Task Replication for Vehicular Edge Computing: A Combinatorial Multi-Armed Bandit based Approach

Yuxuan Sun*, Jinhui Song*, Sheng Zhou*, Xueying Guo†, Zhisheng Niu*

*Beijing National Research Center for Information Science and Technology
Department of Electronic Engineering, Tsinghua University, Beijing, China
Email: {sunyx15, sjh14}@mails.tsinghua.edu.cn, {sheng.zhou, niuzhs}@tsinghua.edu.cn
†Department of Computer Science, University of California, Davis, CA, USA
Email: guoxueying@outlook.com

Abstract—In a vehicular edge computing (VEC) system, some vehicles with surplus computing resources can provide computation task offloading opportunities for other vehicles or pedestrians. However, the vehicular network is highly dynamic, with fast varying channel states and computation loads. These dynamics are difficult to model or to predict, but they have a major impact on the quality of service (QoS) of task offloading, including delay performance and service reliability. Meanwhile, the computing resources in VEC are often redundant due to the high density of vehicles. To improve the QoS of VEC and exploit the abundant computing resources on vehicles, we propose a learning-based task replication algorithm (LTRA) based on combinatorial multi-armed bandit (CMAB) theory, in order to minimize the average offloading delay. LTRA enables multiple vehicles to process the replicas of the same task simultaneously, and vehicles that require computing services can learn the delay performance of other vehicles while offloading tasks. We take the occurrence time of vehicles into consideration, and redesign the utility function of existing CMAB algorithm, so that LTRA can adapt to the time varying network topology of VEC. We use a realistic highway scenario to evaluate the delay performance and service reliability of LTRA through simulations, and show that compared with single task offloading, LTRA can improve the task completion ratio with deadline 0.6s from 80% to 98%.

I. INTRODUCTION

To support autonomous driving and varieties of on-board infotainment services, future vehicles will possess strong computing capabilities. It is predicted that each vehicle needs computing power of about 10^6 dhrystone million instructions executed per second (DMIPS) to enable self-driving [1]. To deliver safety messages or disseminate infotainment contents, vehicles also need to communicate with other vehicles or infrastructures through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication protocols [2], such as dedicated short-range communication (DSRC) [3] protocol and LTE-V [4]. Consequently, vehicles will be connected with each other and abundant in computing resources in the future.

To improve the utilization of vehicular computing resources, the concept of vehicle-as-an-infrastructure has been proposed [5], where vehicles can contribute their surplus computing resources to the network, forming Vehicular Edge Computing (VEC) systems [6]–[8]. VEC has huge potential to enhance edge intelligence, and can enable a variety of emerging applications that require intensive computing. Typical use cases include safety-related cooperative collision avoidance

and collective environment perception for autonomous driving [9], [10], vehicular crowdsensing for road monitoring and parking navigation [11], and entertainments such as virtual reality, augmented reality and cloud gaming for passengers [12].

In the VEC system, computation tasks are generated by on-board driving systems, passengers or pedestrians, and can possibly be executed by vehicles through *task offloading*. In this context, vehicles that provide cloud execution are called service vehicles (SeVs), while vehicles that require task offloading are called task vehicles (TaVs). In the literature, a semi-Markov decision process based formulation for centralized task offloading is given in [13], in order to minimize the average utility related to delay and energy cost. However, centralized task scheduling requires frequently collecting the complete state information of vehicles, and the proposed algorithm is highly complex to run. An alternative way is to offload tasks in a distributed manner, i.e., each TaV makes task offloading decisions individually [14]. In this case, TaV may not be able to obtain the global state information of channel states and computation loads of all available SeVs, which can be learned while offloading tasks based on multi-armed bandit (MAB) theory, as shown in our previous work [15].

Compared with mobile edge computing (MEC) [12], in which computing resources are deployed at static base stations, VEC has two major differences. On the one hand, vehicles move fast, making the network topology and wireless channels vary rapidly over time. On the other hand, the density of SeVs is much higher than static edge clouds, and thus the computing resources of VEC are more redundant than MEC.

To further improve the delay performance and service reliability in VEC system, while exploiting the redundancy of computing resources, *task replication* is a promising method, in which task replicas are offloaded to multiple SeVs at the same time and executed independently. Once one of these SeVs transmits back the result, the task is completed. The basic idea of task replication is to exchange the redundancy of computing resources for QoS improvement. A centralized task replication algorithm is proposed in [16], in order to maximize the probability of completing a task before a given deadline. However, the optimal task assignment policy is derived under the assumption that the arrival of SeV follows Poisson process,

which may not be the realistic vehicle mobility model.

In this paper, we propose a learning-based task replication algorithm (LTRA) based on combinatorial multi-armed bandit (CMAB) theory [17]. To be specific, we first propose a distributed task replication framework, in order to minimize the average offloading delay. Based on CMAB theory, we then design LTRA to deal with the challenge that TaV lacks global state information of channel states and computation loads of candidate SeVs, and characterize the upper bound of its learning regret. We simulate the traffic in a realistic highway scenario via traffic simulator Simulation for Urban MObility (SUMO), and compare LTRA with our previously proposed single offloading algorithm in [15]. Results show that both the average offloading delay and service reliability can be improved substantially through task replication.

The rest of this paper is organized as follows. In Section II, we present the system model and problem formulation. The task replication algorithm is then proposed in Section III, followed by the performance analysis in Section IV. Simulation results are shown in Section V. And finally, we conclude the work in Section VI.

II. System Model and Problem Formulation

A. System Overview

In the VEC system, moving vehicles are classified into two categories according to their roles in task offloading: TaVs and SeVs. TaVs are the vehicles that generate computation tasks that require cloud execution, while SeVs can share their surplus computing resources and execute these computation tasks. Note that each vehicle may be either a TaV or a SeV, and its role can change over time, depending on whether it has surplus computing resources to share.

Each TaV first discovers the SeVs within its communication range for task offloading. In order to maintain a relatively long contact duration, each TaV only selects its neighboring SeVs with the same moving direction as candidates. Such information can be acquired from V2V communication protocols such as beaconing messages in DSRC [3]. Moreover, we do not make any assumptions on the mobility model of vehicles.

We adopt *task replication* technique to improve the reliability, and thus the delay of task offloading. Besides, we consider distributed offloading in this work: each TaV makes the task offloading decision on the SeVs which should be selected to serve each task independently, without inter-vehicle coordinations. An exemplary task replication in VEC system is shown in Fig. 1, where TaV 1 finds SeVs 1-3 as candidates, and decides to offload the current task replicas to SeV 1 and SeV 3.

B. Task Offloading Procedure

Since tasks are offloaded in a distributed manner, we then focus on a single TaV of interest and design the corresponding task offloading algorithm. Consider a discrete-time system with a total number of T time periods. The candidate SeV set at time period t is denoted as $\mathcal{N}_t$, which may change over time. Assume that the density of SeV is high enough such that
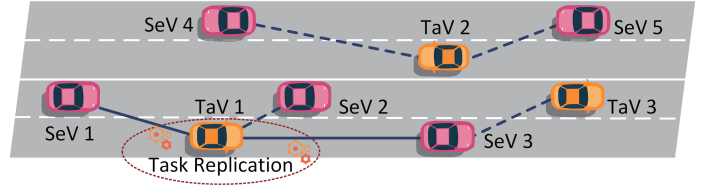

Fig. 1. Task replication in VEC system.

$\mathcal{N}_t \neq \emptyset$ for $\forall t$, otherwise the TaV may seek help from cloud servers at RSUs or in the Internet. The computation task that requires to be offloaded at time period t is modeled by three parameters (x_t, y_t, w_t) according to [12]: x_t is the input data size (in bits) to be transmitted from TaV to SeV, y_t is the output data size (in bits) which is the size of the computation result transmitted back from SeV to TaV, and w_t is the computation intensity (in CPU cycles per bit) representing how many CPU cycles are required to process one bit input data. The total required CPU cycles to execute the task is then given by $x_t w_t$.

There are three offloading steps:

Task upload: During time period t, the TaV selects a subset $\mathcal{S}_t$ of candidate SeVs with $\mathcal{S}_t \subset \mathcal{N}_t$, and offloads the task replicas to them simultaneously. We assume that the number of selected SeVs is fixed as K, i.e., $|\mathcal{S}_t| = K \leq |\mathcal{N}_t|$. For each SeV $n \in \mathcal{N}_t$, denote the uplink wireless channel state as $h_{t,n}^{(u)}$, and the interference power $I_{t,n}^{(u)}$. The channel bandwidth and transmission power are fixed as W and P, and the noise power is σ^2. Then the uplink transmission rate $r_{t,n}^{(u)}$ between the TaV and SeV n can be written as

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

Thus the delay of uploading the task from TaV to SeV n is

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

Task execution: After receiving the input data from TaV, each SeV $n \in \mathcal{S}_t$ executes the task independently. Denote the maximum CPU frequency of SeV n as C_n (in CPU cycles per second). Each SeV may process multiple computation tasks at the same time, either from its own user equipments or other TaVs, and the allocated CPU frequency for the TaV of interest is $f_{t,n} \in [0, C_n]$. Then the computation delay of SeV n in time period t is

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

Result feedback: The computation result is finally transmitted back from each selected SeV $n \in \mathcal{S}_t$ to the TaV. Similar to (1), the downlink transmission rate between SeV n and TaV in time period t is given by

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

where $h_{t,n}^{(d)}$ and $I_t^{(d)}$ are the downlink channel state and the interference at the TaV respectively. Therefore the downlink delay from SeV n to TaV is

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

As a result, the offloading delay $d(t,n)$ of SeV n is the sum of uplink and downlink transmission delay and the computation delay, written as

$$d(t,n) = d_t^{(u)}(t,n) + d_c(t,n) + d_t^{(d)}(t,n). \tag{6}$$

The actual offloading delay of each task that the TaV experiences only depends on

$$\min_{n \in \mathcal{S}_t} d(t,n). \tag{7}$$

However, we still require *all* the other SeVs to finish execution and transmit the result, in order to record the offloading delay for the learning purposes, which will be introduced in detail in Section III.

C. Problem Formulation

Given a total number of T time periods, our objective is to minimize the average offloading delay of tasks, by deciding the subset of SeVs which should be selected to serve each task. The problem is formulated as:

$$\textbf{P1:} \quad \min_{\mathcal{S}_1,...,\mathcal{S}_T} \frac{1}{T} \sum_{t=1}^{T} \min_{n \in \mathcal{S}_t} d(t,n). \tag{8}$$

The delay performance of each SeV mainly depends on three variables: uplink transmission rate $r_{t,n}^{(u)}$, allocated CPU frequency $f_{t,n}$, and downlink transmission rate $r_{t,n}^{(d)}$. If these variables are known to the TaV before it offloads each task, the TaV can then calculate the exact offloading delay $d(t,n)$ of each SeV n, and select single SeV $\arg\min_{n \in \mathcal{N}_t} d(t,n)$. However, due to the movements of vehicles, the transmission rates $r_{t,n}^{(u)}$ and $r_{t,n}^{(d)}$ are fast varying and hard to predict. Meanwhile, the allocated CPU frequency $f_{t,n}$ is not easy to know in prior due to the varying computation loads of SeVs. Thus TaV may lack the exact global state information, and can not distinguish which SeV provides the fastest computation for each task.

Our solution is *learning while offloading*: we let the TaV learn the delay performance of candidate SeVs through delay observations while offloading tasks. To be specific, till time period t, the TaV gets delay records $\{d(1,n), n \in \mathcal{S}_1\}$, ..., $\{d(t-1,n), n \in \mathcal{S}_{t-1}\}$, estimates the delay performance at the current time period, and selects a subset $\mathcal{S}_t = \arg\min_{\mathcal{S} \in \mathcal{N}_t} \min_{n \in \mathcal{S}} d(t,n)$ to offload the task replicas.

III. LEARNING WHILE OFFLOADING

In this section, we design the learning-based task replication algorithm, which guides the TaV to learn the delay performance of candidate SeVs while offloading tasks, in order to minimize the average offloading delay. We consider a simplified scenario by assuming that tasks are of equal input,

output data size $x_t = x_0$, $y_t = y_0$ and computation intensity $w_t = w_0$ for $\forall t$. In fact, tasks often have similar ratio of input to output data size and computation intensity if they are generated by the same kind of applications. And tasks with diverse input data size can be partitioned into several subtasks of the same input data size and offloaded in sequence, e.g., a long video frame for object detection or classification can be divided into short video clips through video segmentation [18].

The task replication is an online sequential decision making problem, which has been investigated under the MAB framework. In classical MAB problem [19], there are a fixed number of base arms with unknown loss distributions. In each time period, a decision maker tries a candidate base arm, observes its loss, and updates the estimates of its loss distribution. The objective is to minimize the cumulative loss over time. The classical MAB problem has been further extended to the CMAB problem [17], where in each time period the decision maker can try a subset of base arms (defined as a super arm), observe the loss of all the base arms composing this super arm, and minimize the cumulative loss of the system.

Our problem is similar to the CMAB problem with non-linear loss function: each candidate SeV corresponds to a base arm with unknown delay distribution, while the TaV is the decision maker who selects a subset $\mathcal{S}_t$ of SeVs in each time period t. The TaV can observe the delay $d(t,n)$ (loss) of all selected SeVs, and the system loss, i.e., the offloading delay, is the minimum of the observed delay $\min_{n \in \mathcal{S}_t} d(t,n)$, which is a non-linear function of each delay.

The major difference between our task replication problem and the existing CMAB problem is that, the candidate SeV set $\mathcal{N}_t$ may change over time since vehicles are moving, and it is difficult to predict when SeVs may appear or disappear, and how long they can act as candidates. How to efficiently learn the delay performance of candidate SeVs under such a dynamic environment has not been investigated in the existing work of CMAB.

We thus take into consideration the time varying feature of candidate SeVs, and revise the existing CMAB algorithm in [20] into learning-based task replication algorithm (LTRA), as shown in Algorithm 1. Let $\tilde{d}(t,n) = \frac{d(t,n)}{d_{max}}$ be the normalized delay, where $d(t,n), \forall n \in \mathcal{S}_t$ is the delay observed by TaV, and d_{max} is the maximum delay allowed of each task offloading. If in time period t, the computation result from SeV n is not successfully received by the TaV till d_{max}, we regard that the task is failed by SeV n, and set the observed delay $d(t,n) = d_{max}$ for learning purpose. And thus $\tilde{d}(t,n) \in [0,1]$. Denote $\hat{D}_n$ as the empirical distribution of the normalized delay $\tilde{d}(t,n)$ of SeV n, and $\hat{F}_n$ the cumulative distribution function (CDF) of $\hat{D}_n$. Let $k_{t,n}$ be the number of tasks offloaded to SeV n so far, β a constant factor, and t_n the occurrence time of SeV n.

In Algorithm 1, Lines 2-4 are the initialization phase, during which the TaV selects a subset of K SeVs that contains at least one newly appeared SeV. Note that the initialization phase not only happens at the beginning of task offloading, but whenever

Algorithm 1 Learning-based Task Replication Algorithm
1: **for** $t = 1, ..., T$ **do**
2:     **if** Any SeV $n \in \mathcal{N}_t$ has not connected to TaV **then**
3:         Connect to any subset $\mathcal{S}_t \in \mathcal{N}_t$ once, with $n \in \mathcal{S}_t$.
4:         Update empirical CDF $\hat{F}_n$ of normalized delay $\tilde{d}(t, n)$ and selected times $k_{t,n}$ for each $n \in \mathcal{S}_t$.
5:     **else**
6:         For each $n \in \mathcal{N}_t$, define CDF $\underline{G}_n$ as

$$\underline{G}_n(x) = \begin{cases} 0 & x = 0, \\ \min\left\{\hat{F}_n(x) + \sqrt{\frac{\beta \ln(t-t_n)}{k_{t-1,n}}}, 1\right\} & 0 < x \le 1 \end{cases}. \tag{9}$$

7:         Select a subset of candidate SeVs, such that

$$\mathcal{S}_t = \arg\min_{\mathcal{S} \subset \mathcal{N}_t} \mathbb{E}_{\underline{D}}\left[\min_{n \in \mathcal{S}} d(t, n)\right], \tag{10}$$

        where $\underline{D}_n$ is the distribution of CDF $\underline{G}_n$, and $\underline{D} = \underline{D}_1 \times \underline{D}_2 \times ... \times \underline{D}_{|\mathcal{N}_t|}$.
8:         Offload the task replica to all the SeV $\forall n \in \mathcal{S}_t$.
9:         Observe delay $d(t, n)$ for each $n \in \mathcal{S}_t$.
10:       Update $\hat{F}_n$ and $k_{t,n} \leftarrow k_{t-1,n} + 1$ for each $n \in \mathcal{S}_t$.
11:     **end if**
12: **end for**

new SeVs occur.

Lines 6-10 are the main learning phase. Due to the non-linearity of the offloading delay $\min_{n \in \mathcal{S}_t} d(t, n)$, the offloading decision $\mathcal{S}_t$ depends on the *entire delay distribution* of each candidate SeVs, rather than the mean of them. Thus the learning algorithm keeps updating the empirical CDF $\hat{F}_n$ to learn the entire distribution, and makes offloading decisions according to $\hat{F}_n$. In Line 6, the CDF $\underline{G}_n(x)$ defined in (9) is a numerical upper confidence bound on the real delay CDF of each SeV n, which can balance the exploration-exploitation tradeoff during the learning process: The TaV tends to *explore* SeVs with fewer selected times $k_{t,n}$ to learn good estimates of their delay distributions, while at the same time to *exploit* SeVs with better empirical delay performance to optimize the instantaneous offloading delay. The padding term $\sqrt{\frac{\beta \ln(t-t_n)}{k_{t-1,n}}}$ also considers the occurrence time t_n of each SeV n, such that the newly appeared SeVs can be well explored, while the empirical information of the existing SeVs can be exploited.

In Line 7, the TaV selects a subset of candidate SeVs that minimizes the expectation of offloading delay according to (10), where $\underline{D}_n$ is the distribution of CDF $\underline{G}_n$, and $\underline{D} = \underline{D}_1 \times \underline{D}_2 \times ... \times \underline{D}_{|\mathcal{N}_t|}$ is the joint distribution of all candidate SeVs. Calculating $\mathcal{S}_t$ is actually a minimum element problem, which can be solved by greedy algorithms [21]. Then the TaV offloads the task replicas to all the selected SeVs $n \in \mathcal{S}_t$, waits for their feedbacks to observe the delay, and finally updates the empirical CDF $\hat{F}_n$ of normalized delay $\tilde{d}(t, n)$ and selected times $k_{t,n}$.

A. Implementation Considerations

Since the offloading delay is continuous, LTRA may suffer from large storage usage and computational complexity as t grows. To be specific, the observed delay values $d(t, n)$ of each SeV n might be different in each time period, and thus the required storage for each empirical CDF $\hat{F}_n$ is $O(t)$. Meanwhile, it takes $O(t)$ time to calculate the numerical upper confidence bound $\underline{G}_n(x)$, and the minimum element problem in (10) is more complex to solve. To reduce the storage usage and computational complexity of the algorithm, we can discretize the empirical CDF $\hat{F}_n$ to be $\tilde{F}_n$, by partitioning the range $[0, 1]$ into l segments with equal interval $\frac{1}{l}$. The support of the discretized CDF $\tilde{F}_n$ is $\{\frac{1}{l}, \frac{2}{l}, ..., 1\}$, and if the normalized delay $\tilde{d}(t, n)$ belongs to $\left(\frac{j-1}{l}, \frac{j}{l}\right]$, the delay used to update $\tilde{F}_n$ is $\frac{j}{l}$.

IV. ALGORITHM PERFORMANCE

In this section, we characterize the performance of the proposed LTRA. To carry out theoretical analysis, we assume that the candidate SeV set $\mathcal{N}_t$ remains constant as $\mathcal{N}$ for the total T time periods, and the delay $d(t, n)$ of each candidate SeV n is independent from other SeVs and i.i.d across time. We will prove later through simulations in Section V that without the aforementioned two assumptions, LTRA can still work well.

Let $\boldsymbol{d}_t = (d(t, 1), ..., d(t, N))$ be the delay vector of all candidate SeVs in time period t with $N = |\mathcal{N}|$, and $L(\boldsymbol{d}_t, \mathcal{S}_t) = \min_{n \in \mathcal{S}_t} d(t, n)$ the loss function. The expected loss $\mu_{\mathcal{S}_t}$ for choosing a subset $\mathcal{S}_t$ of candidate SeVs does not change over time due to the i.i.d assumption, thus we omit the subscript t and let $\mu_{\mathcal{S}} = \mathbb{E}[L(\boldsymbol{d}, \mathcal{S})]$. Moreover, let $\mathcal{S}^* = \arg\min_{\mathcal{S} \subset \mathcal{N}} \mu_{\mathcal{S}}$ be the optimal subset of SeV, and $\mu_{\mathcal{S}^*} = \min_{\mathcal{S} \subset \mathcal{N}} \mu_{\mathcal{S}}$ its expected loss.

Define the cumulative learning regret R_T as

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} L(\boldsymbol{d}_t, \mathcal{S}_t)\right] - T\mu_{\mathcal{S}^*}, \tag{11}$$

which is the expected loss brought with learning as compared to the optimal decisions, since the TaV does not know which candidate SeV performs the best.

For any suboptimal subset $\mathcal{S}$, let the expected delay gap $\Delta_{\mathcal{S}} = (\mu_{\mathcal{S}} - \mu_{\mathcal{S}^*})/d_{max}$. Define

$$\Delta_n = \min\{\Delta_{\mathcal{S}} | \mathcal{S} \subset \mathcal{N}, n \in \mathcal{S}, \mu_{\mathcal{S}} > \mu_{\mathcal{S}^*}\}, \tag{12}$$

and let $\mathcal{N}_s$ be the set of candidate SeVs which is contained in at least one suboptimal subset.

In the following theorem, we provide an upper bound of the cumulative learning regret of LTRA.

Theorem 1. *Let $\beta = \frac{2}{3}$, then R_T is upper bounded by:*

$$R_T \le d_{max}\left(C_1 K \sum_{n \in \mathcal{N}_s} \frac{\ln T}{\Delta_n} + C_2\right), \tag{13}$$

where $C_1 = 2136$ and $C_2 = \left(\frac{\pi^2}{3} + 1\right) N$ are two constants.

Proof. See Appendix A. □

Theorem 1 shows that, our proposed LTRA can provide a delay performance with bounded regret, compared to the genie-aided case, where the delay distributions of candidate SeVs are known in prior. To be specific, the cumulative learning regret grows logarithmically with T, and is also related to the number of selected SeVs and the performance gap between different subsets of SeVs.

V. SIMULATIONS

In this section, we carry out simulations to evaluate the delay performance and service reliability of the proposed LTRA. We first use SUMO[1] to simulate the traffic, and then import the floating car data generated by SUMO into MATLAB to evaluate the performance of LTRA.

The road used for traffic simulation is a 12km segment of G6 Highway in Beijing, with two lanes and two ramps, downloaded from Open Street Map (OSM)[2]. Vehicles come from either the start of the road or the ramps, and when a vehicle meets a ramp, it leaves the highway with a probability of 0.5. The maximum speed allowed of both SeVs and TaVs is 20m/s. The arrival rate of SeV ranges from 0.05s^{-1} to 0.4s^{-1}, and the arrival rate of TaV ranges from 0.01s^{-1} to 0.2s^{-1}.

The floating car data of SUMO includes the type, ID, position, speed and angle of each vehicle, so that we can calculate the distance of each SeV and TaV in MATLAB. The communication range is set to 300m, and the wireless channel $h_{t,n}^{(u)} = h_{t,n}^{(d)} = A_0 d_{st}^{-2}$, with $A_0 = -17.8\text{dB}$ and d_{st} the distance between TaV and SeV, according to [22]. The channel bandwidth $W = 10\text{MHz}$, transmission power $P = 0.1\text{W}$, and noise power $\sigma^2 = 10^{-13}\text{W}$.

For each task, we set the input data size $x_0 = 1\text{Mbits}$, its computation intensity $\omega_0 = 1000\text{Cycles/bit}$, and the output data size is very small and omitted. The maximum CPU frequency C_n of each SeV is uniformly chosen within $[2, 8]\text{GHz}$. In each time period, the allocated CPU frequency for TaVs is randomly distributed from 0 to $60\%C_n$ (each SeV also needs to process tasks generated from its own driving system or UEs, so it can not allocate all the computing resources for TaVs). Note that each SeV may provide service for multiple TaVs at the same time, and in the simulation, tasks offloaded by TaVs are served by the first-come-first-serve queue discipline. Moreover, parameter β in (9) is 0.6, and the default number of discretization segments is 50.

We compare our proposed LTRA with 3 other algorithms: 1) **Genie-aided Policy**: assume that the TaV knows the global state information of all candidate SeVs, and always selects single SeV with minimum delay. This policy can not be realized in the realistic VEC system. 2) **Random Policy**: a naive policy, in which TaV randomly selects a SeV in each time period to offload the task. 3) **Single Offloading**: a learning-based task offloading policy proposed in our previous
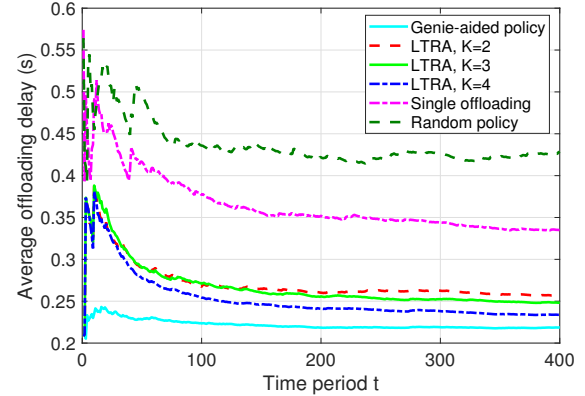
Fig. 2. Average offloading delay of LTRA.

work [15], in which each TaV only selects a *single* SeV for each task.

Fig. 2 shows the average offloading delay of LTRA with the number of task replicas $K = 2, 3, 4$. On average, there are about 8 candidate SeVs and 1 other TaV around our target TaV. It is shown that with task replication, the delay performance is improved to about 0.25s, while learning-based single offloading algorithm can only achieve 0.33s. With increasing number of task replicas, the average offloading delay is closer to the genie-aided policy.

Fig. 3 shows the delay performance and service reliability under different SeV densities. The x-coordinate, ranges from 2 to 11, is the average number of candidate SeVs around our target TaV, and still there is another 1 TaV around. In Fig. 3(a), the average offloading delay of LTRA decreases along with the increasement of candidate SeVs, since LTRA can exploit the redundant computing resources through task replication. However, more task replicas do not always bring performance improvement. When computing resources are insufficient, too many task replicas may lead to long task queues at candidate SeVs, which is not efficient. Fig. 3(b) shows the task completion ratio given deadline 0.6s. When there are more than 3 candidate SeVs, the task completion ratio of LTRA outperforms single offloading. And when there are more than 7 candidate SeVs, the service reliability of LTRA can reach over 98%, while single offloading only achieves about 80%. Therefore, with sufficient computing resources in the VEC system, task replication is a promising method to enhance the reliability of computing services .

In Fig. 4, the density of SeV is fixed, with about 8 candidates around the target TaV. As more TaVs competing the computing resources with each other, the average offloading delay of LTRA increases. To be specific, when the density ratio of TaV to SeV is below 0.3, LTRA with 4 task replicas outperforms LTRA with only 2 replicas. However, as the density ratio grows higher, fewer number of replicas achieves better delay performance. Thus the number of task replica should be carefully selected under different traffic conditions.
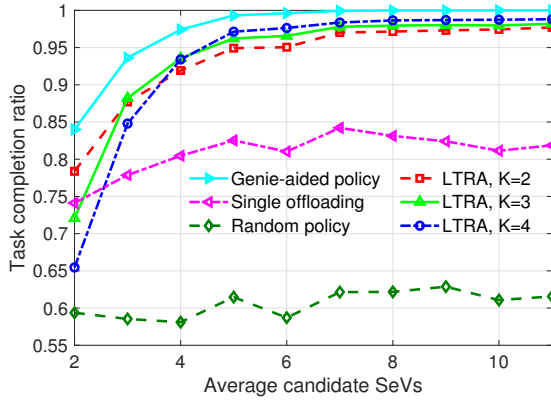
Finally, we explore the impact of discretization level l on
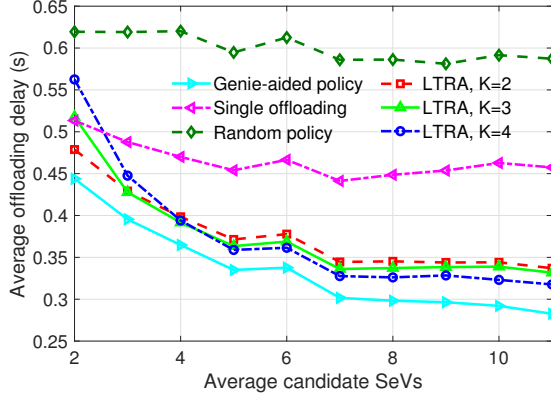
(a) Average offloading delay.

(b) Task completion ratio.

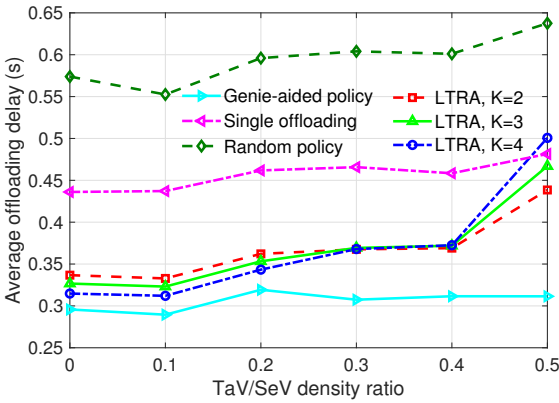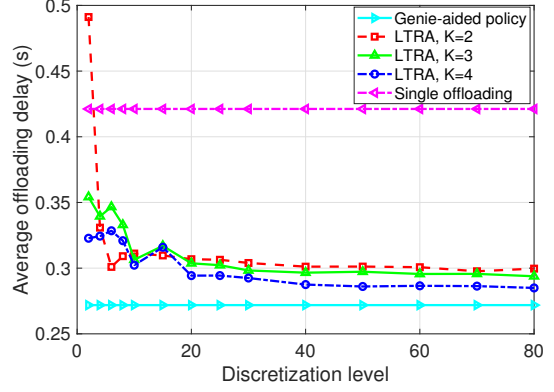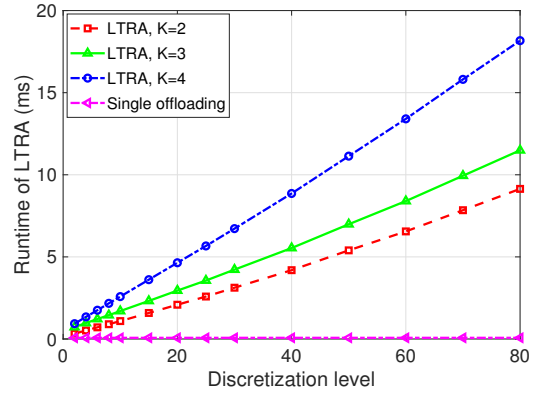Fig. 3. Performance of LTRA under different number of candidate SeVs.

Fig. 4. Average offloading delay of LTRA under different TaV and SeV density ratio.

(a) Average offloading delay.

(b) Runtime of LTRA.

Fig. 5. Impact of discretization level l.

the average offloading delay and the runtime of the algorithm. When the delay distribution is discretized into very few segments, the runtime of LTRA is low, but the average offloading delay is very poor and fluctuates severely. When the discretization level is too high, the delay performance does not improve much, but it takes more time to run LTRA. To both get good estimates of the realistic delay distributions, and save the runtime of LTRA at the same time, the discretization level should be carefully selected. For example, under our settings, the discretization level l should be about 40 to 50.

VI. CONCLUSIONS

In this work, we have investigated the task offloading problem in VEC system, and proposed LTRA by combining the task replication and sequential learning techniques, in order to minimize the average offloading delay. LTRA enables each TaV to learn the delay performance of candidate SeVs while offloading tasks, and can adapt to the highly dynamic vehicular environment. We have carried out simulations under a realistic highway scenario, and compared the delay performance and service reliability of LTRA to the existing single offloading algorithm. Simulation results have shown that the average delay of LTRA is close to the optimal genie-aided policy and better than the single offloading policy. And when there are sufficient SeVs, the performance can be highly improved through a small number of task replications. Specifically, with a given deadline 0.6s, the task completion ratio of LTRA can reach 98% with only two replicas, while single offloading can only achieve about 80%.

Acknowledgment

Appendix A
Proof of Theorem 1

We prove that, under the assumption that the number of candidate SeVs is fixed, our delay minimization problem is equivalent to the reward maximization problem of standard CMAB investigated in [20], and the proposed algorithm LTRA is equivalent to the stochastically dominant confidence bound (SDCB) algorithm proposed in [20].

First, the objective functions are equivalent, since

$$\min_{\mathcal{S}_1,...,\mathcal{S}_T} \frac{1}{T} \sum_{t=1}^{T} \min_{n \in \mathcal{S}_t} d(t,n)$$

$$= d_{max} \min_{\mathcal{S}_1,...,\mathcal{S}_T} \frac{1}{T} \sum_{t=1}^{T} \min_{n \in \mathcal{S}_t} \tilde{d}(t,n)$$

$$\Leftrightarrow \max_{\mathcal{S}_1,...,\mathcal{S}_T} \frac{1}{T} \sum_{t=1}^{T} \left[\max_{n \in \mathcal{S}_t} \left(1 - \tilde{d}(t,n) \right) \right]. \qquad (14)$$

Since $\tilde{d}(t,n) \in [0,1]$, the reward function $R(\boldsymbol{d}_t, \mathcal{S}_t) = \max_{n \in \mathcal{S}_t} \left(1 - \tilde{d}(t,n) \right) \in [0,1]$, satisfying assumption 2 in [20] with upper bound $M = 1$. Also, $R(\boldsymbol{d}_t, \mathcal{S}_t)$ is monotone, which satisfies assumption 3 in [20].

Second, the numerical upper confidence bound $\underline{G}_n(x)$ can be transformed to CDF $\underline{F}_n(x)$ defined in the SDCB algorithm in [20]. Define $\hat{F}_n(x)$ as the CDF of $\tilde{d}(t,n)$, and $\hat{F}'_n(x)$ the CDF of $1 - \tilde{d}(t,n)$. It is easy to see that $\hat{F}(x) = 1 - \hat{F}'(1-x)$. Thus

$$\underline{G}_n(x) = 1 - \underline{F}_n(1-x)$$

$$= 1 - \begin{cases} \max\left\{ \hat{F}'_n(1-x) - \sqrt{\frac{\beta \ln t}{k_{t-1,n}}}, 0 \right\} & 0 \le 1-x < 1, \\ 1 & 1-x = 1 \end{cases}$$

$$= 1 - \begin{cases} \max\left\{ 1 - \hat{F}_n(x) - \sqrt{\frac{\beta \ln t}{k_{t-1,n}}}, 0 \right\} & 0 < x \le 1, \\ 1 & x = 0 \end{cases}$$

$$= \begin{cases} 0 & x = 0, \\ \min\left\{ \hat{F}_n(x) + \sqrt{\frac{\beta \ln t}{k_{t-1,n}}}, 1 \right\} & 0 < x \le 1 \end{cases}. \qquad (15)$$

Substitute the reward upper bound $M = 1$, and let $\alpha = 1$ in Theorem 1 in [20], then (13) can be derived.

References

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

[2] S. Zhang, W. Quan, J. Li, W. Shi, P. Yang, and X. Shen, "Air-ground integrated vehicular network slicing with content pushing and caching," [Online]. Available: https://arxiv.org/abs/1806.03860.

[3] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162-1182, Jul. 2011.

[4] S. Chen, J. Hu, Y. Shi, and L. Zhao, "LTE-V: A TD-LTE-based V2X solution for future vehicular network," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 997-1005, Dec. 2016.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, pp. 3860-3873, Jun. 2016.

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

[7] 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.

[8] 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.

[9] 3GPP, "Study on enhancement of 3GPP support for 5G V2X services," 3GPP TR 22.886, V15.1.0, Mar. 2017,

[10] S. Zhang, J. Chen, F. Lyu, N. Cheng, W. Shi, and X. Shen, "Vehicular communication networks in automated driving era," [Online]. Available: https://arxiv.org/abs/1805.09583.

[11] 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.

[12] 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, Fourthquarter 2017.

[13] 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.

[14] 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.

[15] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," *IEEE Int. Conf. Commun. (ICC)*, Kansas city, MO, USA, May 2018.

[16] 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.

[17] W. Chen, Y. Wang, and Y. Yuan. "Combinatorial multi-armed bandit: General framework and applications," *Int. Conf. on Machine Learning (ICML)*, Atlanta, GA, USA, Jun. 2013.

[18] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, San Francisco, CA, USA, Jun. 2010.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[20] W. Chen, W. Hu, F. Li, J. Li, Y. Liu, and P. Lu, "Combinatorial multi-armed bandit with general reward functions,"*Advances in Neural Information Processing Systems*, vol. 29, 2016.

[21] A. Goel, S. Guha, and K. Munagala, "How to probe for an extreme value,"*ACM Trans. on Algorithms*, vol. 7, no. 1, Nov. 2010.

[22] M. Abdulla, E. Steinmetz and H. Wymeersch, "Vehicle-to-vehicle communications with urban intersection path loss models," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016.