanwhile, the performance achieved in the network operation stage will also affect the decision-making in the network planning stage, which is difficult to be solved by conventional optimization methods. Secondly, since vehicle traffic density varies temporal-spatially, network planning decisions need to be made to optimize long-term performance in the slice lifecycle while accommodating such network *dynamics*. Deep reinforcement learning (RL) is considered as a plausible solution for long-term stochastic optimization.

In this paper, we *first* propose a cost-effective two-stage

network slicing framework for edge-cloud orchestrated vehicular networks, by considering nested network planning and operation stages and effectively leveraging cloud resources. We then apply a network slicing cost model that accounts for slice deployment, resource provision, slice configuration adjustment, and QoS satisfaction. Based on the model, we formulate the network slicing problem as a two-timescale stochastic optimization problem to minimize the network slicing cost. *Second*, to solve the problem, we develop a learning-based algorithm, named Two timescAle netWork Slicing (TAWS). The TAWS exploits the timescale separation structure of decision variables and decouples the problem into two subproblems in different timescales. Regarding the large-timescale network planning subproblem, an RL algorithm is designed to minimize network slicing cost via optimizing slice deployment, edge resource provisioning, and cloud resource provisioning. Regarding the small-timescale network operation subproblem, an optimization algorithm is designed to minimize average service delay via optimizing resource allocation and task dispatching. In addition, the achieved service delay in the network operation stage is incorporated into the reward of the RL-based network planning algorithm, thereby capturing interaction between two stages and enabling *closed-loop* network control. Simulation results on real-world vehicle traces demonstrate that the proposed algorithm outperforms existing benchmarks in terms of reducing network slicing cost.

The remainder of this paper is organized as follows. The system model and problem formulation are presented in Sections II and III, respectively. Section IV describes the proposed TAWS algorithm. Simulation results are given in Section V, along with the conclusion in Section VI.

II. System Model

A. Network Model

As shown in Fig. 1, the network slicing framework consists of several components.

1) Physical network: A two-tier cellular network is deployed for serving on-road vehicles. The set of BSs is denoted by $\mathcal{M}$, including the set of MBSs denoted by $\mathcal{M}_m$ and the set of SBSs denoted by $\mathcal{M}_s$, i.e., $\mathcal{M} = \mathcal{M}_m \cup \mathcal{M}_s$. Each BS has a circular coverage and is equipped with an edge server. In the considered scenario, vehicles driving on the road generate computation tasks over time, which are offloaded to roadside BSs. Those tasks can be either processed at edge servers or dispatched to the remote cloud server via backbone networks. Once completed, computation results are sent back to vehicles.

2) Network slice: Multiple network slices are constructed on top of the physical vehicular network. We consider K delay-sensitive services with differentiated delay requirements, denoted by set $\mathcal{K}$. Let $\theta_k, \forall k \in \mathcal{K}$ denote the tolerable delay of service k. For example, the tolerable delay of objective detection is 100 *ms* [12], whereas the tolerable delay of in-vehicle infotainment can be several hundreds of milliseconds.

3) Network controller: A hierarchical network control architecture is adopted, including an upper-layer software defined networking (SDN) controller that connects to all BSs, and lower-layer local network controllers located at BSs. Those controllers are in charge of network information collection and slicing decision making.

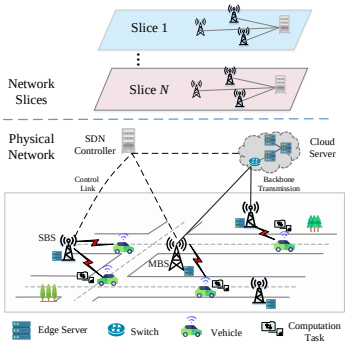


Fig. 1. Network slicing for edge-cloud orchestrated vehicular networks.

B. Two-Stage Network Slicing Framework

we present a two-stage network slicing framework for the considered network. Firstly, a network planning stage operates in the large timescale (referred to as planning windows) to reserve resources at specific network nodes for the constructed slices. The duration of each planning window is denoted by T_p. At each planning window, the SDN controller collects the average vehicle traffic density information in the considered area, based on which planning decisions are made. Secondly, the network operation stage operates in the small timescale (referred to as a operation slots) to dynamically allocate the reserved resources to according to real-time vehicles' service requests and network conditions. The duration of each operation slot is denoted by T_o. A planning window includes multiple operation slots, i.e., $T_p/T_o \in \mathbb{Z}^+$. At each operation slot, local network controller at each BS collects real-time service requests and channel conditions of its associated vehicles, based on which operation decisions are made. Decisions in two stages are detailed as follows.

1) Network Planning Decision Structure: The planning window is indexed by $w \in \mathcal{W} = \{1, 2, ..., W\}$, and planning decisions in each planning window w are as follows.

Slice deployment decision, denoted by $\mathbf{o}^w \in \mathbb{R}^{M_s \times 1}$. Each element is a binary variable, i.e.,

$$o_m^w \in \{0, 1\}, m \in \mathcal{M}_s. \tag{1}$$

If SBS m is activated for slice deployment, we have $o_m^w = 1$; otherwise, $o_m^w = 0$. When service demands are low, deploying slices at a selective subset of BSs can reduce network slicing cost as compared to deploying slices at all BSs while guaranteeing slices' service level agreements (SLAs). This is because running network slicing requires resource virtualization, which incurs network operating costs. For service continuity consideration, we assume that MBSs that cover the entire area are always activated. Note that only when a BS is activated for slice deployment, edge resources can be provisioned.

Edge resource provisioning decision, including radio spectrum and computing resource provisioning at all BSs for all slices, denoted by $\mathbf{B}^w \in \mathbb{R}^{K \times M}$ and $\mathbf{C}^w \in \mathbb{R}^{K \times M}$, respectively. The corresponding elements

$$\{b_{k,m}^w, c_{k,m}^w\} \in \mathbb{Z}^+, \forall k \in \mathcal{K}, m \in \mathcal{M}, \tag{2}$$

represent the number of subcarriers and edge virtual machine (VM) instances provisioned for slice k at BS m, where $\mathbb{Z}^+$ denotes the set of positive integers.[1] The bandwidth of a subcarrier is β, and the computing capability of an edge VM is F_e. Due to the limitation of edge resources, the following capacity constraints are imposed:

$$o_m^w \sum_{k\in\mathcal{K}} b_{k,m}^w \le B_m, o_m^w \sum_{k\in\mathcal{K}} c_{k,m}^w \le C_m, \forall m \in \mathcal{M}, \quad (3)$$

where B_m and C_m represent the total number of subcarriers and the total number of VM instances at BS m, respectively.

Cloud resource provisioning decision, denoted by $\mathbf{h}^w \in \mathbb{R}^{K\times 1}$. Each element

$$h_k^w \in \mathbb{Z}^+, \forall k \in \mathcal{K} \quad (4)$$

denotes the number of cloud VM instances reserved for slice k. The computing capability of a cloud VM is denoted by F_c.

2) Network Operation Decision Structure: Let $t \in \mathcal{T} = \{1, 2, ..., T\}$ denote the index of operation slots in a planning window. At each operation slot t, the following decisions are determined for each slice k.

Radio spectrum allocation decision, denoted by $\mathbf{y}_k^t \in \mathbb{R}^{N^t\times 1}$. The reserved radio spectrum at each BS is allocated to active vehicles within BS's coverage for task offloading. Due to vehicle mobility, the number of vehicles varies across time. Let $\mathcal{N}^t$ denote the set of active vehicles in operation slot t, and $N^t = |\mathcal{N}^t|$. For simplicity, each vehicle associates to the nearest BS. Let $\mathcal{N}_m^t$ denote the set of active vehicles associated to BS m at operation slot t, and $y_{k,n}^t \in \mathbb{R}^+$ represents the fraction of radio spectrum allocated to vehicle n. The total amount of the allocated bandwidth should not exceed the reserved number of subcarriers at each BS, yielding

$$\sum_{n\in\mathcal{N}_m^t} y_{k,n}^t \le b_{k,m}^w, \forall m \in \mathcal{M}^w, \quad (5)$$

where $\mathcal{M}^w$ denotes the set of activated BSs in window w.

Task dispatching decision, denoted by $\mathbf{x}_k^t \in \mathbb{Z}^{M^w\times 1}$. The BS receives computation tasks uploaded from its associated vehicles. The task arrivals of vehicles follow arbitrary stochastic processes. Let $a_{k,n}^t$ denote the number of generated tasks of vehicle n in operation slot t, and the aggregated computation workload at BS m is given by $A_{k,m}^t = \sum_{n\in\mathcal{N}_m^t} a_{k,n}^t$. Processing all tasks at BSs with limited edge computing resources may incur prohibitive high queuing delay, and hence a portion of computation tasks can be dispatched to the remote cloud via backbone networks. Let $x_{k,m}^t$ represent the number of dispatched tasks from BS m in slice k, i.e.,

$$x_{k,m}^t \in \{0, 1, 2, ..., A_{k,m}^t\}, \forall m \in \mathcal{M}^w. \quad (6)$$

The operation decisions impact service delay at each operation slot, which is analyzed in the following subsection.

C. Service Delay Model

The service delay includes task offloading delay and task processing delay at either the edge or the cloud. The following analysis is for delay-sensitive service k.

1) Task offloading delay: The transmission rate of one subcarrier from vehicle n to its associated BS is given by $R_n^t = \beta \log_2\left(1 + \frac{P_v g_n^t}{\beta N_o + \beta I}\right)$, where P_v, g_n^t, N_o, and I represent vehicle's transmission power, instantaneous channel gain, noise spectrum density, and interference spectrum density, respectively. With the allocated radio spectrum $y_{k,n}^t b_{k,m}^w$, task offloading delay of vehicle n is given by $d_{k,n,o}^t = \frac{\xi_k}{y_{k,n}^t b_{k,m}^w R_n^t}, \forall n \in \mathcal{N}_m^t$, where ξ_k (in bits) denotes the task data size of service k.

2) Edge processing delay: Given the task dispatching decision, $A_{k,m}^t - x_{k,m}^t$ tasks are processed at BS m. Let $Q_{k,m}^t$ (in bits) denote the amount of backlogged tasks at BS m. Taking task computation delay and queuing delay into account, edge processing delay at BS m is given by $d_{k,m,e}^t = \frac{\left(Q_{k,m}^t + (A_{k,m}^t - x_{k,m}^t + 1)\xi_k/2\right)\eta_k}{c_{k,m}^w F_e}, \forall m \in \mathcal{M}^w$, where η_k (in cycles/bit) denotes task computation intensity of service k, and $c_{k,m}^w F_e$ is the computing capability of BS m with $c_{k,m}^w$ provisioned edge VMs. The task backlog at BS m is updated by $Q_{k,m}^{t+1} = \left[Q_{k,m}^t + (A_{k,m}^t - x_{k,m}^t)\xi_k - c_{k,m}^w F_e T_o/\eta_k\right]^+$, where $[x]^+ = \max\{x, 0\}$.

3) Cloud computation delay: For BS m, $x_{k,m}^t$ tasks are dispatched via backbone networks and then processed at the cloud, whose delay is given by $d_{k,m,c}^t = d_r^t + \frac{\xi_k \eta_k}{h_k^w F_c}$, where d_r^t denotes the round trip time in the backbone network. The second term represents the task processing delay in the cloud. Note that queuing delay at the cloud is negligible as multi-core cloud servers parallelly process tasks.

As such, the average delay for each computation task is

$$D_k^t(\mathbf{x}_k^t, \mathbf{y}_k^t) = \sum_{m\in\mathcal{M}^w} \sum_{n\in\mathcal{N}_m^t} \frac{d_{k,n,o}^t}{\sum_{m\in\mathcal{M}^w} N_m^t} + \sum_{m\in\mathcal{M}^w} \frac{d_{k,m,e}^t\left(A_{k,m}^t - x_{k,m}^t\right) + d_{k,m,c}^t x_{k,m}^t}{\sum_{m\in\mathcal{M}^w} A_{k,m}^t}. \quad (7)$$

In the above equation, the first term represents the average task offloading delay for each task, and the second term represents the average task processing delay considering workload distribution between the edge and cloud. The service delay averaging all operation slots is calculated by $\bar{D}_k^w = \frac{1}{T}\sum_{t=1}^T D_k^t(\mathbf{x}_k^t, \mathbf{y}_k^t)$.

D. Network Slicing Cost Model

The network slicing cost includes several components.

1) Slice deployment cost: The cost is due to running network slicing at BSs incurs the overhead of resource virtualization, which is given by $\Phi_d^w = q_d \sum_{m\in\mathcal{M}_s} o_m^w$. Here, q_d denotes the unit cost of running network slicing at a BS.

2) Resource provisioning cost: The cost component characterizes resource provisioning cost of edge radio spectrum and computing resources, and cloud computing resources. For simplicity, we assume the unit costs of a subcarrier, an edge VM instance, and a cloud VM instance are the same, denoted by $q_p > 0$. The resource provisioning cost is given by $\Phi_p^w = q_p \sum_{k\in\mathcal{K}} \left(h_k^w + \sum_{m\in\mathcal{M}} \left(o_m^w b_{k,m}^w + o_m^w c_{k,m}^w\right)\right)$.

[1]Memory resource is also allocated to the VM instance to enable task processing, which is matched to its allocated computing resource.

3) Slice adjustment cost: The cost component characterizes the difference between two subsequent planning decisions, i.e., the cost for adjusting the amount of reserved spectrum and computing resources. For computing resources, VM instances can be resized according to service demands via advanced virtualization techniques in practical systems, such as Docker and Kubernetes [13]. Here, q_s represents the unit price of adjusting a unit of reserved network resources. Hence, the slice adjustment cost is given by

$$\Phi_s^w = q_s \mathbb{1}\left\{o_{k,m}^{w-1} = 1 \wedge o_{k,m}^{w} = 1\right\} \cdot \sum_{k\in\mathcal{K}} \left(\left[h_k^w - h_k^{w-1}\right]^+ + \sum_{m\in\mathcal{M}} \left(\left[b_{k,m}^w - b_{k,m}^{w-1}\right]^+ + \left[c_{k,m}^w - c_{k,m}^{w-1}\right]^+ \right) \right), \tag{8}$$

where $\mathbb{1}\{\cdot\}$ is an indicator function and $\mathbb{1}\left\{o_{k,m}^{w-1} = 1 \wedge o_{k,m}^{w} = 1\right\}$ indicates that slices are deployed in both the previous and current planning windows.

4) SLA revenue: The cost component characterizes benefit caused by QoS satisfaction, i.e., the achieved service delay of each slice. The piece-wise revenue function is represented by

$$\Omega_k(D) = \begin{cases} q_r, & \text{if } D < \theta_k^{'}, \\ q_r\left(\frac{D-\theta_k^{'}}{\theta_k-\theta_k^{'}}\right), & \text{if } \theta_k^{'} \le D \le \theta_k, \\ -q_p, & \text{if } D > \theta_k, \end{cases} \tag{9}$$

where $q_r > 0$ is the unit revenue once a slice's SLA is satisfied, and $q_p > 0$ is the unit penalty once the slice's SLA is violated. Obviously, $q_p > q_r$ for discouraging slice's SLA violation. In addition, $\theta_k^{'} < \theta_k$ represents the threshold achieving the highest revenue. For simplicity, we set $\theta_k^{'} = \theta_k/2$ in the simulation. The overall SLA revenue of all slices is given by $\Phi_q^w = \sum_{k\in\mathcal{K}} \Omega_k\left(\bar{D}_k^w\right)$.

Taking all cost components into account, the overall network slicing cost in the entire slice lifecycle (i.e., all planning windows) is given by $\Phi\left(\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w, \{\mathbf{x}_k^t, \mathbf{y}_k^t\}_{t\in\mathcal{T}, k\in\mathcal{K}}\right) = \sum_{w\in\mathcal{W}}\left(\Phi_d^w + \Phi_p^w + \Phi_s^w - \Phi_q^w\right)$, which is adopted to metric network slicing performance.

III. Problem Formulation

The network slicing problem aims to minimize the network slicing cost via determining network planning decisions at each planning window and network operation decisions at each operation slot for each slice, which is formulated as:

$$\mathbf{P}_0: \min_{\substack{\{\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w\}_{w\in\mathcal{W}} \\ \{\mathbf{x}_k^t, \mathbf{y}_k^t\}_{t\in\mathcal{T}, k\in\mathcal{K}, w\in\mathcal{W}}}} \sum_{w\in\mathcal{W}} \Phi\left(\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w\right) \quad \text{s.t. } (1), (2), (3), (4), (5), \text{and } (6). \tag{10a}$$

In Problem $\mathbf{P}_0$, the network planning and operation decision making are coupled in two timescales, which should be jointly optimized. To address the challenge, we first decouple the problem into a large-timescale network planning subproblem and multiple small-timescale network operation subproblems.

Subproblem 1: *Network planning subproblem* is to minimize the network slicing cost across all the planning windows, which is formulated as:

$$\mathbf{P}_1: \min_{\substack{\{\mathbf{o}^w, \mathbf{B}^w, \\ \mathbf{C}^w, \mathbf{h}^w\}_{w\in\mathcal{W}}}} \sum_{w\in\mathcal{W}} \Phi\left(\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w\right) \quad \text{s.t. } (1), (2), (3), \text{and } (4). \tag{11a}$$

Addressing the above subproblem requires network traffic information of all planning windows, which is difficult to be known *a priori*. To solve it, we leverage an RL method to design a network planning algorithm, which makes online decisions under spatial-temporally varying vehicle traffic.

Subproblem 2: *Network operation subproblem* is to schedule network resources of each slice to active vehicles with random task arrivals with the objective of minimizing average service delay, which is formulated as:

$$\mathbf{P}_2: \min_{\mathbf{x}_k^t, \mathbf{y}_k^t} D_k^t(\mathbf{x}_k^t, \mathbf{y}_k^t) \quad \text{s.t. } (5) \text{ and } (6). \tag{12a}$$

In the above subproblem, radio spectrum resource allocation and task dispatching decisions jointly impact the service delay performance. To solve the problem, we analyze the subproblem property and design an optimization algorithm to make real-time network operation decisions.

IV. Learning-Based Network Slicing Algorithm

In this section, we solve two subproblems in Sections IV-A and IV-B, respectively. Finally, we present the TWAS algorithm for jointly optimizing planning and operation decisions in Section IV-C.

A. Network Operation Optimization

We can observe that the radio spectrum allocation decision only impacts offloading delay component, and the task dispatching decision only impacts the computation delay component. Moreover, both decisions are independent in each BS. Hence, the radio spectrum allocation and task dispatching decisions can be optimized individually at each BS.

1) Radio Spectrum Allocation Optimization: From (7), the radio spectrum allocation optimization problem is equivalent to minimizing the task offloading delay at each BS, i.e.,

$$\mathbf{P}_m^r: \min_{\mathbf{y}_k^t} \sum_{n\in\mathcal{N}_m^t} \frac{\xi_k}{y_{k,n}^t b_{k,m}^w R_n^t} \quad \text{s.t. } (5). \tag{13a}$$

The objective function can be proved to be convex since its second-order derivative is positive. In addition, the constraint is convex. Hence, problem $\mathbf{P}_m^r$ is a convex optimization problem. Using the Karush-Kuhn-Tucker conditions [14], the optimal radio spectrum resource allocation decision is

$$(y_{k,n}^t)^\star = \frac{\sqrt{1/R_n^t}}{\sum_{i\in\mathcal{N}_m^t}\sqrt{1/R_i^t}}, \forall n \in \mathcal{N}_m^t. \tag{14}$$

2) Task Dispatching Optimization: Similarly, from (7), task dispatching optimization is to minimize the task processing

delay, which is formulated as:

$$\mathbf{P}_m^w : \min_{x_{k,m}^t} d_{k,m,e}^t \left(A_{k,m}^t - x_{k,m}^t\right) + d_{k,m,c}^t x_{k,m}^t \quad \text{s.t. (6).} \tag{15a}$$

The above objective function can be rewritten as

$$\begin{aligned}\Psi(x_{k,m}^t) &= d_{k,m,e}^t \left(A_{k,m}^t - x_{k,m}^t\right) + d_{k,m,c}^t x_{k,m}^t \\ &= \frac{\nu_1 \xi_k}{2}(x_{k,m}^t)^2 + \left(\nu_2^t - \nu_1\nu_3 - \frac{\xi_k A_{k,m}\nu_1}{2}\right) x_{k,m}^t \\ &+ \nu_1 \nu_3^t A_{k,m}^t.\end{aligned} \tag{16}$$

Here, $\nu_1 = \frac{\eta_k}{c_{k,m}^w F_e} > 0$, $\nu_2^t = d_r^t + \frac{\eta_k \xi_k}{h_k^w F_c}$, and $\nu_3 = Q_{k,m} + \frac{A_{k,m}+1}{2}\xi_k$. Since the second-order derivative of the objective function $\partial^2 \Psi(x_{k,m}^t)/\partial^2 x_{k,m}^t = \nu_1^t \xi_k > 0$, the problem is a convex optimization problem [14]. The optimal task dispatching decision is given by

$$(x_{k,m}^t)^\star = \frac{2\nu_2^t + \xi_k \nu_1 A_{k,m} - 2\nu_1 \nu_3^t}{2\nu_1 \xi_k}, \forall m \in \mathcal{M}^w. \tag{17}$$

B. Network Planing Optimization

The network planning problem is a stochastic optimization problem to minimize the network slicing cost, which can be transformed into a Markov decision process (MDP) [10]. The components of the MDP are defined as follows.

1) Action, which is consistent with planning decisions, including slice deployment, edge radio spectrum and computing resource provisioning at BSs, and cloud computing resource provisioning, i.e., $A^w = \{\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w\}$. The action dimension is $M_s + 2KM + K$.

2) State, which includes average vehicle traffic density in the current planning window and the planning decisions in the previous window due to the switching cost. The entire area is divided into J disjoint regions, and the average vehicle traffic density of all regions is denoted by $\Lambda^w \in \mathbb{R}^{J\times 1}$. As such, the state is given by $S^w = \{\Lambda^w, \mathbf{o}^{w-1}, \mathbf{B}^{w-1}, \mathbf{C}^{w-1}, \mathbf{h}^{w-1}\}$. The state dimension is $2KM + M + K + J$.

3) Reward, which is defined as the inverse of the network slicing cost in the current planning window, i.e., $R^w\left(S^w, A^w\right) = -\Phi\left(\mathbf{o}^w, \mathbf{B}^w, \mathbf{C}^w, \mathbf{h}^w\right)$. Note that minimizing the network slicing cost is equivalent to maximizing the cumulative reward.

Upon state S^w, the learning agent takes action A^w, and the corresponding reward $R^w\left(S^w, A^w\right)$ is obtained, along with the state evolves into new state S^{w+1}. With the above setting, our goal is to obtain an optimal planning policy $\pi^\star \in \Pi$ which makes decisions based on the observed state, thereby maximizing the expected long-term cumulative reward. As such, problem $\mathbf{P}_2$ can be formulated as the following MDP:

$$\mathbf{P}_2' : \max_{\pi\in\Pi} \mathbb{E}\left[\lim_{W\to\infty} \sum_{w=1}^{W} (\varphi)^w R^w\left(S^w, A^w\right) | \pi\right], \tag{18a}$$

where $\varphi > 0$ is the discount factor. Since vehicle traffic density is continuous, the action-state space can be prohibitively large. To address this issue, an RL algorithm can be adopted.

Algorithm 1: TAWS algorithm.

```
for training episode =1, 2, ... do
    for planning window w = 1, 2, ..., W do
        Generate planning decisions via actor network;
        for each slice in parallel do
            for operation slot t = 1, 2, ..., T do
                for each BS in parallel do
                    Make radio spectrum allocation and task
                      dispatching decisions by (14) and (17);
                Calculate instantaneous service delay;
            Measure average service delay within the
              planning window;
        Collect vehicle traffic density of all regions, and
          observe reward R^w and new state S^{w+1};
        Store transition {S^w, A^w, R^w, S^{w+1}} in the
          experience replay buffer;
        Sample a random minibatch transitions from the
          experience replay buffer;
        Update weights of neural networks;
```

C. Proposed TAWS Algorithm

We present the TAWS algorithm to jointly solve the entire network slicing problem $\mathbf{P}_0$, collaboratively integrating RL [15] and optimization methods. The core idea of TAWS is to adopt an RL method for network planning decision making and an optimization method for network operation decision making. The service delay performance is measured at the end of each planning window and then incorporated into the reward in the RL framework, such that the interaction between network planning and operation stages can be captured. The TAWS algorithm is shown in Algorithm 1.

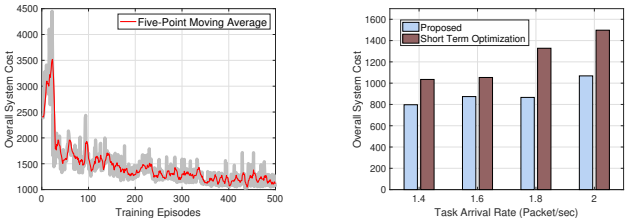
The RL method is based on the deep deterministic policy gradient (DDPG) algorithm [16], which consists of four neural networks, i.e., actor evaluation network, critic evaluation network, actor target network, and critic target network. In the initialization phase, all neural networks and the environment is initialized. The procedure of the TAWS is two-step: 1) Network slicing decisions are generated and executed. The actor network outputs the planning decisions A^w, which is clipped to feasible decision space. The network operation decisions are generated via the optimization method, and the service delay performance is measured at the end of each planning window. The reward R^w can be obtained and the new state can be observed S^{w+1}. The transition tuple $\{S^w, A^w, R^w, S^{w+1}\}$ is stored in the experience replay buffer for updating neural networks; and 2) Neural networks are updated. A mini-batch of transitions are randomly sampled from the experience replay buffer to update the weights of neural networks. Based on which, the critic network is updated by minimizing the loss function. The actor network is updated via policy gradient. Then, actor and critic target networks are updated by slowly copying the weights of evaluation networks.

V. Simulation Results

We evaluate the performance of the proposed algorithm on real-world vehicle traffic traces in urban vehicular networks. We consider a 1,000×1,000 m^2 simulation area, which is covered by two SBSs and an MBS. Each SBS has a coverage

Table I
SIMULATION PARAMETERS.

Parameter	Value	Parameter	Value
N_o	-174 dBm	I	-164 dBm
P_v	27 dBm	β	20 MHz
d_r	0.15 sec	J	16
T_o	1 sec	T_p	10 min
F_c	100 GHz	F_e	10 GHz
B_m	10	C_m	10
ξ_1, ξ_2	$\{0.6, 2\}$ Mbit	η_1, η_2	$\{1000, 200\}$ cycles/bit
θ_1, θ_2	$\{100, 200\}$ ms	θ_1, θ_2	$\{50, 100\}$ ms
q_p	1	q_n	10
q_s	5	q_p	200
q_r	10		



(a) Convergence (b) Network slicing cost

Fig. 2. Performance of the proposed TWAS algorithm.

radius of 300 m, and the MBS located in the centre can cover the entire area. The vehicle traffic density of the simulation area is measured by a unit of a small region of 250×250 m^2, i.e., $J = 16$. This dataset is collected by Didi Chuxing GAIA Initiative[2] and contains vehicle traces in the second ring road in Xi'an collected from taxis that are equipped with GPS devices. The periods of a planning window and an operation slot are set to 10 minutes and 1 second, respectively. The period of the slice lifecycle is set to 4 hours, including 24 planning windows. The task arrivals follow a Poisson process. We construct two slices for supporting two types of delay-sensitive services. One is an object detect service whose service delay requirement is 100 *ms*, while the other is an in-vehicle infotainment service whose service delay requirement is 200 *ms*. Regarding the TWAS algorithm, the learning rates of actor and critic networks are set to 5×10^{-4} and 5×10^{-3}, respectively. The neuron units in hidden layers of both actor and critic networks are set to 128 and 64. Important simulation parameters are summarized in Table I.

As shown in Fig. 2(a), we present the overall network slicing cost with respect to training episodes. All simulation points are processed by a five-point moving average in order to highlight the convergence trend of the proposed solution. It can be seen that the learning-based algorithm has converged after 500 training episodes.

As shown in Fig. 2(b), we compare the performance of the proposed learning-based planning policy and a short term optimization benchmark. The benchmark is to minimize the network slicing cost at each planning window. Since planning decisions are discrete, a simple exhaustive searching method is adopted to obtain the optimal one-shot planning decisions. Firstly, it can be seen that the proposed algorithm can greatly reduce the network slicing cost as compared to the benchmark. Specifically, when the task arrival rate is 2 packets per second, the proposed solution can reduce the overall network slicing cost by 23%. The reason is that the proposed solution takes the switching cost between two consequent planning windows into account, while the benchmark scheme does not. Secondly, the overall network slicing cost increases with the increase of the task arrival rate because more spectrum and computing resources are consumed in a heavy traffic scenario.

[2]Didi Chuxing Dataset: https://gaia.didichuxing.com.

VI. CONCLUSION

In this paper, we have investigated a network slicing problem in edge-cloud orchestrated vehicular networks. A two-stage network slicing algorithm, named TWAS, is proposed to jointly make network planning and operation decisions in an online fashion. It can adapt to network dynamics in different timescales, including spatial-temporally varying vehicle traffic density and random task arrivals. Simulation results have demonstrated that the TAWS can reduce the network slicing cost as compared to benchmarks. For future work, we aim to determine the optimal planning window size for minimizing the network slicing cost under vehicular network dynamics.

REFERENCES

[1] C. Campolo, A. Molinaro, A. Iera, and F. Menichella, "5G network slicing for vehicle-to-everything services," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 38–45, 2017.
[2] A. Kaloxylos, "A survey and an analysis of network slicing in 5G networks," *IEEE Commun. Standards Mag.*, vol. 2, no. 1, pp. 60–65, 2018.
[3] "Telecommunication management; Study on management and orchestration of network slicing for next generation network (Release 15)," 3GPP TR 28.801 V2.0.1, Sep. 2017.
[4] "Network functions virtualisation (NFV) Release 3; evolution and ecosystem; Report on network slicing support with ETSI NFV architecture framework," Tech. Rep. ETSI GR NFV-EVE 012 V3.1.1, Dec. 2017.
[5] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: RAN slicing for a flexible and cost-effective multi-service mobile network architecture," in *Proc. ACM MobiCom*, 2017, pp. 127–140.
[6] X. You *et al.*, "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Sci. China Inf. Sci.*, vol. 64, no. 1, pp. 1–74, 2021.
[7] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 1st. Quart. 2022.
[8] Q. Ye, W. Zhuang, S. Zhang, A. Jin, X. Shen, and X. Li, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9896–9910, Oct. 2018.
[9] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6063–6078, Sep. 2021.
[10] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, Jul. 2021.
[11] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, no. 1, pp. 45–66, Jan. 2020.
[12] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. ASPLOS*, 2018, pp. 751–766.
[13] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sep. 2014.
[14] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[15] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," in *Proc. ICLR*, 2016.
[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016.