Road Side Unit-Assisted Learning-Based Partial Task Offloading for Vehicular Edge Computing System

Song Li, *Member, IEEE,* Weibin Sun, Qiang Ni, *Senior Member, IEEE,* and Yanjing Sun, *Member, IEEE,*

***Abstract*—The rapid development of vehicular networks creates diverse ultra-low latency constrained and computation-intensive applications, which bring challenges to both communication and computation capabilities of the vehicles and their transmission. By offloading tasks to the edge servers or vehicles in the neighbourhood, vehicular edge computing (VEC) provides a cost-efficient solution to this problem. However, the channel state information and network structure in the vehicular network varies fast because of the inherent mobility of vehicle nodes, which brings an extra challenge to task offloading. To address this challenge, we formulate the task offloading in vehicular network as a multi-armed bandit (MAB) problem and propose a novel road side unit (RSU)-assisted learning-based partial task offloading (RALPTO) algorithm. The algorithm enables vehicle nodes to learn the delay performance of the service provider while offloading tasks. Specifically, the RSU could assist the learning process by sharing the learning information among vehicle nodes, which improves the adaptability of the algorithm to the time-varying networks. Simulation results demonstrate that the proposed algorithm achieves lower delay and better learning performance compared with the benchmark algorithms.**

***Index Terms*—Multi-armed bandit, online learning, task offloading, vehicular edge computing;**

I. Introduction

With the rapid development of wireless technologies and vehicular networks, some resource-intensive applications such as autonomous driving, augmented reality (AR), and video streaming emerge in the Internet of Vehicular (IoV) [1], [2], [3]. These advanced applications make the vehicular networks safer, more intelligent, and more convenient at the expense of considerable computation, communication, and storage resources. Although vehicles are equipped with more and more computation and storage resources nowadays, they still cannot guarantee the quality of service (QoS) requirements of the high-level autonomous driving applications and on-board infotainment services with ultra-low latency constraints.

Mobile edge computing (MEC) is a paradigm that could provide low-latency and high-reliability computing services for devices through task offloading [4]. By applying MEC into vehicular networks, vehicular edge computing (VEC) provides a promising solution to meet the requirements of the computation-intensive and delay-sensitive applications in the vehicular networks [5], [6]. Considering a tremendous number of vehicles in the urban vehicular networks, many underutilized vehicular resources can be contributed to vehicular networks for computation services providing [7]. What's more, the edge server deployed in the road side unit (RSU) could also provide computing and storage resources at a close range. However, the channel state and network topologies of the VEC network are time-varying due to the mobility of vehicle nodes. The uncertainty of network conditions brings huge challenges for optimal task offloading performing, such as offloading server selection and task offloading ratio management.

Based on the condition mentioned above, the task offloading decision can be modeled as a multi-armed bandit (MAB) problem [8], regarding each computation service provider as a bandit arm with different and unknown expected payoff. The objective of the MAB problem is to select the optimal sequence of arms to maximize the expected total reward. One of the key issues in the MAB problem is how to balance the trade-off between exploitation and exploration strategy. This problem is even more complicated in the vehicular networks since the candidate arm set changes randomly because of the high mobility of the vehicle nodes, while the traditional MAB problems assume each arm remains unchanged.

Motivated by the above challenge, a novel RSU-assisted learning-based partial task offloading (RALPTO) algorithm based on MAB theory is proposed in this paper to cope with uncertainty caused by the highly dynamic vehicle nodes in VEC networks. The objective of our algorithm is to minimize the average task offloading delay over a period of time. In this paper, motivated by the concept of Vechicle as a Resource (VaaR) [7], we focus on task offloading among vehicles to improve the resource utilization of the vehicular networks. Since the workload of vehicle nodes is fluctuant, we classified the vehicle nodes into two categories: task vehicles (TaVs) and service vehicles (SeVs). TaVs are the vehicles that have task offloading requirements because of its limitted computation capability, and SeVs are the vehicles that can provide surplus resources to execute tasks. The strategy of our algorithm is learning while offloading, which means that TaVs can learn the service capability of SeVs according to its historical offloading delay performance. What's more, RSU is used to assist the sharing of learning information among TaVs, which improves the efficiency of the algorithm. The main contributions of this paper are summarized as follows:

This work was supported by the Fundamental Research Funds for the Central Universities (2020ZDPYMS26). (Corresponding author: Yanjing Sun.)

Song Li, Weibin Sun and Yanjing Sun are with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China (E-mail: lisong@cumt.edu.cn; wbsun@cumt.edu.cn; yjsun@cumt.edu.cn).

Qiang Ni is with the School of Computing and Communications, Lancaster University, LA1 4YW Lancaster, U.K. (e-mail: q.ni@lancaster.ac.uk).

- We investigate a task offloading scenario in the vehicular network, where the channel state information and network topology change over time because of the inherent mobility of vehicle nodes. Partial task offloading is adopted in the VEC system, which allows TaVs to offload a part of the task to SeVs. Thus the computation is performed in parallel on both TaVs and SeVs by partial task offloading.
- We propose a novel RSU-assisted learning-based partial task offloading algorithm based on MAB theory which enables TaVs to learn the delay performance of SeVs while offloading tasks. The algorithm performs in a distributed manner, with low computational complexity and no requirements for global network state information. To enhance the adaptability of the algorithm to the uncertainty in vehicular network, we design a special utility function that allows the algorithm to realize the trade-off between exploration and exploitation strategy according to the contextual information such as task workload and the appearance time of SeVs. To improve the learning efficiceny of the algorithm, we propose a novel user cooperation scheme which allow TaVs share the learned information among each other through RSU.
- We compare the performance of RALPTO algorithm with the state-of-the-art learning-based task offloading algorithms. The numerical results verify that the proposed algorithm can achieve a significant performance improvement in terms of latency reduction.

The rest of this paper is organized as follows. In Section II, related work to this paper are discussed. In Section III, we introduce the system model and the formulation of the problem. The algorithm is proposed in Section IV. Simulation results are presented in Section V and the whole paper is concluded in Section VI.

II. Related Work

By integrates the computing resources of the network edge to provide computing services for vehicles through task offloading, VEC has the potential to meet the ever-growing computation requirements of the advanced vehicular applications. Thus, the task offloading problem in the vehicular network has drawn researchers' attention [9].

Some existing studies focus on the task offloading schemes in VEC aiming to optimize the task delay [9] [10], system cost/overhead [11] [12] or other utility functions [13] - [17]. The authors of [9] proposed a software-defined networking (SDN) based load-balancing task offloading scheme to minimize the processing delay of the vehicles' computation tasks, considering load-balancing of the vehicular edge server. Liu $et\ al.$ [10] studied the task offloading problem from a matching perspective and propose pricing-based matching algorithms for the task offloading to optimize the total network delay. The authors in [11] proposed a mobility-aware task offloading scheme to minimize the system costs, in which the offloading time selection, communication, and computing resource allocations are joint performed optimally. The problem of vehicle-to-everything (V2X) offloading and resource allocation in an SDN-assisted MEC network is investigated in [12] and the optimal offloading decision, transmission power control, subchannel assignment, and computing resource allocation scheme is proposed. Dai $et\ al.$ [13] considered task offloading problem in a multi-user multi-server VEC system and focused on maximizing the system utility by integrating load balancing with offloading. Hui $et\ al.$ [14] developed a game theoretic scheme for collaborative vehicular task offloading in HetNets and proposed a two-stage task offloading mechanism to promote the cooperation among participants with the target of improving the task completion rate and the utilities of the participants. Considering reduce the computation resources for both vehicles and resource providers, the authors of [15] proposed a hybrid intelligent optimization algorithm based on partheno genetic algorithm which performs server selection for tasks. The authors of [16] proposed an efficient computation offloading strategy by using a contract theoretic approach to enhance the utility of service provider in the vehicular network. A hierarchical cloud-based VEC offloading framework are proposed in [17] and an optimal multilevel offloading scheme by using a Stackelberg game theoretic approach are designed.

However, the above researches all assumed that the global information of the vehicular network is available, which ignored the influence of the high mobility of vehicle nodes on channel state information and network topology. Thus these algorithms are not robust in practical vehicular networks with highly dynamic vehicle nodes. To address the challenge posed by the uncertain vehicular networks, some researchers resort to the method of reinforcement learning to adapt to the dynamic environment. Zhang $et\ al.$ [18] formulated the vehicular task offloading as a mortal MAB problem and proposed an online algorithm to optimize the node selection strategy by exploiting the contextual information. To reduce the influence of the mobility of vehicular servers, Liu $et\ al.$ [19] proposed a fluctuation-aware learning-based computation offloading algorithm to improve the delay performance. To improve the delay performance and service reliability of the VEC system, Wang $et\ al.$ [20] exploit the redundancy of computing resources by introducing task replication technique. Sun $et\ al.$ [21] further proposed an adaptive learning-based task offloading algorithm to achieve a lower delay performance, which could adapt to the dynamic VEC environment without frequent state information exchange.

Most of task offloading solutions above mainly focus on binary task offloading, where the computing task is either executed locally or offloaded to the edge server entirely. Since many applications in the IoV can be partitioned, such as virtual reality (VR) and AR applications [4], partial task offloading provides an more efficient solution for these seperatable applications, in which each computing task can be partitioned into two or more sub-tasks with some sub-tasks executed locally and others offloaded to the edge server. Garaali $et\ al$ [22] studied the joint computation offloading and resource allocation problem in MEC networks and proposed a multi-agent reinforcement learning solution. Tuong $et\ al$ [23] investigated a joint partial computation offloading and channel allocation problem in a NOMA-assisted MEC network considering the dynamic network environments with time-varying channels,

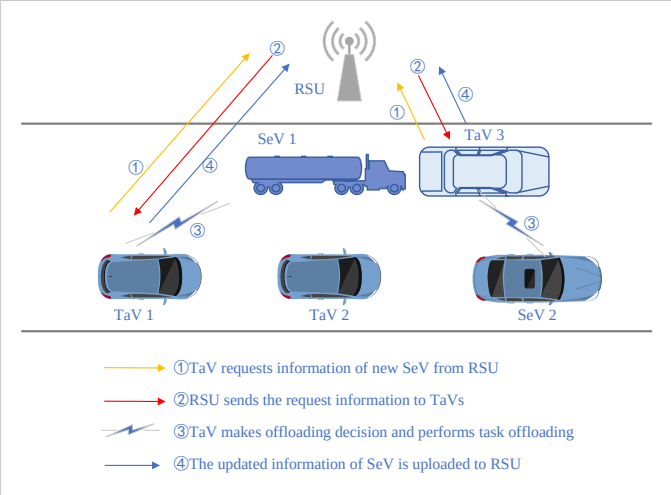

Fig. 1. Architecture of the VEC system.

and proposed a reinforcement learning algorithm to minimized the weighted sum of consumed energy and latency.

According to the learning-based task offloading scheme above, each vehicle makes task offloading desicions based on its own learning information from the environment. In other words, these vehicles perform reinforcement learning in a non-cooperative manner. Different from existing reinforcement learning based task offloading, we propose a novel RSU-assisted learning-based partial task offloading algorithm in this paper. A reinforcement learning method based on MAB theory is adopted in the proposed algorithm to learn the dynamic VEC network state information. Specifically, the learned information is shared among TaVs with the assistance of RSU, which improves the efficiency of the learning process.

III. System Model

In this section, we will introduce the VEC system model considered in this paper first. After that, the delay of each partial task offloading procedure will be analyzed.

A. Network Architecture

The architecture of the VEC network considered in this paper is shown in Fig. 1, which consists of multiple vehicles and one RSU. Since the centralized task offloading management is not practical in the VEC network with uncertain network state information, we considered a distributed task offloading manner which adopts vehicle-to-vehicle (V2V) offloading. In the system model, TaVs generate tasks and offload it to nearby SeVs for execution. Note that to be TaV or SeV depends on the remaining resources of the vehicle, which is not fixed during the trip. The RSU is distributed along the road and communicates with vehicles by vehicle-to-infrastructure (V2I) communication. In the VEC network, SeVs may switch among different TaVs during times because of the high mobility of the vehicles, which reduces the learning efficiency of TaVs. To cope with this challenge, a RSU-assisted user cooperation scheme is proposed in the system model, which allow TaVs to share the learned information among each other. Specifically, TaVs will upload the learned information to the RSU at the end of each time period. At the begining of next time period, RSU will broadcast the aggregated learned information to all TaVs in coverage.

To ensure the quality of connection between TaVs and SeVs, TaVs select the SeVs with the same driving direction as candidates within its communication range. The driving states information such as driving direction and driving speed can be exchanged through vehicular communication protocols such as dedicated short-range communication (DSRC) standard [24]. As shown in Fig. 1, TaV1 has three candidate SeVs within its communication range.

The notations used in this article are listed in Table I.

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

Notation	Definition
t	index of the time period
$N(t)$	the candidate SeV set of TaV at time period t
φ_t	index of the task at time period t
L_t	size of the input data of task φ_t
C_t	computation intensity of computation intensity of task φ_t (in CPU cycles per bit)
$\lambda_{n,t}$	offloading proportion of task φ_t for SeV n
$r^o_{n,t}$	achievalbe uplink transmission rate
$r^b_{n,t}$	achievalbe downlink transmission rate
$f_{l,t}$	CPU frequency of TaV
$f^s_{n,t}$	CPU frequency of SeV n at time period t
$d_o(n,t)$	the transmission delay for task offloading
$d_b(n,t)$	the transmission delay for result feedback
$d_{l,c}(n,t)$	execution delay for local computing
$d_{s,c}(n,t)$	execution delay for offloading sub-task computing
$u_{n,t}$	the sum execution delay for offloading one bit offloaded sub-task
a_t	the offloading decision of TaV at time period t
R_T	the cumulated learning regret till time period T
$k_{n,t}$	the number of times that SeV n has been selected up to time period t
t_n	the occurrence time of SeV n
$v_{n,t}$	the variance of the bit offloading delay of SeV n

B. Task Execution Procedure

Since a distributed task offloading manner is adopted in the system model, we will focus on one TaV and model the task execution process in this subsection. There are four steps for partial task offloading in the system model, including task partition, task offloading, task execution, and result feedback. The process of the partial task offloading is shown in Fig. 2.

1) Task Partition: Considering a discrete-time VEC system where the time is divided into T discrete time periods. The set of candidate SeVs for TaV in time period t is denoted as $N(t)$. Note that $N(t)$ is time-varying and unknown in prior because of the mobility of vehicles. TaV will select a SeV $n \in N(t)$ for task offloading at every time period. The computation task on TaV in the time period t is denoted as $\varphi_t = [L_t, C_t]$, where L_t is the size of the input data for task φ_t (in bits), C_t denotes the computation intensity of the task (in CPU cycles per bit). We assume the tasks in the system model can be partitioned into two sub-tasks at any proportion, and each sub-task can be processed at local TaV or SeV. Denoting $\lambda_{n,t} \in [0,1]$ as the offloading ratio, which means the offloading proportion of

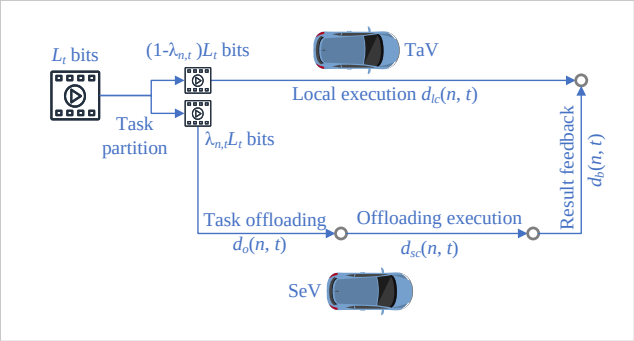

Fig. 2. The process of partial task offloading

task φ_t for SeV n. So the size of the sub-task for offloading can be expressed as $\lambda_{n,t}L_t$, and the size of the sub-task for local computing is $(1-\lambda_{n,t})L_t$.

2) Task Offloading: After selecting a SeV n for task offloading, TaV will split task φ_t in to two sub-tasks for local computing and offloading respectively. Note that the process of local computing and task offloading are conducted simultaneously, which can significantly reduce the task processing delay. By denoting P as the transmission power, B as the channel bandwidth, and $h^o_{n,t}$ as the uplink channel condition between TaV and SeV n, the achievable uplink transmission rate $r^o_{n,t}$ can be written as:

$$r^o_{n,t} = B\log\left(1+\frac{Ph^o_{n,t}}{\sigma^2}\right) \tag{1}$$

where σ^2 represents the noise power. The co-channel interference is avoided since we adopt the orthogonal channel allocation [24].

And the transmission delay for task offloading $d_o(n,t)$ can be expressed as:

$$d_o(n,t) = \frac{\lambda_{n,t}L_t}{r^o_{n,t}} \tag{2}$$

3) Task Execution: Task execution consists of two parts: local sub-task execution and offloading sub-task execution. By denoting f^l_t as the CPU frequency of TaV, the execution delay for local computing $d_{lc}(n,t)$ can be written as:

$$d_{lc}(n,t) = \frac{C_t(1-\lambda_{n,t})L_t}{f^l_t} \tag{3}$$

Denoting the allocated CPU frequency of SeV n as $f^s_{n,t}$, the execution delay for offloading sub-task execution $d_{sc}(n,t)$ can be given by:

$$d_{sc}(n,t) = \frac{C_t\lambda_{n,t}L_t}{f^s_{n,t}} \tag{4}$$

4) Result Feedback: The computation result of the offloaded sub-task will be transferred back from SeV to TaV. Similar to the offloading process, denoting $h^b_{n,t}$ as the downlink channel condition between SeV n and TaV, the achievable downlink transmission rate $r^b_{n,t}$ can be written as:

$$r^b_{n,t} = B\log\left(1+\frac{Ph^b_{n,t}}{\sigma^2}\right) \tag{5}$$

Note that the size of the output result is much smaller compared to the input data in most cases, denoting $L^{'}_t$ as the size of the computation result (in bits), the transmission delay for result feedback $d_b(n,t)$ can be expressed as:

$$d_b(n,t) = \frac{L^{'}_t}{r^b_{n,t}} \tag{6}$$

Since the local computing and task offloading are executed in parallel, the task completion delay $d_{sum}(n,t)$ can be written as

$$d_{sum}(n,t) = \max\left(d_{lc}(n,t), d_{off}(n,t)\right) \tag{7}$$

where $d_{off}(n,t) = d_o(n,t) + d_{sc}(n,t) + d_b(n,t)$.

C. Problem Formulation

Our goal is to minimize the average task execution delay during T time periods by optimizing the offloading decisions and offloading ratio. The problem can be formulated as:

$$P1: \min_{\lambda_{a_t,t}, a_1,\ldots,a_T} \frac{1}{T}\sum_{t=1}^{T} d_{sum}(a_t,t) \tag{8}$$

where optimization variable $a_t \in N(t)$ represents the offloading decision of TaV at time period t.

Intuitively, a_t should be the SeV with the best service capability among the candidate SeV set $N(t)$. To measure the service capability of each SeV, bit offloading delay $u_{n,t}$ is defined, which represents the sum execution delay for offloading one bit offloaded sub-task. $u_{n,t}$ can be expressed as follows:

$$u_{n,t} = \frac{d_{off}(n,t)}{\lambda_{n,t}L_t} \tag{9}$$

From (9), we can observe that the value of $u_{n,t}$ depends on the channel condition between TaV and SeV n and the allocating CPU frequency of SeV n. If $h^o_{n,t}$, $h^b_{n,t}$ and $f^s_{n,t}$ are known for TaV before offloading, it's easy for TaV to choose the optimal SeV according to

$$a_t = \arg\min_{n\in N(t)} u_{n,t} \tag{10}$$

However, due to the inherent mobility of vehicle nodes, the network state information of the VEC system varies fast across time, which is hard to model or predict. In addition, the computation capacity offered by SeV also fluctuates over time. To obtain the accurate network state information, TaV needs to exchange information frequently with SeVs, causing heavy signaling overhead. Therefore, it is hard for TaV to make the optimal offloading decision due to the time-varying network state information in the VEC system. To address this challenge, we designed a novel online learning algorithm based on MAB theory which enables TaV to learn the service capabilities of nearby SeVs. This algorithm will be introduced in detail in Section IV.

IV. RSU-Assist Learning-Based Partial Task Offloading Algorithm

In this section, we propose a novel online learning algorithm based on MAB theory to reduce the task execution delay by guiding TaV to choose the optimal SeV. The performance and complexity of the algorithm are also analyzed in this section.

According to MAB theory, a bandit with K arms is modeled, where each arm has a different and unknown expected payoff. The goal of the MAB algorithm is to maximize the cumulative reward. Considering each SeV as an arm of the bandit which has unknown service capability and TaV as the decision maker, task offloading problem in the VEC system can be modeled as a MAB problem. However, traditional MAB problem assumes that the arms remain unchanged over time, while candidate SeV set is time-varying because of the mobility of the vehicles. Thus, the traditional MAB algorithms such as UCB1 and UCB2 can not be used to solve the task offloading problems in the VEC system.

In this work, we propose a RSU-assisted learning-based partial task offloading algorithm, which allows TaVs to learn the service capability of SeVs while offloading. Note that the service capability is measured by the bit offloading delay $u_{n,t}$. In the learning process, the variation of the SeV set will reduce the learning efficiency. To address this problem, we propose a novel RSU-assisted mechanism which allows TaVs to share the service capability of SeVs they learned with the assistance of RSU. At the end of each time period, TaVs will update the service capability of SeV it selects and upload this information to RSU. The updated information of SeVs will be broadcast to all TaVs by RSU at the beginning of the next period.

At the beginning of each time period, TaV first detects whether a new SeV appears in the SeV set. If a new SeV appears and there is no record about its service capability in RSU, TaV will offload a task to it once. Note that TaV knows nothing about the new appear SeV, so the offloading ratio is set as a predetermined value $\lambda^{'}$. If TaV has the prior information about SeV according to RSU or itself, an appropriate offloading ratio $\lambda_{n,t}$ could be calculated in the follows. Since $d_{lc}(n,t)$ and $d_{off}(n,t)$ in $P1$ are monotonically increasing and decreasing with $\lambda_{n,t}$ respectively, the optimal value of $\lambda_{n,t}$ can be given when $d_{lc}(n,t) = d_{off}(n,t)$. Combining (3) and (9), the optimal task offloading proportion ${\lambda_{n,t}}^*$ can be expressed as:

$$ {\lambda_{n,t}}^* = \frac{C_t}{C_t + f_l^t \overline{u}_{n,t-1}} \tag{11} $$

where $\overline{u}_{n,t-1}$ denotes the observed average bit offloading delay of SeV n till time period $t-1$.

One of the classic problems which MAB theory focuses on is how to balance the exploration and exploitation tradeoff in the learning process. Exploitation means selecting the optimal node based on known information, while exploration means exploring the unknown nodes. The exploitation strategy may result in missing the nodes with better benefits, while the exploration strategy may result in unexpected losses. Considering the tradeoff between exploration and exploitation, a utility function is defined to evaluate the service capability of each SeV as follows:

$$ \hat{u}_{n,t} = \overline{u}_{n,t-1} - \sqrt{\frac{\beta\left(1-\lambda_{n,t}\right)\tilde{L}_t v_{n,t} \ln\left(t-t_n\right)}{k_{n,t-1}}} \tag{12} $$

where parameter β denotes the weight factor, $k_{n,t}$ denotes the number of times that SeV n has been selected up to time period t, and t_n records the occurrence time of SeV n. $\tilde{L}_t$ refers to the normalized value of L_t, $v_{n,t}$ denotes the variance of the bit offloading delay of SeV n. The expression of $\tilde{L}_t$ and $v_{n,t}$ will be introduced later.

The first part of the utility function (12) is the empirical bit offloading delay of SeV n, and the second part is a padding function, which is used to balance the tradeoff between exploration and exploitation. Inspired by the volatile UCB (VUCB) algorithm [25] and UCB1-tuned algorithm [26], the size of the normalized offloaded sub-task $\left(1-\lambda_{n,t}\right)\tilde{L}_t$, SeV occurrence time t_n, the variance of bit offloading delay $v_{n,t}$, and the number of selected times of SeV n $k_{n,t}$ have been considered in the padding function. To be specific, weight factor β is used to adjust the tendency of the algorithm to exploration strategy. The size of the normalized offloaded sub-task can also be regarded as a weight factor of exploitation. Intuitively, if the size of the offloaded sub-task is small, TaV could execute exploration strategy safely without worrying about the large delay. And if the size of the offloaded sub-task is large, TaV should choose the exploitation strategy to reduce the risk of large delay. The normalized $\tilde{L}_t$ can be expressed as:

$$ \tilde{L}_t = \frac{L_t - L_{min}}{L_{max} - L_{min}} \tag{13} $$

where L_{max} and L_{min} denote the maximum and minimum value of the task load, respectively.

What's more, the rest part of padding function $\sqrt{\frac{v_{n,t}\ln(t-t_n)}{k_{n,t-1}}}$ encourages TaV to explore the SeVs with a large variance of bit offloading delay and fewer times of selection, since this means that the SeV has not been fully explored. According to the UCB1-tuned algorithm [27], $v_{n,t}$ can be expressed as:

$$ \begin{aligned} v_{n,t} = \min\Bigg(&\frac{1}{4}, \sum_{\tau=1}^{t-1}\frac{1}{k_{n,t-1}} k_{n,\tau} {u_{n,\tau}}^2 - {u_{n,t-1}}^2 \\ &+ \sqrt{\frac{\beta(1-\lambda_{n,t})\tilde{L}_t v_{n,t}\ln(t-t_n)}{k_{n,t-1}}}\Bigg) \end{aligned} \tag{14} $$

where $u_{\tau,n}$ is the bit offloading delay of SeV n at time period τ observed by TaV.

Then, TaV will choose the SeV with minimal utility function to offload sub-task, which can be expressed as:

$$ a_t = \arg\min_{n \in N(t)} \hat{u}_{n,t} \tag{15} $$

After making the offloading decision, TaV will perform task offloading and execution according to the offloading decision and observe the offloading delay of SeV a_t upon result feedback, and $\overline{u}_{a_t,t}$, $k_{a_t,t}$ will be updated. Finally,

Algorithm 1 RSU-assisted Learning-based Partial Task Offloading (RALPTO) Algorithm

Input: $\lambda^{'}$, $\tilde{L}_t$, f_t^l
Output: $a_1, ..., a_T$, ${\lambda_{a_t,t}}^*$

1: **for** t=1, ..., T **do**
2: **if** New SeV n appears in $N(t)$ **then**
3: **if** The average bit offloading delay of SeV n $\overline{u}_{n,t}$ can be learned from RSU **then**
4: Record $\overline{u}_{n,t}$ according to RSU, $k_{n,t}=1$, $t_n=t$, calculate $\lambda^*_{n,t}$ according to (11).
5: **else** $a_t=n$, $\lambda_{n,t}=\lambda^{'}$, offload the task to SeV a_t and calculate $u_{n,t}$ according to (9), $k_{n,t}=1$, $t_n=t$.
6: **end if**
7: **end if**
8: **if** $(a_t==0)$ **then**
9: Calculate ${\lambda_{n,t}}^*$ for each candidate SeV $n\in N(t)$ according to (11).
10: Calculate the utility function of each candidate SeV $n\in N(t)$ according to (12).
11: Choose the optimal SeV a_t according to (15) and perform task offloading decision.
12: Calculate the bit offloading delay $u_{a_t,t}$ according to (9).
13: **end if**
14: Update $\overline{u}_{a_t,t}\leftarrow\frac{\overline{u}_{a_t,t-1}k_{a_t,t-1}+u_{a_t,t}}{k_{a_t,t-1}+1}$.
15: Update $k_{a_t,t}\leftarrow k_{a_t,t-1}+1$.
16: Upload $\overline{u}_{a_t,t}$ to RSU .
17: **end for**

the updated $\overline{u}_{a_t,t}$ will be uploaded to RSU. The proposed RALPTO algorithm is shown in Algorithm 1.

The outline of Algorithm 1 is summarized as follows:

1) At the beginning of each time period, each TaV first calculates the optimal task offloading proportion according to the observed average offloading delay of each candidate SeV. Next, the service capability of each candidate SeV is derived and the optimal SeV with the best service capability is selected to perform task offloading. Then the bit offloading delay of the selected SeV is then calculated (line 8-line 12).

2) Specifically, if a new SeV appears in the candidate SeV set, TaV first requests its observed average offloading delay from RSU. If there is no record of the new SeV in RSU, TaV will choose the new SeV to perform offloading and the offloading ratio is set as a predetermined value (line 2-line 7).

3) At last, the service capability and the selected times of the selected SeV are updated and uploaded to RSU (line 14-line 16).

V. Performance Analysis

We further analyze the complexity and performance of the RALPTO algorithm in this section. In the proposed RALPTO algorithm, the computational complexity of computing the optimal offloading ratio ${\lambda_{n,t}}^*$ and utility function $\hat{u}_{n,t}$ of all SeV in Line 9 and Line 10 is both $O(N)$, where $N=|N(t)|$ is the size of the SeV set at time period t. The decision-making requires $O(1)$ computational complexity. Therefore, the total complexity of the RALPTO algorithm can be derived as $O(TN)$.

To analyze the performance of the proposed RALPTO algorithm, we adopt the learning regret of task execution delay as the performance metric, which is widely used in MAB theory [28]. Then, the theoretical derivation will prove that the proposed algorithm has a sublinear performance compared to the globally optimal policy.

Learning regret denotes the delay loss between the proposed algorithm and the global optimal policy, which reflects how close the decision made by the algorithm is to the optimal solution. For theoretical analysis, we defined an epoch as the interval during which SeV sets remain identical. Assumes there are B epochs during T time periods, and we denote N_b as the SeV set of the bth epoch. The first and last time period of bth epoch is denoted as t_b and $t_b^{'}$. Let $u_n=E[u_{n,t}]$ denote the mean bit offloading delay of SeV n and $a_b^*=\arg\min_{n\in N(t)}u_n$ denote the index of the optimal SeV at bth epoch with the optimal bit offloading delay μ_b^* , the cumulated learning regret till time period T can be expressed as

$$R_T=\sum_{b=1}^{B}E\left[\sum_{t=t_b}^{t_b^{'}}x_t\left(u_{n,t}-\mu_b^*\right)\right] \tag{16}$$

where x_t denotes the size of offloaded sub-task to SeV n at time period t. Note that x_t varies across the time in practical, for simplicity, we assume the task load remains identical. The research in [21] proves that the conclusion still holds without this assumption.

Denote $\Delta_{n,b}=u_{n,t}-\mu_b^*$ as the learning regret that caused by choosing SeV $n\in N_b$, R_T can be further expressed as

$$\begin{aligned}R_T&=\sum_{b=1}^{B}E\left[x_t\sum_{n\neq a_b^*}k_{n,t}\Delta_{n,b}\right]\\&=x_t\Delta_{n,b}\sum_{b=1}^{B}E\left[\sum_{n\neq a_b^*}k_{n,t}\right]\end{aligned} \tag{17}$$

Lemma 1: The upper bound of the learning regret of the proposed RALPTO algorithm in a given time horizon T can be expressed as:

$$R_T\leq x_t\sum_{b=1}^{B}\left[\sum_{n\neq a_b^*}\left(\frac{8\ln(T)}{\Delta_{n,b}}+\left(1+\frac{\pi^2}{3}\right)\Delta_{n,b}\right)\right] \tag{18}$$

Proof : See Appendix A.

Lemma 1 shows that the learning regret of our proposed RALPTO algorithm is governed by $O(\ln T)$, which achieves sublinear deviation compared to the optimal solution.

VI. Simulation Results

In this section, numerical simulation experiments are performed in synthetic scenario using MATLAB to verify the performance of the proposed RALPTO algorithm. In the simulation 3 TaVs and 30 SeVs are considered and the distance between SeVs and TaVs is randomly distributed within 50m

TABLE II
PARAMETERS TABLE

Parameter	Value
Transmission power P	0.1W
Noise power σ^2	10^{-13}W
Channel bandwidth B	10MHz
Computation intensity of task C_t	10^3 cycles/bit
Weight factor β	0.2
Fixed offloading ratio $\lambda^{'}$	0.5
Maximum CPU frequency of SeV f^s_{max}	5GHz
Maximum CPU frequency of TaV f^l_{max}	3GHz
Communication range of TaV	50m

and changes randomly from -5 m to 5 m in each period. One RSU is deployed to broadcast the service capability of SeVs to TaVs. In the simulation, the wireless transmission can be modeled as $h^o_{n,t} = h^b_{n,t} = A_0 l^{-\varphi}$, where $A_0 = -17.8dB$, l denotes the distance between the sender and receiver, and φ denotes the path loss factor, which randomly distributed from [2, 4] [28]. The data size of the input task L_t uniformly distributed within [5, 8] Mbits. The remaining parameters are listed in Table II. Noticed that the CPU frequency of SeV is randomly distributed from 20% to 50% of the maximum value.

The performance of the proposed RALPTO is validated by comparing with the following benchmark algorithms:

1) **UCB** [27]. UCB chooses SeV to perform task offloading based on the expected offloading delay and upper confidence index. The padding function in UCB is $\sqrt{\frac{\beta \ln t}{k_{t-1,n}}}$, which does not consider the impact of the size of input task and changeable SeV set.

2) **VUCB** [29]. VUCB is similar to UCB, except that the padding function in VUCB is $\sqrt{\frac{\beta \ln(t-t_n)}{k_{t-1,n}}}$, which take the impact of the dynamic SeV set into consideration.

3) **Genie-aided policy**. According to Genie-aided policy, each TaV knows the accurate network state information and makes optimal offloading decisions according to (10) in every time period. Note that genie-aided policy provides the lower bound of delay performance among all reinforcement learning-based algorithms.

4) **Random policy**. In random policy, each TaV randomly selects a SeV for task offloading in each time period.

5) **Full offloading RALPTO**. Full offloading RALPTO is a special case of the RALPTO algorithm which adopts binary task offloading policy instead of partial task offloading policy. In the other word, in the full offloading RALPTO algorithm, TaV will offload the whole task to the selected SeV.

6) **No RSU-assisted RALPTO**. No RSU-assisted RALPTO is another special case of RALPTO algorithm, in which there are no information exchanges among each TaVs. TaVs store the service capability of SeV locally.

The average delay of each algorithm is shown in Fig. 3. It can be observed that the proposed RALPTO algorithm outperforms UCB, VUCB and random algorithms. This is due to the partial task offloading and RSU-assist mechanism adopted in RALPTO improve the adaptability of TaVs to the time-varying VEC environment. Both UCB and VUCB outperform random algorithms, indicating that MAB algorithms

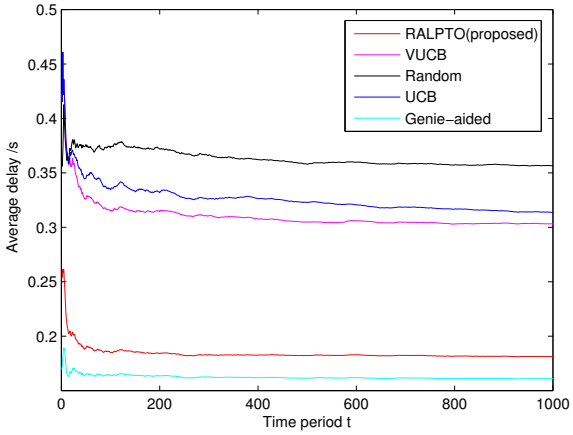

Fig. 3. Average delay of RALPTO algorithm and benchmarks

can effectively adapt to dynamic VEC network environments. The average delay of UCB and VUCB during the initialization phase is relatively high since the exploration dominates the learning process at the beginning. In addition, the VUCB has lower delay than the UCB, demonstrating that considering the impact of dynamic SeV sets in VEC networks is beneficial for improving algorithm performance.

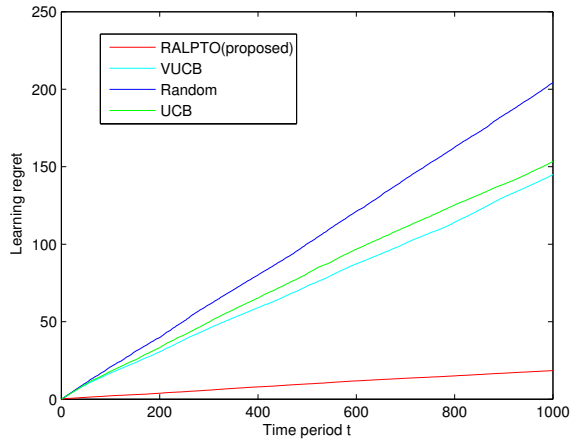

Fig. 4. Learning regret of RALPTO algorithm and benchmarks

The average learning regret of each algorithm is shown in Fig. 4, from which we can observe that the proposed RALPTO algorithm decreases the learning regret by 85% from VUCB. Moreover, the learning regret of RALPTO increases sub-linearly with time period t, which indicates that the proposed algorithm can asymptotically converge to the optimal policy. The learning regret of UCB and VUCB is close, both significantly lower than the random algorithm. The learning regret performance of VUCB is slightly better than that of UCB, which is consistent with the result in Fig. 3.

Fig. 5 presents the ratio of optimal selections of each algorithm, which is the proportion of the number of times each algorithm makes the best decision to the total number of decisions. We can observe that the RALPTO algorithm has

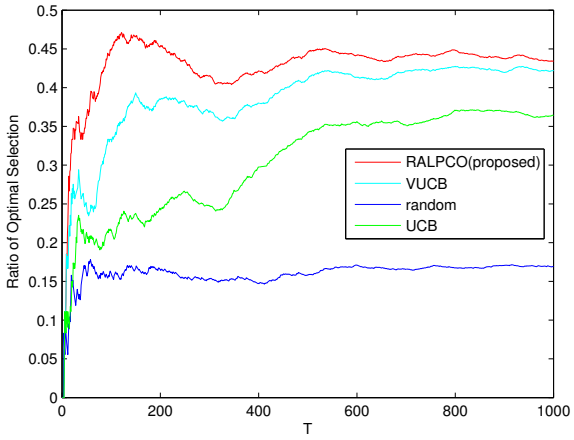

Fig. 5. Ratio of optimal selections of RALPTO algorithm and benchmarks

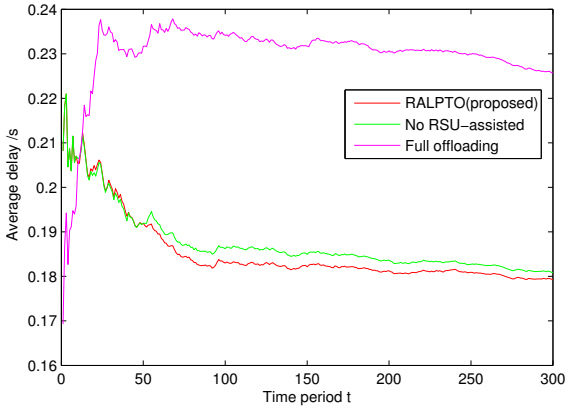

Fig. 6. The delay performance of RALPTO algorithm with different mechanism

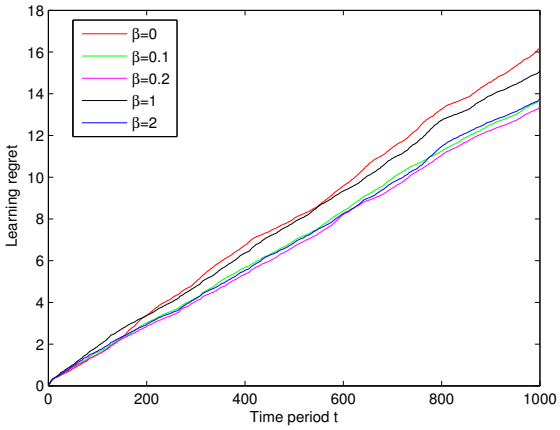

Fig. 7. Cumulated learning regret of RALPTO with different β

the best performance compared to the other benchmark algorithms. Noticed that due to the highly dynamic environment in the simulation, the ratio of optimal selection of all algorithms did not exceed 50%. Combining the analysis of Fig. 3 and Fig. 4, it can be concluded that the RALPTO algorithms still have good performance with less than 45% of the ratio of optimal selection, because suboptimal solutions are obtained which are not far from the best decision. The ratio of optimal selection of VUCB and UCB reach 40% and 35% at around 1000 time periods respectively, and that of random algorithm achieves slightly above 15%.

Fig. 6 evaluates the impact of partial task offloading and RSU-assist mechanism on the delay performance. To explore the impact of these mechanisms on the adaptability of the algorithm to the dynamic environment, we compared the RALPTO algorithm with the Full offloading RALPTO algorithm and No RSU-assisted RALPTO algorithm in Fig. 6. We can observe that the Full offloading RALPTO algorithm has a larger delay compared with the other algorithms, illustrating that partial task offloading can significantly reduce the task execution delay. This is due to the fact that the partial task offloading can make use of the local computation capacity of TaV, and the task can be executed by TaV and SeV in parallel. Noticed that the RALPTO algorithm outperforms the No RSU-assist RALPTO algorithm since the beginning of each epoch, which illustrates that the RSU-assist mechanism is helpful to improve the performance of the RALPTO algorithm in the dynamic vehicular environment.

Fig. 7 presents the impact of the weight factor β on the learning regret, which represents the tendency of the RALPTO algorithm to the exploration strategy. We can observe that during the first 180 periods, the performance of $\beta = 0$ and $\beta = 0.2$ are very close. Then, the learning regret of $\beta = 0$ increases rapidly over the rest of 800 time periods. The learning regret of RALPTO with $\beta = 1$ is slightly lower than that with $\beta = 0$ after 200 time periods. The regrets of RALPTO with $\beta = 2$ and $\beta = 0.1$ are very close and is slightly higher than that with $\beta = 0.2$. Since $\beta = 0$ means the algorithm adopts a pure exploitation strategy, thus the algorithm may stick into a sub-optimal solution for a long time and cause an increase of long-term learning regret when $\beta = 0$. In addition, according to Fig. 7, the learning regret is lowest when $\beta = 0.2$ under our settings.

VII. Conclusion

In this paper, we proposed a novel RSU-assisted learning-based task offloading algorithm (RALPTO) for the task offloading problem in the VEC network to minimize the average offloading delay. In the proposed algorithm, partial task offloading greatly reduces the task execution delay by parallel computing, which makes full use of the local computing resources. The utility function can effectively realize the trade-off between exploitation and exploration strategy, which enhances the adaptability of the algorithm to the dynamic vehicular network. What's more, the TaVs can share the learning information with the assist of RSU, which further improves the efficiency of the algorithm. Simulation results have shown its advantages over existing algorithms. For further research, the task offloading problem considering both vehicles and RSUs as computation servers in the VEC network can be further investigated as an open issue.

APPENDIX A
PROOF OF LEMMA 1

From (17) we know that R_T is mainly determined by $E[k_{n,t}]$. For any predicate ζ, we define $\{\zeta\} = 1$ if the predicate is true and $\{\zeta\} = 0$ if the predicate is false. Let l be an arbitrary positive integer, we have:

$$\begin{aligned}
k_{n,t} &\le 1 + \sum_{t=2}^{T} \{a_t = n\} \\
&\le l + \sum_{t=2}^{T} \{a_t = n, k_{n,t-1} \ge l\} \\
&\le l + \sum_{t=2}^{T} \{\overline{u}_{n,t-1} - I_{n,t-1} \le {u_{t-1}}^* - {I_{t-1}}^*, k_{n,t-1 \ge l}\} \\
&\le l + \sum_{t=2}^{T} \left\{ \min_{l<s_n<t} \overline{u}_{n,s_n} - I_{n,s_n} \le \max_{0<s<t} {u_s}^* - I_s \right\} \\
&\le l + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_n=l}^{t-1} \{\overline{u}_{n,s_n} - I_{n,s_n} \le {u_s}^* - I_s\}
\end{aligned} \tag{19}$$

where $I_{n,t-1}$ denotes the padding function of $\hat{u}_{n,t-1}$.

Intuitively, if SeV n is chosen in time period t, $\hat{u}_{n,t} \le \hat{u}_t^*$ must holds, which is also revealed in the above formula

$$\overline{u}_{n,s_n} - I_{n,s_n} \le {u_s}^* - I_s \tag{20}$$

Equation (20) holds only when at least one of the following three inequalities are true:

$$u_n - u^* < 2I_{n,s_n} \tag{21}$$

$$u^* + I_{n,s_n} \le {\overline{u}_n}^* \tag{22}$$

$$\overline{u}_n + I_{n,s_n} \le u_n \tag{23}$$

Denote $\beta = 2$, from (21) we can obtain:

$$k_{n,t-1} \le \frac{8\ln(t-t_n)}{{\Delta_n}^2} \tag{24}$$

where $\Delta_n = u_n - u^*$.

Denote $l = \left\lceil \frac{8\ln(t-t_n)}{{\Delta_n}^2} \right\rceil$, where $[x]$ means rounding down to x. $\forall s_n > l$ in (21), we can obtain that

$$u_n - u^* - 2I_{n,s_n} \ge u_n - u^* - \Delta_n = 0 \tag{25}$$

Thus it is proven that (21) is false. According to Chernoff-Hoeffding inequality [30], we can obtain the probability that (22) and (23) hold:

$$P\left({\overline{u}_n}^* - u^* \le I_{n,s_n}\right) \le e^{-4\ln(t-t_n)} \tag{26}$$

$$P\left(u_n - \overline{u}_n \ge I_{n,s_n}\right) \le e^{-4\ln(t-t_n)} \tag{27}$$

Thus,

$$\begin{aligned}
E\left(k_{T,n}\right) &\le \left\lceil \frac{8\ln(t-t_n)}{{\Delta_n}^2} \right\rceil + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_n=l}^{t-1} \{ \\
&\quad P({\overline{u}_n}^* - u^* \le I_{n,s_n}) + P\left(u_n - \overline{u}_n \ge I_{n,s_n}\right)\} \\
&\le \frac{8\ln(T)}{{\Delta_n}^2} + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_n=\left\lceil \frac{8\ln(T)}{{\Delta_n}^2} \right\rceil}^{t-1} \left\{2t^{-4}\right\} \\
&\le \frac{8\ln(T)}{{\Delta_n}^2} + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_n=1}^{t-1} \left\{2t^{-4}\right\} \\
&\le \frac{8\ln(T)}{{\Delta_n}^2} + 1 + \frac{\pi^2}{3}
\end{aligned} \tag{28}$$

Combining (28) into (17), the Lemma 1 is proved.

REFERENCES

[1] E. Ahmed and H. Gharavi, "Cooperative vehicular networking: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 996–1014, 2018.

[2] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[3] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] S. Bitam, A. Mellouk, and S. Zeadally, "Vanet-cloud: a generic cloud computing model for vehicular ad hoc networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 96–102, 2015.

[6] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[7] S. Abdelhamid, H. S. Hassanein, and G. Takahara, "Vehicle as a resource (VaaR)," *IEEE Network*, vol. 29, no. 1, pp. 12–17, 2015.

[8] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM journal on computing*, vol. 32, no. 1, pp. 48–77, 2002.

[9] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[10] P. Liu, J. Li, and Z. Sun, "Matching-based task offloading for vehicular edge computing," *IEEE Access*, vol. 7, pp. 27628–27640, 2019.

[11] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26652–26664, 2019.

[12] H. Zhang, Z. Wang, and K. Liu, "V2x offloading and resource allocation in sdn-assisted mec-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.

[13] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[14] Y. Hui, Z. Su, T. H. Luan, C. Li, G. Mao, and W. Wu, "A game theoretic scheme for collaborative vehicular task offloading in 5g hetnets," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16044–16056, 2020.

[15] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, "Joint optimization of computation offloading and task scheduling in vehicular edge computing networks," *IEEE Access*, vol. 8, pp. 10466–10477, 2020.

[16] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 288–294, 2016.

[17] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2017.

[18] R. Zhang, P. Cheng, Z. Chen, S. Liu, Y. Li, and B. Vucetic, "Online learning enabled task offloading for vehicular edge computing," *IEEE Wireless Communications Letters*, vol. 9, no. 7, pp. 928–932, 2020.

[19] Z. Liu, X. Zhang, J. Zhang, D. Tang, and X. Tao, "Learning based fluctuation-aware computation offloading for vehicular edge computing system," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, 2020.

[20] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, "Task replication for vehicular edge computing: A combinatorial multi-armed bandit based approach," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2018.

[21] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061–3074, 2019.

[22] R. Garaali, C. Chaieb, W. Ajib, and M. Afif, "Learning-based task offloading for mobile edge computing," in *ICC 2022 - IEEE International Conference on Communications*, pp. 1659–1664, 2022.

[23] V. D. Tuong, T. P. Truong, T.-V. Nguyen, W. Noh, and S. Cho, "Partial computation offloading in noma-assisted mobile-edge computing systems using deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13196–13208, 2021.

[24] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, 2011.

[25] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Bandit algorithms for social network queries," in *2013 International Conference on Social Computing*, pp. 148–153, 2013.

[26] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[27] D. Bouneffouf, "Finite-time analysis of the multi-armed bandit problem with known trend," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2543–2549, 2016.

[28] M. Abdulla, E. Steinmetz, and H. Wymeersch, "Vehicle-to-vehicle communications with urban intersection path loss models," in *2016 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, 2016.

[29] Z. Bnaya, R. Puzis, R. Stern, and A. Felner, "Social network search as a volatile multi-armed bandit problem," *Human*, vol. 2, no. 2, p. 84, 2013.

[30] H. Chernoff *et al.*, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *The Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952.

Song Li (Member, IEEE) received the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He was a Visiting Scholar with Lancaster University from September 2016 to Auguest 2017. He is currently an Associate Professor with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. His current research interests include B5G/6G wireless networks, multi-access edge computing, industrial Internet of Things and cyber-physical system.

Weibin Sun received the B.Sc. and M.Sc. degrees from the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China, in 2019 and 2022, respectively. He is currently a researcher with Zoomlion Heavy Industry Science And Technology Co., Ltd. His current research interests include mobile edge computing, B5G/6G wireless networks and the Internet of Things.

Qiang Ni (Senior Member, IEEE) received the Ph.D. degree in engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1999. He is currently a Professor with the School of Computing and Communications, Lancaster University, Lancaster, U.K. He has authored or co-authored over 300 papers. He was an IEEE 802.11 Wireless Standard Working Group Voting Member and a contributor to various IEEE wireless standards. His research interests include the future-generation communications and networking, including green communications and networking, millimeter-wave wireless communications, cognitive radio network systems, 5G/6G, software-defined networks, cloud networks, edge computing, dispersed computing, Internet of Things, cyber physical systems, artificial intelligence/machine learning, and vehicular networks.

Yanjing Sun (Member, IEEE) received the Ph.D. degree in information and communication engineering from the China University of Mining and Technology, in 2008. He has been a Professor with the School of Information and Control Engineering, China University of Mining and Technology, since 2012, where he is currently the Director of the Network and Information Center. He is also a Council Member of the Jiangsu Institute of Electronics and a member of Information Technology Working Committee of the China Safety Production Association. His current research interests include wireless communication, the Internet of Things, embedded real-time system, wireless sensor networks, and cycber-physical system.