IEEE ComSoc IEEE Open Journal of the Communications Society

Received XX Month XXXX; revised X Month XXXX; accepted XX Month XXXX. Date of publication XX Month XXXX; date of current version XX Month XXXX.

Digital Object Identifier 10.1109/OJCOMS.2020.1234567

Joint Distributed Computation Offloading and Radio Resource Slicing Based on Reinforcement Learning in Vehicular Networks

Khaled A. Alaghbari[1], Heng-Siong Lim[2,3*], C. Zarakovitis[3], N. M. Abdul Latiff[1], Sharifah Hafizah Syed Ariffin[1] AND S. F. Chien[3,4]

[1]Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310, UTM Johor Bahru, Malaysia.
[2]Faculty of Engineering and Technology, Multimedia University (MMU), 75450 Bukit Beruang, Melaka, Malaysia
[3]Axon logic IKE, ICT Department, 14122, Athens, Greece
[4]MIMOS Berhad, 57000 Kuala Lumpur, Malaysia

CORRESPONDING AUTHOR: Heng-Siong Lim (e-mail: hslim@mmu.edu.my).

This research was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101093069 (Programming Platform for Intelligent Collaborative Deployments over Heterogeneous Edge-IoT Environments)

ABSTRACT Computation offloading in Internet of Vehicles (IoV) networks is a promising technology for transferring computation-intensive and latency-sensitive tasks to mobile-edge computing (MEC) or cloud servers. Privacy is an important concern in vehicular networks, as centralized system can compromise it by sharing raw data from MEC servers with cloud servers. A distributed system offers a more attractive solution, allowing each MEC server to process data locally and make offloading decisions without sharing sensitive information. However, without a mechanism to control its load, the cloud server's computation capacity can become overloaded. In this study, we propose distributed computation offloading systems using reinforcement learning, such as Q-learning, to optimize offloading decisions and balance computation load across the network while minimizing the number of task offloading switches. We introduce both fixed and adaptive low-complexity mechanisms to allocate resources of the cloud server, formulating the reward function of the Q-learning method to achieve efficient offloading decisions. The proposed adaptive approach enables cooperative utilization of cloud resources by multiple agents. A joint optimization framework is established to maximize overall communication and computing resource utilization, where task offloading is performed on a small-time scale at local edge servers, while radio resource slicing is adjusted on a larger time scale at the cloud server. Simulation results using real vehicle tracing datasets demonstrate the effectiveness of the proposed distributed systems in achieving lower computation load costs, offloading switching costs, and reduce latency while increasing cloud server utilization compared to centralized systems.

INDEX TERMS Computation offloading, radio resource slicing, reinforcement learning, Q-learning, distributed system, mobile-edge computing (MEC), cloud computing, Internet of vehicles.

I. INTRODUCTION

In modern Internet of Vehicles (IoV) transportation systems, vehicles are connected in networks to support applications like smart driving, traffic management or augmented reality (AR). These tasks may require significant computing power and low-latency processing [1]. To manage these demands, vehicles can offload some of their computational tasks to nearby servers, such as edge servers, or to more powerful servers, such as cloud servers. This process, called task computation offloading, helps reduce the load on vehicles and improves efficiency. It is important to ensure that server computing capacity and radio communication resources are used optimally, to achieve a balanced computational load while meeting task transmission rates, latency and reliability requirements. Radio access network (RAN) slicing enables resource sharing among base stations (BSs) for finer orchestration, improving utilization and ensuring quality-of-service (QoS) isolation. Network

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

function virtualization (NFV) virtualizes radio resources into a centralized pool, which is then managed and dynamically allocated by a software-defined networking (SDN)-enabled slicing controller based on network traffic and QoS demands [2].

Centralized systems involve offloading computational tasks from vehicles to a central cloud server via a macro base station (MBS), which can handle complex tasks efficiently than vehicles, but may compromise privacy due to the collection of sensitive raw data such as location and behavioural information from vehicles. An inherent limitation of centralized cloud computing system is the long propagation distance from mobile vehicles to the remote cloud server, which is not suitable for massive data and delay sensitive tasks [3]. Moreover, existing centralized methods suffer from dramatic increase in control overhead especially as the system scale increases, hence limiting their application in vehicular networks [4]. In contrast, distributed systems with mobile-edge computing (MEC) paradigm distribute the computation tasks among edge servers located at small base stations (SBSs), preserve privacy in vehicular networks by reducing data exposure to a central point, release the burden on the MBS, and support latency-critical and computation-intensive applications [5]. Deploying MEC servers near BSs enables local processing of computational tasks, reducing latency and supporting time-sensitive applications like autonomous driving and real-time analytics. This proximity makes MEC practical for dynamic vehicular networks in dense urban and suburban areas.

The increasing demands of advanced vehicle applications can overwhelm the distributed servers, resulting in longer computation times and higher power consumption. Additionally, the resources of the distributed servers are expensive and increase the cost of the system [6]. Hence, the resources of the cloud server can be shared among the SBSs, allowing vehicles to offload tasks either to the edge server at SBS within its coverage area or to the cloud server at MBS [7]. However, the cloud server can be overloaded by vehicles at certain SBS without given portion of the resource to other vehicles presented at other SBS. Furthermore, the challenge is still present by limited radio resources and the dynamic nature of vehicular networks that require sophisticated strategies to maintain seamless connectivity and performance. Due to the correlation between the two problems, computing task offloading and radio resource slicing, it is important to design a joint optimization framework to determine an optimal radio resource slicing ratio and computation capacity for efficient computation load balancing.

Unlike existing distributed approaches, our framework introduces a novel joint two-timescale optimization with a threshold ratio mechanism incorporated into optimization problem constraints for cloud resource allocation, balancing computational load while minimizing task-switching cost. We formulated the reinforcement learning (RL) reward function to fairly allocate the computation resources of the cloud server located at macro base station (MBS). Two threshold-ratio based distributed approaches are proposed, the first approach, fixed-threshold distributed system (Dist-fixed), uniformly distributes the resource among the small base stations (SBSs). The second approach, adaptive-threshold distributed system (Dist-adptv), dynamically distributes the cloud computation resource among the SBSs using a fair-proportional approach based on the load requirements which are calculated from previous time slot and sent to the main server to compute suitable ratios for each agent. Q-learning (QL) is used by each agent to optimize the task offloading decisions across scheduling slots in small timescale, to balance the computation load across the network while controlling the number of task offloading switching. A large time-scale convex optimization problem is then solved at the main server, which does not require sensitive data to be shared by the agents, to compute optimal radio resource slicing ratio that maximizes the overall communication resource utilization with guaranteed QoS. The advantage of the Dist-fixed approach is to ensure fairness by uniformly distributing cloud resources across BSs, reducing the risk of overloading specific BSs and simplifying implementation. This makes it well-suited for stable or predictable traffic load environments. Meanwhile, the Dist-adptv approach dynamically adjusts resource allocation based on load requirements, allowing the system to respond effectively to rapid changes in traffic patterns or load imbalances, making it ideal for highly dynamic and dense vehicular scenarios. These mechanisms enable fair and efficient resource utilization while jointly optimizing task offloading and radio resource slicing. Specifically, the contributions of our work can be summarized as follows:

1) We consider a vehicular network that includes distributed edge servers and a cloud server. The vehicles can choose to offload the task either to MBS or SBS. Our solution model consists of two timescales. At the small timescale, we solve stochastic optimization problem using Q-learning technique to minimize the total system cost that includes imbalanced computation load cost and offloading switching cost, and takes into consideration the cloud server computation resource. The Q-learning agents are deployed at each SBS. At the large timescale, convex optimization problem is solved at the main server to obtain radio resource slicing ratios to maximize the overall communication resource utilization.
2) Since the distributed system without a mechanism to control the MBS server resource can cause imbalance load distribution (unaware of the cloud server workload), we propose a fixed distributed (Dist-fixed) approach to fairly share the cloud server resource, and an adaptive (Dist-adptv) approach to divide the computation resource among BSs based on computation load requirements obtained from each agent at the previous time slot.

3) Extensive simulation results are presented using real vehicle traffic dataset to demonstrate the effectiveness of the proposed distributed frameworks compared to centralized scheme in terms of cost of imbalanced computation load, offloading switching, latency, MBS overload rate and utilization.

The remainder of this paper is organized as follows: Section II provides a review of related works. Section III presents the system model that includes network model, communication model, small time-scale computing task offloading model, problem formulation, and large timescale-based radio resource slicing model. Section IV presents our proposed approaches. Section V provides information about the dataset, network simulation parameters, experimental results and discussion for small-time scale and large-time scale analyses. Finally, the conclusion is drawn in Section VI.

II. RELATED WORKS

Existing centralized methods [8] suffer from inherent property of depending on a central controller to handle large computing tasks, causing a dramatic increase in control overhead, especially as the system scale increases, hence limiting their application in vehicular networks [6]. Reinforcement learning (RL) methods have shown great potential in computation task offloading. Papers [9-11] designed different task offloading schemes based on RL techniques, which behave well in reducing average delay and improving resource utilization. However, to fully utilize the resources in the vehicular networks and further enhance the QoS, there is a need to jointly optimize task offloading decisions, allocation of radio resources at the RAN and computation resources at the cloud computing server.

Jiang et al. [12] used Q-learning to obtain an optimal policy for computation offloading and resource allocation in a multi-user MEC system, considering different resource requirements and time-varying system conditions in a dynamic system. The objective was to minimize the long-term energy consumption of all the UEs considering the latency constraint and dynamic computation resource requirements of heterogeneous computation tasks. Dab et al. [13] employed Q-learning in multi-user WiFi-based MEC architecture for task assignment and radio resource allocation to minimize the energy consumption on the mobile terminal side while considering latency constraint. Due to the limited computation capability of the mobile edge computing (MEC) systems, which restrict the scalability of offloading, Gao et al. [14] proposed to jointly optimize the computation resource allocations and offloading decisions for collaborative computing system that combines local computing (mobile device), MEC (edge cloud) and central mobile cloud computing (MCC). Once the task is received by edge cloud, it will be split into parts, one part will be transferred to the central cloud and the other part is executed on the edge-cloud server, enabling parallel processing. Q-learning was used to minimize the system loss function formulated based on time and energy consumptions to optimize offloading decisions. Jiang et al. in [15] proposed to solve the task offloading and resource allocation problem for Internet of Vehicles (IoV) networks using Q-learning. Bayesian classifier was first implemented to classify the task according to latency and energy consumption requirements. Then each vehicle selects one of the two available offloading modes. The first mode is to offload the task to other vehicles through vehicle-to-vehicle (V2V) communication, if the vehicle has a higher energy requirement. Otherwise, it selects to offload the task to an edge server through the MEC offloading mode. In the V2V offloading mode, the radio resources need to be allocated, and in the MEC offload mode, the computing resources need to be designated.

Deep RL often struggles to achieve good performance and the trained system may behave unpredictably if the environment differs even slightly from the training data [16]. To address the problem of instability in the multi-agent environment and to attain queue stability during resource allocation, Kumar et al. [17] proposed a Lyapunov-based multi-agent deep deterministic policy gradient (L-MA DDPG) technique to jointly optimize the task offloading and radio resource allocation. The main objective was to minimize the energy consumption and meet delay requirements between vehicles and edge servers due to vehicle mobility and dynamic environment. Since existing deep learning techniques suffer from slow learning rate and weak adaptability to dynamic multi-user conditions, Sharma et al. [18] proposed using first-order meta-learning with a deep Q-learning method for multi-task offloading in edge-cloud networks. In conventional actor-critic RL network, the large number of parameters makes the training model inefficient, and the usage of one-step temporal difference learning causes slow convergence. Hence, Geng et al. in [6] proposed to use an improved actor–critic with 2D convolution and LSTM layers to extract features, and joint mechanism of prioritized experience replay and adaptive learning to enhance the learning efficiency. The proposed method was used for distributed computation offloading in vehicular edge computation networks with the objective to minimize the delay and energy consumption.

Ye et al. [7] proposed a two-tier framework that integrates radio access network (RAN) slicing and computation offloading for autonomous vehicular networks (AVNs) to address the dynamic nature of AVNs. On a smaller timescale, they optimized task scheduling using a cooperative multi-agent deep Q-learning (MA-DQL) with fingerprint algorithm, to learn the stationary task offloading policy with stabilized learning performance, and to balance computational load and minimize task offloading variations. On a larger timescale, they optimized radio resource slicing among base stations to maximize network utility while ensuring QoS for autonomous driving tasks. However, all local agents' actions need to be synchronized at the main server to calculate a joint system reward, which is then sent back to the agents to train the module. In addition, MA-DQL algorithm can suffer from unstable convergence due to

TABLE 1: Summary of related works

Ref.	Key Approach	Optimization Focus	Framework Type	Learning Algorithm
[6]	Distributed computation offloading	Optimizing computation offloading for delay and energy efficiency	Distributed approach	Deep RL (improved actor–critic network)
[7]	Joint RAN slicing and computation offloading	Optimizing RAN slicing and task offloading (load balancing and offloading switching)	Two-tier framework (centralized and distributed)	DQL
[12]	Joint computation offloading and MEC resource allocation	Minimizing energy consumption for task offloading	Distributed (one MEC considered)	QL
[14]	Task offloading and resource optimization in a collaborative cloud computing system	Task offloading optimization with resource constraints Task splitting ratio.	Centralized (tasks partially executed on edge and cloud)	QL
[19]	joint optimization of computation offloading and bandwidth resource allocation scheme	Total cost of processing tasks includes computation cost and bandwidth leasing cost	Centralized approach	Twin delayed deep deterministic policy gradient (TD3)
Our work	+ Joint RAN slicing and computation offloading. + Dist-fixed and Dist-adptv mechanisms for resource allocation of cloud server.	Optimizing RAN slicing and task offloading (load balancing and offloading switching)	Two-tier framework (centralized and distributed)	QL

factors like insufficient training data and suboptimal exploration strategies leading to divergence in the learned Q-values. Huang et al. [19] used deep RL technique to jointly optimize computation offloading and resource allocation with the aim to minimizing the system cost of processing tasks while meeting the processing latency and transmission rate constraints for IoV networks. The cost of processing tasks contains computation cost and communication bandwidth rental cost. The central controller hosts the DRL agent to make the task offloading and resource allocation decisions. To tackle the challenges of random traffic flow and dynamic network environment scenario, Markov decision process model was employed for formulating the problem, then twin delayed deep deterministic policy gradient (TD3) technique was used to deal with the continuous states and action spaces. However, their proposed method consists of two neural networks, a main network and a target network, each with one actor network and two critic networks. The complexity in the proposed technique was introduced to make the training process more stable. In addition, the DRL agent was employed at the centralized controller which collects private data such as status of vehicles tasks, bandwidth resources and edge server computation resources.

Compared to existing works, our approach differs in methodology and focus. Unlike [7], which maximizes network utility through cooperative MA-DQL for task scheduling, where the rewards of the RL agents are synchronized at the centralized server, our proposed method updates the rewards locally while taking into consideration the limited cloud resources. Specifically our framework employs fixed and adaptive-threshold mechanisms to ensure balanced computation load by fairly and dynamically allocating cloud resources among BSs. Additionally, while studies like [7, 12, 14] optimize delay or energy consumption, our approach incorporates a switching cost to minimize task offloading variations, enhancing stability in dynamic vehicular networks. In contrast to centralized methods such as [19], which require sharing sensitive vehicle data, our framework preserves privacy by enabling efficient distributed optimization without raw data exposure to the centralized server. Furthermore, both our work and [7] adopt a two-tier framework, with the upper layer optimizing RAN slicing over a longer timescale and the lower layer focusing on distributed computation offloading. While [7] employs DQL, our work uses QL for its simplicity, computational efficiency, and suitability for low-complexity scenarios in vehicular networks. Unlike Deep QL or DDPG, QL does not rely on extensive computational resources or large datasets, making it a more practical choice for efficient decision-making in resource-constrained settings. Table 1 summarizes the related studies.

III. SYSTEM MODELS

In this section, we introduce the system models, including the network, communication channel, computation task, offloading switching models, and define their associated variables. Then, we present the problem formulation for both small timescale and large timescale.

A. NETWORK MODEL

We consider a macro-cell network centered around a single main base station (MBS) called S_0, positioned at the center of a cell to offer broad communication coverage along a road segment for vehicles, as illustrated in Fig. 1. This macro-cell is supplemented by several smaller cells, each centered around a small base station (SBS), labelled as $S_1, S_2, \ldots, S_n$. These SBSs are placed near the road within the macro-cell's coverage area to improve network capacity. To support the network's computational needs, there is a main server linked to the MBS for heavy-duty computations. Additionally, each SBS is accompanied by a local server equipped with lightweight computing resources. These local servers are physically connected to their respective SBSs. As vehicles travel along the road segment, they may enter or exit it over time. We assume that each vehicle remains within the coverage range of both the MBS and one of the SBSs, thus maintaining a connection to both base stations. To enable

tractable analysis, we divide the road segment under the MBS's coverage into distinct zones, denoted as $Z_0 = \{0, 1, \ldots, Z-1\}$. Each zone represents a segment of the road. At any given scheduling slot, we assume that the task of offloading decisions for all vehicles within a particular zone are the same. We denote the number of vehicles present in zone z of base station, S_k, at time slot, t, as $N_{k,z,t}$, and we assume that this number remains constant during the time slot.

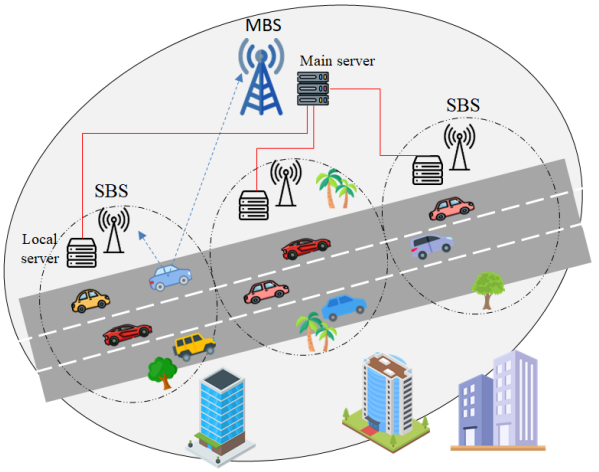


Fig.1. Illustration of the vehicular network with one MBS and three SBSs

B. COMMUNICATION MODEL

According to the Shannon capacity formula, the uplink transmission rate from each vehicle in a specific zone (z) to base station (BS) S_k at a particular scheduling slot (t) is determined by the following equation [7]:

$$r_{k,z,t} = \frac{W_k}{\sum_{z \in Z_k} N_{k,z,t}\, a_{k,z,t}} \log_2\left(1 + I_{k,z,t}\right) \quad (1)$$

where W_k represents the available bandwidth, which equals W_m if $k = 0$ (indicating the MBS), and W_s otherwise (for SBSs). The radio resources on S_k are divided equally among the vehicles connected to it for task offloading at slot t. The term $\sum_{z \in Z_k} N_{k,z,t}\, a_{k,z,t}$ represents the total number of tasks offloaded from vehicles in zone z to server S_k at time slot t. $a_{k,z,t}$ is the task offloading indicator for vehicles in zone z of S_k at slot t. It is set to 1 when all tasks from zone z are offloaded to S_k and 0 otherwise. This decision is dynamic, based on server capacity and latency requirements. $I_{k,z,t}$ represents the uplink signal-to-noise ratio (SNR) or signal-to-interference-plus-noise ratio (SINR). It is calculated differently based on whether k equals 0 (MBS) or not (SBS) as follows:

$$I_{k,z,t} = \frac{P_k G_{k,z,t} \alpha_{k,t}}{\sigma^2}, \quad \text{if } k = 0 \quad (2)$$

$$I_{k,z,t} = \frac{P_k G_{k,z,t} \alpha_{k,t}}{\sum_{j=\{1,2,..n\}, j \neq k} P_j G_{j,z',t} \alpha_{j,t} + \sigma^2}, \quad \text{if } k \neq 0 \quad (3)$$

In (2) and (3), P_k represents the uplink transmission power, which remains constant and uniform for all vehicles under base station S_k during a given planning window. The term $G_{k,z,t}$ signifies the uplink channel gain from vehicles within zone z to base station S_k at slot t. This channel gain includes path loss and log-normal shadowing, and is averaged across a group of vehicles within the zone. $\alpha_{k,t}$ denotes the small-scale Rayleigh fading component under base station S_k at slot t, and σ^2 indicates the average background noise power [20]. Additionally, for SBSs, the interference experienced by a vehicle in zone z under S_k originates from uplink transmissions from vehicles occupying the same zone z' position under every other SBS. These factors play a crucial role in determining the effectiveness and reliability of communication within the network.

C. COMPUTING TASK AND OFFLOADING SWITCHING MODELS (SMALL TIME-SCALE)

We assume that each computing task has a fixed size of H bits and a latency bound requirement, D, which is set to be equal to the duration of a scheduling slot, T. Initially, we compute the computation load ratio of the server connected to base station S_k (where $k = 0, 1, \ldots, n$) at scheduling slot t as:

$$L_{k,t} = \sum_{z \in Z_k} N_{k,z,t}\, a_{k,z,t}\, \frac{\varphi H}{C_k T} \quad (4)$$

where C_k denotes the computation capacity of the server connected to base station S_k, measured in CPU cycles per second. Additionally, φ denotes the computation intensity, indicating the number of CPU cycles needed to process one bit of information. The load ratio should be less than one, if it exceeds one, it indicates that the server is overloaded.

The cost associated with having an imbalance distribution of computation load among the servers at slot t, denoted as $C_{1,t}$, is represented by the maximum instantaneous computation level as [7]:

$$C_{1,t} = \max_{S_k \in B} \{L_{k,t}\} \quad (5)$$

where the set $B = \{S_0, S_1, \ldots, S_n\}$ refers to all the base stations being considered. The cost $C_{1,t}$ is designed to support an even distribution of computational tasks, preventing server overload and underutilization. The cost associated with changing task offloading decisions from slot $t-1$ to slot t, denoted as $C_{2,t}$, involves determining the total number of offloading switching events between the MBS and one of the SBSs for vehicles across all road zones, given by:

$$C_{2,t} = \sum_{S_k \in B} \sum_{z \in Z_k} \sum_{l \in B'/S_k} a_{k,z,t}\, a_{l,z,t-1} \quad (6)$$

where B' represents the set comprising both the MBS and the SBS that covers zone z.

The overall cost of balancing computation load, while also considering the cost associated with switching task offloading decisions at slot t, is determined as a weighted sum of $C_{1,t}$ and $C_{2,t}$, given by [7]:

$$C_t = \beta C_{1,t} + (1 - \beta) C_{2,t} \quad (7)$$

where β is a real-valued weighting factor ranging between 0 and 1. Our primary objective is to balance computation loads

among BSs while minimizing variations in task offloading. Constraints include computation capacity and task offloading latency. We achieve this by employing an MDP (Markov Decision Process) formulation, which captures network states and model the relationship between network states and offloading actions. At each scheduling slot *t*, the formulation includes network states $\mathcal{S}_t$, task offloading actions $\mathcal{A}_t$ extracted from policy $\pi(\mathcal{A}|\mathcal{S})$ that maximize the instantaneous reward function $R(\mathcal{S}_t, \mathcal{A}_t)$, and state transition probabilities $P(\mathcal{S}_{t+1} | \mathcal{S}_t, \mathcal{A}_t)$. S_t includes parameters such as numbers of vehicles $\mathcal{N}_t$, uplink SINR (or SNR) $\mathcal{I}_t$, and task offloading actions $\mathcal{A}_{t-1}$ taken for previous slot $(t-1)$. The problem is presented as a stochastic optimization framework, aims to balance computation load while minimizing variations in task offloading, given by:

$$\mathbf{P1}: \min_{\pi} \beta C_{1,t} + (1-\beta) C_{2,t}$$
$$s.t \begin{cases} a): \sum_{S_l \in B'} a_{l,z,t} \leq 1 & , z \in Z_k, S_k \in B \\ b): \sum_{z \in Z_k} N_{k,z,t}\, a_{k,z,t} \leq \frac{C_k T}{\varphi H} & , S_k \in B \\ c): \frac{H}{r_{k,z,t}} a_{k,z,t} \leq D & , z \in Z_k S_k \in B \end{cases} \quad (8)$$

where constraint (8a) specifies that vehicles in zone *z* must offload tasks to either the MBS or the SBS covering the zone during each time slot, constraint (8b) states that the computation load per time slot on each server must not exceed its capacity, and lastly constraint (8c) dictates that the time needed to offload task (offloading latency) must be less than the required latency bound, *D*.

D. RADIO RESOURCE SLICING (LARGE TIME-SCALE)

In order to accomplish computation load balancing with nominal task offloading switching, we optimized the task offloading decisions across the small time slots. Based on stationary task offloading policy, it is possible to further optimize the radio resource slicing between the available BSs of the vehicular network to efficiently maximize the total communication resource utilization. The average uplink transmission rate from the vehicles in zone *z* to BS can be calculated as [7]:

$$r_{k,z} = \frac{W \gamma_k f_{k,z} R_{k,z}}{M_{k,z}} \quad (9)$$

where $W = (W_m + W_s)$ represents the total radio resources, W_m is bandwidth allocated for MBS and W_s is bandwidth allocated for SBS. γ_k represents the ratio of radio resources sliced for base station S_k, (where $W\gamma_0$ is the radio resources of the MBS, and $W(1-\gamma_0)$ is the radio resource of each SBS). $R_{k,z}$ is the efficiency of the uplink spectrum averaged over *L* time slots, given by:

$$R_{k,z} = \frac{1}{L} \sum_{t=1}^{L} a_{k,z,t} \left(1 + I_{k,z,t}\right) \quad (10)$$

$f_{k,z}$ represents the average fraction of radio resources reserved for the vehicles in zone z of S_k, given by:

$$f_{k,z} = \begin{cases} \frac{M_{k,z}}{\sum_{z \in Z_k} M_{k,z} a_{k,z,}}, & \text{if } a_{k,z,} = 1 \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

where $M_{k,z}$ is the average number of the vehicles in road zone *z* of base station S_k, averaged over *L* time slots.

To determine the average network utility achieved when the tasks are offloaded from the vehicle in zone *z* of S_k, a concave logarithmic function with diminishing marginal value can be used:

$$\mathcal{U}(r_{k,z}) = \log(r_{k,z}) \quad (12)$$

To maximize the network utility for task offloading, a large time-scale RAN slicing problem can be formulated to determine the optimal ratios, γ_k , of sliced radio resources on S_k , given the optimum average fraction of radio resources, $f_{k,z}$, reserved for the vehicles in zone *z* of S_k, as function of $a_{k,z,}$ and $M_{k,z}$, to ensure QoS. The objective function is expressed as follows [7]:

$$\mathbf{P2}: \max_{\gamma_k} \sum_{S_k \in B} \sum_{z \in Z_k} M_{k,z} a_{k,z,} \log(\gamma_k)$$
$$s.t \begin{cases} a): \; a_{l,z,t}(r_{k,z} - r_{min}) \geq 0 & , z \in Z_k, S_k \in B \\ b): \; a_{l,z,t}(r_{k,z} - R_{min}) \geq 0 & , z \in Z_k, S_k \in B \\ c): \; \gamma_0 + \gamma_i = 1 & , i \in \{1,2,\dots,n\} \\ d): \; \gamma_k \in (0,1) & , k \in \{0,1,\dots,n\} \end{cases} \quad (13)$$

where r_{min} is minimum average uplink task transmission rate (in bps) used to statistically ensure the task offloading delay is within the delay bound *D*, given by [21]:

$$r_{min} = -\frac{H \log(\epsilon)}{D \log\left(1 - \frac{\log \epsilon}{\lambda D}\right)} \quad (14)$$

where ϵ is the probability bound of delay violation, and $R_{min} = \lambda H$, where λ is the average task generation rate, given as $\lambda = P_p/T$ where P_p is probability of Bernoulli distribution that was assumed to be used to generate the task at each time slot, P_p is also known as vehicle activation probability. In **P2**, constraint (a) is used to ensure the probability delay bound for task offloading is met, constraint (b) ensures the tasks receive the minimum required frame rate, constraint (c) indicate all the SBSs reuse the same portion of the sliced resources. Finally, the slicing ratio constraint (d) is used to ensure that its value is between 0 and 1. For each large timescale, MATLAB CVX toolbox can be used to solve the convex optimization problem **P2**, where different network parameters are initiated, and different task offloading policy is obtained based on solving **P1**.

IV. PROPOSED APPROACH

In this section, we present our proposed methods to solve the task offloading decision at small timescale, and radio resource slicing at large time-scale.

A. SMALLTIME-SCALE BASED DISTRIBUTED TASK OFFLOADING

The MDP formulation described in (8) can be addressed using reinforcement learning (RL) methods such as Q-learning. Q-learning is a fast and efficient RL technique. Q-

learning solves MDP by iteratively learning an optimal policy. The policy is implicitly defined by selecting actions based on the Q-values stored in the Q-table. The process involves initializing a Q-table to store state-action values, selecting actions based on exploration-exploitation trade-offs, and updating Q-values based on observed rewards and state transitions. Specifically, the Q-value for each state-action pair is iteratively adjusted using the Bellman equation, given by [22]:

$$Q(\mathcal{S}_t, \mathcal{A}_t) = (1-\eta)\, Q(\mathcal{S}_t, \mathcal{A}_t) + \eta \left[R(\mathcal{S}_t, \mathcal{A}_t) + \rho \max_{\mathcal{A}_{t+1}} Q(\mathcal{S}_{t+1}, \mathcal{A}_{t+1}) \right] \quad (15)$$

where η is the learning rate, influencing the weight of new information, and ρ is the reward discount factor, impacting the importance of future rewards. This iterative process continues until convergence, allowing the agent to learn the optimal policy for decision-making in the given environment. Then, the optimal policy is extracted from the action with the highest Q-value for each state from the Q-table. In this context, $\mathcal{S}_t$ denotes the network state characterized by environment observation $\mathcal{S}_t = \{\mathcal{N}_{k,t}, \mathcal{I}_{k,t}, \mathcal{A}_{k,t-1}\}$ at scheduling slot t, where $\mathcal{N}_{k,t} = \{N_{k,z,t}, z \in Z_k\}$, $\mathcal{I}_{k,t} = \{I_{0,z,t}, I_{k,z,t}, z \in Z_k\}$, and $\mathcal{A}_{k,t-1} = \{a_{k,z,t-1}, z \in Z_k\}$. The action is taken by agent at time slot t to determine a system reward function, R_t, formulated as:

$$R_t = -C_t - E_1 \sum_{S_l \in B} \mathbb{I}\left(\sum_{z \in Z_k} N_{k,z,t}\, a_{k,z,t} > \frac{C'_k T}{\varphi H} \right) - E_2 \sum_{S_l \in B} \sum_{z \in Z_k} \mathbb{I}\left(\frac{H}{r_{k,z,t}} a_{k,z,t} > D \right) \quad (16)$$

where E_1 and E_2 represent the penalties incurred when constraints (8b) and (8c) are violated in (**P1**), respectively. The function $\mathbb{I}(\cdot)$ is an indicator function that equals 1 if a condition is met and 0 otherwise. The variable R_t is expressed as a negative function of the computation load balancing cost, incorporating penalties for breaching computation capacity and task offloading latency constraints. For the centralized system and distributed system without threshold value (Dist-NoThrs), $C'_k = C_k$ in equation (16), however, for the distributed system with threshold, it is given as:

$$\begin{aligned} C'_k &= C_k && \text{for } k \neq 0 \\ C'_0 &= C_0 * ro_i && \text{for } k = 0 \text{ and } i \in \{1,2,\dots,K\} \end{aligned} \quad (17)$$

where ro_i represents the ratios that are feedback by the MBS for each SBS to determine how much cloud server resource is allowed for utilization in the next time slot. The proposed algorithm for task offloading can be summarized as follows:

Algorithm 1: Task offloading based on QL:

Initialize total Bandwidth, W, and Initialize bandwidth slicing ratio for MBS, γ_0
W_m = W*γ_0; W_s = W*(1-γ_0);
Initialize Zone number
Obtain number of vehicles $N_{k,z,t}$ at each zone, and their locations from dataset
Generate environment state: $SNR_{k,z,t}$, $SINR_{k,z,t}$, $a_{k,z,t-1}$.
Initialize Q-learning parameters:
- Q-table
- Learning rate, discount factor and total Episodes

```
for any time slot, t = 1: L
    // initialize action for current time slot
    Action zones = zeros(Zone number,1);
    for any episode = 1: total Episodes
        Current zone = 1;
        Total reward = 0;
        while Current zone <= Zone number
            Take Action based on greedy algorithm (exploit the Q-table or explore the environment)
            // Update the action for the current zone
            Action zones(Current zone) = Action;
            // Calculate reward
            Reward is calculated based on equation (16) with inputs ← (Old action, Action zones, SNR_{k,z,t}, SINR_{k,z,t}, N_{k,z,t}, W_m, W_s, …)
            // Update total reward
            Total reward = total reward + reward;
            // Move to next zone
            Next zone = Current zone+1;
            // Update Q table based on Bellman equation as:
            Q(current zone, Action)=(1-α) Q(current zone, Action)+ α(Reward+ γ max(Q(next zone,:) ) );
            // Update to move from current zone to next zone for next step
            Current zone = Next zone;
        end
        Save episode reward;
    end
    // Extract policy from Q-table
    [~, act] = max(Q,[],2);
    // Store the optimized action for the current time slot
    Action zones per slot (t) = act;
    // Update the old action for next time slot
    Old action = act;
    // Based on the optimized action, metrics such as computation load ratio, cost 1, cost 2, total cost and latency can be calculated and saved.
end
```

For a centralized single-agent system, where the main server can control all the network communications, and perform data processing and storage, the RL algorithm is applied only at the main cloud server, however, for our proposed distributed systems; the RL algorithm is applied at each local edge server. We discuss three distributed systems called no threshold value (Dist-NoThrs), fixed threshold value (Dist-fixed) and adaptive threshold value (Dist-adptv). In Dist-NoThrs approach, the system is unaware of the workload on the MBS server, therefore, it can increase the cost of imbalanced computation load. However, by formulating the reward function given to task offloading agent, we design two low-complexity distributed computation offloading strategies to achieve balanced and efficient offloading decisions, reduce latency and increase MBS utilization. The first strategy, Dist-fixed, equally divides the resources of the MBS server among all BSs, for instance, ro_i in equation (17) is set to $1/K$. This scheme

does not require feedback from the agents and does not require assistant from the main server. In the second strategy, Dist-adptv, we assume the main server receives information about the MBS load from each SBS at time slot t, $L_{MBS,i}$. Based on this information, the main server performs simple calculation based on fair-proportional strategy to determine the normalized MBS load, $NL_{MBS,i}$, and obtain ratio values that divide the MBS resources among the available SBS for next time slot $t+1$. A higher ratio indicates a higher MBS resource assigned to handle the load. The ratio is then feedback to the SBS for RL agents to be used in the reward function, equation (16). The proposed adaptive algorithm can be summarized as follows:

Algorithm 2: Cloud server computing resource allocation used in Dist-adptv scheme

// Set initial ratio:
$ro_i = 1/K$, where K is the number of SBSs
// Calculate total load on the MBS server from all SBSs (i) at time slot t:
$$TL_{MBS} = \sum_{i=1}^{K} L_{MBS,i}$$
// Calculate normalized MBS loads from all SBSs at time slot t:
$NL_{MBS,i} = L_{MBS,i} \,/\, TL_{MBS}$
// Obtain ratios for each local server at SBS for next time slot t+1:
$ro_i = NL_{MBS,i}$

In a distributed system where there is no load control over the MBS resources, although it offers flexibility, it can lead to overloading the MBS if all local servers offload heavily. Thus, unbalanced load distribution across the network can decrease overall performance. The fixed distributed approach may address this issue; if one SBS experiences a heavier workload, it cannot offload tasks beyond a fixed limit, which might result in underutilization of MBS resources. The adaptive distributed approach offers a good balance between flexibility and resource utilization. Hence, the local server can offload tasks to MBS as needed, potentially up to its full capacity while not exceeding load ratio of 1, thereby maximizing the overall resource utilization. The adaptive distributed approach is suitable for systems with variable and unpredictable workloads.

B. LARGE TIME-SCALE BASED RADIO RESOURCE SLICING

In the previous section, on small timescale, we optimize the task offloading decision across scheduling time slots to achieve computation load balance and minimize offloading switching costs. If the radio resources are not properly sliced, it may lead to an imbalance in the network-wide computation load, potentially violating the task transmission delay constraint given in (8c) of **P1** [7]. To ensure balanced task offloading, the radio resource slicing ratios on each BS must be optimized as in (**P2**). Hence, the two-timescale problems (**P1**) and (**P2**) should be solved together to obtain a set of optimal slicing ratios for computation load balancing so that the communication and computing resource utilization is jointly optimized. In this context, the large-time scale network slicing optimization technique is conducted at MBS main server every large timescale interval, which only requires non-privacy-sensitive parameters that were introduced in section 3.4 to solve **P2**, such as $r_{k,z}$, $a_{k,z}$, and $M_{k,z}$. To achieve that the following algorithm can be implemented to establish a joint optimization framework with algorithm 1, that is used for solving the small timescale problem. Algorithm 3 starts by initializing the slicing ratio, performing algorithm 1 to get the offloading decision using RL, then obtaining the stationary task offloading policy, and solving **P2** optimization problem by using the CVX tool to achieve optimal slicing ratio that will be used for the next large timescale slot.

Algorithm 3: Large timescale radio resource slicing

// Initialize Q-table and $a_{k,z,t-1}$ to zeros.
// Initialize total Bandwidth, W.
Initialize bandwidth slicing ratio for MBS, γ_0.
for any large scale-time, t = 1: period
  // Calculate the allocated bandwidth for MSB and SBS based on γ_0:
  W_m = W*γ_0; W_s = W*(1-γ_0);
  // Perform the small time-scale to get the offloading action $a_{k,z,t}$ (as in Algorithm 1)
  for any time slot, t = 1: L
    Apply algorithm 1 for task offloading based on QL
  end
  // Large time-scale equations:
  - Calculate uplink spectrum efficiency averaged over L scheduling slots, eq (10).
  - Calculate average fraction of radio resources, eq (11).
  - Calculate average uplink transmission rate, eq (9).
  - Solve the objective function P2 by CVX toolbox, eq (13) to get γ_0.
  - Update γ_0 for next large time-scale.
end

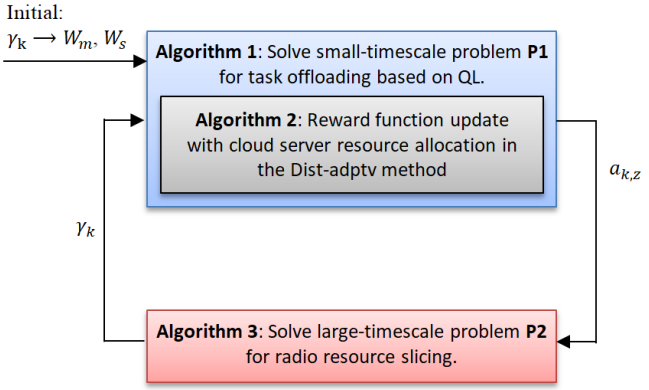


Fig. 2. The interaction between algorithms 1, 2, and 3 to form the proposed joint computation offloading and RAN slicing framework

Based on algorithm 3, the RAN slicing of the communication resources, such as bandwidth, can be dynamically allocated based on the network load requirements while adhering to constraints like delay bound and a minimum guaranteed frame rate for task transmission. The purpose of this optimization process is to maximize the

overall communication and computing resource utilization while ensuring QoS in the vehicular network. The interaction between Algorithms 1, 2 and 3 is highlighted at Fig. 2. The framework operates iteratively, with Algorithms 1 managing local task offloading decisions, Algorithm 2 handling cloud server resource allocation, and Algorithm 3 optimizing high-level radio resource slicing. This process ensures coordinated optimization across the framework's layers.

V. RESULTS AND DISCUSSION

This section presents the traffic dataset used in our analysis, simulation parameters, a comparison between traditional centralized and distributed systems, and our results and discussion for small timescale and large timescale analyses.

A. DATASET

The used dataset contains records of vehicular mobility along a three-lane highway called A6 in Madrid City [23]. It captures the position of each vehicle every 500 milliseconds along a 10-kilometer stretch of the road. The dataset includes columns for timestamp, vehicle label, vehicle position, lane number, and vehicle speed (ranging from 45 to 110 kilometers per hour). For our analysis, we focused on the first 1.5 kilometers of the highway over a period of 30 minutes. Each minute is divided into 120 time slots, each lasting 500 milliseconds. We assumed the road segment is divided into 15 zones, each with a length of 100m. We assumed there is one MBS and three SBSs, each SBS covers five zones, and all the zones are under the coverage area of the MBS. Based on this assumption, we extracted the number of vehicles and their locations at each zone for each time slot over a 30–minute period from the given dataset.

Fig. 3 illustrates the count of vehicles present within the first 1.5 km of the highway during each time slot over the 30-minute duration. The figure highlights the dynamic nature of the traffic flow, where the volume rapidly increases and then fluctuates over the time. The highest volume is recorded between the 12-minute and 19–minute marks, with the highest number of vehicles reaching 124.

B. SIMULATION

The parameters used in our simulation for vehicular network setup, load computation, and Q-learning are given in Table 2. Our simulations include two analyses. In the first part, small time-scaling is evaluated for achieving offloading balance and minimizing the offloading switching using Q-learning at each time slot, where the radio resource is fixed during all time slots. In the second part, large time-scale is evaluated at each minute for optimizing the allocated radio resource using the CVX toolbox, where the Q-learning is used again to achieve load balancing and minimize the offloading switching but based on the optimized or allocated radio resource for each minute.

TABLE 2: Simulation parameters

System parameters	Value
The uplink transmission power from each vehicle to the MBS	27dBm
The uplink transmission power from each vehicle to the SBS	23dBm
Total Bandwidth (radio resource)	20MHz
MBS height	25m
SBS height	15m
MBS distance to road	25m
SBS distance to road	15m
Number of zones	15
Zone length	100m
Path Loss model	$128.1 + 37.6\log_{10}(d)$, where d is in km
Noise power	-104 dBm
Log-normal shadowing	8 dB
Computation parameters	**Value**
Task size, H	100kbits
MBS server capacity, C_0	3.6 GHz (CPU cycles per second)
SBS server capacity, C_k	2.4 GHz (CPU cycles per second)
Computation intensity, φ	300 cycles per bit
Time slot duration, T	500 ms
Delay bound, D	500 ms
Q-Learning parameters	**Value**
Learning rate	0.1
Discount factor	0.9
Episode	200
Penalty (E1, E2)	2000, 10000
Large time-scale parameters	**Value**
Activation probability, P_p	0.6
Probability bound of delay violation, ϵ	10^{-3}

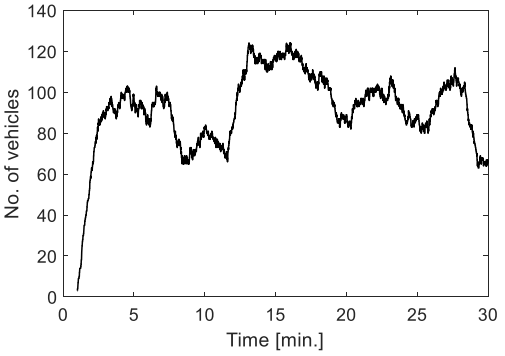


Fig.3. Number of vehicles extracted from the dataset within the first 1.5 km of the highway over 30-minute duration

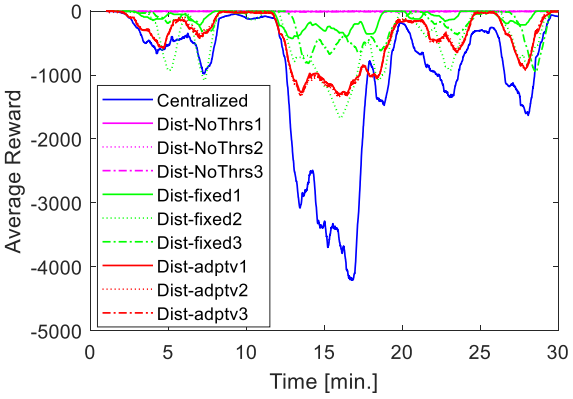


Fig.4. Average reward obtained by QL for centralized system, and distributed systems for the three SBSs, with $\gamma_0 = 0.9$, $\beta = 0.9999$ and task size of 100kbits

C. SMALL RIME-SCALE ANALYSIS

In this section, we conduct a small timescale analysis focusing on the costs associated with computation load and offloading switching. Throughout this analysis, the bandwidth slicing ratio remains fixed at a certain value. It's important to highlight the differences in approach between the centralized and distributed systems. In the centralized system, reinforcement learning (RL) is applied at the main server, and the cost function is influenced by the maximum load among the MBS, and all the SBS 1, 2, and 3. Conversely, in the distributed system, RL is applied at the local servers of SBS 1, 2, and 3. Each local server only considers its own load and the load of the MBS, rather than the combined load of all stations.

The results in this section are smoothed by using a moving average (MA) technique to remove short-term fluctuations and emphasize longer-term patterns [24]. Specifically, the results presented in this section are averaged across 120 time slots, each representing one minute.

Fig. 4 depicts a comparison of the average reward achieved by the QL at each minute (averaged over 120 time slots using the MA filter) for the three approaches. The results of the distributed systems show the average reward received by each QL agent working at each SBS. As previously explained in equation (15), negative penalties are imposed for exceeding computation capacity and task offloading latency constraints. We can see that the centralized system has greater penalties than the distributed

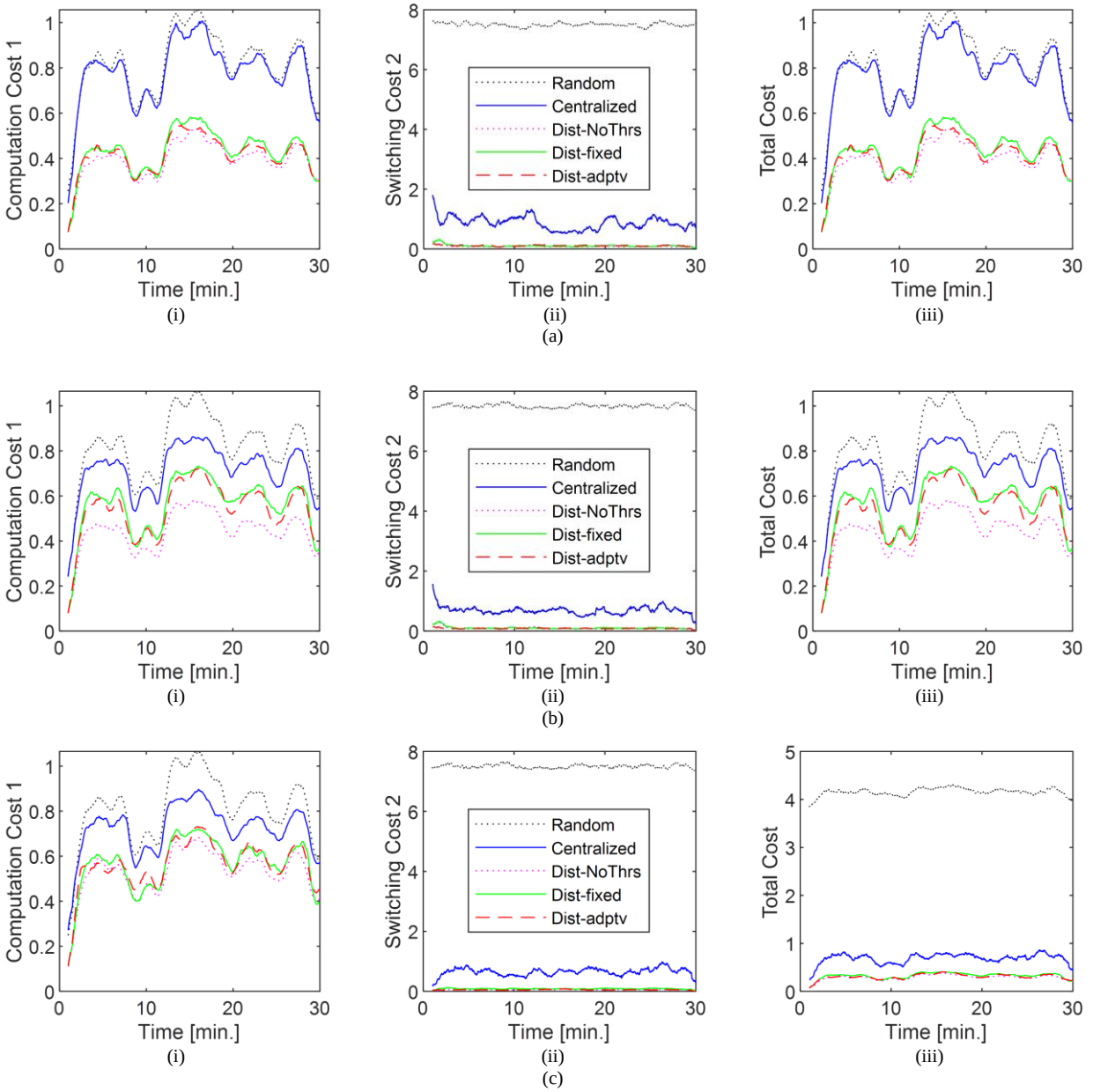


Fig.5. (i) Computation cost (cost 1), (ii) switching cost (cost 2), and (iii) total cost with (a) $\gamma_0 = 0.9, \beta = 0.9999$, (b) $\gamma_0 = 0.9, \beta = 0.9999$, and (c) $\gamma_0 = 0.6, \beta = 0.5$, at task size of 100kbits.

systems, particularly during peak vehicle traffic between 12 and 19 minutes. This indicates that the QL faced a difficult task in achieving offloading decisions that reduced computation load and offloading switches when compared with distributed systems. This is because the centralized system must optimize the offloading problem across three SBSs, whereas with distributed systems, each SBS has a QL agent that optimizes the problem independently of the other SBS. Distributed system with fixed thresholding value performs better than adjustable thresholding scheme. The distributed system without thresholding value achieves the best performance, with extremely minimal penalties given to QL agents; however, further investigation is required to evaluate other metrics.

Fig. 5 illustrates the computation load cost, offloading switching cost, and total cost based on equations (5), (6), and (7) respectively, for both centralized and distributed systems. The plots for the distributed system represent average costs across the three local servers. Notably, the overall pattern of the maximum computation load ratio (cost 1) is proportional to the number of vehicle plot given in Fig. 3. The results demonstrate that distributed systems offer lower computation load and offloading switching costs compared to centralized systems. This is attributed to the fact that while centralized systems consider the overall system load, distributed systems distribute decision-making among local servers, enabling more efficient decisions based on local load constraints only. Additionally, Fig. 5 presents costs under various scenarios, with different values for the bandwidth slicing ratio γ_0 and cost weight β. It can be seen that, when $\gamma_0 = 0.9$ and $\beta = 0.9999$ (as shown in Fig. 5(a)), the centralized computation cost is notably high, aligning with random offloading decisions. However, as γ_0 decreases to 0.6 (as depicted in Fig. 5(b)), the computation cost of the centralized system decreases accordingly. However, the computation costs of the distributed systems tend to slightly increase, but they remain lower than the centralized system. Furthermore, the distributed system demonstrates lower switching offloading costs (cost 2) compared to the centralized system. The random method tends to result in higher switching costs, given that offloading decisions are made randomly. The value of β significantly impacts the total cost; a very small β places more emphasis on cost 1, while a high β value emphasizes more on cost 2, as evident in the plots (iii) of Fig. 5. When comparing the three distributed systems, we notice slight differences in the first and third scenarios. However, in the second scenario, when $\gamma_0 = 0.6$, we observe that the distributed model with adaptive-thresholding achieves a lower cost than the fixed-thresholding method. Meanwhile, the distributed system without thresholding offers the best costs but comes with the trade-off of increasing other metrics, as we'll explore in the next figures.

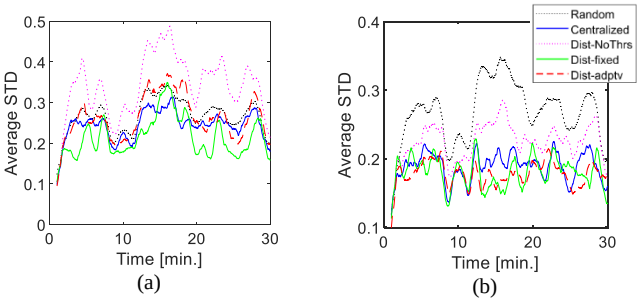


Fig.6. Average standard deviation of the loads with (a) $\gamma_0 = 0.9$, (b) $\gamma_0 = 0.6$, with $\beta = 0.9999$ and task size of 100kbits.

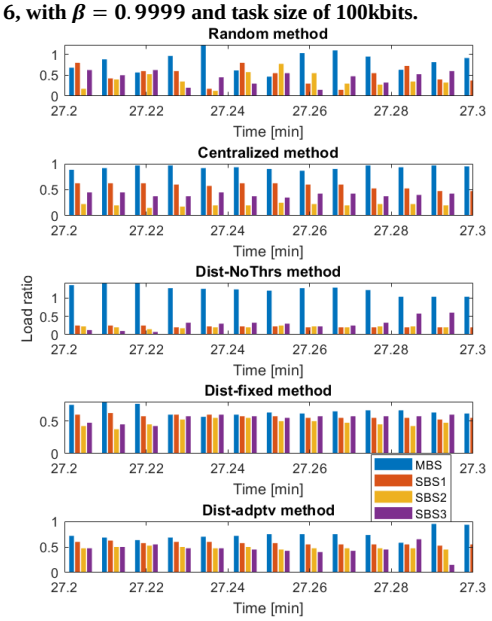
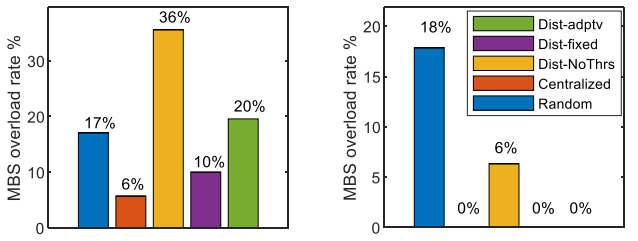


Fig.7. Example of the computation load ratio among the MBS and the three SBSs at $\gamma_0 = 0.9, \beta = 0.9999$ with task size of 100kbits for different methods

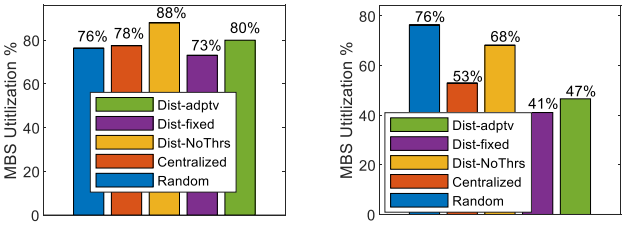
The average standard deviations (STD) of the loads are depicted in Fig 6. It is observed that the distributed system without thresholds exhibits a high standard deviation, sometimes even exceeding that of the random offloading method as in Fig. 6a. This occurs because it fails to achieve load balancing among the MBS and SBSs, occasionally excessively utilizing resources from the MBS, as exemplified in Fig. 7. Moreover, individual local servers may consume high resources from the MBS compared to others, leading to the total computation load of the MBS exceeding the desired ratio of 1. In other words, the distributed system with no thresholding doesn't take into consideration the limited resources of the main server. However, distributed systems with threshold values, such as fixed or adaptive thresholds, can effectively limit and regulate the load on the MBS to an acceptable level compared to the absence of thresholding methods, as demonstrated by the computation load bars in Fig. 7. The distributed method with a fixed threshold value yielded lower STD performance because each local server is allocated a predetermined and uniform amount of computation resources to utilize.



(a) (b)

Fig.8. MBS overload with (a) $\gamma_0 = 0.9$, (b) $\gamma_0 = 0.6$, with $\beta = 0.9999$ and task size of 100kbits.

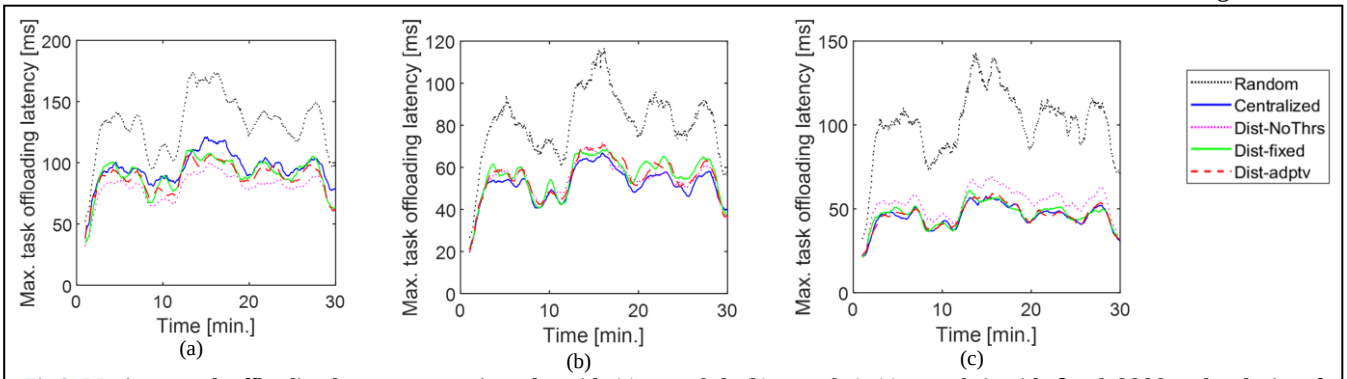
MBS overload rate is depicted in Fig. 8 for various methods, it measures how often the load on MBS exceeds ratio of 1. The centralized system performs best at an MBS bandwidth slicing ratio of $\gamma_0 = 0.9$, whereas the Dist-fixed and Dist-adptv techniques perform similarly to the centralized system in the second scenario ($\gamma_0 = 0.6$). The adaptive strategy has a little greater overload rate than the Dist-fixed method at $\gamma_0 = 0.9$, but this comes with an increase in MBS utilization, as we will see in the next discussion. The distributed system without thresholds has high overload rate at various settings; however, the distributed system with thresholding value (fixed or adaptive) was able to lower this overload rate significantly.
MBS utilization rate is illustrated in Fig. 9, quantifying how the MBS resources are utilized for different methods. It is calculated based on the MBS load ratio and averaged over the 30 periods. It is noted that, in contrast to the random decision-making system, the distributed system without thresholds displays a high MBS usage at various settings, however, as seen earlier in Fig. 8, it is highly overloads the MBS. At $\gamma_0 = 0.9$, the adaptive strategy outperforms the centralized and fixed approach by 2% and 7%, respectively. The centralized system is then outperforms the Dist-adptv and Dist-fixed systems at lower γ_0. However the adaptive approach still performs better than the Dist-fixed scheme when the bandwidth slicing ratio is reduced to γ_0 = 0.6 with 6% improvement in MBS utilization.



(a) (b)

Fig.9. MBS utilization with (a) $\gamma_0 = 0.9$, (b) $\gamma_0 = 0.6$, with $\beta = 0.9999$ and task size of 100kbits.

Fig. 10 depicts the maximum task offloading delay per time slot, taking into account the time it takes to transmit task data from vehicles to either the MBS server or the SBS server. The general pattern of the latency plots are proportional to the number of vehicle plots shown in Fig. 3. We can observe that the random offloading decision



(a) (b) (c)

Fig.9. Maximum task offloading latency versus time plot with (a) $\gamma_0 = 0.9$, (b) $\gamma_0 = 0.6$, (c) $\gamma_0 = 0.4$, with $\beta = 0.9999$ and task size of 100,000 bits.

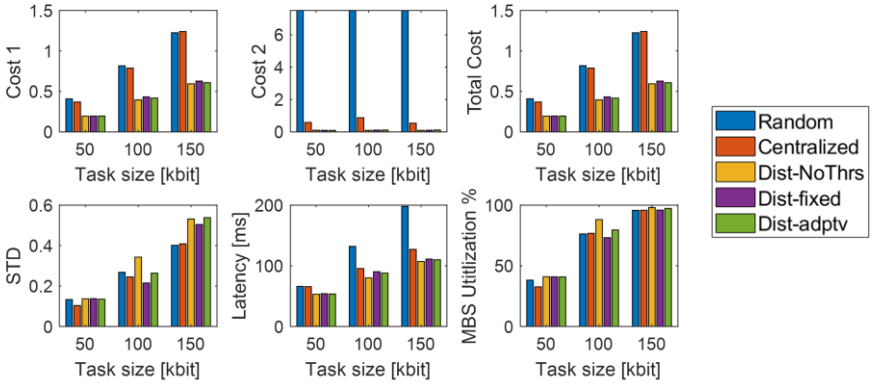


Fig.11. Overall performance metrics at $\gamma_0 = 0.9$ with different task size

approach has the maximum latency. At a bandwidth slicing ratio of $\gamma_0 = 0.9$, the centralized system has higher offloading latency than distributed systems. Reducing the slicing ratio to 0.6 improves the offloading latency of the centralized system. As γ_0 decreases to 0.4, the fixed and adaptive techniques perform similarly to the centralized system, but the distributed system without thresholding has higher latency.

In Fig. 11, we investigated the average performance of different metrics such as cost1, cost2, total cost, STD, latency, and MBS utilization at different task sizes with $\gamma_0 = 0.9$. Increasing the task size causes the cost1, total cost, STD, latency and MBS utilization to increase accordingly. The results confirm that distributed systems have lower computation and offloading switching costs than centralized system. The centralized system performs better than the distributed systems in term of STD at task size of 50kbits and 150kbits, indicating that the centralized system can achieve a more balanced load distribution compared to other methods; nevertheless, it should be mentioned that at large task sizes, such as 150 kbits, the computation cost of the centralized system exceeds the ratio of one. Moreover, this results in increasing the offloading delay of the centralized scheme. Distributed systems have lower latency performance than centralized systems at task size of 50kbits and 150 kbits. The MBS utilization of the distributed schemes is better than centralized system at task size of 50 kbits, and it is nearly fully exploited by all methods at 150 kbits task size (96%, 98%, 96%, and 97% by centralized, Dist-NoThrs, Dist-fixed, and Dist-adptv respectively). The Dist-adptv scheme shows slight MBS utilization than Dist-fixed scheme at task size of 100 kbits and 150 kbits.

D. LARGE TIME-SCALE ANALYSIS

In this section, we discuss large time-scale analysis. We evaluated the centralized and distributed schemes by solving **P2**, with the initial slicing ratio of MBS's radio resource (γ_0) set to 0.9 and for the SBS is set to 0.1 ($W_m = 19$MHz and $W_s = 2$MHz).

Fig. 12 (a) illustrates the radio slicing ratio γ_0 (right y-axis) and total cost (including load computation and offloading switching) (left y-axis) over 30 minutes after applying the large time-scale algorithm for the four approaches. The figure demonstrates the adaptation of the centralized, Dist-fixed and Dist-adptv schemes to the network load variations due to vehicles mobility change over time. Most methods start adapting to the load variations after 3 minutes. We observe that the slicing ratio γ_0 is adjusted every minute, and it is proportional to computation load. The maximum and minimum γ_0 values obtained by the four schemes are [0.8, 0.54], [0.67, 0.5], [0.77, 0.52] and [0.76, 0.47] for centralized, Dist-NoThrs, Dist-fixed and Dist-adptv, respectively. The centralized,

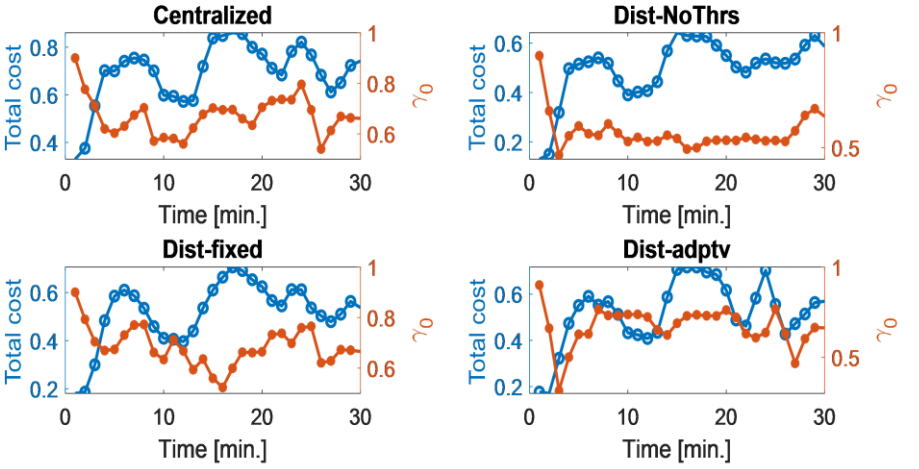


Fig.12. Adaptation of the radio resource slicing ratio γ_0 with total cost

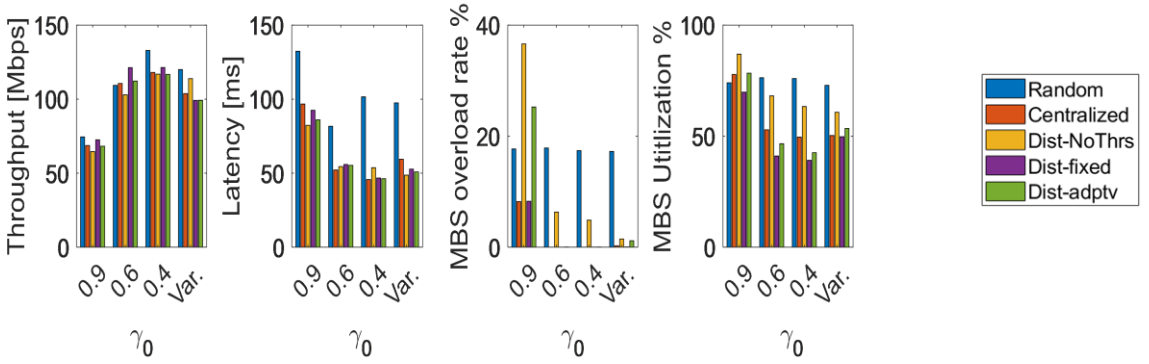


Fig.13. Performance metrics with MBS radio slicing ratio, γ_0, for small time-scale (with fixed γ_0) and large time-scale (variable ratio γ_0) at task size of 100kbits

Dist-fixed and Dist-adptv shows good correlation with computation load compared to Dist-NoThrs scheme.

In Fig. 13, we investigated the impact of decreasing the MBS bandwidth slicing ratio, γ_0, from 0.9 to 0.6 and 0.4 (fixed slicing ratio all time), and using variable slicing ratios (var.) (dynamic ratio changes over large time-scale, i.e., one minute in our case) obtained after applying the large time-scale algorithm. As the fixed slicing ratio decreases, we observe a significant increase in throughput, and a notable decrease in latency, MBS overload rate, and MBS utilization. Reducing the slicing ratio from 0.9 to 0.4 markedly boosts throughput by 49 Mbps, 53 Mbps, 49 Mbps, and 49 Mbps, while substantially reducing offloading delay by 50 ms, 28 ms, 45 ms and 40 ms for centralized, Dist-NoThrs, Dist-fixed and Dist-adptv, respectively. At $\gamma_0 = 0.4$, centralized, Dist-fixed and Dist-adptv schemes achieved almost equivalent latency. For variable slicing ratios case, throughput increases compared to fixed slicing ratio $\gamma_0 = 0.9$, but is slightly less than when $\gamma_0 = 0.6$ and 0.4, specifically, increasing by 35 Mbps, 49 Mbps, 26 Mbps and 31 Mbps in throughput and decreasing by 38 ms, 33 ms, 41 ms and 35 ms in latency, for centralized, Dist-NoThrs, Dist-fixed and Dist-adptv, respectively. The slight improvement in variable slicing ratios case compared to the fixed ratio occurs because, in the variable ratios scenario, the slicing ratios fluctuate between 0.8 and 0.5 based on network demands. It is also observed that the MBS overload rate for the distributed system without a threshold is significantly higher compared to other counterparts at $\gamma_0 = 0.9$. The centralized system and Dist-fixed exhibit low MBS overload, with a lower MBS usage percentage for the Dist-fixed scheme at $\gamma_0 = 0.9$. However, the MBS overload rates of centralized, Dist-fixed and Dist-adptv drop dramatically to 0% by reducing the slicing ratio to 0.6 and 0.4. On the other hand, the Dist-NoThrs shows overload rate of 6% and 5% at slicing ratios of 0.6 and 0.4, respectively. For variable ratios, the Dist-NoThrs overload reduces to 1.5%, while the centralized and Dist-adptv schemes reduce to 0.26% and 1.2%, respectively, while the Dist-fixed reduces to 0%. Finally, the MBS utilization for all schemes exceeds 70% at $\gamma_0 = 0.9$, however, this percentage decreases by reducing the MBS radio resource from 18MHz to 12MHz and 8MHz. The centralized system maintains similar MBS utilization at $\gamma_0 = 0.4$ and with variable ratios, but, the Dist-fixed and Dist-adptv schemes demonstrate better MBS utilization in variable ratio scenario compare to $\gamma_0 = 0.6$ and 0.4 cases. The Dist-adptv scheme achieves 3% better utilization than the centralized and Dist-fixed schemes. Our investigation reveals that using variable MBS slicing ratios strategy enhances throughput and reduces latency effectively. The Dist-adptv method achieves the best MBS utilization and the Dist-fixed method demonstrating the lowest overload rates, highlighting their effectiveness in optimizing task offloading in dynamic vehicular networks.

VI. CONCLUSION

In this work, we studied jointly two time-scale task offloading and RAN slicing in vehicular networks using both centralized and distributed approaches. We proposed two distributed schemes for the allocation of cloud server resources: Dist-fixed and Dist-adptv. In the small time-scale, task offloading was optimized using the Q-learning method, considering constraints such as server capacity and offloading latency. The objective was to achieve load balancing with minimal offloading variations among edge and cloud servers. The analysis of the small time-scale technique showed that distributed systems demonstrated lower balanced computation load costs, offloading switching cost and task completion latency compared to the centralized system at different task sizes. The centralized system showed better standard deviation (STD) at task sizes of 50 kbits and 150 kbits, while Dist-fixed achieved better STD at a task size of 100 kbits. Distributed systems exhibited good MBS utilization compared with the centralized system at lower task sizes and nearly equivalent MBS utilization at higher task sizes. A large time-scale RAN slicing strategy was further investigated to dynamically adjust the radio resource slicing ratios proportionally to the network's load to maximize the overall radio resource utilization. The analysis of the large times-scale technique showed that the RAN slicing ratios for all schemes, except Dist-NoThrs, effectively adapted to dynamic network loads caused by vehicle mobility. A significant observation was the substantial performance improvement observed with distributed systems such as Dist-fixed and Dist-adptv. For instance, the throughput of Dist-adptv scheme increased by 31 Mbps, and latency decreased by 35 ms compared to the fixed slicing ratio of $\gamma_0 = 0.9$. Additionally, Dist-adptv achieved up to 3% better MBS utilization than both Dist-fixed and centralized systems, while the Dist-fixed scheme reduced overload rates to 0% with task size of 100 kbits, demonstrating the effectiveness of the distributed schemes in optimizing task offloading in dynamic vehicular network environments. In future work, we aim to explore the proposed scheme in more secure approach such as federated reinforcement learning, and compare its performance with other RL techniques to further validate its effectiveness.

APPENDIX

TABLE 3 Main symbols with their definitions

Symbol	Definition
S_k	The k th base station (BS) under consideration
$a_{k,z,t}$	Task offloading decision in zone z of base station, S_k, at time slot, t
$N_{k,z,t}$	Number of vehicles in zone z of S_k at time slot, t
W_k	The available bandwidth for k th BS
$G_{k,z,t}$	Uplink channel gain from vehicles within zone z to S_k at slot t
$I_{k,z,t}$	Uplink SNR/SINR for vehicles in zone z of S_k at slot t
$r_{k,z,t}$	Uplink transmission rate
P_k	Uplink transmission power for vehicles under S_k
$\alpha_{k,t}$	Small-scale Rayleigh fading component under S_k at slot t
σ^2	Average background noise power
$L_{k,t}$	Computation load associated with S_k at slot t
φ	Computation intensity
H	Computing task size
C_k	Computation capacity on the server connected to S_k
T	The duration of a task scheduling slot

D	Task offloading latency requirement
C_t	Total cost of computation load balancing at slot t
$a_{k,z}$	Stationary task offloading in zone z of S_k
$r_{k,z}$	Average uplink transmission rate from the vehicles in zone z to S_k
γ_k	Ratio of sliced radio resources on S_k
$f_{k,z}$	Average fraction of radio resources reserved for vehicles in zone z of S_k
$R_{k,z}$	Uplink spectrum averaged over L time slots
$M_{k,z}$	Average number of vehicles in zone z of S_k

REFERENCES

[1] Talebkhah, M., Sali, A., Khodamoradi, V., Khodadadi, T., & Gordan, M. "Task offloading for edge-IoV networks in the industry 4.0 era and beyond: A high-level view." Engineering Science and Technology, an International Journal, 54, 2024, pp 101699.https://doi.org/10.1016/j.jestch.2024.101699.

[2] X. Shen et al., "AI-Assisted Network-Slicing Based Next-Generation Wireless Networks," in IEEE Open Journal of Vehicular Technology, vol. 1, pp. 45-66, 2020, doi: 10.1109/OJVT.2020.2965100.

[3] Zhongyu Wang, Tiejun Lv, Zheng Chang,"Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing", Computer Networks, Vol. 205, 2022, doi.org/10.1016/j.comnet.2021.108732.

[4] Yang, Kun, et al. "A novel hierarchical distributed vehicular edge computing framework for supporting intelligent driving." Ad Hoc Networks, 153 (2024): 103343. https://doi.org/10.1016/j.adhoc.2023.103343

[5] S. Li, S. Lin, L. Cai, W. Li and G. Zhu, "Joint Resource Allocation and Computation Offloading With Time-Varying Fading Channel in Vehicular Edge Computing," in IEEE Transactions on Vehicular Technology, vol. 69, no. 3, pp. 3384-3398, March 2020, doi: 10.1109/TVT.2020.2967882.

[6] L. Geng, H. Zhao, J. Wang, A. Kaushik, S. Yuan and W. Feng, "Deep-Reinforcement-Learning-Based Distributed Computation Offloading in Vehicular Edge Computing Networks," in IEEE Internet of Things Journal, vol. 10, no. 14, pp. 12416-12433, 15 July 15, 2023, doi: 10.1109/JIOT.2023.3247013.

[7] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang and X. Shen, "Joint RAN Slicing and Computation Offloading for Autonomous Vehicular Networks: A Learning-Assisted Hierarchical Approach," in IEEE Open Journal of Vehicular Technology, vol. 2, pp. 272-288, 2021, doi: 10.1109/OJVT.2021.3089083.

[8] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, Y. Yang, Efficient dependent task offloading for multiple applications in MEC-cloud system, IEEE Trans. Mob. Comput. (2021) 1, http://dx.doi.org/10.1109/TMC.2021.3119200.

[9] M. Tang, V.W. Wong, Deep reinforcement learning for task offloading in mobile edge computing systems, IEEE Trans. Mob. Comput. 21 (6) (2022) 1985–1997, http://dx.doi.org/10.1109/TMC.2020.3036871.

[10] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, Y. Zhang, Deep learning empowered task offloading for mobile edge computing in urban informatics, IEEE Internet Things J. 6 (5) (2019) 7635–7647, http://dx.doi.org/10.1109/ JIOT.2019.2903191.

[11] X. Wang, Z. Ning, S. Guo, Multi-agent imitation learning for pervasive edge computing: a decentralized computation offloading algorithm, IEEE Trans. Parallel Distrib. Syst. 32 (2) (2021) 411–425, http://dx.doi.org/10.1109/TPDS.2020. 3023936.

[12] K. Jiang, H. Zhou, D. Li, X. Liu and S. Xu, "A Q-learning based Method for Energy-Efficient Computation Offloading in Mobile Edge Computing," 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 2020, pp. 1-7, doi: 10.1109/ICCCN49398.2020.9209738.

[13] B. Dab, N. Aitsaadi and R. Langar, "Q-Learning Algorithm for Joint Computation Offloading and Resource Allocation in Edge Cloud," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 2019, pp. 45-52.

[14] Z. Gao, W. Hao, Z. Han and S. Yang, "Q-Learning-Based Task Offloading and Resources Optimization for a Collaborative Computing System," in IEEE Access, vol. 8, pp. 149011-149024, 2020, doi: 10.1109/ACCESS.2020.3015993.

[15] [F. Jiang, W. Liu, J. Wang and X. Liu, "Q-Learning Based Task Offloading and Resource Allocation Scheme for Internet of Vehicles," 2020 IEEE/CIC International Conference on Communications in China (ICCC), Chongqing, China, 2020, pp. 460-465, doi: 10.1109/ICCC49849.2020.9238925.

[16] Stuart Russel and Peter Norvig, "Artificial Intelligence: A Modern Approach", 4th US Edition, Pearson, 2021.

[17] A. S. Kumar, L. Zhao and X. Fernando, "Task Offloading and Resource Allocation in Vehicular Networks: A Lyapunov-Based Deep Reinforcement Learning Approach," in IEEE Transactions on Vehicular Technology, vol. 72, no. 10, pp. 13360-13373, Oct. 2023, doi: 10.1109/TVT.2023.3271613.

[18] N. Sharma, A. Ghosh, R. Misra and S. K. Das, "Deep Meta Q-Learning Based Multi-Task Offloading in Edge-Cloud Systems," in IEEE Transactions on Mobile Computing, vol. 23, no. 4, pp. 2583-2598, April 2024, doi: 10.1109/TMC.2023.3264901.

[19] J. Huang, J. Wan, B. Lv, Q. Ye and Y. Chen, "Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning," in IEEE Systems Journal, vol. 17, no. 2, pp. 2500-2511, June 2023, doi: 10.1109/JSYST.2023.3249217.

[20] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," IEEE J. Sel. Areas Commun., vol. 37, no. 10, pp. 2282–2292, Oct. 2019

[21] 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.

[22] C. J. Watkins and P. Dayan, "Q-learning," Mach. Learn., vol. 8, no. 3/4, pp. 279–292, 1992.

[23] M. Gramaglia, O. Trullols-Cruces, D. Naboulsi, M. Fiore and M. Calderon, "Mobility and connectivity in highway vehicular networks: A case study in Madrid", Computer Communications, vol. 78, pp. 28-44, 2016, doi.org/10.1016/j.comcom.2015.10.014.

[24] H. -H. Chang, H. Chen, J. Zhang and L. Liu, "Decentralized Deep Reinforcement Learning Meets Mobility Load Balancing," in IEEE/ACM Transactions on Networking, vol. 31, no. 2, pp. 473-484, April 2023, doi: 10.1109/TNET.2022.3176528.