Adaptive Learning-Based Task Offloading for Vehicular Edge Computing Systems

Yuxuan Sun, *Student Member, IEEE*, Xueying Guo, *Member, IEEE*, Jinhui Song, Sheng Zhou, *Member, IEEE*, Zhiyuan Jiang, *Member, IEEE*, Xin Liu, *Fellow, IEEE*, and Zhisheng Niu, *Fellow, IEEE*

***Abstract*—The vehicular edge computing system integrates the computing resources of vehicles, and provides computing services for other vehicles and pedestrians with task offloading. However, the vehicular task offloading environment is dynamic and uncertain, with fast varying network topologies, wireless channel states, and computing workloads. These uncertainties bring extra challenges to task offloading. In this paper, we consider the task offloading among vehicles, and propose a solution that enables vehicles to learn the offloading delay performance of their neighboring vehicles while offloading computation tasks. We design an adaptive learning based task offloading (ALTO) algorithm based on the multi-armed bandit theory, in order to minimize the average offloading delay. ALTO works in a distributed manner without requiring frequent state exchange, and is augmented with input-awareness and occurrence-awareness to adapt to the dynamic environment. The proposed algorithm is proved to have a sublinear learning regret. Extensive simulations are carried out under both synthetic scenario and realistic highway scenario, and results illustrate that the proposed algorithm achieves low delay performance, and decreases the average delay up to 30% compared with the existing upper confidence bound based learning algorithm.**

***Index Terms*—Vehicular edge computing, task offloading, online learning, multi-armed bandit.**

I. Introduction

BY DEPLOYING computing resources at the edge of the network, mobile edge computing (MEC) can provide low-latency, high-reliability computing services for mobile devices [2], [3]. A major problem in MEC is how to perform *task offloading*, i.e., whether or not to offload each task, and how to manage radio and computing resources to execute tasks, which has been widely investigated recently, see surveys [4]–[6] and technical papers [7]–[9].

To support autonomous driving and a vast variety of on-board infotainment services, vehicles are equipped with substantial computing and storage resources. It is forecast that each self-driving car will have computing power of 10^6 dhrystone million instructions executed per second (DMIPS) in the near future[10], which is tens of times that of the current laptops. Vehicles and infrastructures like road side units (RSUs) can contribute their computing resources to the network. This forms the Vehicular Edge Computing (VEC) system [11]–[13], that can process computation tasks from vehicular driving systems, on-board mobile devices and pedestrians for various applications.

In this paper, we focus on the task offloading among vehicles, i.e., the driving systems or passengers of some vehicles generate computation tasks, while some other surrounding vehicles can provide computing services. We call the vehicles that require task offloading *task vehicles (TaVs)*, and vehicles who can help to execute tasks *service vehicles (SeVs)*. We design a distributed task offloading algorithm to minimize the average delay, where the task offloading decision is made by each TaV individually.

Multiple SeVs might be available to process each task, and a key challenge is the lack of accurate state information of SeVs in the dynamic VEC environment. The network topology and the wireless channel states vary rapidly due to the movements of vehicles [14], and the computation workloads of SeVs fluctuate across time. These factors are difficult to model or to predict, so that the TaV has no idea *in prior* which SeV performs the best in terms of delay performance.

Our solution is *learning while offloading*, i.e., the TaV is able to learn the delay performance while offloading tasks. To be specific, we adopt the multi-armed bandit (MAB) framework to design our task offloading algorithm [15]. The classical MAB problem aims at balancing the exploration and exploitation tradeoff in the learning process: to explore different candidate actions that lead to good estimates of their reward distributions, while to exploit the learned information to select the empirically optimal actions. The upper confidence bound (UCB) based algorithms, such as UCB1 and UCB2, have been proposed with strong performance guarantee [15], and applied to the wireless networks to learn the unknown environments [16]–[18].

Manuscript received June 22, 2018; revised November 19, 2018; accepted January 11, 2019. Date of publication January 28, 2019; date of current version April 16, 2019. This work was supported in part by the Nature Science Foundation of China under Grants 61871254, 91638204, 61571265, 61861136003, and 61621091, in part by the National Key R&D Program of China under Grant 2018YFB0105005, in part by the NSF under Grants CNS-1547461, CNS-1718901, and IIS-1838207, and in part by the Intel Collaborative Research Institute for Intelligent and Automated Connected Vehicles. The review of this paper was coordinated by the Guest Editors of the Special Section on Fog/Edge Computing for Autonomous and Connected Cars. This paper was presented in part at the IEEE International Conference Communication, Kansas City, MO, USA, May 2018 [1]. *(Corresponding author: Sheng Zhou.)*

Y. Sun, J. Song, S. Zhou, and Z. Niu are with the Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: sunyx15@mails.tsinghua.edu.cn; sjh14@mails.tsinghua.edu.cn; sheng.zhou@tsinghua.edu.cn; niuzhs@tsinghua.edu.cn).

X. Guo and X. Liu are with the Department of Computer Science, University of California, Davis, CA 95616 USA (e-mail: guoxueying@outlook.com; xinliu@ucdavis.edu).

Z. Jiang is with the Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China (e-mail: zhiyjiang@foxmail.com).

Digital Object Identifier 10.1109/TVT.2019.2895593

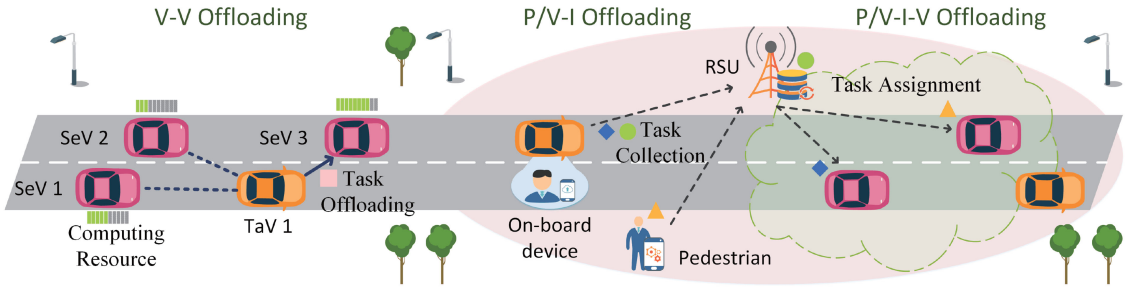


Fig. 1. An illustration of the VEC architecture and three major offloading modes.

However, in our task offloading problem, the movements of vehicles lead to a dynamic candidate SeV set, and the workload of each task is time-varying, leading to a varying cost in exploring the suboptimal actions. These factors have not been addressed by existing MAB schemes, which motivates us to specifically adapt the MAB framework in the vehicular task offloading scenario. Our key contributions include:

(1) We propose an adaptive learning-based task offloading (ALTO) algorithm based on MAB theory, in order to guide the task offloading of TaVs and minimize the average offloading delay. ALTO algorithm works in a distributed manner and enables the TaV to learn the delay performance of candidate SeVs while offloading tasks. The proposed algorithm is of low computational complexity, and does not require the exchange of accurate state information like channel states and computing workloads between vehicles, so that it is easy to implement in the real VEC system.

(2) Two kinds of *adaptivity* are augmented with the proposed ALTO algorithm: *input-awareness* and *occurrence-awareness*, by adjusting the exploration weight according to the workloads of tasks and the appearance time of SeVs. Different from our previous theoretical work [19] which only considers time-varying workloads of tasks with fixed actions, we consider a more general case with dynamic candidate SeVs (actions), and prove that ALTO can effectively balance the exploration and exploitation in the dynamic vehicular environment with sublinear learning regret.

(3) Extensive simulations are carried out under a synthetic scenario, as well as a realistic highway scenario using system level simulator Veins. Results illustrate that our proposed algorithm can achieve low delay performance, and provide guidelines for the settings of key design parameters.

The rest of this paper is organized as follows. We introduce the related work in Section II. The system model and problem formulation is introduced in Section III, and the ALTO algorithm is then proposed in Section IV. The learning regret is analyzed in Section V. Simulation results are then provided in Section VI, and finally comes the conclusions in Section VII.

II. Related Work

A. VEC Architecture and Use Cases

An illustration of the VEC architecture is shown in Fig. 1. The development of vehicle-to-everything (V2X) communication techniques enable vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and vehicle-to-pedestrian (V2P) communications, so that tasks can be offloaded to other vehicles through different kinds of routes. Specifically, there are three major offloading modes:

- *Vehicle-Vehicle (V-V) Offloading*: Vehicles directly offload tasks to their surrounding vehicles with surplus computing resources in a distributed manner. In this case, each individual vehicle may not be able to acquire the global state information for task offloading decisions, and there might be no coordinations for task scheduling.
- *Pedestrian/Vehicle-Infrastructure-Vehicle (P/V-I-V) Offloading*: When there are no other neighboring vehicles for task offloading, one solution is that tasks are first offloaded to the infrastructures alongside, and then assigned to other vehicles in a centralized manner.
- *Pedestrian/Vehicle-Infrastructure (P/V-I) Offloading*: In this mode, tasks are offloaded to the infrastructures for direct processing.

Similar to the traditional cloud computing services, the VEC system can provide infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS) [13], and support a wide variety of applications. For example, cooperative collision avoidance and collective environment perception are necessary for safety driving, where sensing data is generated by a group of vehicles and processed by some of them [20], [21]. In vehicular crowd sensing, the video recordings and images are generated by vehicles and required to be analyzed in real time, in order to supervise the traffic, monitor the road conditions and navigate car parkings [22]. The computing resources of vehicles may be underutilized by the aforementioned vehicular applications [11], which can further provide services for entertainments and multimedia applications, such as cloud gaming, virtual reality, augmented reality and video trans-coding [23].

B. Task Offloading Algorithms

There are some existing efforts investigating the task scheduling and computing resource management problem in VEC. A software-defined VEC architecture is proposed in [13]. Inspired by the software-defined network, a centralized controller is designed to periodically collect the state information of vehicles, including mobility and resource occupation, and manage radio and computing resources upon task requests. In terms of P/V-I-V offloading, a semi-Markov decision based centralized task assignment problem is formulated in [24], in order to minimize the average system cost by jointly considering the delay of tasks

and the energy consumption of mobile devices. Ref. [25] further introduces task replication technique to improve the service reliability of VEC, where task replicas can be offloaded to multiple vehicles to be processed simultaneously. However, a key drawback of the centralized framework is that, it requires frequent state information update to optimize the system performance, which is of high signaling overhead.

An alternative method is to make task offloading decisions by the task generators in a distributed manner. An autonomous vehicular edge framework which enables V-V and V-I offloading is proposed in [23], followed by a task scheduling algorithm based on ant colony optimization. However, when the number of vehicles is large, the computational complexity can be quite high. We will design a distributed task offloading algorithm with low complexity.

III. System Model and Problem Formulation

A. V-V Offloading: System Overview

We consider V-V offloading in the VEC system, where vehicles involved in the task offloading are classified into two categories: *TaVs* are the vehicles that generate and offload computation tasks for cloud execution, while *SeVs* are the vehicles with sufficient computing resources that can provide computing services. Note that the role of each vehicle depends on the sufficiency of its computing resources, and is not fixed to TaV or SeV during the trip.

TaVs can offload tasks to their neighboring SeVs. Each TaV may have multiple candidate SeVs that can process the tasks, and each task is offloaded to a single SeV and executed by it. As shown in Fig. 1, for TaV 1, there are 3 candidate SeVs (SeV 1-3), and currently the task is offloaded to SeV 3.

In this work, we design distributed task offloading algorithm to minimize the delay performance, by letting each TaV decide which SeV should serve each task independently, without inter-TaV cooperations. Moreover, we do not make any assumptions on the service disciplines of SeVs, nor the mobility models of vehicles.

B. Task Offloading Procedure

Since offloading decisions are made in a distributed manner, we then focus on a single TaV of interest and model the task offloading problem. Consider a discrete-time VEC system. There are four procedures for task offloading within each time period:

SeV discovery: The TaV discovers neighboring SeVs within its communication range, and selects those in the same moving direction as candidates. Here the driving states of each vehicle, including speed, location and moving direction, can be acquired by other neighboring vehicles through vehicular communication protocols. For example, in dedicated short-range communication (DSRC) standard [26], the periodic beaconing messages can provide these state information. Denote the candidate SeV set in time period t by $\mathcal{N}(t)$, which may change across time since vehicles are moving. And due to the unknown mobility model, candidate SeVs in the future are unknown in prior. Besides, assume that $\mathcal{N}(t) \neq \emptyset$ for $\forall t$, otherwise the TaV can seek help from RSUs along the road, which is beyond the scope of this paper.

Task upload: After updating the candidate SeV set $\mathcal{N}(t)$ at the beginning of each time period, the TaV selects one SeV $n \in \mathcal{N}(t)$ and uploads the computation task. Denote the input data size of the task generated in time period t by x_t (in bits), which is required to be transmitted from TaV to SeV. The uplink wireless channel state between TaV and SeV $n \in \mathcal{N}(t)$ is denoted by $h_{t,n}^{(u)}$, and the interference power at SeV n is $I_{t,n}^{(u)}$. We assume that the wireless channel state remains static during the uploading process of each computation task. Given the fixed transmission power P, channel bandwidth W and noise power σ^2, the uplink transmission rate $r_{t,n}^{(u)}$ between the TaV and SeV n is

$$r_{t,n}^{(u)} = W \log_2 \left(1 + \frac{P h_{t,n}^{(u)}}{\sigma^2 + I_{t,n}^{(u)}} \right). \tag{1}$$

And the transmission delay $d_{\mathrm{up}}(t,n)$ of uploading the task to SeV n in time period t is given by

$$d_{\mathrm{up}}(t,n) = \frac{x_t}{r_{t,n}^{(u)}}. \tag{2}$$

Task execution: The selected SeV n processes the task after receiving the input data from the TaV. For the task generated in time period t, the total workload is given by $x_t w_t$, where w_t is computation intensity (in CPU cycles per bit) representing how many CPU cycles are required to process one bit input data [4]. The computation intensity w_t of the task mainly depends on the nature of applications.

The computing capability of SeV n is described by its maximum CPU frequency F_n (in CPU cycles per bit), and the allocated CPU frequency to the task of TaV in time period t is denoted by $f_{t,n}$. The SeV may deal with multiple computation tasks simultaneously, and adopt dynamic frequency and voltage scaling (DVFS) technique to dynamically adjust the CPU frequency [27], and thus we have $f_{t,n} \in [0, F_n]$. We assume that $f_{t,n}$ remains static during each time period t, and each computation task can be completed within each time period due to the timely requirements. Tasks of larger workloads can be further partitioned into multiple subtasks [18], [28], so that each subtask is offloaded to and processed by a SeV within one time period. Then the computation delay can be written as

$$d_{\mathrm{com}}(t,n) = \frac{x_t w_t}{f_{t,n}}. \tag{3}$$

Result feedback: Upon the completion of task execution, the selected SeV n transmits back the result to the TaV. Let $h_{t,n}^{(d)}$ denote the downlink wireless channel state, which is assumed to be static during the transmission of each result. The interference at the TaV is denoted by $I_t^{(d)}$. Similar to (2), the downlink transmission rate $r_{t,n}^{(d)}$ from SeV n to TaV can be written as

$$r_{t,n}^{(d)} = W \log_2 \left(1 + \frac{P h_{t,n}^{(d)}}{\sigma^2 + I_t^{(d)}} \right). \tag{4}$$

The data volume of the computation result in time period t is denoted by y_t (in bits), and thus the downlink transmission delay

from SeV n to the TaV is

$$d_{\mathrm{dow}}(t,n) = \frac{y_t}{r_{t,n}^{(d)}}. \tag{5}$$

Then the sum delay $d_{\mathrm{sum}}(t,n)$ of offloading the task to SeV n in time period t can be given by

$$d_{\mathrm{sum}}(t,n) = d_{\mathrm{up}}(t,n) + d_{\mathrm{com}}(t,n) + d_{\mathrm{dow}}(t,n). \tag{6}$$

C. Problem Formulation

Consider a total number of T time periods. Our objective is to minimize the average offloading delay, by guiding the task offloading decisions of the TaV on which SeV should serve each task. The task offloading problem is formulated as

$$\mathbf{P1}: \min_{a_1,\ldots,a_T} \frac{1}{T}\sum_{t=1}^{T} d_{\mathrm{sum}}(t,a_t), \tag{7}$$

where a_t is the optimization variable, which represents the index of SeV selected in time period t, with $a_t \in \mathcal{N}(t)$.

Availability of state information: The state information related to the delay performance can be classified into two categories based on its ownership: parameters of each task, including the input and output data volumes x_t, y_t and computation intensity w_t, are known by the TaV upon the generation of each task. The uplink and downlink transmission rates $r_{t,n}^{(u)}$, $r_{t,n}^{(d)}$ and the allocated CPU frequency $f_{t,n}$ are closely related to the SeV. If all these states are exactly known by the TaV before offloading each task, the sum delay $d_{\mathrm{sum}}(t,n)$ of SeV $n \in \mathcal{N}(t)$ can then be calculated, and the optimization problem **P1** is easy to solve with

$$a_t = \min_{n\in\mathcal{N}_t} d_{\mathrm{sum}}(t,n). \tag{8}$$

However, due to the mobility of vehicles, the transmission rates vary fast across and are difficult to predict. Since there is no cooperation between TaVs, the computation loads at SeVs dynamically change, making the allocated CPU frequency vary across time. Moreover, exchanging these state information between the TaV and all candidate SeVs causes high signaling overhead. Therefore, the TaV may lack the state information of SeVs, and can not realize which SeV provides the lowest delay when making offloading decisions.

Learning while offloading: To overcome the unavailability of the state information of SeVs, we propose the approach *learning while offloading*: the TaV can observe and learn the delay performance of candidate SeVs while offloading computation tasks. Specifically, the SeV a_t in time period t is selected according to the historical delay observations $d(1,a_1), d(2,a_2), ..., d(t-1,a_{t-1})$, without acquiring the exact transmission rates and CPU frequency. We aim to design a learning algorithm that minimizes the expectation of offloading delay, written as

$$\mathbf{P2}: \min_{a_1,\ldots,a_T} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} d_{\mathrm{sum}}(t,a_t)\right]. \tag{9}$$

In the rest of the paper, we consider a simplified version of **P2** by assuming that the input data size x_t of task is time-varying, but the computation intensity w_t and the ratio of output and input data volume y_t/x_t remains constant across time. In practical, this is a valid assumption when tasks are generated by the same type of application. Let $y_t/x_t = \alpha_0$ and $w_t = \omega_0$ for $\forall t$. Then the sum delay of offloading the task to SeV n in time period t can be transformed as

$$d_{\mathrm{sum}}(t,n) = x_t\left(\frac{1}{r_{t,n}^{(u)}} + \frac{\alpha_0}{r_{t,n}^{(d)}} + \frac{\omega_0}{f_{t,n}}\right). \tag{10}$$

Define the *bit offloading delay* as

$$u(t,n) = \frac{1}{r_{t,n}^{(u)}} + \frac{\alpha_0}{r_{t,n}^{(d)}} + \frac{\omega_0}{f_{t,n}}, \tag{11}$$

which represents the sum delay of offloading one bit input data of the task to SeV n in time period t. The bit offloading delay $u(t,n)$ reflects the service capability of each candidate SeV, which is what the TaV needs to learn.

Finally, the optimization problem can be written as

$$\mathbf{P3}: \min_{a_1,\ldots,a_T} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} x_t u(t,n)\right]. \tag{12}$$

IV. Adaptive Learning-Based Task Offloading Algorithm

In this section, we develop a learning-based task offloading algorithm based on MAB, which enables the TaV to learn the delay performance of candidate SeVs and minimizes the expected offloading delay.

Our task offloading problem **P3** requires online sequential decision making, which can be solved according to the MAB theory. Each SeV corresponds to an arm whose loss (bit offloading delay) is governed by an unknown distribution. The TaV is the decision maker who tries an arm at a time and learns the estimation of its loss, in order to minimize the expectation of cumulative loss across time. However, the variations of input data size x_t and candidate SeV set $\mathcal{N}_t$ incapacitate existing algorithms of MAB, such as UCB1 and UCB2, in the VEC system.

In this work, we propose an Adaptive Learning-based Task Offloading (ALTO) algorithm which is aware of both the input data size of tasks and the occurrence of vehicles, as shown in Algorithm 1. Parameter β is a constant weight, and $k_{t,n}$ records the number of tasks that have been offloaded to SeV n up till time t. The occurrence time of SeV n is recorded by t_n, and the input data size x_t is normalized to be $\tilde{x}_t$ within [0, 1] as:

$$\tilde{x}_t = \max\left\{\min\left(\frac{x_t - x^-}{x^+ - x^-}, 1\right), 0\right\}, \tag{13}$$

where x^+ and x^- are the upper and lower thresholds to normalize x_t. In particular, if $x^+ = x^-$, $\tilde{x}_t = 0$ when $x_t \le x^-$, and $\tilde{x}_t = 1$ when $x_t > x^-$.

In Algorithm 1, Lines 3-5 are the initialization phase, which is called whenever new SeVs occur as candidates. The TaV selects the newly appeared SeV n once and offloads the task, in order to get an initial estimation of its bit offloading delay.

Algorithm 1 ALTO: Adaptive Learning-based Task Offloading Algorithm.

1: **Input**: parameters α_0, ω_0, β, x^+ and x^-.
2: **for** $t = 1, ..., T$ **do**
3: **if** Any SeV $n \in \mathcal{N}(t)$ has not connected to TaV **then**
4: Connect to SeV n once.
5: Update $\bar{u}_{t,n} = d_{\text{sum}}(t,n)/x_t$, $k_{t,n} = 1$, $t_n = t$.
6: **else**
7: Observe x_t, calculate $\tilde{x}_t$.
8: Calculate the utility function of each candidate SeV $n \in \mathcal{N}(t)$:

$$\hat{u}_{t,n} = \bar{u}_{t-1,n} - \sqrt{\frac{\beta(1-\tilde{x}_t)\ln(t-t_n)}{k_{t-1,n}}}. \quad (14)$$

9: Offload the task to SeV a_t, such that:

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

10: Observe the sum offloading delay $d_{\text{sum}}(t, a_t)$.
11: Update $\bar{u}_{t,a_t} \leftarrow \frac{\bar{u}_{t-1,a_t} k_{t-1,a_t} + d_{\text{sum}}(t,a_t)/x_t}{k_{t-1,a_t}+1}$.
12: Update $k_{t,a_t} \leftarrow k_{t-1,a_t} + 1$.
13: **end if**
14: **end for**

Lines 7-12 are the main loop of the learning process, inspired by the volatile UCB (VUCB) algorithm [29] and the our previous work on opportunistic MAB [19]. During each time period, the TaV gets the data volume x_t before offloading the task and calculates $\tilde{x}_t$. The utility function defined in (14) is used to evaluate the service capability of each SeV, which consists of the empirical bit offloading delay $\bar{u}_{t,n}$ and a padding function. Specifically, $\bar{u}_{t,n}$ is the average bit offloading delay of SeV n observed until time period t. And the padding function jointly considers the input data size and occurrence time of each SeV, in order to balance the exploration and exploitation in the learning process, and adapt to the dynamic VEC environment. The offloading decision is then made according to (15), by selecting the SeV with minimum utility. Finally, the offloading delay is observed upon result feedback, and $\bar{u}_{t,a_t}$ and k_{t,a_t} is updated.

Two kinds of adaptivity of the algorithm are highlighted as follows.

Input-awareness: The input data size x_t can be regarded as a weight factor on the offloading delay. Intuitively, when x_t is small, even if the TaV selects a poorly performed SeV, the sum offloading delay will not be too large. On the other hand, when x_t is large, selecting a SeV with weak service capability brings great delay degradation. Therefore, the padding function is proportional to $\sqrt{1-\tilde{x}_t}$ that is non-increasing as x_t grows, so that ALTO explores more when x_t is small, while exploits more when x_t is large.

Occurrence-awareness: The random presences of SeVs are also considered, and the proposed ALTO algorithm has occurrence-awareness. To be specific, for any newly appeared SeV, $\sqrt{\frac{\ln(t-t_n)}{k_{t-1,n}}}$ is large due to the small number of selections $k_{t-1,n}$, so that ALTO tends to explore more. Meanwhile, ALTO is able to exploit the learned information of any existing SeV, since more times of connections lead to a small value of the padding function.

A. Complexity

In our proposed ALTO algorithm, the computational complexity of calculating the utility functions of all candidate SeVs in Line 8 is $O(N)$, where $N = |\mathcal{N}(t)|$ is the number of candidate SeVs in time period t. The task offloading decision made in Line 9 is a minimum seeking problem, with complexity $O(N)$. Updating the empirical bit offloading delay $\bar{u}_{t,a_t}$ and offloaded times k_{t,a_t} has a complexity of $O(1)$. Therefore, within each time period, the total computational complexity of running ALTO to offload one task is $O(N)$. Assume that there are totally M tasks required to be offloaded in the VEC system. Since TaVs offload tasks independently, the total amount of computation is $O(MN)$.

An ant colony optimization based distributed task offloading algorithm is proposed in [23]. According to Section V-D, the computational complexity is $O(KM^2N)$, where K is the number of iterations required by the ant colony optimization. Therefore, ALTO is of lower complexity than the existing algorithm in [23].

B. Signaling Overhead

Considering the distributed V-V offloading case, the complete-state task offloading (CSTO) policy is that, the TaV obtains the accurate state information of all candidate SeVs, evaluates their delay performance, and selects the SeV with minimum offloading delay. Compared with the CSTO policy, our proposed ALTO algorithm is of lower signaling overhead and much easier to implement in the real VEC system.

First, the uplink and downlink wireless channel states, allocated CPU frequency and interference of each candidate SeV are not required to know by the ALTO algorithm. Therefore, for each TaV, offloading a task can save at least N signaling messages for the state information of the N candidate SeVs, and MN signaling messages can be saved for M tasks. Second, when a SeV is serving multiple TaVs simultaneously, the CSTO policy needs to know the task workload of TaVs to allocate computing resources of the SeV. In this case, more signaling messages are generated by the CSTO policy. Last but not least, frequent signaling exchange may lead to additional collisions and retransmissions, and the delayed state information may not be accurate. The proposed ALTO algorithm enables each TaV to learn the state information of SeVs instead of obtaining them from signaling messages, and thus reduces the signaling overhead.

V. Performance Analysis

In this section, we characterize the delay performance of the proposed ALTO algorithm. We adopt the *learning regret* of delay as the performance criteria, which is widely used in the MAB theory. Compared with the existing UCB based algorithms in

[15], two major modifications in ALTO are the occurrence time t_n and normalized input $\tilde{x}_t$. We first evaluate their impacts on the learning regret separately, and then jointly analyze these two factors.

A. Definition of Learning Regret

Define an *epoch* as the interval during which candidate SeVs remain identical. The total number of epochs during the considered T time periods is denoted by B, and let $\mathcal{N}_b$ be the candidate SeV set of the bth epoch, where $b = 1, 2, ..., B$. Let t_b and t'_b be the start and end time of the bth epoch, with $t_1 = 1$ and $t'_B = T$.

For theoretical analysis, we assume that for each SeV n, its bit offloading delay $u(t, n)$ is i.i.d. over time and independent of others. We will show in Section VI through simulation results that without this assumption, ALTO still works well.

Define the mean bit offloading delay of each candidate SeV n as $\mu_n = \mathbb{E}_t[u(t, n)]$. During each epoch, let $\mu_b^* = \min_{n\in\mathcal{N}_b} \mu_n$ be the optimal bit offloading delay, and $a_b^* = \arg\min_{n\in\mathcal{N}_b} \mu_n$ the index of the optimal SeV. Note that μ_b^* and a_b^* are unknown in prior.

The learning regret represents the expected cumulative performance loss of sum offloading delay brought by the learning process, which is compared with the genie-aided optimal policy where the TaV always selects the SeV with maximum service capability. The learning regret by time period T can be written as

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

In the following subsections, we will characterize the upper regret bound of ALTO algorithm.

B. Regret Analysis Under Dynamic SeV Set and Identical Input

We first assume that the input data size is not time-varying, and analyze the learning regret under varying SeV set. Let $x_t = x_0$ for $\forall t$, and $x^+ = x^- = x_0$, then $\tilde{x}_t = 0$. The utility function (14) becomes

$$\hat{u}_{t,n} = \bar{u}_{t-1,n} - \sqrt{\frac{\beta \ln(t - t_n)}{k_{t-1,n}}}, \tag{17}$$

and the learning regret

$$R_T = x_0 \sum_{b=1}^{B} \mathbb{E}\left[\sum_{t=t_b}^{t'_b} \left(u(t, n) - \mu_b^*\right)\right]. \tag{18}$$

Also, define the maximum bit offloading delay during the T time periods as $u_m = \sup_{t,n} u(t, n)$, the performance difference between any suboptimal SeV $n \in \mathcal{N}_b$ and the optimal SeV in the bth epoch $\delta_{n,b} = (\mu_n - \mu_b^*)/u_m$. Let $\beta = \beta_0 u_m^2$, where β_0 is a constant.

The learning regret within each epoch is upper bounded in Lemma 1.

Lemma 1: Let $\beta_0 = 2$, the learning regret of ALTO with dynamic SeV set and identical input data size has an upper bound in each epoch. Specifically, in the bth epoch:

$$R_b \le x_0 u_m \left[\sum_{n\neq a_b^*} \frac{8 \ln(t'_b - t_n)}{\delta_{n,b}} + \left(1 + \frac{\pi^2}{3}\right) \sum_{n\neq a_b^*} \delta_{n,b}\right]. \tag{19}$$

Proof: See Appendix A. ■

Then we have the following Theorem 1 that provides the upper bound of the learning regret over T time periods.

Theorem 1: Let $\beta_0 = 2$. For a given time horizon T, the total learning regret R_T of ALTO dynamic SeV set and identical input data size has an upper bound as follows:

$$R_T \le x_0 u_m \sum_{b=1}^{B} \left[\sum_{n\neq a_b^*} \frac{8 \ln T}{\delta_{n,b}} + O(1)\right]. \tag{20}$$

Proof: See Appendix B. ■

Theorem 1 implies that, our proposed ALTO algorithm provides a sublinear learning regret compared to the genie-aided optimal policy. To be specific, within each epoch, the learning regret is governed by $O(\ln T)$, and inversely proportional to the performance difference $\delta_{n,b}$ of optimal SeV and suboptimal SeV $n \neq a_b^*$. Moreover, for any finite time horizon T with B epochs, ALTO achieves $O(B \ln T)$ learning regret.

Remark 1: The random appearance and disappearance of SeVs affect the number of epochs B and the learning regret $O(B \ln T)$. Within a fixed number of time periods, higher randomness of SeVs results in a more dynamic environment, and thus higher learning regret.

Remark 2: To prove Lemma 1 and Theorem 1, we have to normalize the bit offloading delay $u(t, n)$ within [0, 1] for $\forall t, n$, by setting $u_m = \sup_{t,n} u(t, n)$. In practical, the exact value of u_m is not easy to acquire in prior. Instead, u_m can be set to the maximum $u(t, n)$ that has been observed till the current time period.

C. Regret Analysis Under Varying Input and Fixed Candidate SeVs

We then characterize the upper bound of the learning regret within a single epoch, and consider that the input data size x_t is random and continuous. Let $B = 1$. The optimal SeV is $a^* = \arg\min_{n\in\mathcal{N}_1} \mu_n$, and its mean bit offloading delay $\mu^* = \min_{n\in\mathcal{N}_1} \mu_n$. The learning regret can be simplified as

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} x_t (u(t, n) - \mu^*)\right]. \tag{21}$$

The following theorem bounds the learning regret under varying input data size and fixed candidate SeV set.

Theorem 2: Let $\beta_0 = 2$, and $\mathbb{P}\{x_t \le x^-\} > 0$. For any finite time horizon T, we have:

(1) When $x^+ \ge x^-$, the expected number of tasks $k_{T,n}$ offloaded to any SeV $n \neq a^*$ can be bounded as

$$\mathbb{E}[k_{T,n}] \le \frac{8 \ln T}{\delta_n^2} + O(1). \tag{22}$$

(2) With $x^+ = x^-$, the learning regret can be bounded as

$$R_T \leq u_m \sum_{n \neq a^*} \left[\frac{8 \ln T \mathbb{E}[x_t | x_t \leq x^-]}{\delta_n} + O(1) \right], \quad (23)$$

where $\mathbb{E}[x_t | x_t \leq x^-]$ is the expectation of x_t on the condition that $x_t \leq x^-$, $u_m = \sup_{t,n} u(t,n)$, and $\delta_n = (\mu_n - \mu^*)/u_m$.

Proof: See Appendix C. ■

According to Theorem 2, the time order of the learning regret is $O(\ln T)$, indicating that under time-varying input data volume, the TaV is still able to learn which SeV performs the best, and achieves a sublinear deviation compared to the genie-aided optimal policy.

Recall that compared to the existing UCB based algorithms, the major modification under varying input is the introduction of normalized input $\tilde{x}_t$, which dynamically adjusts the weight of exploration and exploitation. As shown in (23), the consideration of $\tilde{x}_t$ brings an coefficient $\mathbb{E}[x_t | x_t \leq x^-]$ to the learning regret. When the input data size is fixed to x_0, the coefficient of the learning regret of conventional UCB algorithms is x_0. Therefore, by properly selecting the lower threshold x^-, we have $\mathbb{E}[x_t | x_t \leq x^-] < x_0$. This implies that the proposed ALTO algorithm can take the opportunity to explore when x_t is small, and achieve lower learning regret.

Moreover, when the task offloading scenario is simplified to the case with fixed candidate SeVs and identical input data size, the proposed ALTO algorithm reduces to a conventional UCB algorithm, and the lower bound of the learning regret has been investigated in [30]–[32], which is provided in Appendix D. Specifically, the regret lower bound of conventional UCB algorithms is $x_0 u_m \sum_{n \neq a^*} \frac{\delta_n \ln T}{D(n, a^*)}$, where $D(n, a^*)$ is the Kullback-Leibler divergence of the bit offloading delay distributions. Therefore, in the case with varying input, the regret upper bound of ALTO is even possible to be smaller than the lower bound of conventional UCB algorithms, due to the input-awareness.

D. Joint Consideration of Occurrence-Awareness and Input-Awareness

Finally, we analyze the learning regret by jointly considering the occurrence of vehicles and the variations of input data size. Although these two factors are independent with each other, they actually couple together in the utility function (14), and collectively balance the exploration and exploitation in the learning process. Therefore, it is quite difficult to derive the upper bound of the learning regret in this case.

We study a special case with periodic input and fixed bit offloading delay, and derive the theoretical upper bound to provide some insights. To be specific, assume that the input data size $x_t = \epsilon_0$ when t is even, and $x_t = 1 - \epsilon_1$ when t is odd, where $\epsilon_0, \epsilon_1 \in [0, 0.5)$. Let $x^+ = 1$, and $x^- = \epsilon_0$, thus $\tilde{x}_t = 0$ when $x_t = \epsilon_0$, and $\tilde{x}_t = 1 - \frac{\epsilon_1}{1-\epsilon_0}$ when $x_t = 1 - \epsilon_1$ Consider two SeVs appear at t_1 and t_2 respectively, and $t_1 \neq t_2$. Then there are 2 epochs during T time periods, and we only need to focus on the second epoch, since the first epoch only has one SeV available. The bit offloading delay of each SeV is fixed, with $u(t,n) = \mu_n$ for $\forall t, n = 1, 2$, but unknown in prior. Without loss of generality, let $\mu_1 \leq \mu_2$, and $\Delta = (\mu_2 - \mu_1)/\mu_2$.

The learning regret can be written as

$$\begin{aligned} R_T &= \mathbb{E}\left[\sum_{\max\{t_1, t_2\}}^{T} (u(t,n) - \mu_1) \right] \\ &= (\mu_2 - \mu_1) \mathbb{E}\left[k_{T,2}^{(2)} \right], \end{aligned} \quad (24)$$

where $k_{T,2}^{(2)}$ represents how many times SeV 2 is selected in the second epoch.

The upper bound for learning regret of ALTO algorithm under periodic input and fixed bit offloading delay is given in the following theorem.

Theorem 3: Let $\beta_0 = 2$. With periodic input data size and fixed bit offloading delay, we have:

$$R_T \leq \frac{2 \mu_2 \epsilon_0 \ln T}{\Delta} + O(1). \quad (25)$$

Proof: See Appendix E. ■

The learning regret in (25) indicates that, when jointly considering the time-varying feature of input data size and candidate SeV set, the proposed ALTO algorithm still achieves $O(\ln T)$ regret, and focuses on the exploration only when the input is low ($x_t = \epsilon_0$).

Conjecture 1: The proposed ALTO algorithm with random continuous input data size and dynamic SeV set achieves $O(B \ln T)$ learning regret.

The conjecture follows the insight that, when the candidate SeV set is identical over time, the learning regret can be derived in a general case with random continuous input and random bit offloading delay, as shown in (23). When the occurrence time of each SeV is different, within single epoch, the learning regret in (25) resembles (23), both governed by the time order $O(\ln T)$. Following the similar generalization method in [19], we may draw a similar conclusion that with random continuous input data size and dynamic SeV set, the learning regret within an epoch is $O(\ln T)$, and the total learning regret is $O(B \ln T)$.

VI. Simulations

To evaluate the average delay performance and learning regret of the proposed ALTO algorithm, we carry out simulations in this section. We start from a synthetic scenario to evaluate the impact of key parameters, and then simulate a realistic highway scenario using system level simulator Veins[1] (VEhicles in Network Simulations) to further verify the proposed ALTO algorithm.

A. Simulation Under Synthetic Scenario

We carry out simulations in the synthetic scenario using MATLAB. Consider one TaV of interest, with 8 SeVs that appear as candidates during $T = 3000$ time periods. The communication range is set to 200m. The distance of the TaV and each candidate SeV ranges within $[10, 200]$ m, and changes randomly

[1] http://veins.car2x.org/

TABLE I
CANDIDATE SEVS AND MAXIMUM CPU FREQUENCY

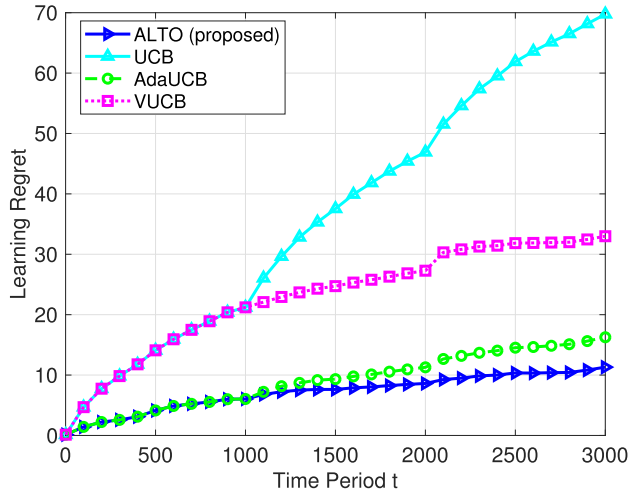
Index of SeV	1	2	3	4	5	6	7	8
F_n (GHz)	3.5	4.5	5	5.5	3	6.5	6	4
Epoch 1 (time 1∼1000)	√	√	√	√	√	–	–	–
Epoch 2 (time 1001∼2000)	√	√	√	√	×	√	√	–
Epoch 3 (time 2001∼3000)	×	√	√	√	×	×	√	√

from -10 m to 10 m in each time period. The occurrence and disappearance time of SeVs, as well as their maximum CPU frequency F_n are shown in Table I. There are 3 epochs, and each lasts 1000 time periods. In the first epoch, there are 5 candidate SeVs. At the beginning of the second epoch, a less powerful SeV 5 disappears and SeVs 6 and 7 with higher computing capability appear. At the beginning of the third epoch, SeVs 1 and 6 disappear, while SeV 8 with suboptimal computing capability arrives. Note that the occurrence and disappearance time of SeVs are unknown to the TaV in prior.

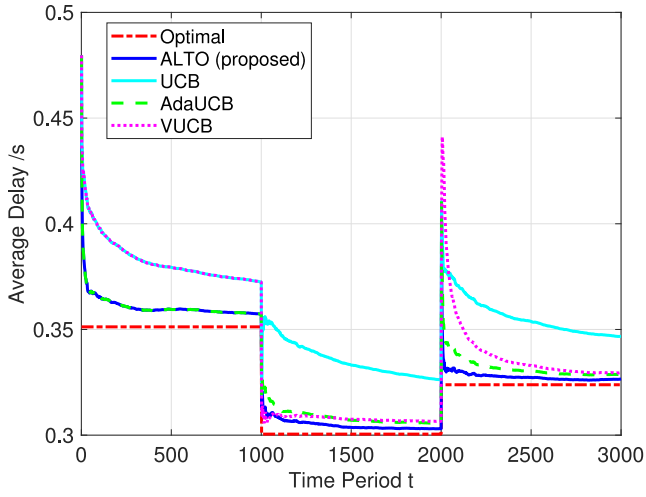
The input data size x_t follows uniform distribution within $[0.2, 1]$ Mbits. The computation intensity is set to $\omega_0 = 1000$ Cycles/bit, and the upper and lower thresholds are selected such that $\mathbb{P}\{x \le x^-\} = 0.05$ and $x^+ = x^-$. Recall that for each SeV, the allocated CPU frequency $f_{t,n}$ to the TaV is a fraction of the maximum CPU frequency, which is randomly distributed from $20\%F_n$ to $50\%F_n$. The wireless channel state is modeled by an inverse power law $h_{t,n}^{(u)} = h_{t,n}^{(d)} = A_0 l^{-2}$, with $A_0 = -17.8$ dB, and l is the distance between TaV and SeV [33]. Other default parameters include: transmission power $P = 0.1$ W, channel bandwidth $W = 10$ MHz, noise power $\sigma^2 = 10^{-13}$ W, and weight factor $\beta_0 = 0.5$.

In Fig. 2, the proposed ALTO algorithm is compared with three existing learning algorithms under the MAB framework. 1) **UCB** is proposed in [15], which is neither input-aware nor occurrence-aware, with padding function $\sqrt{\frac{\beta \ln t}{k_{t-1,n}}}$. 2) **VUCB** is aware of the occurrence of SeVs, with padding function $\sqrt{\frac{\beta \ln (t-t_n)}{k_{t-1,n}}}$ [29]. 3) **AdaUCB** is input-aware, with padding function $\sqrt{\frac{\beta(1-\tilde{x}_t)\ln t}{k_{t-1,n}}}$ [19]. Note that in the first epoch, VUCB is equivalent to UCB, and AdaUCB is equivalent to ALTO. Besides, in the **Optimal** genie-aided policy, the TaV always connects to the SeV with minimum expected delay, which is the delay lower bound of the learning algorithm.

The comparison of learning regret is shown in Fig. 2(a), which provides two major observations as follows. First, the proposed ALTO algorithm performs the best among the four learning algorithms. To be specific, both VUCB and AdaUCB achieve lower learning regret compared with UCB algorithm, which means that either input-awareness or occurrence-awareness brings adaptivity to the dynamic VEC environment and reduces loss of delay performance through learning. The joint consideration of these two factors further optimizes the exploration-exploitation tradeoff, and decreases the learning regret by 85%, 65% and 30% from that of UCB, VUCB and AdaUCB respectively. Second, the learning regret of ALTO grows sublinearly with time t, indicating that the TaV can asymptotically converge to the SeV with optimal delay performance. As shown in Fig. 2(b), during each epoch, the average delay of ALTO converges faster to the optimal delay than other learning algorithms, and achieves close-to-optimal delay performance.



(a) Learning regret.



(b) Average delay.

Fig. 2. Comparison of ALTO algorithm and existing learning algorithms in terms of the learning regret and average delay.

We then consider a single epoch and set SeVs 2-7 in Table I as candidates for 3000 time periods. Fig. 3 evaluates the impact of weight factor β_0 on the learning regret. When $\beta_0 = 0$, there is no exploration in the learning process, and the learning regret is drastically worse than those of $\beta_0 > 0$, since ALTO may stick to a suboptimal SeV for a long time. When $\beta_0 > 0$, the learning regret grows up slightly as β_0 increases. Although the existing effort shows that the sublinear learning regret is achieved when $\beta_0 > 0.5$ [31], in our simulation, the learning regret is lower when $\beta_0 = 0.2$. The reason may be that only a small number of explorations can help the TaV to find the optimal SeV under our settings.

Finally, we try different pairs of upper and lower thresholds for normalizing the input data size, and evaluate the effect on the learning regret. Define $\mathbb{P}\{x \le x^+\} = \rho^+$ and $\mathbb{P}\{x \le x^-\} = \rho^-$, as the probability that the input data size is

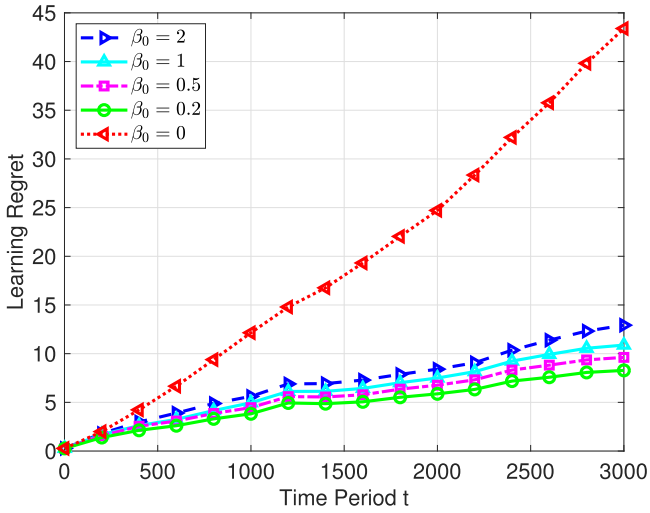


Fig. 3. Learning regret of ALTO under different weight factors β_0.

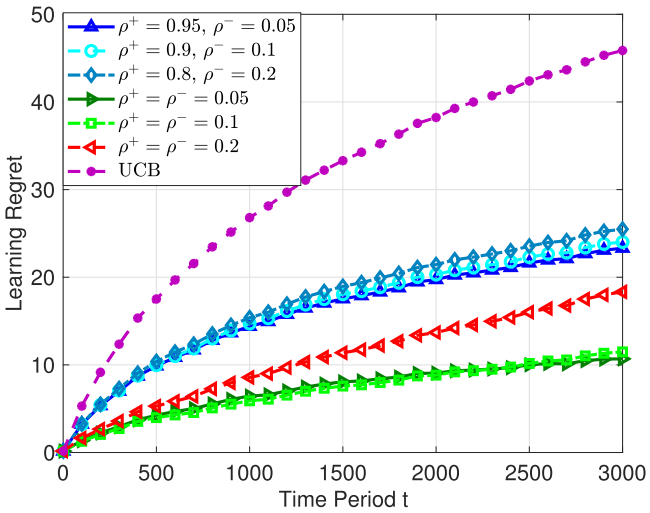


Fig. 4. Learning regret of ALTO under different normalized factors x^+ and x^-, with $\mathbb{P}\{x \le x^+\} = \rho^+$ and $\mathbb{P}\{x \le x^-\} = \rho^-$.

higher (or lower) than the upper (or lower) threshold. Two kinds of thresholds are selected: 1) $\rho^+ = \rho^-$, indicating that $x^+ = x^-$ and explorations happen only when $x \le x^-$. 2) $1 - \rho^+ = \rho^-$, where explorations also happen when the input data size is between x^- and x^+. As shown in Fig. 4, the proposed ALTO algorithm always outperforms UCB algorithm. Moreover, the learning regret under $\rho^+ = \rho^-$ is lower than the case when $1 - \rho^+ = \rho^-$, and achieves the lowest when $\rho^+ = \rho^- = 0.05$ under our settings, which we set as default.

B. Simulation Under Realistic Highway Scenario

In this subsection, simulations are further carried out using system level simulator *Veins*, in order to evaluate the average delay of ALTO under a realistic highway scenario.

The simulation platform Veins integrates a traffic simulator *Simulation of Urban MObility (SUMO)*[2] and a network

[2]http://www.sumo.dlr.de/userdoc/SUMO.html

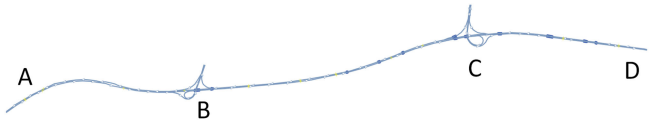
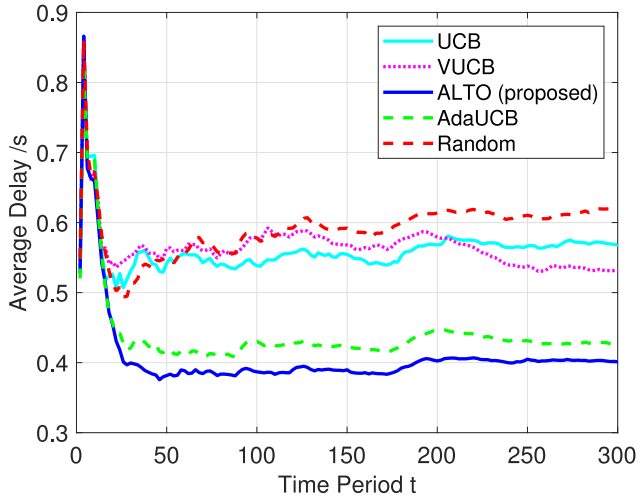
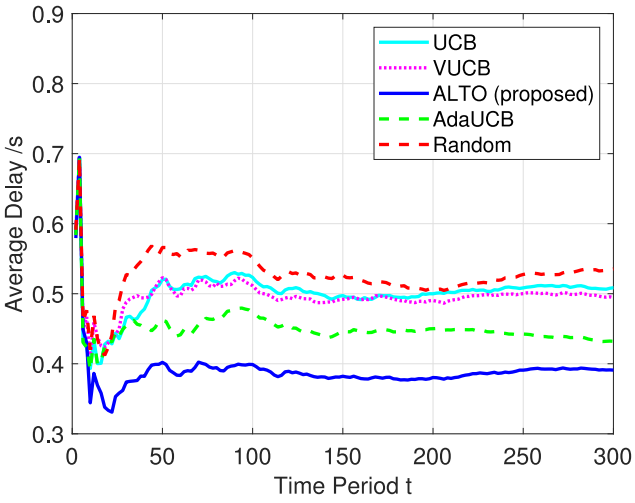


Fig. 5. The highway map used in Veins.



(a) The arrival probability of SeVs from A to D is $p_{\mathrm{AD}} = 0.1$.



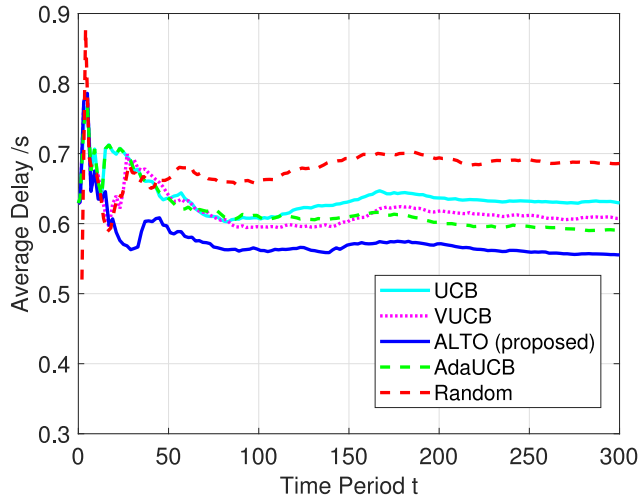
(b) The arrival probability of SeVs from A to D is $p_{\mathrm{AD}} = 0.2$.

Fig. 6. The average delay performance of ALTO algorithm in the highway scenario with 1 TaV.

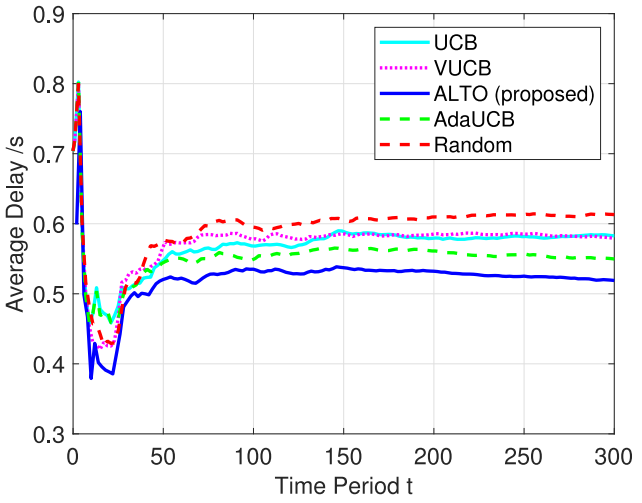
simulator *OMNeT++*,[3] and enables to use real maps from *Open Street Map (OSM)*.[4] Vehicular communication protocols including IEEE 802.11p for PHY layer and IEEE 1609.4 for MAC layer are supported by Veins, together with a two-ray interference model [34] which captures the feature of vehicular channel better.

A 12 km segment of G6 Highway in Beijing is downloaded from OSM and used in our simulation, with two lanes and two ramps, as shown in Fig. 5. The maximum speed of TaVs and

[3]https://www.omnetpp.org/documentation
[4]http://www.openstreetmap.org/



(a) The arrival probability of SeVs from A to D is $p_{\mathrm{AD}} = 0.1$.



(b) The arrival probability of SeVs from A to D is $p_{\mathrm{AD}} = 0.2$.

Fig. 7. The average delay performance of ALTO algorithm in the highway scenario with 10 TaVs, whose inter-arrival time is fixed to 10 s.

SeVs is set to 60 km/h. The TaV moves from A to D, and SeVs have three routes: A to D, A to C and B to D. The arrival of SeVs is modeled by Bernoulli distribution, with probability $p_{\mathrm{AC}} = p_{\mathrm{BD}} = 0.05$, and p_{AD} ranging from 0.1 to 0.2 (e.g., p_{AC} is the probability of the generation of a SeV which departs at A and leaves the road from C at each second). Besides the aforementioned UCB, VUCB and AdaUCB algorithms, we also adopt a naive **Random** policy as a baseline, where the TaV randomly selects a SeV for task offloading in each time period.

Fig. 6 shows the average delay performance with a single TaV, which means the density of SeV is much higher than that of TaV. And in Fig. 7, we consider 10 TaVs that depart every 10 seconds. In this case, each TaV is within some other TaVs' communication range, and thus they might compete for bandwidth and computing resources. We make three major observations as follows. First, the proposed ALTO algorithm always outperforms the other learning algorithms and the random policy, illustrating that ALTO can adapt to the vehicular environment better. To be specific, compared with the UCB algorithm, when $p_{\mathrm{AD}} = 0.1$, ALTO can reduce the average delay by about 30% under single TaV case (Fig. 6(a)), and 13% under multi-TaV scenario (Fig. 7(a)). Second, the average delay grows up when the density of TaV becomes high, since each SeV may serve multiple TaVs simultaneously. Besides, as shown in Fig. 7, when the density of TaV is high, the average delay performance decreases as the arrival probability of SeV increases, since the computing resources are more sufficient.

VII. Conclusions

In this paper, we have studied the task offloading problem in vehicular edge computing (VEC) systems, and proposed an adaptive learning-based task offloading (ALTO) algorithm to minimize the average offloading delay. The proposed algorithm enables each task vehicle (TaV) to learn the delay performance of service vehicles (SeVs) in a distributed manner, without frequent exchange of state information. Considering the time-varying features of task workloads and candidate SeVs, we have modified the existing multi-armed bandit (MAB) algorithms to be input-aware and occurrence-aware, so that ALTO algorithm is able to adapt to the dynamic vehicular task offloading environment. Theoretical analysis has been carried out, providing a sublinear learning regret of the proposed algorithm. We have evaluated the average delay and learning regret of ALTO under a synthetic scenario and a realistic highway scenario, and shown that the proposed algorithm can achieve low delay performance, and decrease the learning regret up to 85% and the average delay up to 30%, compared with the classical upper confidence bound algorithm.

As future work, we plan to formulate the task offloading problem based on adversarial MAB framework [32], where no stochastic assumptions are made on the delay performance of SeVs. The adversarial setting makes learning more difficult, but may perform better under more complicated vehicular environments such as urban scenarios. Besides, we plan to consider the joint resource allocation of vehicles and infrastractures in the VEC system, in order to further optimize the delay performance.

Appendix A
Proof of Lemma 1

In the bth epoch, the learning regret is

$$\begin{aligned} R_b &= x_0 \mathbb{E}\left[\sum_{t=t_b}^{t_b'} u(t,n) - \mu_b^*\right] \\ &= x_0 \mathbb{E}\left[\sum_{n\in\mathcal{N}_b} k_{n,b} u_m \delta_{n,b}\right] \\ &= x_0 u_m \sum_{n\neq a_b^*} \delta_{n,b} \mathbb{E}[k_{n,b}], \end{aligned} \tag{26}$$

where $k_{n,b}$ is the number of tasks offloaded to SeV $n \in \mathcal{N}_b$ in the bth epoch. According to Lemma 1 in [29] and Theorem 1 in [15], when $\beta_0 = 2$, the expected number of tasks offloaded to a

suboptimal SeV has an upper bound as follows

$$\mathbb{E}[k_{n,b}] \leq \frac{8\ln(t'_b - t_n)}{\delta^2_{n,b}} + 1 + \frac{\pi^2}{3}. \tag{27}$$

Substituting (27) into (26), we get:

$$\begin{aligned} R_b &= x_0 u_m \sum_{n \neq a^*_b} \delta_{n,b} \mathbb{E}[k_{n,b}] \\ &\leq x_0 u_m \left[\sum_{n \neq a^*_b} \frac{8\ln(t'_b - t_n)}{\delta_{n,b}} + \left(1 + \frac{\pi^2}{3}\right) \sum_{n \neq a^*_b} \delta_{n,b} \right]. \end{aligned} \tag{28}$$

Thus we can prove Lemma 1.

Appendix B
Proof of Theorem 1

We have $t'_b \leq T$ for $\forall b = 1, 2, ..., B$. Following Lemma 1, the learning regret in the bth epoch can be bounded from above as:

$$\begin{aligned} R_b &\leq x_0 u_m \left[\sum_{n \neq a^*_b} \frac{8\ln(t'_b - t_n)}{\delta_{n,b}} + \left(1 + \frac{\pi^2}{3}\right) \sum_{n \neq a^*_b} \delta_{n,b} \right] \\ &\leq x_0 u_m \left[\sum_{n \neq a^*_b} \frac{8\ln T}{\delta_{n,b}} + O(1) \right]. \end{aligned} \tag{29}$$

By summing over the learning regrets of the B epochs, we have:

$$R_T = \sum_{b=1}^{B} R_b \leq x_0 u_m \sum_{b=1}^{B} \left[\sum_{n \neq a^*_b} \frac{8\ln T}{\delta_{n,b}} + O(1) \right]. \tag{30}$$

Thus Theorem 1 is proved.

Appendix C
Proof of Theorem 2

When $\beta_0 = 2$ and $B = 1$, the utility function in (14) is

$$\hat{u}_{t,n} = \bar{u}_{t-1,n} - u_m \sqrt{\frac{2(1-\tilde{x}_t)\ln t}{k_{t-1,n}}}. \tag{31}$$

The decision making function in (15) can be written as

$$\begin{aligned} a_t &= \arg\min_{n \in \mathcal{N}_1} \hat{u}_{t,n} \\ &= \arg\min_{n \in \mathcal{N}_1} \left\{ \bar{u}_{t-1,n} - u_m \sqrt{\frac{2(1-\tilde{x}_t)\ln t}{k_{t-1,n}}} \right\} \\ &= \arg\min_{n \in \mathcal{N}_1} \left\{ \frac{\bar{u}_{t-1,n}}{u_m} - \sqrt{\frac{2(1-\tilde{x}_t)\ln t}{k_{t-1,n}}} \right\} \\ &= \arg\max_{n \in \mathcal{N}_1} \left\{ 1 - \frac{\bar{u}_{t-1,n}}{u_m} + \sqrt{\frac{2(1-\tilde{x}_t)\ln t}{k_{t-1,n}}} \right\}. \end{aligned} \tag{32}$$

The learning regret can be written as

$$\begin{aligned} R_T &= \mathbb{E}\left[\sum_{t=1}^{T} x_t (u(t,n) - \mu^*) \right] \\ &= u_m \mathbb{E}\left[\sum_{t=1}^{T} x_t \left\{ \left(1 - \frac{\mu^*}{u_m}\right) - \left(1 - \frac{u(t,n)}{u_m}\right) \right\} \right]. \end{aligned} \tag{33}$$

Since $1 - \frac{\bar{u}_{t-1,n}}{u_m} \in [0,1]$, and $1 - \frac{u(t,n)}{u_m} \in [0,1]$, the task offloading problem can be transformed to the opportunistic bandit problem defined in Section III in our previous work [19], with equivalent definitions of learning regret, utility and decision making (as shown in [19], eq. (1-3)). By leveraging Lemma 7 and Appendix C.2 in [19], we can get the upper bound of $\mathbb{E}[k_{T,n}]$, as shown in Theorem 2(1). By leveraging Theorem 3 and Appendix C.2 in [19], we can get the upper bound of the learning regret R_T, as shown in Theorem 2(2).

Appendix D
Regret Lower Bound

The regret lower bound of classical UCB algorithms has been investigated in [30]–[32]. In the following, we provide a regret lower bound of ALTO in a simple task offloading case, with identical input data size x_0 and fixed candidate set of SeVs $\mathcal{N}$ (and thus the index of epoch b is omitted).

Lemma 2: When the candidate SeV set is not time-varying, and the input data size is identical over time, the learning regret can be bounded from above as:

$$R_T \geq x_0 u_m \sum_{n \neq a^*} \frac{\delta_n \ln T}{D(n, a^*)}, \tag{34}$$

where $D(n, a^*)$ is the Kullback-Leibler divergence of the bit offloading delay distributions of SeV n and optimal SeV a^*.

Proof: With fixed SeV set and identical input data size, the proposed ALTO algorithm reduces to the classical UCB algorithm. According to [30], Theorem 5, when $T \to +\infty$, the number of tasks offloaded to a suboptimal SeV n can be bounded as follows

$$\mathbb{E}[k_{T,n}] \geq \frac{\ln T}{D(n, a^*)}. \tag{35}$$

Substituting (35) into (26), the learning regret R_T can be bounded as

$$R_T = x_0 u_m \sum_{n \neq a^*} \delta_n \mathbb{E}[k_{T,n}] \geq x_0 u_m \sum_{n \neq a^*} \frac{\delta_n \ln T}{D(n, a^*)}. \tag{36}$$

■

Appendix E
Proof of Theorem 3

The proof of Theorem 3 follows the similar idea in [19], while the major difference is that the two SeVs appear at t_1 and t_2 respectively. Let $t_0 = \max\{t_1, t_2\}$. We only needs to bound the learning regret in the second epoch, from time t_0 to time T.

We first bound the number of tasks offloaded to the suboptimal SeV.

Lemma 3: With periodic input of tasks and fixed bit offloading delay of SeVs,

$$k_{t,2}^{(2)} \leq \frac{\beta_0 \ln t}{\Delta^2} + 1. \tag{37}$$

Proof: First, (37) holds for $t = t_0$ and $t_0 + 1$. For $t_0 \geq t_0 + 2$, we prove the lemma by contradiction. For simplicity, we use $k_{t,2}$ rather than $k_{t,2}^{(2)}$. If (37) does not hold, there exists at least one $\tau \geq t_0 + 2$, such that

$$k_{\tau-1,2} \leq \frac{\beta_0 \ln(\tau-1)}{\Delta^2} + 1, \tag{38}$$

$$k_{\tau,2} > \frac{\beta_0 \ln \tau}{\Delta^2} + 1. \tag{39}$$

Since $\ln \tau > \ln(\tau - 1)$, SeV 2 is selected at time τ.

According to the utility function in (15), when $x_t = \epsilon_0$,

$$\mu_1 - \sqrt{\frac{\beta \ln(\tau - t_1)}{k_{\tau-1,1}}} \geq \mu_2 - \sqrt{\frac{\beta \ln(\tau - t_2)}{k_{\tau-1,2}}}. \tag{40}$$

Thus $\Delta = \frac{\mu_2 - \mu_1}{\mu_2} < \frac{1}{\mu_2}\sqrt{\frac{\beta \ln(\tau - t_2)}{k_{\tau-1,2}}} \leq \sqrt{\frac{\beta_0 \ln \tau}{k_{\tau-1,2}}}$, and $k_{\tau-1,2} < \frac{\beta_0 \ln \tau}{\Delta^2}$. Then $k_{\tau,2} \leq k_{\tau-1,2} + 1 < \frac{\beta_0 \ln \tau}{\Delta^2} + 1$.

Similar proof can be carried out when $x_t = 1 - \epsilon_1$. Thus we prove Lemma 3. ■

Then we prove that the proposed ALTO algorithm can explore sufficiently, such that when the input data size is large, it always selects the optimal SeV 1.

Lemma 4: With periodic input of tasks and fixed bit offloading delay of SeVs, there exists T_1, such that $a_t = 1$ when $t \geq T_1$ and $x_t = 1 - \epsilon_1$.

Proof: First, define an auxiliary function

$$h(t) = \frac{\beta_0 \ln(2t - t_2)}{\Delta^2}\left(1 + \sqrt{\frac{2\beta_0 \ln 2t}{\Delta^2(2t - 1 - t_0)}}\right)^{-2}, \tag{41}$$

and $f(t) = \int_{t_0}^{t} \min(h'(s), 1)\mathrm{d}s + h(t_0)$. We prove that $k_{2t,2} \geq f(t)$. It is easy to see that $k_{2t,2} \geq f(t)$ holds when $t = t_0$ and $t_0 + 1$. Assume that there exists $\tau \geq t_0 + 2$, such that $k_{2(\tau-1),2} \geq f(\tau - 1)$, but $k_{2\tau,2} < f(\tau)$. Since $f(\tau) - f(\tau - 1) = \int_{\tau-1}^{\tau} \min(h'(s), 1)\mathrm{d}s \leq 1$, and $k_{2(\tau-1),2}, k_{2\tau-1,2}, k_{2\tau,2}$ are integers, we have $k_{2(\tau-1),2} = k_{2\tau-1,2} = k_{2\tau,2}$. Thus SeV 1 is selected at time 2τ.

When $t = 2\tau$, $x_t = \epsilon_0$. According to the utility function in (15), we have

$$\mu_1 - \sqrt{\frac{\beta \ln(2\tau - t_1)}{k_{2\tau-1,1}}} \leq \mu_2 - \sqrt{\frac{\beta \ln(2\tau - t_2)}{k_{2\tau-1,2}}}. \tag{42}$$

Thus

$$\Delta = \frac{\mu_2 - \mu_1}{\mu_2} \geq \sqrt{\frac{\beta_0 \ln(2\tau - t_2)}{k_{2\tau-1,2}}} - \sqrt{\frac{\beta_0 \ln(2\tau - t_1)}{k_{2\tau-1,1}}}. \tag{43}$$

When τ is sufficiently large, $k_{2\tau-1,1} \geq (2\tau - 1 - t_0)/2$. Then

$$\Delta = \frac{\mu_2 - \mu_1}{\mu_2} \geq \sqrt{\frac{\beta_0 \ln(2\tau - t_2)}{k_{2\tau-1,2}}} - \sqrt{\frac{2\beta_0 \ln(2\tau - t_1)}{2\tau - 1 - t_0}}. \tag{44}$$

And thus $k_{2\tau,2} = k_{2\tau-1,2} \geq h(\tau) \geq f(\tau)$, which contradicts the assumption.

Therefore, $k_{2t,2} \geq f(t)$ holds for any $t \geq t_0$.

When $x_t = 1 - \epsilon_1$, t is odd. Let $t = 2\tau + 1$, the utility function of SeV 2 is

$$\begin{aligned}\hat{u}_{t,2} &= \bar{u}_{t-1,2} - \sqrt{\frac{\beta\epsilon_1 \ln(2\tau + 1 - t_2)}{(1 - \epsilon_0)k_{2\tau,2}}} \\ &\geq \mu_2 - \sqrt{\frac{\beta\epsilon_1 \ln(2\tau + 1 - t_2)}{(1 - \epsilon_0)f(\tau)}}\end{aligned} \tag{45}$$

Note that $\frac{1-\epsilon_0}{\epsilon_1} > 1$. There exists T_1, such that when $t \geq T_1$, $\frac{\ln(2\tau+1-t_2)}{f(\tau)} < \frac{\Delta^2}{\beta_0}\frac{1-\epsilon_0}{\epsilon_1}$. Therefore,

$$\begin{aligned}\hat{u}_{t,2} &\geq \mu_2 - \sqrt{\frac{\beta\epsilon_1 \ln(2\tau + 1 - t_2)}{(1 - \epsilon_0)f(\tau)}} \\ &> \mu_2 - \sqrt{\frac{\beta\epsilon_1}{(1 - \epsilon_0)}\frac{\Delta^2}{\beta_0}\frac{1 - \epsilon_0}{\epsilon_1}} \\ &= \mu_2 - \mu_2\Delta = \mu_1 > \hat{u}_{t,1},\end{aligned} \tag{46}$$

which indicates that SeV 1 is selected. Thus Lemma 4 is proved. ■

Finally, by letting $\beta_0 = 2$ and combining Lemma 3 and Lemma 4, Theorem 3 can be derived.

REFERENCES

[1] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *Proc. IEEE Int. Conf. Commun.*, Kansas City, MO, USA, May 2018, pp. 1–7.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," ETSI, Sophia Antipolis, France, White Paper No. 11, 2015.

[3] Y. Y. Shih, W. H. Chung, A. C. Pang, T. C. Chiu, and H. Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Netw.*, vol. 31, no. 1, pp. 52–58, Feb. 2017.

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

[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1628–1656, Third Quarter 2017.

[6] W. Yu *et al.*, "A survey on the edge computing for the Internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[7] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2016.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[9] A. L. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 45–57, Jan.–Mar. 2018.

[10] Intel, "Self-driving car technology and computing requirements, 2014. " [Online] Available: https://www.intel.com/content/www/us/en/automotive/driving-safety-advanced-driver-assistance-systems-self-driving-technology-paper.html

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

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

[13] J. S. Choo, M. Kim, S. Pack, and G. Dan, "The software-defined vehicular cloud: A new level of sharing the road," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 78–88, Jun. 2017.

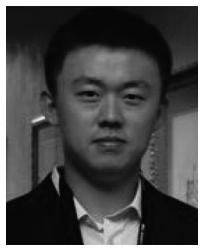
[14] X. Cheng, C. Wang, B. Ai, and H. Aggoune, "Envelope level crossing rate and average fade duration of nonisotropic vehicle-to-vehicle Ricean fading channels." *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 62–72, Feb. 2014.
[15] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002.
[16] L. Chen, S. Iellamo, and M. Coupechoux, "Opportunistic spectrum access with channel switching cost for cognitive radio networks," in *Proc. IEEE Int. Conf. Commun.*, Kyoto, Japan, Jun. 2011, pp. 1–5.
[17] C. Shen, C. Tekin, and M. van der Schaar, "A non-stochastic learning approach to energy efficient mobility management," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3854–3868, Dec. 2016.
[18] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2637–2646, Nov. 2017.
[19] H. Wu, X. Guo, and X. Liu, "Adaptive exploration-exploitation tradeoff for opportunistic bandits," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 5306–5314.
[20] *Study on Enhancement of 3GPP Support for 5G V2X Services*, 3GPP TR 22.886, V15.1.0, Mar. 2017.
[21] S. Zhang, J. Chen, F. Lyu, N. Cheng, W. Shi, and X. Shen, "Vehicular communication networks in automated driving era," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 26–32, Sep. 2018.
[22] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 146–152, Jun. 2017.
[23] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.
[24] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen, "An SMDP-based resource allocation in vehicular cloud computing systems," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7920–7928, Dec. 2015.
[25] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis and implications on road traffic," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 93–107, Feb. 2018.
[26] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE*, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.
[27] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O.Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[28] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, San Francisco, CA, USA, Jun. 2010, pp. 2141–2148.
[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, pp. 84–98, 2013.
[30] A. Salomon, J. Y. Audibert, and I. E. Alaoui, "Regret lower bounds and extended upper confidence bounds policies in stochastic multi-armed bandit problem," Dec. 2011. [Online]. Available: https://arxiv.org/abs/1112.3827
[31] S. Bubeck, "Bandits games and clustering foundations," Ph.D. dissertation, Universite des Sciences et Technologie de Lille-Lille I, Villeneuve-d'Ascq, France, 2010.
[32] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, Dec. 2012.
[33] M. Abdulla, E. Steinmetz, and H. Wymeersch, "Vehicle-to-vehicle communications with urban intersection path loss models," in *Proc. IEEE Global Commun. Conf.*, Washington, DC, USA, Dec. 2016, pp. 1–6.
[34] C. Sommer, S. Joerer, and F. Dressler, "On the applicability of two-ray path loss models for vehicular network simulation," in *Proc. IEEE Veh. Netw. Conf.*, Seoul, South Korea, Nov. 2012, pp. 64–69.

Yuxuan Sun received the B.S. degree in telecommunications engineering from Tianjin University, Tianjin, China, in 2015. She is currently working toward the Ph.D. degree in electronic engineering with Tsinghua University, Beijing, China. Her research interests include mobile edge computing and vehicular cloud computing.

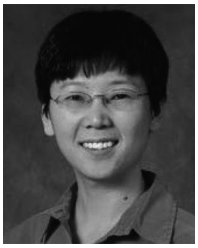
Xueying Guo (S'14–M'17) received the B.E. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2011 and 2017, respectively. From October 2013 to October 2014, she was a Visiting Scholar at the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. She is currently a Postdoctoral Researcher with the Department of Computer Science, University of California, Davis, CA, USA. Her research interests include machine learning, reinforcement learning, and data-driven networking. She was the recipient of the Best Student Paper Award in the 25th International Teletraffic Congress (ITC) in 2013.

Jinhui Song received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2018. He is currently working toward the M.S. degree in electrical and computer engineering with the University of Illinois at Urbana-Champaign, Champaign, IL, USA, where he is currently a Research Assistant. His research interests include mobile edge computing and resource allocation in networking system.

Sheng Zhou received the B.E. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively. From January to June 2010, he was a Visiting Student at the Wireless System Lab, Department of Electrical Engineering, Stanford University, Stanford, CA, USA. From November 2014 to January 2015, he was a Visiting Researcher with the Central Research Lab, Hitachi Ltd., Tokyo, Japan. He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include cross-layer design for multiple antenna systems, mobile edge computing, vehicular networks, and green wireless communications.



Zhiyuan Jiang received the B.E. and Ph.D. degrees from Tsinghua University, Beijing, China, in 2010 and 2015, respectively. He is currently an Associate Professor with the School of Communication and Information Engineering, Shanghai University, Shanghai, China. He visited University of Southern California during 2013–2014 and 2017–2018. He was an Experienced Researcher and a Wireless Signal Processing Scientist with the Ericsson and Intel Labs in 2015–2016 and 2018, respectively. His main research interests include sequential decision making in wireless networks and the design, implementation of multi-antenna systems.



Xin Liu received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 2002. She is currently a Professor in computer science with the University of California, Davis (UC Davis). Before joining UC Davis, she was a Postdoctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. During 2012–2014, she took a leave of absence and was with Microsoft Research Asia. Her research interests fall in the general areas of communication networks and machine learning, including cellular networks, cognitive radio networks, heterogeneous mesh networks, network information theory, network security, and wireless sensor networks. She currently focuses on data-driven approach in networking, including applications in 5G, IoT, data security, and edge computing. She is also interested in machine learning applications in healthcare and veterinarian medicine.

Prof. Liu was a recipient of the Computer Networks Journal Best Paper of Year Award in 2003 for her work on opportunistic scheduling, the NSF CAREER award for her research on Smart-Radio-Technology-Enabled Opportunistic Spectrum Utilization, in 2005, and the Outstanding Engineering Junior Faculty Award from the College of Engineering, University of California, Davis in 2005. She became a Chancellor's Fellow in 2011.

Zhisheng Niu received the graduate degree from Beijing Jiaotong University, Beijing, China, in 1985, and the M.E. and D.E. degrees from the Toyohashi University of Technology, Toyohashi, Japan, in 1989 and 1992, respectively. During 1992–1994, he worked for Fujitsu Laboratories Ltd., Japan, and in 1994 joined with Tsinghua University, Beijing, China, where he is currently a Professor with the Department of Electronic Engineering. His major research interests include queueing theory, traffic engineering, mobile Internet, radio resource management of wireless networks, and green communication and networks. He was the Chair of Emerging Technologies Committee (2014–2015), Director for Conference Publications (2010–2011), and the Director for Asia–Pacific Board (2008–2009) in IEEE Communication Society, and currently the Director for Online Contents (2018–2019) and a Area Editor for the Transactions on Green Communications and Networking. He was the recipient of the Outstanding Young Researcher Award from Natural Science Foundation of China in 2009 and the Best Paper Award from IEEE Communication Society Asia–Pacific Board in 2013. He was also selected as a Distinguished Lecturer of the IEEE Communication Society (2012–2015) as well as the IEEE Vehicular Technologies Society (2014–2018). He is a fellow of IEICE.