Online Learning Enabled Task Offloading for Vehicular Edge Computing

Rui Zhang, Peng Cheng, Zhuo Chen, Sige Liu, Yonghui Li, *Fellow, IEEE*, and Branka Vucetic, *Life Fellow, IEEE*

***Abstract*—Vehicular edge computing pushes the cloud computing capability to the distributed network edge nodes, enabling computation-intensive and latency-sensitive computing services for smart vehicles through task offloading. However, the inherent mobility introduces fast variation of network structure, which are usually unknown *a priori*. In this letter, we formulate the vehicular task offloading as a mortal multi-armed bandit problem, and develop a new online algorithm to enable distributed decision making on the node selection. The key is to exploit the contextual information of edge nodes and transform the infinite exploration space to a finite one. Theoretically, we prove that the proposed algorithm has a sublinear learning regret. Simulation results verify its effectiveness.**

***Index Terms*—Online learning, task offloading, vehicle edge computing.**

I. Introduction

THE BOOMING vehicle-to-everything technology is accelerating the development of the Internet of Vehicles and catalyzing emerging in-vehicle computation-intensive applications, such as self-driving and augmented reality-supported games [1]. Although a smart vehicle accessing the remote cloud with abundant computation resources is a viable option, the inherent high transmission latency is a practical limitation. To address this issue, vehicular edge computing (VEC) [2] was proposed as an attractive solution, where neighbouring vehicles with excess computation resources can serve as distributed edge nodes. Offloading a computationally tedious task to these physically closer edge nodes can largely reduce transmission latency, resulting in a substantial improvement in user quality of experience.

A variety of vehicular task offloading algorithms have been proposed in the literature, and most work focuses on centralized designs. That is, a central controller is deployed to frequently collect vehicle state information such as mobility, available resources, requested tasks, and identities [3]–[7], and then schedule task offloading for each vehicle. Centralized task offloading was formulated as a semi-Markov decision process in [3], aiming to minimize the weighted sum of system delay and energy cost. In [4], the tradeoff between the delay and power consumption is further investigated based on matching theory.

Decentralized task offloading is an enabling technology eliminating the need for a central controller. Under this umbrella, each vehicle can independently make the decision on neighboring edge node selection, and then perform task offloading in a distributed manner [8], [9]. However, a vehicle is unlikely to have prior knowledge of the state information of neighbouring edge nodes. Furthermore, the time-varying network structure, where neighboring edge nodes randomly appear and vanish due to their inherent mobility, will bring a significant challenge in the development of distributed task offloading algorithms.

The essence of the distributed task offloading problem is to enable a vehicle to directly interact with the edge nodes through task offloading, learn their state information, and map decision history to the current offloading decision. This can be formulated as a multi-armed bandit (MAB) problem when the set of edge nodes remains unchanged. Specifically, each neighboring edge node is treated as an independent arm, and its associated bit cost is dominated by the computing capability drawn from an unknown distribution. Therefore, "playing" an arm at each round is equivalent to selecting an edge node to perform task offloading. This requires trading off the exploitation (select the current best node with past knowledge) and exploration (try other nodes to gain more accurate information). However, the MAB herein is mortal as the set of candidate arms evolves with time due to their mobility, which makes classical solutions such as the UCB1 algorithm [10] no longer effective.

To address this mortal MAB problem, in this letter we propose a novel online learning (OL) algorithm. The OL algorithm leverages the contextual information that the computing capabilities of all candidate arms (edge nodes) come from a finite number of distributions. Given two arms with the same distribution of computing capabilities, our algorithm transfers the learning result of the first arm into the second one. In this case, the original infinite exploration space can be transformed to a finite one, regardless of arms appearing or vanishing. Furthermore, in this scenario we improve the calculation of conventional confidence interval in the UCB1 algorithm. Theoretically, we prove that the proposed OL algorithm has a sublinear regret (the gap to an oracle benchmark). Simulation results demonstrate that the proposed algorithm outperforms the existing ones in terms of the cumulative bit cost.

Manuscript received January 16, 2020; revised February 7, 2020; accepted February 10, 2020. Date of publication February 14, 2020; date of current version July 9, 2020. This work was supported by ARC under Grant DE190100162. The associate editor coordinating the review of this article and approving it for publication was B. Makki. *(Corresponding author: Peng Cheng.)*

Rui Zhang, Peng Cheng, Sige Liu, Yonghui Li, and Branka Vucetic are with the School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2134, Australia (e-mail: rui.zhang1@sydney.edu.au; peng.cheng@sydney.edu.au; sige.liu@sydney.edu.au; yonghui.li@sydney.edu.au; branka.vucetic@sydney.edu.au).

Zhuo Chen was with CSIRO DATA61, Sydney, NSW 2122, Australia (e-mail: zhuo.chen@ieee.org).

Digital Object Identifier 10.1109/LWC.2020.2973985

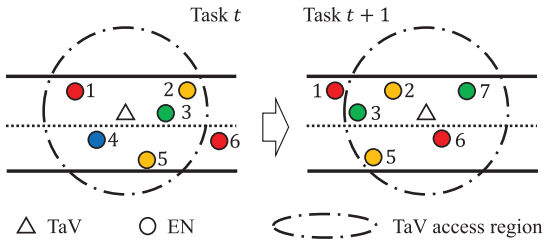


Fig. 1. A typical VEC system with a TaV and several ENs, where $\mathcal{M}^t = \{1, 2, 3, 4, 5\}$ and $\mathcal{M}^{t+1} = \{2, 3, 5, 6, 7\}$. In this example, the x_m^t of EN 1 and 6 (2 and 5, 3 and 7) follow the same distribution.

II. System Model

A. VEC System Model

Consider a typical VEC system shown in Fig. 1. The vehicles involved in vehicle-to-vehicle task offloading include a task vehicle (TaV) indexed by 0, and a set of edge notes (ENs), indexed by $\mathcal{M} = \{1, 2, \ldots\}$. The TaV generates tasks, while ENs provide edge computational resources. The operational timeline is discretized based on task unit. Given task t with a moderate size,[1] the TaV can either execute it locally or offload it to an EN in $\mathcal{M}^t \subseteq \mathcal{M}$ within the TaV access region. Due to inherent mobility, $\mathcal{M}^t$ varies with tasks (see Fig. 1). The TaV interacts with accessible ENs continuously and updates $\mathcal{M}^t$ on a real-time basis. For convenience, we denote $\mathcal{N}^t = \mathcal{M}^t \cup \{0\}$ to cover the TaV and corresponding ENs simultaneously.

Local execution and performing task offloading incur transmission/computation latency and energy consumption. For local execution, the system latency and energy consumption of task t can be given by $D_0^t = c^t/u_0^t$ and $E_0^t = P_0 c^t/u_0^t$, respectively, where c^t is the required central processing unit (CPU) cycles for task t, $u_0^t = u_0$ is the CPU frequency of the TaV, and $P_0 = \rho u_0^2$ is the computing power of the TaV with the effective switched capacitance $\rho = 10^{-11}$ depending on the chip architecture [12].

On the other hand, for task offloading to EN $m \in \mathcal{M}^t$, the system latency of task t originates from task transmission and EN execution, which is given by

$$D_m^t = \frac{c^t}{u_m^t} + \frac{s^t}{r_m^t}, \tag{1}$$

where s^t is the data size of task t, and u_m^t is the CPU frequency of EN m allocated to the TaV, which is usually unknown *a priori*. The achievable uplink transmission rate is given by

$$r_m^t = W \log_2\left(1 + \frac{P_s \alpha_m^t \|h_m^t\|^2}{N_0}\right), \tag{2}$$

where W is the TaV transmit bandwidth, P_s is the TaV transmission power, $\alpha_m^t = 128.1 + 37.6 \log_{10}(d_m^t)$(dB) is the large-scale fading gain following the 3GPP path loss model [13], and d_m^t is the transmission distance available to the TaV using LIDAR and radar [14]. h_m^t is the small-scale fading gain which follows Rayleigh distribution with unit variance, and N_0 is the noise power. Given the orthogonal channel allocation [15], the co-channel interference can be avoided. Furthermore, the cross-channel interference can be ignored according to the experimental results in [16]. In this letter, we assume that the TaV and ENs are moving in the same direction. Due to the small relative speed, the Doppler shift is not significant and α_m^t and h_m^t remain the same during the uplink transmission for each task. We also assume that the data size of task results is small so that the download latency is omitted [17]. The system energy consumption of completing task t is

$$E_m^t = \frac{P_m c^t}{u_m^t} + \frac{P_s s^t}{r_m^t}, \tag{3}$$

where $P_m = \rho (u_m^t)^2$ is the EN computing power.

To take into account both the system latency and energy consumption, we define the cost function as the weighted sum over the energy consumption and the delay, which is given by

$$f(t, m) = \eta_1 D_m^t + \eta_2 E_m^t, \qquad m \in \mathcal{N}^t, \tag{4}$$

where η_1 and η_2 denote the scalar weights of system latency and energy consumption, respectively. Similar cost functions are also widely adopted in the existing literature [7], [18], [19].

B. Problem Formulation

Without loss of generality, we assume that s^t and c^t vary with tasks, but the computation intensity $\lambda = c^t/s^t$ remains constant. This usually corresponds to the case that tasks are generated by the same type of application [20]. In this letter, we further define the bit cost of task t as

$$f_0(t, m) = \frac{f(t, m)}{s^t} = \eta_1 D_m^{0,t} + \eta_2 E_m^{0,t}, \tag{5}$$

where $D_m^{0,t} = D_m^t/s^t$ and $E_m^{0,t} = E_m^t/s^t$ denote the bit latency and bit energy consumption, respectively. From another aspect, the system cost comes from the computation and communication parts, thus (5) can be also written as

$$f_0(t, m) = x_m^t + y_m^t, \tag{6}$$

where

$$x_m^t = \frac{\lambda \eta_1}{u_m^t} + \lambda \eta_2 \rho \left(u_m^t\right)^2, \quad y_m^t = \begin{cases} 0, & m = 0, \\ \dfrac{\eta_1 + \eta_2 P_s}{r_m^t}, & m \neq 0. \end{cases} \tag{7}$$

Note that x_m^t and y_m^t correspond to the bit cost of computation and communication, respectively.

Next, we elaborate on additional contextual information of the CPU frequency u_m^t. Assume that u_m^t is a draw from an unknown distribution [8]. If the ENs have the same type of CPU, their u_m^t follow the same distribution and so do their $x_m^t \sim G_i$, $i \in \{1, \ldots, S\}$, and S is the total number of distributions. For convenience, we introduce a set of binary indicator variables $z_{m,i} \in \{0, 1\}$, where $z_{m,i} = 1$ if x_m^t follows G_i, and $z_{m,i} = 0$ otherwise. Denote the expectation of x_m^t by $\bar{x}_m = \mathbb{E}[x_m^t]$ and the mean value of G_i by μ_i, we have $\mu_i = \bar{x}_m$ when $z_{m,i} = 1$. An example is shown in Fig. 1, where we use the same color to mark the ENs whose x_m^t follow the same distribution. The final goal of the TaV is to minimize the expected bit cost up to a finite T tasks. This requires the TaV to perform decision making and choose $m \in \mathcal{N}^t$ in each round. Clearly, the past decisions will affect

[1] A task with a larger workload can be further partitioned into multiple subtasks [8], [11].

the current one, and the task offloading problem is to minimize the cumulative bit cost of all tasks

$$\mathcal{P}: \min_{m\in\mathcal{N}^t} \mathbb{E}\left[\sum_{t=1}^{T} f_0(t,m)\right]. \tag{8}$$

C. The Oracle Benchmark and the Regret

Before presenting our online learning algorithm, we first give an oracle benchmark to $\mathcal{P}$, provided that the TaV has the prior knowledge of $\bar{x}_m$ for $m \in \mathcal{N}^t$. In this case, the optimal solution can be given by

$$m^* = \arg\min_{m\in\mathcal{N}^t} \{\bar{x}_m + y_m^t\}, \tag{9}$$

where the TaV always chooses $m \in \mathcal{N}^t$ with the minimal expected bit cost (y_m^t can always be calculated). However, in practice, the TaV has no knowledge of $\bar{x}_m$, and the TaV has to learn $\bar{x}_m$ through task offloading and observing the resultant latency D_m^t. Therefore, it is necessary to develop an online learning algorithm to strike a tradeoff in EN selection, targeting an EN whose x_m^t is not well learned (exploration), or an EN which is believed to cause the minimal bit cost (exploitation). Typically, this belongs to a MAB problem [10], where each EN and the TaV can be regarded as a candidate arm (hereafter we use the concept of the arm to represent EN and TaV). Furthermore, due to the mobility, one arm may have a finite lifetime (e.g., EN 1 in Fig. 1), and a new arm may appear (e.g., EN 7). Generally, this can be regarded as a mortal MAB problem [21], where the set of candidate arms is evolving with time. Due to its challenges, the conventional solutions to the classical MAB are usually ineffective. Given the oracle benchmark, the regret of a learning algorithm with respect to this benchmark is defined as

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} f_0(t,m)\right] - \mathbb{E}\left[\sum_{t=1}^{T} f_0(t,m^*)\right]. \tag{10}$$

III. The Online Learning Algorithm

In this section, we propose a novel OL algorithm to address the mortal MAB problem, and then analyze its performance in terms of regret.

A. Algorithm Structure

In essence, our OL algorithm leverages the contextual information G_i of x_m^t ($z_{m,i} = 1$) for each arm. Given two arms whose x_m^t follow the same distribution, our algorithm transfers the learning result of the first arm into the second one. In this case, the original infinite exploration space can be transformed to a finite one, regardless of arms appearing or vanishing.

The OL algorithm in its entirety is presented in Algorithm 1, which consists of the initialization stage and two components in each round: exploration-exploitation tradeoff (EET) and counter update. In the initialization (Lines 1-7), we introduce the counters L_i and K_i for the arms that follow G_i (i.e., $z_{m,i} = 1$). L_i is the number of times that G_i has been selected up to now, while K_i is the number of times that distribution G_i has appeared. We also set $\hat{\mu}_i = 0$, which is the average value of x_m^t ($z_{m,i} = 1$) observed until now. For the arm related to the TaV ($z_{0,i} = 1$), we set $L_i = 1$, $K_i = 1$, and $\hat{\mu}_i = \lambda\eta_1/u_0 + \lambda\eta_2\rho u_0$.

Depending on L_i, the EET component (Lines 9-17) is divided into two cases. For the first case (Lines 9-11), there exists arm $m \in \mathcal{M}^t$ that satisfies $z_{m,i} = 1$ and $L_i = 0$, and the EET component is reduced to pure exploration. Considering there might be multiple arms satisfying the above requirement, the arm with the minimal bit communication cost y_m^t will be chosen, i.e.,

$$m' = \arg\min_{m\in\mathcal{M}^t, z_{m,i}=1} y_m^t. \tag{11}$$

After selecting arm m', we can measure the latency $D_{m'}^t$. On this basis, $x_{m'}^t$ can be calculated as

$$x_{m'}^t = \frac{\eta_1\left(D_{m'}^t r_{m'}^t - s_{m'}^t\right)}{s_{m'}^t r_{m'}^t} + \frac{\eta_2\rho\lambda\left(c_{m'}^t r_{m'}^t\right)^2}{\left(D_{m'}^t r_{m'}^t - s_{m'}^t\right)^2}, \tag{12}$$

and set $\hat{\mu}_i = x_{m'}^t$. For the second case (Lines 12-17), we have $L_i > 0$ for all $i \in \{1,\ldots,S\}$. We design a utility function for arm *m* as

$$U_m^t = \sum_{i=1}^{S}\left[z_{m,i}(\hat{\mu}_i - C_i)\right] + y_m^t, \tag{13}$$

where $C_i = \sqrt{2\beta \ln K_i / L_i}$ is the confidence interval with a normalization parameter $\beta = (\lambda/u_0)^2$ and the arm with the minimal U_m^t is chosen, i.e.,

$$m' = \arg\min_{m\in\mathcal{N}^t} U_m^t. \tag{14}$$

Then, $\hat{\mu}_i$ is updated based on the latency $D_{m'}^t$.

Finally, in the counter update (Lines 18-23), the counters K_i and L_i are updated accordingly. As the only computation before each arm selection is (11) or (14), the algorithm can be implemented in real time.

B. Analysis of the Regret

In this section, we analyze the regret upper bound of the OL algorithm. Without loss of generality, we assume that $\mu_i < \mu_j$ for $1 \le i < j \le S$. We also define

$$\Delta_{i,j}^{+} = \mu_j + \nu_j^{+} - \left(\mu_i + \nu_i^{-}\right), \tag{15}$$

$$\Delta_{i,j}^{-} = \mu_j + \nu_j^{-} - \left(\mu_i + \nu_i^{+}\right), \tag{16}$$

where

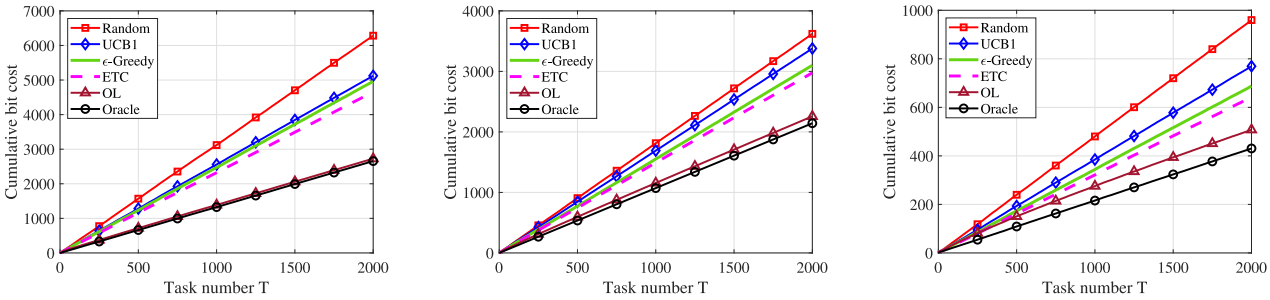
$$\nu_i^{-} = \min_{m\in\mathcal{M}, z_{m,i}=1} y_m^t, \quad \nu_i^{+} = \max_{m\in\mathcal{M}, z_{m,i}=1} y_m^t. \tag{17}$$

If there are arm m_1 with $z_{m_1,i} = 1$, and m_2 with $z_{m_2,j} = 1$, the arm m_1 should be chosen due to $\mu_i < \mu_j$. The regret naturally arises if the arm m_2 is chosen instead, and we denote the number of times it happens by $N_{i,j}$. Here, we can prove that the following lemma.

Lemma 1: For a total number of *T* tasks, $N_{i,j}$ is bounded as

$$\mathbb{E}\left[N_{i,j}\right] \le \left(1 - z_{0,j}\right)\left[\frac{4\beta \ln T}{\left(\Delta_{i,j}^{-}\right)^2} + \mathcal{O}(1)\right]. \tag{18}$$

Proof: See Appendix A. ∎



(a) $\eta_1 = 0$/ms and $\eta_2 = 1$/J. (b) $\eta_1 = 0.5$/ms and $\eta_2 = 0.5$/J. (c) $\eta_1 = 1$/ms and $\eta_2 = 0$/J.

Fig. 2. The cumulative bit cost versus task number T.

Algorithm 1 The OL Algorithm

for $i = 1, \ldots, S$ **do** ▷ Initialization
  if $z_{0,i} = 0$ **then**
    Set $L_i = 0$, $K_i = 0$, and $\hat{\mu}_i = 0$.
  else
    Set $L_i = 1$, $K_i = 1$, and $\hat{\mu}_i = \lambda\eta_1/u_0 + \lambda\eta_2\rho u_0$.
  end if
end for
for $t = 1, \ldots, T$ **do**
  if $\exists m \in \mathcal{M}^t$, s.t. $z_{m,i} = 1$ and $L_i = 0$ **then** ▷ EET
    Choose arm m' given in (11) and get latency $D^t_{m'}$.
    Set $\hat{\mu}_i = x^t_{m'}$ which is given in (12).
  else
    Choose arm m' given in (14) and get latency $D^t_{m'}$.
    for $i = 1, \ldots, S$ **do**
      Update $\hat{\mu}_i \leftarrow \dfrac{\hat{\mu}_i L_i + x^t_{m'} z_{m',i}}{L_i + z_{m',i}}$.
    end for
  end if
  Update $L_i \leftarrow L_i + 1$. ▷ Counter update
  for $i = 1, \ldots, S$ **do**
    if $\exists m \in \mathcal{M}^t$, s.t. $z_{m,i} = 1$ **then**
      Update $K_i \leftarrow K_i + 1$.
    end if
  end for
end for

Lemma 1 indicates that $N_{i,j}$ increases logarithmically with T. On this basis, we can arrive at the following theorem.

Theorem 1: For a total number of T tasks, $R(T)$ can be bounded as

$$R(T) \le \mathbb{E}\left[\sum_{j=2}^{S}\sum_{i=1}^{j-1}\left(N_{i,j} - N_{i-1,j}\right)\Delta^+_{i,j}\right] \le [4\beta \ln T + \mathcal{O}(1)] \sum_{j=2, z_{0,j}=0}^{S}\sum_{i=1}^{j-1}\frac{\Delta^+_{i,j} - \Delta^+_{i+1,j}}{\left(\Delta^-_{i,j}\right)^2}. \quad (19)$$

Proof: See Appendix B. ■

Theorem 1 indicates that the regret $R(T)$ can be well constrained with a larger $\Delta^-_{i,j}$, and it is sublinear in the task horizon T, i.e., $R(T) = \mathcal{O}(T^\gamma)$ with $\gamma < 1$ [20]. This regret bound guarantees that the OL algorithm has asymptotically optimal performance, since $\lim_{T\to\infty} R(T)/T = 0$ holds. This means that the OL algorithm converges to the oracle benchmark.

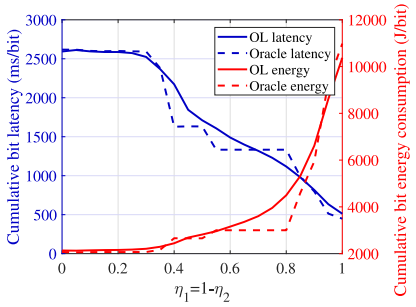


Fig. 3. The cumulative bit latency and energy consumption versus η_1 with $T = 2000$.

IV. Simulation Results

In this section, we present numerical results to demonstrate the superiority of the OL algorithm. In the simulation, the distance d^t_m is draw from a uniform distribution $\mathcal{U}(0, 0.2)$ in kilometers, and the number of ENs in the region $M^t \ge 5$. We set $c^t \in [1, 5] \times 10^8$ CPU cycles, $s^t \in [1, 5] \times 10^4$ bits, $\lambda = 10^4$ cycles/bit, $N_0 = -174$ dBm/Hz, $P_s = 24$ dBm, $W = 0.8$ MHz, $S = 11$, $u_0 = 1.5 \times 10^7$ Hz, and $\mathbb{E}[u^t_m] \in \{0.5, 1, 1.5, 2, 2.5, 3, 4, 5, 6, 7\} \times 10^8$ Hz ($m \ne 0$).

We compare the cumulative bit cost of the OL algorithm with the oracle and several other algorithms in Fig. 2 with different values of η_1 and η_2. The random algorithm randomly chooses arm $m \in \mathcal{N}^t$ (TaV or EN). The ϵ-Greedy algorithm chooses arm $m \in \mathcal{N}^t$ with the minimal estimated bit cost based on current knowledge, but attempts to select other arms with a probability 0.1. The explore-then-commit (ETC) algorithm explores each arm for a certain number of times (we set 1 in our simulation), and then stick to the one with the minimal bit cost [20]. The UCB1 algorithm chooses arms based on their expected cost and upper confidence indexes [10]. It is clearly shown in Fig. 2 that the OL algorithm outperforms all the algorithms except the oracle one in all three scenarios. In particular, the OL algorithm performs very close to the oracle one, and the gap between them increases when η_1 increases.

Fig. 3 shows the impact of weighting parameters η_1 and η_2 on the cumulative bit latency and energy consumption (see (5)) of the OL algorithm and the oracle one with $T = 2000$. In particular, we set $\eta_1 + \eta_2 = 1$. We find that with the increase of the latency weighting parameter η_1, both algorithms care more about latency performance, and their cumulative bit latency decreases. We also find that the cumulative bit latency of the OL algorithm decreases with η_1 smoothly, while that of the oracle algorithm keeps stable when $0 \le \eta_1 \le 0.3$,

$0.4 \leq \eta_1 \leq 0.5$, and $0.55 \leq \eta_1 \leq 0.8$. This shows that when η_1 is located within these ranges, the bit cost is dominated by x_m^t (y_m^t has a minor impact) and the oracle algorithm always chooses the arms (best ones) whose x_m^t follow the same distribution. Instead, the OL algorithm without a perfect knowledge of $\bar{x}_m$ may try other arms, but it keeps close to the oracle one. A Similar conclusion can be drawn for η_2 and cumulative bit energy consumption.

V. Conclusion

In this letter, we studied the distributed task offloading in a vehicular edge computing system. We formulated the offloading as a mortal multi-armed bandit problem, which is quite challenging due to the mortal feature where the set of candidate arms is evolving with time. To address this problem, we proposed a novel OL algorithm by incorporating contextual information into the candidate arm. We proved that the proposed algorithm has a sublinear learning regret. Simulation results shown its advantages over existing algorithms.

Appendix A
Proof of Lemma 1

Proof: For $1 \leq i < j \leq S$ where $z_{0,j} = 0$, let $Q_{i,j} = 4\beta \ln T/(\Delta_{i,j}^-)^2$, then $N_{i,j}$ can be bounded as

$$\mathbb{E}[N_{i,j}] \leq Q_{i,j} + \sum_{t=Q_{i,j}}^{T} \mathbb{1}\{\forall_{k=1}^{i}(\hat{\mu}_j^t - C_j^t \leq \hat{\mu}_k^t - C_k^t)\}. \quad (20)$$

where $\hat{\mu}_i^t$ and C_i^t denote the value of $\hat{\mu}_i$ and C_i on round t, respectively. Define an event $\mathcal{E}$ as $\hat{\mu}_i^t \in [\mu_i - C_i^t, \mu_i + C_k^t]$ for $k \in \{1, \ldots, i, j\}$. Since $L_j^t > Q_{i,j}$ and $K_j^t < T$, when $\mathcal{E}$ happens, we have

$$\hat{\mu}_j^t - C_j^t \geq \mu_j - 2C_j^t > \mu_k \geq \hat{\mu}_k^t - C_k^t. \quad (21)$$

Therefore, $\hat{\mu}_j^t - C_j^t > \hat{\mu}_k^t - C_k^t$ when $\mathcal{E}$ happens. With the help of Hoeffding's inequality, we can rewrite (20) as

$$\begin{aligned}
&\mathbb{E}\big[N_{i,j}\big] \\
&\leq Q_{i,j} + \sum_{t=Q_{i,j}}^{T} \left(\mathbb{1}\left\{\forall_{k=1}^{i}\left(\hat{\mu}_j^t - C_j^t \leq \hat{\mu}_k^t - C_k^t\right)\right\}, \neg\mathcal{E}\right) \\
&\leq Q_{i,j} + \sum_{t=Q_{i,j}}^{T} \neg\mathcal{E} \leq Q_{i,j} + \sum_{t=1}^{\infty} \frac{i+1}{t^4} = Q_{i,j} + \mathcal{O}(1). \quad (22)
\end{aligned}$$

For $1 \leq i < j \leq S$ where $z_{0,j} = 1$, we have $\hat{\mu}_j^t \geq \hat{\mu}_i^t - C_i^t$. Thus, $\mathbb{E}[N_{i,j}] = 0$. Therefore, we finish the proof. ■

Appendix B
Proof of Theorem 1

Proof: Based on Lemma 1, we have

$$\begin{aligned}
R(T) &\leq \mathbb{E}\left[\sum_{j=2}^{S}\sum_{i=1}^{j-1}\big(N_{i,j} - N_{i-1,j}\big)\Delta_{i,j}^+\right] \\
&= \sum_{j=2}^{S}\sum_{i=1}^{j-1}\mathbb{E}\big[N_{i,j}\big]\left(\Delta_{i,j}^+ - \Delta_{i+1,j}^+\right) \\
&= [4\beta \ln T + \mathcal{O}(1)] \sum_{j=2, z_{0,j}=0}^{S}\sum_{i=1}^{j-1}\frac{\Delta_{i,j}^+ - \Delta_{i+1,j}^+}{\left(\Delta_{i,j}^-\right)^2}, \quad (23)
\end{aligned}$$

where $N_{0,j} = 0$. ■

References

[1] H. Hartenstein and K. Laberteaux, *VANET: Vehicular Applications and Inter-Networking Technologies*, vol. 1. Chichester, U.K.: Wiley, 2010.

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

[3] 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.

[4] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A distributed and context-aware task assignment mechanism for collaborative mobile edge computing," *Sensors*, vol. 18, no. 8, p. E2423, 2018.

[5] L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Communicating or computing over the MAC: Function-centric wireless networks," *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6127–6138, Sep. 2019.

[6] M. Qin, L. Chen, N. Zhao, Y. Chen, F. R. Yu, and G. Wei, "Power-constrained edge computing with maximum processing capacity for IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4330–4343, Jun. 2019.

[7] M. Qin, L. Chen, N. Zhao, Y. Chen, R. Yu, and G. Wei, "Computing and relaying: Utilizing mobile edge computing for P2P communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1582–1594, Feb. 2020.

[8] Y. Sun *et al.*, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[9] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2019.

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

[11] 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.

[12] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[13] "Evolved universal terrestrial radio access, version 9.0.0," 3GPP, Sophia Antipolis, France, Rep. TR 36.931, May 2011.

[14] C. Ilas, "Electronic sensing technologies for autonomous ground vehicles: A review," in *Proc. 8th Int. Symp. Adv. Topics Elect. Eng. (ATEE)*, May 2013, pp. 1–6.

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

[16] V. Rai, F. Bai, J. Kenney, and K. Laberteaux, "Cross-channel interference test results: A report from the VSC-A project," document 802.11 07-2133–00-000p, IEEE, Piscataway, NJ, USA, Jul. 2007. [Online]. Available: https://mentor.ieee.org/802.11/dcn/07/11–07-2133–00-000p-crosschannel-interference-test-results-a-reportfrom-the-vsc-a-project.ppt

[17] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1910–1920, Aug. 2017.

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

[19] S. Khalili and O. Simeone, "Inter-layer per-mobile optimization of cloud mobile computing: A message-passing approach," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 6, pp. 814–827, 2016.

[20] L. Chen, J. Xu, S. Ren, and P. Zhou, "Spatio–temporal edge service placement: A bandit learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 12, pp. 8388–8401, Dec. 2018.

[21] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal, "Mortal multi-armed bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 273–280.